Vine Copulas based modeling
Generating samples for high dimensional data

HOANG Caroline [*1]

Abstract

In some domains such as medical one and in particular EEG, having data can be expensive and generating it may be the way. We need a model to create artificial data from the same distribution as the original data. (Vine) Copulas are the best tool to find this model instead of using Chow-Liu Trees that are maximum-weight spanning trees of mutual information which its estimation is notoriously difficult. However, using Vine copulas for high dimensional data is costly and exponentially scale with the dimension d. The time computation is $d!2^{(d-2)(d-3)/2-1}$. We aim to reach the ideal trade-off between computation time efficiency and model/generated data quality by finding an ideal structure.

Keywords: Bicopula, Dependence, Distribution, Probabilisitic trees, Structure

1. Introduction

Vine copula is a probabilistic trees model with higher-order factorization of probability distribution than Chow-Liu tree which is a 2nd order factorization which provide better approximation. Building completely from scratch a vine copula model contains 3 steps: Find the vine tree structure, the pair copula families and estimate the corresponding parameters for each edges. As Vine copula is computationally costly for high dimensional data, exploring every tree structure is heavy and to avoid it, we will build the tree structure following a specific criteria. As finding the pair copula families and its parameters scale with the depth and number of trees, we will consider pruning it to decrease time computation. However, this may leads to a model that doesn't respect the real data distribution. Therefore, we will introduce different kind of model evalutations.

2. Preliminaries

We begin by defining a copula.

Definition 1 (Copula). d-dimensional copula is a function $C : [0,1]^d \to [0,1]$ with the following properties: (i) For every $u = (u_1; ...; u_d) \in [0,1]^d$,

$$C(u) = 0 \text{ if at least one coordinate of u is 0,}$$

and $\forall j \in [|1,d|]$,

$$C(1,..,1,u_j,1,...,1) = u_j$$

(ii) C is d-increasing (see Nelsen (2006)).
d-dimensional copulas are multivariate distribution functions with uniform margins.

Theorem 1 (Sklar). Let $X_1, ..., X_n$ be a random sample of a d-dimensional random vector $\mathbf{X}$ with distribution function F and continuous, univariate margins $F_1, ...F_d$. There exists a unique copula C on $[0,1]^d$ such that

$$F(\mathbf{x}) = C(F_1(x_1), ..., F_d(x_d))$$

for $\mathbf{x} = (x1, ..., xd) \in \mathbb{R}^d$ with associated density $f(x_1,..,x_d) = c(F_1(x_1),..,F_d(x_d))f_1(x_1)...f_d(x_d)$ with copula density c.

Definition 2 (Non-parametric estimators for margins F_j). Let n be the number of observation and $j \in [|1,d|]$

$$\widetilde{F}_j(t) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{X_{ij} \le t\}}$$

Thus we can define the pseudo-observation

Definition 3 (Pseudo-observation). $\forall i \in [|1,n|], \forall j \in [|1,d|]$,

$$U_{i,j} = \frac{R_{ij}}{n+1} = \frac{n}{n+1}\widetilde{F}_j(X_{ij})$$

where R_{ij} is the rank of X_{ij} and the scaling factor $\frac{n}{n+1}$ to make the velues in $[0,1]^d$ needed for maximum pseudo-likelihood estimation and thus parameters estimation.

Definition 4 (Empirical copula). The empirical copula is defined by $\forall \mathbf{u} \in [0,1]^d$

$$C_n(\mathbf{u}) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{\mathbf{U}_i \le \mathbf{u}\}} = \frac{1}{n}\sum_{i=1}^{n}\prod_{i=j}^{d} \mathbf{1}_{\{U_{ij} \le u_{ij}\}}$$

where U_{ij} is the pseudo-observation.

Theorem 2 (Mutual information of a copula). Let c be the copula density associated to a copula C. Then the MI of a copula is defined by:

$$I = \int_{[0,1]^d} c(u) log c(u) du$$

(Jian & Zengqi, 2018)

Result 1 Given the work of (Gao & al., 2015), we can approximate the mutual information for strongly dependent Variables applied to multivariate data using KSG estimator method defined by:

$$I_{KSG,k} = (d-1)\psi(N) + \psi(k) - (d-1)/k - \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{d}\psi(n_{x_j}(i))$$

where k number of neighbours, ϕ is the digamma function and n_{x_j} is the number of points at a distance less than or equal to $\epsilon_{i,k}^{j}/2$ which is the maximum distance between the point number i and its k nearest neighborhood in the jth dimension.

3. Methods

Our work consists mainly of 4 things. The first one is building the structure that we will use in order to fit the copula to the data. The second is for the given model, implementing an algorithm for sampling. Then, we have to figure out how our model is good by doing model evaluation. Finally, as the copula is within $[0,1]^d$, which is the copula domain, we need to transform the generated data to the marginal space.

3.1. Building the structure

The structure of a copula satisfies the following statement:

- There are at most $d-1$ trees
- For a given tree each edge becames a node in the next tree
- Let $N = \{N_1, ..., N_d\}$ and edge set $E = \{E_1, ..., E_{d-1}\}$ by associating each edge $e = j(e), k(e)|D(e) \in E_i$ with a bivariate copula density $c_{j(e),k(e)|D(e)}$. Let $\mathbf{X}_{D(e)}$ be the sub random vector of $\mathbf{X} = (X1, ..., Xd)$ indicated by the indices contained in $D(e)$ and let $f_r, r \in [|1,d|]$ be the marginal densities. The associated copula density is (Brechmann, 2010) :

$$f(\mathbf{x}) = \prod_{r=1}^{d} f_r(x_r) \times \prod_{i=1}^{d-1}\prod_{e in E_i} c_{j(e),k(e)|D(e)}(F(x_{j(e)|\mathbf{x}_{D(e)}})F(x_{k(e)|\mathbf{x}_{D(e)}})) \quad (1)$$

This structure can be represented within a matrix, either by a R-vine-structure when we encounter a R-vine-structure or by the following matrix $M \in [|1,d|]^{d\times d}$:

- $M[i,j]_{i\in[|1,d|],j\in[|1,d|]}$ where i is the variable and j the tree level. The value corresponds to the variable involved in the couple $(i, M[i,j])$ in the tree j
- $$\begin{pmatrix} m_{1,1} & ... & ... & ... & m_{1,d-1} & 0 \\ ... & ... & ... & m_{d-1-i,i+1} & 0 & 0 \\ ... & ... & m_{d-1-i+1,i} & 0 & 0 & 0 \\ ... & m_{d-1-i+2,i-1} & 0 & ... & 0 & 0 \\ m_{d-1,1} & 0 & ... & ... & 0 & 0 \\ 0 & 0 & ... & ... & 0 & 0 \end{pmatrix}$$

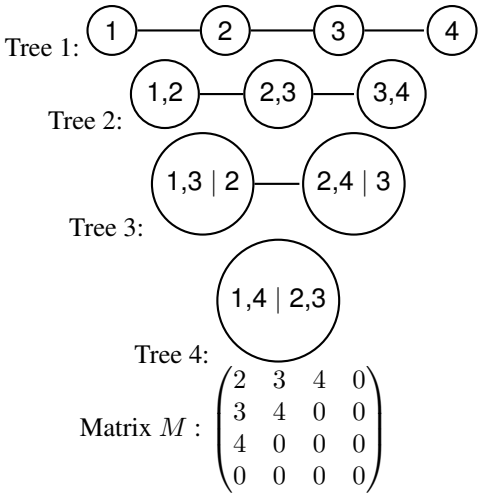


Figure 1. Example of structure for 4 variables and its matrix M.

The structure above is a R-vine structure.
In order to choose each pair, we will take the Kendall's τ[1] criteria that represents the correlation between those two variables. We will also consider the mutual information criteria. We will construct each tree by taking the pair in which the absolute Kendall's τ is the biggest to the lowest value. The resulting structure is a R-vine-structure.
However, this structure is still costly as we need to find the pair copula families and their parameters. Thus, we

[1] https://en.wikipedia.org/wiki/Kendall_rank_correlation_coefficient

will consider two type of simplification based on threshold. Threshold is the processus of setting the Kendall's τ to 0 under a certain threshold value. In this case, we consider those pairs of variables independent.

Pruning at first level By setting some Kendall's τ to 0 we don't have to draw an edge between those nodes in the first level which implies having severals independent trees and then severals independent copulas with a R-vine structure. The consequence for the copula density is that it's equal to the product of the copula density of those sub independant copulas. Applying the example from (Fig.1) by cutting the edge [2,3] gives :

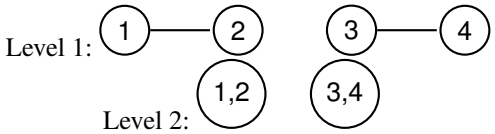


Figure 2. Example of structure for 4 variables with pruning at first level

This decreases time computation as it's the same as computing two independent copula models with lower dimension.

Pruning at every level This method remove edge at each level which lead to many pairs that we don't have to look for its family and associated parameters. It is faster than pruning at first level but the resulting structure isn't a R-vine structure and we don't have independent copula that we can compute separately. Applying the example from (Fig.1) by cutting the edge [3,4] for the first level and the edge [(1,2), (2,3)], we have :

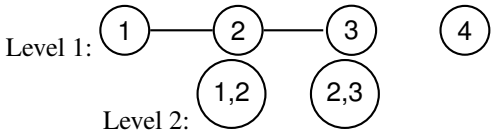


Figure 3. Example of structure for 4 variables with pruning at all level

3.2. Generating samples

We will use an adapting version of the sampling algorithm[2] for R-vine structure to our structure representation. We proceed by generating samples from variable d to 1 by crossing trees from top to bottom.

Algorithm 1 Generating samples from personalized structure

Input: n number of samples, M structure matrix associated the copula model
$A \in [0,1]^{n\times d\times d}$, $B \in [0,1]^{n\times d\times d}$, $v \in [0,1]^{n\times d\times d}$, $u \in [0,1]^{n\times d}$, $w \in [0,1]^{n\times d}$ random uniform $[0,1]$
$A_{d,d,:} = B_{d,d,:} v_{d,d,:} = u_{:,d} \leftarrow w_{:,d}$
for $i = 2$ **to** d **do**
  try:
  $i_{\_} \leftarrow d - i + 1$
  $N \leftarrow M_{i_{\_},:}$ {Gets the list of variables paired with i for each tree}
  $k \leftarrow length(N)$
  for $j = 1$ **to** k **do**
    $value \leftarrow v_{i+1:,N_{k-j+1},:idx}$ {idx is the the greatest variable in which its value isn't 0}
    $v_{i,N_{k-j+1},:} \leftarrow C^{-1}_{i_{\_}|N_{1:k-j+1}}(A_{i_{\_},j_{\_}+1,:} \mid value)$
    $A_{i_{\_},j_{\_},:} \leftarrow v_{i,N_{k-j+1},:}$
  end for
  $u_{:,i_{\_}} = v_{i_{\_},i_{\_},:} = B_{i_{\_},i_{\_},:} \leftarrow A_{i_{\_},i_{\_},:}$
  for $j = k$ **to** 1 **do**
    $j_{\_} \leftarrow d - j - |i - k - 1|$
    $value \leftarrow v_{N_{k-j+1},i_+1:,:}$ {idx is the the greatest variable in which its value isn't 0}
    $v_{N_{k-j+1},i,:} \leftarrow C_{N_{1:k-j+1}|i_{\_}}(value \mid B_{i_{\_},j_{\_}-1,:})$
    $B_{i_{\_},j_{\_},:} = v_{N_{k-j+1},i_{\_},:}$
  end for
  catch (variable independent) :
  $u_{:,d-i-1} = v_{d-i-1,d-i-1,:} = A_{d-i-1,d-1,:} = B_{d-i-1,d-1,:} \leftarrow w_{:,d-i-1}$
end for
Output: u the simulated data in copula space.

[2]https://lewiscoleblog.com/monte-carlo-methods-3

3.3. Model evaluation

We want to assess whether it is reliable and adequate for future use, i.e., evaluate it in absolute terms. Model evaluation therefore proceeds by comparing characteristics of the observed data, which was used for model specification, with simulated observations from the specified copula model. We will consider mostly graphical evaluation between pairs of variables such as contour plot, general QQ-plot, empirical copula distribution for lower and uper tail and we will consider the mutual information metric, the mahalanobis distance and a statistical test such a K-S test.

QQ plot QQ-plot is used to compare the distribution of the observed and theorical distribution. This plot allows to compare two empirical cdf's, in our case the empirical cdf's of the observed and simulated data, where the observed data comes from the unknown true distribution and we want to investigate if the distribution induced by copula model is close to this true distribution. We compute :

$$w_i^K = \frac{1}{n}\sum_{j=1}^{n} \mathbf{1}_{u_{jr}^K \leq u_{is}^K, u_{js}^K \leq u_{ir}^K}$$

and

$$w_i = \frac{1}{n}\sum_{j=1}^{n} \mathbf{1}_{u_{jr} \leq u_{is}, u_{js} \leq u_{ir}}$$

for $i \in [|1, n|]$ and $r, s \in [|1, d|]$ where n is the number of obersation and (r, s) a pair of variable.

Empirical copula estimation and tail dependence If one is particularly interested in an accurate modeling of the joint tail behavior of variables, it might be interesting to consider the empirical copula distribution functions in the tails,

$$C_n(u_1, ..., u_d) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{U_{i1} \leq u1, ..., U_{id} \leq ud}$$

the pseudo-samples $U_1, ..., U_n$. These can be considered as samples from the underlying copula C. In this case it's interesting to look at bivariate one So we compute $C_n(\alpha, \alpha)$ (lower tail) and $C_n(1-\alpha, 1-\alpha)$ (upper tail) for $\alpha \in [0, 0.1]$. It allows us to observe the tail dependances, if the simulated one respects the behavior from observed one that we can see by computing the bivariate plot.

Bivariate scatter plot It plots in copula space the pair of variables. It tells the form of dependence between them, especially the tail.

Data mean distribution We need alternative quantities. The most commonly used one is given by the mean of the copula data over its d components $S_i^K = \frac{1}{d}\sum_{r=1}^{d} u_{ir}^K$, and $S_i = \frac{1}{d}\sum_{r=1}^{d} u_{ir}$ for all $i \in [|1, n|]$ where n is the number of samples/observation. The appropriateness of model(K) can then be assessed by comparing histograms and empirical quantiles based on set of $\{S_i^K, i = 1, ..., n\}$ and $\{S_i, i = 1, ..., n\}$ We can extand the formula by adding personalize weight for each dimension. This approaches may not be in the copula space meaningful for independence pairs which means that we need to be careful about the variables chosen since independent induces an unirform distribution.

Statistical test We will use the KS-test on single variables in the copula to assess the similarity between the generated data and the observed data at the variable level. As it's only for single variable, we can choose a threshold for p-value for interpretation and count the number of variables where the hypothesis that those distributions (obeserved and simulated) are similar. We will use as well a metric called the Mahalanobis distance, which measures the distance between two multivariate datasets, taking into account the covariance structure. The smaller the value is, the better it is as it means that the 2 datasets are similar as the distance is closer.

3.4. Transformation from copula space to marginal space

Our idea for this transformation is to use the actual dataset to reverse the process of the computation of non-parametric estimator for margins. Given its definition, $\widetilde{F}_j$ is an increasing function but not necessarly strict. Thus, it doesn't ensure the existence of an invert function. However, in the domain of the dataset, we can find an unique value $\widetilde{F}_j(t)$ for t an observed value. As, we can't define the invert function, two choices can be considered: for a given value u_{ij} in the copula space, we find a set of value s where $\forall x_{ij} \in s, u_{i,j} = \frac{n}{n+1}\widetilde{F}_j(x_{ij})$ and choose one of the value of the set randomly depending of the frequence. The drawback of this method is that we are limited in the dataset values, so we will consider the other solution which is performing a monotone cubic interpolation[3] over the dataset used to fit the model which preserves the monoticity of the dataset being interpolated. Thus, for each simulated data in the copula space, we can find it's corresponding value in the marginal space. The issue of this method is that the interpolation doesn't work well with discrete value and in particular binary one. To handle it, we will use the nearest value in the copula space that is associated to an observed value.

4. Experiments

Code provided here:
We implement our work with the help of an existing toolbox

[3] https://en.wikipedia.org/wiki/Monotone_cubic_interpolation

Algorithm 2 Copula space to marginal space

Input: $X \in \mathbb{R}^{n \times d}$ the observed data, $F \in [0,1]^{n \times d}$ matrix where $F_{i,j} = \widetilde{F}_j(X_{ij})$, $U \in [0,1]^{n \times d}$ the simulated data.
for $j = 1$ **to** d **do**
  $S \leftarrow \{X_{1,j}, ..., X_{n,j}\}$
  $MCI_j \leftarrow$ Performing a monotone cubic interpolation over
  for $i = 1$ **to** n **do**
    $M_{i,j} \leftarrow MCI_j^{-1}(U_{i,j})$
    if $M_{i,j} > max(S)$ or $M_{i,j} < min(S)$ **then**
      $M_{i,j} \leftarrow x \in \{X_{1,j}, ..., X_{n,j}\}$ where $\frac{n}{n+1}\widetilde{F}_j(X_{ij})$ is the closest value to $U_{i,j}$
    end if
  end for
end for
Output: $M \in \mathbb{R}^{n \times d}$ the simulated data in marginal space.

in python 'vinecopulib'[4] and we create an another library.

4.1. Data

The dataset used for the experiments is a private time series dataset that comes from miles. We have access to the data of sensors related to mills such as quantity, grinding table speed, hot gas generator, clamping temperature, water injection, classifier speed, fan speed and flap positions. There are over 226 variables. We will use 27402 instances for the experiments. We will only look at a revelant subset of 49 columns including 13 variables explained below where we know the signification for interpretability. We suppose that our variables are continious. Dealing with discrete variables is not explored and it's not useful.

4.2. Experiment methods

We will build four copula models: vine copula using the existing tool that search for the best structure itself with automatic threshold and truncation, vine copula without pruning, with pruining at first level and at every level. We will choose a threshold value equals 0.10 by looking at (Fig.) which will set the pair as independant. This value will only be applied for first level. We will choose 0.01 for all level as it's a sensitive parameters. We will display the distribution of the observed and simulated data by doing the mean for each instance in the marginal space as in the copula space there are discrete data simulated as continious and independently which make the distribution uniform (Fig. 4). For the five first model we use 32 threads to compute faster. We will display the time it took the model to be build which include: structure to search, family to find for each pairs and its parameters. Then we simulate the same amount of data from the observed on to do a two-samples KS-test on each variable and count the number of tests that is against the hypothesis of different distribution. We will take $\alpha = 0.05$. Then we will perform the Mahalanobis distance for each point and perform a boxplot to see if the point corresponds to the observed one or if it's totally different. The last case implies a bad model.

[4] https://vinecopulib.github.io/pyvinecopulib/

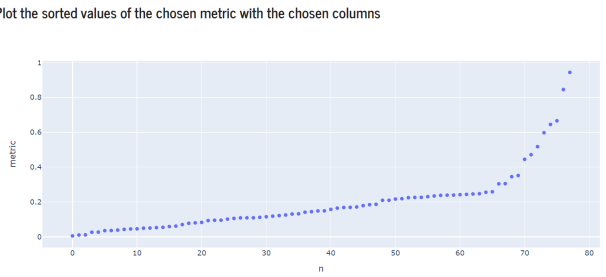


Figure 4. Kendall's τ of each pairs sorted by value

4.3. Results

We can notice that the execution time difference between each model. The pyvinecopulib model takes less time by choosing simplified model and seems to have good result accoding to the table. A full search pyvinecopulib model one takes way longer (more than 3200 min). However, the bivariate plot says otherwise and often end up with a relationship close to independant one. As the time computiation increasing expotionally with the dimension, we can see how the execution time is reduced by adding pruning. We want to focus in the performance and the difference between the first model and the others.
The KS-test told us that there are for most 4 variables in which the simulated and observed one have similar distribution. The other 9 may be due to the fact that they are simulated as independant which generate uniform data, thus for discrete variable the KS-test will not verify the hypothesis.
Pruning at all level may not be the right choices as we are missing information. We can see it in the (Fig. 5), the bivariate plot doesn't correspond much than the others with the observed data. The QQ-plot confirms that it doesn't reflect the dependency of the observed data. So pruning at first level is a better choice and takes here twice less time than without pruning for a threshold of 0.10, so we ignore most of pair combinaisons.
We can see the difference between pruning and without pruning in Fig.6. We don't take the Hot gazs generator but the time in second instead. Those 2 variables in the pruning structure are in different trees which mean they are independant. This is reflected in the pyvinecopulib model and the

Variable	Description	Unit
Feed quantity	A51-WF01.FC01	t/h
Grinding table speed	A56-MD01.SC01	% of nominal speed
Hot gaz generator	A56-HG01.XC01	% of maximalburner load
Climping pressure	A56-HS01.PZ13, A56-HS01.PZ24	bar
Water injection	A56-WI01.FC01	m^3/h
Classifier speed	A56-SR01.SC01	% of nominal speed
Fan speed	A56-FN02.SC01	% of nominal speed
Flap positions	A56-DA01.ZC01, A56-DA02.ZC01, A56-DA03.ZC01	% of opening
Date	time	timestamp

Table 1. Variables signification

We devide the time into year, months, day and the day time in second and keep variables that are significant (here the day time)

Type model	Execution time	KS-test	Upper fence	Lower fence	Mediane	Q1	Q3	Min	Max
Pyvinecopulib	36 min 58 s	34/49	8.07	4.61	6.32	5.90	6.77	4.55	14.22
Without pruning	176 min 48	30/49	43.31	14.22	17.10	27.58	20.31	14.22	110.37
With pruning first level	114 min 54	31/49	37.01	7.14	15.24	11.70	21.83	7.17	106.57
With pruning all level	60 min 45	26/49	1767.50	7.92	433.12	184.55	817.98	7.92	2531.48

Table 2. Performances of each model

pruning ones. However, without pruning the plot is different and reflects more the observed data. This small changes can be ignored, depending of how complex and accurate we want our model to be. This is what the threshold is here for. We need to find a trade-off between time computation and complexity of our model and accuracy. Here, we can ignore the dependance of the time and the feed quantity if we are in big dimension.

5. Related work

For simplificity, we create a prototype/application on Dash python that use our own package to perform and provide data visualisation, pre-processing, structure searching,modelisation of the structure as a graph, generating samples and model evaluation.

6. Conclusion

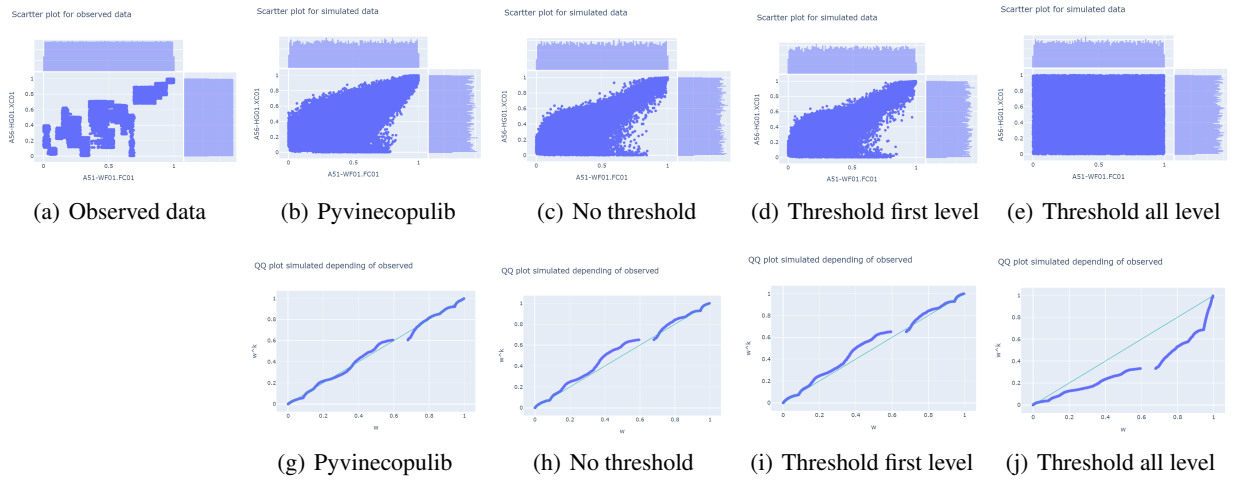
We have presented a modelisation for high dimension data using vine copula by fixing a structure to reduce significantly time computation. The parameter threshold is a sensitive parameter that may work for very small value but the result isn't the best for a pruning at all level that is similar to using a Gaussian copula which is way faster to compute. Therefore, if we want to use the threshold, we may consider pruning only the first level.

7. Further work

We need to find another method for model evaluation in especially for all variables. We mostly can only compare by pairs with graphical tools the observed and simualted data. Only the distribution of the mean for each instance can be used as a global view. We can consider mutual information for copula. It can be used to know how good our model is and compare each other models. We can use the product of the density copula of each independent trees and still using the existing library. However, we need to find a way for threshold at all level. As for the gaussian one, we may have access to its density and compute it for comparison. We also need to compute the mutual information from the original data. Multivariate mutual information is difficult to compute and we can use an approximation using the KSG estimator. However this method is significant for strong dependency.
How the structure thus the model is different for no threshold if we use the mutual information criteria?
Our implementation isn't optimal enough, especially the computation from copula space to marginal space due to lack of python appropriate function. It takes 45min for 6400 instances. Paralellized some task is a way.

References

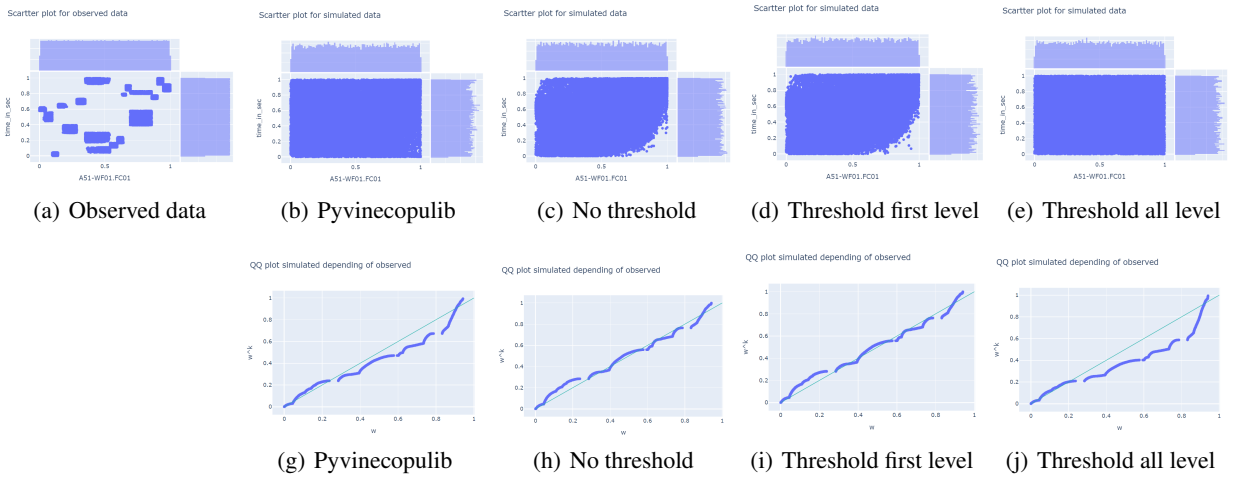
Brechmann, E. C. *Truncated and simplffed regular vines and their applications*. PhD thesis, Technische Universitat Munchen, 2010.



(a) Observed data (b) Pyvinecopulib (c) No threshold (d) Threshold first level (e) Threshold all level

(g) Pyvinecopulib (h) No threshold (i) Threshold first level (j) Threshold all level

Figure 5. Bivarite plot in copula space and is attached QQ plot for the variable A56-HG01.XC01 depending of A51-WF01.FC01



(a) Observed data (b) Pyvinecopulib (c) No threshold (d) Threshold first level (e) Threshold all level

(g) Pyvinecopulib (h) No threshold (i) Threshold first level (j) Threshold all level

Figure 6. Bivarite plot in copula space and is attached QQ plot for the variable time_in_sec depending of A51-WF01.FC01

Gao, S. and al. *Ecient Estimation of Mutual Information for Strongly Dependent Variables*. PhD thesis, Information Sciences Institute and University of Southern California, 2015.

Jian, M. and Zengqi, S. *Mutual Information Is Copula Entropy*. PhD thesis, Tsinghua University, 2018.