heterocop: an R package for Gaussian copula semi-parametric inference for mixed data

Ekaterina Tomilina

2024-09-12

This package enables the user to quantify dependencies between mixed (continuous, discrete, binary) variables in the framework of a Gaussian copula model by estimating the correlation matrix of the copula. It has been presented in the article "Mixed copula-based models for semi-parametric network inference: an application to multi-omics data" (Tomilina, Mazo, and Jaffrézic (2024)).

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. We propose an extension of this model to the case of 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:

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

Estimation

In order to estimate the correlation matrix of the copula, the rho_estim function uses the pairwise maximum likelihood estimator (Mazo, Rau, and Karlis (2023)). It returns the estimated correlation matrix of the copula and takes as arguments the data set and the variable types in a vector. In the example below, we have used a subset of the ICGC data set (Zhang, Bajari, and D. (2019)) which contains 5 RNA-seq, 5 protein and 5 mutation variables. We have specified the variable types, where a "C" stands for "continuous" and a "D" for "discrete".

```
data(icgc_data)
R <- rho_estim(icgc_data,c(rep("C",10),rep("D",5)))
```

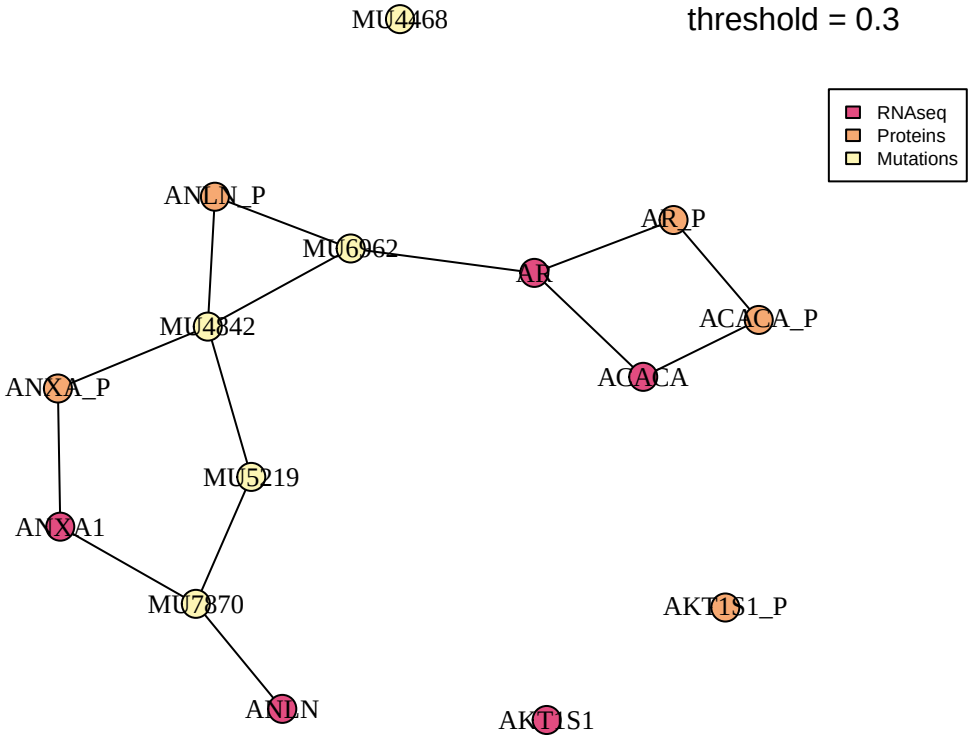
	ACACA	AKT1S1	ANLN	ANXA1	AR	ACACA_P	AKT1S1_P	ANLN_P	ANXA_P	AR_P	MU5219	MU4468	MU7870	MU4842	MU6962
ACACA	1.00	-0.13	0.09	-0.32	0.36	0.58	0.01	-0.07	-0.11	0.23	0.00	0.10	-0.06	-0.11	-0.34
AKT1S1	-0.13	1.00	-0.13	-0.16	-0.19	0.11	0.07	0.12	-0.13	0.14	-0.01	-0.01	-0.14	0.09	-0.16
ANLN	0.09	-0.13	1.00	0.13	-0.36	-0.07	0.29	0.22	0.07	-0.47	-0.35	-0.19	0.37	-0.25	-0.12
ANXA1	-0.32	-0.16	0.13	1.00	-0.36	-0.22	0.07	0.06	0.62	-0.37	0.05	0.06	0.48	0.27	0.12
AR	0.36	-0.19	-0.36	-0.36	1.00	0.14	-0.17	-0.16	-0.18	0.67	0.13	0.13	-0.18	0.06	0.37
ACACA_P	0.58	0.11	-0.07	-0.22	0.14	1.00	-0.07	-0.29	-0.28	0.37	-0.05	-0.01	-0.28	-0.47	-0.70
AKT1S1_P	0.01	0.07	0.29	0.07	-0.17	-0.07	1.00	0.17	-0.21	-0.15	-0.17	0.11	0.15	0.02	-0.20
ANLN_P	-0.07	0.12	0.22	0.06	-0.16	-0.29	0.17	1.00	-0.06	-0.21	-0.09	-0.18	-0.15	0.49	0.54
ANXA_P	-0.11	-0.13	0.07	0.62	-0.18	-0.28	-0.21	-0.06	1.00	-0.36	0.24	0.09	0.17	0.31	0.09

AR_P	0.23	0.14	-0.47	-0.37	0.67	0.37	-0.15	-0.21	-0.36	1.00	0.10	0.19	-0.28	-0.25	0.05
MU5219	0.00	-0.01	-0.35	0.05	0.13	-0.05	-0.17	-0.09	0.24	0.10	1.00	-0.95	0.48	0.37	-0.90
MU4468	0.10	-0.01	-0.19	0.06	0.13	-0.01	0.11	-0.18	0.09	0.19	-0.95	1.00	0.29	0.18	-0.95
MU7870	-0.06	-0.14	0.37	0.48	-0.18	-0.28	0.15	-0.15	0.17	-0.28	0.48	0.29	1.00	-0.90	-0.94
MU4842	-0.11	0.09	-0.25	0.27	0.06	-0.47	0.02	0.49	0.31	-0.25	0.37	0.18	-0.90	1.00	0.84
MU6962	-0.34	-0.16	-0.12	0.12	0.37	-0.70	-0.20	0.54	0.09	0.05	-0.90	-0.95	-0.94	0.84	1.00

Graphical representation

The cor_network_graph function enables to visualize the obtained network. It takes as arguments the dataset, the correlation matrix, the threshold and a legend.

```
cor_network_graph(icgc_data,R,TS=0.3,legend=c(rep("RNAseq",5),rep("Proteins",5),rep("Mutations",5)))
```



Simulation

Our package is also able to simulate data distributed according to the Gaussian copula model. Two functions enable us to generate two types of correlation matrices: block-wise and sparse. The diag_block_matrix function enables the user to get a block-wise correlation matrix. It takes as arguments a vector containing the block sizes and a vector containing the coefficients of each block. An example is shown below.

```
R <- diag_block_matrix(c(3,2),c(0.4,0.8))
```

1.0	0.4	0.4	0.0	0.0
0.4	1.0	0.4	0.0	0.0
0.4	0.4	1.0	0.0	0.0
0.0	0.0	0.0	1.0	0.8
0.0	0.0	0.0	0.8	1.0

The matrix_gen function enables the user to generate a sparse correlation matrix of initial sparsity parameter γ, which has to be specified. It is based on the Cholesky decomposition where a lower triangular matrix of

sparsity γ is generated before being multiplied by its transpose in order to obtain the final matrix. Note that the initial parameter is not equal to the final parameter, which is also returned by the function. In the example below, the first element of the list is the resulting matrix, and the second element of the list is the final sparsity parameter.

```
R <- matrix_gen(5,0.81)
```

The CopulaSim function enables the user to generate a data set which CDF can be expressed as a Gaussian copula of correlation matrix R (to be specified). In the example below, we first generate a block diagonal correlation matrix R and then generate the data set. Then, CopulaSim takes as arguments the number of observations, the correlation matrix of the copula, a vector containing the probability distributions and their parameters, the number of repetitions of each distribution, and enables the user to randomize their order. It returns a list of three elements: the data frame containing the generated data, the correlation matrix, and the permutation realized on the rows and columns of R order after randomization.

```
R <- diag_block_matrix(c(3,5,2),c(0.7,0.3,0.5))
CopulaSim(5,R,c("qnorm(0,1)","qexp(0.5)","qbinom(4,0.8)"),c(5,3,2),random=TRUE)
#> [[1]]
#>           X1          X2         X3          X4         X5        X6        X7
#> 1  0.8302497  2.11159882 -0.4420444  0.03795749 -0.2027720 1.3933248 1.1016845
#> 2 -0.2280789 -0.49059467  0.6741935 -0.66448769 -0.1546064 0.3468675 2.8779010
#> 3  0.4216491  1.71449368 -0.7903717  0.25992966 -0.2676998 0.8384409 0.1904884
#> 4 -1.8320914 -0.81387121  0.1769601 -0.68142751 -0.6785899 0.9950933 1.4186486
#> 5  1.2641885 -0.08896631  1.2679297  2.27640110  1.8373691 2.6813502 2.0255898
#>          X8 X9 X10
#> 1 2.7959501  3   1
#> 2 1.1340180  3   3
#> 3 0.2271502  3   3
#> 4 2.2635732  3   3
#> 5 3.3180732  4   3
#>
#> [[2]]
#>       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
#>  [1,]  1.0  0.3  0.3  0.3  0.0  0.3  0.0  0.0  0.0   0.0
#>  [2,]  0.3  1.0  0.3  0.3  0.0  0.3  0.0  0.0  0.0   0.0
#>  [3,]  0.3  0.3  1.0  0.3  0.0  0.3  0.0  0.0  0.0   0.0
#>  [4,]  0.3  0.3  0.3  1.0  0.0  0.3  0.0  0.0  0.0   0.0
#>  [5,]  0.0  0.0  0.0  0.0  1.0  0.0  0.7  0.0  0.0   0.7
#>  [6,]  0.3  0.3  0.3  0.3  0.0  1.0  0.0  0.0  0.0   0.0
#>  [7,]  0.0  0.0  0.0  0.0  0.7  0.0  1.0  0.0  0.0   0.7
#>  [8,]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.5   0.0
#>  [9,]  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.5  1.0   0.0
#> [10,]  0.0  0.0  0.0  0.0  0.7  0.0  0.7  0.0  0.0   1.0
#>
#> [[3]]
#>  [1] 10  5  2  6  8  4  3  1  9  7
```

Additionally, the gauss_gen function, which is used in CopulaSim, generates the latent Gaussian variables linked by the correlation matrix R. Its only arguments are the correlation matrix R and the number of observations.

```
latent_data <- gauss_gen(R,10)
```

X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
0.34	0.63	0.48	0.55	0.62	0.65	0.32	0.05	0.58	0.63
0.26	0.62	0.45	0.83	0.24	0.68	0.25	0.41	0.99	1.00

0.76	0.31	0.88	0.20	0.30	0.47	0.66	0.73	0.03	0.09
0.98	0.97	1.00	0.30	0.12	0.52	0.37	0.82	0.49	0.05
0.25	0.40	0.37	0.79	0.67	0.50	0.91	0.28	0.06	0.31
0.24	0.94	0.78	0.55	0.07	0.12	0.92	0.23	0.19	0.17
0.95	0.83	0.95	0.01	0.02	0.20	0.03	0.06	0.48	0.40
0.22	0.68	0.85	0.05	0.93	0.22	0.39	0.54	0.47	0.62
0.02	0.10	0.01	0.46	0.50	0.93	0.76	0.36	0.12	0.51
0.22	0.54	0.75	0.64	0.75	0.46	0.39	0.10	0.97	0.52

References

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

Mazo, Rau, and Karlis. 2023. "A Randomized Pariwise Likelihood Method for Complex Statistical Inferences." *Journal of American Statistical Association*.

Tomilina, Mazo, and Jaffrézic. 2024. "Mixed Copula-Based Models for Semi-Parametric Network Inference: An Application to Multi-Omics Data."

Zhang, J., R. Bajari, and Andric D. 2019. "The International Cancer Genome Consortium Data Portal." *Nat Biotechnol.* https://doi.org/doi:10.1038/s41587-019-0055-9.