Chapter 1

Results

1.1 Examples

In this section, we will investigate how the algorithms Algorithm 1 and Algorithm 2 works in junction and, if so, observe how the algorithm can fail and what may be done to correct such cases. Initially, a few simple examples involving exponentiated multivariate Gaussians $\boldsymbol{Y}$.

Example 1.1. *Exponentiated multivariate Gaussian*

Let us consider a simple case with $\mathbf{Y} = e^{\mathbf{X}}$ *(element wise exponentiation) where* $X \sim \mathcal{N}(\mathbf{0}, \Sigma)$ *where*

$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0.9\sigma_1\sigma_2 & 0 \\ 0.9\sigma_1\sigma_2 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{bmatrix}$$

It is clear that to Algorithm 1, the mean is of no importance as it simply corresponds to a scaling of the Y_i *variables. Furthermore, because of Corollary 1.2.1, theoretically, due to the uniqueness of the Copula* C *(as* $\boldsymbol{Y}$ *is continuous) we should expect near equal or very similar results for* $\boldsymbol{Y}$ *and* $\boldsymbol{X}$ *from Algorithm 1. Additionally, different* σ *corresponds to different scaling of* $\boldsymbol{X}$*, and thus we*

should observe equal or near equal G_{dir} for all $\boldsymbol{Y}$. Initially, we shall see how this hypothesis holds up to the following three examples

$$\boldsymbol{\sigma} = (0.07, 0.3, 0.9), \quad \boldsymbol{\sigma} = (1, 1, 1), \quad \boldsymbol{\sigma} = (1, 2, 3)$$

In order for the sample size to not influence the results, we simulate a generous number of samples, namely, for the following results we have used $n = 10{,}000$ samples. For $\boldsymbol{\sigma} = (1, 1, 1)$, Algorithm 1 and Algorithm 2 returns the following (using $\alpha = 1$ and $\beta = 0.99$)

$$G_{dir} = \begin{bmatrix} -0.33396 & 0.6660 & 0.02512 \\ 0.6660 & -0.3341 & 0.02730 \\ 0.02512 & 0.02730 & -0.0020583 \end{bmatrix} \tag{1.1}$$

Similarly, for $\boldsymbol{\sigma} = (0.07, 0.3, 0.9)$:

$$G_{dir} = \begin{bmatrix} -0.3335 & 0.6665 & 0.01414 \\ 0.6665 & -0.3335 & 0.01418 \\ 0.01414 & 0.01418 & -0.00060124 \end{bmatrix} \tag{1.2}$$

Finally, for $\boldsymbol{\sigma} = (1, 2, 3)$:

$$G_{dir} = \begin{bmatrix} -0.1490 & 0.09535 & 0.3599 \\ 0.09535 & -0.2989 & 0.5831 \\ 0.3599 & 0.5831 & -0.4037 \end{bmatrix}$$

For $\boldsymbol{\sigma} = (1, 1, 1)$ and $\boldsymbol{\sigma} = (0.07, 0.3, 0.9)$ we observe the most resemblance to the Σ, although the resulting G_{dir} deviate in the final column. The difference is likely produced by Algorithm 1 as if the resulting G_{obs} was the same, then so would G_{dir} and from the above argument, we know that theoretically this should be the case. For the final example, $\boldsymbol{\sigma} = (1, 2, 3)$, we see a completely different result and immediately suspect that there must be some numerical errors. Investigating the partial results of Algorithm 1 we immediately see a flaw in the supposedly uniform variables U_i as shown in figure Figure 1.1

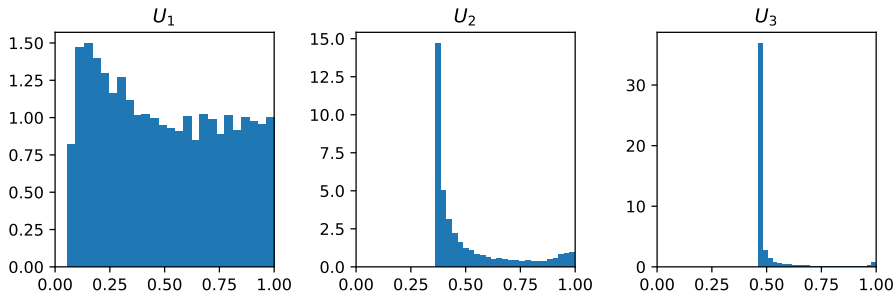


Figure 1.1: The samples transformed using $U_i = F_i(X_i)$ for $\boldsymbol{\sigma} = (1, 2, 3)$. These should be uniformly distributed, but clearly this is not the case for U_2 and U_3. Even U_1 does not quite resemble 10,000 samples from a uniform distribution.

	U_1	U_2	U_3
D_n	0.066209	0.36014	0.46285
p-value	0	0	0

Table 1.1: based on 10,000 samples for $\boldsymbol{\sigma} = (1, 2, 3)$.

Before handling this, the non-uniformity of U_1 in Figure 1.1 is likely also present in the case when $\boldsymbol{\sigma} = (1, 1, 1)$. Indeed, Figure 1.2 shows that this is indeed the case.

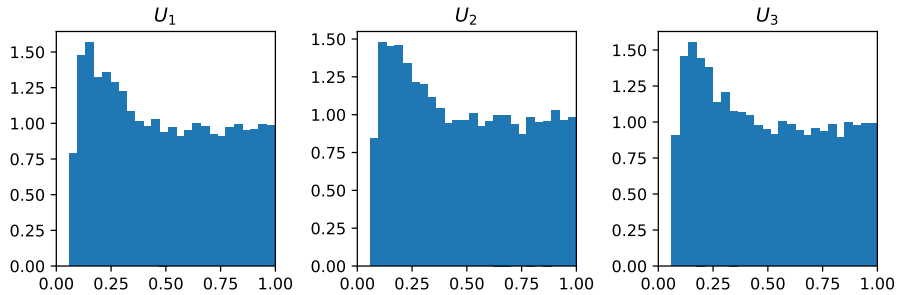


Figure 1.2: The samples transformed using $U_i = F_i(X_i)$ for $\boldsymbol{\sigma} = (1, 1, 1)$.

	U_1	U_2	U_3
D_n	0.068408	0.066808	0.070809
p-value	0	0	0

Table 1.2: based on 10,000 samples for $\boldsymbol{\sigma} = (1, 1, 1)$.

Finally, just to be sure, $\boldsymbol{\sigma} = (0.07, 0.3, 0.9)$ is also shown in Figure 1.3 and seems very reasonable, except for U_3.

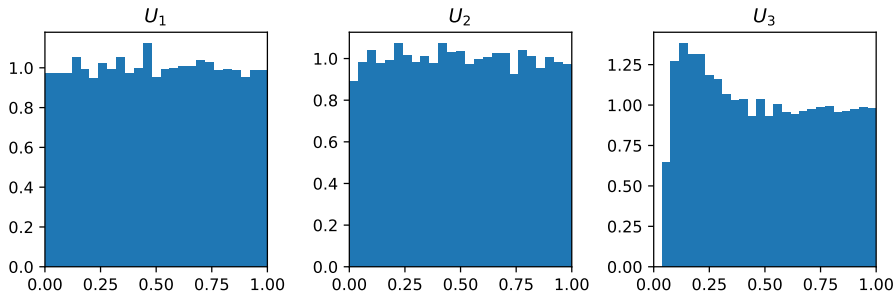


Figure 1.3: The samples transformed using $U_i = F_i(X_i)$ for $\boldsymbol{\sigma} = (0.07, 0.3, 0.9)$.

	U_1	U_2	U_3
D_n	0.00581897	0.0068066	0.050908
p-value	0.88645	0.74179	0

Table 1.3: based on 10,000 samples for $\boldsymbol{\sigma} = (0.07, 0.3, 0.9)$.

From the above examples, it seems that the larger the variance, the worse the uniforms turn out. Reasons for this could include numerical issues when trying to calculate $u_i^{(j)}$ *form* $y_i^{(j)}$ *by* $u_i^{(j)} = \int_{-\infty}^{y_i^{(j)}} f_i(y)\,dy$ *and bad fitting of the kernel density estimate from observations. In particular, for values similar, which happens in the case for large* σ *such that we observe large negative realizations of* X_i, $y_i^{(j)}$ *are almost 0, and when computing the integral could result in identical values. Furthermore, from Figure 1.4 we see that indeed the fit is quite poor. Note that we have zoomed in on the interval* $[-200, 200]$ *which contains* 96.2% *of observations. The poor fit is primarily due to the use of Scott's Rule as discussed above which in this case overshoots the optimal bandwidth by a lot.*

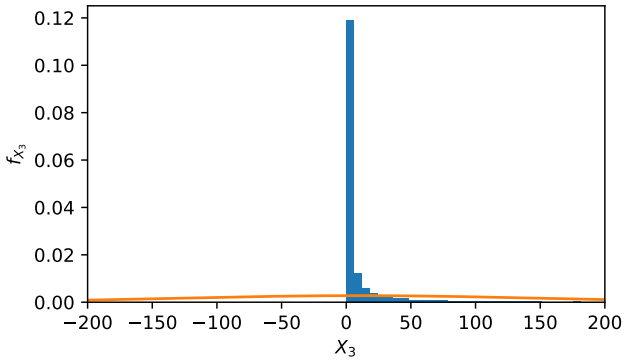


Figure 1.4

The poor fit also explains the high concentration of U_3 *around* 0.5 *in Figure 1.1 as only* 54.5% *of the probability mass lies above* 0.

However, also here Corollary 1.2.1 proves to be useful. Namely, we can get rid of the numerical issues by transforming Y_i *using e.g.* $\log(\cdot)$ *or* $(\cdot)^p$ *for* $p > 0$ *to get even out the observations more. As the first simply inverts the initial transformation of* X_i*, we choose the latter as a more interesting case. In particular, choosing* $p < 1$ *will result in a more even distribution. In the following,* $p = 1/10$ *has been used to transform* $\boldsymbol{Y}$ *prior to running Algorithm 1 and the resulting* $u_i^{(j)}$ *is shown in Figure 1.5.*

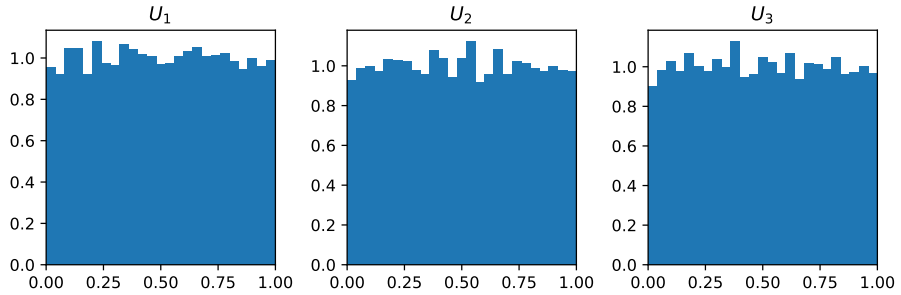


Figure 1.5

The resulting $u_i^{(j)}$ *now seem to follow a uniform distribution and indeed the KDE fits much better as seen in Figure 1.6.*

	U_1	U_2	U_3
D_n	0.0061099	0.0061435	0.0073148
p-value	0.84838	0.84368	0.65690

Table 1.4: based on 10,000 samples for $\boldsymbol{\sigma} = (1, 2, 3)$ with power transform.

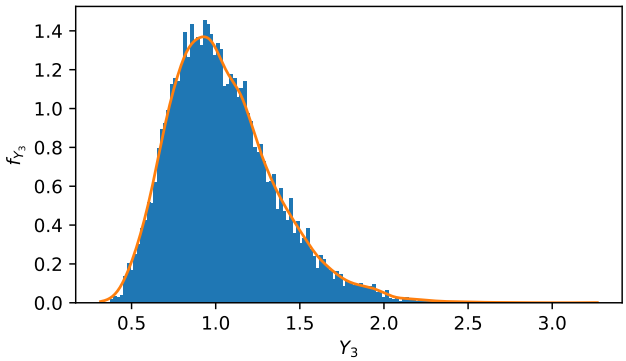
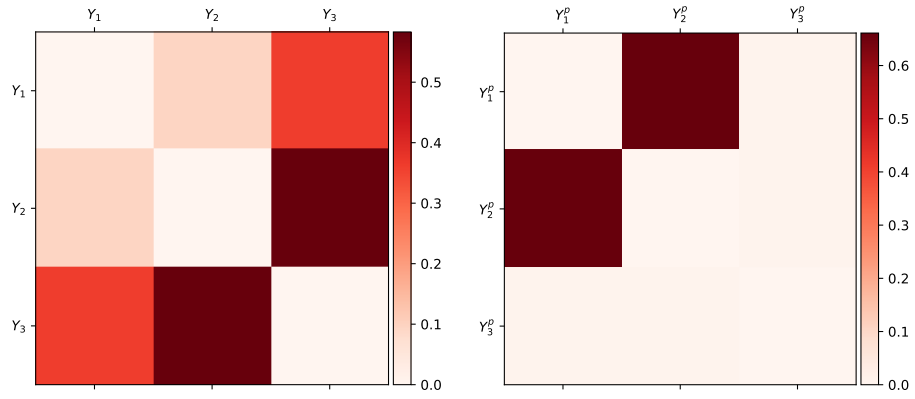


Figure 1.6

Turning to Algorithm 1 and Algorithm 2 we now find that G_{dir} is given by

$$G_{dir} = \begin{bmatrix} -0.3290 & 0.6610 & 0.008440 \\ 0.6610 & -0.3290 & 0.008150 \\ 0.008440 & 0.008150 & -0.0002061 \end{bmatrix}$$

Which is indeed much more comparable with the result from before in Equation 1.1 and Equation 1.2. The difference between G_{dir} from $\boldsymbol{Y}$ and $\boldsymbol{Y}^p$ is clearly visible in Figure 1.7 and also Figure 1.7b resembles the original correlation structure.



(a) **(b)**

Figure 1.7: G_{dir} resulting from 10,000 samples from multi variate Gaussian with $\boldsymbol{\sigma} = (1, 2, 3)$ in **(a)** with raw samples from $\boldsymbol{Y}$ and in **(b)** the transformed data corresponding to $\boldsymbol{Y}^p$.

Finally, to end this example we shall compare with some theoretical results. Namely, the output G_{obs} of Algorithm 1 can also be calculated theoretically. For this, we shall use Proposition 1.11 which permits a theoretical result, namely

$$G_{obs} = \begin{bmatrix} 0 & -\frac{1}{2}\ln\left(1-\rho_{12}^2\right) & -\frac{1}{2}\ln\left(1-\rho_{13}^2\right) \\ -\frac{1}{2}\ln\left(1-\rho_{21}^2\right) & 0 & -\frac{1}{2}\ln\left(1-\rho_{23}^2\right) \\ -\frac{1}{2}\ln\left(1-\rho_{31}^2\right) & -\frac{1}{2}\ln\left(1-\rho_{32}^2\right) & 0 \end{bmatrix} \cong \begin{bmatrix} 0 & 0.83037 & 0 \\ 0.83037 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Similarly, prior to deconvolution, using just the sampled $\boldsymbol{X}$ (i.e. no exponential transform), Algorithm 1 returns

$$G_{obs} = \begin{bmatrix} 0. & 0.71841756 & 0.01781815 \\ 0.71841756 & 0. & 0.01769672 \\ 0.01781815 & 0.01769672 & 0. \end{bmatrix}$$

Test om denne G er lige den teoretiske. Eller nærmere, argumenter for hvorfor vi ikke laver en test, eller hvad man kunne gøre. Har samplet fra en simultan normalfordeling, så kan lave en til en mellem MI og korrelation.

From the confidence density for the correlation ρ given the emperical correlation

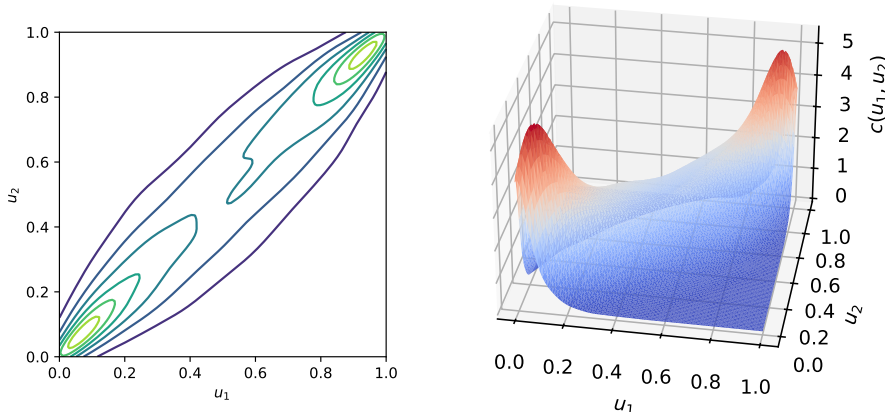
r is given by

$$f(\rho \mid r, \nu) = \frac{\nu(\nu-1)\Gamma(\nu-1)}{\sqrt{2\pi}\Gamma(\nu+\frac{1}{2})} \frac{\left(1-r^2\right)^{\frac{\nu-1}{2}} \left(1-\rho^2\right)^{\frac{\nu-2}{2}}}{(1-r\rho)^{\frac{2\nu-1}{2}}} F\left(\frac{3}{2}, -\frac{1}{2}, \nu+\frac{1}{2}, \frac{1+r\rho}{2}\right)$$

from the mutual information, we can calculate the absolute correlation. Notice that the density does not change when reversing both r and ρ simultaniously, thus, wlog, assume $r \geq 0$, then we can calculate a CI for ρ (which will be negated if we had used $-r$ instead and thus would be identical when taking the absolute value). If the original CI $[a, b]$ contains 0 i.e. $a < 0$, we shall write the CI for the absolute correlation as $[0, b]$ instead. This way, we can compare the absoulte correlations and see if they are the same (by checking if the CI contains the theoretical correlation) by [?]. Using numerical integration (fast enough with high numerical accuracy from many bins, 1 mil bins, yielding probability mass 1.0000000000008133), can compute CI for absolute correlation

??

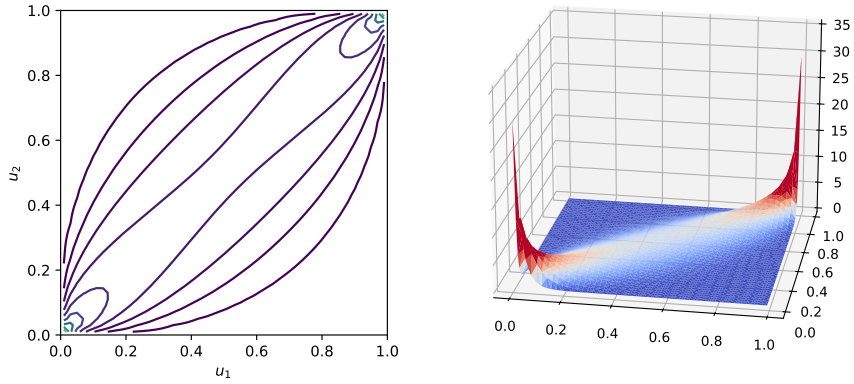
Clearly these are not equal, but in this case, the error is suspected to originate from the estimated joint density. For example, considering X_1 and X_2, we compare the estimated joint copula density and compare to the theoretical reference til et sted hvor gausisk copula står shown in Figure 1.8 and Figure 1.9 respectively.



(a) **(b)**

Figure 1.8: Estimated copula density c with $\rho = 0.9$ corresponding to X_1 and X_2.

The noticeable difference is in the corners $(0,0)$ *and* $(1,1)$ *where the theoretical copula density tends to infinity whereas the estimated density has modes at* $(0.1,0.1)$ *and* $(0.9,0.9)$*. In particular, simply rescaling the copula density in Algorithm 1 does not resemble the theoretical boundary which is a known issue reference til artikel om undershoot peaks og boundary conditions for KDE. A better approach may be to use jackknifing link til afsnit of jackknifing, som også indeholder reference til artikel hvor dette gøres.*



(a) **(b)**

Figure 1.9: Theoretical copula density c with $\rho = 0.9$ corresponding to X_1 and X_2.

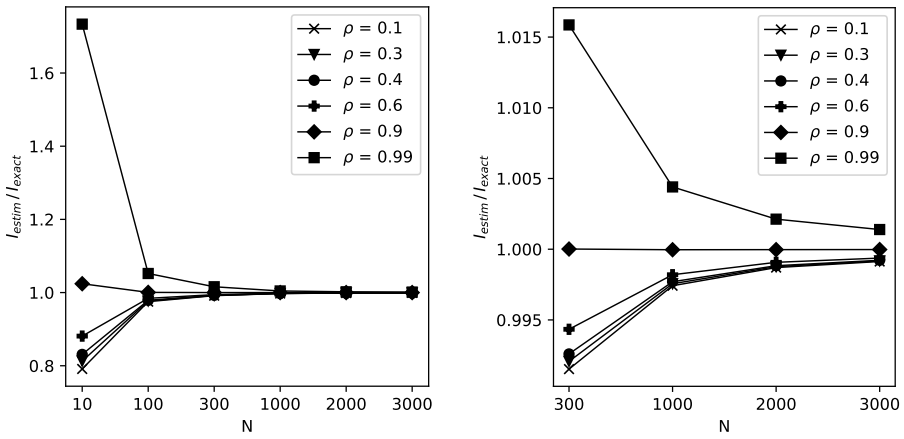
We note however, that the underlying structure is still captured i.e. that Y_1 *and* Y_2 *covary while* Y_3 *does not inform* Y_1 *or* Y_2 *and vice versa.*

We continue with a similar example to the previous one. The key difference is the number of variables and a more complicated correlation structure to test the algorithms further.

Example 1.2. *From Example 1.1 we saw how one could handle some numerical issues. Thus, in this example we shall not bother ourselves with such computations and merely focus on the correlation structure. In particular, we shall sample* $\boldsymbol{X}$ *from a 10 dimensional*

1.2 sammenligning af metoder for at finde MI

Sammenligning af gammel metode og "min"



(a) **(b)**

Figure 1.10: Evaluation of MI for new method for different N. Bør sammenlignes med artiekl fundet (har sat i bibtex) og original papers (ikke Kina)

ved høj korrelation i.e. tæt på laver dimensionel manifold, skal der bruges mange, som i rigtig mange samples i mesh.

Inkluder flere exempler end blot gaussian as done by [?]

1.2.1 10D gaussian example

casuality svarer til at lave nedre/øvre trekant. Er der forskel i at gør edet før og efter for en symmetrisk matrix? - Ja, men begge metoder på 10 eksempel giver gode resultater. Kommenter at det er matematisk meget forskelligt at filtrere først og så ND efter og omvendt

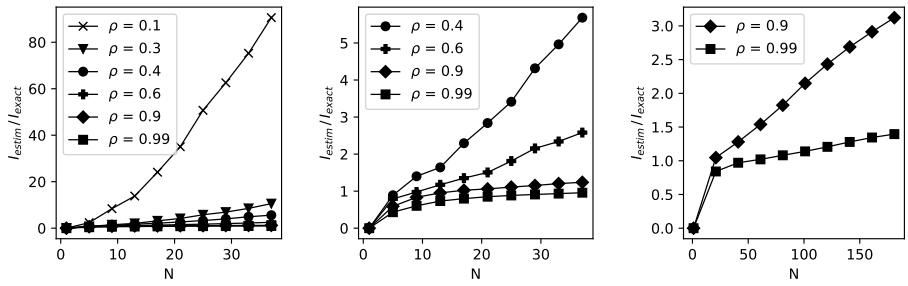


Figure 1.11: Evaluation of MI for old method for different N. Ligner der er knæk ved fporhold lig 1. Men ved næremere undersøgelse blev det fundet ud af at det ikke helt er tilfældet, og derudover vil der skulle laves en algoritmisk måde at finde dette knæk på. Savitzky–Golay filter kunne være en mulighed, eller gruppere e.g. 5 forskellige bins og tag gennemsnit. Efter smoothing kan anden afledte tæt på 0 bruges, til at finde hvornår stykket bliver fladt (tilnærmelses vist)