Package ‘GenOrd’

September 12, 2015

Type Package

Title Simulation of Discrete Random Variables with Given Correlation Matrix and Marginal Distributions

Version 1.4.0

Date 2015-09-11

Author Alessandro Barbiero, Pier Alda Ferrari

Maintainer Alessandro Barbiero <alessandro.barbiero@unimi.it>

Description A gaussian copula based procedure for generating samples from discrete random variables with prescribed correlation matrix and marginal distributions.

License GPL

LazyLoad yes

Depends mvtnorm, Matrix, MASS, stats

NeedsCompilation no

Repository CRAN

Date/Publication 2015-09-12 17:19:55

R topics documented:

GenOrd-package *Simulation of Discrete Random Variables with Given Correlation Matrix and Marginal Distributions*

Description

The package implements a procedure for generating samples from a multivariate discrete random variable with pre-specified correlation matrix and marginal distributions. The marginal distributions are linked together through a gaussian copula. The procedure is developed in two steps: the first step (function `ordcont`) sets up the gaussian copula in order to achieve the desired correlation matrix on the target random discrete components; the second step (`ordsample`) generates samples from the target variables. The procedure can handle both Pearson's and Spearman's correlations, and any finite support for the discrete variables. The intermediate function `contord` computes the correlations of the multivariate discrete variable derived from correlated normal variables through discretization. Function `corrcheck` returns the lower and upper bounds of the correlation coefficient of each pair of discrete variables given their marginal distributions, i.e., returns the range of feasible bivariate correlations.

This version has fixed some drawbacks in terminology in the previous version; the only actual change concerns the parameter `cormat` in the `ordsample` function. Further examples of implementation have been added.

Details

Package:	GenOrd
Type:	Package
Version:	1.4.0
Date:	2015-09-11
License:	GPL
LazyLoad:	yes

Author(s)

Alessandro Barbiero, Pier Alda Ferrari

Maintainer: Alessandro Barbiero <alessandro.barbiero@unimi.it>

References

P.A. Ferrari, A. Barbiero (2012) Simulating ordinal data, Multivariate Behavioral Research, 47(4), 566-589

A. Barbiero, P.A. Ferrari (2014) Simulation of correlated Poisson variables. Applied Stochastic Models in Business and Industry, doi 10.1002/asmb.2072

See Also

`contord`, `ordcont`, `corrcheck`, `ordsample`

`contord` *Correlations of discretized variables*

Description

The function computes the correlation matrix of the k variables, with given marginal distributions, derived discretizing a k-variate standard normal variable with given correlation matrix

Usage

```
contord(marginal, Sigma, support = list(), Spearman = FALSE)
```

Arguments

`marginal` a list of k elements, where k is the number of variables. The i-th element of `marginal` is the vector of the cumulative probabilities defining the marginal distribution of the i-th component of the multivariate variable. If the i-th component can take k_i values, the i-th element of `marginal` will contain $k_i - 1$ probabilities (the k_i-th is obviously 1 and shall not be included).

`Sigma` the correlation matrix of the standard multivariate normal variable

`support` a list of k elements, where k is the number of variables. The i-th element of `support` is the vector containing the ordered values of the support of the i-th variable. By default, the support of the i-th variable is $1, 2, ..., k_i$

`Spearman` if `TRUE`, the function finds Spearman's correlations (and it is not necessary to provide `support`), if `FALSE` (default) Pearson's correlations

Value

the correlation matrix of the discretized variables

Author(s)

Alessandro Barbiero, Pier Alda Ferrari

See Also

`ordcont`, `ordsample`, `corrcheck`

Examples

```
# consider 4 discrete variables
k <- 4
# with these marginal distributions
marginal <- list(0.4,c(0.3,0.6), c(0.25,0.5,0.75), c(0.1,0.2,0.8,0.9))
# generated discretizing a multivariate standard normal variable
# with correlation matrix
Sigma <- matrix(0.5,4,4)
diag(Sigma) <- 1
# the resulting correlation matrix for the discrete variables is
contord(marginal, Sigma)
# note all the correlations are smaller than the original 0.6
# change Sigma, adding a negative correlation
Sigma[1,2] <- -0.15
Sigma[2,1] <- Sigma[1,2]
Sigma
# checking whether Sigma is still positive definite
eigen(Sigma)$values # all >0, OK
contord(marginal, Sigma)
```

corrcheck *Checking correlations for feasibility*

Description

The function returns the lower and upper bounds of the correlation coefficients of each pair of discrete variables given their marginal distributions, i.e., returns the range of feasible bivariate correlations.

Usage

```
corrcheck(marginal, support = list(), Spearman = FALSE)
```

Arguments

`marginal` a list of k elements, where k is the number of variables. The i-th element of `marginal` is the vector of the cumulative probabilities defining the marginal distribution of the i-th component of the multivariate variable. If the i-th component can take k_i values, the i-th element of `marginal` will contain $k_i - 1$ probabilities (the k_i-th is obviously 1 and shall not be included).

`support` a list of k elements, where k is the number of variables. The i-th element of `support` is the vector containing the ordered values of the support of the i-th variable. By default, the support of the i-th variable is $1, 2, ..., k_i$

`Spearman` `TRUE` if we consider Spearman's correlation, `FALSE` (default) if we consider Pearson's correlation

Value

The functions returns a list of two matrices: the former contains the lower bounds, the latter the upper bounds of the feasible pairwise correlations (on the extra-diagonal elements)

Author(s)

Alessandro Barbiero, Pier Alda Ferrari

See Also

`contord`, `ordcont`, `ordsample`

Examples

```
# four variables
k <- 4
# with 2, 3, 4, and 5 categories (Likert scales, by default)
kj <- c(2,3,4,5)
# and these marginal distributions (set of cumulative probabilities)
marginal <- list(0.4, c(0.6,0.9), c(0.1,0.2,0.4), c(0.6,0.7,0.8,0.9))
corrcheck(marginal) # lower and upper bounds for Pearson's rho
corrcheck(marginal, Spearman=TRUE) # lower and upper bounds for Spearman's rho
# change the supports
support <- list(c(0,1), c(1,2,4), c(1,2,3,4), c(0,1,2,5,10))
corrcheck(marginal, support=support) # updated bounds
```

`ordcont` *Computing the "intermediate" correlation matrix for the multivariate standard normal in order to achieve the "target" correlation matrix for the multivariate discrete variable*

Description

The function computes the correlation matrix of the k-dimensional standard normal r.v. yielding the desired correlation matrix `Sigma` for the k-dimensional r.v. with desired marginal distributions `marginal`

Usage

```
ordcont(marginal, Sigma, support = list(), Spearman = FALSE,
epsilon = 1e-06, maxit = 100)
```

Arguments

`marginal` a list of k elements, where k is the number of variables. The i-th element of `marginal` is the vector of the cumulative probabilities defining the marginal distribution of the i-th component of the multivariate variable. If the i-th component can take k_i values, the i-th element of `marginal` will contain $k_i - 1$ probabilities (the k_i-th is obviously 1 and shall not be included).

`Sigma` the target correlation matrix of the discrete variables

`support` a list of k elements, where k is the number of variables. The i-th element of `support` is the vector containing the ordered values of the support of the i-th variable. By default, the support of the i-th variable is $1, 2, ..., k_i$

`Spearman` if `TRUE`, the function finds Spearman's correlations (and it is not necessary to provide `support`), if `FALSE` (default) Pearson's correlations

`epsilon` the maximum tolerated error between target and actual correlations

`maxit` the maximum number of iterations allowed for the algorithm

Value

a list of five elements

`SigmaC` the correlation matrix of the multivariate standard normal variable

`SigmaO` the actual correlation matrix of the discretized variables (it should approximately coincide with the target correlation matrix `Sigma`)

`Sigma` the target correlation matrix of the discrete variables

`niter` a matrix containing the number of iterations performed by the algorithm, one for each pair of variables

`maxerr` the actual maximum error (the maximum absolute deviation between actual and target correlations of the discrete variables)

Note

For some choices of `marginal` and `Sigma`, there may not exist a feasible k-variate probability mass function or the algorithm may not provide a feasible correlation matrix `SigmaC`. In this case, the procedure stops and exits with an error. The value of the maximum tolerated absolute error `epsilon` on the elements of the correlation matrix for the target r.v. can be set by the user: a value between 1e-6 and 1e-2 seems to be an acceptable compromise assuring both the precision of the results and the convergence of the algorithm; moreover, a maximum number of iterations can be chosen (`maxit`), in order to avoid possible endless loops

Author(s)

Alessandro Barbiero, Pier Alda Ferrari

See Also

`contord`, `ordsample`, `corrcheck`

Examples

```
# consider a 4-dimensional ordinal variable
k <- 4
# with different number of categories
kj <- c(2,3,4,5)
# and uniform marginal distributions
marginal <- list(0.5, (1:2)/3, (1:3)/4, (1:4)/5)
```

```
corrcheck(marginal)
# and the following correlation matrix
Sigma <- matrix(c(1,0.5,0.4,0.3,0.5,1,0.5,0.4,0.4,0.5,1,0.5,0.3,0.4,0.5,1),
4, 4, byrow=TRUE)
Sigma
# the correlation matrix of the standard 4-dimensional standard normal
# ensuring Sigma is
res <- ordcont(marginal, Sigma)
res[[1]]
# change some marginal distributions
marginal <- list(0.3, c(1/3, 2/3), c(1/5, 2/5, 3/5), c(0.1, 0.2, 0.4, 0.6))
corrcheck(marginal)
# and notice how the correlation matrix of the multivariate normal changes...
res <- ordcont(marginal, Sigma)
res[[1]]
# change Sigma, adding a negative correlation
Sigma[1,2] <- -0.2
Sigma[2,1] <- Sigma[1,2]
Sigma
# checking whether Sigma is still positive definite
eigen(Sigma)$values # all >0, OK
res <- ordcont(marginal, Sigma)
res[[1]]
```

`ordsample` *Drawing a sample of discrete data*

Description

The function draws a sample from a multivariate discrete variable with correlation matrix `Sigma` and prescribed marginal distributions `marginal`

Usage

```
ordsample(n, marginal, Sigma, support = list(), Spearman = FALSE,
cormat = "discrete")
```

Arguments

`n` the sample size

`marginal` a list of k elements, where k is the number of variables. The i-th element of `marginal` is the vector of the cumulative probabilities defining the marginal distribution of the i-th component of the multivariate variable. If the i-th component can take k_i values, the i-th element of marginal will contain $k_i - 1$ probabilities (the k_i-th is obviously 1 and shall not be included).

`Sigma` the target correlation matrix of the multivariate discrete variable

`support` a list of k elements, where k is the number of variables. The i-th element of `support` is the vector containing the ordered values of the support of the i-th variable. By default, the support of the i-th variable is $1, 2, ..., k_i$

`Spearman` if `TRUE`, the function finds Spearman's correlations (and it is not necessary to provide `support`), if `FALSE` (default) Pearson's correlations

`cormat` `"discrete"` if the `Sigma` in input is the target correlation matrix of the multivariate discrete variable; `"continuous"` if the `Sigma` in input is the intermediate correlation matrix of the multivariate standard normal

Value

a $n \times k$ matrix of values drawn from the k-variate discrete r.v. with the desired marginal distributions and correlation matrix

Author(s)

Alessandro Barbiero, Pier Alda Ferrari

See Also

`contord`, `ordcont`, `corrcheck`

Examples

```
# Example 1

# draw a sample from a bivariate ordinal variable
# with 4 of categories and asymmetrical marginal distributions
# and correlation coefficient 0.6 (to be checked)
k <- 2
marginal <- list(c(0.1,0.3,0.6), c(0.4,0.7,0.9))
corrcheck(marginal) # check ok
Sigma <- matrix(c(1,0.6,0.6,1),2,2)
# sample size 1000
n <- 1000
# generate a sample of size n
m <- ordsample(n, marginal, Sigma)
head(m)
# sample correlation matrix
cor(m) # compare it with Sigma
# empirical marginal distributions
cumsum(table(m[,1]))/n
cumsum(table(m[,2]))/n # compare them with the two marginal distributions

# Example 1bis

# draw a sample from a bivariate ordinal variable
# with 4 of categories and asymmetrical marginal distributions
# and Spearman correlation coefficient 0.6 (to be checked)
k <- 2
marginal <- list(c(0.1,0.3,0.6), c(0.4,0.7,0.9))
corrcheck(marginal, Spearman=TRUE) # check ok
Sigma <- matrix(c(1,0.6,0.6,1),2,2)
# sample size 1000
n <- 1000
```

```
# generate a sample of size n
m <- ordsample(n, marginal, Sigma, Spearman=TRUE)
head(m)
# sample correlation matrix
cor(rank(m[,1]),rank(m[,2])) # compare it with Sigma
# empirical marginal distributions
cumsum(table(m[,1]))/n
cumsum(table(m[,2]))/n # compare them with the two marginal distributions

# Example 1ter

# draw a sample from a bivariate random variable
# with binomial marginal distributions (n=3, p=1/3 and n=4, p=2/3)
# and Pearson correlation coefficient 0.6 (to be checked)
k <- 2
marginal <- list(pbinom(0:2, 3, 1/3),pbinom(0:3, 4, 2/3))
marginal
corrcheck(marginal, support=list(0:3, 0:4)) # check ok
Sigma <- matrix(c(1,0.6,0.6,1),2,2)
# sample size 1000
n <- 1000
# generate a sample of size n
m <- ordsample(n, marginal, Sigma, support=list(0:3,0:4))
head(m)
# sample correlation matrix
cor(m) # compare it with Sigma
# empirical marginal distributions
cumsum(table(m[,1]))/n
cumsum(table(m[,2]))/n # compare them with the two marginal distributions

# Example 2

# draw a sample from a 4-dimensional ordinal variable
# with different number of categories and uniform marginal distributions
# and different correlation coefficients
k <- 4
marginal <- list(0.5, c(1/3,2/3), c(1/4,2/4,3/4), c(1/5,2/5,3/5,4/5))
corrcheck(marginal)
# select a feasible correlation matrix
Sigma <- matrix(c(1,0.5,0.4,0.3,0.5,1,0.5,0.4,0.4,0.5,1,0.5,0.3,0.4,0.5,1),
4, 4, byrow=TRUE)
Sigma
# sample size 100
n <- 100
# generate a sample of size n
set.seed(1)
m <- ordsample(n, marginal, Sigma)
# sample correlation matrix
cor(m) # compare it with Sigma
# empirical marginal distribution
cumsum(table(m[,4]))/n # compare it with the fourth marginal
head(m)
# or equivalently...
```

```
set.seed(1)
res <- ordcont(marginal, Sigma)
res[[1]] # the intermediate correlation matrix of the multivariate normal
m <- ordsample(n, marginal, res[[1]], cormat="continuous")
head(m)

# Example 3
# simulation of two correlated Poisson r.v.
# modification to GenOrd sampling function for Poisson distribution
ordsamplep<-function (n, lambda, Sigma)
{
   k <- length(lambda)
   valori <- mvrnorm(n, rep(0, k), Sigma)
   for (i in 1:k)
   {
   valori[, i] <- qpois(pnorm(valori[,i]), lambda[i])
   }
   return(valori)
}
# number of variables
k <- 2
# Poisson parameters
lambda <- c(2, 5)
# correlation matrix
Sigma <- matrix(0.25, 2, 2)
diag(Sigma) <- 1
# sample size
n <- 10000
# preliminar stage: support TRUNCATION
# required for recovering the correlation matrix
# of the standard bivariate normal
# truncation error
epsilon <- 0.0001
# corresponding maximum value
kmax <- qpois(1-epsilon, lambda)
# truncated marginals
l <- list()
for(i in 1:k)
{
l[[i]] <- 0:kmax[i]
}
marg <- list()
for(i in 1:k)
{
marg[[i]] <- dpois(0:kmax[i],lambda[i])
marg[[i]][kmax[i]+1] <- 1-sum(marg[[i]][1:(kmax[i])])
}
cm <- list()
for(i in 1:k)
{
cm[[i]] <- cumsum(marg[[i]])
cm[[i]] <- cm[[i]][-(kmax[i]+1)]
}
```

```
# check feasibility of correlation matrix
RB <- corrcheck(cm, support=l)
RL <- RB[[1]]
RU <- RB[[2]]
Sigma <= RU & Sigma >= RL # OK
res <- ordcont(cm, Sigma, support=l)
res[[1]]
Sigma <- res[[1]]
# draw the sample
m <- ordsamplep(n, lambda, Sigma)
# sample correlation matrix
cor(m)
head(m)

# Example 4
# simulation of 4 correlated binary and Poisson r.v.'s (2+2)
# modification to GenOrd sampling function
ordsamplep <- function (n, marginal, lambda, Sigma)
{
k <- length(lambda)
valori <- mvrnorm(n, rep(0, k), Sigma)
for(i in 1:k)
{
if(lambda[i]==0)
{
valori[, i] <- as.integer(cut(valori[, i], breaks = c(min(valori[,i]) - 1,
               qnorm(marginal[[i]]), max(valori[, i]) + 1)))
valori[, i] <- support[[i]][valori[, i]]
}
else
{
valori[, i] <- qpois(pnorm(valori[,i]), lambda[i])
}
}
return(valori)
}
# number of variables
k <- 4
# Poisson parameters (only 3rd and 4th are Poisson)
lambda <- c(0, 0, 2, 5)
# 1st and 2nd are Bernoulli with p=0.5
marginal <- list()
marginal[[1]] <- .5
marginal[[2]] <- .5
marginal[[3]] <- 0
marginal[[4]] <- 0
# support
support <- list()
support[[1]] <- 0:1
support[[2]] <- 0:1
# correlation matrix
Sigma <- matrix(0.25, k, k)
diag(Sigma) <- 1
```

```
# sample size
n <- 10000
# preliminar stage: support TRUNCATION
# required for recovering the correlation matrix
# of the standard bivariate normal
# truncation error
epsilon <- 0.0001
# corresponding maximum value
kmax <- qpois(1-epsilon, lambda)
# truncated marginals
for(i in 3:4)
{
support[[i]] <- 0:kmax[i]
}
marg <- list()
for(i in 3:4)
{
marg[[i]] <- dpois(0:kmax[i],lambda[i])
marg[[i]][kmax[i]+1] <- 1-sum(marg[[i]][1:(kmax[i])])
}
for(i in 3:4)
{
marginal[[i]] <- cumsum(marg[[i]])
marginal[[i]] <- marginal[[i]][-(kmax[i]+1)]
}
# check feasibility of correlation matrix
RB <- corrcheck(marginal, support=support)
RL <- RB[[1]]
RU <- RB[[2]]
Sigma <= RU & Sigma >= RL # OK
# compute correlation matrix of the 4-variate standard normal
res <- ordcont(marginal, Sigma, support=support)
res[[1]]
Sigma <- res[[1]]
# draw the sample
m <- ordsamplep(n, marginal, lambda, Sigma)
# sample correlation matrix
cor(m)
head(m)
```

Index