Heterocop

Ekaterina Tomilina

2023-06-18

This package enables the user to simulate data which multivariate cumulative distribution function (CDF) can be associated to a Gaussian copula. It can also estimate the correlation matrix of the copula.

Context

When working with d heterogeneous vectors $X_1, ..., X_d$, it can be complicated to infer a correlation network as well-known statistical methods do not work in case of mixed data. Indeed, most classical network inference methods such as glasso or Gaussian graphical models rely on the assumption that the data follow a Gaussian distribution. Recently, the Non-paranormal distribution was introduced for network inference for continuous, non-Gaussian data (Liu (2009)). It consists in a transformation of the cumulative distribution functions via a Gaussian copula and provides results in the continuous case. The aim of my PhD thesis, which is currently in process, is to extend this method to discrete and mixed variables.

The Model

Let $X_1, ..., X_d$ be d $n-$dimensional vectors of any type (continuous, mixed, discrete...). Let $F_1, ..., F_d$ denote their marginal CDFs, Φ^{-1} the inverse of the standard normal CDF and Φ_Σ the Gaussian CDF of correlation matrix Σ. We define the Gaussian copula in the following way:

$$F(X_1, ..., X_d) = C_\Sigma(F_1(X_1), ..., F_d(X_d)) := \Phi_\Sigma(\Phi^{-1}(F_1(X_1)), ..., \Phi^{-1}(F_d(X_d)))$$

NB: In classical network inference, we often use the precision matrix as we study conditional correlations. For the moment, our method only estimates the correlation matrix of the copula and we suppose that it can be expressed as a block matrix.

Data simulation

When the correlation matrix Σ and the marginal laws $F_1, ..., F_d$ are known, one can easily simulate $X_1, ..., X_d$ via `CopulaSim` which uses the generalized inverse method.

For example, let us suppose that we want to simulate 6 variables on 10 observations which joint CDF can be expressed via a Gaussian copula of correlation matrix

$$\begin{pmatrix} 1 & 0.2 & 0 & 0 & 0 & 0 \\ 0.2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 0 & 0.5 & 1 \end{pmatrix}$$

and where two of the variables follow a $\mathcal{E}(1)$ distribution, two of them a $\mathcal{N}(0, 1)$ distribution, and the remaining two a $\mathcal{N}(0.76, 2)$ distribution, we can run the following code

```
CopulaSim(10, c(2,2,2), c(0.2,0.1,0.5), list(function(p){qexp(p=p, rate=1)}, function(p){qnorm(p=p, mea
```

which provides us with the following results:

X1	X2	X3	X4	X5	X6
1.7056649	0.0082197	-0.4183352	1.3217250	0.0785178	-2.7526541
0.0534913	-0.5068320	4.1114721	1.3617981	0.7153550	3.7497630
0.2965800	0.2590553	-0.6316534	-0.2805566	1.4036919	-1.0424930
-0.5968745	0.7382471	0.8419935	0.3771468	0.2029957	0.8074479
0.0302296	1.0314776	3.3862483	1.8628831	1.1185338	3.9369113
-0.4281937	0.6021521	-2.5918756	-2.5022283	0.1335696	-2.4479234
0.4241074	0.4477801	4.2030026	-4.4919697	3.4554332	1.3104547
-0.9292858	0.6594167	1.2344751	-1.9549175	2.2287371	3.3751610
0.3469102	-1.2598578	2.9491101	1.9781932	0.4173552	-0.6038013
0.2853150	-0.0761829	0.0173461	-1.7222104	3.8791097	1.6819310

`CopulaSim` uses the generalized inverse method, that is:

- Step 1: Simulation of a vector $(U_1, ..., U_d)$ of probabilities for Gaussian data linked by a correlation matrix Σ.
- Step 2: For each U_j, $X_j = F_j^{\leftarrow}(U_j)$ where $F_j^{\leftarrow}$ denotes the quantile function of X_j

The package also enables the user to simulate the vector of probabilities via the `GCopula` function that is actually used in `CopulaSim`.

For example, if we want to simulate the probabilities for 3 Gaussian vectors of length 10 that are linked by the correlation matrix

$$\begin{pmatrix} 1 & 0.2 & 0.4 \\ 0.2 & 1 & 0.7 \\ 0.4 & 0.7 & 1 \end{pmatrix}$$

we simply have to run the following code:

```
R <- matrix(c(1,0.2,0.4,0.2,1,0.7,0.4,0.7,1),3,3)
n<- 10
GCopula(R,n)
```

This function returns three gaussian vectors linked by the correlation matrix Σ such as in the table below:

U1	U2	U3
0.9673474	0.4207003	0.4785577
0.3491988	0.8527469	0.6491750
0.7321148	0.3922034	0.4124209
0.7384666	0.6346403	0.2915581
0.6113077	0.2480264	0.1161423
0.0535747	0.7531260	0.1910281
0.5363120	0.2817649	0.1800963
0.5745939	0.7162650	0.7022488
0.0686167	0.6445953	0.3166103
0.7107739	0.5600119	0.7545418

The package also provides the user with the function `diag_block_matrix` which simulates a block-wise correlation matrix in which the size of blocks and the coefficients have to be specified as arguments. It is also used by `CopulaSim`. For example, the following code

```
diag_block_matrix(c(2,5,3), c(0.2,0.75,0.5))
```

returns the following matrix

$$\begin{pmatrix} 1 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.75 & 0.75 & 0.75 & 0.75 & 0 & 0 & 0 \\ 0 & 0 & 0.75 & 1 & 0.75 & 0.75 & 0.75 & 0 & 0 & 0 \\ 0 & 0 & 0.75 & 0.75 & 1 & 0.75 & 0.75 & 0 & 0 & 0 \\ 0 & 0 & 0.75 & 0.75 & 0.75 & 1 & 0.75 & 0 & 0 & 0 \\ 0 & 0 & 0.75 & 0.75 & 0.75 & 0.75 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 1 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0.5 & 1 \end{pmatrix}$$

Correlation matrix estimation

Besides simulating the data, the package also enables the user to estimate the correlation matrix of the copula via the `rho_estim` function.

The function takes as arguments a data frame containing $(X_1, ..., X_d)$ as well as the shape (vector or matrix) depending on the dimension of the data and returns a correlation matrix or a correlation coefficient.

For example, the code below takes as an argument the data from above.

```
rho_estim(data,rep("C",6))
```

It returns the following estimated correlation matrix:

```
#>             [,1]        [,2]       [,3]       [,4]        [,5]       [,6]
#> [1,]  1.0000000 -0.70461584  0.2457813  0.2829643 -0.22506958 -0.6164883
#> [2,] -0.7046158  1.00000000 -0.1205067 -0.4634922 -0.02313926  0.4562612
#> [3,]  0.2457813 -0.12050665  1.0000000  0.3022521  0.56828888  0.7871006
#> [4,]  0.2829643 -0.46349218  0.3022521  1.0000000 -0.53824338  0.3354042
#> [5,] -0.2250696 -0.02313926  0.5682889 -0.5382434  1.00000000  0.7152089
#> [6,] -0.6164883  0.45626118  0.7871006  0.3354042  0.71520886  1.0000000
```

To do so, it maximizes the pairwise likelihood, that is (Mazo (2022)):

$$L_n(\Sigma) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d}\sum_{j'=j+1}^{d} \log f_{jj'}(X_{ij}, X_{ij'}; \widehat{F}_j, \widehat{F}_{j'}, \Sigma)$$

where the matrix Σ denotes the correlation matrix of the copula described above, X_j and X'_j a pair of variables and $\hat{F}_j$ and $\hat{F}_{j'}$ the corresponding empirical distribution functions of said variables. Moreover, $f_{jj'}$ denotes the pairwise density of the pair X_j, $X_{j'}$ which expression differs depending on the nature of the variables. When both variables are continuous, it takes the following expression:

$$f(x_j, x_{j'}, \widehat{F}_j, \widehat{F}_{j'}, \rho) = c_\rho(\widehat{F}_j(x_j), \widehat{F}_{j'}(x_j)) f_j(x_j) f_{j'}(x_{j'})$$

where

$$c_\rho(u, v) = \frac{1}{\sqrt{1-\rho^2}} \exp\left(\frac{2\rho\Phi^{-1}(u)\Phi^{-1}(v) - \rho^2(\Phi^{-1}(u)^2 + \Phi^{-1}(v)^2)}{2(1-\rho^2)}\right)$$

denotes the density of the Gaussian copula. When both variables are discrete, it takes the following expression:

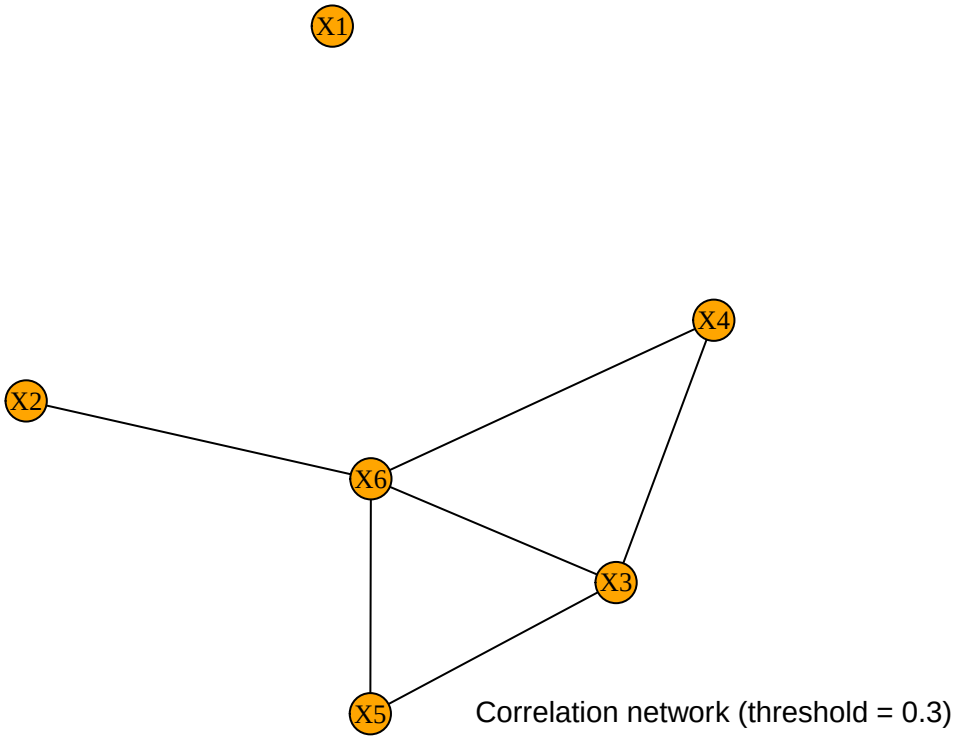
$$\mathbb{P}(X_j = x_j, X_{j'} = x_{j'}) = C_\rho(\widehat{F}_j(x_j), \widehat{F}_{j'}(x_{j'})) + C_\rho(\widehat{F}_j(x_j - \epsilon_j)\widehat{F}_{j'}(x_{j'} - \epsilon_{j'})) - C_\rho(\widehat{F}_j(x_j - \epsilon_j), \widehat{F}_{j'}(x_{j'})) - C_\rho(\widehat{F}_j(x_j), \widehat{F}_{j'}(x_{j'} - \epsilon_{j'}))$$

If X_j is continuous and $X_{j'}$ is discrete, it takes the expression below:

$$f(x_j, x_{j'}) = f_1(x_1) \int_{\widehat{F}_{j'}(x_{j'}-\epsilon_{j'})}^{\widehat{F}_{j'}(x_{j'})} c_\rho(\widehat{F}_j(x_j), v')\lambda(dv')$$

Finally, the package enables the user to plot the correlation network from a dataframe containing the initial variables via the function `cor_network_graph` in which you have to specify your correlation threshold, whether the length of the edges is proportional to the weight of the variables, and the color of the vertices. For example, if we take the dataset from above, the function below returns its correlation graph thresholded at 0.3.

```
cor_network_graph(data, rep("C",6), 0.3, TRUE, "orange")
```



Liu. 2009. "The Nonparanormal: Semiparametric Estimation of High Dimensional Undirected Graphs." *Journal of Machine Learning Research.*

Mazo. 2022. "A Randomized Pariwise Likelihood Method for Complex Statistical Inferences."