Bayesian Estimation of Generalized Partition of Unity Copulas

Andreas Masuhr[†]

73/2018

[†] Department of Economics, University of Münster, Germany

Bayesian Estimation of Generalized Partition of Unity Copulas

Andreas Masuhr

Westfälische Wilhelms-Universität Münster

Abstract

This paper proposes the first MCMC based algorithms to estimate Generalized Partition of Unity Copulas (GPUC), a class of nonparametric copulas, recently introduced by Pfeifer et al. (2016). The first one is a random walk Metropolis-Hastings (RW-MH) algorithm and the second one a random blocking randwom walk Metropolis-Hastings algorithm (RBRW-MH); both algorithms can be used adaptively tuned. Additionally, (flat) conditional prior distributions are derived from the volume of the sampling space of GPUC matrices using importance sampling. An algorithm to randomly sample GPUC matrices are proposed, as well. Simulation studies to compare the efficiency of the presented algorithms are carried out and both algorithms are utilized to estimate the copula of a data set containing new infections with Campylobacteriosis in two German states (Berlin and Brandenburg).

1 Introduction

A bivariate copula is a function $C : [0,1]^2 \to [0,1]$ with the following properties[1]:

1. $C(u,1) = C(0,v) = 0$
2. $C(u,1)u; C(1,v) = v$
3. $C(u,v)$ is 2-increasing, i.e. for $u_1,\ u_2$ and $v_1,\ v_2$ with $u_1 \leq u_2$ and $v_1 \leq v_2$ it holds:

$$C(u_2,v_2) - C(u_2,v_1) - C(u_1,v_2) + C(u_1,v_1) \geq 0.$$

Then, using Sklar's theorem, any bivariate distribution $F_{XY}(x,y)$ can be split up into its marginal distributions $F_X(x)$ and $F_Y(y)$ and its copula $C(u,v)$, with $u = F_X(x)$ and $v = F_Y(y)$ the following way:

$$F_{XY}(x,y) = C\big(F_X(x), F_Y(y)\big). \tag{1}$$

If X and Y are continuous random variables, then the copula $C(u,v)$ is unique. Using this representation allows to treat the marginal distributions and their dependence structure, separately and hence, adds more flexibility to the modeling process. Various families of copulas have emerged in the past years 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 certain generator functions to create copulas. To add even more flexibility in higher dimensions all these copula families might be mixed using vines (Bedford and Cooke (2002)) or, within the class of Archimedean copulas bivariate copulas can be organized hierarchically (this idea was first mentioned by Joe (2001) and further developed Savu and Trede (2010)). Yet another way to construct copulas is to use a nonparametric approach by trying to directly mimic the characteristics of the data. The simplest way of doing so is to just use the empirical copula $C^*(u,v)$:

$$C^*(u,v) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{(U_i<u, V_i<v)}.$$

Another nonparametric copula is the Bernstein copula (Baker (2008)). This copula uses Bernstein polynomials to smoothly approximate the copula density. One method to estimate Bernstein copulas is an EM-Algorithm, proposed by

[1]For more on copulas, see Nelsen (2006)

Dou et al. (2016). Recently, another family of nonparametric copulas has been developed, the so called Generalized Partition of Unity Copulas (GPUC; Pfeifer et al. (2016)). In contrast to the Bernstein copula the copula density over the unit hypercube is no longer approximated by a finite mixture of functions but by an infinite mixture. As a special case GPUCs also nest the Bernstein copula. This paper proposes Bayesian MCMC based algorithms to estimate GPUCs and proceeds as follows: the second section introduces the Generalized Partition of Unity Copulas. Chapter three introduces the algorithm to sample parameter matrices of Generalized Partition of Unity Copulas. This algorithm is utilized in section four to construct conditionally flat prior distributions for the aforementioned parameter matrices. Section five, in turn, presents two MCMC algorithms to sample from the posterior distribution of these parameters. Both algorithms are compared based on simulated and are applied to estimate the contemporaneous copula of an infection data set. Finally, section seven concludes.

2 Generalized Partition of Unity Copulas

The concept of Generalized Partition of Unity Copulas (GPUC) was first introduced by Pfeifer et al. (2016) and applied in the context of risk management Pfeifer et al. (2017). A GPUC is a copula defined by the density function:

$$c(u,v) = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} M_{ij}\frac{\phi_i(u)}{\alpha_i}\frac{\varphi_j(v)}{\gamma_j}, \tag{2}$$

with $\alpha_i = \sum_{j=0}^{\infty} M_{ij} = \int_0^1 \phi_i(u)du$ and $\gamma_j = \sum_{i=0}^{\infty} M_{ij} = \int_0^1 \varphi_j(v)dv$. The *generating functions* $\phi_i(u)$ and $\varphi_j(v)$ can be considered as probability mass functions of discrete random variables over the non-negative integers $\mathbb{Z}^+$ with parameters u and v, respectively and hence, $\sum_{i=0}^{\infty} \phi_i(u) = \sum_{j=0}^{\infty} \varphi(v) = 1$. In the subsequent paper I only consider bivariate GPUCs with $\phi_i(u) = \varphi_j(u)$ for $i = j$, i.e. copulas with identical generating functions. Equation (2) denotes the fully parametrized form GPUC with an infinitely large parameter matrix $\boldsymbol{M}$. For estimation, I consider a reduced form GPUC, i.e.

$$c(u,v) = \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} M_{ij}\frac{\phi_i(u)}{\alpha_i}\frac{\varphi_j(v)}{\gamma_j} + \sum_{i=m}^{\infty}\frac{\phi_i(u)\varphi_i(v)}{\alpha_i}. \tag{3}$$

This form requires a $m \times m$ matrix to parameterize the lower left part of $[0,1]^2$. Accordingly, the upper right part can be parametrized by additional parameters of the generating functions $\phi_i(u)$ and $\varphi_j(v)$.

Proposition 2.0.1. *All fully parameterized GPUCs nest the independence copula if and only if* $M_{ij} = \alpha_i \gamma_j \; \forall i,\; j$

This property of GPUCs can easily be verified as

$$c(u,v) = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} M_{ij} \frac{\phi_i(u)}{\alpha_i} \frac{\varphi_j(v)}{\gamma_j} = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \phi_i(u)\varphi_j(v) = 1 \quad \square$$

Consequently, all reduced form GPUCs don't include the independence copula anymore and hence care must be taken when specifying the size of the parameter Matrix $\boldsymbol{M}$ before estimation. As an example of a GPUC, consider as generating function the probability mass function of a negative binomial random variable:

$$\phi_i(u) = \binom{\beta + i - 1}{i} (1-u)^{\beta} u^i.$$

Therefore,

$$\alpha_i = \int_0^1 \phi_i(u)du = \int_0^1 \binom{\beta+i-1}{i}(1-u)^{\beta}u^i du = \frac{\beta}{(\beta+i)(\beta+i+1)}$$

and finally the density is given by,

$$c(u,v) = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \frac{M_{ij}\beta^2 \binom{\beta+i+1}{i}\binom{\beta+j+1}{j}(1-u)^{\beta}(1-v)^{\beta}u^i v^j}{(\beta+1)^2(\beta+i+1)(\beta+j+1)} \tag{4}$$

for the full form and

$$\begin{aligned} c(u,v) = &\sum_{i=0}^{m-1}\sum_{j=0}^{m-1} \frac{M_{ij}\beta^2 \binom{\beta+i+1}{i}\binom{\beta+j+1}{j}(1-u)^{\beta}(1-v)^{\beta}u^i v^j}{(\beta+1)^2(\beta+i+1)(\beta+j+1)} \\ &+ \sum_{i=m}^{\infty} \frac{(\beta+1)(\beta+i+1)}{\beta} \binom{\beta+i-1}{i}^2 (1-u)^{\beta}(1-v)^{\beta}u^i v^i \end{aligned} \tag{5}$$

for the reduced form GPUC. Using negative binomial distributions as generating functions results in a copula with upper tail dependence, according to Pfeifer et al. (2016).

By using binomial distributions as generating functions, the Bernstein Copula is obtained. Note, that both negative Binomial GPUC and Bernstein Copula can be represented by mixtures of Beta distributions, offering an appealing way to simulate them (see appendix). The aim of this paper is to estimate the parameter matrix of a reduced form GPUC, i.e. a matrix $\boldsymbol{M}$ from the space

$$\mathcal{M}_{\beta} = \left\{ \boldsymbol{M} : M_{ij} \geq 0 \forall i,j;\; \sum_{i=1}^{m} M_{i,j} = \alpha_j;\; \sum_{j=1}^{m} M_{i,j} = \alpha_i \right\}. \tag{6}$$

Accordingly, one can define the set $\mathcal{G}_\beta$ of $(m-1)\times(m-1)$ GPUC matrix candidates by

$$\mathcal{G}_\beta = \Bigg\{ \boldsymbol{G} : G_{ij} \geq 0 \forall i,j;\ \alpha_i - \alpha_m \leq \sum_{j=1}^{m-1} G_{ij} \leq \alpha_i;\ \alpha_j - \alpha_m \leq \sum_{i=0}^{m-1} G_{ij} \leq \alpha_j; \\ \sum_{k=1}^{m-1} \alpha_k - \alpha_m \leq \sum_{i=1}^{m-1}\sum_{j=1}^{m-1} G_{ij} \leq \sum_{k=1}^{m-1} \alpha_k \Bigg\}. \tag{7}$$

$\mathcal{G}_\beta$ includes all $(m-1)\times(m-1)$ matrices that can be expanded into eligible GPUC parameter matrices by adding an m-th row and m-th column to $\boldsymbol{G}$ and setting the values of these row and column equal to

$$G_{i,m} = \alpha_i - \sum_{j=1}^{m-1} G_{i,j} \text{ and } G_{m,j} = \alpha_j - \sum_{i=1}^{m-1} G_{i,j}.$$

3 Sampling GPUC Matrices

In order to use a MCMC-based sampling scheme this section proposes an algorithm to draw from the space of (reduced form) GPUC parameter matrices. To that end, the matrices $\boldsymbol{M}^+$ and $\boldsymbol{M}^-$ that trace upper and lower limits of the parameter matrix $\boldsymbol{M}$ during the iterations of the algorithm; $\boldsymbol{M}^+$ and $\boldsymbol{M}^-$ are initialized as:

$$\boldsymbol{M}_0^+|\beta = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \dots & \alpha_m \\ \alpha_2 & \alpha_2 & \alpha_3 & \dots & \alpha_m \\ \alpha_3 & \alpha_3 & \alpha_3 & \dots & \alpha_m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_m & \alpha_m & \alpha_m & \dots & \alpha_m \end{bmatrix} \text{ and}$$

$$\boldsymbol{M}_0^-|\beta = \begin{bmatrix} \max\{\alpha_1 - \sum_{j=2}^m \alpha_j, 0\} & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}. \tag{8}$$

Now, to sample a parameter matrix M, the following algorithm is proposed:

```
Initialize $\boldsymbol{M} = \boldsymbol{0}$;
Initialize $m \times m$ matrices $\boldsymbol{R} = \boldsymbol{0}$ and $\boldsymbol{C} = \boldsymbol{0}$;
Initialize $\boldsymbol{M}_0^+$ and $\boldsymbol{M}_0^-$ as in (8);
for i in 1:m do
    for j in 1:m do
        Draw $m_{i,j}$ uniformly distributed on $[M_{ij}^-, M_{ij}^+]$;
        Compute $r_i = \alpha_i - \sum_{k=1}^{j} \boldsymbol{M}_{ik}$ and $c_j = \alpha_j - \sum_{k=1}^{i} \boldsymbol{M}_{kj}$;
        Set $\boldsymbol{R}_{ij} = r_i$ and $\boldsymbol{C}_{ij} = c_j$;
        Update $\boldsymbol{M}^+$ and $\boldsymbol{M}^-$ according to:
            • $\boldsymbol{M}^+ = pmin\{\boldsymbol{R}, \boldsymbol{C}\}$, where pmin is the element wise minimum;
            • $\boldsymbol{M}_{ij}^- = max\left\{\alpha_i - \sum_{k=1}^{j} \boldsymbol{M}_{ik} - \sum_{k=j+1}^{m} \boldsymbol{M}_{ik}^+,\ \alpha_j - \sum_{k=1}^{i} \boldsymbol{M}_{kj} - \sum_{k=i+1}^{m} \boldsymbol{M}_{kj}^+,\ 0\right\} \forall i, j$;
    end
end
```

Algorithm 1: Sampling GPUC parameter matrices.

The proof that all matrices generated with this algorithm are GPUC parameter matrices follows by contradiction: Assume a generated matrix $\widetilde{\boldsymbol{M}}$ is not a GPUC matrix. This might happen in three cases:

1. $\widetilde{\boldsymbol{M}}_{ij} < 0$ for some i, j.
2. $\sum_{k=1}^{m} \widetilde{\boldsymbol{M}}_{kj} \neq \alpha_j$ or $\sum_{k=1}^{m} \widetilde{\boldsymbol{M}}_{ik} \neq \alpha_i$ for some i, j
3. $\sum_{i=1}^{m} \sum_{j=1}^{m} \widetilde{\boldsymbol{M}}_{ij} \neq \sum_{j=1}^{m} \alpha_j = \sum_{i=1}^{m} \alpha_i$

Now, it is sufficient to show that the first case cannot happen and that the subsequent cases would require the former one(s) to happen: For all elements of $\widetilde{\boldsymbol{M}}$ the first case cannot occur, since the minimum $\boldsymbol{M}_{ij}^-$ in each draw is limited from below by 0. The maximum of each cell is non negative as well, which again is shown by contradiction: Asssume $\boldsymbol{M}_{ij}^+$ is negative for the first time in the algorithm. Consequently, either $\alpha_i < \sum_{k=1}^{j-1} \boldsymbol{M}_{ik}$ or $\alpha_j < \sum_{k=1}^{i-1} \boldsymbol{M}_{kj}$ which requires either $\boldsymbol{M}_{i-1,j}^+ > \alpha_j - \sum_{k=1}^{i-2} M_{kj}$ or $\boldsymbol{M}_{i,j-1}^+ > \alpha_i - \sum_{k=1}^{j-2} M_{ik}$, which in turn is ruled out by the definition of the matrices $\boldsymbol{R}$ and $\boldsymbol{C}$.

If $\widetilde{\boldsymbol{M}}_{i,j} \geq 0 \forall i, j$, then by definition of $\boldsymbol{R}$ and $\boldsymbol{C}$ it follows that $\widetilde{\boldsymbol{M}}_{i,m} = \alpha_i - \sum_{k=1}^{m-1} \widetilde{\boldsymbol{M}} i, k$ and hence the second case cannot happen, either. Consequently, it is trivial that the third case cannot happen as well, and hence the algorithm

always generates eligible GPUC parameter matrices.
Note, however that this is only the case if the matrices $\boldsymbol{M}^+$ and $\boldsymbol{M}^-$ are initialized as in (8).

4 A Prior distribution on M

The aim of any Bayesian estimation method is to obtain the posterior distribution of the parameters of interest (or at least a sample of it), given the data. In the case of a GPUC, the posterior distribtuion is $p(\boldsymbol{M}, \beta | u, v) \propto p(\boldsymbol{M}, \beta) p(u, v | \boldsymbol{M}, \beta)$. Now, the joint prior of $\boldsymbol{M}$ and β, $p(\boldsymbol{M}, \beta)$ can be factorized into $p(\boldsymbol{M}, \beta) = p(\beta) \cdot p(\boldsymbol{M}|\beta)$. If both, the prior for β and the conditional prior for $\boldsymbol{M}$ are assumed to be flat, the challenge is to identify the volume of the space that $\boldsymbol{M}$ can be sampled from, given β in order to determine the value of the conditional prior of $\boldsymbol{M}$.
To illustrate this consider the simplest case possible, i.e. estimating a 2×2 matrix $\boldsymbol{M}$. In this case there is only one free parameter, $\boldsymbol{M}_{1,1}$ with $\boldsymbol{M}_{1,2}, \boldsymbol{M}_{2,1}$ and $\boldsymbol{M}_{2,2}$ being computed by the constraints on the row and column sums of $\boldsymbol{M}$. Consequently $\boldsymbol{M}_{1,1}$ is drawn from the interval $[\alpha_1 - \alpha_2, \alpha_1]$.[2] Now, generally, α_i depends on β, for example in the case of a Negative Binomial GPUC: $\alpha_i = \frac{\beta}{(\beta+i-1)(\beta+i)}$ and thus, an uninformative prior for a 2×2 matrix $\boldsymbol{M}$ given β is a function of β with $p(\boldsymbol{M}|\beta) = \beta + 3 + \frac{2}{\beta}$.
For arbitrary values of m, a conditionally flat prior can be described by introducing an auxiliary $(m-1) \times (m-1)$ random matrix $\boldsymbol{B}|\beta \sim U(\boldsymbol{0}, \boldsymbol{M}_0^+)$, where $\boldsymbol{M}_0^+|\beta$ is defined as in (8). Now, the prior for a $m \times m$ matrix $\boldsymbol{M}$ is given by

$$
\begin{aligned}
p(\boldsymbol{M}|\beta) &= \left(\prod_{i=1}^{m-1} \prod_{j=1}^{m-1} \boldsymbol{M}_{0;i,j}^+ \int \mathbf{1}_{\{\boldsymbol{B} \in \mathcal{G}\}} \left(\prod_{i=1}^{m-1} \prod_{j=1}^{m-1} \boldsymbol{M}_{i,j}^+ \right)^{-1} d\boldsymbol{B} \right)^{-1}, \\
&= \left(E_{\boldsymbol{B}} \left[\mathbf{1}_{\{\boldsymbol{B} \in \mathcal{G}\}} | \beta \right] \prod_{i=1}^{m-1} \prod_{j=1}^{m-1} \boldsymbol{M}_{i,j}^+ \right)^{-1} \\
&= \frac{f(\boldsymbol{B}|\beta)}{E_{\boldsymbol{B}} \left[\mathbf{1}_{\{\boldsymbol{B} \in \mathcal{G}\}} | \beta \right]} \qquad (9)
\end{aligned}
$$

where $\mathcal{G}$ is the set of $(m-1) \times (m-1)$ matrices that are eligible candidates for $m \times m$ GPUC parameter matrices as defined in (7) and $\mathbf{1}_{\{\boldsymbol{B} \in \mathcal{G}\}}$ denotes the indicator function that is equal to one if $\boldsymbol{B} \in \mathcal{G}$ and 0, elsewhere. Evaluating

[2]That is, because $\boldsymbol{M}_{1,1}^+ = \alpha_1$ and $\boldsymbol{M}_{1,1}^- = \alpha_1 - \alpha_2$

this integral for large m gets very cumbersome using classical approximation schemes due to the high dimensionality.
As a solution, an importance sampling approach is proposed to estimate the above expectation. Following Greenberg (2008), an estimate of $E[g(x)] = \int g(x)f(x)dx$ with $X \sim F(\cdot)$ and density $f(x)$ can be computed considering the integral

$$E[g(x)] = \int \frac{g(x)f(x)}{h(x)} h(x)dx$$

and approximating it by a sample of sample of N values X_n from the importance distribution $h(x)$ and computing

$$E[g(x)] \approx \frac{1}{N} \sum_{n=1}^{N} g(x_n) \frac{f(x_n)}{h(x_n)}.$$

Translated into the setting of GPUCs, $p(\boldsymbol{M}|\beta)$ is obtained by drawing a sample of N values $\boldsymbol{B}_n$ from an importance distribution $h(\boldsymbol{B})$ and then the expectation is approximated by a weighted average over the drawn sample:

$$E_{\boldsymbol{B}}[\mathbf{1}_{\{\boldsymbol{B}\in\mathcal{G}\}}|\beta]^{-1} \approx \left(\frac{1}{N} \sum_{n=1}^{N} \mathbf{1}_{\{\boldsymbol{B}_n\in\mathcal{G}\}} \frac{f(\boldsymbol{B}_n|\beta)}{h(\boldsymbol{B}_n|\beta)} \right)^{-1}, \tag{10}$$

where $f(\boldsymbol{B}_n|\beta) = \left(\prod_{i=1}^{m-1} \prod_{j=1}^{m-1} \boldsymbol{M}_{i,j}^{+} \right)^{-1}$. Here, the importance distribution $h(\boldsymbol{B}|\beta)$ is chosen according to the sampling scheme of algorithm 1. Formally, drawing a candidate $\boldsymbol{B}$ from $h(\boldsymbol{B}|\beta)$ is split into drawing from a set of conditional distributions when sampling a candidate GPUC $\boldsymbol{B}$ matrix, i.e.:

$$\begin{aligned} h(\boldsymbol{B}|\beta) =& p(B_{1,1}) \cdot p(B_{1,2}|M_{1,1}) \cdot \ldots \cdot p(B_{1,m-1}|B_{1,1}, \ldots, B_{1,m-2}) \\ & \cdot p(B_{m-1,m-1}|B_{1,1}, \ldots, B_{m-1,m-2}) = \prod_{i=1}^{m-1} \prod_{j=1}^{m-1} (\boldsymbol{M}_{i,j,*}^{+} - \boldsymbol{M}_{i,j,*}^{-})^{-1}, \end{aligned} \tag{11}$$

where $\boldsymbol{M}_{i,j,*}^{+}$ and $\boldsymbol{M}_{i,j,*}^{-}$ are updated before each subsequent element is drawn and conditioning on β is dropped for readability. Consequently, every draw from $h(\boldsymbol{B})$ is a eligible GPUC parameter matrix candidate and hence, (9) simplifies to:

$$p(\boldsymbol{M}|\beta) \approx \left(\frac{1}{N} \sum_{n=1}^{N} h(\boldsymbol{B}_n|\beta)^{-1} \right)^{-1}. \tag{12}$$

A comparison between a naive Monte Carlo approximation and a importance sampling approximation using both 50,000 draws for $m = 4$ (i.e. 9 parameters need to be drawn in each matrix) is given in figure 1. Both approaches work for 4×4 matrices but the variance of the importance sampling estimator is already substantially lower. Additionally, only about 0.03% of all matrices drawn by the naive Monte Carlo simulation are eligible candidates for GPUC matrices. This proportion diminishes sharply as m increases and hence, makes the determination of the conditional prior of $\boldsymbol{M}$ via naive Monte Carlo simulation infeasible.

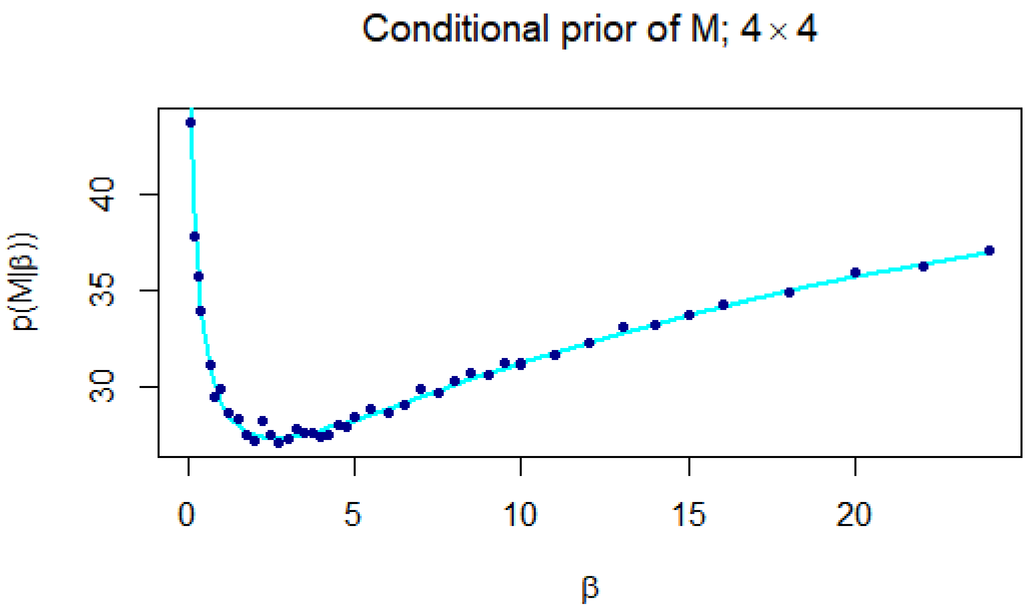

Figure 1: *Comparison of a naive Monte Carlo approximation (black dots) and the importance sampling approach (cyan line). 50,000 draws for each value of β for a negative binomial GPUC.*

Once a sample is obtained, the prior is approximated as a function of β using a log-polynomial model of order k:

$$log(\widehat{P(\boldsymbol{M}|\beta)}) = \sum_{i=0}^{k} \alpha_i \beta^i + \alpha_{k+1} log(\beta) \tag{13}$$

which serves as a very accurate approximation. Table 4 in the appendix displays estimates for $k = 5$ and $m = 3, \ldots, 20$.

5 MCMC algorithms for GPUC Matrices

Besides putting a (conditional) prior on $\boldsymbol{M}$ another necessity is to recompute the parameter matrix whenever a new value of $\widetilde{\beta}$ is proposed, because $\boldsymbol{M} \notin \mathcal{M}|\widetilde{\beta}$, i.e $\boldsymbol{M}$ is no longer an eligible GPUC parameter matrix. Hence, the following algorithm[3] will be used to reshape the GPUC matrix $\boldsymbol{M}$:

```
Initialize $\boldsymbol{M}_c = \boldsymbol{M}$, $\boldsymbol{M}_r = \boldsymbol{M}$ and $\boldsymbol{M}_n = \boldsymbol{M}$; while $|\sum_{i=1}^{m} M_{n;i,j} - \alpha_j| \geq \varepsilon$ for
  any $j$ or $|\sum_{i=j}^{m} M_{n;i,j} - \alpha_i| \geq \varepsilon$ for any $i$ do
    Set $M_{r;i,\cdot} = M_{r;i,\cdot} \cdot \frac{\alpha_i(\beta_t)}{\alpha_i(\beta_{t-1})}$ for all $i$;
    Set $M_{c;\cdot,j} = M_{c;\cdot,j} \cdot \frac{\alpha_j(\beta_t)}{\alpha_j(\beta_{t-1})}$ for all $j$;
    Set $\boldsymbol{M}_n = \frac{\boldsymbol{M}_r + \boldsymbol{M}_c}{2}$;
    Set $\boldsymbol{M}_c = \boldsymbol{M}_n$ and $\boldsymbol{M}_r = \boldsymbol{M}_n$;
end
Set $\boldsymbol{M} = \boldsymbol{M}_n$
```

Algorithm 2: Reshaping GPUC matrices

5.1 Random Walk Metropolis-Hastings

To build a classical random walk Metropolis Hastings algorithm and use the posterior sample for parameter inference,two properties of GPUC parameter matrices are important:

Proposition 5.1.1. *Consider a GPUC parameter matrix* $\boldsymbol{M} \in \mathcal{M}_\beta$. *Further, set* $(i^*, j^*) = argmin_{i,j}\left(0.5 - \left|\frac{M_{i,j}}{\alpha_{max(i,j)}} - 0.5\right|\right)$. *Then, for two arbitrary Matrices* $\boldsymbol{G} \in \mathcal{M}_\beta$ *and* $\boldsymbol{H} \in \mathcal{M}_\beta$, $\boldsymbol{M}^* = \boldsymbol{M} + b(\boldsymbol{G} - \boldsymbol{H}) \in \mathcal{M}_\beta$, *if* $b \leq min\left(0.5 - \left|\frac{M_{i^*,j^*}}{\alpha_{max(i^*,j^*)}} - 0.5\right|, 0.5 - \left|\frac{\alpha_1 - g}{\sum_{i=2}^{m} \alpha_i} - 0.5\right|\right)$.

For the proof consider an arbitrary cell of $\boldsymbol{M}$, $M_{k,l} = g$ with $k \geq l$. Consequently no more than $\alpha_k - g$ can be additionally put in $M_{k,l}$ but by adding $b(G_{k,l} - H_{k,l})$ at most $b\alpha_k$ might be added and hence,

$$g + b\alpha_k \leq \alpha_k \Leftrightarrow b \leq 1 - \frac{g}{\alpha_k}.$$

[3] For a similar algorithm, the alternately scaling of rows and columns proof of convergence for positive matrices is given in Sinkhorn and Knopp (1967) (for the special case of Bernstein Copulas) and Sinkhorn (1967) (for the all GPUC matrices). A generalization to fully indecomposable matrices is found in Marshall and Olkin (1968).

On the other side, $M_{k,l}$ must not decreased by more than g if $k \neq 1$ or $l \neq 1$ or $\alpha_1 - \sum_{i=2}^{m} \alpha_i \leq 0$ and might be decreased at most by $b\alpha_k$ by adding $b(G_{k,l} - H_{k,l})$ and hence,

$$g - b\alpha_k \geq 0 \Leftrightarrow b \leq \frac{g}{\alpha_k}.$$

Accordingly, $b \leq 0.5 - \left| \frac{g}{\alpha_k} - 0.5 \right|$.
If $k = 1$ and $l = 1$ and $\alpha_1 - \sum_{i=2}^{m} \alpha_i > 0$, then $M_{1,1}$ might be decreased by no more than $g - \alpha_1 + \sum_{i=2}^{m} \alpha_i$ but by adding $b(G_{1,1} - H_{1,1})$ at most $b \sum_{i=2}^{m} \alpha_i$ might be subtracted. Hence,

$$g - b \sum_{i=2}^{m} \alpha_i \geq \alpha_1 - \sum_{i=2}^{m} \alpha_i \Leftrightarrow b \leq \frac{g - \alpha_1}{\sum_{i=2}^{m} \alpha_i} + 1$$

On the other hand, $M_{1,1}$ must not increased to more than α_1 and can be increased by adding $b(\boldsymbol{G}_{1,1} - \boldsymbol{H}_{1,1})$ by no more than $\sum_{i=2}^{m} \alpha_i$. Thus,

$$g + b \sum_{i=2}^{m} \alpha_i \leq \alpha_1 \Leftrightarrow b \leq \frac{\alpha_1 - g}{\sum_{i=2}^{m} \alpha_i}.$$

Consequently, $b \leq 0.5 - \left| \frac{\alpha_1 - g}{\sum_{i=2}^{m} \alpha_i} - 0.5 \right|$
The proof for $k \leq l$ follows analogously and is omitted, here.
Proposition 5.1.1 states that it is always possible to find a value b to add the difference of two GPUC matrices to another matrix so that this sum is still an eligible GPUC matrix, a property that will be helpful in formulating a RW-MH algorithm.

Proposition 5.1.2. *Consider two matrices $\mathbf{M}_{m \times m} \in \mathcal{M}_\beta$ and $\mathbf{N}_{m \times m} \in \mathcal{M}_\beta$. Then for $\lambda \in [0,1]$, $\lambda \boldsymbol{M} + (1-\lambda)\boldsymbol{N} \in \mathcal{M}_\beta$.*

For a proof it is sufficient to assure that $\sum_{i=0}^{m-1} \lambda M_{ij} + (1-\lambda) N_{ij} = \sum_{i=0}^{m-1} \lambda M_{ij} + \sum_{i=0}^{m-1} (1-\lambda) N_{ij} = \alpha_j$. All other properties to fit definition (6) are fulfilled, by definition.
With this at hand, it is now possible to construct a random walk Metropolis

Hastings algorithm the following way:

```
Initialize parameters $\boldsymbol{M}^{(\mathbf{0})}$ and $\beta^{(0)}$;
for $i$ in 1:N do
    Propose $\widetilde{\beta}^{(i)} = \beta^{(i-1)} + \varepsilon$, with $\varepsilon \sim N(0, \sigma_\beta)$;
    Recompute $\boldsymbol{M}^{(\boldsymbol{i})}$ using algorithm 2;
    Propose $\widetilde{\boldsymbol{M}}^{(i)} = \boldsymbol{M}^{(i)} + \gamma(\rho_1 - \rho_2)$, with $\rho_1, \rho_2 \sim F(\boldsymbol{M})$ and small $\gamma$;
    If $\widetilde{\boldsymbol{M}}^{(i)} \in \mathcal{M}$, accept $\widetilde{\boldsymbol{M}}^{(i)}$ with probability
     $a = min\left(\frac{p(\widetilde{\boldsymbol{M}}^{(i)}, \widetilde{\beta}^{(i)} | u,v)}{p(\boldsymbol{M}^{(i-1)}, \beta^{(i-1)} | u,v)}, 1\right)$
end
```

Algorithm 3: A random walk Metropolis-Hasting algorithm for GPUC.

In this algorithm, adding $b(\rho_1 - \rho_2)$ ensures the random walk property of the proposal. The decreasing scaling parameter b if helpful to ensure a relatively stable acceptance ratio, because $\widetilde{\boldsymbol{M}}^{(i)} \in \mathcal{M}$ holds for a large share of proposed matrices. On the down side a low scaling parameter b leads to poor mixing of the Markov chain and hence, slow convergence.

5.2 Random Blocking Random Walk Metropolis-Hastings

To overcome this issue, an alternative, random blocking sampling scheme is proposed. This sampler was originated by Chib and Ramamurthy (2010) in the framework of estimating large DSGE models. Chib and Ramamurthy (2010) propose to divide the random walk Metropolis-Hastings (RW-MH) algorithm into two steps. The first step consists of randomly generating blocks of parameters whereas the second step is the classical RW-MH over the randomly blocked parameters. Both, the blocks and parameters are drawn in each iteration. In the setting of estimating DSGE models "genetaring random blocks is [...] straightforward and does not require comment", in the setting of this paper, however, the method of generating blocks is essential, due to the nature of the parameter space. The randomized blocking random walk Metropolis Hastings

(RBRW-MH) algorithm used here can be summarized as follows:

```
Initialize $\boldsymbol{R} = \mathbf{1}_{m\times m}$; $\boldsymbol{M}^{(0)}$ and $\beta^{(0)}$;
for $t$ in 1:$N$ do
    Propose $\beta^{(t)} = \beta^{(t-1)} + \varepsilon$, with $\varepsilon \sim N(0, \sigma_\beta)$;
    Recompute $\boldsymbol{M}^{(t)}$ using algorithm 2;
    Accept $\left(\boldsymbol{M}^{(t)}, \beta^{(t)}\right)$ with probability $\alpha = min\left(\frac{p(\boldsymbol{M}^{(t)},\beta^{(t)}|u,v)}{p(\boldsymbol{M}^{(t-1)},\beta^{(t-1)}|u,v)}, 1\right)$;
    Set $\widetilde{\boldsymbol{M}}^{(0)} = \boldsymbol{M}^{(t)}$;
    for $r$ in 1:$(m^2/4)$ do
        Draw $i$ and $j$ from $\{1, \ldots, m\}$, with $R_{i,j} = 1$;
        Draw $k$ from $\{1, \ldots, m\} \setminus \{i, s : R_{i,s} = 0\}$;
        Draw $l$ from $\{1, \ldots, m\} \setminus \{j, t : R_{t,j} = 0\}$;
        Set $R_{i,j} = R_{k,l} = R_{i,l} = R_{k,j} = 0$;
        Draw $\varrho \sim N(0, \gamma)$;
        Set $\widetilde{\boldsymbol{M}}^{(r)} = \widetilde{\boldsymbol{M}}^{(r-1)}$;
        Compute $\widetilde{M}^{(r)}_{i,j} = \widetilde{M}^{(r)}_{i,j} + \varrho$, $\widetilde{M}^{(r)}_{i,l} = \widetilde{M}^{(r)}_{i,l} - \varrho$, $\widetilde{M}^{(r)}_{k,j} = \widetilde{M}^{(r)}_{k,j} - \varrho$,
         $\widetilde{M}^{(r)}_{k,l} = \widetilde{M}^{(r)}_{k,l} + \varrho$;
        Accept $(\widetilde{\boldsymbol{M}}^{(r)})$ with probability $\alpha = min\left(\frac{p(\widetilde{\boldsymbol{M}}^{(r)},\beta^{(t)}|u,v)}{p(\widetilde{\boldsymbol{M}}^{(r-1)},\beta^{(t)}|u,v)}, 1\right)$;
    end
    Set $\boldsymbol{M}^{(t)} = \widetilde{\boldsymbol{M}}^{(m^2/4)}$;
    Set $\boldsymbol{R} = \mathbf{1}_{m\times m}$;
end
```

Algorithm 4: Randomized Blocking Random Walk Metropolis Hastings for Generalized Partition of Unity Copulas.

The algorithm proposes a new value for β at the beginning of each iteration. Once β is proposed, the parameter matrix needs to be recomputed in order to have correct row and column wise sums. After accepting or rejecting the new value of β, the algorithm selects four elements from $\boldsymbol{M}$ according to Table 1 and adds ϱ to the elements (i, j) and (k, l) and subtracts ϱ from elements (i, l) and (k, j). The choice of ϱ in the algorithm ensures that the prequisites for proposition 5.1.1 are always met, and hence, the proposed matrix $\widetilde{\boldsymbol{M}}^{(r)} \in \mathcal{M}$.

$$M=\begin{bmatrix} M_{1,1} & \dots & M_{1,j} & \dots & M_{1,l} & \dots & M_{1,m} \\ \vdots & & \vdots & & \vdots & & \vdots \\ M_{i,1} & \dots & M_{i,j} & \dots & M_{i,l} & \dots & M_{i,m} \\ \vdots & & \vdots & & \vdots & & \vdots \\ M_{k,1} & \dots & M_{k,j} & \dots & M_{k,l} & \dots & M_{k,m} \\ \vdots & & \vdots & & \vdots & & \vdots \\ M_{m,1} & \dots & M_{m,j} & \dots & M_{m,l} & \dots & M_{m,m} \end{bmatrix}$$

Table 1: *Choosing four elements from M*

In order achieve a short burn in phase, start values can be set equal to:

$$\boldsymbol{M}_{ij}^{(0)} = \frac{1}{N}\sum_{k=1}^{N} \mathbf{1}_{\left(u_k \le \frac{i}{m};\ u_k > \frac{i-1}{m},\ v_k \le \frac{j}{m},\ v_k > \frac{j-1}{m}\right)}, \tag{14}$$

in case of the Bernstein Copula, where $f_b()$ is the density function of a beta distribution. Any zero values in $\boldsymbol{M}^{(0)}$ should be increased, marginally and afterwards algorithm 2 should be used so ensure that the initial matrices are actual GPUC parameter matrices. Avoiding any zeros in the initial matrix is necessary to prevent algorithms 3 and 4 from degenerating.
Once a new parameter matrix $\widetilde{\boldsymbol{M}}^{(r)}$ is proposed, the computing the likelihood $p(u, v|\widetilde{\boldsymbol{M}}^{(r)}, \beta^{(t)})$ boils down to calculating:

$$\begin{aligned} p\left(u, v|\widetilde{\boldsymbol{M}}^{(r)}, \beta^{(t)}\right) &= \prod_{s=1}^{N} c\left(u_s, v_s|\widetilde{\boldsymbol{M}}^{(r)}, \beta^{(t)}\right) \\ &= \prod_{s=1}^{N} c\left(u_s, v_s|\widetilde{\boldsymbol{M}}^{(r-1)}, \beta^{t}\right) \qquad +\varrho\left(\frac{\phi_{i-1}(u_s)\phi_{j-1}(v_s)}{\alpha_{i-1}\alpha_{j-1}}\right) \\ &\qquad\qquad +\varrho\left(\frac{\phi_{k-1}(u_s)\phi_{l-1}(v_s)}{\alpha_{k-1}\alpha_{l-1}}\right) \\ &\qquad\qquad -\varrho\left(\frac{\phi_{i-1}(u_s)\phi_{l-1}(v_s)}{\alpha_{i-1}\alpha_{l-1}}\right) \\ &\qquad\qquad -\varrho\left(\frac{\phi_{k-1}(u_s)\phi_{j-1}(v_s)}{\alpha_{k-1}\alpha_{j-1}}\right) \end{aligned}$$

and hence, the acceptance probability α can be computed very efficiently. In fact, a single iteration of the outer loop of algorithm 4 is of the same complexity as one iteration of the RW-MH sampler in Algorithm 3: $\mathcal{O}(m^2 n)$.

5.3 Adaptive proposal distributions

Allowing the scaling factor γ to be time varying (with respect to the iteration in the Markov chain), algorithms 3 and 4 can be used in an adaptive fashion. Determining a sufficiently large value of γ is crucial for both, achieving a well mixing Markov chain and effectively exploring the full posterior distribution, on the one hand, but too large values lead to a large fraction of proposed matrices that are not eligible GPUC matrices according to proposition 5.1.1, on the other hand. From this proportion of non eligible matrices it is possible to derive a strategy, when to decrease γ. In the case of the RW-MH, γ is to be decreased, once the proportion of non eligible matrices within the last 100 steps of the sampler exceeds a threshold, e.g. 85%. In case of the RBRW-MH algorithm, the author proposes to consider a newly proposed matrix as not eligible once more than 95% of all four-parameter-swaps lead to non eligible matrices. Subsequently, the adaption decision continues as in the RW-MH algorithm. With this adaption scheme at hand it is no longer important do find a suitable value of γ from the start, but it is possible to start with a too large value and decrease γ as the algorithm iterates.

5.4 Asymmetric proposal distributions

Each time, β is proposed in either algorithm 3 or algorithm 4, $\boldsymbol{M}$ needs to be altered according to algorithm 2, because row and column sum restrictions of $\boldsymbol{M}$ shift with changing β. Hence, the proposal distribution of β is, in fact, a joint proposal for β and $\boldsymbol{M}$. Denote by σ_β the standard deviation of the proposal distribution of β, then the implicit proposal of $\boldsymbol{M}$ can be skewed, if σ_β is too large (in comparison to β). Figure 2 shows the maximum of σ_β (exemplary for $\boldsymbol{M}_{4\times4}$) depending on β in order to guarantee low skewness[4]. The upper limits of σ_β were computed by minimizing

$$f(\omega) = \left| \frac{1}{R} \sum_{i=1}^{R} |\omega| - \underline{\omega} \right|,$$

where ω denotes skewness. This approach allows a typical value of skewness of $\underline{\omega}$ to remain in the proposal that is small enough to not distort the Markov chain of the posterior sample.

[4]Skewness was estimated using medcouple (Brys et al. (2004)) implemented the R package `robustbase`. Medcouple is a robust estimator of skewness and bounded by 1 in absolute value.

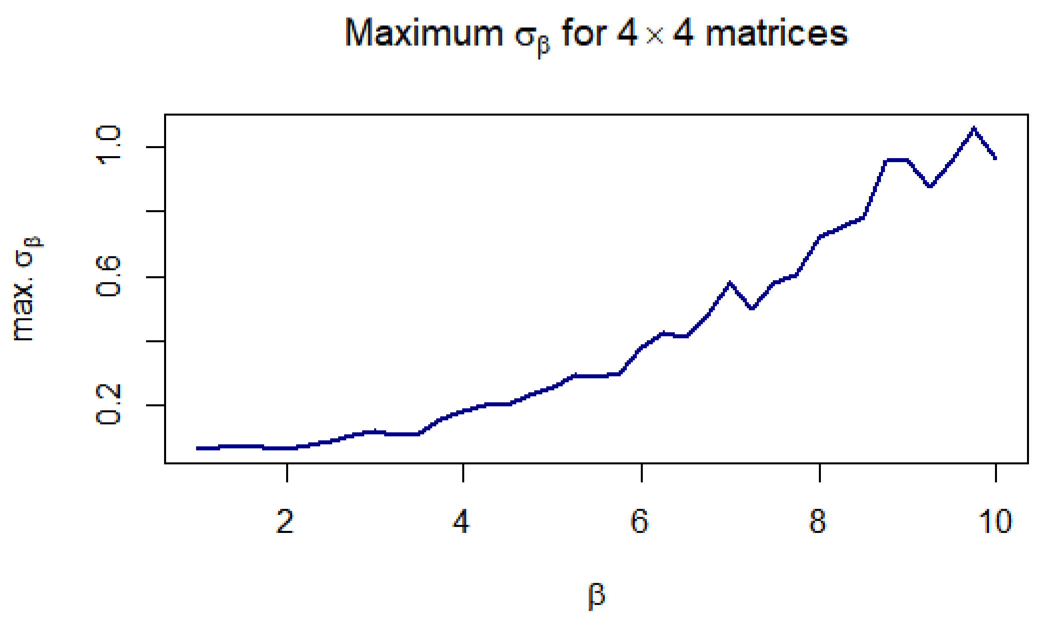

Figure 2: *Maximum σ_β to have a skewness of at most 0.05 for a 4×4 matrix $\boldsymbol{M}$ and negative binomial generating functions.*

Low β requires a very narrow proposal distribution to fulfill the conditions of a RW-MH algorithm, leading to slow exploration of the posterior and a strongly auto correlated posterior sample. Thus, a two stage estimation approach is proposed in case of low β.

```
First Stage: Run either of the samplers from Algorithms 3 and 4 for $N_1$
  iterations.
Compute $\hat{\beta} = \underset{i}{\operatorname{argmax}}\ p(\boldsymbol{M}_i, \beta_i | u, v)$.
Second Stage: Run either of the samplers for $N_2$ iterations with $\beta = \hat{\beta}$.
```

Algorithm 5: Two stage estimation scheme.

The first stage uses either sampler to get a sample from β in order to compute a point estimate that is used in the second stage to draw a sample from the (conditional) posterior of $\boldsymbol{M}$ given $\hat{\beta}$. The posterior mode is used as point estimate, here because the samplers don't necessarily sample from the posterior distribution and hence, median and mean of the sample in the first stage are likely to be distorted.

6 Applications

This section presents two applications of the proposed methods. The first subsection contains simulation studies of both Bernstein copula and negative binomial GPUC and compares the two approaches with respect to their convergence and performance. The second subsection applies the RBRW-MH sampler to a data set containing infection data according to Infektionsschutzgesetz (IfSG) in Germany.

6.1 Simulation studies

This section compares both proposed samplers by using simulated data sets. The first data set is generated from a negative binomial GPUC with $\beta = 4$ and a diagonal matrix $\boldsymbol{M}$. In this setting reduced form and normal form of the copula coincide and consequently, the dimension of $\boldsymbol{M}$ should not play a major role regarding the ability of the copula to adequately mimic the data. Both samplers are compared estimating a 4×4 matrix and a 10×10 matrix and are run for $15,000$ iterations. The second and third data sets are generated by Bernstein Copulas with parameter matrices

$$\boldsymbol{M}_1 = \begin{bmatrix} 1 & 1 & 1 & 13 \\ 1 & 1 & 13 & 1 \\ 1 & 13 & 1 & 1 \\ 13 & 1 & 1 & 1 \end{bmatrix} \cdot \frac{1}{64},$$

$$\boldsymbol{M}_2 = \begin{bmatrix} 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 \end{bmatrix}$$

and both samplers are run for $5,000$ iterations. All simulated data samples are of size $n = 5,000$. Table 3 shows the computing times (CT) in seconds and efficient sample sizes (ESS) of both samplers. The random blocking Metropolis-Hastings sampler outperforms the classical Metropolis-Hastings algorithm estimating the Bernstein Copula and both algorithms are close to each other in estimating the negative binomial copula. The time, necessary to compute a single, independent observation of the posterior [5] is similar between RBRW-MH algorithm (NB-GPUC: 48min and 57min; Bernstein copula: 0.044s and 0.26s) and RW-MH algorithm (NB-GPUC: 25min and 89min; Bernstein copula: 0.056min and 0.25min). By investigating both, the likelihood and trace plots for the Bernstein copula (see figure 3) it is clear, that in the case of a 4×4 matrix the RBRW-MH algorithm converged after roughly 200 iterations, whereas the RW-MH algorithm converged after $2,000$ iterations. In the case of a 10×10 matrix the RBWH converged after approximately 750 iterations whereas the RW-MH did not converge at all after $5,000$ iterations.

		ESS	**CT**	$max(lnL))$
NB GPUC 4×4	RW-MH	22	562	2958
	RBRW-MH	9	434	2958
NB GPUC 10×10	RW-MH	7	623	2950
	RBRW-MH	8	462	2925
Bernstein Copula 4×4	RW-MH	16	0.9	514
	RBRW-MH	72	3.2	514
Bernstein Copula 10×10	RW-MH	4	1	1461
	RBRW-MH	25	6.5	1700

Table 2: *Effective Sample Size (ESS), computing time (CT) and the maximum log likelihood of the proposed MCMC algorithms using simulated data.*

6.2 Campylobacteriosis infections

This section analyses a data set containing infections with Campylobacteriosis in Berlin and Brandenburg. Campylobacteriosis is the infection with Campylobacter bacteria and is a common human bacterial infections of the gastrointestinal system . In Germany, Campylobacteriosis is among the notifiable diseases and has to be reported to government authorities. Data was obtained from Robert Koch-Institut (2019) and comprises of weekly data from 2001 up to and including the tenth week of 2019, i.e. 949 weeks, in total. In total 326,349

[5]The efficient sample size (ESS) is a measure for the number of independent draws from the posterior; see eg. Gelman et al. (2004). Computing ESS was done using the *coda* package in R.

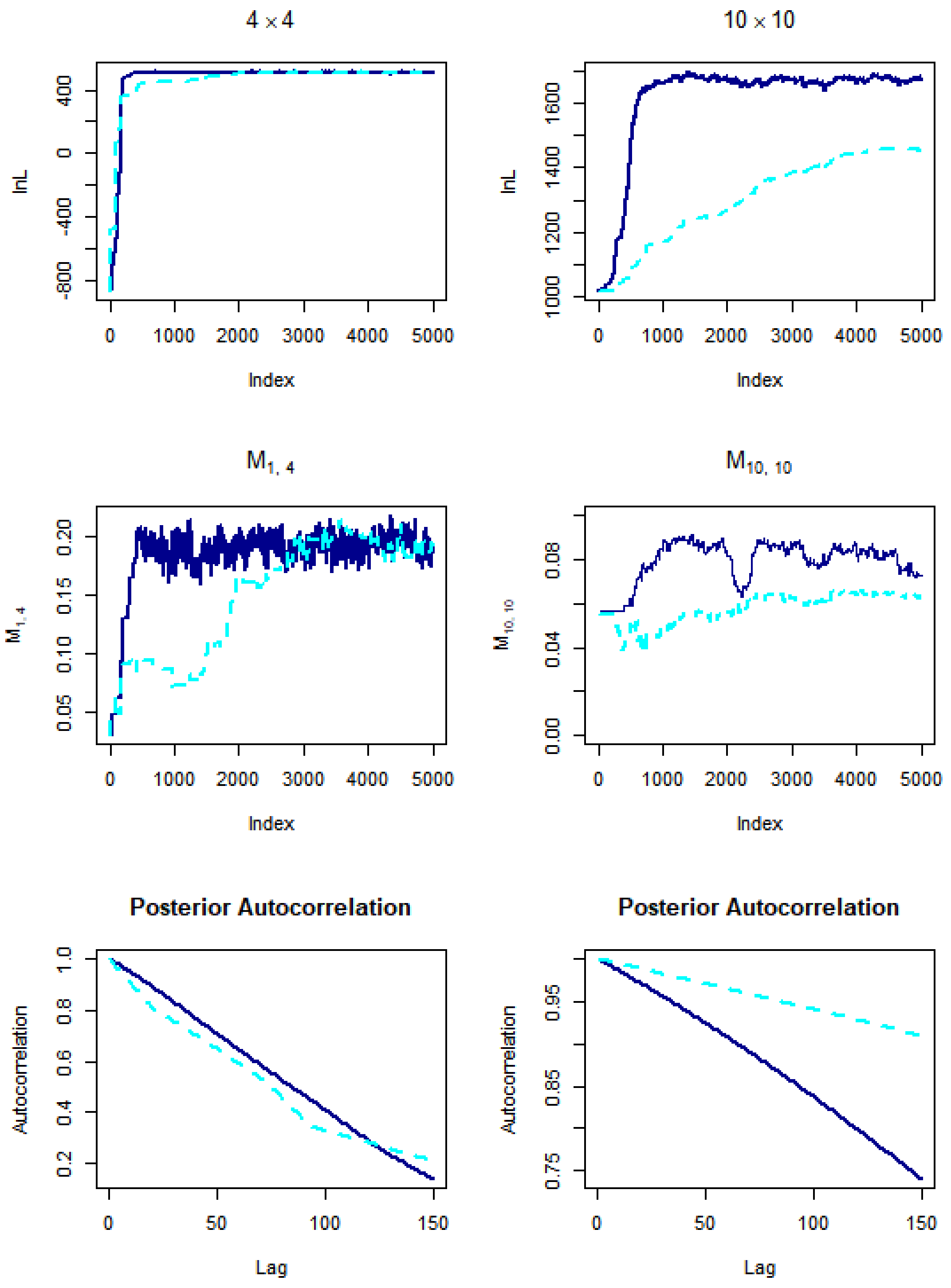

Figure 3: *Comparison of the RBRW-MH and the RW-MH algorithm when estimating Bernstein copulas. The left column shows results for a simulated data set with* $\boldsymbol{M} = \boldsymbol{M}_1$ *and the right column with* $\boldsymbol{M} = \boldsymbol{M}_2$*. The dark blue lines represent the RBRW-MH and the dashed cyan lines represent the RW-MH algorithm. Start values were set to a mixture of independent and diagonal Bernstein copulas. The second row shows the Markov chain of an exemplary matrix entry for each copula.*

infections are recorded in Brandenburg and 344,338 infections are recorded in Berlin. Increases and decreases of new infections usually occur within the same week in both regions and there is no indicator of a lead lag relationship between the regions[6]. Nevertheless, there is a strong (rank) correlation between the number of new infections in both regions with Kendall's τ of 0.55. The RBRW-MH algorithm is used to compare the fit of both, a negative binomial GPUC and a Bernstein copula for different matrix sizes ($m = 4, 6, 8, 10, 12$). Table 3 shows adjusted AIC and BIC for the mode of a posterior sample of $1,000 \cdot (5 + m)$ draws.Determining the optimal number of rows and columns $\hat{m}$ of $\boldsymbol{M}$ is crucial for the Bernstein copula to solve the trade-off between a smooth and a very detailed copula. Rose (2015) proposes to choose $\hat{m}$ for a d-dimensional Bernstein copula to minimize the mean squared error, according to

$$m_{opt} = |\theta| \left(n^{\frac{2}{d+4}} \right),$$

where θ is any measure of concordance. Unfortunately, this approach yields too small matrices in cases of strong dependency. As can be seen from figure 5 in the appendix, the minimum number of rows and columns of $\boldsymbol{M}$ to represent $\tau = 0.55$ is approximately 7 (the proposed value according to Rose (2015) would be approximately 5). Consequently, choosing $m = 7$ would result in an almost diagonal $\boldsymbol{M}$ and hence, does not leave room for any free parameters. To that end, both copulas are compared according to AIC and BIC, adjusted by the minimum dimensions of $\boldsymbol{M}$ required to achieve the Kendall's τ estimated from the data[7], i.e.

$$\begin{aligned} AIC_a &= -2lnL + 2(max(m - m_{min}, 0))^2 \\ BIC_a &= -2lnL + ln(n)(max(m - m_{min}, 0))^2 \end{aligned}$$

As table 3 indicates the best fit is achieved using a negative Binomial GPUC with a 4×4 matrix $\boldsymbol{M}$ and $\beta = 3.52$ leading to an upper tail dependence coefficient of approximately 0.71 according to Pfeifer et al. (2016). The best fitting Bernstein copula is given using a 10×10 matrix (AIC) or a 8×8 matrix

[6]This might also very well be due to the short incubation period of Campylobacteriosis of only a single to seven days and the weekly frequency of the data. The rank correlations between contemporaneous observations is 0.55, if Berlin leads it is 0.48 and if Brandenburg leads it is 0.50.

[7]Note that in the case of negative Binomial GPUCs adjusted and non adjusted AIC and BIC coincide.

	NB-GPUC			BC	
m	AIC	BIC	β	AIC_a	BIC_a
4×4	**-953**	**-857**	3.52	-634	-635
6×6	-900	-726	3.90	-732	-732
8×8	-844	-533	4.10	**-777**	**-772**
10×10	-769	-284	4.19	**-777**	-733
12×12	-675	≥ 0	4.33	-757	-636

Table 3: *Comparison of negative binomial GPUC and Bernstein copula using the RBRW-MH algorithm.*

(BIC), respectively.
A simulation of the fitted copulas (negative Binomial 4×4, Bernstein 10×10) can be found in figure 4. The dark blue dots are a random sample of size 10,000 drawn with the estimated parameters, the cyan dots depict the pseudo observations. Obviously, the data fit the inherent shape of the negative Binomial GPUC very well, whereas the Bernstein copula fails to capture the dependence in the upper tail, adequately.

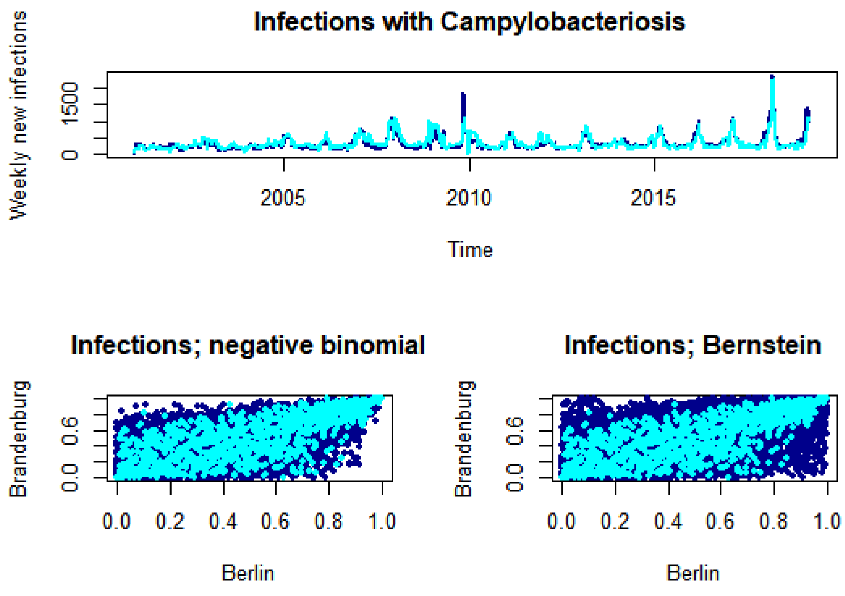

Figure 4: *New infections with Campylobacteriosis in Berlin and Brandenburg from 2001 to 2019 (top) and copula estimates: negative binomial GPUC (left) and Bernstein copula (right).*

7 Concluding Remarks

This paper proposes the first MCMC based algorithms to estimate GPUCs. An algorithm to sample random GPUC matrices is introduced and proven and building on that and using the properties of GPUC matrices, a random walk Metropolis-Hastings is proposed. To further improve efficiency of the sampler, random blocking is added to the algorithm utilizing the restrictions on row and column sums of GPUC matrices in order to change only a small number of parameters at a time. It is shown that the likelihood can be computed very efficiently within each step of the random blocking and hence, both algorithms have the same complexity. Simulation studies show that the RBRW-MH algorithm at least as efficient as the RW-MH algorithm (and is much more efficient in case of the Bernstein copula).
In order to allow unbiased estimation of GPUC matrices, conditionally (on β) flat priors are introduced, based on the volume of the sampling space of GPUC matrices, utilizing importance sampling.
In the end, the negative binomial GPUC proofs to be a very accurate tool of describing real world data, in this case new infections with Campylobacteriosis in Berlin and Brandenburg.

8 Appendix

8.1 Log-polynomial fit of the prior

m	$const.$	β	β^2	β^3	β^4	β^5	$log(beta)$
3	−8.168	−2.700	0.264	−0.017	0.001	−0.00001	3.661
4	−24.396	−5.269	0.475	−0.030	0.001	−0.00001	8.523
5	−52.010	−8.234	0.690	−0.042	0.001	−0.00002	15.414
6	−92.579	−11.511	0.908	−0.054	0.002	−0.00002	24.353
7	−147.882	−14.515	1.029	−0.057	0.002	−0.00002	35.040
8	−217.445	−18.972	1.360	−0.077	0.002	−0.00003	48.653
9	−307.830	−20.788	1.363	−0.088	0.004	−0.0001	61.793
10	−416.362	−24.294	1.089	−0.039	0.001	−0.00001	80.206
11	−543.687	−32.864	1.984	−0.101	0.003	−0.00003	102.120
12	−709.766	−29.688	0.539	0.052	−0.004	0.0001	118.946
13	−891.356	−35.922	0.527	0.074	−0.005	0.0001	144.568
14	−1,123.319	−24.221	−2.140	0.317	−0.015	0.0002	153.370
15	-1,337.284	-67.008	5.310	-0.426	0.019	-0.0003	208.712
16	-1,658.292	-42.240	1.154	-0.091	0.006	-0.0001	210.521
17	-1,943.941	-73.886	4.443	-0.301	0.013	-0.0002	265.396
18	-2,333.162	-52.893	-2.322	0.467	-0.023	0.0003	289.127
19	-2,750.608	-50.188	-1.457	0.220	-0.008	0.0001	311.079
20	-3,158.849	-79.790	1.214	0.115	-0.006	0.0001	368.077

Table 4: *Estimates of the log-polynomial model for a conditionally flat prior of the negative binomial GPUC.*

8.2 Minimum matrix size

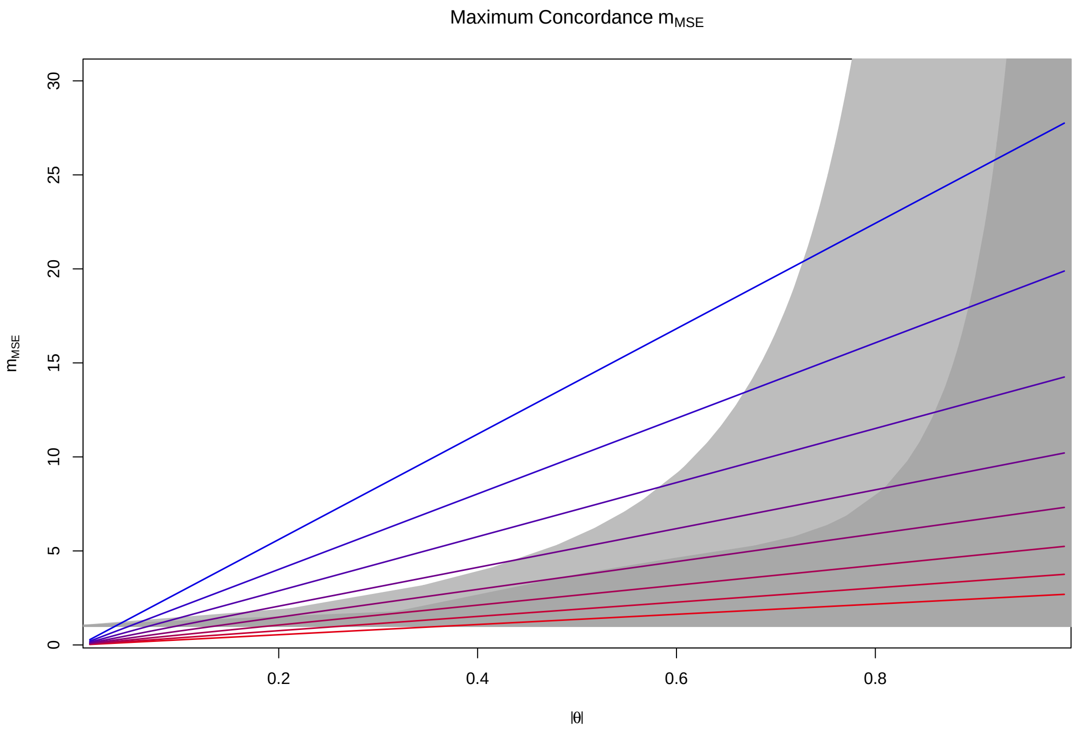

Figure 5: *Minimum grid size m, necessary to model dependence levels given by Kendall's τ (light gray) or Spearman's ρ (gray). The lines display the estimators of m for sample sizes from e^2 (red) to e^9. m is chosen to minimize the. mean squared error*

8.3 Beta-Mixture representation of negative Binomial GPUC and Bernstein copula.

Consider a random variable $X \sim Beta(p, q)$, then the density of X is given by

$$f_X^{(p,q)}(x) = \frac{(p+q-1)!}{(p+1)!(q+1)!} x^{p-1}(1-x)^{q-1}.$$

The density function of a negative binomial GPUC with parameter matrix $\boldsymbol{M}$ and shape parameter β is given by

$$\begin{aligned} c(u,v) &= \sum_i \sum_j \frac{M_{i,j}}{\alpha_i \alpha_j} \binom{\beta+i-1}{i} \binom{\beta+j-1}{j} (1-u)^\beta (1-v)^\beta u^i v^j \\ &= \sum_i \sum_j \frac{M_{i,j}}{\alpha_i \alpha_j} \frac{(\beta+i-1)!}{i!(\beta-1)!} \frac{(\beta+j-1)!}{j!(\beta-1)!} (1-u)^\beta (1-v)^\beta u^i v^j. \end{aligned}$$

Now, for the negative Binomial GPUC:

$$\alpha_i = \int_0^1 \binom{\beta+i-1}{i} u^i (1-u)^\beta du = \frac{\beta}{(\beta+i)(\beta+i+1)}$$

and hence,

$$\begin{aligned} c(u,v) &= \sum_i \sum_j \Bigg(M_{i,j} \frac{(\beta+i)(\beta+i+1)(\beta+j)(\beta+j+1)}{\beta^2} \frac{(\beta+i-1)!}{i!(\beta-1)!} \frac{(\beta+j-1)!}{j!(\beta-1)!} \\ &\qquad (1-u)^\beta (1-v)^\beta u^i v^j \Bigg) \\ &= \sum_i \sum_j M_{ij} \frac{(\beta+i+1)!(\beta+j+1)!}{i!\beta j!\beta} (1-u)^\beta (1-v)^\beta u^i v^i \\ &= \sum_i \sum_j M_{ij} f^{(i+1,\beta+1)}(u) f^{(j+1,\beta+1)}(v) \quad \square \end{aligned}$$

The density function of a Bernstein copula with parameter matrix $\boldsymbol{M}_{(m \times m)}$ and $\alpha_i = \frac{1}{m}$ is given by:

$$\begin{aligned} c(u,v) &= \sum_i \sum_j m^2 M_{ij} \binom{m-1}{i} \binom{m-1}{j} u^i (1-u)^{m-i-1} v^j (1-v)^{m-j-1} \\ &= \sum_i \sum_j M_{ij} \frac{m!}{i!(m-i-1)!} \frac{m!}{j!(m-j-1)!} u^i (1-u)^{m-i-1} v^j (1-v)^{m-j-1} \\ &= \sum_i \sum_j M_{ij} f^{(i+1,m-i)}(u) f^{(j+1,m-j)}(v) \quad \square \end{aligned}$$

References

Baker, R., 2008. An order-statistics-based method for constructing multivariate distributions with fixed marginals. Journal of Multivariate Analysis 99, 2312–2327. doi:10.1016/j.jmva.2008.02.019.

Bedford, T., Cooke, R.M., 2002. Vines–a new graphical model for dependent random variables. The Annals of Statistics 30, 1031–1068. doi:10.1214/aos/1031689016.

Brys, G., Hubert, M., Struyf, A., 2004. A robust measure of skewness. Journal of Computational and Graphical Statistics 13, 996–1017. doi:10.1198/106186004X12632.

Chib, S., Ramamurthy, S., 2010. Tailored randomized block mcmc methods with application to dsge models. Journal of Econometrics 155, 19 – 38. URL: http://www.sciencedirect.com/science/article/pii/S0304407609001900, doi:https://doi.org/10.1016/j.jeconom.2009.08.003.

Dou, X., Kuriki, S., Lin, G.D., Richards, D., 2016. Em algorithms for estimating the bernstein copula. Computational Statistics & Data Analysis 93, 228–245. doi:10.1016/j.csda.2014.01.009.

Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B., 2004. Bayesian data analysis. Texts in statistical science. 2. ed. ed., Chapman & Hall, Boca Raton, Fla.

Greenberg, E., 2008. Introduction to Bayesian econometrics. Cambridge University Press, Cambridge.

Joe, H., 2001. Multivariate models and dependence concepts. volume 73 of *Monographs on statistics and applied probability*. 1. crc reprint ed., Chapman & Hall/CRC, Boca Raton, Fla.

Marshall, A.W., Olkin, I., 1968. Scaling of matrices to achieve specified row and column sums. Numerische Mathematik 12, 83–90. doi:10.1007/BF02170999.

Nelsen, R.B., 2006. An Introduction to Copulas. Springer Series in Statistics. second edition ed., Springer Science+Business Media Inc, New York, NY. doi:10.1007/0-387-28678-0.

Pfeifer, D., Mändle, A., Ragulina, O., 2017. Data driven partition-of-unity copulas with applications to risk management URL: https://arxiv.org/abs/1703.05047v2.

Pfeifer, D., Tsatedem, H.A., Mändle, A., Girschig, C., 2016. New copulas based on general partitions-of-unity and their applications to risk management. Dependence Modeling 4, 225. doi:10.1515/demo-2016-0006.

Robert Koch-Institut, 2019. Survstat@rki 2.0. URL: https://survstat.rki.de.

Rose, D., 2015. Modeling and estimating multivariate dependence structures with the bernstein copula. URL: http://nbn-resolving.de/urn:nbn:de:bvb:19-187572.

Savu, C., Trede, M., 2010. Hierarchies of archimedean copulas. Quantitative Finance 10, 295–304. doi:10.1080/14697680902821733.

Sinkhorn, R., 1967. Diagonal equivalence to matrices with prescribed row and column sums. The American Mathematical Monthly 74, 402. doi:10.2307/2314570.

Sinkhorn, R., Knopp, P., 1967. Concerning nonnegative matrices and doubly stochastic matrices. Pacific Journal of Mathematics 21, 343–348. doi:10.2140/pjm.1967.21.343.