Package ‘SimMultiCorrData’

June 28, 2018

Type Package

Title Simulation of Correlated Data with Multiple Variable Types

Version 0.2.2

Author Allison Cynthia Fialkowski

Maintainer Allison Cynthia Fialkowski <allijazz@uab.edu>

Description
Generate continuous (normal or non-normal), binary, ordinal, and count (Poisson or Negative Binomial) variables with a specified correlation matrix. It can also produce a single continuous variable. This package can be used to simulate data sets that mimic real-world situations (i.e. clinical or genetic data sets, plasmodes). All variables are generated from standard normal variables with an imposed intermediate correlation matrix. Continuous variables are simulated by specifying mean, variance, skewness, standardized kurtosis, and fifth and sixth standardized cumulants using either Fleishman's third-order (<DOI:10.1007/BF02293811>) or Headrick's fifth-order (<DOI:10.1016/S0167-9473(02)00072-5>) polynomial transformation. Binary and ordinal variables are simulated using a modification of the ordsample() function from 'GenOrd'. Count variables are simulated using the inverse cdf method. There are two simulation pathways which differ primarily according to the calculation of the intermediate correlation matrix. In Correlation Method 1, the intercorrelations involving count variables are determined using a simulation based, logarithmic correlation correction (adapting Yahav and Shmueli's 2012 method, <DOI:10.1002/asmb.901>). In Correlation Method 2, the count variables are treated as ordinal (adapting Barbiero and Ferrari's 2015 modification of GenOrd, <DOI:10.1002/asmb.2072>). There is an optional error loop that corrects the final correlation matrix to be within a user-specified precision value of the target matrix. The package also includes functions to calculate standardized cumulants for theoretical distributions or from real data sets, check if a target correlation matrix is within the possible correlation bounds (given the distributions of the simulated variables), summarize results (numerically or graphically), to verify valid power method pdfs, and to calculate lower standardized kurtosis bounds.

Depends R (>= 3.3.0)

License GPL-2

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

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Suggests knitr, rmarkdown, printr, testthat

VignetteBuilder knitr

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

NeedsCompilation no

Repository CRAN

Date/Publication 2018-06-28 17:37:55 UTC

R topics documented:

`calc_final_corr` *Calculate Final Correlation Matrix*

Description

This function calculates the final correlation matrix based on simulated variable type (ordinal, continuous, Poisson, and/or Negative Binomial). The function is used in `rcorrvar` and `rcorrvar2`. This would not ordinarily be called directly by the user.

Usage

```
calc_final_corr(k_cat, k_cont, k_pois, k_nb, Y_cat, Yb, Y_pois, Y_nb)
```

Arguments

`k_cat`	the number of ordinal (r >= 2 categories) variables
`k_cont`	the number of continuous variables
`k_pois`	the number of Poisson variables
`k_nb`	the number of Negative Binomial variables
`Y_cat`	the ordinal (r >= 2 categories) variables
`Yb`	the continuous variables
`Y_pois`	the Poisson variables
`Y_nb`	the Negative Binomial variables

Value

a correlation matrix

See Also

`rcorrvar`, `rcorrvar2`

---

calc_fisherk *Find Standardized Cumulants of Data based on Fisher's k-statistics*

Description

This function uses Fisher's k-statistics to calculate the mean, standard deviation, skewness, standardized kurtosis, and standardized fifth and sixth cumulants given a vector of data. The result can be used as input to `find_constants` or for data simulation.

Usage

```
calc_fisherk(x)
```

Arguments

`x`	a vector of data

Value

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

References

Fisher RA (1928). Moments and Product Moments of Sampling Distributions. Proc. London Math. Soc. 30, 199-238. doi: 10.1112/plms/s230.1.199.

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

`calc_theory`, `calc_moments`, `find_constants`

Examples

```
x <- rgamma(n = 10000, 10, 10)
calc_fisherk(x)
```

calc_lower_skurt *Find Lower Boundary of Standardized Kurtosis for Polynomial Transformation*

Description

This function calculates the lower boundary of standardized kurtosis for Fleishman's Third-Order (method = "Fleishman", doi: 10.1007/BF02293811) or Headrick's Fifth-Order (method = "Polynomial", doi: 10.1016/S01679473(02)000725), given values of skewness and standardized fifth and sixth cumulants. It uses nleqslv to search for solutions to the multi-constraint Lagrangean expression in either fleish_skurt_check or poly_skurt_check. When Headrick's method is used (method = "Polynomial"), if no solutions converge and a vector of sixth cumulant correction values (Six) is provided, the smallest value is found that yields solutions. Otherwise, the function stops with an error.

Each set of constants is checked for a positive correlation with the underlying normal variable (using power_norm_corr) and a valid power method pdf (using pdf_check). If the correlation is <= 0, the signs of c1 and c3 are reversed (for method = "Fleishman"), or c1, c3, and c5 (for method = "Polynomial"). It will return a kurtosis value with constants that yield in invalid pdf if no other solutions can be found (valid.pdf = "FALSE"). If a vector of kurtosis correction values (Skurt) is provided, the function finds the smallest value that produces a kurtosis with constants that yield a valid pdf. If valid pdf constants still can not be found, the original invalid pdf constants (calculated without a correction) will be provided. If no solutions can be found, an error is given and the function stops. Please note that this function can take considerable computation time, depending on the number of starting values (n) and lengths of kurtosis (Skurt) and sixth cumulant (Six) correction vectors. Different seeds should be tested to see if a lower boundary can be found.

Usage

```
calc_lower_skurt(method = c("Fleishman", "Polynomial"), skews = NULL,
  fifths = NULL, sixths = NULL, Skurt = NULL, Six = NULL,
  xstart = NULL, seed = 104, n = 50)
```

Arguments

method	the method used to find the constants. "Fleishman" uses a third-order polynomial transformation and requires only a skewness input. "Polynomial" uses Headrick's fifth-order transformation and requires skewness plus standardized fifth and sixth cumulants.
skews	the skewness value
fifths	the standardized fifth cumulant (if method = "Fleishman", keep NULL)
sixths	the standardized sixth cumulant (if method = "Fleishman", keep NULL)
Skurt	a vector of correction values to add to the lower kurtosis boundary if the constants yield an invalid pdf, ex: Skurt = seq(0.1, 10, by = 0.1)
Six	a vector of correction values to add to the sixth cumulant if no solutions converged, ex: Six = seq(0.05, 2, by = 0.05)

xstart initial value for root-solving algorithm (see nleqslv). If user specified, must be input as a matrix. If NULL generates n sets of random starting values from uniform distributions.

seed the seed value for random starting value generation (default = 104)

n the number of initial starting values to use (default = 50). More starting values require more calculation time.

Value

A list with components:

Min a data.frame containing the skewness, fifth and sixth standardized cumulants (if method = "Polynomial"), constants, a valid.pdf column indicating whether or not the constants generate a valid power method pdf, and the minimum value of standardized kurtosis ("skurtosis")

C a data.frame of valid power method pdf solutions, containing the skewness, fifth and sixth standardized cumulants (if method = "Polynomial"), constants, a valid.pdf column indicating TRUE, and all values of standardized kurtosis ("skurtosis"). If the Lagrangean equations yielded valid pdf solutions, this will also include the lambda values, and for method = "Fleishman", the Hessian determinant and a minimum column indicating TRUE if the solutions give a minimum kurtosis. If the Lagrangean equations yielded invalid pdf solutions, this data.frame contains constants calculated from find_constants using the kurtosis correction.

Invalid.C if the Lagrangean equations yielded invalid pdf solutions, a data.frame containing the skewness, fifth and sixth standardized cumulants (if method = "Polynomial"), constants, lambda values, a valid.pdf column indicating FALSE, and all values of standardized kurtosis ("skurtosis"). If method = "Fleishman", also the Hessian determinant and a minimum column indicating TRUE if the solutions give a minimum kurtosis.

Time the total calculation time in minutes

start a matrix of starting values used in root-solver

SixCorr1 if Six is specified, the sixth cumulant correction required to achieve converged solutions

SkurtCorr1 if Skurt is specified, the kurtosis correction required to achieve a valid power method pdf (or the maximum value attempted if no valid pdf solutions could be found)

Notes on Fleishman Method

The Fleishman method *can not generate valid power method distributions* with a ratio of $skew^2/skurtosis > 9/14$, where skurtosis is kurtosis - 3. This prevents the method from being used for any of the *Chi-squared distributions*, which have a constant ratio of $skew^2/skurtosis = 2/3$.

Symmetric Distributions: All symmetric distributions (which have skew = 0) possess the same lower kurtosis boundary. This is solved for using optimize and the equations in Headrick & Sawilowsky (2002, doi: 10.3102/10769986025004417). The result will always be: c0 = 0, c1 = 1.341159, c2 = 0, c3 = -0.1314796, and minimum standardized kurtosis = -1.151323. Note that this set of constants does NOT generate a valid power method pdf. If a Skurt vector of kurtosis correction values is provided, the function will find the smallest addition that yields a valid pdf. This value is 1.16, giving a lower kurtosis boundary of 0.008676821.

Asymmetric Distributions: Due to the square roots involved in the calculation of the lower kurtosis boundary (see Headrick & Sawilowsky, 2002), this function uses the absolute value of the skewness. If the true skewness is less than zero, the signs on the constants c0 and c2 are switched after calculations (which changes skewness from positive to negative without affecting kurtosis).

Verification of Minimum Kurtosis: Since differentiability is a local property, it is possible to obtain a local, instead of a global, minimum. For the Fleishman method, Headrick & Sawilowsky (2002) explain that since the equation for kurtosis is not "quasiconvex on the domain consisting only of the nonnegative orthant," second-order conditions must be verified. The solutions for lambda, c1, and c3 generate a global kurtosis minimum if and only if the determinant of a bordered Hessian is less than zero. Therefore, this function first obtains the solutions to the Lagrangean expression in `fleish_skurt_check` for a given skewness value. These are used to calculate the standardized kurtosis, the constants c1 and c3, and the Hessian determinant (using `fleish_Hessian`). If this determinant is less than zero, the kurtosis is indicated as a minimum. The constants c0, c1, c2, and c3 are checked to see if they yield a continuous variable with a positive correlation with the generating standard normal variable (using `power_norm_corr`). If not, the signs of c1 and c3 are switched. The final set of constants is checked to see if they generate a valid power method pdf (using `pdf_check`). If a `Skurt` vector of kurtosis correction values is provided, the function will find the smallest value that yields a valid pdf.

Notes on Headrick's Method

The *sixth cumulant correction vector* (`Six`) may be used in order to aid in obtaining solutions which converge. The calculation methods are the same for symmetric or asymmetric distributions, and for positive or negative skew.

Verification of Minimum Kurtosis: For the fifth-order approximation, Headrick (2002, doi: 10.1016/S01679473(02)000725) states "it is assumed that the hypersurface of the objective function [for the kurtosis equation] has the appropriate (quasiconvex) configuration." This assumption alleviates the need to check second-order conditions. Headrick discusses steps he took to verify the kurtosis solution was in fact a minimum, including: 1) substituting the constant solutions back into the 1st four Lagrangean constraints to ensure the results are zero, 2) substituting the skewness, kurtosis solution, and standardized fifth and sixth cumulants back into the fifth-order equations to ensure the same constants are produced (i.e. using `find_constants`), and 3) searching for values below the kurtosis solution that solve the Lagrangean equation. This function ensures steps 1 and 2 by the nature of the root-solving algorithm of `nleqslv`. Using a sufficiently large n (and, if necessary, executing the function for different seeds) makes step 3 unnecessary.

Reasons for Function Errors

The most likely cause for function errors is that no solutions to `fleish_skurt_check` or `poly_skurt_check` converged. If this happens, the simulation will stop. Possible solutions include: a) increasing the number of initial starting values (`n`), b) using a different seed, or c) specifying a `Six` vector of sixth cumulant correction values (for `method` = "Polynomial"). 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)`). Due to the nature of the integration involved in `calc_theory`, the results are approximations. Greater accuracy can be achieved by increasing the number of subdivisions (`sub`) used in the integration process. For example, in order to ensure that skew is exactly 0 for symmetric distributions.

References

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

Hasselman B (2018). nleqslv: Solve Systems of Nonlinear Equations. R package version 3.3.2. https://CRAN.R-project.org/package=nleqslv

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.

See Also

nleqslv, fleish_skurt_check, fleish_Hessian, poly_skurt_check, power_norm_corr, pdf_check, find_constants

Examples

```
# Normal distribution with Fleishman transformation
calc_lower_skurt("Fleishman", 0, 0, 0)

## Not run:

# This example takes considerable computation time.

# Reproduce Headrick's Table 2 (2002, p.698): note the seed here is 104.
# If you use seed = 1234, you get higher Headrick kurtosis values for V7 and V9.
# This shows the importance of trying different seeds.

options(scipen = 999)

V1 <- c(0, 0, 28.5)
V2 <- c(0.24, -1, 11)
V3 <- c(0.48, -2, 6.25)
V4 <- c(0.72, -2.5, 2.5)
```

```
V5 <- c(0.96, -2.25, -0.25)
V6 <- c(1.20, -1.20, -3.08)
V7 <- c(1.44, 0.40, 6)
V8 <- c(1.68, 2.38, 6)
V9 <- c(1.92, 11, 195)
V10 <- c(2.16, 10, 37)
V11 <- c(2.40, 15, 200)

G <- as.data.frame(rbind(V1, V2, V3, V4, V5, V6, V7, V8, V9, V10, V11))
colnames(G) <- c("g1", "g3", "g4")

# kurtosis correction vector (used in case of invalid power method pdf constants)
Skurt <- seq(0.01, 2, 0.01)

# sixth cumulant correction vector (used in case of no converged solutions for
# method = "Polynomial")
Six <- seq(0.1, 10, 0.1)

# Fleishman's Third-order transformation
F_lower <- list()
for (i in 1:nrow(G)) {
  F_lower[[i]] <- calc_lower_skurt("Fleishman", G[i, 1], Skurt = Skurt,
                                   seed = 104)
}

# Headrick's Fifth-order transformation
H_lower <- list()
for (i in 1:nrow(G)) {
  H_lower[[i]] <- calc_lower_skurt("Polynomial", G[i, 1], G[i, 2], G[i, 3],
                                   Skurt = Skurt, Six = Six, seed = 104)
}

# Approximate boundary from PoisBinOrdNonNor
PBON_lower <- G$g1^2 - 2

# Compare results:
# Note: 1) the lower Headrick kurtosis boundary for V4 is slightly lower than the
#          value found by Headrick (-0.480129), and
#       2) the approximate lower kurtosis boundaries used in PoisBinOrdNonNor are
#          much lower than the actual Fleishman boundaries, indicating that the
#          guideline is not accurate.
Lower <- matrix(1, nrow = nrow(G), ncol = 12)
colnames(Lower) <- c("skew", "fifth", "sixth", "H_valid.skurt",
                     "F_valid.skurt", "H_invalid.skurt", "F_invalid.skurt",
                     "PBON_skurt", "H_skurt_corr", "F_skurt_corr",
                     "H_time", "F_time")

for (i in 1:nrow(G)) {
  Lower[i, 1:3] <- as.numeric(G[i, 1:3])
  Lower[i, 4] <- ifelse(H_lower[[i]]$Min[1, "valid.pdf"] == "TRUE",
                        H_lower[[i]]$Min[1, "skurtosis"], NA)
  Lower[i, 5] <- ifelse(F_lower[[i]]$Min[1, "valid.pdf"] == "TRUE",
                        F_lower[[i]]$Min[1, "skurtosis"], NA)
```

```
  Lower[i, 6] <- min(H_lower[[i]]$Invalid.C[, "skurtosis"])
  Lower[i, 7] <- min(F_lower[[i]]$Invalid.C[, "skurtosis"])
  Lower[i, 8:12] <- c(PBON_lower[i], H_lower[[i]]$SkurtCorr1,
                      F_lower[[i]]$SkurtCorr1,
                      H_lower[[i]]$Time, F_lower[[i]]$Time)
}
Lower <- as.data.frame(Lower)
print(Lower[, 1:8], digits = 4)

#   skew fifth  sixth H_valid.skurt F_valid.skurt H_invalid.skurt F_invalid.skurt PBON_skurt
# 1  0.00  0.00  28.50      -1.0551     0.008677         -1.3851         -1.1513    -2.0000
# 2  0.24 -1.00  11.00      -0.8600     0.096715         -1.2100         -1.0533    -1.9424
# 3  0.48 -2.00   6.25      -0.5766     0.367177         -0.9266         -0.7728    -1.7696
# 4  0.72 -2.50   2.50      -0.1319     0.808779         -0.4819         -0.3212    -1.4816
# 5  0.96 -2.25  -0.25       0.4934     1.443567          0.1334          0.3036    -1.0784
# 6  1.20 -1.20  -3.08       1.2575     2.256908          0.9075          1.1069    -0.5600
# 7  1.44  0.40   6.00           NA     3.264374          1.7758          2.0944     0.0736
# 8  1.68  2.38   6.00           NA     4.452011          2.7624          3.2720     0.8224
# 9  1.92 11.00 195.00       5.7229     5.837442          4.1729          4.6474     1.6864
# 10 2.16 10.00  37.00           NA     7.411697          5.1993          6.2317     2.6656
# 11 2.40 15.00 200.00           NA     9.182819          6.6066          8.0428     3.7600

Lower[, 9:12]

#    H_skurt_corr F_skurt_corr H_time F_time
# 1          0.33         1.16  1.757  8.227
# 2          0.35         1.15  1.566  8.164
# 3          0.35         1.14  1.630  6.321
# 4          0.35         1.13  1.537  5.568
# 5          0.36         1.14  1.558  5.540
# 6          0.35         1.15  1.602  6.619
# 7          2.00         1.17  9.088  8.835
# 8          2.00         1.18  9.425 11.103
# 9          1.55         1.19  6.776 14.364
# 10         2.00         1.18 11.174 15.382
# 11         2.00         1.14 10.567 18.184

# The 1st 3 columns give the skewness and standardized fifth and sixth cumulants.
# "H_valid.skurt" gives the lower kurtosis boundary that produces a valid power method pdf
#    using Headrick's approximation, with the kurtosis addition given in the "H_skurt_corr"
#     column if necessary.
# "F_valid.skurt" gives the lower kurtosis boundary that produces a valid power method pdf
#    using Fleishman's approximation, with the kurtosis addition given in the "F_skurt_corr"
#     column if necessary.
# "H_invalid.skurt" gives the lower kurtosis boundary that produces an invalid power method
#     pdf using Headrick's approximation, without the use of a kurtosis correction.
# "F_valid.skurt" gives the lower kurtosis boundary that produces an invalid power method
#     pdf using Fleishman's approximation, without the use of a kurtosis correction.
# "PBON_skurt" gives the lower kurtosis boundary approximation used in the PoisBinOrdNonNor
#     package.
# "H_time" gives the computation time (minutes) for Headrick's method.
# "F_time" gives the computation time (minutes) for Fleishman's method.
```

```
## End(Not run)
```

`calc_moments` *Find Standardized Cumulants of Data by Method of Moments*

Description

This function uses the method of moments to calculate the mean, standard deviation, skewness, standardized kurtosis, and standardized fifth and sixth cumulants given a vector of data. The result can be used as input to `find_constants` or for data simulation.

Usage

```
calc_moments(x)
```

Arguments

x	a vector of data

Value

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

References

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.

See Also

`calc_fisherk`, `calc_theory`, `find_constants`

Examples

```
x <- rgamma(n = 10000, 10, 10)
calc_moments(x)
```

`calc_theory` *Find Theoretical Standardized Cumulants for Continuous Distributions*

Description

This function calculates the theoretical mean, standard deviation, skewness, standardized kurtosis, and standardized fifth and sixth cumulants given either a Distribution name (plus up to 4 parameters) or a pdf (with specified lower and upper support bounds). The result can be used as input to `find_constants` or for data simulation.

Note: Due to the nature of the integration involved in calculating the standardized cumulants, the results are approximations. Greater accuracy can be achieved by increasing the number of subdivisions (sub) used in the integration process. However, the user may need to round the cumulants (i.e. using `round(x, 8)`) before using them in other functions (i.e. `find_constants`, `calc_lower_skurt`, `nonnormvar1`, `rcorrvar`, `rcorrvar2`) in order to achieve the desired results. For example, in order to ensure that skew is exactly 0 for symmetric distributions.

Usage

```
calc_theory(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"), params = NULL,
  fx = NULL, lower = NULL, upper = NULL, sub = 1000)
```

Arguments

`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". 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)
`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
`upper`	the upper support bound for a supplied fx, else keep NULL

sub the number of subdivisions to use in the integration; if no result, try increasing sub (requires longer computation time)

Value

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

References

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

Thomas W. Yee (2018). VGAM: Vector Generalized Linear and Additive Models. R package version 1.0-5. https://CRAN.R-project.org/package=VGAM

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

See Also

calc_fisherk, calc_moments, find_constants

Examples

```
options(scipen = 999)

# Pareto Distribution: params = c(alpha = scale, theta = shape)
calc_theory(Dist = "Pareto", params = c(1, 10))

# Generalized Rayleigh Distribution: params = c(alpha = scale, mu/sqrt(pi/2) = shape)
calc_theory(Dist = "Rayleigh", params = c(0.5, 1))

# Laplace Distribution: params = c(location, scale)
calc_theory(Dist = "Laplace", params = c(0, 1))

# Triangle Distribution: params = c(a, b)
calc_theory(Dist = "Triangular", params = c(0, 1))
```

cdf_prob *Calculate Theoretical Cumulative Probability for Continuous Variables*

Description

This function calculates a cumulative probability using the theoretical power method cdf $F_p(Z)(p(z)) = F_p(Z)(p(z), F_Z(z))$ up to $sigma * y + mu = delta$, where $y = p(z)$, after using pdf_check. If the given constants do not produce a valid power method pdf, a warning is given. The formulas were obtained from Headrick & Kowalchuk (2007, doi: 10.1080/10629360600605065).

Usage

```
cdf_prob(c, method = c("Fleishman", "Polynomial"), delta = 0.5, mu = 0,
  sigma = 1, lower = -1000000, upper = 1000000)
```

Arguments

c	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
method	the method used to find the constants. "Fleishman" uses a third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.
delta	the value $sigma * y + mu$, where $y = p(z)$, at which to evaluate the cumulative probability
mu	mean for the continuous variable
sigma	standard deviation for the continuous variable
lower	lower bound for integration of the standard normal variable Z that generates the continuous variable
upper	upper bound for integration

Value

A list with components:

`cumulative probability` the theoretical cumulative probability up to delta

`roots` the roots z that make $sigma * p(z) + mu = delta$

References

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

find_constants, pdf_check

Examples

```
# Normal distribution with Headrick's fifth-order PMT:
cdf_prob(c = c(0, 1, 0, 0, 0, 0), "Polynomial", delta = qnorm(0.05))

## Not run:
# Beta(a = 4, b = 2) Distribution:
con <- find_constants(method = "Polynomial", skews = -0.467707, skurts = -0.375,
                      fifths = 1.403122, sixths = -0.426136)$constants
cdf_prob(c = con, method = "Polynomial", delta = 0.5)

## End(Not run)
```

---

chat_nb — *Calculate Upper Frechet-Hoeffding Correlation Bound: Negative Binomial - Normal Variables*

Description

This function calculates the upper Frechet-Hoeffding bound on the correlation between a Negative Binomial variable and the normal variable used to generate it. It is used in findintercorr_cat_nb and findintercorr_cont_nb in calculating the intermediate MVN correlations. This extends the method of Amatya & Demirtas (2015, doi: 10.1080/00949655.2014.953534) to Negative Binomial variables. This function would not ordinarily be called directly by the user.

Usage

```
chat_nb(size, prob = NULL, mu = NULL, n_unif = 10000, seed = 1234)
```

Arguments

size	a vector of size parameters for the Negative Binomial variables (see NegBinomial)
prob	a vector of success probability parameters
mu	a vector of mean parameters (*Note: either prob or mu should be supplied for all Negative Binomial variables, not a mixture; default = NULL)
n_unif	the number of uniform random numbers to generate in calculating the bound (default = 10000)
seed	the seed used in random number generation (default = 1234)

Value

A scalar equal to the correlation upper bound.

References

Please see references for chat_pois.

See Also

findintercorr_cat_nb, findintercorr_cont_nb, findintercorr

---

chat_pois *Calculate Upper Frechet-Hoeffding Correlation Bound: Poisson - Normal Variables*

Description

This function calculates the upper Frechet-Hoeffding bound on the correlation between a Poisson variable and the normal variable used to generate it. It is used in findintercorr_cat_pois and findintercorr_cont_pois in calculating the intermediate MVN correlations. This uses the method of Amatya & Demirtas (2015, doi: 10.1080/00949655.2014.953534). This function would not ordinarily be called directly by the user.

Usage

```
chat_pois(lam, n_unif = 10000, seed = 1234)
```

Arguments

lam	a vector of lambda (> 0) constants for the Poisson variables (see Poisson)
n_unif	the number of uniform random numbers to generate in calculating the bound (default = 10000)
seed	the seed used in random number generation (default = 1234)

Value

A scalar equal to the correlation upper bound.

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. Sur les tableaux de correlation dont les marges sont donnees. Ann. l'Univ. Lyon SectA. 1951;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.

See Also

findintercorr_cat_pois, findintercorr_cont_pois, findintercorr

---

denom_corr_cat — *Calculate Denominator Used in Intercorrelations Involving Ordinal Variables*

Description

This function calculates part of the the denominator used to find intercorrelations involving ordinal variables or variables that are treated as ordinal (i.e. count variables in the method used in rcorrvar2). It uses the formula given by Olsson et al. (1982, doi: 10.1007/BF02294164) in describing polyserial and point-polyserial correlations. For an ordinal variable with r >= 2 categories, the value is given by:

$$\sum_{j=1}^{r-1} \phi(\tau_j) * (y_{j+1} - y_j),$$

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 $\Phi^{-1}(\sum_{i=1}^{j} Pr(Y = y_i))$. This function would not ordinarily be called directly by the user.

Usage

```
denom_corr_cat(marginal, support)
```

Arguments

`marginal`	a vector of cumulative probabilities defining the marginal distribution of the 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 vector of containing the ordered support values

Value

A scalar

References

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

See Also

`ordnorm`, `rcorrvar`, `rcorrvar2`, `findintercorr_cont_cat`, `findintercorr_cont_pois2`, `findintercorr_cont_nb2`

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

Description

This function corrects the final correlation of simulated variables to be within a precision value (`epsilon`) of the target correlation. It updates the pairwise intermediate MVN correlation iteratively in a loop until either the maximum error is less than epsilon or the number of iterations exceeds the maximum number set by the user (`maxit`). It uses `error_vars` to simulate all variables and calculate the correlation of all variables in each iteration. 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` has been modified in the following ways:

1) It works for continuous, ordinal (r >= 2 categories), and count variables.

2) The initial correlation check has been removed because this intermediate correlation Sigma from `rcorrvar` or `rcorrvar2` 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) A final "fail-safe" check was added at the end of the iteration loop where if the absolute error between the final and target pairwise correlation is still > 0.1, the intermediate correlation is set equal to the target correlation (if `extra_correct` = "TRUE").

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

Usage

```
error_loop(k_cat, k_cont, k_pois, k_nb, Y_cat, Y, Yb, Y_pois, Y_nb, marginal,
  support, method, means, vars, constants, lam, size, prob, mu, n, seed,
  epsilon, maxit, rho0, Sigma, rho_calc, extra_correct)
```

Arguments

`k_cat`	the number of ordinal (r >= 2 categories) variables
`k_cont`	the number of continuous variables
`k_pois`	the number of Poisson variables
`k_nb`	the number of Negative Binomial variables
`Y_cat`	the ordinal variables generated from `rcorrvar` or `rcorrvar2`
`Y`	the continuous (mean 0, variance 1) variables
`Yb`	the continuous variables with desired mean and variance
`Y_pois`	the Poisson variables
`Y_nb`	the Negative Binomial variables
`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
`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
`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`
`lam`	a vector of lambda (> 0) constants for the Poisson variables (see `Poisson`)
`size`	a vector of size parameters for the Negative Binomial variables (see `NegBinomial`)
`prob`	a vector of success probability parameters
`mu`	a vector of mean parameters (*Note: either `prob` or `mu` should be supplied for all Negative Binomial variables, not a mixture)
`n`	the sample size
`seed`	the seed value for random number generation
`epsilon`	the maximum acceptable error between the final and target correlation matrices; 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 rcorrvar or rcorrvar2
rho_calc	the final correlation matrix calculated in rcorrvar or rcorrvar2
extra_correct	if "TRUE", a final "fail-safe" check is used at the end of the iteration loop where if the absolute error between the final and target pairwise correlation is still > 0.1, the intermediate correlation is set equal to the target correlation

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

Yb 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

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.

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.

See Also

ordcont, rcorrvar, rcorrvar2, findintercorr, findintercorr2

---

error_vars *Generate Variables for Error Loop*

Description

This function simulates the continuous, ordinal (r >= 2 categories), Poisson, or Negative Binomial variables used in error_loop. It is called in each iteration, regenerates all variables, and calculates the resulting correlation matrix. This function would not ordinarily be called directly by the user.

Usage

```
error_vars(marginal, support, method, means, vars, constants, lam, size, prob,
  mu, Sigma, rho_calc, q, r, k_cat, k_cont, k_pois, k_nb, Y_cat, Y, Yb, Y_pois,
  Y_nb, n, seed)
```

Arguments

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
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
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
lam	a vector of lambda (> 0) constants for the Poisson variables (see Poisson)
size	a vector of size parameters for the Negative Binomial variables (see NegBinomial)
prob	a vector of success probability parameters

mu	a vector of mean parameters (*Note: either prob or mu should be supplied for all Negative Binomial variables, not a mixture)
Sigma	the 2 x 2 intermediate correlation matrix generated by error_loop
rho_calc	the 2 x 2 final correlation matrix calculated in error_loop
q	the row index of the 1st variable
r	the column index of the 2nd variable
k_cat	the number of ordinal (r >= 2 categories) variables
k_cont	the number of continuous variables
k_pois	the number of Poisson variables
k_nb	the number of Negative Binomial variables
Y_cat	the ordinal variables generated from error_loop
Y	the continuous (mean 0, variance 1) variables
Yb	the continuous variables with desired mean and variance
Y_pois	the Poisson variables
Y_nb	the Negative Binomial variables
n	the sample size
seed	the seed value for random number generation

Value

A list with the following components:

Sigma the intermediate MVN correlation matrix

rho_calc the calculated final correlation matrix generated from Sigma

Y_cat the ordinal variables

Y the continuous (mean 0, variance 1) variables

Yb the continuous variables with desired mean and variance

Y_pois the Poisson variables

Y_nb the Negative Binomial variables

References

Please see references for error_loop.

See Also

ordcont, rcorrvar, rcorrvar2, error_loop findintercorr *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 rcorrvar. The ordering of the variables must be ordinal, continuous, Poisson, and Negative Binomial (note that it is possible for k_cat, k_cont, k_pois, and/or k_nb to be 0). The function first checks that the target correlation matrix rho is positive-definite and the marginal distributions for the ordinal variables are cumulative probabilities with r - 1 values (for r categories). There is a warning given at the end of simulation if the calculated intermediate correlation matrix Sigma is not positive-definite. This function is called by the simulation function rcorrvar, and would only be used separately if the user wants to find the intermediate correlation matrix only. The simulation functions also return the intermediate correlation matrix.

Usage

```
findintercorr(n, k_cont = 0, k_cat = 0, k_pois = 0, k_nb = 0,
  method = c("Fleishman", "Polynomial"), constants, marginal = list(),
  support = list(), nrand = 100000, lam = NULL, size = NULL,
  prob = NULL, mu = NULL, rho = NULL, seed = 1234, epsilon = 0.001,
  maxit = 1000)
```

Arguments

n	the sample size (i.e. the length of each simulated variable)
k_cont	the number of continuous variables (default = 0)
k_cat	the number of ordinal (r >= 2 categories) variables (default = 0)
k_pois	the number of Poisson variables (default = 0)
k_nb	the number of 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

nrand	the number of random numbers to generate in calculating the bound (default = 10000)
lam	a vector of lambda (> 0) constants for the Poisson variables (see Poisson)
size	a vector of size parameters for the Negative Binomial variables (see NegBinomial)
prob	a vector of success probability parameters
mu	a vector of mean parameters (*Note: either prob or mu should be supplied for all Negative Binomial variables, not a mixture; default = NULL)
rho	the target correlation matrix (*must be ordered ordinal, continuous, Poisson, Negative Binomial*; default = NULL)
seed	the seed value for random number generation (default = 1234)
epsilon	the maximum acceptable error between the final and target correlation matrices (default = 0.001) in the calculation of ordinal intermediate correlations with ordnorm
maxit	the maximum number of iterations to use (default = 1000) in the calculation of ordinal intermediate correlations with ordnorm

Value

the intermediate MVN correlation matrix

Overview of Correlation Method 1

The intermediate correlations used in correlation method 1 are more simulation based than those in correlation 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.

The processes used to find the intermediate correlations for each variable type are described below. Please see the corresponding function help page for more information:

Ordinal Variables

Correlations are computed pairwise. If both variables are binary, the method of Demirtas et al. (2012, doi: 10.1002/sim.5362) is used to find the tetrachoric correlation (code adapted from `Tetra.Corr.BB`). This method is based on Emrich and Piedmonte's (1991, doi: 10.1080/00031305.1991.10475828) work, in which the joint binary distribution is determined from the third and higher moments of a multivariate normal distribution: Let Y_1 and Y_2 be binary variables with $E[Y_1] = Pr(Y_1 = 1) = p_1$, $E[Y_2] = Pr(Y_2 = 1) = p_2$, and correlation ρ_{y1y2}. Let $\Phi[x_1, x_2, \rho_{x1x2}]$ be the standard bivariate normal cumulative distribution function, given by:

$$\Phi[x_1, x_2, \rho_{x1x2}] = \int_{-\infty}^{x_1} \int_{-\infty}^{x_2} f(z_1, z_2, \rho_{x1x2}) dz_1 dz_2$$

where

$$f(z_1, z_2, \rho_{x1x2}) = [2\pi\sqrt{1 - \rho_{x1x2}^2}]^{-1} * exp[-0.5(z_1^2 - 2\rho_{x1x2} z_1 z_2 + z_2^2)/(1 - \rho_{x1x2}^2)]$$

Then solving the equation

$$\Phi[z(p_1), z(p_2), \rho_{x1x2}] = \rho_{y1y2}\sqrt{p_1(1 - p_1)p_2(1 - p_2)} + p_1 p_2$$

for ρ_{x1x2} gives the intermediate correlation of the standard normal variables needed to generate binary variables with correlation ρ_{y1y2}. Here $z(p)$ indicates the pth quantile of the standard normal distribution.

Otherwise, `ordnorm` is called for each pair. If the resulting intermediate matrix is not positive-definite, there is a warning given because it may not be possible to find a MVN correlation matrix that will produce the desired marginal distributions after discretization. Any problems with positive-definiteness are corrected later.

Continuous Variables

Correlations are computed pairwise. `findintercorr_cont` is called for each pair.

Poisson Variables

`findintercorr_pois` is called to calculate the intermediate MVN correlation for all Poisson variables.

Negative Binomial Variables

`findintercorr_nb` is called to calculate the intermediate MVN correlation for all Negative Binomial variables.

Continuous - Ordinal Pairs

`findintercorr_cont_cat` is called to calculate the intermediate MVN correlation for all Continuous and Ordinal combinations.

Ordinal - Poisson Pairs

`findintercorr_cat_pois` is called to calculate the intermediate MVN correlation for all Ordinal and Poisson combinations.

Ordinal - Negative Binomial Pairs

findintercorr_cat_nb is called to calculate the intermediate MVN correlation for all Ordinal and Negative Binomial combinations.

Continuous - Poisson Pairs

findintercorr_cont_pois is called to calculate the intermediate MVN correlation for all Continuous and Poisson combinations.

Continuous - Negative Binomial Pairs

findintercorr_cont_nb is called to calculate the intermediate MVN correlation for all Continuous and Negative Binomial combinations.

Poisson - Negative Binomial Pairs

findintercorr_pois_nb is called to calculate the intermediate MVN correlation for all Poisson and Negative Binomial combinations.

References

Please see rcorrvar for additional references.

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.

Inan G & Demirtas H (2016). BinNonNor: Data Generation with Binary and Continuous Non-Normal Components. R package version 1.3. https://CRAN.R-project.org/package=BinNonNor

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

See Also

find_constants, rcorrvar

Examples

```
## Not run:

# Binary, Ordinal, Continuous, Poisson, and Negative Binomial Variables

options(scipen = 999)
seed <- 1234
n <- 10000

# Continuous Distributions: Normal, t (df = 10), Chisq (df = 4),
# Beta (a = 4, b = 2), Gamma (a = 4, b = 4)
Dist <- c("Gaussian", "t", "Chisq", "Beta", "Gamma")

# calculate standardized cumulants
# those for the normal and t distributions are rounded to ensure the
# correct values (i.e. skew = 0)
```

```
M1 <- round(calc_theory(Dist = "Gaussian", params = c(0, 1)), 8)
M2 <- round(calc_theory(Dist = "t", params = 10), 8)
M3 <- calc_theory(Dist = "Chisq", params = 4)
M4 <- calc_theory(Dist = "Beta", params = c(4, 2))
M5 <- calc_theory(Dist = "Gamma", params = c(4, 4))
M <- cbind(M1, M2, M3, M4, M5)
M <- round(M[-c(1:2),], digits = 6)
colnames(M) <- Dist
rownames(M) <- c("skew", "skurtosis", "fifth", "sixth")
means <- rep(0, length(Dist))
vars <- rep(1, length(Dist))

# calculate constants
con <- matrix(1, nrow = ncol(M), ncol = 6)
for (i in 1:ncol(M)) {
 con[i, ] <- find_constants(method = "Polynomial", skews = M[1, i],
                            skurts = M[2, i], fifths = M[3, i],
                            sixths = M[4, i])
}

# Binary and Ordinal Distributions
marginal <- list(0.3, 0.4, c(0.1, 0.5), c(0.3, 0.6, 0.9),
                 c(0.2, 0.4, 0.7, 0.8))
support <- list()

# Poisson Distributions
lam <- c(1, 5, 10)

# Negative Binomial Distributions
size <- c(3, 6)
prob <- c(0.2, 0.8)

ncat <- length(marginal)
ncont <- ncol(M)
npois <- length(lam)
nnb <- length(size)

# Create correlation matrix from a uniform distribution (-0.8, 0.8)
set.seed(seed)
Rey <- diag(1, nrow = (ncat + ncont + npois + nnb))
for (i in 1:nrow(Rey)) {
  for (j in 1:ncol(Rey)) {
    if (i > j) Rey[i, j] <- runif(1, -0.8, 0.8)
    Rey[j, i] <- Rey[i, j]
  }
}

# Test for positive-definiteness
library(Matrix)
if(min(eigen(Rey, symmetric = TRUE)$values) < 0) {
  Rey <- as.matrix(nearPD(Rey, corr = T, keepDiag = T)$mat)
}
```

```
# Make sure Rey is within upper and lower correlation limits
valid <- valid_corr(k_cat = ncat, k_cont = ncont, k_pois = npois,
                    k_nb = nnb, method = "Polynomial", means = means,
                    vars = vars, skews = M[1, ], skurts = M[2, ],
                    fifths = M[3, ], sixths = M[4, ], marginal = marginal,
                    lam = lam, size = size, prob = prob, rho = Rey,
                    seed = seed)

# Find intermediate correlation
Sigma1 <- findintercorr(n = n, k_cont = ncont, k_cat = ncat, k_pois = npois,
                        k_nb = nnb, method = "Polynomial", constants = con,
                        marginal = marginal, lam = lam, size = size,
                        prob = prob, rho = Rey, seed = seed)
Sigma1


## End(Not run)
```

`findintercorr2` *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 `rcorrvar2`. The ordering of the variables must be ordinal, continuous, Poisson, and Negative Binomial (note that it is possible for `k_cat`, `k_cont`, `k_pois`, and/or `k_nb` to be 0). The function first checks that the target correlation matrix `rho` is positive-definite and the marginal distributions for the ordinal variables are cumulative probabilities with r - 1 values (for r categories). There is a warning given at the end of simulation if the calculated intermediate correlation matrix `Sigma` is not positive-definite. This function is called by the simulation function `rcorrvar2`, and would only be used separately if the user wants to find the intermediate correlation matrix only. The simulation functions also return the intermediate correlation matrix.

Usage

```
findintercorr2(n, k_cont = 0, k_cat = 0, k_pois = 0, k_nb = 0,
  method = c("Fleishman", "Polynomial"), constants, marginal = list(),
  support = list(), lam = NULL, size = NULL, prob = NULL, mu = NULL,
  pois_eps = NULL, nb_eps = NULL, rho = NULL, epsilon = 0.001,
  maxit = 1000)
```

Arguments

`n`	the sample size (i.e. the length of each simulated variable)
`k_cont`	the number of continuous variables (default = 0)
`k_cat`	the number of ordinal (r >= 2 categories) variables (default = 0)

`k_pois`	the number of Poisson variables (default = 0)
`k_nb`	the number of 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 (> 0) constants for the Poisson variables (see `Poisson`)
`size`	a vector of size parameters for the Negative Binomial variables (see `NegBinomial`)
`prob`	a vector of success probability parameters
`mu`	a vector of mean parameters (*Note: either `prob` or `mu` should be supplied for all Negative Binomial variables, not a mixture; default = NULL)
`pois_eps`	a vector of length `k_pois` containing the truncation values (i.e. = rep(0.0001, k_pois); default = NULL)
`nb_eps`	a vector of length `k_nb` containing the truncation values (i.e. = rep(0.0001, k_nb); default = NULL)
`rho`	the target correlation matrix (*must be ordered ordinal, continuous, Poisson, Negative Binomial*; default = NULL)
`epsilon`	the maximum acceptable error between the final and target correlation matrices (default = 0.001) in the calculation of ordinal intermediate correlations with `ordnorm`
`maxit`	the maximum number of iterations to use (default = 1000) in the calculation of ordinal intermediate correlations with `ordnorm`

Value

the intermediate MVN correlation matrix

Overview of Correlation Method 2

The intermediate correlations used in correlation method 2 are less simulation based than those in correlation 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 `ordnorm`.

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.

The processes used to find the intermediate correlations for each variable type are described below. Please see the corresponding function help page for more information:

Ordinal Variables

Correlations are computed pairwise. If both variables are binary, the method of Demirtas et al. (2012, doi: 10.1002/sim.5362) is used to find the tetrachoric correlation (code adapted from `Tetra.Corr.BB`). This method is based on Emrich and Piedmonte's (1991, doi: 10.1080/00031305.1991.10475828) work, in which the joint binary distribution is determined from the third and higher moments of a multivariate normal distribution: Let Y_1 and Y_2 be binary variables with $E[Y_1] = Pr(Y_1 = 1) = p_1$, $E[Y_2] = Pr(Y_2 = 1) = p_2$, and correlation ρ_{y1y2}. Let $\Phi[x_1, x_2, \rho_{x1x2}]$ be the standard bivariate normal cumulative distribution function, given by:

$$\Phi[x_1, x_2, \rho_{x1x2}] = \int_{-\infty}^{x_1} \int_{-\infty}^{x_2} f(z_1, z_2, \rho_{x1x2}) dz_1 dz_2$$

where

$$f(z_1, z_2, \rho_{x1x2}) = [2\pi\sqrt{1 - \rho_{x1x2}^2}]^{-1} * exp[-0.5(z_1^2 - 2\rho_{x1x2} z_1 z_2 + z_2^2)/(1 - \rho_{x1x2}^2)]$$

Then solving the equation

$$\Phi[z(p_1), z(p_2), \rho_{x1x2}] = \rho_{y1y2}\sqrt{p_1(1 - p_1)p_2(1 - p_2)} + p_1 p_2$$

for ρ_{x1x2} gives the intermediate correlation of the standard normal variables needed to generate binary variables with correlation ρ_{y1y2}. Here $z(p)$ indicates the pth quantile of the standard normal distribution.

Otherwise, `ordnorm` is called for each pair. If the resulting intermediate matrix is not positive-definite, there is a warning given because it may not be possible to find a MVN correlation matrix that will produce the desired marginal distributions after discretization. Any problems with positive-definiteness are corrected later.

Continuous Variables

Correlations are computed pairwise. `findintercorr_cont` is called for each pair.

Poisson Variables

`max_count_support` is used to find the maximum support value given the vector pois_eps of truncation values. This is used to create a Poisson marginal list consisting of cumulative probabilities for each variable (like that for the ordinal variables). Then `ordnorm` is called to calculate the intermediate MVN correlation for all Poisson variables.

Negative Binomial Variables

`max_count_support` is used to find the maximum support value given the vector nb_eps of truncation values. This is used to create a Negative Binomial marginal list consisting of cumulative probabilities for each variable (like that for the ordinal variables). Then `ordnorm` is called to calculate the intermediate MVN correlation for all Negative Binomial variables.

Continuous - Ordinal Pairs

`findintercorr_cont_cat` is called to calculate the intermediate MVN correlation for all Continuous and Ordinal combinations.

Ordinal - Poisson Pairs

The Poisson marginal list is appended to the ordinal marginal list (similarly for the support lists). Then `ordnorm` is called to calculate the intermediate MVN correlation for all Ordinal and Poisson combinations.

Ordinal - Negative Binomial Pairs

The Negative Binomial marginal list is appended to the ordinal marginal list (similarly for the support lists). Then `ordnorm` is called to calculate the intermediate MVN correlation for all Ordinal and Negative Binomial combinations.

Continuous - Poisson Pairs

`findintercorr_cont_pois2` is called to calculate the intermediate MVN correlation for all Continuous and Poisson combinations.

Continuous - Negative Binomial Pairs

`findintercorr_cont_nb2` is called to calculate the intermediate MVN correlation for all Continuous and Negative Binomial combinations.

Poisson - Negative Binomial Pairs

The Negative Binomial marginal list is appended to the Poisson marginal list (similarly for the support lists). Then `ordnorm` is called to calculate the intermediate MVN correlation for all Poisson and Negative Binomial combinations.

References

Please see rcorrvar2 for additional references.

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.

Inan G & Demirtas H (2016). BinNonNor: Data Generation with Binary and Continuous Non-Normal Components. R package version 1.3. https://CRAN.R-project.org/package=BinNonNor

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

See Also

find_constants, rcorrvar2

Examples

```
## Not run:

# Binary, Ordinal, Continuous, Poisson, and Negative Binomial Variables

options(scipen = 999)
seed <- 1234
n <- 10000

# Continuous Distributions: Normal, t (df = 10), Chisq (df = 4),
# Beta (a = 4, b = 2), Gamma (a = 4, b = 4)
Dist <- c("Gaussian", "t", "Chisq", "Beta", "Gamma")

# calculate standardized cumulants
# those for the normal and t distributions are rounded to ensure the
# correct values (i.e. skew = 0)

M1 <- round(calc_theory(Dist = "Gaussian", params = c(0, 1)), 8)
M2 <- round(calc_theory(Dist = "t", params = 10), 8)
M3 <- calc_theory(Dist = "Chisq", params = 4)
M4 <- calc_theory(Dist = "Beta", params = c(4, 2))
M5 <- calc_theory(Dist = "Gamma", params = c(4, 4))
M <- cbind(M1, M2, M3, M4, M5)
M <- round(M[-c(1:2),], digits = 6)
colnames(M) <- Dist
rownames(M) <- c("skew", "skurtosis", "fifth", "sixth")
means <- rep(0, length(Dist))
vars <- rep(1, length(Dist))

# calculate constants
con <- matrix(1, nrow = ncol(M), ncol = 6)
for (i in 1:ncol(M)) {
 con[i, ] <- find_constants(method = "Polynomial", skews = M[1, i],
                            skurts = M[2, i], fifths = M[3, i],
                            sixths = M[4, i])
}
```

```
# Binary and Ordinal Distributions
marginal <- list(0.3, 0.4, c(0.1, 0.5), c(0.3, 0.6, 0.9),
                 c(0.2, 0.4, 0.7, 0.8))
support <- list()

# Poisson Distributions
lam <- c(1, 5, 10)

# Negative Binomial Distributions
size <- c(3, 6)
prob <- c(0.2, 0.8)

ncat <- length(marginal)
ncont <- ncol(M)
npois <- length(lam)
nnb <- length(size)

# Create correlation matrix from a uniform distribution (-0.8, 0.8)
set.seed(seed)
Rey <- diag(1, nrow = (ncat + ncont + npois + nnb))
for (i in 1:nrow(Rey)) {
  for (j in 1:ncol(Rey)) {
    if (i > j) Rey[i, j] <- runif(1, -0.8, 0.8)
    Rey[j, i] <- Rey[i, j]
  }
}

# Test for positive-definiteness
library(Matrix)
if(min(eigen(Rey, symmetric = TRUE)$values) < 0) {
  Rey <- as.matrix(nearPD(Rey, corr = T, keepDiag = T)$mat)
}

# Make sure Rey is within upper and lower correlation limits
valid <- valid_corr2(k_cat = ncat, k_cont = ncont, k_pois = npois,
                     k_nb = nnb, method = "Polynomial", means = means,
                     vars = vars, skews = M[1, ], skurts = M[2, ],
                     fifths = M[3, ], sixths = M[4, ],
                     marginal = marginal, lam = lam,
                     pois_eps = rep(0.0001, npois),
                     size = size, prob = prob,
                     nb_eps = rep(0.0001, nnb),
                     rho = Rey, seed = seed)

# Find intermediate correlation
Sigma2 <- findintercorr2(n = n, k_cont = ncont, k_cat = ncat,
                         k_pois = npois, k_nb = nnb,
                         method = "Polynomial", constants = con,
                         marginal = marginal, lam = lam, size = size,
                         prob = prob, pois_eps = rep(0.0001, npois),
                         nb_eps = rep(0.0001, nnb), rho = Rey)
Sigma2
```

```
## End(Not run)
```

---

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

Description

This function calculates a `k_cat x k_nb` intermediate matrix of correlations for the k_cat ordinal (r >= 2 categories) and k_nb Negative Binomial variables. It extends the method of Amatya & Demirtas (2015, doi: 10.1080/00949655.2014.953534) to ordinal - Negative Binomial pairs. 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 (see `chat_nb`) 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 `findintercorr` and `rcorrvar`. This function would not ordinarily be called by the user.

Usage

```
findintercorr_cat_nb(rho_cat_nb, marginal, size, prob, mu = NULL,
  nrand = 100000, seed = 1234)
```

Arguments

`rho_cat_nb`	a `k_cat x k_nb` matrix of target correlations among ordinal and Negative Binomial variables
`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 `NegBinomial`)
`prob`	a vector of success probability parameters
`mu`	a vector of mean parameters (*Note: either `prob` or `mu` should be supplied for all Negative Binomial variables, not a mixture; default = NULL)
`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 findintercorr_cat_pois

See Also

chat_nb, findintercorr, rcorrvar

---

findintercorr_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. It extends the method of Amatya & Demirtas (2015, doi: 10.1080/00949655.2014.953534) to ordinal - Poisson pairs. 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 (see chat_pois) 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 findintercorr and rcorrvar. This function would not ordinarily be called by the user.

Usage

```
findintercorr_cat_pois(rho_cat_pois, marginal, lam, nrand = 100000,
  seed = 1234)
```

Arguments

rho_cat_pois	a k_cat x k_pois matrix of target correlations among ordinal and Poisson variables
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 (> 0) constants for the Poisson variables (see Poisson)
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. Sur les tableaux de correlation dont les marges sont donnees. Ann. l'Univ. Lyon SectA. 1951;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.

See Also

`chat_pois`, `findintercorr`, `rcorrvar`

`findintercorr_cont` *Calculate Intermediate MVN Correlation for Continuous Variables Generated by Polynomial Transformation*

Description

This function finds the roots to the equations in `intercorr_fleish` or `intercorr_poly` using `nleqslv`. It is used in `findintercorr` and `findintercorr2` to find 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. It works for two or three variables in the case of `method` = "Fleishman", or two, three, or four variables in the case of `method` = "Polynomial". Otherwise, Headrick & Sawilowsky (1999, doi: 10.1007/BF02294317) recommend using the technique of Vale & Maurelli (1983, doi: 10.1007/BF02293687), in which the intermediate correlations are found pairwise and then eigen value decomposition is used on the intermediate correlation matrix. This function would not ordinarily be called by the user.

Usage

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

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 either 2, 3, or 4 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	a matrix of target correlations among continuous variables; if nrow(rho_cont) = 1, it represents a pairwise correlation; if nrow(rho_cont) = 2, 3, or 4, it represents a correlation matrix between two, three, or four variables

Value

a list containing the results from nleqslv

References

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

Hasselman B (2018). nleqslv: Solve Systems of Nonlinear Equations. R package version 3.3.2. https://CRAN.R-project.org/package=nleqslv

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.

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

See Also

poly, fleish, power_norm_corr, pdf_check, find_constants, intercorr_fleish, intercorr_poly, nleqslv findintercorr_cont_cat *Calculate Intermediate MVN Correlation for Continuous - Ordinal Variables*

Description

This function calculates a k_cont x k_cat intermediate matrix of correlations for the k_cont continuous and k_cat ordinal (r >= 2 categories) variables. It extends the method of Demirtas et al. (2012, doi: 10.1198/tast.2011.10090) in simulating binary and non-normal data using the Fleishman transformation by:

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

2) allowing for ordinal variables with more than 2 categories.

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 discretized to produce an ordinal variable Y2) 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. The point-polyserial correlation is given by:

$$\rho_{y2,z2} = (1/\sigma_{y2}) * \sum_{j=1}^{r-1} \phi(\tau_j)(y2_{j+1} - y2_j)$$

where

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

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:

$$\rho_{y1,z1} = c1 + 3c3 + 15c5$$

where c5 = 0 if method = "Fleishman". The function is used in findintercorr and findintercorr2. This function would not ordinarily be called by the user.

Usage

```
findintercorr_cont_cat(method = c("Fleishman", "Polynomial"), constants,
  rho_cont_cat, marginal, support)
```

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_cat — a k_cont x k_cat matrix of target correlations among continuous and ordinal variables

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)

support — a list of length equal to k_cat; the i-th element is a vector of containing the r ordered support values

Value

a k_cont x k_cat matrix whose rows represent the k_cont continuous variables and columns represent the k_cat ordinal variables

References

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.

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.

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

See Also

power_norm_corr, find_constants, findintercorr, findintercorr2 findintercorr_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 Negative Binomial 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 (see chat_nb) and the power method correlation (described in Headrick & Kowalchuk, 2007, doi: 10.1080/10629360600605065) between Y1 and Z1. The function is used in findintercorr and rcorrvar. This function would not ordinarily be called by the user.

Usage

```
findintercorr_cont_nb(method, constants, rho_cont_nb, size, prob, mu = NULL,
  nrand = 100000, 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
size	a vector of size parameters for the Negative Binomial variables (see NegBinomial)
prob	a vector of success probability parameters
mu	a vector of mean parameters (*Note: either prob or mu should be supplied for all Negative Binomial variables, not a mixture; default = NULL)
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 `findintercorr_cont_pois`.

See Also

`chat_nb`, `power_norm_corr`, `find_constants`, `findintercorr`, `rcorrvar`

`findintercorr_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 count (Poisson and 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`).

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 `max_count_support`, the point-polyserial correlation is given by:

$$\rho_{y2,z2} = (1/\sigma_{y2}) \sum_{j=1}^{r-1} \phi(\tau_j)(y2_{j+1} - y2_j)$$

where

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

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:

$$\rho_{y1,z1} = c1 + 3c3 + 15c5$$

, where c5 = 0 if `method` = "Fleishman". The function is used in `findintercorr2` and `rcorrvar2`. This function would not ordinarily be called by the user.

Usage

```
findintercorr_cont_nb2(method, constants, rho_cont_nb, nb_marg, nb_support)
```

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
nb_marg	a list of length equal to k_nb; 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 findintercorr2 and rcorrvar2
nb_support	a list of length equal to k_nb; the i-th element is a vector of containing the r ordered support values, with a minimum of 0 and maximum determined via max_count_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 additional references in findintercorr_cont_cat.

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, findintercorr2, rcorrvar2

---

findintercorr_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. 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 (see chat_pois) and the power method correlation (described in Headrick & Kowalchuk, 2007, doi: 10.1080/10629360600605065) between Y1 and Z1. The function is used in findintercorr and rcorrvar. This function would not ordinarily be called by the user.

Usage

```
findintercorr_cont_pois(method, constants, rho_cont_pois, lam, nrand = 100000,
  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
lam	a vector of lambda (> 0) constants for the Poisson variables (see Poisson)
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.

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

Frechet M. Sur les tableaux de correlation dont les marges sont donnees. Ann. l'Univ. Lyon SectA. 1951;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, 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.

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.

See Also

`chat_pois`, `power_norm_corr`, `find_constants`, `findintercorr`, `rcorrvar`

`findintercorr_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 count 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`).

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 `max_count_support`, the point-polyserial correlation is given by:

$$\rho_{y2,z2} = (1/\sigma_{y2}) \sum_{j=1}^{r-1} \phi(\tau_j)(y2_{j+1} - y2_j)$$

where

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

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:

$$\rho_{y1,z1} = c1 + 3c3 + 15c5$$

, where c5 = 0 if `method` = "Fleishman". The function is used in `findintercorr2` and `rcorrvar2`. This function would not ordinarily be called by the user.

Usage

```
findintercorr_cont_pois2(method, constants, rho_cont_pois, pois_marg,
  pois_support)
```

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
`pois_marg`	a list of length equal to `k_pois`; 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 `findintercorr2` and `rcorrvar2`
`pois_support`	a list of length equal to `k_pois`; the i-th element is a vector of containing the r ordered support values, with a minimum of 0 and maximum determined via `max_count_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 findintercorr_cont_cat.

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, findintercorr2, rcorrvar2

---

findintercorr_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:

$$\rho_{z1,z2} = (1/b) * log((\rho_{y1,y2} - c)/a)$$

, 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 $\rho_{z1,z2}$ >= 1, $\rho_{z1,z2}$ is set to 0.99; if $\rho_{z1,z2}$ <= -1, $\rho_{z1,z2}$ is set to -0.99

3) simulating Negative Binomial variables.

The function is used in findintercorr and rcorrvar. This function would not ordinarily be called by the user.

Usage

```
findintercorr_nb(rho_nb, size, prob, mu = NULL, nrand = 100000,
  seed = 1234)
```

Arguments

rho_nb	a k_nb x k_nb matrix of target correlations
size	a vector of size parameters for the Negative Binomial variables (see NegBinomial)
prob	a vector of success probability parameters
mu	a vector of mean parameters (*Note: either prob or mu should be supplied for all Negative Binomial variables, not a mixture; default = NULL)

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 `findintercorr_pois`.

See Also

`PoisNor-package`, `findintercorr_pois`, `findintercorr_pois_nb`, `findintercorr`, `rcorrvar`

`findintercorr_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) $\rho_{y1,y2}$ are simulated. Then the intermediate correlation is found as follows:

$$\rho_{z1,z2} = (1/b) * log((\rho_{y1,y2} - c)/a)$$

, 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 $\rho_{z1,z2}$ >= 1, $\rho_{z1,z2}$ is set to 0.99; if $\rho_{z1,z2}$ <= -1, $\rho_{z1,z2}$ is set to -0.99.

The function is used in `findintercorr` and `rcorrvar`. This function would not ordinarily be called by the user.

Note: The method used here is also used in the packages `PoisBinOrdNor-package` and `PoisBinOrdNonNor-package` by Demirtas et al. (2017), but without my modifications.

Usage

```
findintercorr_pois(rho_pois, lam, nrand = 100000, seed = 1234)
```

Arguments

rho_pois	a k_pois x k_pois matrix of target correlations
lam	a vector of lambda (> 0) constants for the Poisson variables (see Poisson)
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.

Amatya A & Demirtas H (2016). PoisNor: Simultaneous Generation of Multivariate Data with Poisson and Normal Marginals. R package version 1.1. https://CRAN.R-project.org/package=PoisNor

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

Demirtas H, Hu Y, & Allozi R (2017). PoisBinOrdNor: Data Generation with Poisson, Binary, Ordinal and Normal Components. R package version 1.4. https://CRAN.R-project.org/package=PoisBinOrdNor

Demirtas H, Nordgren R, & Allozi R (2017). PoisBinOrdNonNor: Generation of Up to Four Different Types of Variables. R package version 1.3. https://CRAN.R-project.org/package=PoisBinOrdNonNor

Frechet M. Sur les tableaux de correlation dont les marges sont donnees. Ann. l'Univ. Lyon SectA. 1951;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.

See Also

PoisNor-package, findintercorr_nb, findintercorr_pois_nb, findintercorr, rcorrvar `findintercorr_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:

$$\rho_{z1,z2} = (1/b) * log((\rho_{y1,y2} - c)/a)$$

, 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 $\rho_{z1,z2}$ >= 1, $\rho_{z1,z2}$ is set to 0.99; if $\rho_{z1,z2}$ <= -1, $\rho_{z1,z2}$ is set to -0.99

3) simulating Negative Binomial variables. The function is used in `findintercorr` and `rcorrvar`. This function would not ordinarily be called by the user.

Usage

```
findintercorr_pois_nb(rho_pois_nb, lam, size, prob, mu = NULL,
  nrand = 100000, seed = 1234)
```

Arguments

`rho_pois_nb`	a `k_pois x k_nb` matrix of target correlations
`lam`	a vector of lambda (> 0) constants for the Poisson variables (see `Poisson`)
`size`	a vector of size parameters for the Negative Binomial variables (see `NegBinomial`)
`prob`	a vector of success probability parameters
`mu`	a vector of mean parameters (*Note: either `prob` or `mu` should be supplied for all Negative Binomial variables, not a mixture; default = NULL)
`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 findintercorr_pois.

See Also

PoisNor-package, findintercorr_pois, findintercorr_nb, findintercorr, rcorrvar

---

find_constants *Find Power Method Transformation Constants*

Description

This function calculates Fleishman's third or Headrick's fifth-order constants necessary to transform a standard normal random variable into a continuous variable with the specified skewness, standardized kurtosis, and standardized fifth and sixth cumulants. It uses multiStart to find solutions to fleish or nleqslv for poly. Multiple starting values are used to ensure the correct solution is found. If not user-specified and method = "Polynomial", the cumulant values are checked to see if they fall in Headrick's Table 1 (2002, p.691-2, doi: 10.1016/S01679473(02)000725) of common distributions (see Headrick.dist). If so, his solutions are used as starting values. Otherwise, a set of n values randomly generated from uniform distributions is used to determine the power method constants.

Each set of constants is checked for a positive correlation with the underlying normal variable (using power_norm_corr) and a valid power method pdf (using pdf_check). If the correlation is <= 0, the signs of c1 and c3 are reversed (for method = "Fleishman"), or c1, c3, and c5 (for method = "Polynomial"). These sign changes have no effect on the cumulants of the resulting distribution. If only invalid pdf constants are found and a vector of sixth cumulant correction values (Six) is provided, each is checked for valid pdf constants. The smallest correction that generates a valid power method pdf is used. If valid pdf constants still can not be found, the original invalid pdf constants (calculated without a sixth cumulant correction) will be provided if they exist. If not, the invalid pdf constants calculated with the sixth cumulant correction will be provided. If no solutions can be found, an error is given and the result is NULL.

Usage

```
find_constants(method = c("Fleishman", "Polynomial"), skews = NULL,
  skurts = NULL, fifths = NULL, sixths = NULL, Six = NULL,
  cstart = NULL, n = 25, seed = 1234)
```

Arguments

method	the method used to find the constants. "Fleishman" uses a third-order polynomial transformation and requires skewness and standardized kurtosis inputs. "Polynomial" uses Headrick's fifth-order transformation and requires all four standardized cumulants.
skews	the skewness value

skurts
: the standardized kurtosis value (kurtosis - 3, so that normal variables have a value of 0)

fifths
: the standardized fifth cumulant (if method = "Fleishman", keep NULL)

sixths
: the standardized sixth cumulant (if method = "Fleishman", keep NULL)

Six
: a vector of correction values to add to the sixth cumulant if no valid pdf constants are found, ex: `Six = seq(1.5, 2,by = 0.05)`; longer vectors take more computation time

cstart
: initial value for root-solving algorithm (see multiStart for method = "Fleishman" or nleqslv for method = "Polynomial"). If user-specified, must be input as a matrix. If NULL and all 4 standardized cumulants (rounded to 3 digits) are within 0.01 of those in Headrick's common distribution table (see Headrick.dist data), uses his constants as starting values; else, generates n sets of random starting values from uniform distributions.

n
: the number of initial starting values to use with root-solver. More starting values require more calculation time (default = 25).

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

Value

A list with components:

constants a vector of valid or invalid power method solutions, c("c0","c1","c2","c3") for method = "Fleishman" or c("c0","c1","c2","c3","c4","c5") for method = "Polynomial"

valid "TRUE" if the constants produce a valid power method pdf, else "FALSE"

SixCorr1 if Six is specified, the sixth cumulant correction required to achieve a valid pdf

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. Possible solutions include: a) increasing the number of initial starting values (n), b) using a different seed, or c) specifying a Six vector of sixth cumulant correction values (for method = "Polynomial"). 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)`). Due to the nature of the integration involved in calc_theory, the results are approximations. Greater accuracy can be achieved by increasing the number of subdivisions (sub) used in the integration process. For example, in order to ensure that skew is exactly 0 for symmetric distributions.

2) In addition, 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

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

Hasselman B (2018). nleqslv: Solve Systems of Nonlinear Equations. R package version 3.3.2. https://CRAN.R-project.org/package=nleqslv

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.

Varadhan R, Gilbert P (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, http://www.jstatsoft.org/v32/i04/

See Also

multiStart, nleqslv, fleish, poly, power_norm_corr, pdf_check

Examples

```
# Exponential Distribution
find_constants("Fleishman", 2, 6)

## Not run:
# Compute third-order power method constants.

options(scipen = 999) # turn off scientific notation

# Laplace Distribution
find_constants("Fleishman", 0, 3)

# Compute fifth-order power method constants.

# Logistic Distribution
find_constants(method = "Polynomial", skews = 0, skurts = 6/5, fifths = 0,
               sixths = 48/7)

# with correction to sixth cumulant
find_constants(method = "Polynomial", skews = 0, skurts = 6/5, fifths = 0,
               sixths = 48/7, Six = seq(1.7, 2, by = 0.01))
```

```
## End(Not run)
```

---

fleish *Fleishman's Third-Order Polynomial Transformation Equations*

Description

This function contains Fleishman's third-order polynomial transformation equations (doi: 10.1007/BF02293811). It is used in find_constants to find the constants c1, c2, and c3 (c0 = -c2) that satisfy the equations given skewness and standardized kurtosis values. It can be used to verify a set of constants satisfy the equations. Note that there exist solutions that yield invalid power method pdfs (see power_norm_corr, pdf_check). This function would not ordinarily be called by the user.

Usage

```
fleish(c, a)
```

Arguments

c	a vector of constants c1, c2, c3; note that find_constants returns c0, c1, c2, c3
a	a vector c(skewness, standardized kurtosis)

Value

a list of length 3; if the constants satisfy the equations, returns 0 for all list elements

References

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

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

See Also

poly, power_norm_corr, pdf_check, find_constants

Examples

```
# Laplace Distribution
fleish(c = c(0.782356, 0, 0.067905), a = c(0, 3))
```

fleish_Hessian *Fleishman's Third-Order Transformation Hessian Calculation for Lower Boundary of Standardized Kurtosis in Asymmetric Distributions*

Description

This function gives the second-order conditions necessary to verify that a kurtosis is a global minimum. A kurtosis solution from `fleish_skurt_check` is a global minimum if and only if the determinant of the bordered Hessian, H, is less than zero (see Headrick & Sawilowsky, 2002, doi: 10.3102/10769986025004417), where

$$|\bar{H}| = matrix(c(0, dg(c1, c3)/dc1, dg(c1, c3)/dc3,$$

$$dg(c1, c3)/dc1, d^2F(c1, c3, \lambda)/dc1^2, d^2F(c1, c3, \lambda)/(dc3dc1),$$

$$dg(c1, c3)/dc3, d^2F(c1, c3, \lambda)/(dc1dc3), d^2F(c1, c3, \lambda)/dc3^2), 3, 3, byrow = TRUE)$$

Here, $F(c1, c3, \lambda) = f(c1, c3) + \lambda * [\gamma_1 - g(c1, c3)]$ is the Fleishman Transformation Lagrangean expression (see `fleish_skurt_check`). Headrick & Sawilowsky (2002) gave equations for the second-order derivatives $d^2F/dc1^2$, $d^2F/dc3^2$, and $d^2F/(dc1dc3)$. These were verified and $dg/dc1$ and $dg/dc3$ were calculated using `D` (see `deriv`). This function would not ordinarily be called by the user.

Usage

```
fleish_Hessian(c)
```

Arguments

`c`	a vector of constants c1, c3, lambda

Value

A list with components:

`Hessian` the Hessian matrix H

`H_det` the determinant of H

References

Please see references for `fleish_skurt_check`.

See Also

`fleish_skurt_check`, `calc_lower_skurt` fleish_skurt_check *Fleishman's Third-Order Transformation Lagrangean Constraints for Lower Boundary of Standardized Kurtosis in Asymmetric Distributions*

Description

This function gives the first-order conditions of the Fleishman Transformation Lagrangean expression $F(c1, c3, \lambda) = f(c1, c3) + \lambda * [\gamma_1 - g(c1, c3)]$ used to find the lower kurtosis boundary for a given non-zero skewness in `calc_lower_skurt` (see Headrick & Sawilowsky, 2002, doi: 10.3102/10769986025004417). Here, $f(c1, c3)$ is the equation for standardized kurtosis expressed in terms of c1 and c3 only, λ is the Lagrangean multiplier, γ_1 is skewness, and $g(c1, c3)$ is the equation for skewness expressed in terms of c1 and c3 only. It should be noted that these equations are for $\gamma_1 > 0$. Negative skew values are handled within `calc_lower_skurt`. Headrick & Sawilowsky (2002) gave equations for the first-order derivatives $dF/dc1$ and $dF/dc3$. These were verified and $dF/d\lambda$ was calculated using `D` (see `deriv`). The second-order conditions to verify that the kurtosis is a global minimum are in `fleish_Hessian`. This function would not ordinarily be called by the user.

Usage

```
fleish_skurt_check(c, a)
```

Arguments

`c`	a vector of constants c1, c3, lambda
`a`	skew value

Value

A list with components:

$dF(c1, c3, \lambda)/d\lambda = \gamma_1 - g(c1, c3)$

$dF(c1, c3, \lambda)/dc1 = df(c1, c3)/dc1 - \lambda * dg(c1, c3)/dc1$

$dF(c1, c3, \lambda)/dc3 = df(c1, c3)/dc3 - \lambda * dg(c1, c3)/dc3$

If the suppled values for c and skew satisfy the Lagrangean expression, it will return 0 for each component.

References

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

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.

See Also

fleish_Hessian, calc_lower_skurt

Headrick.dist — *Examples of Constants Calculated by Headrick's Fifth-Order Polynomial Transformation*

Description

Selected symmetrical and asymmetrical theoretical densities with their associated values of skewness (gamma1), standardized kurtosis (gamma2), and standardized fifth (gamma3) and sixth (gamma4) cumulants. Constants were calculated by Headrick using his fifth-order polynomial transformation and given in his Table 1 (2002, p. 691-2, doi: 10.1016/S01679473(02)000725). Note that the standardized cumulants for the Gamma(10, 10) distribution do not arise from using $\alpha = 10,\ \beta = 10$. Therefore, either there is a typo in the table or Headrick used a different parameterization.

Usage

```
data(Headrick.dist)
```

Format

An object of class "data.frame"; Colnames are distribution names; rownames are standardized cumulant names followed by c0, ..., c5.

References

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)

Examples

```
z <- rnorm(10000)
g <- Headrick.dist$Gamma_a10b10[-c(1:4)]
gamma_a10b10 <- g[1] + g[2] * z + g[3] * z^2 + g[4] * z^3 + g[5] * z^4 +
                g[6] * z^5
summary(gamma_a10b10)
```

`H_params` *Parameters for Examples of Constants Calculated by Headrick's Fifth-Order Polynomial Transformation*

Description

These are the parameters for `Headrick.dist`, which contains selected symmetrical and asymmetrical theoretical densities with their associated values of skewness (gamma1), standardized kurtosis (gamma2), and standardized fifth (gamma3) and sixth (gamma4) cumulants. Constants were calculated by Headrick using his fifth-order polynomial transformation and given in his Table 1 (2002, p. 691-2, doi: 10.1016/S01679473(02)000725). Note that the standardized cumulants for the Gamma(10, 10) distribution do not arise from using $\alpha = 10,\ \beta = 10$. Therefore, either there is a typo in the table or Headrick used a different parameterization.

Usage

```
data(H_params)
```

Format

An object of class "data.frame"; Colnames are distribution names as inputs for `calc_theory`; rownames are param1, param2.

References

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)

`intercorr_fleish` *Fleishman's Third-Order Polynomial Transformation Intermediate Correlation Equations*

Description

This function contains Fleishman's third-order polynomial transformation intermediate correlation equations (Headrick & Sawilowsky, 1999, doi: 10.1007/BF02294317). It is used in `findintercorr` and `findintercorr2` to find the intermediate correlation for standard normal random variables which are used in the Fleishman polynomial transformation. It can be used to verify a set of constants and an intermediate correlation satisfy the equations for the desired post-transformation correlation. It works for two or three variables. Headrick & Sawilowsky recommended using the technique of Vale & Maurelli (1983, doi: 10.1007/BF02293687), in the case of more than 3 variables, in which the intermediate correlations are found pairwise and then eigen value decomposition is used on the correlation matrix. Note that there exist solutions that yield invalid power method pdfs (see `power_norm_corr`, `pdf_check`). This function would not ordinarily be called by the user.

Usage

```
intercorr_fleish(r, c, a)
```

Arguments

r	either a scalar, in which case it represents pairwise intermediate correlation between standard normal variables, or a vector of 3 values, in which case: $$r[1] * r[2] = \rho_{z1,z2},\ r[1] * r[3] = \rho_{z1,z3},\ r[2] * r[3] = \rho_{z2,z3}$$
c	a matrix with either 2 or 3 rows, each a vector of constants c0, c1, c2, c3, like that returned by `find_constants`
a	a matrix of target correlations among continuous variables; if `nrow(a) = 1`, it represents a pairwise correlation; if `nrow(a) = 2 or 3`, it represents a correlation matrix between two or three variables

Value

a list of length 1 for pairwise correlations or length 3 for three variables; if the inputs satisfy the equations, returns 0 for all list elements

References

Please see references for `findintercorr_cont`.

See Also

`fleish`, `power_norm_corr`, `pdf_check`, `find_constants`

---

intercorr_poly *Headrick's Fifth-Order Polynomial Transformation Intermediate Correlation Equations*

Description

This function contains Headrick's fifth-order polynomial transformation intermediate correlation equations (2002, doi: 10.1016/S01679473(02)000725). It is used in `findintercorr` and `findintercorr2` to find the intermediate correlation for standard normal random variables which are used in the Headrick polynomial transformation. It can be used to verify a set of constants and an intermediate correlation satisfy the equations for the desired post-transformation correlation. It works for two, three, or four variables. Headrick recommended using the technique of Vale & Maurelli (1983, doi: 10.1007/BF02293687), in the case of more than 4 variables, in which the intermediate correlations are found pairwise and then eigen value decomposition is used on the correlation matrix. Note that there exist solutions that yield invalid power method pdfs (see `power_norm_corr`, `pdf_check`). This function would not ordinarily be called by the user.

Usage

```
intercorr_poly(r, c, a)
```

Arguments

`r`	either a scalar, in which case it represents pairwise intermediate correlation between standard normal variables, or a vector of 3 values, in which case: $$r[1]*r[2] = \rho_{z1,z2},\ r[1]*r[3] = \rho_{z1,z3},\ r[2]*r[3] = \rho_{z2,z3}$$ or a vector of 4 values, in which case: $$r0 = r[5]*r[6],\ r0*r[1]*r[2] = \rho_{z1,z2},\ r0*r[1]*r[3] = \rho_{z1,z3}$$ $$r0*r[2]*r[3] = \rho_{z2,z3},\ r0*r[1]*r[4] = \rho_{z1,z4},\ r0*r[2]*r[4] = \rho_{z2,z4},$$ $$r0*r[3]*r[4] = \rho_{z3,z4}$$
`c`	a matrix with either 2, 3, or 4 rows, each a vector of constants c0, c1, c2, c3, like that returned by `find_constants`
`a`	a matrix of target correlations among continuous variables; if `nrow(a) = 1`, it represents a pairwise correlation; if `nrow(a) = 2, 3, or 4`, it represents a correlation matrix between two, three, or four variables

Value

a list of length 1 for pairwise correlations, length 3 for three variables, or length 6 for four variables; if the inputs satisfy the equations, returns 0 for all list elements

References

Please see references for `findintercorr_cont`.

See Also

`poly`, `power_norm_corr`, `pdf_check`, `find_constants`

`max_count_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 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 values pois_eps and nb_eps may differ for each variable. The function is used in `findintercorr2` and `rcorrvar2`. This function would not ordinarily be called by the user.

Usage

```
max_count_support(k_pois, k_nb, lam, pois_eps = NULL, size, prob, mu = 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 (> 0) constants for the Poisson variables (see Poisson)
pois_eps	a vector of length k_pois containing the truncation values (i.e. = rep(0.0001, k_pois); default = NULL)
size	a vector of size parameters for the Negative Binomial variables (see NegBinomial)
prob	a vector of success probability parameters
mu	a vector of mean parameters (*Note: either prob or mu should be supplied for all Negative Binomial variables, not a mixture; default = NULL)
nb_eps	a vector of length k_nb containing the truncation values (i.e. = rep(0.0001, k_nb); default = NULL)

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.

See Also

findintercorr2, rcorrvar2

nonnormvar1 *Generation of One Non-Normal Continuous Variable Using the Power Method*

Description

This function simulates one non-normal continuous variable using either Fleishman's Third-Order (`method` = "Fleishman", doi: 10.1007/BF02293811) or Headrick's Fifth-Order (`method` = "Polynomial", doi: 10.1016/S01679473(02)000725) Polynomial Transformation. If only one variable is desired and that variable is continuous, this function should be used. The power method transformation is a computationally efficient algorithm that simulates continuous distributions through the method of moments. 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:

$Y = c_0 + c_1 * Z + c_2 * Z^2 + c_3 * Z^3 + c_4 * Z^4 + c_5 * Z^5$,

where $Z\ N(0, 1)$, and c_4 and c_5 both equal 0 for Fleishman's method. The real constants are calculated by `find_constants`. All variables are simulated with mean 0 and variance 1, and then transformed to the specified mean and variance at the end.

The required parameters for simulating continuous variables include: mean, variance, skewness, standardized kurtosis (kurtosis - 3), and standardized fifth and sixth cumulants (for `method` = "Polynomial"). If the goal is to simulate a theoretical distribution (i.e. Gamma, Beta, Logistic, etc.), these values can be obtained using `calc_theory`. If the goal is to mimic an empirical data set, these values can be found using `calc_moments` (using the method of moments) or `calc_fisherk` (using Fisher's k-statistics). 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)`). Due to the nature of the integration involved in `calc_theory`, the results are approximations. Greater accuracy can be achieved by increasing the number of subdivisions (`sub`) used in the integration process. For example, in order to ensure that skew is exactly 0 for symmetric distributions.

For some sets of cumulants, it is either not possible to find power method constants or the calculated constants do not generate valid power method pdfs. In these situations, adding a value to the sixth cumulant may provide solutions (see `find_constants`). If simulation results indicate that a continuous variable does not generate a valid pdf, the user can try `find_constants` with various sixth cumulant correction vectors to determine if a valid pdf can be found.

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 example and the **Comparison of Simulated Distribution to Theoretical Distribution or Empirical Data** vignette.

Usage

```
nonnormvar1(method = c("Fleishman", "Polynomial"), means = 0, vars = 1,
  skews = 0, skurts = 0, fifths = 0, sixths = 0, Six = NULL,
  cstart = NULL, n = 10000, seed = 1234)
```

Arguments

method	the method used to generate the continuous variable. "Fleishman" uses Fleishman's third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.
means	mean for the continuous variable (default = 0)
vars	variance (default = 1)
skews	skewness value (default = 0)
skurts	standardized kurtosis (kurtosis - 3, so that normal variables have a value of 0; default = 0)
fifths	standardized fifth cumulant (not necessary for method = "Fleishman"; default = 0)
sixths	standardized sixth cumulant (not necessary for method = "Fleishman"; default = 0)
Six	a vector of correction values to add to the sixth cumulant if no valid pdf constants are found, ex: `Six = seq(0.01, 2, by = 0.01)`; if no correction is desired, set `Six = NULL` (default)
cstart	initial values for root-solving algorithm (see multiStart for method = "Fleishman" or nleqslv for method = "Polynomial"). If user specified, must be input as a matrix. If NULL and all 4 standardized cumulants (rounded to 3 digits) are within 0.01 of those in Headrick's common distribution table (see Headrick.dist data), uses his constants as starting values; else, generates n sets of random starting values from uniform distributions.
n	the sample size (i.e. the length of the simulated variable; default = 10000)
seed	the seed value for random number generation (default = 1234)

Value

A list with the following components:

constants a data.frame of the constants

continuous_variable a data.frame of the generated continuous variable

summary_continuous a data.frame containing a summary of the variable

summary_targetcont a data.frame containing a summary of the target variable

sixth_correction the sixth cumulant correction value

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

Constants_Time the time in minutes required to calculate the constants

Simulation_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 $\gamma_3^2/\gamma_4 = 2/3$. However, if the fifth and sixth cumulants do not exist, the Fleishman approximation should be used.

Overview of Simulation Process

1) The constants are calculated for the continuous 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 `Six` vector of sixth cumulant correction values (if `method` = "Polynomial"). Errors regarding constant calculation are the most probable cause of function failure.

2) An intermediate standard normal variate X of length n is generated.

3) Summary statistics are calculated.

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 vector of sixth cumulant correction values 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)`).

2) In addition, 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

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

find_constants

Examples

```
# Normal distribution with Headrick's fifth-order PMT:
N <- nonnormvar1("Polynomial", 0, 1, 0, 0, 0, 0)

## Not run:
# Use Headrick & Kowalchuk's (2007) steps to compare a simulated exponential
# (mean = 2) variable to the theoretical exponential(mean = 2) density:

# 1) Obtain the standardized cumulants:
stcums <- calc_theory(Dist = "Exponential", params = 0.5) # rate = 1/mean

# 2) Simulate the variable:
H_exp <- nonnormvar1("Polynomial", means = 2, vars = 2, skews = stcums[3],
                    skurts = stcums[4], fifths = stcums[5],
                    sixths = stcums[6], n = 10000, seed = 1234)

H_exp$constants
#           c0        c1       c2         c3          c4           c5
# 1 -0.3077396 0.8005605 0.318764 0.03350012 -0.00367481 0.0001587076

# 3) Determine whether the constants produce a valid power method pdf:

H_exp$valid.pdf
# [1] "TRUE"

# 4) Select a critical value:

# Let alpha = 0.05.
y_star <- qexp(1 - 0.05, rate = 0.5) # note that rate = 1/mean
y_star
# [1] 5.991465

# 5) Solve m_{2}^{1/2} * p(z') + m_{1} - y* = 0 for z', where m_{1} and
# m_{2} are the 1st and 2nd moments of Y*:

# The exponential(2) distribution has a mean and standard deviation equal
# to 2.
# Solve 2 * p(z') + 2 - y_star = 0 for z'
# p(z') = c0 + c1 * z' + c2 * z'^2 + c3 * z'^3 + c4 * z'^4 + c5 * z'^5

f_exp <- function(z, c, y) {
  return(2 * (c[1] + c[2] * z + c[3] * z^2 + c[4] * z^3 + c[5] * z^4 +
            c[6] * z^5) + 2 - y)
}

z_prime <- uniroot(f_exp, interval = c(-1e06, 1e06),
                   c = as.numeric(H_exp$constants), y = y_star)$root
z_prime
```

```
# [1] 1.644926

# 6) Calculate 1 - Phi(z'), the corresponding probability for the
# approximation Y to Y* (i.e. 1 - Phi(z') = 0.05), and compare to target
# value alpha:

1 - pnorm(z_prime)
# [1] 0.04999249

# 7) Plot a parametric graph of Y* and Y:

plot_sim_pdf_theory(sim_y = as.numeric(H_exp$continuous_variable[, 1]),
                    Dist = "Exponential", params = 0.5)

# Note we can also plot the empirical cdf and show the cumulative
# probability up to y_star:

plot_sim_cdf(sim_y = as.numeric(H_exp$continuous_variable[, 1]),
             calc_cprob = TRUE, delta = y_star)


## End(Not run)
```

---

ordnorm *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 >= 2 categories) variables or variables that are treated as ordinal (i.e. count variables in the Barbiero & Ferrari, 2015 method used in `rcorrvar2`, 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 `contord` 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 it is assumed the user has done this before simulation using `valid_corr` or `valid_corr2`

2) the final positive-definite check has been removed

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

4) a final "fail-safe" check was added at the end of the iteration loop where if the absolute error between the final and target pairwise correlation is still > 0.1, the intermediate correlation is set equal to the target correlation.

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 `rcorrvar` and `rcorrvar2` using the method of Higham (2002) and the `nearPD` function.

Usage

```
ordnorm(marginal, rho, support = list(), epsilon = 0.001, maxit = 1000)
```

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 correlation matrices (default = 0.001); smaller epsilons 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

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

`ordcont`, `rcorrvar`, `rcorrvar2`, `findintercorr`, `findintercorr2`

pdf_check — *Check Polynomial Transformation Constants for Valid Power Method PDF*

Description

This function determines if a given set of constants, calculated using Fleishman's Third-Order (method = "Fleishman", doi: 10.1007/BF02293811) or Headrick's Fifth-Order (method = "Polynomial", doi: 10.1016/S01679473(02)000725) Polynomial Transformation, yields a valid pdf. This requires 1) the correlation between the resulting continuous variable and the underlying standard normal variable (see power_norm_corr) is > 0, and 2) the constants satisfy certain constraints (see Headrick & Kowalchuk, 2007, doi: 10.1080/10629360600605065).

Usage

```
pdf_check(c, method)
```

Arguments

c	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
method	the method used to find the constants. "Fleishman" uses a third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.

Value

A list with components:

rho_pZ the correlation between the continuous variable and the underlying standard normal variable

valid.pdf "TRUE" if the constants produce a valid power method pdf, else "FALSE"

References

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

fleish, poly, power_norm_corr, find_constants

Examples

```
# Normal distribution
pdf_check(c(0, 1, 0, 0, 0, 0), "Polynomial")

## Not run:
# Chi-squared (df = 1) Distribution (invalid power method pdf)
con <- find_constants(method = "Polynomial", skews = sqrt(8), skurts = 12,
                      fifths = 48*sqrt(2), sixths = 480)$constants
pdf_check(c = con, method = "Polynomial")

# Beta (a = 4, b = 2) Distribution (valid power method pdf)
con <- find_constants(method = "Polynomial", skews = -0.467707,
                      skurts = -0.375, fifths = 1.403122,
                      sixths = -0.426136)$constants
pdf_check(c = con, method = "Polynomial")


## End(Not run)
```

---

plot_cdf *Plot Theoretical Power Method Cumulative Distribution Function for Continuous Variables*

Description

This plots the theoretical power method cumulative distribution function:

$$F_p(Z)(p(z)) = F_p(Z)(p(z), F_Z(z)),$$

as given in Headrick & Kowalchuk (2007, doi: 10.1080/10629360600605065). It is a parametric plot with $sigma * y + mu$, where $y = p(z)$, on the x-axis and $F_Z(z)$ on the y-axis, where z is vector of n random standard normal numbers (generated with a seed set by user). Given a vector of polynomial transformation constants, the function generates $sigma * y + mu$ and calculates the theoretical cumulative probabilities using $F_p(Z)(p(z), F_Z(z))$. If `calc_cprob` = TRUE, the cumulative probability up to $delta = sigma * y + mu$ is calculated (see `cdf_prob`) and the region on the plot is filled with a dashed horizontal line drawn at $F_p(Z)(delta)$. The cumulative probability is stated on top of the line. It returns a `ggplot2-package` object so the user can modify as necessary. The graph parameters (i.e. `title`, `color`, `fill`, `hline`) are `ggplot2-package` parameters. It works for valid or invalid power method pdfs.

Usage

```
plot_cdf(c = NULL, method = c("Fleishman", "Polynomial"), mu = 0,
  sigma = 1, title = "Cumulative Distribution Function", ylower = NULL,
  yupper = NULL, calc_cprob = FALSE, delta = 5, color = "dark blue",
  fill = "blue", hline = "dark green", n = 10000, seed = 1234,
  text.size = 11, title.text.size = 15, axis.text.size = 10,
  axis.title.size = 13, lower = -1000000, upper = 1000000)
```

Arguments

c	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
method	the method used to generate the continuous variable $y = p(z)$. "Fleishman" uses Fleishman's third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.
mu	mean for the continuous variable (default = 0)
sigma	standard deviation for the continuous variable (default = 1)
title	the title for the graph (default = "Cumulative Distribution Function")
ylower	the lower y value to use in the plot (default = NULL, uses minimum simulated y value)
yupper	the upper y value (default = NULL, uses maximum simulated y value)
calc_cprob	if TRUE (default = FALSE), cdf_prob is used to find the cumulative probability up to $delta = sigma * y + mu$ and the region on the plot is filled with a dashed horizontal line drawn at $F_p(Z)(delta)$
delta	the value $sigma * y + mu$, where $y = p(z)$, at which to evaluate the cumulative probability
color	the line color for the cdf (default = "dark blue")
fill	the fill color if calc_cprob = TRUE (default = "blue)
hline	the dashed horizontal line color drawn at delta if calc_cprob = TRUE (default = "dark green")
n	the number of random standard normal numbers to use in generating $y = p(z)$ (default = 10000)
seed	the seed value for random number generation (default = 1234)
text.size	the size of the text displaying the cumulative probability up to delta if calc_cprob = TRUE
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
lower	lower bound for cdf_prob
upper	upper bound for cdf_prob

Value

A ggplot2-package object.

References

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.

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

See Also

find_constants, cdf_prob, ggplot2-package, geom_path, geom_abline, geom_ribbon

Examples

```
## Not run:
# Logistic Distribution: mean = 0, sigma = 1

# Find standardized cumulants
stcum <- calc_theory(Dist = "Logistic", params = c(0, 1))

# Find constants without the sixth cumulant correction
# (invalid power method pdf)
con1 <- find_constants(method = "Polynomial", skews = stcum[3],
                       skurts = stcum[4], fifths = stcum[5],
                       sixths = stcum[6], n = 25, seed = 1234)

# Plot cdf with cumulative probability calculated up to delta = 5
plot_cdf(c = con1$constants, method = "Polynomial",
         title = "Invalid Logistic CDF", calc_cprob = TRUE, delta = 5)

# Find constants with the sixth cumulant correction
# (valid power method pdf)
con2 <- find_constants(method = "Polynomial", skews = stcum[3],
```

```
                    skurts = stcum[4], fifths = stcum[5],
                    sixths = stcum[6], Six = seq(1.5, 2, 0.05))

# Plot cdf with cumulative probability calculated up to delta = 5
plot_cdf(c = con2$constants, method = "Polynomial",
         title = "Valid Logistic CDF", calc_cprob = TRUE, delta = 5)

## End(Not run)
```

---

plot_pdf_ext *Plot Theoretical Power Method Probability Density Function and Target PDF of External Data for Continuous Variables*

Description

This plots the theoretical power method probability density function:

$$f_p(Z)(p(z)) = f_p(Z)(p(z), f_Z(z)/p'(z)),$$

as given in Headrick & Kowalchuk (2007, doi: 10.1080/10629360600605065), and target pdf. It is a parametric plot with $sigma * y + mu$, where $y = p(z)$, on the x-axis and $f_Z(z)/p'(z)$ on the y-axis, where z is vector of n random standard normal numbers (generated with a seed set by user; length equal to length of external data vector). `sigma` is the standard deviation and `mu` is the mean of the external data set. Given a vector of polynomial transformation constants, the function generates $sigma * y + mu$ and calculates the theoretical probabilities using $f_p(Z)(p(z), f_Z(z)/p'(z))$. The target distribution is also plotted given a vector of external data. This external data is required. The y values are centered and scaled to have the same mean and standard deviation as the external data. If the user wants to only plot the power method pdf, `plot_pdf_theory` should be used instead with `overlay = FALSE`. It returns a `ggplot2-package` object so the user can modify as necessary. The graph parameters (i.e. `title`, `power_color`, `target_color`, `nbins`) are `ggplot2-package` parameters. It works for valid or invalid power method pdfs.

Usage

```
plot_pdf_ext(c = NULL, method = c("Fleishman", "Polynomial"),
  title = "Probability Density Function", ylower = NULL, yupper = NULL,
  power_color = "dark blue", ext_y = NULL, target_color = "dark green",
  target_lty = 2, seed = 1234, 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

c	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`
method	the method used to generate the continuous variable y = p(z). "Fleishman" uses Fleishman's third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.

`title`	the title for the graph (default = "Probability Density Function")
`ylower`	the lower y value to use in the plot (default = NULL, uses minimum simulated y value)
`yupper`	the upper y value (default = NULL, uses maximum simulated y value)
`power_color`	the line color for the power method pdf (default = "dark blue")
`ext_y`	a vector of external data (required)
`target_color`	the histogram color for the target pdf (default = "dark green")
`target_lty`	the line type for the target pdf (default = 2, dashed line)
`seed`	the seed value for random number generation (default = 1234)
`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_cdf`.

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

See Also

`find_constants`, `calc_theory`, `ggplot2-package`, `geom_path`, `geom_density`

Examples

```
## Not run:
# Logistic Distribution

seed = 1234

# Simulate "external" data set
set.seed(seed)
ext_y <- rlogis(10000)

# Find standardized cumulants
```

```
stcum <- calc_theory(Dist = "Logistic", params = c(0, 1))

# Find constants without the sixth cumulant correction
# (invalid power method pdf)
con1 <- find_constants(method = "Polynomial", skews = stcum[3],
                      skurts = stcum[4], fifths = stcum[5],
                      sixths = stcum[6])

# Plot invalid power method pdf with external data
plot_pdf_ext(c = con1$constants, method = "Polynomial",
             title = "Invalid Logistic PDF", ext_y = ext_y,
             seed = seed)

# Find constants with the sixth cumulant correction
# (valid power method pdf)
con2 <- find_constants(method = "Polynomial", skews = stcum[3],
                      skurts = stcum[4], fifths = stcum[5],
                      sixths = stcum[6], Six = seq(1.5, 2, 0.05))

# Plot invalid power method pdf with external data
plot_pdf_ext(c = con2$constants, method = "Polynomial",
             title = "Valid Logistic PDF", ext_y = ext_y,
             seed = seed)

## End(Not run)
```

`plot_pdf_theory` *Plot Theoretical Power Method Probability Density Function and Target PDF by Distribution Name or Function for Continuous Variables*

Description

This plots the theoretical power method probability density function:

$$f_p(Z)(p(z)) = f_p(Z)(p(z), f_Z(z)/p'(z)),$$

as given in Headrick & Kowalchuk (2007, doi: 10.1080/10629360600605065), and target pdf (if overlay = TRUE). It is a parametric plot with $sigma * y + mu$, where $y = p(z)$, on the x-axis and $f_Z(z)/p'(z)$ on the y-axis, where z is vector of n random standard normal numbers (generated with a seed set by user). Given a vector of polynomial transformation constants, the function generates $sigma * y + mu$ and calculates the theoretical probabilities using $f_p(Z)(p(z), f_Z(z)/p'(z))$. If `overlay` = TRUE, the target distribution is also plotted given either a distribution name (plus up to 4 parameters) or a pdf function fx. If a target distribution is specified, y is scaled and then transformed so that it has the same mean and variance as the target distribution. It returns a `ggplot2-package` object so the user can modify as necessary. The graph parameters (i.e. `title`, `power_color`, `target_color`, `target_lty`) are `ggplot2-package` parameters. It works for valid or invalid power method pdfs.

Usage

```
plot_pdf_theory(c = NULL, method = c("Fleishman", "Polynomial"), mu = 0,
  sigma = 1, title = "Probability Density Function", ylower = NULL,
  yupper = NULL, power_color = "dark blue", overlay = 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"), params = NULL, fx = NULL, lower = NULL, upper = NULL,
  n = 100, seed = 1234, 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

`c`	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`
`method`	the method used to generate the continuous variable $y = p(z)$. "Fleishman" uses Fleishman's third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.
`mu`	the desired mean for the continuous variable (used if `overlay = FALSE`, otherwise variable centered to have the same mean as the target distribution)
`sigma`	the desired standard deviation for the continuous variable (used if `overlay = FALSE`, otherwise variable scaled to have the same standard deviation as the target distribution)
`title`	the title for the graph (default = "Probability Density Function")
`ylower`	the lower y value to use in the plot (default = NULL, uses minimum simulated y value)
`yupper`	the upper y value (default = NULL, uses maximum simulated y value)
`power_color`	the line color for the power method pdf (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 bounds = ylower, yupper)
`target_color`	the line color for the target pdf (default = "dark green")
`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". 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)
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
n	the number of random standard normal numbers to use in generating $y = p(z)$ (default = 100)
seed	the seed value for random number generation (default = 1234)
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_cdf.

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

See Also

find_constants, calc_theory, ggplot2-package, geom_path

Examples

```
## Not run:
# Logistic Distribution

# Find standardized cumulants
stcum <- calc_theory(Dist = "Logistic", params = c(0, 1))

# Find constants without the sixth cumulant correction
# (invalid power method pdf)
con1 <- find_constants(method = "Polynomial", skews = stcum[3],
                       skurts = stcum[4], fifths = stcum[5],
                       sixths = stcum[6])
```

```
# Plot invalid power method pdf with theoretical pdf overlayed
plot_pdf_theory(c = con1$constants, method = "Polynomial",
         title = "Invalid Logistic PDF", overlay = TRUE,
         Dist = "Logistic", params = c(0, 1))

# Find constants with the sixth cumulant correction
# (valid power method pdf)
con2 <- find_constants(method = "Polynomial", skews = stcum[3],
                      skurts = stcum[4], fifths = stcum[5],
                      sixths = stcum[6], Six = seq(1.5, 2, 0.05))

# Plot valid power method pdf with theoretical pdf overlayed
plot_pdf_theory(c = con2$constants, method = "Polynomial",
         title = "Valid Logistic PDF", overlay = TRUE,
         Dist = "Logistic", params = c(0, 1))

## End(Not run)
```

---

plot_sim_cdf — *Plot Simulated (Empirical) Cumulative Distribution Function for Continuous, Ordinal, or Count Variables*

Description

This plots the cumulative distribution function of simulated continuous, ordinal, or count data using the empirical cdf Fn (see stat_ecdf). Fn is a step function with jumps i/n at observation values, where i is the number of tied observations at that value. Missing values are ignored. For observations $y = (y1, y2, ..., yn)$, Fn is the fraction of observations less or equal to t, i.e., $Fn(t) = sum[yi <= t]/n$. If calc_cprob = TRUE and the variable is *continuous*, the cumulative probability up to $y = delta$ is calculated (see sim_cdf_prob) and the region on the plot is filled with a dashed horizontal line drawn at Fn(delta). The cumulative probability is stated on top of the line. This fill option does not work for ordinal or count variables. The function returns a ggplot2-package object so the user can modify as necessary. The graph parameters (i.e. title, color, fill, hline) are ggplot2-package parameters. It works for valid or invalid power method pdfs.

Usage

```
plot_sim_cdf(sim_y, title = "Empirical Cumulative Distribution Function",
  ylower = NULL, yupper = NULL, calc_cprob = FALSE, delta = 5,
  color = "dark blue", fill = "blue", hline = "dark green",
  text.size = 11, 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 = "Empirical Cumulative Distribution Function")
`ylower`	the lower y value to use in the plot (default = NULL, uses minimum simulated y value)
`yupper`	the upper y value (default = NULL, uses maximum simulated y value)
`calc_cprob`	if TRUE (default = FALSE) and `sim_y` is continuous, `sim_cdf_prob` is used to find the empirical cumulative probability up to y = delta and the region on the plot is filled with a dashed horizontal line drawn at $Fn(delta)$
`delta`	the value y at which to evaluate the cumulative probability (default = 5)
`color`	the line color for the cdf (default = "dark blue")
`fill`	the fill color if `calc_cprob` = TRUE (default = "blue)
`hline`	the dashed horizontal line color drawn at `delta` if `calc_cprob` = TRUE (default = "dark green")
`text.size`	the size of the text displaying the cumulative probability up to `delta` if `calc_cprob` = TRUE
`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_cdf`.

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

See Also

`ecdf`, `sim_cdf_prob`, `ggplot2-package`, `stat_ecdf`, `geom_abline`, `geom_ribbon`

Examples

```
## Not run:
# Logistic Distribution: mean = 0, variance = 1
seed = 1234

# Find standardized cumulants
stcum <- calc_theory(Dist = "Logistic", params = c(0, 1))

# Simulate without the sixth cumulant correction
# (invalid power method pdf)
```

```
Logvar1 <- nonnormvar1(method = "Polynomial", means = 0, vars = 1,
                      skews = stcum[3], skurts = stcum[4],
                      fifths = stcum[5], sixths = stcum[6], seed = seed)

# Plot cdf with cumulative probability calculated up to delta = 5
plot_sim_cdf(sim_y = Logvar1$continuous_variable,
             title = "Invalid Logistic Empirical CDF",
             calc_cprob = TRUE, delta = 5)

# Simulate with the sixth cumulant correction
# (valid power method pdf)
Logvar2 <- nonnormvar1(method = "Polynomial", means = 0, vars = 1,
                      skews = stcum[3], skurts = stcum[4],
                      fifths = stcum[5], sixths = stcum[6],
                      Six = seq(1.5, 2, 0.05), seed = seed)

# Plot cdf with cumulative probability calculated up to delta = 5
plot_sim_cdf(sim_y = Logvar2$continuous_variable,
             title = "Valid Logistic Empirical CDF",
             calc_cprob = TRUE, delta = 5)

# Simulate one binary and one ordinal variable (4 categories) with
# correlation 0.3
Ordvars = rcorrvar(k_cat = 2, marginal = list(0.4, c(0.2, 0.5, 0.7)),
                   rho = matrix(c(1, 0.3, 0.3, 1), 2, 2), seed = seed)

# Plot cdf of 2nd variable
plot_sim_cdf(Ordvars$ordinal_variables[, 2])


## End(Not run)
```

---

plot_sim_ext — *Plot Simulated Data and Target External Data for Continuous or Count Variables*

Description

This plots simulated continuous or count data and overlays external data, both as histograms. The external data is a required input. The simulated data is centered and scaled to have the same mean and variance as the external data set. If the user wants to only plot simulated data, `plot_sim_theory` should be used instead with `overlay = FALSE`. It returns a `ggplot2-package` object so the user can modify as necessary. The graph parameters (i.e. `title`, `power_color`, `target_color`, `nbins`) are `ggplot2-package` parameters. It works for valid or invalid power method pdfs.

Usage

```
plot_sim_ext(sim_y, title = "Simulated Data Values", ylower = NULL,
```

```
yupper = NULL, power_color = "dark blue", ext_y = NULL,
target_color = "dark green", nbins = 100, 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)

`yupper` the upper y value (default = NULL, uses maximum simulated y value)

`power_color` the histogram fill color for the simulated variable (default = "dark blue")

`ext_y` a vector of external data (required)

`target_color` the histogram fill color for the target data (default = "dark green")

`nbins` the number of bins to use in generating the histograms (default = 100)

`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_cdf`.

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

See Also

`ggplot2-package`, `geom_histogram`

Examples

```
## Not run:
# Logistic Distribution: mean = 0, variance = 1

seed = 1234

# Simulate "external" data set
set.seed(seed)
ext_y <- rlogis(10000)

# Find standardized cumulants
stcum <- calc_theory(Dist = "Logistic", params = c(0, 1))

# Simulate without the sixth cumulant correction
# (invalid power method pdf)
Logvar1 <- nonnormvar1(method = "Polynomial", means = 0, vars = 1,
                      skews = stcum[3], skurts = stcum[4],
                      fifths = stcum[5], sixths = stcum[6],
                      n = 10000, seed = seed)

# Plot simulated variable and external data
plot_sim_ext(sim_y = Logvar1$continuous_variable,
             title = "Invalid Logistic Simulated Data Values",
             ext_y = ext_y)

# Simulate with the sixth cumulant correction
# (valid power method pdf)
Logvar2 <- nonnormvar1(method = "Polynomial", means = 0, vars = 1,
                      skews = stcum[3], skurts = stcum[4],
                      fifths = stcum[5], sixths = stcum[6],
                      Six = seq(1.5, 2, 0.05), n = 10000, seed = seed)

# Plot simulated variable and external data
plot_sim_ext(sim_y = Logvar2$continuous_variable,
             title = "Valid Logistic Simulated Data Values",
             ext_y = ext_y)

# Simulate 2 Poisson distributions (means = 10, 15) and correlation 0.3
# using Method 1
Pvars <- rcorrvar(k_pois = 2, lam = c(10, 15),
                  rho = matrix(c(1, 0.3, 0.3, 1), 2, 2), seed = seed)

# Simulate "external" data set
set.seed(seed)
ext_y <- rpois(10000, 10)

# Plot 1st simulated variable and external data
plot_sim_ext(sim_y = Pvars$Poisson_variable[, 1], ext_y = ext_y)


## End(Not run)
```

plot_sim_pdf_ext *Plot Simulated Probability Density Function and Target PDF of External Data for Continuous or Count Variables*

Description

This plots the pdf of simulated continuous or count data and overlays the target pdf computed from the given external data vector. The external data is a required input. The simulated data is centered and scaled to have the same mean and variance as the external data set. If the user wants to only plot simulated data, `plot_sim_theory` should be used instead (with `overlay = FALSE`). It returns a `ggplot2-package` object so the user can modify as necessary. The graph parameters (i.e. `title`, `power_color`, `target_color`, `target_lty`) are `ggplot2-package` parameters. It works for valid or invalid power method pdfs.

Usage

```
plot_sim_pdf_ext(sim_y, title = "Simulated Probability Density Function",
  ylower = NULL, yupper = NULL, power_color = "dark blue", ext_y = NULL,
  target_color = "dark green", target_lty = 2, 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)
`yupper`	the upper y value (default = NULL, uses maximum simulated y value)
`power_color`	the histogram color for the simulated variable (default = "dark blue")
`ext_y`	a vector of external data (required)
`target_color`	the histogram color for the target pdf (default = "dark green")
`target_lty`	the line type for the target pdf (default = 2, dashed line)
`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_cdf.

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

See Also

ggplot2-package, geom_density

Examples

```
## Not run:
# Logistic Distribution: mean = 0, variance = 1

seed = 1234

# Simulate "external" data set
set.seed(seed)
ext_y <- rlogis(10000)

# Find standardized cumulants
stcum <- calc_theory(Dist = "Logistic", params = c(0, 1))

# Simulate without the sixth cumulant correction
# (invalid power method pdf)
Logvar1 <- nonnormvar1(method = "Polynomial", means = 0, vars = 1,
                      skews = stcum[3], skurts = stcum[4],
                      fifths = stcum[5], sixths = stcum[6],
                      n = 10000, seed = seed)

# Plot pdfs of simulated variable (invalid) and external data
plot_sim_pdf_ext(sim_y = Logvar1$continuous_variable,
                 title = "Invalid Logistic Simulated PDF", ext_y = ext_y)

# Simulate with the sixth cumulant correction
# (valid power method pdf)
Logvar2 <- nonnormvar1(method = "Polynomial", means = 0, vars = 1,
                      skews = stcum[3], skurts = stcum[4],
                      fifths = stcum[5], sixths = stcum[6],
                      Six = seq(1.5, 2, 0.05), n = 10000, seed = 1234)

# Plot pdfs of simulated variable (valid) and external data
plot_sim_pdf_ext(sim_y = Logvar2$continuous_variable,
                 title = "Valid Logistic Simulated PDF", ext_y = ext_y)

# Simulate 2 Poisson distributions (means = 10, 15) and correlation 0.3
# using Method 1
Pvars <- rcorrvar(k_pois = 2, lam = c(10, 15),
```

```
                  rho = matrix(c(1, 0.3, 0.3, 1), 2, 2), seed = seed)

# Simulate "external" data set
set.seed(seed)
ext_y <- rpois(10000, 10)

# Plot pdfs of 1st simulated variable and external data
plot_sim_pdf_ext(sim_y = Pvars$Poisson_variable[, 1], ext_y = ext_y)


## End(Not run)
```

`plot_sim_pdf_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 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. If the variable is Negative Binomial, the parameters must be size and success probability (not mu). The function returns a `ggplot2-package` object so the user can modify as necessary. The graph parameters (i.e. `title`, `power_color`, `target_color`, `target_lty`) are `ggplot2-package` parameters. It works for valid or invalid power method pdfs.

Usage

```
plot_sim_pdf_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)
- `yupper` the upper y value (default = NULL, uses maximum simulated y value)
- `power_color` the line color for the simulated variable
- `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
- `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)
- `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_cdf.

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

See Also

calc_theory, ggplot2-package, geom_path, geom_density

Examples

```
## Not run:
# Logistic Distribution: mean = 0, variance = 1
seed = 1234

# Find standardized cumulants
stcum <- calc_theory(Dist = "Logistic", params = c(0, 1))

# Simulate without the sixth cumulant correction
# (invalid power method pdf)
Logvar1 <- nonnormvar1(method = "Polynomial", means = 0, vars = 1,
                       skews = stcum[3], skurts = stcum[4],
                       fifths = stcum[5], sixths = stcum[6],
                       n = 10000, seed = seed)

# Plot pdfs of simulated variable (invalid) and theoretical distribution
plot_sim_pdf_theory(sim_y = Logvar1$continuous_variable,
                    title = "Invalid Logistic Simulated PDF",
                    overlay = TRUE, Dist = "Logistic", params = c(0, 1))

# Simulate with the sixth cumulant correction
# (valid power method pdf)
Logvar2 <- nonnormvar1(method = "Polynomial", means = 0, vars = 1,
                       skews = stcum[3], skurts = stcum[4],
                       fifths = stcum[5], sixths = stcum[6],
                       Six = seq(1.5, 2, 0.05), n = 10000, seed = seed)

# Plot pdfs of simulated variable (invalid) and theoretical distribution
plot_sim_pdf_theory(sim_y = Logvar2$continuous_variable,
                    title = "Valid Logistic Simulated PDF",
                    overlay = TRUE, Dist = "Logistic", params = c(0, 1))

# Simulate 2 Negative Binomial distributions and correlation 0.3
# using Method 1
NBvars <- rcorrvar(k_nb = 2, size = c(10, 15), prob = c(0.4, 0.3),
                  rho = matrix(c(1, 0.3, 0.3, 1), 2, 2), seed = seed)

# Plot pdfs of 1st simulated variable and theoretical distribution
plot_sim_pdf_theory(sim_y = NBvars$Neg_Bin_variable[, 1], overlay = TRUE,
                    cont_var = FALSE, Dist = "Negative_Binomial",
                    params = c(10, 0.4))
```

```
## End(Not run)
```

---

plot_sim_theory — *Plot Simulated Data and Target Distribution Data by Name or Function for Continuous or Count Variables*

Description

This plots simulated continuous or count data and overlays data (if overlay = TRUE) generated from the target distribution, which is specified by name (plus up to 4 parameters) or pdf function fx (plus support bounds). Due to the integration involved in evaluating the cdf using fx, only continuous fx may be supplied. Both are plotted as histograms. 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. If the variable is Negative Binomial, the parameters must be size and success probability (not mu). It returns a ggplot2-package object so the user can modify as necessary. The graph parameters (i.e. title, power_color, target_color, target_lty) are ggplot2-package parameters. It works for valid or invalid power method pdfs.

Usage

```
plot_sim_theory(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)
yupper	the upper y value (default = NULL, uses maximum simulated y value)
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)

`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

Please see the references for plot_cdf.

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

See Also

calc_theory, ggplot2-package, geom_histogram

Examples

```
## Not run:
# Logistic Distribution: mean = 0, variance = 1
seed = 1234

# Find standardized cumulants
stcum <- calc_theory(Dist = "Logistic", params = c(0, 1))

# Simulate without the sixth cumulant correction
# (invalid power method pdf)
Logvar1 <- nonnormvar1(method = "Polynomial", means = 0, vars = 1,
                       skews = stcum[3], skurts = stcum[4],
                       fifths = stcum[5], sixths = stcum[6],
                       n = 10000, seed = seed)

# Plot simulated variable (invalid) and data from theoretical distribution
plot_sim_theory(sim_y = Logvar1$continuous_variable,
                title = "Invalid Logistic Simulated Data Values",
                overlay = TRUE, Dist = "Logistic", params = c(0, 1),
                seed = seed)

# Simulate with the sixth cumulant correction
# (valid power method pdf)
Logvar2 <- nonnormvar1(method = "Polynomial", means = 0, vars = 1,
                       skews = stcum[3], skurts = stcum[4],
                       fifths = stcum[5], sixths = stcum[6],
                       Six = seq(1.5, 2, 0.05), n = 10000, seed = seed)

# Plot simulated variable (valid) and data from theoretical distribution
plot_sim_theory(sim_y = Logvar2$continuous_variable,
                title = "Valid Logistic Simulated Data Values",
                overlay = TRUE, Dist = "Logistic", params = c(0, 1),
                seed = seed)

# Simulate 2 Negative Binomial distributions and correlation 0.3
# using Method 1
NBvars <- rcorrvar(k_nb = 2, size = c(10, 15), prob = c(0.4, 0.3),
                   rho = matrix(c(1, 0.3, 0.3, 1), 2, 2), seed = seed)

# Plot pdfs of 1st simulated variable and theoretical distribution
plot_sim_theory(sim_y = NBvars$Neg_Bin_variable[, 1], overlay = TRUE,
                cont_var = FALSE, Dist = "Negative_Binomial",
                params = c(10, 0.4))
```

```
## End(Not run)
```

---

poly *Headrick's Fifth-Order Polynomial Transformation Equations*

Description

This function contains Headrick's fifth-order polynomial transformation equations (2002, doi: 10.1016/S01679473(02)000725). It is used in `find_constants` to find the constants c1, c2, c3, c4, and c5 ($c0 = -c2 - 3 * c4$) that satisfy the equations given skewness, standardized kurtosis, and standardized fifth and sixth cumulant values. It can be used to verify a set of constants satisfy the equations. Note that there exist solutions that yield invalid power method pdfs (see `power_norm_corr`, `pdf_check`). This function would not ordinarily be called by the user.

Usage

```
poly(c, a)
```

Arguments

c	a vector of constants c1, c2, c3, c4, c5; note that `find_constants` returns c0, c1, c2, c3, c4, c5
a	a vector c(skewness, standardized kurtosis, standardized fifth cumulant, standardized sixth cumulant)

Value

a list of length 5; if the constants satisfy the equations, returns 0 for all list elements

References

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, 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

fleish, power_norm_corr, pdf_check, find_constants

Examples

```
# Laplace Distribution
poly(c = c(0.727709, 0, 0.096303, 0, -0.002232), a = c(0, 3, 0, 30))
```

---

poly_skurt_check *Headrick's Fifth-Order Transformation Lagrangean Constraints for Lower Boundary of Standardized Kurtosis*

Description

This function gives the first-order conditions of the multi-constraint Lagrangean expression

$$F(c1, ..., c5, \lambda_1, ..., \lambda_4) = f(c1, ..., c5) + \lambda_1 * [1 - g(c1, ..., c5)]$$
$$+\lambda_2 * [\gamma_1 - h(c1, ..., c5)] + \lambda_3 * [\gamma_3 - i(c1, ..., c5)]$$
$$+\lambda_4 * [\gamma_4 - j(c1, ..., c5)]$$

used to find the lower kurtosis boundary for a given skewness and standardized fifth and sixth cumulants in calc_lower_skurt. The partial derivatives are described in Headrick (2002, doi: 10.1016/S01679473(02)000725), but he does not provide the actual equations. The equations used here were found with D (see deriv). Here, $\lambda_1, ..., \lambda_4$ are the Lagrangean multipliers, $\gamma_1, \gamma_3, \gamma_4$ are the user-specified values of skewness, fifth cumulant, and sixth cumulant, and f, g, h, i, j are the equations for standardized kurtosis, variance, fifth cumulant, and sixth cumulant expressed in terms of the constants. This function would not ordinarily be called by the user.

Usage

```
poly_skurt_check(c, a)
```

Arguments

c	a vector of constants c1, ..., c5, lambda1, ..., lambda4
a	a vector of skew, fifth standardized cumulant, sixth standardized cumulant

Value

A list with components:

$dF/d\lambda_1 = 1 - g(c1, ..., c5)$

$dF/d\lambda_2 = \gamma_1 - h(c1, ..., c5)$

$dF/d\lambda_3 = \gamma_3 - i(c1, ..., c5)$

$dF/d\lambda_4 = \gamma_4 - j(c1, ..., c5)$

$dF/dc1 = df/dc1 - \lambda_1 * dg/dc1 - \lambda_2 * dh/dc1 - \lambda_3 * di/dc1 - \lambda_4 * dj/dc1$

$dF/dc2 = df/dc2 - \lambda_1 * dg/dc2 - \lambda_2 * dh/dc2 - \lambda_3 * di/dc2 - \lambda_4 * dj/dc2$

$dF/dc3 = df/dc3 - \lambda_1 * dg/dc3 - \lambda_2 * dh/dc3 - \lambda_3 * di/dc3 - \lambda_4 * dj/dc3$

$dF/dc4 = df/dc4 - \lambda_1 * dg/dc4 - \lambda_2 * dh/dc4 - \lambda_3 * di/dc4 - \lambda_4 * dj/dc4$

$dF/dc5 = df/dc5 - \lambda_1 * dg/dc5 - \lambda_2 * dh/dc5 - \lambda_3 * di/dc5 - \lambda_4 * dj/dc5$

If the suppled values for c and a satisfy the Lagrangean expression, it will return 0 for each component.

References

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, 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

calc_lower_skurt

`power_norm_corr` *Calculate Power Method Correlation*

Description

This function calculates the correlation between a continuous variable, Y1, generated using a third or fifth- order polynomial transformation and the generating standard normal variable, Z1. The power method correlation (described in Headrick & Kowalchuk, 2007, doi: 10.1080/10629360600605065) is given by: $\rho_{y1,z1} = c1 + 3 * c3 + 15 * c5$, where c5 = 0 if method = "Fleishman". A value <= 0 indicates an invalid pdf and the signs of c1 and c3 should be reversed, which could still yield an invalid pdf. All constants should be checked using pdf_check to see if they generate a valid pdf.

Usage

```
power_norm_corr(c, method)
```

Arguments

`c`	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`
`method`	the method used to find the constants. "Fleishman" uses a third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.

Value

A scalar equal to the correlation.

References

Please see references for `pdf_check`.

See Also

`fleish`, `poly`, `find_constants`, `pdf_check`

Examples

```
# Beta(a = 4, b = 2) Distribution
power_norm_corr(c = c(0.108304, 1.104252, -0.123347, -0.045284, 0.005014,
                      0.001285),
                method = "Polynomial")

# Switch signs on c1, c3, and c5 to get negative correlation (invalid pdf):
power_norm_corr(c = c(0.108304, -1.104252, -0.123347, 0.045284, 0.005014,
                      -0.001285),
                method = "Polynomial")
```

`rcorrvar` *Generation of Correlated Ordinal, Continuous, Poisson, and/or Negative Binomial Variables: Correlation Method 1*

Description

This function simulates `k_cat` ordinal, `k_cont` continuous, `k_pois` Poisson, and/or `k_nb` Negative Binomial variables with a specified correlation matrix `rho`. The variables are generated from multivariate normal variables with intermediate correlation matrix `Sigma`, calculated by `findintercorr`, and then transformed. The *ordering* of the variables in `rho` must be *ordinal* (r >= 2 categories), *continuous*, *Poisson*, and *Negative Binomial* (note that it is possible for `k_cat`, `k_cont`, `k_pois`, and/or `k_nb` to be 0). The vignette **Overall Workflow for Data Simulation** provides a detailed example discussing the step-by-step simulation process and comparing correlation methods 1 and 2.

Usage

```
rcorrvar(n = 10000, k_cont = 0, k_cat = 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(), marginal = list(), support = list(), nrand = 100000,
  lam = NULL, size = NULL, prob = NULL, mu = NULL, Sigma = NULL,
  rho = NULL, cstart = NULL, seed = 1234, errorloop = FALSE,
  epsilon = 0.001, maxit = 1000, extra_correct = TRUE)
```

Arguments

`n`	the sample size (i.e. the length of each simulated variable; default = 10000)
`k_cont`	the number of continuous variables (default = 0)
`k_cat`	the number of ordinal (r >= 2 categories) variables (default = 0)
`k_pois`	the number of Poisson variables (default = 0)
`k_nb`	the number of Negative Binomial variables (default = 0)
`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.
`means`	a vector of means for the k_cont continuous variables (i.e. = rep(0, k_cont))
`vars`	a vector of variances (i.e. = rep(1, k_cont))
`skews`	a vector of skewness values (i.e. = rep(0, k_cont))
`skurts`	a vector of standardized kurtoses (kurtosis - 3, so that normal variables have a value of 0; i.e. = rep(0, k_cont))
`fifths`	a vector of standardized fifth cumulants (not necessary for `method` = "Fleishman"; i.e. = rep(0, k_cont))
`sixths`	a vector of standardized sixth cumulants (not necessary for `method` = "Fleishman"; i.e. = rep(0, k_cont))
`Six`	a list of vectors of correction values to add to the sixth cumulants if no valid pdf constants are found, ex: `Six = list(seq(0.01, 2,by = 0.01), seq(1, 10,by = 0.5))`; if no correction is desired for variable Y_i, set set the i-th list component equal to NULL
`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
`nrand`	the number of random numbers to generate in calculating intermediate correlations (default = 10000)

`lam` a vector of lambda (> 0) constants for the Poisson variables (see `Poisson`)

`size` a vector of size parameters for the Negative Binomial variables (see `NegBinomial`)

`prob` a vector of success probability parameters

`mu` a vector of mean parameters (*Note: either `prob` or `mu` should be supplied for all Negative Binomial variables, not a mixture; default = NULL)

`Sigma` an intermediate correlation matrix to use if the user wants to provide one (default = NULL)

`rho` the target correlation matrix (*must be ordered ordinal, continuous, Poisson, Negative Binomial*; default = NULL)

`cstart` a list containing initial values for root-solving algorithm used in `find_constants` (see `multiStart` for `method` = "Fleishman" or `nleqslv` for `method` = "Polynomial"). If user specified, each list element must be input as a matrix. If no starting values are specified for a given continuous variable, that list element should be NULL. If NULL and all 4 standardized cumulants (rounded to 3 digits) are within 0.01 of those in Headrick's common distribution table (see `Headrick.dist` data), uses his constants as starting values; else, generates n sets of random starting values from uniform distributions.

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

`errorloop` if TRUE, uses `error_loop` to attempt to correct the final correlation (default = FALSE)

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

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

`extra_correct` if TRUE, within each variable pair, if the maximum correlation error is still greater than 0.1, the intermediate correlation is set equal to the target correlation (with the assumption that the calculated final correlation will be less than 0.1 away from the target)

Value

A list whose components vary based on the type of simulated variables. Simulated variables are returned as data.frames:

If **ordinal variables** are produced:

`ordinal_variables` the generated ordinal variables,

`summary_ordinal` a list, where the i-th element contains a data.frame with column 1 = target cumulative probabilities and column 2 = simulated cumulative probabilities for ordinal variable Y_i

If **continuous variables** are produced:

`constants` a data.frame of the constants,

`continuous_variables` the generated continuous variables,

`summary_continuous` a data.frame containing a summary of each variable,

`summary_targetcont` a data.frame containing a summary of the target variables, sixth_correction a vector 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:

Poisson_variables the generated Poisson variables,

summary_Poisson a data.frame containing a summary of each variable

If **Negative Binomial variables** are produced:

Neg_Bin_variables the generated Negative Binomial variables,

summary_Neg_Bin a data.frame containing a summary of each variable

Additionally, the following elements:

correlations the final correlation matrix,

Sigma1 the intermediate correlation before the error loop,

Sigma2 the intermediate correlation matrix after the error loop,

Constants_Time the time in minutes required to calculate the constants,

Intercorrelation_Time the time in minutes required to calculate the intermediate correlation matrix,

Error_Loop_Time the time in minutes required to use the error loop,

Simulation_Time the total simulation time in minutes,

niter a matrix of the number of iterations used for each variable in the error loop,

maxerr the maximum final correlation error (from the target rho).

If a particular element is not required, the result is NULL for that element.

Variable Types and Required Inputs

1) **Continuous Variables:** Continuous variables are simulated using either Fleishman's third-order (method = "Fleishman", doi: 10.1007/BF02293811) or Headrick's fifth-order (method = "Polynomial", doi: 10.1016/S01679473(02)000725) power method transformation. This is a computationally efficient algorithm that simulates continuous distributions through the method of moments. 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:

$Y = c_0 + c_1 * Z + c_2 * Z^2 + c_3 * Z^3 + c_4 * Z^4 + c_5 * Z^5$,

where $Z~N(0, 1)$, and c_4 and c_5 both equal 0 for Fleishman's method. The real constants are calculated by find_constants. All variables are simulated with mean 0 and variance 1, and then transformed to the specified mean and variance at the end.

The required parameters for simulating continuous variables include: mean, variance, skewness, standardized kurtosis (kurtosis - 3), and standardized fifth and sixth cumulants (for method = "Polynomial"). If the goal is to simulate a theoretical distribution (i.e. Gamma, Beta, Logistic, etc.), these values can be obtained using calc_theory. If the goal is to mimic an empirical data set, these values can be found using calc_moments (using the method of moments) or calc_fisherk (using Fisher's k-statistics). 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)`). Due to the nature of the integration involved in `calc_theory`, the results are approximations. Greater accuracy can be achieved by increasing the number of subdivisions (`sub`) used in the integration process. For example, in order to ensure that skew is exactly 0 for symmetric distributions.

For some sets of cumulants, it is either not possible to find power method constants or the calculated constants do not generate valid power method pdfs. In these situations, adding a value to the sixth cumulant may provide solutions (see `find_constants`). When using Headrick's fifth-order approximation, if simulation results indicate that a continuous variable does not generate a valid pdf, the user can try `find_constants` with various sixth cumulant correction vectors to determine if a valid pdf can be found.

2) **Binary and Ordinal 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. The required inputs for ordinal variables are the cumulative marginal probabilities and support values (if desired). The probabilities should be combined into a list of length equal to the number of ordinal 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).

Note for binary variables: the user-suppled probability should be the probability of the 1^{st} (lower) support value. This would ordinarily be considered the probability of *failure* (q), while the probability of the 2^{nd} (upper) support value would be considered the probability of *success* ($p = 1 - q$). The support values should be combined into a separate list. The i^{th} element is a vector containing the r ordered support values.

3) **Count Variables:** Count variables are generated using the inverse cdf method. The cumulative distribution function 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: $Y = F_y^{-1}(\Phi(Z))$. For Poisson variables, the lambda (mean) value should be given. For Negative Binomial variables, the size (target number of successes) and either the success probability or the mean should be given. 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.

More details regarding the variable types can be found in the **Variable Types** vignette.

Overview of Correlation Method 1

The intermediate correlations used in correlation 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 Method 1 and Method 2** vignette for more information and an 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 $\gamma_3^2/\gamma_4 = 2/3$. However, if the fifth and sixth cumulants 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 vector of sixth cumulant correction values 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)`).

2) In addition, 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) As mentioned above, the feasibility of the final correlation matrix rho, given the distribution parameters, should be checked first using `valid_corr`. 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 (r >= 2 categories). 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

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.

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

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

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

Hasselman B (2018). nleqslv: Solve Systems of Nonlinear Equations. R package version 3.3.2. https://CRAN.R-project.org/package=nleqslv

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.

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.

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

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

Varadhan R, Gilbert P (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/

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.

See Also

find_constants, findintercorr, multiStart, nleqslv

Examples

```
Sim1 <- rcorrvar(n = 1000, k_cat = 1, k_cont = 1, method = "Polynomial",
  means = 0, vars = 1, skews = 0, skurts = 0, fifths = 0, sixths = 0,
  marginal = list(c(1/3, 2/3)), support = list(0:2),
  rho = matrix(c(1, 0.4, 0.4, 1), 2, 2))

## Not run:

# Binary, Ordinal, Continuous, Poisson, and Negative Binomial Variables

options(scipen = 999)
seed <- 1234
n <- 10000

Dist <- c("Logistic", "Weibull")
Params <- list(c(0, 1), c(3, 5))
Stcum1 <- calc_theory(Dist[1], Params[[1]])
Stcum2 <- calc_theory(Dist[2], Params[[2]])
Stcum <- rbind(Stcum1, Stcum2)
rownames(Stcum) <- Dist
colnames(Stcum) <- c("mean", "sd", "skew", "skurtosis", "fifth", "sixth")
Stcum
Six <- list(seq(1.7, 1.8, 0.01), seq(0.10, 0.25, 0.01))
marginal <- list(0.3)
lam <- 0.5
size <- 2
prob <- 0.75

Rey <- matrix(0.4, 5, 5)
diag(Rey) <- 1

# Make sure Rey is within upper and lower correlation limits
valid <- valid_corr(k_cat = 1, k_cont = 2, k_pois = 1, k_nb = 1,
                    method = "Polynomial", means = Stcum[, 1],
                    vars = Stcum[, 2]^2, skews = Stcum[, 3],
                    skurts = Stcum[, 4], fifths = Stcum[, 5],
                    sixths = Stcum[, 6], Six = Six, marginal = marginal,
                    lam = lam, size = size, prob = prob, rho = Rey,
                    seed = seed)

# Simulate variables without error loop
Sim1 <- rcorrvar(n = n, k_cat = 1, k_cont = 2, k_pois = 1, k_nb = 1,
                 method = "Polynomial", means = Stcum[, 1],
                 vars = Stcum[, 2]^2, skews = Stcum[, 3],
                 skurts = Stcum[, 4], fifths = Stcum[, 5],
                 sixths = Stcum[, 6], Six = Six, marginal = marginal,
                 lam = lam, size = size, prob = prob, rho = Rey,
                 seed = seed)
```

```
names(Sim1)

# Look at the maximum correlation error
Sim1$maxerr

Sim1_error = round(Sim1$correlations - Rey, 6)

# interquartile-range of correlation errors
quantile(as.numeric(Sim1_error), 0.25)
quantile(as.numeric(Sim1_error), 0.75)

# Simulate variables with error loop
Sim1_EL <- rcorrvar(n = n, k_cat = 1, k_cont = 2,
                    k_pois = 1, k_nb = 1, method = "Polynomial",
                    means = Stcum[, 1], vars = Stcum[, 2]^2,
                    skews = Stcum[, 3], skurts = Stcum[, 4],
                    fifths = Stcum[, 5], sixths = Stcum[, 6],
                    Six = Six, marginal = marginal, lam = lam,
                    size = size, prob = prob, rho = Rey,
                    seed = seed, errorloop = TRUE)
# Look at the maximum correlation error
Sim1_EL$maxerr

EL_error = round(Sim1_EL$correlations - Rey, 6)

# interquartile-range of correlation errors
quantile(as.numeric(EL_error), 0.25)
quantile(as.numeric(EL_error), 0.75)

# Look at results
# Ordinal variables
Sim1_EL$summary_ordinal

# Continuous variables
round(Sim1_EL$constants, 6)
round(Sim1_EL$summary_continuous, 6)
round(Sim1_EL$summary_targetcont, 6)
Sim1_EL$valid.pdf

# Count variables
Sim1_EL$summary_Poisson
Sim1_EL$summary_Neg_Bin

# Generate Plots

# Logistic (1st continuous variable)
# 1) Simulated Data CDF (find cumulative probability up to y = 0.5)
plot_sim_cdf(Sim1_EL$continuous_variables[, 1], calc_cprob = TRUE,
             delta = 0.5)

# 2) Simulated Data and Target Distribution PDFs
plot_sim_pdf_theory(Sim1_EL$continuous_variables[, 1], Dist = "Logistic",
                    params = c(0, 1))
```

```
# 3) Simulated Data and Target Distribution
plot_sim_theory(Sim1_EL$continuous_variables[, 1], Dist = "Logistic",
                params = c(0, 1))


## End(Not run)
```

---

rcorrvar2 *Generation of Correlated Ordinal, Continuous, Poisson, and/or Negative Binomial Variables: Correlation Method 2*

Description

This function simulates k_cat ordinal, k_cont continuous, k_pois Poisson, and/or k_nb Negative Binomial variables with a specified correlation matrix rho. The variables are generated from multivariate normal variables with intermediate correlation matrix Sigma, calculated by findintercorr2, and then transformed. The *ordering* of the variables in rho must be *ordinal* (r >= 2 categories), *continuous*, *Poisson*, and *Negative Binomial* (note that it is possible for k_cat, k_cont, k_pois, and/or k_nb to be 0). The vignette **Overall Workflow for Data Simulation** provides a detailed example discussing the step-by-step simulation process and comparing methods 1 and 2.

Usage

```
rcorrvar2(n = 10000, k_cont = 0, k_cat = 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(), marginal = list(), support = list(), lam = NULL,
  pois_eps = rep(0.0001, 2), size = NULL, prob = NULL, mu = NULL,
  nb_eps = rep(0.0001, 2), Sigma = NULL, rho = NULL, cstart = NULL,
  seed = 1234, errorloop = FALSE, epsilon = 0.001, maxit = 1000,
  extra_correct = TRUE)
```

Arguments

n	the sample size (i.e. the length of each simulated variable; default = 10000)
k_cont	the number of continuous variables (default = 0)
k_cat	the number of ordinal (r >= 2 categories) variables (default = 0)
k_pois	the number of Poisson variables (default = 0)
k_nb	the number of Negative Binomial variables (default = 0)
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.
means	a vector of means for the k_cont continuous variables (i.e. = rep(0, k_cont))
vars	a vector of variances (i.e. = rep(1, k_cont))

skews
: a vector of skewness values (i.e. = rep(0, k_cont))

skurts
: a vector of standardized kurtoses (kurtosis - 3, so that normal variables have a value of 0; i.e. = rep(0, k_cont))

fifths
: a vector of standardized fifth cumulants (not necessary for method = "Fleishman"; i.e. = rep(0, k_cont))

sixths
: a vector of standardized sixth cumulants (not necessary for method = "Fleishman"; i.e. = rep(0, k_cont))

Six
: a list of vectors of correction values to add to the sixth cumulants if no valid pdf constants are found, ex: `Six = list(seq(0.01, 2,by = 0.01), seq(1, 10,by = 0.5))`; if no correction is desired for variable Y_i, set set the i-th list component equal to NULL

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 (> 0) constants for the Poisson variables (see `Poisson`)

pois_eps
: a vector of length k_pois containing the truncation values (default = rep(0.0001, 2))

size
: a vector of size parameters for the Negative Binomial variables (see `NegBinomial`)

prob
: a vector of success probability parameters

mu
: a vector of mean parameters (*Note: either `prob` or `mu` should be supplied for all Negative Binomial variables, not a mixture; default = NULL)

nb_eps
: a vector of length k_nb containing the truncation values (default = rep(0.0001, 2))

Sigma
: an intermediate correlation matrix to use if the user wants to provide one (default = NULL)

rho
: the target correlation matrix (*must be ordered ordinal, continuous, Poisson, Negative Binomial*; default = NULL)

cstart
: a list containing initial values for root-solving algorithm used in `find_constants` (see `multiStart` for `method` = "Fleishman" or `nleqslv` for `method` = "Polynomial"). If user specified, each list element must be input as a matrix. If no starting values are specified for a given continuous variable, that list element should be NULL. If NULL and all 4 standardized cumulants (rounded to 3 digits) are within 0.01 of those in Headrick's common distribution table (see `Headrick.dist` data), uses his constants as starting values; else, generates n sets of random starting values from uniform distributions.

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

errorloop
: if TRUE, uses `error_loop` to attempt to correct the final correlation (default = FALSE)

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

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

`extra_correct` if TRUE, within each variable pair, if the maximum correlation error is still greater than 0.1, the intermediate correlation is set equal to the target correlation (with the assumption that the calculated final correlation will be less than 0.1 away from the target)

Value

A list whose components vary based on the type of simulated variables. Simulated variables are returned as data.frames:

If **ordinal variables** are produced:

`ordinal_variables` the generated ordinal variables,

`summary_ordinal` a list, where the i-th element contains a data.frame with column 1 = target cumulative probabilities and column 2 = simulated cumulative probabilities for ordinal variable Y_i

If **continuous variables** are produced:

`constants` a data.frame of the constants,

`continuous_variables` the generated continuous variables,

`summary_continuous` a data.frame containing a summary of each variable,

`summary_targetcont` a data.frame containing a summary of the target variables,

`sixth_correction` a vector 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:

`Poisson_variables` the generated Poisson variables,

`summary_Poisson` a data.frame containing a summary of each variable

If **Negative Binomial variables** are produced:

`Neg_Bin_variables` the generated Negative Binomial variables,

`summary_Neg_Bin` a data.frame containing a summary of each variable

Additionally, the following elements:

`correlations` the final correlation matrix,

`Sigma1` the intermediate correlation before the error loop,

`Sigma2` the intermediate correlation matrix after the error loop,

`Constants_Time` the time in minutes required to calculate the constants,

`Intercorrelation_Time` the time in minutes required to calculate the intermediate correlation matrix,

`Error_Loop_Time` the time in minutes required to use the error loop, `Simulation_Time` the total simulation time in minutes,

`niter` a matrix of the number of iterations used for each variable in the error loop,

`maxerr` the maximum final correlation error (from the target rho).

If a particular element is not required, the result is NULL for that element.

Variable Types and Required Inputs

1) **Continuous Variables:** Continuous variables are simulated using either Fleishman's third-order (`method` = "Fleishman", doi: 10.1007/BF02293811) or Headrick's fifth-order (`method` = "Polynomial", doi: 10.1016/S01679473(02)000725) power method transformation. This is a computationally efficient algorithm that simulates continuous distributions through the method of moments. 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:

$Y = c_0 + c_1 * Z + c_2 * Z^2 + c_3 * Z^3 + c_4 * Z^4 + c_5 * Z^5$,

where $Z\ N(0, 1)$, and c_4 and c_5 both equal 0 for Fleishman's method. The real constants are calculated by `find_constants`. All variables are simulated with mean 0 and variance 1, and then transformed to the specified mean and variance at the end.

The required parameters for simulating continuous variables include: mean, variance, skewness, standardized kurtosis (kurtosis - 3), and standardized fifth and sixth cumulants (for `method` = "Polynomial"). If the goal is to simulate a theoretical distribution (i.e. Gamma, Beta, Logistic, etc.), these values can be obtained using `calc_theory`. If the goal is to mimic an empirical data set, these values can be found using `calc_moments` (using the method of moments) or `calc_fisherk` (using Fisher's k-statistics). 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)`). Due to the nature of the integration involved in `calc_theory`, the results are approximations. Greater accuracy can be achieved by increasing the number of subdivisions (`sub`) used in the integration process. For example, in order to ensure that skew is exactly 0 for symmetric distributions.

For some sets of cumulants, it is either not possible to find power method constants or the calculated constants do not generate valid power method pdfs. In these situations, adding a value to the sixth cumulant may provide solutions (see `find_constants`). When using Headrick's fifth-order approximation, if simulation results indicate that a continuous variable does not generate a valid pdf, the user can try `find_constants` with various sixth cumulant correction vectors to determine if a valid pdf can be found.

2) **Binary and Ordinal 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. The required inputs for ordinal variables are the cumulative marginal probabilities and support values (if desired). The probabilities should be combined into a list of length equal to the number of ordinal 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).

Note for binary variables: the user-suppled probability should be the probability of the 1^{st} (lower) support value. This would ordinarily be considered the probability of *failure* (q), while the probability of the 2^{nd} (upper) support value would be considered the probability of *success* ($p = 1 - q$).

The support values should be combined into a separate list. The i^{th} element is a vector containing the r ordered support values.

3) **Count Variables:** Count variables are generated using the inverse cdf method. The cumulative distribution function 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: $Y = F_y^{-1}(\Phi(Z))$. For Poisson variables, the lambda (mean) value should be given. For Negative Binomial variables, the size (target number of successes) and either the success probability or the mean should be given. 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. In addition, a vector of total cumulative probability truncation values should be provided (one for Poisson and one for Negative Binomial). These values represent the amount of probability removed from the range of the cdf's F_Y when creating finite supports. The value may vary by variable, but a good default value is 0.0001 (suggested by Barbiero & Ferrari, 2015, doi: 10.1002/asmb.2072).

More details regarding the variable types can be found in the **Variable Types** vignette.

Overview of Correlation Method 2

The intermediate correlations used in correlation method 2 are less simulation based than those in correlation 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 `ordnorm`.

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 Method 1 and Method 2** vignette for more information and an 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 $\gamma_3^2/\gamma_4 = 2/3$. However, if the fifth and sixth cumulants 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 vector of sixth cumulant correction values 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)`).

2) In addition, 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) As mentioned above, the feasibility of the final correlation matrix rho, given the distribution parameters, should be checked first using `valid_corr2`. 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 (r >= 2 categories). 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`

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.

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

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

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

Hasselman B (2018). nleqslv: Solve Systems of Nonlinear Equations. R package version 3.3.2. `https://CRAN.R-project.org/package=nleqslv`

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.

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.

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

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

Varadhan R, Gilbert P (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/`

See Also

`find_constants`, `findintercorr2`, `multiStart`, `nleqslv`

Examples

```
Sim1 <- rcorrvar2(n = 1000, k_cat = 1, k_cont = 1, method = "Polynomial",
  means = 0, vars = 1, skews = 0, skurts = 0, fifths = 0, sixths = 0,
  marginal = list(c(1/3, 2/3)), support = list(0:2),
  rho = matrix(c(1, 0.4, 0.4, 1), 2, 2))

## Not run:

# Binary, Ordinal, Continuous, Poisson, and Negative Binomial Variables

options(scipen = 999)
seed <- 1234
n <- 10000

Dist <- c("Logistic", "Weibull")
Params <- list(c(0, 1), c(3, 5))
Stcum1 <- calc_theory(Dist[1], Params[[1]])
Stcum2 <- calc_theory(Dist[2], Params[[2]])
Stcum <- rbind(Stcum1, Stcum2)
rownames(Stcum) <- Dist
colnames(Stcum) <- c("mean", "sd", "skew", "skurtosis", "fifth", "sixth")
Stcum
Six <- list(seq(1.7, 1.8, 0.01), seq(0.10, 0.25, 0.01))
marginal <- list(0.3)
```

```
lam <- 0.5
pois_eps <- 0.0001
size <- 2
prob <- 0.75
nb_eps <- 0.0001

Rey <- matrix(0.4, 5, 5)
diag(Rey) <- 1

# Make sure Rey is within upper and lower correlation limits
valid2 <- valid_corr2(k_cat = 1, k_cont = 2, k_pois = 1, k_nb = 1,
                      method = "Polynomial", means = Stcum[, 1],
                      vars = Stcum[, 2]^2, skews = Stcum[, 3],
                      skurts = Stcum[, 4], fifths = Stcum[, 5],
                      sixths = Stcum[, 6], Six = Six, marginal = marginal,
                      lam = lam, pois_eps = pois_eps, size = size,
                      prob = prob, nb_eps = nb_eps, rho = Rey,
                      seed = seed)

# Simulate variables without error loop
Sim2 <- rcorrvar2(n = n, k_cat = 1, k_cont = 2, k_pois = 1, k_nb = 1,
                  method = "Polynomial", means = Stcum[, 1],
                  vars = Stcum[, 2]^2, skews = Stcum[, 3],
                  skurts = Stcum[, 4], fifths = Stcum[, 5],
                  sixths = Stcum[, 6], Six = Six, marginal = marginal,
                  lam = lam, pois_eps = pois_eps, size = size,
                  prob = prob, nb_eps = nb_eps, rho = Rey,
                  seed = seed)
names(Sim2)

# Look at the maximum correlation error
Sim2$maxerr

Sim2_error = round(Sim2$correlations - Rey, 6)

# interquartile-range of correlation errors
quantile(as.numeric(Sim2_error), 0.25)
quantile(as.numeric(Sim2_error), 0.75)

# Simulate variables with error loop
Sim2_EL <- rcorrvar2(n = n, k_cat = 1, k_cont = 2, k_pois = 1, k_nb = 1,
                     method = "Polynomial", means = Stcum[, 1],
                     vars = Stcum[, 2]^2, skews = Stcum[, 3],
                     skurts = Stcum[, 4], fifths = Stcum[, 5],
                     sixths = Stcum[, 6], Six = Six, marginal = marginal,
                     lam = lam, pois_eps = pois_eps, size = size,
                     prob = prob, nb_eps = nb_eps, rho = Rey,
                     seed = seed, errorloop = TRUE)
# Look at the maximum correlation error
Sim2_EL$maxerr

EL_error = round(Sim2_EL$correlations - Rey, 6)
```

```
# interquartile-range of correlation errors
quantile(as.numeric(EL_error), 0.25)
quantile(as.numeric(EL_error), 0.75)

# Look at results
# Ordinal variables
Sim2_EL$summary_ordinal

# Continuous variables
round(Sim2_EL$constants, 6)
round(Sim2_EL$summary_continuous, 6)
round(Sim2_EL$summary_targetcont, 6)
Sim2_EL$valid.pdf

# Count variables
Sim2_EL$summary_Poisson
Sim2_EL$summary_Neg_Bin

# Generate Plots

# Logistic (1st continuous variable)
# 1) Simulated Data CDF (find cumulative probability up to y = 0.5)
plot_sim_cdf(Sim2_EL$continuous_variables[, 1], calc_cprob = TRUE,
             delta = 0.5)

# 2) Simulated Data and Target Distribution PDFs
plot_sim_pdf_theory(Sim2_EL$continuous_variables[, 1], Dist = "Logistic",
                    params = c(0, 1))

# 3) Simulated Data and Target Distribution
plot_sim_theory(Sim2_EL$continuous_variables[, 1], Dist = "Logistic",
                params = c(0, 1))


## End(Not run)
```

separate_rho *Separate Target Correlation Matrix by Variable Type*

Description

This function separates the target correlation matrix rho by variable type (ordinal, continuous, Poisson, and/or Negative Binomial). The function is used in findintercorr, rcorrvar, findintercorr2, and rcorrvar2. This would not ordinarily be called directly by the user.

Usage

```
separate_rho(k_cat, k_cont, k_pois, k_nb, rho)
```

Arguments

`k_cat`	the number of ordinal (r >= 2 categories) variables
`k_cont`	the number of continuous variables
`k_pois`	the number of Poisson variables
`k_nb`	the number of Negative Binomial variables
`rho`	the target correlation matrix

Value

a list containing the target correlation matrix components by variable combination

See Also

`findintercorr`, `rcorrvar`, `findintercorr2`, `rcorrvar2`

`SimMultiCorrData` *Simulation of Correlated Data with Multiple Variable Types*

Description

SimMultiCorrData generates continuous (normal or non-normal), binary, ordinal, and count (Poisson or Negative Binomial) variables with a specified correlation matrix. It can also produce a single continuous variable. This package can be used to simulate data sets that mimic real-world situations (i.e. clinical data sets, plasmodes, as in Vaughan et al., 2009, doi: 10.1016/j.csda.2008.02.032). All variables are generated from standard normal variables with an imposed intermediate correlation matrix. Continuous variables are simulated by specifying mean, variance, skewness, standardized kurtosis, and fifth and sixth standardized cumulants using either Fleishman's Third-Order (doi: 10.1007/BF02293811) or Headrick's Fifth-Order (doi: 10.1016/S01679473(02)000725) Polynomial Transformation. 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 differ primarily according to the calculation of the intermediate correlation matrix. In **Correlation Method 1**, the intercorrelations involving count variables are determined using a simulation based, logarithmic correlation correction (adapting Yahav and Shmueli's 2012 method, doi: 10.1002/asmb.901). In **Correlation Method 2**, the count variables are treated as ordinal (adapting Barbiero and Ferrari's 2015 modification of `GenOrd-package`, doi: 10.1002/asmb.2072). There is an optional error loop that corrects the final correlation matrix to be within a user-specified precision value. The package also includes functions to calculate standardized cumulants for theoretical distributions or from real data sets, check if a target correlation matrix is within the possible correlation bounds (given the distributions of the simulated variables), summarize results, numerically or graphically, to verify valid power method pdfs, and to calculate lower standardized kurtosis bounds.

Vignettes

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

1) **Benefits of SimMultiCorrData and Comparison to Other Packages** describes some of the ways **SimMultiCorrData** improves upon other simulation packages.

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

3) **Function by Topic** describes each function, separated by topic.

4) **Comparison of Correlation Method 1 and Correlation Method 2** describes the two simulation pathways that can be followed.

5) **Overview of Error Loop** details the algorithm involved in the optional error loop that improves the accuracy of the simulated variables' correlation matrix.

6) **Overall Workflow for Data Simulation** gives a step-by-step guideline to follow with an example containing continuous (normal and non-normal), binary, ordinal, Poisson, and Negative Binomial variables. It also demonstrates the use of the standardized cumulant calculation function, correlation check functions, the lower kurtosis boundary function, and the plotting functions.

7) **Comparison of Simulation Distribution to Theoretical Distribution or Empirical Data** gives a step-by-step guideline for comparing a simulated univariate continuous distribution to the target distribution with an example.

8) **Using the Sixth Cumulant Correction to Find Valid Power Method Pdfs** demonstrates how to use the sixth cumulant correction to generate a valid power method pdf and the effects this has on the resulting distribution.

Functions

This package contains 3 *simulation* functions:

`nonnormvar1`, `rcorrvar`, and `rcorrvar2`

8 data description (*summary*) functions:

`calc_fisherk`, `calc_moments`, `calc_theory`, `cdf_prob`, `power_norm_corr`, `pdf_check`, `sim_cdf_prob`, `stats_pdf`

8 *graphing* functions:

`plot_cdf`, `plot_pdf_ext`, `plot_pdf_theory`, `plot_sim_cdf`, `plot_sim_ext`, `plot_sim_pdf_ext`, `plot_sim_pdf_theory`, `plot_sim_theory`

5 *support* functions:

`calc_lower_skurt`, `find_constants`, `pdf_check`, `valid_corr`, `valid_corr2`

and 30 *auxiliary* functions (should not normally be called by the user, but are called by other functions):

`calc_final_corr`, `chat_nb`, `chat_pois`, `denom_corr_cat`, `error_loop`, `error_vars`, `findintercorr`, `findintercorr2`, `findintercorr_cat_nb`, `findintercorr_cat_pois`, `findintercorr_cont`, `findintercorr_cont_cat`, `findintercorr_cont_nb`, `findintercorr_cont_nb2`, `findintercorr_cont_pois`, `findintercorr_cont_pois2`, `findintercorr_nb`, `findintercorr_pois`, `findintercorr_pois_nb`, `fleish`, `fleish_Hessian`, `fleish_skurt_check`, `intercorr_fleish`, `intercorr_poly`, max_count_support, ordnorm, poly, poly_skurt_check, separate_rho, var_cat

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.

Amatya A & Demirtas H (2016). MultiOrd: Generation of Multivariate Ordinal Variates. R package version 2.2. https://CRAN.R-project.org/package=MultiOrd

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

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.

Demirtas H, Nordgren R, & Allozi R (2017). PoisBinOrdNonNor: Generation of Up to Four Different Types of Variables. R package version 1.3. https://CRAN.R-project.org/package=PoisBinOrdNonNor

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

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

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

Hasselman B (2018). nleqslv: Solve Systems of Nonlinear Equations. R package version 3.3.2. https://CRAN.R-project.org/package=nleqslv

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.

Kaiser S, Traeger D, & Leisch F (2011). Generating Correlated Ordinal Random Values. Technical Report Number 94, Department of Statistics, University of Munich. `https://epub.ub.uni-muenchen.de/12157/1/kaiser-tr-94-ordinal.pdf`

Leisch F, Kaiser AWS, & Hornik K (2010). orddata: Generation of Artificial Ordinal and Binary Data. R package version 0.1/r4.

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

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

Varadhan R, Gilbert P (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.

See Also

Useful link: `https://github.com/AFialkowski/SimMultiCorrData`

`sim_cdf_prob` *Calculate Simulated (Empirical) Cumulative Probability*

Description

This function calculates a cumulative probability using simulated data and Martin Maechler's `ecdf` function. Fn is a step function with jumps i/n at observation values, where i is the number of tied observations at that value. Missing values are ignored. For observations $y = (y1, y2, ..., yn)$, Fn is the fraction of observations less or equal to t, i.e., $Fn(t) = sum[yi <= t]/n$. This works for continuous, ordinal, or count variables.

Usage

```
sim_cdf_prob(sim_y, delta = 0.5)
```

Arguments

`sim_y`	a vector of simulated data
`delta`	the value y at which to evaluate the cumulative probability

Value

A list with components:

`cumulative_prob` the empirical cumulative probability up to delta

`Fn` the empirical distribution function

See Also

`ecdf`, `plot_sim_cdf`

Examples

```
# Beta(a = 4, b = 2) Distribution:
x <- rbeta(10000, 4, 2)
sim_cdf_prob(x, delta = 0.5)
```

---

stats_pdf *Calculate Theoretical Statistics for a Valid Power Method PDF*

Description

This function calculates the 100*`alpha` percent symmetric trimmed mean (0 < `alpha` < 0.50), median, mode, and maximum height of a valid power method pdf, after using `pdf_check`. It will stop with an error if the pdf is invalid. The equations are those from Headrick & Kowalchuk (2007, doi: 10.1080/10629360600605065).

Usage

```
stats_pdf(c, method = c("Fleishman", "Polynomial"), alpha = 0.025, mu = 0,
  sigma = 1, lower = -10, upper = 10, sub = 1000)
```

Arguments

c	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
method	the method used to find the constants. "Fleishman" uses Fleishman's third-order polynomial transformation and "Polynomial" uses Headrick's fifth-order transformation.
alpha	proportion to be trimmed from the lower and upper ends of the power method pdf (default = 0.025)
mu	mean for the continuous variable (default = 0)
sigma	standard deviation for the continuous variable (default = 1)
lower	lower bound for integration of the standard normal variable Z that generates the continuous variable (default = -10)
upper	upper bound for integration (default = 10)
sub	the number of subdivisions to use in the integration; if no result, try increasing sub (requires longer computation time; default = 1000)

Value

A vector with components:

trimmed_mean the trimmed mean value

median the median value

mode the mode value

max_height the maximum pdf height

References

Please see references for pdf_check.

See Also

find_constants, pdf_check

Examples

```
stats_pdf(c = c(0, 1, 0, 0, 0, 0), method = "Polynomial", alpha = 0.025)

## Not run:
# Beta(a = 4, b = 2) Distribution:
con <- find_constants(method = "Polynomial", skews = -0.467707,
                      skurts = -0.375, fifths = 1.403122,
                      sixths = -0.426136)$constants
stats_pdf(c = con, method = "Polynomial", alpha = 0.025)

## End(Not run)
```

valid_corr *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 `rcorrvar`. 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 >= 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.

Note: Some pieces of the function code have been adapted from Demirtas, Hu, & Allozi's (2017) `validation_specs`. This function (`valid_corr`) extends the methods to:

1) non-normal continuous variables generated by Fleishman's third-order or Headrick's fifth-order polynomial transformation method, and

2) Negative Binomial variables (including all pairwise correlations involving them).

Please see the **Comparison of Method 1 and Method 2** vignette for more information regarding method 1.

Usage

```
valid_corr(k_cat = 0, k_cont = 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(), marginal = list(), lam = NULL, size = NULL,
  prob = NULL, mu = NULL, rho = NULL, n = 100000, seed = 1234)
```

Arguments

`k_cat`	the number of ordinal (r >= 2 categories) variables (default = 0)
`k_cont`	the number of continuous variables (default = 0)
`k_pois`	the number of Poisson variables (default = 0)
`k_nb`	the number of 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.
`means`	a vector of means for the k_cont continuous variables (i.e. = rep(0, k_cont))
`vars`	a vector of variances (i.e. = rep(1, k_cont))
`skews`	a vector of skewness values (i.e. = rep(0, k_cont))
`skurts`	a vector of standardized kurtoses (kurtosis - 3, so that normal variables have a value of 0; i.e. = rep(0, k_cont))

fifths	a vector of standardized fifth cumulants (not necessary for method = "Fleishman"; i.e. = rep(0, k_cont))
sixths	a vector of standardized sixth cumulants (not necessary for method = "Fleishman"; i.e. = rep(0, k_cont))
Six	a list of vectors of correction values to add to the sixth cumulants if no valid pdf constants are found, ex: `Six = list(seq(0.01, 2,by = 0.01), seq(1, 10,by = 0.5))`; if no correction is desired for variable Y_i, set the i-th list component equal to NULL
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())
lam	a vector of lambda (> 0) constants for the Poisson variables (see Poisson)
size	a vector of size parameters for the Negative Binomial variables (see NegBinomial)
prob	a vector of success probability parameters
mu	a vector of mean parameters (*Note: either prob or mu should be supplied for all Negative Binomial variables, not a mixture; default = NULL)
rho	the target correlation matrix (*must be ordered ordinal, continuous, Poisson, Negative Binomial*; default = NULL)
n	the sample size (i.e. the length of each simulated variable; default = 100000)
seed	the seed value for random number generation (default = 1234)

Value

A list with components:

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.

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 vector of sixth cumulant correction values 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)`). Due to the nature of the integration involved in calc_theory, the results are approximations. Greater accuracy can

be achieved by increasing the number of subdivisions (sub) used in the integration process. For example, in order to ensure that skew is exactly 0 for symmetric distributions.

2) In addition, 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.

The Generate, Sort, and Correlate (GSC, Demirtas & Hedeker, 2011, doi: 10.1198/tast.2011.10090) Algorithm

The GSC algorithm is a flexible method for determining empirical correlation bounds when the theoretical bounds are unknown. The steps are as follows:

1) Generate independent random samples from the desired distributions using a large number of observations (i.e. N = 100,000).

2) Lower Bound: Sort the two variables in opposite directions (i.e., one increasing and one decreasing) and find the sample correlation.

3) Upper Bound: Sort the two variables in the same direction and find the sample correlation.

Demirtas & Hedeker showed that the empirical bounds computed from the GSC method are similar to the theoretical bounds (when they are known).

The Frechet-Hoeffding Correlation Bounds

Suppose two random variables Y_i and Y_j have cumulative distribution functions given by F_i and F_j. Let U be a uniform(0,1) random variable, i.e. representing the distribution of the standard normal cdf. Then Hoeffing (1940) and Frechet (1951) showed that bounds for the correlation between Y_i and Y_j are given by

$$(corr(F_i^{-1}(U), F_j^{-1}(1-U)), corr(F_i^{-1}(U), F_j^{-1}(U)))$$

The processes used to find the correlation bounds for each variable type are described below:

Ordinal Variables

Binary pairs: The correlation bounds are determined as in Demirtas et al. (2012, doi: 10.1002/sim.5362), who used the method of Emrich & Piedmonte (1991, doi: 10.1080/00031305.1991.10475828). The joint distribution is determined by "borrowing" the moments of a multivariate normal distribution. For two binary variables Y_i and Y_j, with success probabilities p_i and p_j, the lower correlation bound is given by

$$max(-\sqrt{(p_ip_j)/(q_iq_j)},\ -\sqrt{(q_iq_j)/(p_ip_j)})$$

and the upper bound by

$$min(\sqrt{(p_iq_j)/(q_ip_j)},\ \sqrt{(q_ip_j)/(p_iq_j)})$$

Here, $q_i = 1 - p_i$ and $q_j = 1 - p_j$.

Binary-Ordinal or Ordinal-Ordinal pairs: Randomly generated variables with the given marginal distributions are used in the GSC algorithm to find the correlation bounds.

Continuous Variables

Continuous variables are randomly generated using constants from find_constants and a vector of sixth cumulant correction values (if provided.) The GSC algorithm is used to find the lower and upper bounds.

Poisson Variables

Poisson variables with the given means (lam) are randomly generated using the inverse cdf method. The Frechet-Hoeffding bounds are used for the correlation bounds.

Negative Binomial Variables

Negative Binomial variables with the given sizes and success probabilities (prob) or means (mu) are randomly generated using the inverse cdf method. The Frechet-Hoeffding bounds are used for the correlation bounds.

Continuous - Ordinal Pairs

Randomly generated ordinal variables with the given marginal distributions and the previously generated continuous variables are used in the GSC algorithm to find the correlation bounds.

Ordinal - Poisson Pairs

Randomly generated ordinal variables with the given marginal distributions and randomly generated Poisson variables with the given means (lam) are used in the GSC algorithm to find the correlation bounds.

Ordinal - Negative Binomial Pairs

Randomly generated ordinal variables with the given marginal distributions and randomly generated Negative Binomial variables with the given sizes and success probabilities (prob) or means (mu) are used in the GSC algorithm to find the correlation bounds.

Continuous - Poisson Pairs

The previously generated continuous variables and randomly generated Poisson variables with the given means (lam) are used in the GSC algorithm to find the correlation bounds.

Continuous - Negative Binomial Pairs

The previously generated continuous variables and randomly generated Negative Binomial variables with the given sizes and success probabilities (prob) or means (mu) are used in the GSC algorithm to find the correlation bounds.

Poisson - Negative Binomial Pairs

Poisson variables with the given means (lam) and Negative Binomial variables with the given sizes and success probabilities (prob) or means (mu) are randomly generated using the inverse cdf method. The Frechet-Hoeffding bounds are used for the correlation bounds.

References

Please see rcorrvar 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. Sur les tableaux de correlation dont les marges sont donnees. Ann. l'Univ. Lyon SectA. 1951;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.

Hakan Demirtas, Yiran Hu and Rawan Allozi (2017). PoisBinOrdNor: Data Generation with Poisson, Binary, Ordinal and Normal Components. R package version 1.4. https://CRAN.R-project.org/package=PoisBinOrdNor

See Also

find_constants, rcorrvar

Examples

```
valid_corr(n = 1000, k_cat = 1, k_cont = 1, method = "Polynomial",
  means = 0, vars = 1, skews = 0, skurts = 0, fifths = 0, sixths = 0,
  marginal = list(c(1/3, 2/3)), rho = matrix(c(1, 0.4, 0.4, 1), 2, 2))

## Not run:

# Binary, Ordinal, Continuous, Poisson, and Negative Binomial Variables

options(scipen = 999)
seed <- 1234
n <- 10000

# Continuous Distributions: Normal, t (df = 10), Chisq (df = 4),
#                           Beta (a = 4, b = 2), Gamma (a = 4, b = 4)
Dist <- c("Gaussian", "t", "Chisq", "Beta", "Gamma")

# calculate standardized cumulants
# those for the normal and t distributions are rounded to ensure the
# correct values (i.e. skew = 0)

M1 <- round(calc_theory(Dist = "Gaussian", params = c(0, 1)), 8)
M2 <- round(calc_theory(Dist = "t", params = 10), 8)
M3 <- calc_theory(Dist = "Chisq", params = 4)
M4 <- calc_theory(Dist = "Beta", params = c(4, 2))
M5 <- calc_theory(Dist = "Gamma", params = c(4, 4))
M <- cbind(M1, M2, M3, M4, M5)
M <- round(M[-c(1:2),], digits = 6)
```

```
colnames(M) <- Dist
rownames(M) <- c("skew", "skurtosis", "fifth", "sixth")
means <- rep(0, length(Dist))
vars <- rep(1, length(Dist))

# Binary and Ordinal Distributions
marginal <- list(0.3, 0.4, c(0.1, 0.5), c(0.3, 0.6, 0.9),
                 c(0.2, 0.4, 0.7, 0.8))
support <- list()

# Poisson Distributions
lam <- c(1, 5, 10)

# Negative Binomial Distributions
size <- c(3, 6)
prob <- c(0.2, 0.8)

ncat <- length(marginal)
ncont <- ncol(M)
npois <- length(lam)
nnb <- length(size)

# Create correlation matrix from a uniform distribution (-0.8, 0.8)
set.seed(seed)
Rey <- diag(1, nrow = (ncat + ncont + npois + nnb))
for (i in 1:nrow(Rey)) {
  for (j in 1:ncol(Rey)) {
    if (i > j) Rey[i, j] <- runif(1, -0.8, 0.8)
    Rey[j, i] <- Rey[i, j]
  }
}

# Test for positive-definiteness
library(Matrix)
if(min(eigen(Rey, symmetric = TRUE)$values) < 0) {
  Rey <- as.matrix(nearPD(Rey, corr = T, keepDiag = T)$mat)
}

# Make sure Rey is within upper and lower correlation limits
valid <- valid_corr(k_cat = ncat, k_cont = ncont, k_pois = npois,
                    k_nb = nnb, method = "Polynomial", means = means,
                    vars = vars, skews = M[1, ], skurts = M[2, ],
                    fifths = M[3, ], sixths = M[4, ], marginal = marginal,
                    lam = lam, size = size, prob = prob, rho = Rey,
                    seed = seed)

## End(Not run)
```

`valid_corr2` *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 simulation with rcorrvar2. 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 >= 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.

Note: Some pieces of the function code have been adapted from Demirtas, Hu, & Allozi's (2017) validation_specs. This function (valid_corr2) extends the methods to:

1) non-normal continuous variables generated by Fleishman's third-order or Headrick's fifth-order polynomial transformation method,

2) Negative Binomial variables (including all pairwise correlations involving them), and

3) Count variables are treated as ordinal when calculating the bounds since that is the intermediate correlation calculation method.

Please see the **Comparison of Method 1 and Method 2** vignette for more information regarding method 2.

Usage

```
valid_corr2(k_cat = 0, k_cont = 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(), marginal = list(), lam = NULL, pois_eps = NULL,
  size = NULL, prob = NULL, mu = NULL, nb_eps = NULL, rho = NULL,
  n = 100000, seed = 1234)
```

Arguments

`k_cat`	the number of ordinal (r >= 2 categories) variables (default = 0)
`k_cont`	the number of continuous variables (default = 0)
`k_pois`	the number of Poisson variables (default = 0)
`k_nb`	the number of 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.
`means`	a vector of means for the k_cont continuous variables (i.e. = rep(0, k_cont))
`vars`	a vector of variances (i.e. = rep(1, k_cont))
`skews`	a vector of skewness values (i.e. = rep(0, k_cont))
`skurts`	a vector of standardized kurtoses (kurtosis - 3, so that normal variables have a value of 0; i.e. = rep(0, k_cont))
`fifths`	a vector of standardized fifth cumulants (not necessary for `method` = "Fleishman"; i.e. = rep(0, k_cont))

sixths	a vector of standardized sixth cumulants (not necessary for method = "Fleishman"; i.e. = rep(0, k_cont))
Six	a list of vectors of correction values to add to the sixth cumulants if no valid pdf constants are found, ex: `Six = list(seq(0.01, 2,by = 0.01), seq(1, 10,by = 0.5))`; if no correction is desired for variable Y_i, set the i-th list component equal to NULL
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())
lam	a vector of lambda (> 0) constants for the Poisson variables (see Poisson)
pois_eps	a vector of length k_pois containing the truncation values (i.e. = rep(0.0001, k_pois); default = NULL)
size	a vector of size parameters for the Negative Binomial variables (see NegBinomial)
prob	a vector of success probability parameters
mu	a vector of mean parameters (*Note: either prob or mu should be supplied for all Negative Binomial variables, not a mixture; default = NULL)
nb_eps	a vector of length k_nb containing the truncation values (i.e. = rep(0.0001, k_nb); default = NULL)
rho	the target correlation matrix (*must be ordered ordinal, continuous, Poisson, Negative Binomial*; default = NULL)
n	the sample size (i.e. the length of each simulated variable; default = 100000)
seed	the seed value for random number generation (default = 1234)

Value

A list with components:

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.

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 vector of sixth cumulant correction

values 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)`). Due to the nature of the integration involved in `calc_theory`, the results are approximations. Greater accuracy can be achieved by increasing the number of subdivisions (`sub`) used in the integration process. For example, in order to ensure that skew is exactly 0 for symmetric distributions.

2) In addition, 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.

The Generate, Sort, and Correlate (GSC, Demirtas & Hedeker, 2011, doi: 10.1198/tast.2011.10090) Algorithm

The GSC algorithm is a flexible method for determining empirical correlation bounds when the theoretical bounds are unknown. The steps are as follows:

1) Generate independent random samples from the desired distributions using a large number of observations (i.e. N = 100,000).

2) Lower Bound: Sort the two variables in opposite directions (i.e., one increasing and one decreasing) and find the sample correlation.

3) Upper Bound: Sort the two variables in the same direction and find the sample correlation.

Demirtas & Hedeker showed that the empirical bounds computed from the GSC method are similar to the theoretical bounds (when they are known).

The processes used to find the correlation bounds for each variable type are described below:

Ordinal Variables

Binary pairs: The correlation bounds are determined as in Demirtas et al. (2012, doi: 10.1002/sim.5362), who used the method of Emrich & Piedmonte (1991, doi: 10.1080/00031305.1991.10475828). The joint distribution is determined by "borrowing" the moments of a multivariate normal distribution. For two binary variables Y_i and Y_j, with success probabilities p_i and p_j, the lower correlation bound is given by

$$max(-\sqrt{(p_i p_j)/(q_i q_j)},\ -\sqrt{(q_i q_j)/(p_i p_j)})$$

and the upper bound by

$$min(\sqrt{(p_i q_j)/(q_i p_j)},\ \sqrt{(q_i p_j)/(p_i q_j)})$$

Here, $q_i = 1 - p_i$ and $q_j = 1 - p_j$.

Binary-Ordinal or Ordinal-Ordinal pairs: Randomly generated variables with the given marginal distributions are used in the GSC algorithm to find the correlation bounds.

Continuous Variables

Continuous variables are randomly generated using constants from `find_constants` and a vector of sixth cumulant correction values (if provided.) The GSC algorithm is used to find the lower and upper bounds.

Poisson Variables

The maximum support values, given the vector of cumulative probability truncation values (pois_eps) and vector of means (lam), are calculated using max_count_support. The finite supports are used to determine marginal distributions for each Poisson variable. Randomly generated variables with the given marginal distributions are used in the GSC algorithm to find the correlation bounds.

Negative Binomial Variables

The maximum support values, given the vector of cumulative probability truncation values (nb_eps) and vectors of sizes and success probabilities (prob) or means (mu), are calculated using max_count_support. The finite supports are used to determine marginal distributions for each Negative Binomial variable. Randomly generated variables with the given marginal distributions are used in the GSC algorithm to find the correlation bounds.

Continuous - Ordinal Pairs

Randomly generated ordinal variables with the given marginal distributions and the previously generated continuous variables are used in the GSC algorithm to find the correlation bounds.

Ordinal - Poisson Pairs

Randomly generated ordinal and Poisson variables with the given marginal distributions are used in the GSC algorithm to find the correlation bounds.

Ordinal - Negative Binomial Pairs

Randomly generated ordinal and Negative Binomial variables with the given marginal distributions are used in the GSC algorithm to find the correlation bounds.

Continuous - Poisson Pairs

The previously generated continuous variables and randomly generated Poisson variables with the given marginal distributions are used in the GSC algorithm to find the correlation bounds.

Continuous - Negative Binomial Pairs

The previously generated continuous variables and randomly generated Negative Binomial variables with the given marginal distributions are used in the GSC algorithm to find the correlation bounds.

Poisson - Negative Binomial Pairs

Randomly generated variables with the given marginal distributions are used in the GSC algorithm to find the correlation bounds.

References

Please see rcorrvar2 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. Sur les tableaux de correlation dont les marges sont donnees. Ann. l'Univ. Lyon SectA. 1951;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.

Hakan Demirtas, Yiran Hu and Rawan Allozi (2017). PoisBinOrdNor: Data Generation with Poisson, Binary, Ordinal and Normal Components. R package version 1.4. https://CRAN.R-project.org/package=PoisBinOrdNor

See Also

find_constants, rcorrvar2

Examples

```
valid_corr2(n = 1000, k_cat = 1, k_cont = 1, method = "Polynomial",
  means = 0, vars = 1, skews = 0, skurts = 0, fifths = 0, sixths = 0,
  marginal = list(c(1/3, 2/3)), rho = matrix(c(1, 0.4, 0.4, 1), 2, 2))

## Not run:

# Binary, Ordinal, Continuous, Poisson, and Negative Binomial Variables

options(scipen = 999)
seed <- 1234
n <- 10000

# Continuous Distributions: Normal, t (df = 10), Chisq (df = 4),
#                           Beta (a = 4, b = 2), Gamma (a = 4, b = 4)
Dist <- c("Gaussian", "t", "Chisq", "Beta", "Gamma")

# calculate standardized cumulants
# those for the normal and t distributions are rounded to ensure the
# correct values (i.e. skew = 0)

M1 <- round(calc_theory(Dist = "Gaussian", params = c(0, 1)), 8)
M2 <- round(calc_theory(Dist = "t", params = 10), 8)
M3 <- calc_theory(Dist = "Chisq", params = 4)
M4 <- calc_theory(Dist = "Beta", params = c(4, 2))
M5 <- calc_theory(Dist = "Gamma", params = c(4, 4))
M <- cbind(M1, M2, M3, M4, M5)
M <- round(M[-c(1:2),], digits = 6)
```

```
colnames(M) <- Dist
rownames(M) <- c("skew", "skurtosis", "fifth", "sixth")
means <- rep(0, length(Dist))
vars <- rep(1, length(Dist))

# Binary and Ordinal Distributions
marginal <- list(0.3, 0.4, c(0.1, 0.5), c(0.3, 0.6, 0.9),
                 c(0.2, 0.4, 0.7, 0.8))
support <- list()

# Poisson Distributions
lam <- c(1, 5, 10)

# Negative Binomial Distributions
size <- c(3, 6)
prob <- c(0.2, 0.8)

ncat <- length(marginal)
ncont <- ncol(M)
npois <- length(lam)
nnb <- length(size)

# Create correlation matrix from a uniform distribution (-0.8, 0.8)
set.seed(seed)
Rey <- diag(1, nrow = (ncat + ncont + npois + nnb))
for (i in 1:nrow(Rey)) {
  for (j in 1:ncol(Rey)) {
    if (i > j) Rey[i, j] <- runif(1, -0.8, 0.8)
    Rey[j, i] <- Rey[i, j]
  }
}

# Test for positive-definiteness
library(Matrix)
if(min(eigen(Rey, symmetric = TRUE)$values) < 0) {
  Rey <- as.matrix(nearPD(Rey, corr = T, keepDiag = T)$mat)
}

# Make sure Rey is within upper and lower correlation limits
valid <- valid_corr2(k_cat = ncat, k_cont = ncont, k_pois = npois,
                     k_nb = nnb, method = "Polynomial", means = means,
                     vars = vars, skews = M[1, ], skurts = M[2, ],
                     fifths = M[3, ], sixths = M[4, ], marginal = marginal,
                     lam = lam, pois_eps = rep(0.0001, npois),
                     size = size, prob = prob, nb_eps = rep(0.0001, nnb),
                     rho = Rey, seed = seed)


## End(Not run)
```

---

var_cat *Calculate Variance of Binary or Ordinal Variable*

Description

This function calculates the variance of a binary or ordinal (r > 2 categories) variable. It uses the formula given by Olsson et al. (1982, doi: 10.1007/BF02294164) in describing polyserial and point-polyserial correlations. The function is used to find intercorrelations involving ordinal variables or variables that are treated as ordinal (i.e. count variables in the method used in `rcorrvar2`). For an ordinal variable with r >= 2 categories, the variance is given by:

$$\sum_{j=1}^{r} {y_j}^2 * p_j - \left(\sum_{j=1}^{r} y_j * p_j\right)^2$$

. Here, y_j is the j-th support value and p_j is $Pr(Y = y_j)$. This function would not ordinarily be called by the user.

Usage

```
var_cat(marginal, support)
```

Arguments

`marginal`	a vector of cumulative probabilities defining the marginal distribution of the 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 vector of containing the ordered support values

Value

A scalar equal to the variance

References

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

See Also

`ordnorm`, `rcorrvar`, `rcorrvar2`, `findintercorr_cont_cat`, `findintercorr_cont_pois2`, `findintercorr_cont_nb2`

Index