Research Article **Open Access**

Andreas Masuhr and Mark Trede

Bayesian estimation of generalized partition of unity copulas

https://doi.org/10.1515/demo-2020-0007
Received February 18, 2020; accepted June 8, 2020

Abstract: This paper proposes a Bayesian estimation algorithm to estimate Generalized Partition of Unity Copulas (GPUC), a class of nonparametric copulas recently introduced by [18]. The first approach is a random walk Metropolis-Hastings (RW-MH) algorithm, the second one is a random blocking random walk Metropolis-Hastings algorithm (RBRW-MH). Both approaches are Markov chain Monte Carlo methods and can cope with flat priors. We carry out simulation studies to determine and compare the efficiency of the algorithms. We present an empirical illustration where GPUCs are used to nonparametrically describe the dependence of exchange rate changes of the crypto-currencies Bitcoin and Ethereum.

Keywords: copulas, partition-of-unity, Bayesian estimation, cryptocurrencies

MSC: 62H05, 62H12, 62G07, 62P20, 91B05

1 Introduction

Copulas are attractive tools to model multivariate dependence structures in a flexible way. They are widely applied in fields where it is crucial to have a good working model of the joint distribution, e.g. in finance or portfolio management. In the past years, various families of copulas have been suggested ranging from the classic families that are generated by inversion of multivariate distributions like the Gaussian or t-copula to the class of Archimedean copulas that use generator functions to create copulas. To add even more flexibility in higher dimensions all these copula families can be mixed using vines [3] or, within the class of Archimedean copulas, bivariate copulas can be organized hierarchically [10, 20].

A disadvantage of parametric copula families is the obvious risk of misspecification. A misspecified copula will result in misleading statistical inference. An obvious alternative to parametric copula models is to use nonparametric copulas that directly capture the characteristics of the data. The simplest way is, of course, the empirical copula. Another nonparametric copula is the Bernstein copula [1] which builds on Bernstein polynomials to smooth the estimated copula density. Bernstein copulas can be estimated by the EM algorithm [7].

Recently, [18] and [15] developed a more general family of nonparametric copulas, the so called Generalized Partition of Unity Copulas (GPUC). In contrast to Bernstein copulas, the copula density over the unit hypercube is no longer approximated by a finite mixture of functions but by an infinite mixture. Bernstein copulas are nested as a special case of GPUCs. [16] expand the concept of GPUCs to continuous partitions of the unit hypercube and hence, a continuous parameter space. In this paper we tackle the question how to fit GPUCs resulting from a discrete partition of unity.

Andreas Masuhr: Institute of Econometrics, Department of Economics, University of Münster, Am Stadtgraben 9, 48143 Münster, Germany
Mark Trede: Institute of Econometrics, Department of Economics, University of Münster, Am Stadtgraben 9, 48143 Münster, Germany, E-Mail: mark.trede@uni-muenster.de

[8] propose an MCMC based estimation approach suited for all copulas that can be parameterized by a doubly stochastic matrix. To fit the copula, they indirectly draw the doubly stochastic parameter matrix from its Hilbert space representation in a random walk Metropolis-Hastings sampler.

This paper proposes and applies Bayesian methods to estimate GPUCs. The rest of the paper proceeds as follows. Section 2 reviews the Generalized Partition of Unity Copulas and their properties. Section 3 presents two Bayesian Markov chain Monte Carlo algorithms to sample from the posterior distribution of the GPUC parameters. In section 4, the algorithms are compared in a simulation study. Our substantive contribution is an empirical application in section 5. We consider the joint return behaviour of two of the leading crpytocurrencies: Bitcoin and Ethereum. We show how GPUCs can be applied to nonparametrically model the dependence between their exchange rates. Finally, section 6 concludes.

2 Generalized Partition of Unity Copulas

The concept of Generalized Partition of Unity Copulas (GPUC) was first introduced by [18] and applied in the context of risk management [17]. We briefly review the definition and main properties of GPUCs. A GPUC is a bivariate copula[1] defined by the density function

$$c(u,v) = \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} M_{ij}\frac{\phi_i(u)}{\alpha_i}\frac{\varphi_j(v)}{\gamma_j}, \tag{1}$$

with $\alpha_i = \sum_{j=1}^{\infty} M_{ij} = \int_0^1 \phi_i(u)du$ and $\gamma_j = \sum_{i=1}^{\infty} M_{ij} = \int_0^1 \varphi_j(v)dv$. The generating functions $\phi_i(u)$ and $\varphi_j(v)$ can be considered as probability mass functions of discrete random variables over the positive integers with parameters u and v, and hence, $\sum_{i=1}^{\infty}\phi_i(u) = \sum_{j=1}^{\infty}\varphi_j(v) = 1$. For simplicity we consider GPUCs with $\phi_i(u) = \varphi_i(u)$ (and hence $\alpha_i = \gamma_i$) for all i, i.e. copulas with identical generating functions. Note that this does not imply symmetry since the parameter matrix M need not be symmetric. The generating function may have no additional parameter, but it may also depend on one or more parameters. We denote the generic parameter (vector) as θ and write $\phi_{\theta,i}$ for the generating function.

The independence copula is nested by GPUCs if $M_{ij} = \alpha_i\alpha_j$ for all i, j since

$$c(u,v) = \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} M_{ij}\frac{\phi_{\theta,i}(u)}{\alpha_i}\frac{\phi_{\theta,j}(v)}{\alpha_j} = \sum_{i=1}^{\infty}\phi_{\theta,i}(u)\sum_{j=1}^{\infty}\phi_{\theta,j}(v) = 1$$

for $0 \le u, v \le 1$.

Equation (1) denotes the fully parametrized form GPUC with an infinitely dimensional parameter matrix M. For modelling and estimation purposes, we impose the following restrictions on the parameter matrix M: For $i > m$ or $j > m$

$$M_{ij} = \begin{cases} \alpha_i & \text{if } i = j \\ 0 & \text{else} \end{cases}$$

for some (sufficiently large) m. No restrictions are imposed for $i, j \in \{1, \dots, m\}$. The reduced form copula density is

$$c(u,v) = c_A(u,v) + c_B(u,v) \tag{2}$$

where

$$c_A(u,v) = \sum_{i=1}^{m}\sum_{j=1}^{m} M_{ij}\frac{\phi_{\theta,i}(u)}{\alpha_i}\frac{\phi_{\theta,j}(v)}{\alpha_j}$$

1 While it is possible to define GPUCs for arbitrary dimensions [17], we refrain from doing so because the estimation algorithms become computationally very costly for higher dimensions.

$$c_B(u,v) = \sum_{i=m+1}^{\infty} \frac{\phi_{\theta,i}(u)\phi_{\theta,i}(v)}{\alpha_i}.$$

In general, reduced form GPUCs do not nest the independence copula and, hence, care must be taken when specifying the size of the parameter matrix M before estimation.[2]

The aim of this paper is to estimate the parameters of a reduced form GPUC. Obviously, the natural parametrization of GPUCs is in terms of θ and M. However, it turns out that this choice is inconvenient for estimation since the row and column sums of M may depend on θ. We therefore decided to parametrize the GPUCs in a different fashion.

Let $\mathcal{D}_m$ denote the space of $m \times m$ doubly stochastic matrices, and define

$$\mathcal{M}_\theta = \left\{ M : M_{ij} \geq 0 \; \forall i,j; \; \sum_{i=1}^{m} M_{ij} = \alpha_j; \; \sum_{j=1}^{m} M_{ij} = \alpha_i \right\}$$

where the row and column sums may depend on θ. For given m, we parametrize a GPUC by θ and a doubly stochastic matrix $D \in \mathcal{D}_m$. For any value of θ, each element of $\mathcal{D}_m$ can be mapped to a unique element of $\mathcal{M}_\theta$ by the Sinkhorn algorithm [21]:

1. For $i = 1, \ldots, m$ determine α_i.
2. For $r = 0$ initialize $M^{(r)} = D$. For iteration $r + 1$ do the following steps.
3. For each row $i = 1, \ldots, m$, compute for $j = 1, \ldots, m$

$$\bar{M}_{ij} = M_{ij}^{(r)} \cdot \frac{\alpha_i}{\sum_{k=1}^{m} M_{ik}^{(r)}}.$$

4. For each column $j = 1, \ldots, m$, compute for $i = 1, \ldots, m$

$$M_{ij}^{(r+1)} = \bar{M}_{ij} \cdot \frac{\alpha_j}{\sum_{k=1}^{m} M_{kj}^{(r+1)}}.$$

5. Check if all row sums and column sums are (close to) the desired marginal values $\alpha_1, \ldots, \alpha_m$. If they are, return $M^{(r+1)}$. Otherwise, continue the next iteration at step 3.

In this paper, we consider two generating functions: the probability mass functions of the binomial distribution (over support $\{1, 2, \ldots, m\}$) and the negative binomial distribution (over support $\{1, 2, \ldots, \}$).

2.1 Binomial generating function

The generating function has no additional parameter but only depends on m,

$$\phi_i(u) = \binom{m-1}{i-1} u^{i-1}(1-u)^{m-i}$$

for $i = 1, \ldots, m$ and $\phi_i(u) = 0$ for $i > m$. Note that we shifted the usual support of the binomial distribution such that the smallest possible value is 1. This generator implies that (for $i = 1, \ldots, m$)

$$\alpha_i = \frac{1}{m}.$$

The resulting copulas are Bernstein copulas (see also [19]) with density

$$c(u,v) = m^2 \sum_{i=1}^{m} \binom{m-1}{i-1} u^{i-1}(1-u)^{m-i} \sum_{j=1}^{m} M_{ij} \binom{m-1}{j-1} v^{j-1}(1-v)^{m-j}.$$

Since $\alpha_i = 0$ for $i > m$ this equals also the reduced form copula.

2 The reduced form GPUCs can closely approximate the independence copula if the α_i are close to zero for $i > m$. If they are zero, the independence copula is nested.

2.2 Negative binomial generating function

The generating function has a single shape parameter $\beta > 0$. For $i = 1, 2, \ldots$

$$\phi_{\beta,i}(u) = \frac{\Gamma(\beta + i - 1)}{\Gamma(\beta)\Gamma(i)}(1-u)^{\beta}u^{i-1}.$$

The negative binomial distribution is an attractive generating function for many applications since it yields copulas with upper tail dependence [18]. In addition, negative binomial GPUC copulas can be represented as mixtures of beta distributions, offering an appealing way to simulate them.

The row and column sums are

$$\alpha_i = \int_0^1 \frac{\Gamma(\beta + i - 1)}{\Gamma(\beta)\Gamma(i)}(1-u)^{\beta}u^{i-1}du = \frac{\beta}{(\beta + i - 1)(\beta + i)}. \tag{3}$$

The copula density of the full form is

$$\begin{aligned} c(u,v) &= \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} M_{ij} \frac{\frac{\Gamma(\beta+i-1)}{\Gamma(\beta)\Gamma(i)}(1-u)^{\beta}u^{i-1}\frac{\Gamma(\beta+j-1)}{\Gamma(\beta)\Gamma(j)}(1-v)^{\beta}v^{j-1}}{\frac{\beta}{(\beta+i-1)(\beta+i)}\frac{\beta}{(\beta+j-1)(\beta+j)}} \\ &= (1-u)^{\beta}(1-v)^{\beta}\sum_{i=1}^{\infty}\sum_{j=1}^{\infty} M_{ij}\frac{\Gamma(\beta+i+1)}{\Gamma(i)\Gamma(\beta+1)}\frac{\Gamma(\beta+j+1)}{\Gamma(j)\Gamma(\beta+1)}u^{i-1}v^{j-1} \end{aligned}$$

and for the reduced form (2) we obtain

$$\begin{aligned} c_A(u,v) &= (1-u)^{\beta}(1-v)^{\beta}\sum_{i=1}^{m}\sum_{j=1}^{m} M_{ij}\frac{\Gamma(\beta+i+1)}{\Gamma(i)\Gamma(\beta+1)}\frac{\Gamma(\beta+j+1)}{\Gamma(j)\Gamma(\beta+1)}u^{i-1}v^{j-1} \\ c_B(u,v) &= (1-u)^{\beta}(1-v)^{\beta}\sum_{i=m+1}^{\infty}\left(\frac{\Gamma(\beta+i-1)}{\Gamma(i)\Gamma(\beta)}\right)^2(uv)^{i-1}\frac{(\beta+i-1)(\beta+i)}{\beta}. \end{aligned}$$

For integer β the infinite sum can be explicitly evaluated as a finite sum [18]. Since we do not restrict the posterior distribution of β to have integer support, we approximate the infinite sum by a sufficiently long finite sum.

3 Estimation algorithm

Suppose we have i.i.d. observations

$$Y = (u_1, v_1), \ldots, (u_n, v_n)$$

from a GPUC with parameter (vector) θ and parameter matrix D of known dimension $m \times m$. We assume that there is no prior knowledge about θ and D and, hence, we use flat priors that only incorporate the restriction that the parameters must lie inside the parameter space, e.g. that $\theta = \beta > 0$ for the negative binomial generator.

To draw samples from the joint posterior distribution $p(\theta, D|Y)$ we set up a standard Gibbs sampler iterating between the conditional posteriors $p(\theta|D, Y)$ and $p(D|\theta, Y)$. Let $\theta^{(r)}$ and $D^{(r)}$ denote the r-th draw from the respective posteriors, where $r = 1, \ldots, B + R$ with B being the length of the burn-in phase and R the number of Monte-Carlo iterations. If the generating function has no parameter θ, there is no need to iterate between the two Gibbs steps. Instead, one then only draws from $p(D|Y)$. However, in the following description of the algorithm we presuppose that the generating function has a parameter (vector) θ.

The Gibbs step $p(\theta|D, Y)$ is simple. We generate a draw from $p(\theta|D, Y)$ using the Metropolis-Hastings algorithm. The proposal for iteration $r + 1$ of the Markov chain is

$$\theta^{pr} = \theta^{(r)} + \varepsilon$$

with $\varepsilon \sim N(0, \sigma_\varepsilon^2)$. If θ is a parameter vector, then ε can be drawn from a multivariate normal distribution. The proposal is accepted with probability

$$\min\left(\frac{L(\theta^{pr}, D^{(r)}|Y)}{L(\theta^{(r)}, D^{(r)}|Y)}, 1\right)$$

where the logarithm of the likelihood is

$$\ln L(\theta, D|Y) = \sum_{i=1}^{n} \ln c(u_i, v_i)$$

with the copula density defined as in (2). If the proposal is accepted we set $\theta^{(r+1)} = \theta^{pr}$; if it is rejected then $\theta^{(r+1)} = \theta^{(r)}$. If the proposal is outside the parameter space its likelihood is zero, and the proposal is rejected.

Drawing from the high-dimensional $p(D|\theta, Y)$ is more intrictate. We make use of the approach suggested by [6] and adapt it in two different ways as explained in the next subsections[3].

3.1 Random Walk Metropolis-Hastings

In our random walk Metropolis-Hastings (RW-MH) algorithm the proposal for the doubly stochastic matrix is

$$D^{pr} = D^{(r)} + hF$$

where the random matrix F is constructed as follows: Draw two distinct rows $i_1, i_2 \in \{1, \ldots, m\}$ and two distinct columns $j_1, j_2 \in \{1, \ldots, m\}$, and set

$$F_{i_1,j_1} = 1, \quad F_{i_1,j_2} = -1, \quad F_{i_2,j_1} = -1, \quad F_{i_2,j_2} = 1.$$

All other elements of F are set to zero. The step size h is drawn from a uniform distribution on the interval $[-g, g]$. By construction, D^{pr} is a doubly stochastic matrix as long as all elements are positive. The proposal is accepted with probability

$$\min\left(\frac{L(\theta^{(r+1)}, D^{pr}|Y)}{L(\theta^{(r+1)}, D^{(r)}|Y)}, 1\right). \tag{4}$$

If the proposal is accepted we set $D^{(r+1)} = D^{pr}$; if it is rejected then $D^{(r+1)} = D^{(r)}$. For evaluating the copula density at D^{pr} it is not necessary to compute $c_B(u, v)$ again as it does not depend on the parameter matrix D^{pr} but only on θ which does not change in this Gibbs step.

In this algorithm, adding hF ensures the symmetry of the random walk proposal distribution. The scaling parameter h is set such that the acceptance ratio is relatively stable. While this algorithm is simple, draws from the posterior of the parameter matrix D do not mix well since it might take many iterations until an element of D is randomly chosen and altered. This effect is the more severe the larger is m.

3.2 Random Blocking Random Walk Metropolis-Hastings

To overcome poor mixing, we propose an alternative random blocking sampling scheme along the lines of [5] who introduced random blocking in the DSGE framework. [5] propose to divide the random walk Metropolis-Hastings algorithm into two steps. The first step consists of randomly generating blocks of parameters. The second step is the classical RW-MH over the randomly blocked parameters. Both the blocks and parameters are drawn in each iteration. For DSGE models, "[g]enerating random blocks [...] is [...] straightforward and does not require comment" [5, p. 22]. In the setting of this paper, however, the blocking method is essential. For ease of exposition we assume that m is even. Our random blocking random walk Metropolis-Hastings (RBRW-MH) algorithm for the parameter matrix D can be summarized as follows.

3 [17] propose to derive D directly from the data. For given θ their estimates perform well as initial values for our sampler.

1. Initialize a matrix of ones, $R = 1_{m\times m}$.
2. Draw one element uniformly from the set of index pairs $\{(i, j) : R_{ij} = 1\}$. Then set $R_{ij} = 0$.
3. Draw one element uniformly from the set of index pairs

$$\{(k, l) : R_{kl} = 1, R_{il} = 1, R_{kj} = 1\}.$$

   Then set $R_{kl} = R_{il} = R_{kj} = 0$.
4. Set $F = 0_{m\times m}$.
5. Set $F_{ij} = F_{kl} = 1$ and $F_{il} = F_{kj} = -1$.
6. Draw h from a uniform distribution on $[-g, g]$ and compute the proposal $D^{pr} = D^{(r)} + hF$.
7. Accept the proposal with probability (4). If any element of D^{pr} is negative, do not accept. If the proposal is accepted we set $D^{(r)} = D^{pr}$.
8. If all elements of R are zero, set $D^{(r+1)} = D^{(r)}$ and stop the algorithm. If not, continue with step 2. Since m is assumed to be even, all elements of D are subjected to the algorithm after $m^2/4$ steps.

This algorithm generates proposals for all elements of D in the Gibbs step, and consequently achieves much better mixing than RW-MH.

3.3 Adaptive proposal distributions

Allowing the scaling factors σ_ε and g to be varying (with respect to the iteration in the Markov chain), the sampling algorithms can be used in an adaptive fashion. Determining a sufficiently large value of g is crucial for both achieving a well mixing Markov chain and effectively exploring the full posterior distribution. However, too large values lead to a large fraction of rejected proposals.

The adaptation strategy is based on acceptance ratios. In the case of the RW-MH, g is decreased once the acceptance ratio falls below 0.15. For the RBRW-MH algorithm we suggest to decrease g if more than 95% of the four-parameter swaps are rejected. As to σ_ε we adapt it such that the acceptance ratio of the θ-proposals is roughly 0.4.

4 Simulation study

In this section we investigate the flexibility of the Bayesian estimation approach by fitting GPUCs to simulated data drawn from known parametric copulas. We consider (i) the normal copula with relatively high correlation (0.7), and (ii) a Gumbel copula with parameter $\alpha = 2$. The normal copula displays no tail dependence, the tail dependence index of the Gumbel copula is 0.586.

For each simulation, we draw $n = 5000$ observations $(u_1, v_1), \ldots, (u_n, v_n)$ from the parametric copula. Then we run either the RWMH or the RB-RWMH approach over $B = 5000$ burn-in iterations and $R = 50000$ iterations to draw from the joint posterior of the GPUC. We set the dimension of the matrix D to $m \times m$ with $m = 20$. The distributions of the random walk increments are governed by $g = 1/m$ (for the elements of D) both for the binomial and negative binomial generators, and $\sigma_\varepsilon = 0.15$ for β in case of the negative binomial generator (the parameter β is absent in GPUCs with the binomial generator). For the simulation study we did not adapt these parameters but kept them fix.

To visualize that the MCMC approach yields draws from the posterior distribution, figure 1 shows the histogram of β (top left panel), the trace plots for β (top right panel), and the trace plots of two elements of the matrix D (bottom left: D_{11}, bottom right: $D_{1,20}$) when fitting a GPUC with a negative binomial generator to the observations from a Gumbel distribution. Apparently the algorithm has converged to the posterior distribution after the burn-in period. Since mixing is only moderately strong, we show the thinned plots (with thinning parameter 10).

As the true copulas are known in the simulations, we can compute the distance of the nonparametrically estimated copulas from the true ones. As distance measure we employ the absolute distance of the copula

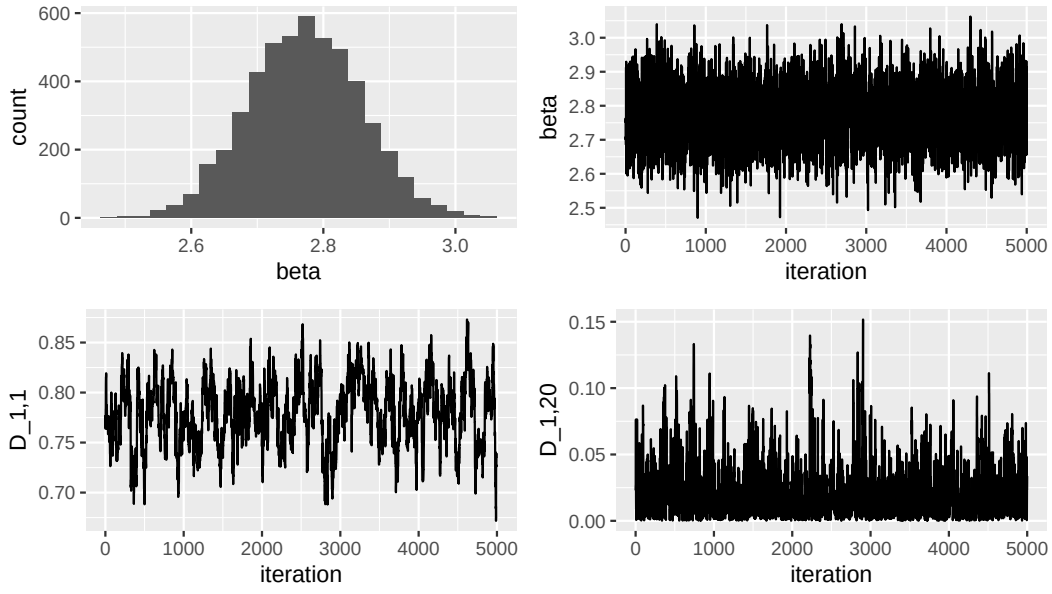

Figure 1: Histogram of the marginal posterior distribution of β (top left), trace plot of the thinned process of β (top right), trace plot of the thinned process of D_{11} (bottom left), trace plot of the thinned process of $D_{1,20}$ (bottom right).

Table 1: Simulated distance of nonparametrically estimated GPUC from parametric copulas. The distance is measured by the integrated absolute difference between the copulas.

		Copula	
generator	MH	normal (0.7)	Gumbel (2)
binomial	RW	0.0688	0.1101
	RB-RW	0.0717	0.1224
negative	RW	0.2294	0.1520
binomial	RB-RW	0.2368	0.1380

densities integrated over the unit square,

$$\int_0^1 \int_0^1 |c_0(u,v) - \hat{c}(u,v)| du dv$$

where c_0 is the true copula (i.e. either the normal copula or the Gumbel copula in our simulations), and $\hat{c}$ is the GPUC copula with matrix $\hat{D} = R^{-1} \sum_{r=1}^{R} D^{(r)}$ (and, for the negative binomial generator, similarly for $\hat{\beta}$).

The results show that the GPUC with a binomial generator and $m = 20$ fits both the normal copula and the Gumbel copula better, even though it cannot reflect the upper tail dependence of the Gumbel copula.

5 Empirical illustration: Returns of cryptocurrencies

Cryptocurrencies constitute a new asset class and are an interesting portfolio component for investors. The two most widely traded cryptocurrencies are Bitcoin and Ethereum. Recently, much research has been conducted about the statistical properties of the return distributions of cryptocurrencies. The return is defined as the logarithm of the change in the exchange rate to the US dollar.

All studies consistently find that: returns are clearly non-normally distributed; the mean return as well as the volatility (as measured by the standard deviation) are an order of magnitude larger than for traditional

currencies; the tails are very heavy. [13] estimates a tail index of slightly less than 2 implying that the variance does not exist. Other studies find a tail index above 2, e.g. [14], [4] or [22]. [14] also estimate the extremal index to quantify the amount of clustering of extreme returns. The extremal index tends to be smaller than for normal currencies, showing that extreme movements of the Bitcoin price are more clustered. [22] examine more time series properties and find volatility clustering, i.e. the autocorrelation of returns is small while the autocorrelation of absolute returns is larger. Volatility clustering is also documented and modeled by [11].

There is consensus in the literature that cryptocurrencies cannot be regarded as normal currencies. [2] argue that Bitcoin does not have the defining properties of money: it is not a general medium of exchange, nor is it a widely used accounting unit, nor can it act as a value storage. They argue that Bitcoin is an asset for speculation. [9] find that Bitcoin returns behave more like stock returns than like currency returns. [12] detect a significant momentum effect and attention effect for Bitcoin returns. They conclude that Bitcoin does not behave like a traditional currency.

If cryptocurrencies are added to a portfolio it is important to have an accurate model of their joint return distribution that mirrors its relevant features. We illustrate how GPUCs can help model the (bivariate) dependence structure of Bitcoin and Ethereum both at the level of daily prices and at the high-frequency level.

5.1 Daily return data

First, we consider daily data (provided by Bitfinex). The observation period starts on 2 January 2017 and ends on 22 October 2019 resulting in 1024 observations. In line with the other studies on the statistical properties of cryptocurrencies, the annualized mean returns and standard deviations are substantially larger than for traditional currencies. The annualized means are 0.51 for Bitcoin and 0.75 for Ethereum, the standard deviations are 0.7 (Bitcoin) and 0.93 (Ethereum).

Figure 2 depicts the time series of daily prices of Bitcoin and Ethereum (in US dollar). The bubble-like peaks at the end of 2017 (Bitcoin) or in early 2018 (Ethereum) are clearly visible. The period of price deflation lasted until early 2019. Figure 3 shows the daily returns. Extreme daily returns of more than 10 (or even 20) percent in absolute terms happen much more often than for traditional currencies. Volatility clusters are also discernible. This conforms with return autocorrelations of -0.016 (Bitcoin) and 0.023 (Ethereum) and autocorrelations of the absolute returns of 0.159 (Bitcoin) and 0.203 (Ethereum).

Figure 4 shows the scatterplot of the empirical edfs (i.e. the ranks normalized to the unit interval) of Bitcoin and Ethereum returns. The plot indicates that there is a rather strong positive dependence between the daily returns. However, sometimes extremely large Bitcoin returns can coincide with extremely small Ethereum returns and vice versa. Further, a visual inspection of the plot suggests that there is lower tail dependence but no upper tail dependence.

We fit nonparametric GPUCs both with binomial and negative binomial generators to the data. Since the negative binomial generator implies copulas with upper tail dependence we rotate the empirical observations to transform their apparent lower tail dependence into upper tail dependence. Figures 5 shows the nonparametrically estimated copula density of the joint returns (left panel: binomial generator, right panel: negative binomial generator). Note that the negative binomial generator induces a markedly stronger upper tail dependence while the binomial generator implies a more symmetric tail behaviour.

5.2 High frequency return data

Cryptocurrencies are continuously traded worldwide. Our high-frequency dataset[4] records the prices and returns at 10 minute intervals (without weekends) starting on 8 April 2019, 21:10 and ending on 22 October 2019, 07:40 (UTC). The number of observations is 20562. As the observation window is different from the daily data,

4 The data source is Bloomberg Generic Pricing London.

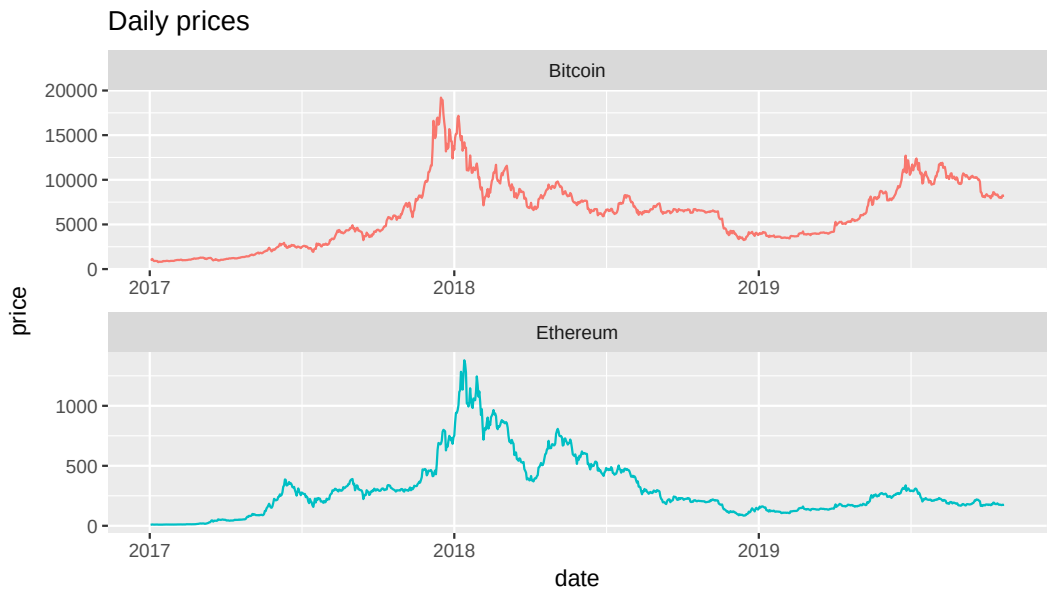

Figure 2: Time series of Bitcoin and Ethereum daily prices

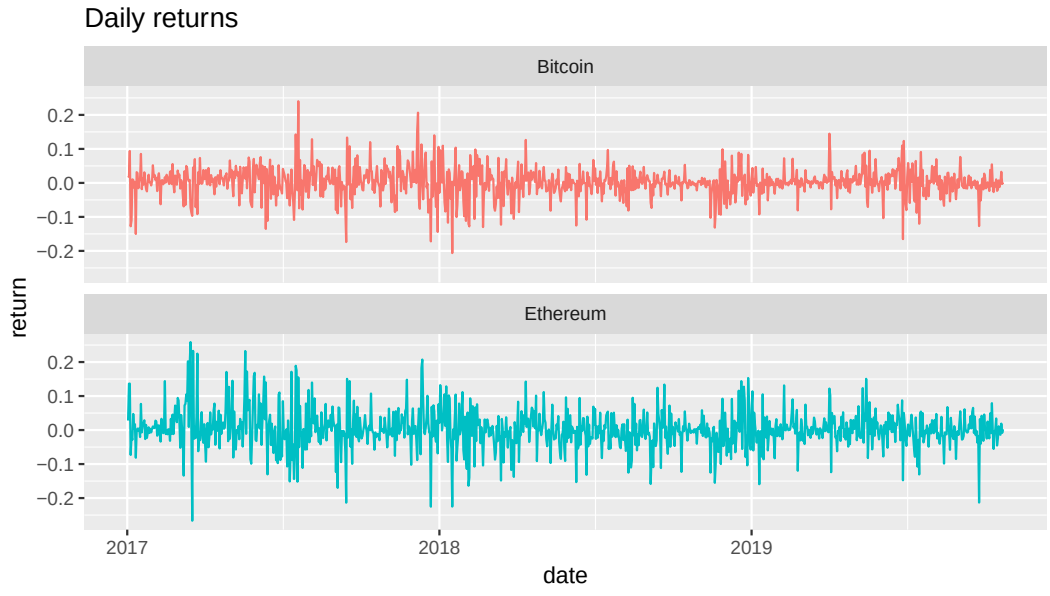

Figure 3: Time series of Bitcoin and Ethereum daily returns

the annualized mean returns are also rather different. The annualized mean returns are 0.78 (Bitcoin) and −0.08 (Ethereum), the annualized standard deviation are closer to the daily values: 0.79 and 0.86. The first order autocorrelation is again small (−0.015 and −0.002), the autocorrelation of absolute returns is higher than for daily data (0.245 and 0.242). Figures 6 and 7 show the high-frequency time series of prices and returns. Figure 8 shows the scatterplot of the empirical edfs of Bitcoin and Ethereum returns. The structure appears to be more symmetric than for daily data.

The estimated copula densities are depicted in figure 9 (left: binomial generator, right: negative binomial generator). The nonparametrically estimated densities are rather dissimilar. The tail dependence symmetry is

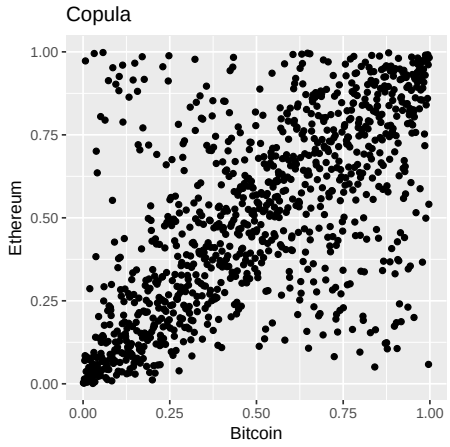

Figure 4: Scatterplot of empirical cdfs of daily returns for Bitcoin and Ethereum

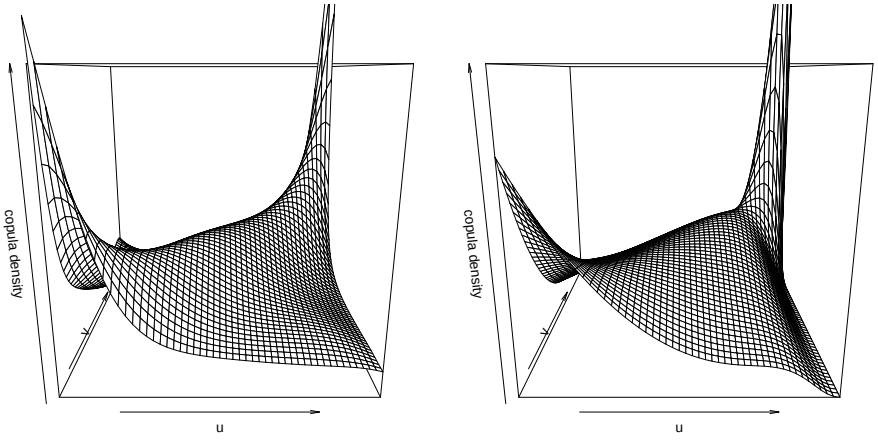

Figure 5: Estimated copula densities of daily Bitcoin and Ethereum returns for binomial (left) and negative binomial (right) generators, the scales are the same in both plots

reflected by the GPUC with binomial generator but to a much lesser extent by the GPUC with negative binomial generator. We conclude that GPUCs with a negative binomial generator do not fit copulas with lower (or upper and lower) tail dependence sufficiently accurately.

6 Concluding Remarks

This paper proposes Bayesian methods to estimate generalized partition of unity copulas (GPUCs). While the approach is applicable to general GPUCs we explicitly consider GPUCs with (a) binomial generating functions (i.e. Bernstein copulas) and (b) negative binomial generating functions. GPUCs with a negative binomial generator exhibit upper tail dependence which prima facie makes them attractive for many fields of applications. We introduce a parametrization for the GPUCs that enables us to use random walk Metropolis-Hastings algorithms. To improve the mixing properties of the sampler, we suggest random blocking for the elements of the parameter matrix. The random walk step sizes can be chosen in an adaptive fashion. In a simulation study, we show that GPUCs with a binomial generator (Bernstein copulas) fit both the normal and Gumbel copula better than GPUCs with a negative binomial generator for a moderate sample size of 5000. This holds true

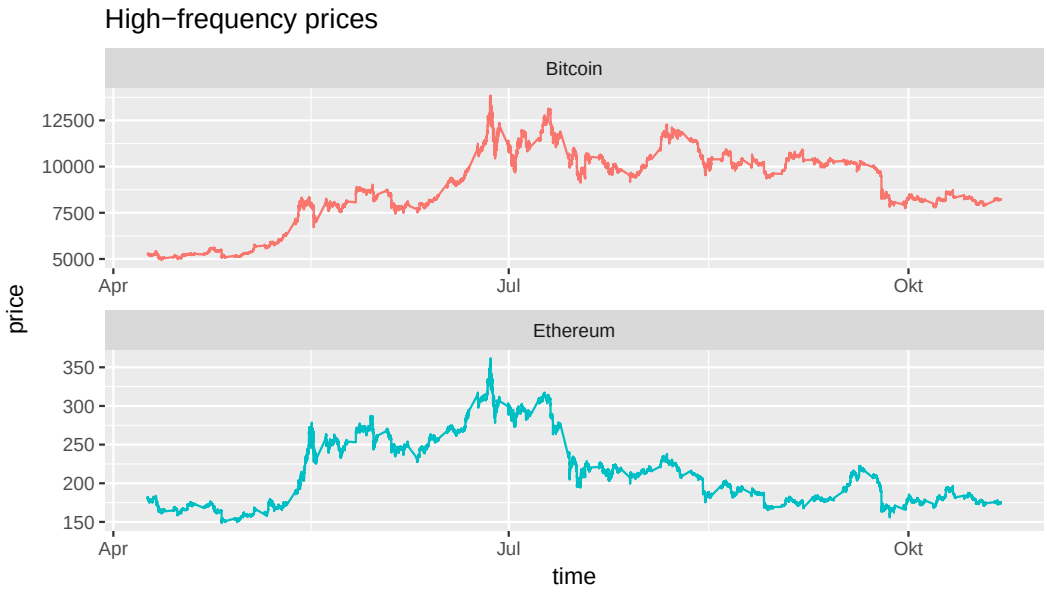

Figure 6: Time series of Bitcoin and Ethereum HF prices

Figure 7: Time series of Bitcoin and Ethereum HF returns

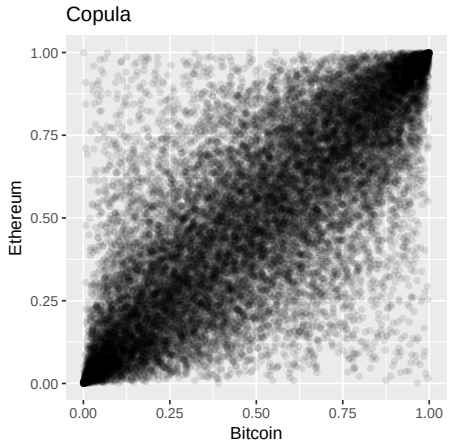

Figure 8: Scatterplot of empirical cdfs of 10-minute returns for Bitcoin and Ethereum

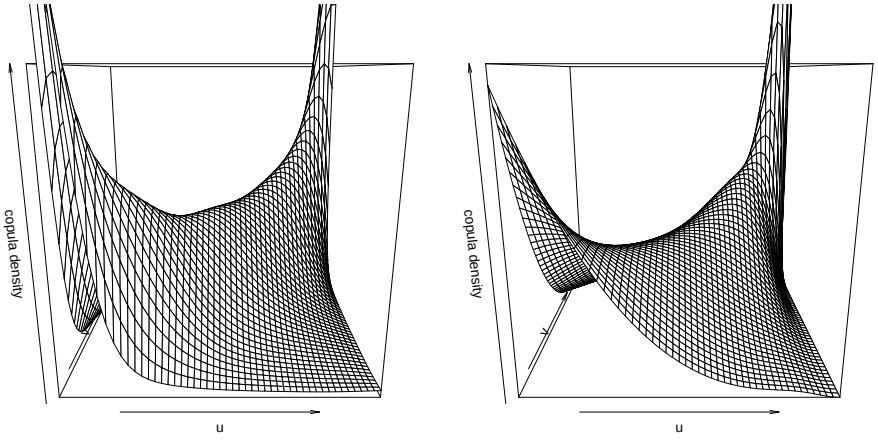

Figure 9: Estimated copula densities of high-frequency Bitcoin and Ethereum returns for binomial (left) and negative binomial (right) generators, the scales are the same in both plots

despite the upper tail dependence of the Gumbel copula which cannot be reproduced by Bernstein copulas unless m is large. The apparent advantage of negative binomial generators turns out not to be pertinent in (at least some) practical applications. Our empirical illustration shows that important main features of the joint distribution of daily and high-frequency returns of the crypto-currencies Bitcoin and Ethereum are well captured by GPUCs. Again we find that GPUCs with a negative binomial generator tend to fit the empirical joint distribution of daily and high-frequency returns less closely than Bernstein copulas.

References

[1] Baker, R. (2008). An order-statistics-based method for constructing multivariate distributions with fixed marginals. *J. Multivariate Anal. 99*(10), 2312–2327.

[2] Bariviera, A. F., M. J. Basgall, W. Hasperué, and M. Naiouf (2017). Some stylized facts of the Bitcoin market. *Physica A 484*, 82–90.

[3] Bedford, T. and R. M. Cooke (2002). Vines–a new graphical model for dependent random variables. *Ann. Statist. 30*(4), 1031–1068.

[4] Begušić, S., Z. Kostanjčar, H. E. Stanley, and B. Podobnik (2018). Scaling properties of extreme price fluctuations in Bitcoin markets. *Physica A 510*, 400–406.
[5] Chib, S. and S. Ramamurthy (2010). Tailored randomized block MCMC methods with application to DSGE models. *J. Econometrics 155*(1), 19–38.
[6] Diaconis, P., G. Lebeau, and L. Michel (2012). Gibbs/Metropolis algorithms on a convex polytope. *Math. Z. 272*(1-2), 109–129.
[7] Dou, X., S. Kuriki, G. D. Lin, and D. Richards (2016). EM algorithms for estimating the Bernstein copula. *Comput. Statist. Data Anal. 93*, 228–245.
[8] Guillotte, S. and F. Perron (2012). Bayesian estimation of a bivariate copula using the Jeffreys prior. *Bernoulli 18*(2), 496–519.
[9] Hu, A. S., C. A. Parlour, and U. Rajan (2019). Cryptocurrencies: Stylized facts on a new investible instrument. *Financ. Manag. 48*(4), 1049–1068.
[10] Joe, H. (1997). *Multivariate Models and Dependence Concepts*. CRC Press, Boca Raton FL.
[11] Kyriazis, N. A., K. Daskalou, M. Arampatzis, P. Prassa, and E. Papaioannou (2019). Estimating the volatility of cryptocurrencies during bearish markets by employing GARCH models. *Heliyon 5*(8), Article ID e02239, 8 pages.
[12] Liu, Y. and A. Tsyvinski (2018). Risks and returns of cryptocurrency. Available at https://www.nber.org/papers/w24877.
[13] Osterrieder, J. (2017). The statistics of Bitcoin and cryptocurrencies. *Proceedings of the* 2017 *International Conference on Economics, Finance and Statistics*, pp. 285–289. Atlantis press, Paris.
[14] Osterrieder, J. and J. Lorenz (2017). A statistical risk assessment of Bitcoin and its extreme tail behavior. *Ann. Financ. Econ. 12*(1), Article ID 1750003, 19 pages.
[15] Pfeifer, D., A. Mändle, and O. Ragulina (2017). New copulas based on general partitions-of-unity and their applications to risk management (part II). *Depend. Model. 5*, 246–255.
[16] Pfeifer, D., A. Mändle, O. Ragulina, and C. Girschig (2019). New copulas based on general partitions-of-unity (part III) - the continuous case. *Depend. Model. 7*, 181–201.
[17] Pfeifer, D., A. Mändle, and O. Ragulina (2017). Data driven partition-of-unity copulas with applications to risk management. Available at https://arxiv.org/abs/1703.05047.
[18] Pfeifer, D., H. A. Tsatedem, A. Mändle, and C. Girschig (2016). New copulas based on general partitions-of-unity and their applications to risk management. *Depend. Model. 4*, 123–140.
[19] Sancetta, A. and S. Satchell (2004). The Bernstein copula and its applications to modeling and approximations of multivariate distributions. *Econom. Theory 20*(3), 535–562.
[20] Savu, C. and M. Trede (2010). Hierarchies of Archimedean copulas. *Quant. Finance 10*(3), 295–304.
[21] Sinkhorn, R. (1964). A relationship between arbitrary positive matrices and doubly stochastic matrices. *Ann. Math. Statist. 35*(2), 876–879.
[22] Zhang, W., P. Wang, X. Li, and D. Shen (2018). Some stylized facts of the cryptocurrency market. *Appl. Econ. 50*(55), 5950–5965.