`COVSIM`: AN R PACKAGE FOR SIMULATING NON-NORMAL DATA FOR STRUCTURAL EQUATION MODELS USING COPULAS

STEFFEN GRØNNEBERG, NJÅL FOLDNES, AND KATERINA M. MARCOULIDES

Abstract. This introduction to the R package `covsim` is a slightly modified version of Grønneberg, Foldnes & Marcoulides (2022), accepted for publication in the *Journal of Statistical Software*. In this version of the paper, references of code in the online supplementary material should be read as references to Appendix B on p. 42. When Grønneberg et al. (2022) is assigned a journal volume, this material will also be available on the website of the Journal of Statistical Software.

In factor analysis and structural equation modeling non-normal data simulation is traditionally performed by specifying univariate skewness and kurtosis together with the target covariance matrix. However, this leaves little control over the univariate distributions and the multivariate copula of the simulated vector. In this paper we explain how a more flexible simulation method called vine-to-anything (VITA) may be obtained from copula-based techniques, as implemented in a new R package, covsim. VITA is based on the concept of a regular vine, where bivariate copulas are coupled together into a full multivariate copula. We illustrate how to simulate continuous and ordinal data for covariance modeling, and how to use the new package discnorm to test for underlying normality in ordinal data. An introduction to copula and vine simulation is provided in the appendix.

1. Introduction

Structural equation modeling (SEM) and factor analysis are regularly applied to data in the psychological, educational, business, behavioral, and medical sciences. The central component in these methods is the covariance matrix from which the model parameters are identified. In this article we present software for simulating from a class of distributions with a fixed covariance matrix, which therefore can be used in SEM simulation studies. This distributional class is more flexible than the methods currently in use, and may therefore extend the range of conditions investigated with simulations.

When the examined data are continuous, the most popular SEM estimation method used is the normal-theory based maximum likelihood (NTML) method (Jöreskog, 1967) which is asymptotically efficient when data are normally distributed. NTML estimation is a special case of a class of moment based estimators known as minimum discrepancy function estimators (see e.g., Shapiro, 1983), and is therefore known to be consistent also under non-normality. When using classical standard errors with NTML, valid inference is attained mainly when the data are normal. While several standard error and test statistic formulas have been proposed in order to robustify inference with NTML and other minimum discrepancy estimators under non-normality (e.g., Marcoulides, Foldnes & Grønneberg, 2019; Satorra & Bentler, 1988; Wu & Lin, 2016), their performance depends heavily on the distribution and sample size of the data (e.g., Curran, West & Finch, 1996; Foldnes & Olsson, 2015; Fouladi, 2000; Grønneberg & Foldnes, 2019b). In settings with ordered-categorical data, least squares estimation based on polychoric correlations is the most prevalent estimation method (Christoffersson, 1977; Muthén, 1984). Polychoric correlations are essentially the correlations among continuous bivariate normally distributed vectors underlying the observed ordinal data, but may be heavily biased outside normality (Foldnes & Grønneberg, 2019b,2). Hence, under both continuous and ordinal data analyses, the normality assumption is a central starting point for estimation and inference.

Unfortunately, in most empirical research situations data are seldom drawn from populations in which the normality assumption holds exactly (Cain, Zhang & Yuan, 2017; Micceri, 1989). And while several estimation and inference methods that do not assume normality have been suggested (for an overview, see Tarka, 2018), it is in most conditions not feasible to analytical derive results on their performance as a function of the data generating distribution. Monte Carlo simulation studies have as a result become essential tools for evaluating the behavior of various aspects of SEM techniques, such as parameter and standard error bias, performance of test statistics, and power calculations, relative to distributional characteristics (Boomsma, 2013). The external validity of these studies is weakened if the chosen data generation mechanism does not resemble the real-world distributions encountered in the relevant field of practice. To be able to model such distributions, we need simulation methods that can match a given covariance matrix but still offer distributional flexibility.

The aims of the present paper are twofold. Firstly, to present the R (R Core Team, 2020) package covsim (Foldnes & Grønneberg, 2020a), which implements non-normal data simulation methods proposed by Grønneberg & Foldnes (2017) and Foldnes & Olsson (2016). Both methods generate data with a prescribed covariance matrix. Our emphasis will be on the former method, called vine-to-anything (VITA), since it offers greater flexibility that we deem particularly useful to SEM methodologists. For instance, the flexibility of VITA renders it uniquely well-suited for being employed in simulation studies with ordinal SEM, as further discussed in Section 6.2.

VITA is a simulation method based on vine copulas. Copulas are multivariate distributions with uniform marginals, and vine copulas is a special type of copulas. Since copulas, vines and multivariate simulation theory are not well known to SEM practitioners and methodologists, the second aim of the paper is to introduce these topics during our presentation of the VITA method and the covsim package. We also include a technical appendix with an elementary though mathematically complete introduction to multivariate simulation theory with vine copulas, as this seems to be missing from the literature.

We next give an overview of statistical software for drawing data from non-normal multivariate distributions with a predefined covariance matrix. The classical and still most frequently used approach is that of Vale & Maurelli (1983), where the user specifies the univariate skewness and kurtosis. This method is currently the only option in popular commercial software such as EQS (Bentler, 2006) and LISREL (Jöreskog & Sörbom, 2006), and in the widely used R package lavaan (Rosseel, 2012). Other approaches that also focus on controlling moments have recently been proposed. The independent generator approach proposed by Foldnes & Olsson (2016) can match pre-specified univariate skewness and kurtosis, and is more flexible than the Vale-Maurelli method. This method is available in the `rIG` function in package covsim, and its use is described in a later section. Recently Qu, Liu & Zhang (2019) used independent generator variables in a method which controls multivariate skewness and kurtosis, at the expense of control over univariate skewness and kurtosis. This method is available in package mnonr (Qu & Zhang, 2020). A method that fully controls the univariate distributions (not only the lower-order moments) is the NORTA method of Cario & Nelson (1997), which is implemented in package SimCorMultRes (Touloumis, 2016). The Vale-Maurelli, independent generator and NORTA approaches have the great benefit of being technically easy to analyse and implement. For instance, the technical tractability allows the asymptotic covariance matrix of the empirical covariances to be exactly calculated (Foldnes & Grønneberg, 2017b). The simplicity of the methods also allows for fast simulation. However, the simplicity and speed of these methods come at a cost: NORTA always has a normal copula (Cario & Nelson, 1997), while Vale-Maurelli in most cases has a normal copula (Foldnes & Grønneberg, 2015). This means that the true multivariate dependence structure does not depart from that of the multivariate normal distribution. In addition, only NORTA completely controls the univariate marginals. To the best of our knowledge, besides the approach taken in the present article, there is only one method that offers some control of the copula when simulating from distributions with a given covariance matrix. Mair, Satorra & Bentler (2012) proposed a two-stage data-generation process where a very large sample is first simulated from a copula combined with marginal specification, whose distribution we denote by F_{pre}. Then the inverse of a square root of the sample covariance matrix from this large sample is computed. To simulate data, in the second stage a sample of desired size is drawn from F_{pre}, and multiplied by first the inverse of the square root matrix from the previous stage and then by a square root matrix of the target covariance. The two-stage approach guarantees that the rows are *iid*, and it follows from construction that the simulated vector has the correct population covariance matrix. Also, the simulated vector has a non-normal copula, provided F_{pre} was chosen to have a non-normal copula. However, both the margins and the copula are distorted by the post-multiplication of square root matrices. That is, although F_{pre} is fully specified in terms of multivariate copula and univariate distributions, the simulated vector does not inherit this copula nor the margins and control is lost both in terms of copula and marginal distributions. Mair et al. (2012) illustrated their code using common multivariate copula families using Gumbel and Clayton copulas, as implemented in package copula (Hofert, Kojadinovic, Maechler & Yan, 2013). An implementation of this method is available in package simsem (Pornprasertmanit, Miller, Schoemann & Jorgensen, 2020). The flexibility of this implementation is presently limited to a rather restricted class of multivariate copulas, comprising elliptical, Archimedean, extreme-value and some other copula families available in package copula.

VITA improves upon the approach of Mair et al. (2012) by allowing complete control of p the marginal distributions, of the bivariate copulas of a chosen set of $p-1$ pairs of variables, and of certain conditional bivariate copulas of the remaining $(p-1)(p-2)/2$ pairs of variables. The increased degree of control and flexibility of our approach relative to existing methods is made possible by employing the powerful multivariate copula-based construction called a regular vine. A primary aim of the present article is to present and illustrate our approach using the newly developed covsim package.

The remainder of the article is organized in the following way. First, we explain the copula in a two-dimensional setting. We then demonstrate a very flexible copula-based approach to non-normal simulation in the two-dimensional case. An important advantage of this approach is that the copula class and the exact marginal distributions of the two-dimensional case may be fully specified by the user. Next, we develop the full multivariate extension, which still allows for complete control of the marginal distributions, and considerable flexibility in the dependence structure. We then detail the implementation of the covsim package, and end the paper with two additional examples: first we show how to simulate from a non-normal continuous SEM with fixed parameters, and then we show how to simulate data for ordinal SEM. Example code is provided throughout the paper, and complete replication code is available in the online supplementary material. The appendix provides an introduction to implementing the simulations on a computer, and follows the progression of the paper.

Throughout this article we illustrate the capacity of covsim in the context of simple structural equation modeling settings. We use only a limited set of non-normal distributional conditions in each illustration, and we caution that the external validity of our findings is therefore limited. To enhance the validity a larger range of distributional and sample size conditions must be included.

2. The bivariate case

We start by considering the bivariate case. Our aim is to introduce the concept of a copula and how it can be used to simulate non-normal random variables with a given correlation. In subsequent sections we extend our simulation procedure to the general multivariate case. For a textbook treatment of copula theory, see Nelsen (2007). Note that this book, together with most books on copulas, assume that the reader has a strong mathematical background, including some measure theory, and that we do not assume such a background in the current presentation. Some useful introductory papers on copulas which can be read without such a background are Frees & Valdez (1998); Genest & Favre (2007); Yan et al. (2007).

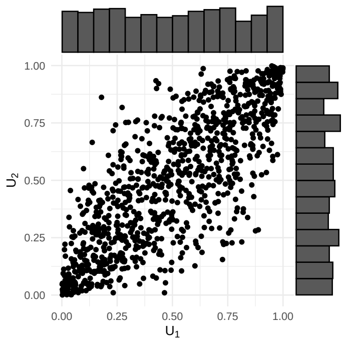
A copula is a distribution with uniform univariate margins. Copulas are used to describe the dependency structure between variables, when taking the marginal distributions out of the equation. There are many classes of copulas, and within each class there is typically a parameter that controls the strength of dependence. We start with the normal copula. Let $\Phi(x)$ denote the cumulative distribution function (CDF) of a standard normal distribution, and let $\Phi^2(x, y; \rho)$ denote the CDF of the bivariate standard normal distribution with correlation parameter ρ, i.e., $\Phi^2(x, y; \rho) = P(Z_1 \leq x, Z_2 \leq y)$ where Z_1, Z_2 are bivariate normal and standardized, and have correlation ρ. Then the normal copula with parameter $\rho \in (-1, 1)$ is given by

$$C^N(u_1, u_2; \rho) = \Phi^2(\Phi^{-1}(u_1), \Phi^{-1}(u_2); \rho).$$

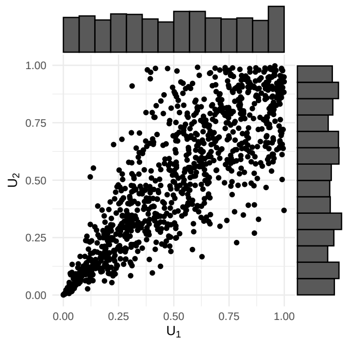
As an example of a non-normal copula class, consider Clayton copulas, which are parametrized by the dependence parameter $\theta \in (0, \infty)$:

$$C^{Cl}(u_1, u_2; \theta) = (u_1^{-\theta} + u_2^{-\theta} - 1)^{-1/\theta}.$$

Clayton copulas are useful for modeling lower tail dependence, a measure of dependence between two variables in the lower left tail of the joint distribution. Figure 1 depicts random draws of size $n = 1000$ from each of these copulas. We set $\rho = 0.8$ for the normal copula and $\theta = 3.4$ for the Clayton copula. In both Figure 1a and 1b we see that the marginal empirical distributions are close to uniform. A notable difference is the lower tail dependence in Figure 1b which does not appear in Figure 1a.



(A) Normal copula with $\rho = 0.8$.



(B) Clayton copula with $\theta = 3.4$.

FIGURE 1. Random samples of size $n = 1000$ drawn from two bivariate copulas.

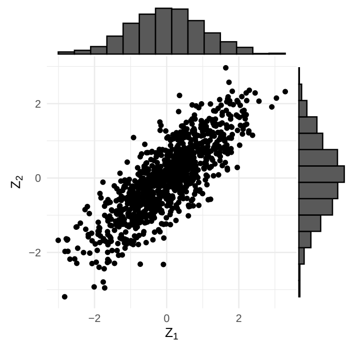
Bivariate copulas are important since they constitute one of two fundamental building blocks for bivariate distributions. The other building block consists of the two univariate marginal distributions. A fundamental theorem (Sklar, 1959) guarantees that any bivariate distribution may be decoupled into a bivariate copula and the two marginal distributions, and vice versa; given two marginal distributions $F_1(x_1)$ and $F_2(x_2)$ and a copula $C(u_1, u_2; \theta)$, then

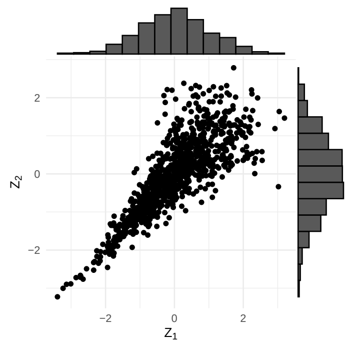
$$F(x_1, x_2) := C(F_1(x_1), F_2(x_2); \theta) \tag{1}$$

is a valid bivariate CDF, whose univariate margins are distributed according to $F_1(x_1)$ and $F_2(x_2)$. For instance, if F_1 and F_2 are the standard normal distribution, the bivariate distributions stemming from the normal copula with $\rho = 0.8$, and from the Clayton copula with $\theta = 3.4$, will both result in bivariate distributions with standard normal marginals, and with a Pearson correlation of 0.8. That is, setting $\theta = 3.4$ yields a Clayton copula such that when combined with standard normal marginals will yield a distribution with $\rho = 0.8$. Figure 2 shows random samples from these two distributions, obtained by applying the standard normal quantile function to the observations in Figure 1, which will change the marginals of the simulated data to be standard normal. Figure 2 depicts two very different bivariate distributions. Although sharing the same standard normal marginal distributions, and the same correlation coefficient 0.8, it is clear that the distribution in Figure 2b is far from the bivariate normal distribution in Figure 2a.

This illustration hints at the following process for a researcher that wants to simulate data from a bivariate distribution with pre-specified covariance and univariate marginals:

(1) Specify marginal distributions F_1 and F_2 and specify a target covariance.





(A) Normal distribution with $\rho = 0.8$. (B) Clayton distribution with $\theta = 3.4$.

FIGURE 2. Random samples of size $n = 1000$ drawn from two bivariate distributions with standard normal marginals and correlation 0.8.

(2) Specify a bivariate copula class $C(u_1, u_2; \theta)$, with dependence parameter θ.
(3) Use a numerical procedure to determine θ_0 so that the coupled distribution $C(F_1(x_1), F_2(x_2); \theta_0)$ has the pre-specified covariance.

For a given set of marginals, and a given copula, the set of attainable covariances is usually constrained. Then, in step 3 there is no solution θ_0. In such a case, the copula class or the marginal specifications should be adjusted.

The three steps are conducted in the covsim package in R as follows.

```
R> library("covsim")
R> mnorm <- list(list(distr = "norm"), list(distr = "norm"))
R> sigma.target <- matrix(c(1, 0.8, 0.8, 1), 2)
R> set.seed(1)
R> calibrated.vita <- vita(mnorm, target, family_set = "clayton")
R> summary(calibrated.vita)
R> library("rvinecopulib")
R> cov(rvine(10^5, calibrated.vita))
```

In the last two lines we verify that the target covariance matrix has been attained, using the `rvine` function from package rvinecopulib (Nagler & Vatter, 2019). This package offers fast simulation from vines.

As indicated above, we may simulate from a distribution of the form $C(F_1(x_1), F_2(x_2); \theta)$ by first simulating (U_1, U_2) from the copula C, and then apply the quantile functions of the marginals to each coordinate. That is, $(F_1^{-1}(U_1), F_2^{-1}(U_2))$ has marginals F_1, F_2 and copula C, meaning its full distribution equals $C(F_1(x_1), F_2(x_2); \theta)$. See the technical appendix for an explanation for why this is so and how to simulate from a copula.

3. The trivariate case: Introducing vines

In the previous section we studied bivariate copulas, and the calibration of their dependence parameter so that the coupling of given marginals will meet a target covariance. There are many classes of bivariate copulas, but few classes of higher-dimensional copulas. In this section we will circumvent the lack of parametric multivariate copula classes by using a statistical construction called a regular vine (Bedford & Cooke, 2002). Vines allow us to construct multivariate copula distributions by combining two-dimensional copulas. For the purpose of covariance modeling and simulation, the procedure detailed here was originally proposed by Grønneberg & Foldnes (2017).

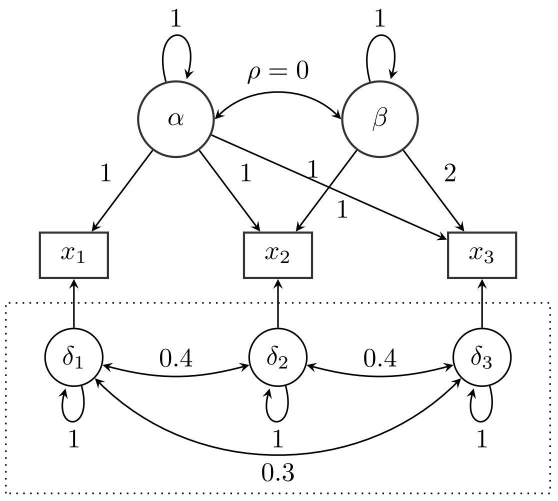


FIGURE 3. Linear growth curve population model with correlated residual errors.

Let us first proceed to the case of three variables. Our goal is to construct distributions with given marginal univariate distributions for each of the three variables, and with a given 3×3 covariance matrix. Imagine a researcher is concerned with whether non-normal correlated errors in growth curve modeling may affect the quality of inference for the correlation ρ between the intercept and slope factors. The default in growth curve modeling is to assume that residual errors are mutually independent across measurement occasions. However, correlated errors may be meaningful as they represent carryover effects from previous occasions not accounted for by the intercept and linear slope latent variable (Grimm & Widaman, 2010; Marcoulides, 2019). The issue of how residual error structures in latent growth curve modeling should be specified (e.g., as constrained, free, independent, autocorrelated, homogeneous, or non-homogeneous) is currently of great concern in the SEM literature, as it is now becoming much more recognized that considerable bias in the latent variable variance–covariance matrix can arise from the improper specification of these errors (Dimitrov, 2002; Grimm & Widaman, 2010; Laenen, Alonso, Molenberghs & Vangeneugden, 2009; Marcoulides, 2019; Van De Schoot, Sijbrandij, Winter, Depaoli & Vermunt, 2017).

The researcher wants to simulate data with correlated residual errors and sets up a simple linear growth model, see Figure 3, where the population values are indicated: There is zero correlation ρ between the slope and the intercept, all latent variables have unit variance, and the errors $\delta = (\delta_1, \delta_2, \delta_3)'$ are correlated with covariance matrix

$$\Sigma_\delta = \begin{pmatrix} 1 & 0.4 & 0.3 \\ 0.4 & 1 & 0.4 \\ 0.3 & 0.4 & 1 \end{pmatrix}.$$

The researcher is concerned with non-normality in the error vector δ, and therefore wants to construct a trivariate non-normal distribution whose covariance is Σ_δ.

We remark that the illustrations in the present article do not discuss the reasons behind specific choices of marginal distributions and dependence structure. Such choices depend on the purpose of the simulation study. Our aim in this and following illustrative analyses is simply to demonstrate how vine constructions work. However, we note that routines exist to select best-fitting vine structures and bicopula families relative to an existing real-world dataset (e.g., function `vinecop` in package rvinecopulib). This could be done to increase external validity of simulation studies. For an example of how to construct a VITA distribution based on a well-known empirical dataset, see Grønneberg & Foldnes (2017, Sec. 3.2). General papers on the selection and usage of copulas are Embrechts, Lindskog & McNeil (2001); Genest & Favre (2007); Grønneberg & Hjort (2014); Yan et al. (2007), an influential paper on vine based modeling is Aas, Czado, Frigessi & Bakken (2009), and a book with practical issues on vine modeling is Joe & Kurowicka (2011).

First, the researcher considers the three univariate error distributions, and decides that the first error should be standard normally distributed, the second error should be a scaled chi-squared distribution with one degree of freedom (DF), and the third error should follow a scaled Student's t distribution with five DFs. The scalings are necessary to obtain unit variance in the two latter distributions. Clearly, it might be questioned whether the case of different error distributions for the same variable at different measurement occasions is realistic, but since our main purpose here is to illustrate the flexibility of VITA, we proceed with three different error distributions.

Even though the marginal distributions in δ have now been specified, there are still many trivariate distributions with these marginals and with covariance Σ_δ. The specification will be complete once the copula of δ is selected, but this must be done cautiously and in a way that ensures δ has covariance Σ_δ. Let us denote the copula as V. The joint distribution of δ will result when we couple together three marginals using V. That is, the CDF F_δ of δ is given by

$$F_\delta(a, b, c) = V(\Phi(a), G_1(b), G_2(c)),$$

where G_1 and G_2 are the CDFs of the scaled chi-square and t distributions, respectively. As in the bivariate case, simulation from the above distribution involves first simulating (U_1, U_2, U_3) from V, and then applying the quantile functions of the marginals to each coordinate of this vector. That is, the final simulated vector will be $\delta = (\Phi^{-1}(U_1), G_1^{-1}(U_2), G_2^{-1}(U_3))$. To construct the vine V the researcher decides to couple the uniform marginals U_1 and U_2 with a Clayton copula, and U_2 and U_3 with a Joe copula. The dependence parameters of each of these bivariate copulas is numerically determined as described in the previous section, so that $\mathrm{corr}(\Phi^{-1}(U_1), G_2^{-1}(U_2)) = \mathrm{corr}(G_2^{-1}(U_2), G_3^{-1}(U_3)) = 0.4$. The hard part is now to couple U_1 with U_3 such that $\mathrm{corr}(\delta_1, \delta_3) = \mathrm{corr}(\Phi^{-1}(U_1), G_3^{-1}(U_3)) = 0.3$, and to achieve this we next introduce the concept of a vine.

Vines are convenient graphical tree structure models that can be used to build up high dimensional distributions from conditional two-dimensional copulas. Vines therefore decompose the multivariate copula into a hierarchy of bivariate copulas. A vine on p variables can be represented as a set of connected trees $V = \{T_1, ..., T_{p-1}\}$, where the edges of tree j are the nodes of tree $j+1$, $j = 1, \ldots, p-2$ and are used to facilitate the picking out of various distributional characteristics, see Figure 4 for our current illustration with $p = 3$. The first tree has the variables as its nodes, and an edge between two variables means that these two variables are unconditionally coupled as in the previous section. In our case, we chose at the beginning to couple U_1 with U_2 and to couple U_2 with U_3. This corresponds to the tree at the bottom of Figure 4. The second tree has the edges of the first tree as its nodes. In our case the first tree has only two edges: U_1, U_2 and U_2, U_3. The second tree must therefore join U_1, U_2 and U_2, U_3. This tree has one single edge, which is denoted by $U_1, U_3|U_2$. That is, the second tree specifies the copula between U_1 and U_3, conditional on U_2. Note that we could have chosen a different tree at the first level, with edges, say, U_1, U_2 and U_1, U_3, which would yield a different distribution. Also the bivariate copulas chosen for coupling pairs of variables could have been chosen differently, yielding other types of vine distributions. However, we do not explore the flexibility of vines in the present paper. We see in Figure 4 that there are a total of three edges in the vine, and that the edges correspond to the pairwise correlations among the three variables. This holds also in higher-dimensional vines: There is an exact correspondence between the edges in the set of trees, and pairs of variables. So each off-diagonal element in the covariance matrix corresponds to a unique edge in the vine. The researcher's goal now is to define the distribution of U_1 and U_3. As suggested in Figure 4, this is done by specifying the distribution of U_1 and U_3, conditional on U_2. The researcher chooses a Frank copula for this distribution.

In terms of the joint density function f of $\delta = (\delta_1, \delta_2, \delta_3)'$, its general form is:

$$f(a,b,c) = f_1(a)f_2(b)f_3(c) \cdot c_{12}(F_1(a), F_2(b)) \cdot c_{13}(F_1(a), F_3(c)) \cdot c_{13|2}(F_{1|2}(a|b), F_{3|2}(c|b)),$$

where f_1, f_2, f_3 are chosen marginal density distributions of $\delta_1, \delta_2, \delta_3$ respectively, where $c_{12}, c_{13}, c_{13|2}$ are the chosen bivariate copulas of (δ_1, δ_2), (δ_1, δ_3) and (δ_1, δ_3) conditioned on δ_2, respectively. Also, F_1, F_2, F_3 are CDFs of $\delta_1, \delta_2, \delta_3$ respectively, and $F_{1|2}, F_{3|2}$ are conditional CDFs of δ_1 given δ_2, and of δ_3 given δ_2, respectively. These CDFs are consequences of the chosen bivariate copulas and marginals. A full discussion with formulas for these conditional CDFs and of the joint density is included in the appendix. An advantage of vine distributions, compared to other multivariate simulation approaches where covariance matrices are specified (e.g, Qu et al., 2019; Ruscio & Kaczetow, 2008), is the above explicit formula for the distribution of the simulated vector.

Returning to the illustrative example, we sum up the researcher's specifications for the residual error vector δ:

- δ_1 follows a standard normal distribution, δ_2 follows a scaled chi-square distribution with one DF, and δ_3 follows a scaled t distribution with five DFs.
- The vine structure is given in Figure 4.
- A Clayton copula density for c_{12}, with dependence parameter calibrated so that $\Phi^{-1}(U_1)$ and $G_2^{-1}(U_2)$ have correlation 0.4.
- A Joe copula density for c_{23}, with dependence parameter calibrated so that $G_2^{-1}(U_2)$ and $G_3^{-1}(U_3)$ have correlation 0.4.

- A Frank copula density for $c_{13|2}$, with dependence parameter calibrated so that $\Phi^{-1}(U_1)$ and $G_2^{-1}(U_3)$ have correlation 0.3.

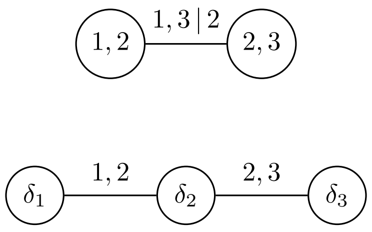


FIGURE 4. A three-dimensional regular vine.

To construct multivariate distributions where the marginals and the covariances are pre-specified, Grønneberg & Foldnes (2017) proposed the use of vines, resulting in the VIne-To-Anything (VITA) method. In our illustration, the researcher's requests may be fulfilled by constructing a VITA distribution using the covsim package as follows:

```
R> sigma.target <- matrix(c(1, 0.4, 0.3, 0.4, 1, 0.4, 0.3, 0.4, 1), 3)
R> margins <- list(list(distr = "norm"), list(distr = "chisq", df = 1),
+    list(distr = "t", df = 5))
R> pcs <- list(list(bicop_dist("clayton"), bicop_dist("joe")),
+    list(bicop_dist("frank")))
R> vine_cop <- vinecop_dist(pcs, structure = dvine_structure(1:3))
R> margin.variances <- c(1, 2, 5/3)
R> pre <- diag(sqrt(margin.variances/diag(sigma.target)))
R> vita.target <- pre %*% sigma.target %*% pre
R> set.seed(1)
R> calibrated.vita <- vita(margins, target.vita, vc = vine_cop, verbose = T)
R> post <- diag(1/diag(pre))
R> vita.sample <- rvine(10^5, calibrated.vita) %*% post
R> round(cov(vita.sample) - sigma.target, 2)
```

In the last lines of code above, we simulated a $n = 10^5$ sample from the calibrated VITA distribution using the function `rvine` from the R-package rvinecopulib (Nagler & Vatter, 2019). The purpose of the last line is to confirm that the covariance matrix in the simulated sample is close to the target matrix. For the six non-rendundant elements in the covariance matrix, the mean absolute deviation was $5 \cdot 10^{-4}$.

A visualization of $n = 1000$ randomly drawn error vectors is presented in Figure 5. Note that, expectedly, the first marginal distribution is approximately standard normal, while the second and third marginal distributions are in accordance with scaled chi-square and Student's t distributions.

Now, having constructed a VITA distribution for the residual errors, the researcher may use simulation to assess whether the quality of NTML inference for ρ, the correlation between the intercept and slope factors, deteriorates under non-normal residual errors. As a benchmark, the researcher simulates from a fully normal distribution on the observed variables x_1, x_2, and x_3. For the non-normal case, the researcher first simulates VITA residual errors, and then combines these with

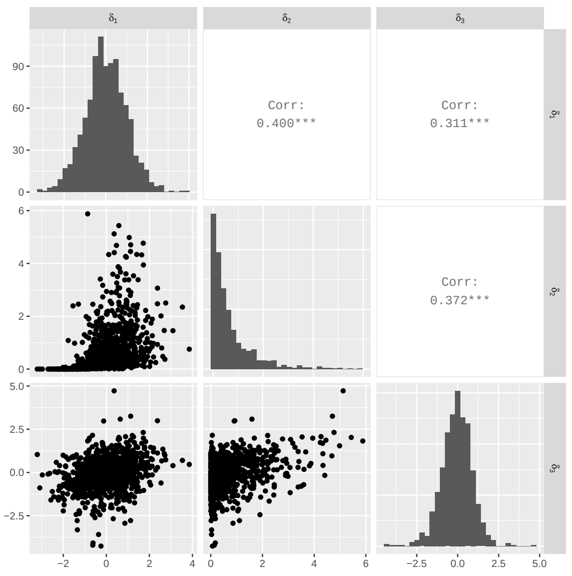


FIGURE 5. Scatterplots and histograms for a $n = 1000$ sample drawn from a three-dimensional VITA distribution.

simulated intercept and slope values, each drawn from standard normal distributions, to obtain simulated observations on x_1, x_2 and x_3. The researcher replicates 1000 samples of size $n = 1000$ from both the fully normal distribution and the distribution with residual errors stemming from VITA.[1] The growth model was estimated with seven free parameters: Three correlated residual errors; the residual error variances, which were constrained to be the same at each of the three measurement occasions; the correlation ρ; and the means of the intercept and slope variables. The model has one DF. As a measure of inference quality for ρ the researcher decides to calculate the confidence interval coverage rate, at the 95% confidence level, for $\rho = 0$, using classical standard errors that assume exact normality. Under full normality, the coverage rate of 0.94 was close to nominal. With non-normal error vector, the coverage rate was 0.905. Hence, the researcher found some support for the claim that non-normality in the error vector may affect the quality of intercept-slope correlation NTML inference.

3.1. **The independent generator approach.** The `covsim` package exports, in addition to `vita`, the function `rIG`. This simulation function is not based on a copula perspective and does not allow for full specification of the univariate marginal distributions. Instead it is closer in approach to the method of Vale and Maurelli (Vale & Maurelli, 1983), where only univariate skewness and kurtosis is prespecified. However, the independent generator (IG) algorithm (Foldnes & Olsson, 2016) is more flexible than the Vale and Maurelli method, defining a larger class of non-normal distributions for each set of skewness and kurtosis values. Although the

[1]Simulation code is provided in the online supplementary material.

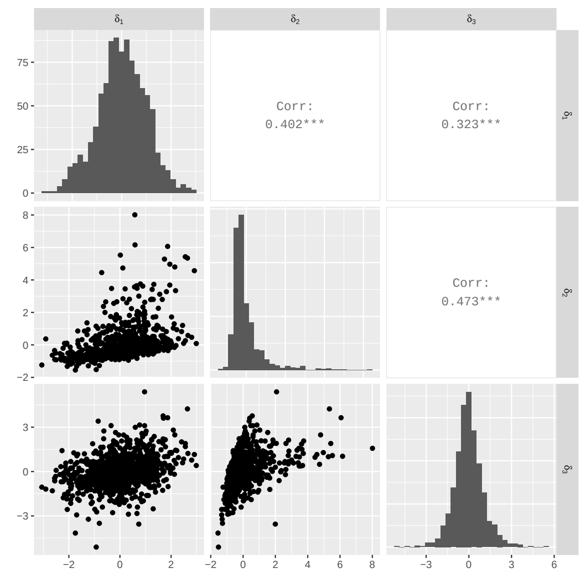


FIGURE 6. Scatterplots and histograms for a $n = 1000$ sample drawn from a three-dimensional IG distribution.

main focus of the present manuscript is the flexible use of bivariate copulas in simulating non-normal data with given marginals and covariance matrix, we here for completeness give a short introduction to the IG algorithm.

The IG transform represents the non-normal vector ξ stochastically as

$$\xi = AX$$

where A is a square matrix and X a vector consisting of mutually independent generator variables with unit variance. The user specifies desired skewness and kurtosis values in ξ, and the IG algorithm numerically determines the skewness and kurtosis in each generator variable to match the desired values. The matrix A is a square root of the specified covariance matrix Σ. In `rIG` the user may specify a triangular square or a symmetrical square root matrix, which gives two different distributions. Also the marginal distributions for X may be freely chosen, further expanding the distributional class defined by IG. In its current implementation, `rIG` uses the Pearson family of distributions (Pearson, 1895). Let us reconsider the marginal distributions used above. We note that the chi-square distribution with one DF and the Student's t distribution with five DFs have skewness $\sqrt{8}$ and 0, respectively, and kurtosis 12 and 6, respectively. In the following code we ask for an IG distribution that matches the first four moments of the three marginal distributions considered in the previous section. The scatterplot is given in Figure 6.

```
R> set.seed(1)
R> ig.sample <- rIG(N = 10^3, sigma.target = sigma.target, reps = 1,
+    skewness = c(0, sqrt(8), 0), excesskurtosis = c(0, 12, 6))
```

4. A six-dimensional growth curve illustration

In this section we use the flexibility of VITA to further study the effect of non-normality in growth curve residual error vectors on normal-theory based inference. We focus on the chi-square statistic of model fit. We consider a linear growth curve with scores across six time-points. We assume that the errors δ_i, $i = 1, \ldots, 6$, have unit variance, and that they are autocorrelated according to the following banded structure (i.e., a Toeplitz structure):

$$\Sigma_\delta = \begin{pmatrix} 1 & & & & & \\ 0.5 & 1 & & & & \\ 0.2 & 0.5 & 1 & & & \\ 0 & 0.2 & 0.5 & 1 & & \\ 0 & 0 & 0.2 & 0.5 & 1 & \\ 0 & 0 & 0 & 0.2 & 0.5 & 1 \end{pmatrix}$$

We calibrate the following three VITA distributions for the δ vector:

- **VITA1:** $\delta_1, \delta_2, \delta_3, \delta_4, \delta_5$, and δ_6 are standard normal, and all 18 bivariate copulas are of Clayton type
- **VITA2:** δ_1 is standard normal, while $\delta_2, \delta_3, \delta_4, \delta_5$, and δ_6 are chi-square distributed with $5, 4, 3, 2$, and 1 degrees of freedom, respectively. The chi-square distributions are scaled to have unit variance. All 18 bivariate copulas are normal.
- **VITA3:** δ_1 is standard normal, while $\delta_2, \delta_3, \delta_4, \delta_5$, and δ_6 are chi-square distributed with $5, 4, 3, 2$, and 1 degrees of freedom, respectively. The chi-square distributions are scaled to have unit variance. All 18 bivariate copulas are of type Clayton.

Using the `vita` function in package covsim the code is as follows:

```
R> residual.covariance <- toeplitz(1:6)
R> residual.covariance[residual.covariance > 3] <- 0
R> residual.covariance[residual.covariance == 2] <- 0.5
R> residual.covariance[residual.covariance == 3] <- 0.2
R> margins.nonnorm <- list(list(distr = "norm"),
+    list(distr = "chisq", df = 5), list(distr = "chisq", df = 4),
+    list(distr = "chisq", df = 3), list(distr = "chisq", df = 2),
+    list(distr = "chisq", df = 1))
R> margins.norm <- list(list(distr = "norm"), list(distr = "norm"),
+    list(distr = "norm"), list(distr = "norm"),
+    list(distr = "norm"), list(distr = "norm"))
R> margin.variances <- c(1, 10, 8, 6, 4, 2)
R> sigma.target <- diag(sqrt(margin.variances)) %*% residual.covariance %*%
+    diag(sqrt(margin.variances))
R> set.seed(1)
R> vita1 <- vita(margins.norm, residual.covariance, family_set = "clayton")
R> set.seed(1)
R> vita2 <- vita(margins.nonnorm, sigma.target, family_set = "gauss")
R> set.seed(1)
R> vita3 <- vita(margins.nonnorm, sigma.target, family_set = "clayton")
```

Data generation first simulates independent random draws from the standard normal for the intercept and slope variables, and then adds the residual errors simulated from VITA distributions. A growth model with 15 degrees of freedom, which correctly specifies the structure for Σ_δ, is fitted to the data. Our research question is to what extent non-normality in the errors affects the sampling distribution, and in particular, the Type I error control of the regular normal-theory chi-square statistic T_{NTML}. Since VITA1, VITA2, and VITA3 are different distributions, with different mixes of marginal and copula non-normality, there might also be insights to draw from their differential effect on the chi-square test. One way of conducting this research is to use conventional small-sample simulations and to calculate rejection rates over many replications. Here we choose a different approach. We calculate the exact asymptotic distribution of T_{NTML} in each distributional condition. This will also give us asymptotic Type I error rates.[2] First, we simulate a very large $n = 10^6$ sample from each of the VITA distributions. Then the model is fitted to each of the three datasets, and we extract the eigenvalues of the matrix $U\Gamma$ (see, e.g., Foldnes & Grønneberg (2017a) for further details). Theory (Box, 1954) dictates that T_{NTML} is asymptotically distributed as the weighted sum of independent chi-square distributions, each with one degree of freedom, where the weights are the eigenvalues of $U\Gamma$. This allows us to calculate the density of T_{NTML} under the three distributional conditions. Under multivariate normality, this density is that of the nominal chi-square distribution with 15 degrees of freedom, which is used to calculate asymptotic Type I error control of T_{NTML}. Figure 7 depicts the asymptotic sampling distribution of T_{NTML} under four conditions, namely multivariate normality, and the distributions involving the three VITA error distributions. It is seen that T_{NTML} becomes inflated as we move from multivariate normality, and as we progress through the three VITA error distributions. The vertical line represents the critical value when referring T_{NTML} to its critical value at the $\alpha = 0.05$ level of significance. VITA1 has standard normal marginals and a non-normal copula. In this condition the asymptotic type I error control is 7.3%, quite close to the nominal level. VITA2 has four non-normal marginals, and a normal copula, and affects T_{NTML} to a larger extent than VITA1. The asymptotic rejection rate under VITA2 is 29.4%, which is far above the nominal 5% level. VITA3 introduces more non-normality compared to VITA2, by having a non-normal copula. The effect on T_{NTML} is critical, whose asymptotic rejection rate is 50.6% under VITA3 errors. In sum we see that non-normality in the residual error vector may markedly inflate the rejection rates of T_{NTML}, but we may speculate that the effect is mild as long as the univariate marginals are normally distributed.

5. The implementation of VITA in covsim

In this section we briefly explain how the function `vita` implements the VITA algorithm. Grønneberg & Foldnes (2017) provided as supplementary material a VITA implementation using package VineCopula (Schepsmeier, Stoeber, Brechmann, Graeler, Nagler & Erhardt, 2018) for constructing and simulating from regular vines. This package is no longer in active development, and package rvinecopulib Nagler & Vatter (2019) was instead used in `vita`. The most important benefits of rvinecopulib relative to VineCopula for our purposes is a sleeker and more modern

[2]In the online supplementary material are given code for conventional small-sample simulations that confirms our upcoming findings.

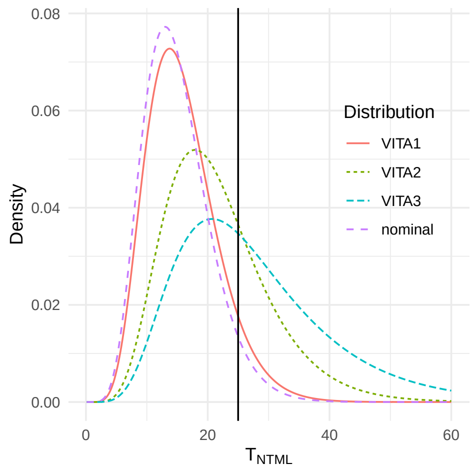


FIGURE 7. The asymptotic density of T_{NTML} under four conditions. nominal=Chi-square distribution with 15 degrees of freedom. VITA1, VITA2, and VITA3 denote three kinds of non-normal residual distributions. Vertical line represents critical value at $\alpha = 0.05$.

API and shorter simulation runtimes. In experiments (see supplementary material) with a five-dimensional vine, on a computer with 4 CPU cores, simulation runtimes at a sample size of $n = 1000$ was shorter with rvinecopulib compared to VineCopula by a factor of four. Also, as explained below, the initial calibration of VITA parameters involves a series of large-sample random draws from regular vines, which means that VITA calibration is computationally demanding. The root-finding routine provided by Grønneberg & Foldnes (2017) has been improved in `vita`, by splitting it into a high-speed routine which identifies an interval for the root, followed by high-precision root-finding in this interval, based on curve-fitting. This two-stage root-finding routine is faster than the basic method in Grønneberg & Foldnes (2017). Combined with faster simulation times in package rvinecopulib, the calibration time for a five-dimensional vine using `vita` instead of the original code provided by Grønneberg & Foldnes (2017) was reduced by a factor of 13.

The main arguments to `vita` are

- margins. A list that specifies the univariate marginal distributions.
- sigma.target. The target covariance matrix.
- vc. A vine copula structure in the format defined by package rvinecopulib. That is, a specification of a hierarchy of $p-1$ trees, and, for each tree node, a bivariate copula family. If not provided by the user, `vita` will initialize vc as follows. The vine structure of vc is specified as the simplest regular vine, namely the D-vine on p dimensions. See Figures 4 and 10 for the D-vine with $p = 3$ and $p = 4$, respectively. In addition, the bivariate copula family in each node in the D-vine will be taken as the first element of the argument family_set.
- family_set. A vector that specifies which bivariate copula families are to be calibrated. If vc is provided by the user, and the algorithm can not identify

a feasible solution for the family dictated by vc, the algorithm instead tries to calibrate the dependence parameter for the first family in family_set. If not successful, an attempt is made to calibrate the parameter in the second family, and so forth. If vc is not provided, the algorithm attempts first to calibrate the dependence parameter in the first member of family_set, and if not succesful, the second member, and so forth.

The above arguments specify a class of VITA distributions, parameterized by the $p(p-1)/2$ dependence parameters in the bivariate copulas. The task of `vita` is to numerically determine the values of these dependence parameters so that the resulting VITA distribution has the required covariance matrix given by sigma.target. That is, `vita` searches for a dependence parameter value θ in each of the copulas, so that the covariance matrix of the resulting full distribution is sigma.target, up to numerical precision. This search can be done in the same order as when simulating from a vine, building our way up the tree, connecting more and more distributions with pairwise conditional distributions. As shown in Grønneberg & Foldnes (2017), the correlation of each pair of variables is typically a strictly increasing function of θ for most single parameter copulas, making the numerical search well behaved. Unfortunately, there is no simple formula for the correlation matrix of a vine. Worse still, no simple formula can be derived for pairwise bivariate distributions connected at higher levels of the vine tree. In the implementation of VITA in `vita`, we resort to Monte Carlo simulation to approximate the required correlation.

VITA calibrates each pairwise bivariate distribution and combines them to form the full vine distribution in a specific order. A formal algorithmic description of the methodology is given in Grønneberg & Foldnes (2017). We here informally summarize the main steps of the method, and later describe in technical detail the new implementation of the root finding procedure that underlies the calibration.

As explained in more detail in the appendix, each pair (i, j), where $1 \leq i, j \leq p$, is connected once in the vine. This is also the case in the covariance matrix. Let (σ_{ij}) be the target covariance matrix in sigma.target, and let the parameter of the bivariate copula connecting the (i, j) distribution be parametrized by θ_{ij}. The first pairs of bivariate distributions that are calibrated are those connected at the lowest level of the vine tree. For illustration, we consider the vine given in Figure 4 (p.10). We see that $(1, 2)$ are connected at the lowest tree. We may therefore simulate directly from this bivariate distribution. This distribution depends on a parameter θ_{12}. We may choose this parameter in such a way that the covariance of the resulting bivariate distribution matches the required covariance given in σ_{12}. This matching is non-trivial and is described in technical detail below. A similar matching may be done for all other marginals connected at the lowest level of the vine, which here is only $(2, 3)$. These calibrations are done independently of each other. Now the vine in Figure 4 have only two levels, and there is only one bivariate margin left to be matched, namely $(1, 3)$, which is connected at the topmost level. The distribution of $(1, 3)$ is derivable from the full vine structure, and the conditional copula of $(1, 3)$ given 2 is parametrized using a bivariate copula a parameter θ_{13}. To simulate realizations from $(1, 3)$ we need to know the distributions at the lower level of the tree, as well as specifying the value of θ_{13}. The distributions of the lowest level of the tree have already been fixed, and we may, for varying values of θ_{13}, simulate the full three-dimensional vine distribution, compute the covariance of $(1, 3)$, and select the θ_{13} value that yields a covariance equal to σ_{13}. We have then calibrated the full three-dimensional vine.

For higher dimensions, this idea has to be iterated several times to identify all parameters in an order that enable us to always simulate from the required bivariate variables in the vine. In order to briefly illustrate how to calibrate a higher dimensional vine, consider the four-dimensional vine in Figure 10 in the appendix (p.34). Comparing the vine in Figures 4 and 10, we see that the three dimensional vine in Figure 4 is included within the structure of the vine of Figure 10 as a subset of its connections. That is, the vine in Figure 4 is a sub-vine of the vine of Figure 10 that comprise the variables $(1, 2, 3)$: The vines are equal, with the exception that the four-dimensional vine also has to connect marginal 4 with the remaining variables, which is done using the additional structure given in Figure 10. But to simulate only $(1, 2, 3)$ from the four-dimensional vine in Figure 10, we only need to know the three-dimensional vine in Figure 4.

To calibrate the four-dimensional vine in Figure 10, we may therefore continue where we left off when calibrating the three-dimensional vine in Figure 4. After calibrating the new bivariate distribution connected at the lowest level, namely $(3, 4)$, the next step is to calibrate all distributions at the second level. Only one such distribution is left, namely $(2, 4)$. By the same reasoning as earlier, we may simulate from the sub-vine that enables the simulation of $(2, 3, 4)$: The parameters of the $(2, 3)$ and $(3, 4)$ distributions have already been fixed. Therefore, the sub-vine expressing the full distribution of $(2, 3, 4)$ only have one free parameter, namely θ_{24}, which may be varied to get a covariance between $(2, 4)$ to match up with σ_{24}. As in the first level of the tree, the matching of the parameters at the second level of the tree are done independently of each other, and can be done in any order. However, to calibrate all connections at a given level, all connections at lower levels have already to be calibrated from before.

The final matching required for the four-dimensional vine is then to work with the single distribution connected at the third level of the vine, namely the $(1, 4)$ distribution, so that the $(1, 4)$ marginal has covariance equal to σ_{14}. All parameters of the vine are now fixed with the exception of θ_{14}, and this parameter may be varied until the required covariance is induced.

The calibration order of variable pairs in `vita` is as follows: All copulas at the lowest level are calibrated, then the next level, and so on up the the highest level. As mentioned in Grønneberg & Foldnes (2017), other orders are possible, as exemplified above, but the order is immaterial as long as unique solutions for reaching the desired covariances exist.

In each calibration step, numerical integration done via Monte Carlo simulation and a search for the solution of an equation must be performed. We now detail how this is done in the implementation of `vita`. Let (U_i, U_j) be distributed according to the copula of the sub-vine required to simulate the (i, j) distribution as described above. Due to the order we have traversed the vine, there is always only one free parameter θ_{ij} of this distribution that is free, and is used to match up the required covariance σ_{ij}. In the following description we omit the i, j subscript from θ, to reduce the notational burden. As explained in more detail in the appendix, we apply the corresponding inverse quantile functions according to the entries in margins to calculate the covariance induced by a given θ: Let us denote by $\sigma_{ij}(\theta)$ the covariance between the resulting variables $F_i^{-1}(U_i)$ and $F_j^{-1}(U_j)$. Our aim is now to determine θ so that $\sigma_{ij}(\theta) = \sigma_{ij}$. Unfortunately there are no analytical expressions available for $\sigma_{ij}(\theta)$ except in very special cases, but the covariance may be approximated by simulating a large sample of size n of (U_i, U_j), applying the quantile functions F_i^{-1} and F_j^{-1} to each simulated variable, and calculating the resulting sample covariance, which we denote by $\hat{\sigma}_{ij}(\theta)$. Then $\hat{\sigma}_{ij}(\theta)$ will converge in probability to $\sigma_{ij}(\theta)$ as n increases. However, with large n, simulating these samples is time-consuming, so `vita` is implemented in two stages.

(1) Initial high-speed calibration. In this stage we use the modest sample size $n = 1500$ to determine $\hat{\sigma}_{ij}(\theta)$, using function `uniroot` in the stats package. That is, we approximate θ by finding the root $\hat{\theta}_n$ of the discrepancy function $\hat{\sigma}_{ij}(\theta) - \sigma_{ij}$. We expect $\hat{\theta}_n$ to be quite close to θ, but it contains random error, so we repeat this procedure and approximate θ a small number of times (the argument numrootpoints). This results in a number of root candidates $\hat{\theta}_n^1, \hat{\theta}_n^2, \ldots, \hat{\theta}_n^{\text{numrootpoints}}$, which are independent and identically distributed random variables. A standard t-based confidence interval for the dependence parameter θ is then constructed from these approximate roots, using a high level of confidence (as specified by the argument conflevel).

(2) Final high-precision calibration. In this stage, we evaluate the discrepancy $\hat{\sigma}_{ij}(\theta) - \sigma_{ij}$ to a high precision at a small number (as specified by argument numpoints) of equally spaced points across the confidence interval determined in the first stage. The approximation is done by simulating from a very large sample (n is equal to the argument Nmax) in each of these points. We then fit a second degree polynomial to the discrepancy values and use `uniroot` to locate the root of this polynomial, which yields our final estimate for θ.

If, for any pair of variables, the calibration does not find a solution θ, the algorithm changes the bivariate family to the next entry in family_set. If no solution is found, `vita` terminates with an error message. This means that there is no VITA distribution with the given marginals, vine structure and bivariate families that can attain sigma.target. To proceed, the user could then rerun `vita` with, e.g., a different vine structure.

As mentioned in the introduction, traditional approaches to non-normal covariance modeling only specifies the lower-order univariate moments, and do not offer any control of the multivariate aspects of the simulated vector, with the exception of covariance matching. As we have demonstrated, the VITA approach is more flexible. However, the cost of increased flexibility is increased computing time necessary to calibrate the VITA distribution. The default values for arguments Nmax and numpoints in `vita` guarantees a highly precise VITA calibration. That is, the calibrated VITA distribution will have a covariance matrix almost numerically indistinguishable from sigma.target. In higher dimensions, this precision comes at the cost of long calibration running times. Table 1 gives calibration times on a computer (2.3 GHz 8-Core Intel Core i9) using the default options in `vita`, with target correlation among all variable pairs equal to $\rho = 0.3$, for increasing dimensionality. It is seen that approaching 20 dimensions, calibration time exceeds one hour, while simulating 1000 samples, each of size $n = 1000$, requires less than a minute. So the calibration step, which is only executed once, is time-consuming, while repeated simulation from the calibrated VITA is relatively fast. Foldnes &

Grønneberg (2021) calibrated and simulated from VITA distributions in twenty dimensions in an extensive simulation design. However, given that the median number of observed variables in empirical SEM studies is close to 20 (Li, 2016), using `vita` for larger models, with say 50 dimensions, will entail days of calibration time with the default options. In such cases the user may lower the argument `Nmax` from 10^6 to 10^5, thereby reducing calibration time by a factor of 10. For instance, for dimension 40 calibration was achieved in 4.5 hours using option `Nmax`$=10^5$. Even with reduced precision, the calibrated VITA distribution has a covariance matrix almost equal to the target covariance. Among the 780 pair-wise correlations estimated in a $n = 10^6$ sample drawn from the 40 dimensional calibrated VITA distribution with `Nmax`$=10^5$, 748 were within a 0.005 distance of the target $\rho = 0.3$, and all were within a 0.01 of $\rho = 0.3$.

If high precision is important in an application, a formal test of equality of covariance matrices should be performed. This may be done by computing as test statistic the quadratic form of the discrepancies between the sample covariances and the target covariances, weighted by the inverse of estimated asymptotic covariance matrix of the covariances (Mair et al., 2012).

Dimension	5	10	15	20	25	30
Calibration (hrs)	0.006	0.065	0.401	1.447	3.675	8.100
Simulation (hrs)	0.001	0.004	0.009	0.015	0.024	0.034

Table 1. Calibration times in hours under the default `Nmax`$=10^6$. Simulation of 1000 samples, each of size $n = 10^3$.

To precisely (`Nmax`$=10^6$) calibrate VITA distributions with 50 or more dimensions, our current implementation will demand unrealistically long running times. The bottleneck of the calibration algorithm consists of simulating a large sample (`Nmax`) from a regular vine. This simulated sample is then used to compute a single covariance. If we distribute the large-sample simulation to several computers, the desired covariance of all the simulated realizations across computers can be computed based on sums, cross-products and sums of squares from each computer. Hence, the VITA algorithm may conveniently be distributed across a network of computers. Such functionality may be included in future versions of `covsim`.

6. Further examples

We here consider some further applications of VITA. In Section 6.1, we consider a 20 dimensional SEM example with continuous data. In Section 6.2 we discuss simulation of ordinal SEMs and show how this can be done with VITA.

6.1. **Using VITA to simulate continuous data for SEM.**

In most SEM simulation studies the methodologist first specifies a SEM model together with its population parameter values. Then the study is conducted by drawing random samples from a distribution whose covariance matrix equals the model-implied covariance matrix. As an example, consider the SEM whose structural part is depicted in Figure 8. The model has five factors, and is representative of medium-sized SEM in applied studies (Li, 2016). Each factor has four indicators, yielding a total of 20 dimensions. The factor loadings for the indicators were set to $0.8, 0.7, 0.6$ and 0.5 within each factor, and the corresponding residual variances were set to 0.5. The correlation

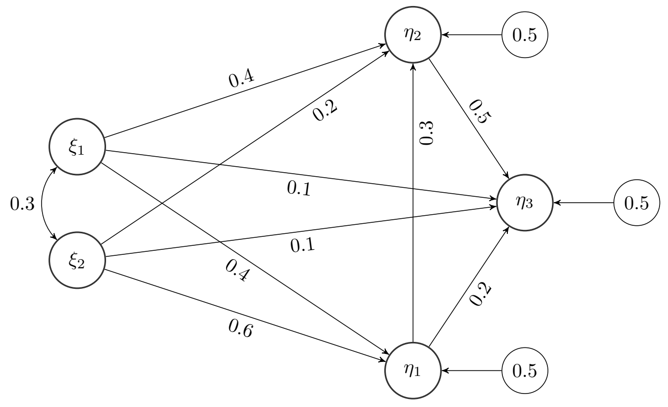


FIGURE 8. Structural model for a medium-sized SEM. Indicator variables not depicted.

was set to 0.3 between the two exogeneous factors, each of which had unit variance. The residual variances for the endogeneous factors were also set to 0.5. Using the package lavaan we can compute the target covariance matrix implied by these population parameters as follows.

```
R> sem.pop <- '
Ksi1 =~ start(.8) * x1 + start(.7) * x2 + start(.6) * x3 + start(.5) * x4
Ksi2 =~ start(.8) * x5 + start(.7) * x6 + start(.6) * x7 + start(.5) * x8
Eta1 =~ start(.8) * y1 + start(.7) * y2 + start(.6) * y3 + start(.5) * y4
Eta2 =~ start(.8) * y5 + start(.7) * y6 + start(.6) * y7 + start(.5) * y8
Eta3 =~ start(.8) * y9 + start(.7) * y10 + start(.6) * y11 + start(.5) * y12
Eta1 ~ start(.4) * Ksi1 + start(.6) * Ksi2
Eta2 ~ start(.4) * Ksi1 + start(.2) * Ksi2 + start(.3) * Eta1
Eta3 ~ start(.1) * Ksi1 + start(.1) * Ksi2 + start(.2) * Eta1 + start(.5) * Eta2
Ksi1 ~~ start(.3) * Ksi2; Eta1 ~~ start(.5) * Eta1; Eta2 ~~ start(.5) * Eta2
Eta3 ~~ start(.5) * Eta3; x1 ~~ start(.5) * x1; x2 ~~ start(.5) * x2
x3 ~~ start(.5) * x3; x4 ~~ start(.5) * x4; x5 ~~ start(.5) * x5
x6 ~~ start(.5) * x6; x7 ~~ start(.5) * x7; x8 ~~ start(.5) * x8
y1 ~~ start(.5) * y1; y2 ~~ start(.5) * y2; y3 ~~ start(.5) * y3
y4 ~~ start(.5) * y4; y5 ~~ start(.5) * y5; y6 ~~ start(.5) * y6
y7 ~~ start(.5) * y7; y8 ~~ start(.5) * y8; y9 ~~ start(.5) * y9
y10 ~~ start(.5) * y10; y11 ~~ start(.5) * y11; y12 ~~ start(.5) * y12'
R> sigma.target <- lavInspect(sem(sem.pop, data = NULL), "sigma.hat")
```

Next, we fit a VITA distribution with normal marginals to the target covariance matrix. This is a variant of a data generating distribution used in the simulation study of Foldnes & Grønneberg (2021). First, the margins are scaled to match the target variances. Then, we calibrate a VITA distribution. Note that we do not specify which family of copulae to use, so the default Clayton copula is used. Finally, a list of 1000 samples, each of sample size 1000, is drawn from the calibrated vita distribution.

```
R> marginsnorm <- lapply(X = sqrt(diag(sigma.target)),
+    function(X) list(distr = "norm", sd = sqrt(X)))
```

```
R> vitadist <- vita(marginsnorm, sigma.target)
R> randomsamples <- replicate(10^3, rvine(10^3, vitadist))
```

As discussed previously, the calibration step is time-consuming in higher dimensions. Here, with 20 variables, the calibration step required 1.8 hours (again using a 2.3 GHz 8-Core Intel Core i9 CPU). This step is only performed once. When completed, random samples can be drawn at a relatively fast rate. Producing 1000 samples each of size 1000 took one minute to complete. Finally, we note that the calibration step may be performed faster by specifying option Nmax=10^5 when calling `vita`, at the expense of reduced precision in covariance matching.

6.2. **Using VITA to simulate ordinal-categorical data for SEM.** A major approach for SEM with ordinal data is to impose a threshold model to the data, which postulates that the categorical data arise from discretization of an underlying continuous vector which is multivariate normally distributed. Many influential simulation studies (e.g., Flora & Curran, 2004; Li, 2016; Quiroga, 1994; Rhemtulla, Brosseau-Liard & Savalei, 2012) have investigated the robustness of ordinal SEM against violation of non-normality, using the Vale Maurelli approach. However, Grønneberg & Foldnes (2019a) showed that the Vale Maurelli approach is not suitable for ordinal data simulation in the context of covariance modeling. We here briefly show how to simulate ordinal bivariate data by discretizing bivariate VITA distributions. For an observed ordinal variable, there is no way to identify which underlying univariate distribution that produced the data, since the thresholds may be transformed to accommodate all continuous univariate distributions.

As argued in Foldnes & Grønneberg (2019a,2), it is advantageous to keep the marginals fixed during simulation. Since VITA offers exact control of marginals, it is uniquely suited for simulation studies with ordinal data for SEM. We will here set the marginals to standard normal. When simulating fully normal data, both the marginals and the copula are normal. We will let the copula be non-normal, but the marginals will be normal.

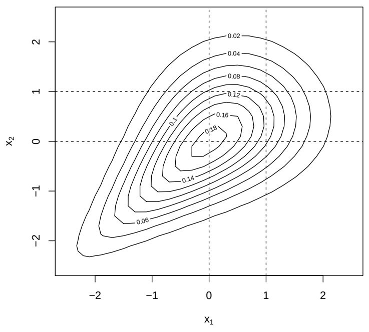
For illustration, we assume that the underlying correlation in a continuous bivariate distribution with standard normal marginals is $\rho = 0.5$, and we discretize into three categories using thresholds $\tau_1 = 0$ and $\tau_2 = 1$. This means that we consider simulated data of the form

$$X_i = \begin{cases} 1, & \text{if } \xi_i \le \tau_1 \\ 2, & \text{if } \tau_1 < \xi_i \le \tau_2 \\ 3, & \text{if } \xi_i > \tau_2 \end{cases} = \begin{cases} 1, & \text{if } \xi_i \le 0 \\ 2, & \text{if } 0 < \xi_i \le 1 \\ 3, & \text{if } \xi_i > 1 \end{cases}$$

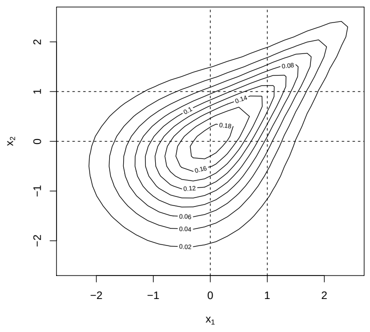
for $i = 1, 2$, where (ξ_1, ξ_2) is a continuous random vector simulated using VITA. Both ordinal variables have proportions $0.5, 0.34$, and 0.16. We inquire whether the polychoric correlation estimator used in ordinal SEM becomes biased when we replace the bivariate normal with a Clayton or a Joe copula. So first, we determine parameters for the latter two copulas such that, when marginals are standard normal, the Pearson correlation is 0.5.

```
R> sigma.target <- matrix(c(1, 0.5, 0.5, 1), 2)
R> set.seed(1)
R> vita_clayton <- vita(list(list(distr = "norm"), list(distr = "norm")),
+    sigma.target, family_set = "clayton")
R> set.seed(1)
R> vita_joe <- vita(list(list(distr = "norm"), list(distr = "norm")),
+    sigma.target, family_set = "joe")
R> clayton.disc <- apply(rvine(10^3, vita_clayton), 2, cut,
+    breaks = c(-Inf, 0, 1, Inf), labels = FALSE)
```

Contour plots of these calibrated copulas are given in Figure 9.



(A) Clayton copula.



(B) Joe copula.

FIGURE 9. VITA distributions with standard normal marginals. The dashed lines represent thresholds $\tau_1 = 0$ and $\tau_2 = 1$ used for discretization.

After discretizing the distribution with standard normal marginals and a Clayton copula shown in Figure 9a and the distribution with standard normal marginals and a Joe copula in Figure 9b, the resulting population contingency tables are

	1	2	3
1	0.334	0.123	0.043
2	0.123	0.146	0.072
3	0.043	0.072	0.044

and

	1	2	3
1	0.333	0.146	0.021
2	0.146	0.149	0.047
3	0.021	0.047	0.091

respectively. It is seen that under the Clayton and Joe copula the probabilities that both ordinal variables take their maximum value are 4.4% and 9.1%, respectively. Such discrepancies in the bivariate ordinal distribution affects the normal-theory based polychoric estimator: The population value of the polychoric correlation under the Clayton and Joe copula is 0.42 and 0.60, respectively. Given that the true underlying Pearson correlation for both distributions in Figure 9 is 0.5, this shows that the polychoric correlation may become strongly (downwards or upwards) biased when underlying normality is violated. In this illustration, the lower tail dependency in the Clayton copula, combined with the chosen thresholds, results in strong downward bias. And the upper tail dependency in the Joe copula leads to strong upward bias.

The sensitivity of the polychoric correlation to underlying non-normality poses a threat to the popular practice of conducting SEM with ordinal data based on the polychoric correlation matrix. Even though a proposed SEM model fits well to the underlying data, it might fit poorly to the polychoric correlation matrix, since the latter might be biased due to non-normality in the underlying data. The result might be biased estimates and inflated Type I error rates, and it is therefore

Underlying distribution	Normal	Clayton	Joe
$n = 100$	0.068	0.178	0.210
$n = 500$	0.056	0.608	0.838

Table 2. Rejection rates of bootstrap test for underlying normality.

important to assess whether the ordinal dataset is compatible with the underlying normality assumption. Foldnes & Grønneberg (2019b) proposed a bootstrap test for this purpose, which is implemented in the R package discnorm (Foldnes & Grønneberg, 2020b). It is used as follows.

```
library("discnorm")
bootTest(clayton.disc)
```

We conducted a small simulation study on the Type I error control and power of the bootstrap test in the context of the present bivariate illustration. We generated 500 samples of size $n = 100$ and of size $n = 500$ and collected the rejection rate of the bootstrap test for ordinal data stemming from discretization of a normal distribution, the Clayton VITA, and the Joe VITA, using the same set of thresholds as depicted in Figure 9. The rejection rates are given in Table 2. The bootstrap test maintains Type I error rates well, but has low power to detect the underlying normality in both the Clayton and Joe VITA distributions at the smallest sample size. Expectedly, at the larger sample size $n = 500$ the power to detect the underlying non-normality is higher. The non-normality in the Joe VITA is more detectable than the non-normality in the Clayton VITA at both sample sizes.

7. Conclusion

The VITA approach, implemented in the R package covsim, is very flexible, since it accommodates user-specified marginal distributions and also offers a great deal of control over the bivariate dependencies in the simulated vector. This control is in contrast to more established methodology, like the Vale & Maurelli (1983) method. In most cases, Vale Maurelli has an exactly normal copula (Foldnes & Grønneberg, 2015), and does not allow the specification of the resulting distribution except the covariance matrix, skewness and kurtosis. The increased flexibility of VITA, however, comes at a cost. VITA, being based on the statistical concept of a regular vine, is a more complex construction than the Vale Maurelli transform. Also, VITA calibration is more computationally demanding than is the case for the Vale Maurelli transform. In the appendix we have given an introduction to the statistical machinery underlying vines and VITA, such as bivariate copulas and their use in constructing regular vines. We also give an introduction to how VITA simulation is performed on a computer. Numerical illustrations of applying VITA to simulate non-normal residual error structures in growth curve modeling were presented, demonstrating the effects of different kinds of non-normality on inference. Also, we have illustrated how VITA may be used to simulate continuous non-normal data from a SEM, and to simulate ordinal data in a way that properly violates the underlying normality assumption. For ordinal data, we illustrated a new bootstrap test, implemented in the R package discnorm, for the central assumption of underlying normality.

Appendix A. Technical appendix: How simulation is done on a computer

A.1. **How univariate and bivariate simulations are performed on a computer.** Throughout the paper, we assume that the marginals of the distribution we simulate from are continuous. In SEM, this is mostly without loss of generality, as ordinal variables are usually modelled as discretizations of a continuous random vector, see Section 6.2. Note that this also applies to a large class of IRT models, see, e.g., Foldnes & Grønneberg (2019a); Takane & De Leeuw (1987).

A.1.1. *Univariate simulation.* We first review a standard method to simulate from a continuous univariate distribution with cumulative distribution function F_1. This is covered in most standard statistics text books, see, e.g., Rice (2006, Proposition D, Section 2.3).

We assume further that F_1 is strictly increasing, and therefore has an inverse F_1^{-1}. Since F_1 is a continuous distribution function, it will be continuous and increasing, but not necessarily strictly increasing (i.e., there may be flat regions), necessitating the use of more complex arguments, such as those in Chapter 1 of Shorack & Wellner (2009). We will throughout the paper pedagogically assume such strict monotonicity, and thereby avoiding such complex arguments.

Recall that the inverse function F_1^{-1} is defined as the solution to the equation $F_1(x) = u$ with respect to x, and where $0 < u < 1$. Clearly, this solution depends on u, and we therefore denote the solution as a function of u, that is, $F_1^{-1}(u)$. Since $F_1^{-1}(u) = x$ where $F_1(x) = u$ we get that $F(F^{-1}(u)) = F(x) = u$. We may compute $F_1^{-1}(u)$ by solving

$$F_1(x) - u = 0 \tag{2}$$

for u. Since F_1 is increasing and continuous, with $F_1(-\infty) = 0$ and $F_1(\infty) = 1$, eq. (2) has a single solution, which can be found either analytically, or using any standard root finding procedure.

Now let U be a univariate random variable with the uniform distribution on $[0, 1]$, denoted by $U \sim U[0, 1]$. This means that $P(U \leq x) = xI\{0 \leq x \leq 1\} + I\{x > 1\}$ where $I\{A\}$ is the indicator function of A which is 1 is A is true, and zero otherwise. We may then let

$$X = F_1^{-1}(U).$$

The distribution of X is F_1. To see this, we start with $P(X \leq x) = P(F_1^{-1}(U) \leq x)$. Applying F_1 on both sides of the inequality is allowed since F_1 is increasing. Since $F(F_1^{-1}(U)) = U$, this gives

$$\begin{aligned} P(F_1^{-1}(U) \leq x) &= P(F_1(F_1^{-1}(U)) \leq F_1(x)) = P(U \leq F_1(x)) \\ &= F_1(x)I\{0 \leq F_1(x) \leq 1\} + I\{F_1(x) > 1\}. \end{aligned}$$

Since F_1 is a cumulative distribution function, we have $0 \leq F_1(x) \leq 1$, and therefore the first indicator function is always one, while the second is always zero. Therefore, we conclude that $P(X \leq x) = F_1(x)$ as required.

A.1.2. *Imposing required marginals on copula-distributions.* Let us now consider the more complex bivariate case, which is not as well-known as the univariate case. Firstly, let us assume that we are able to simulate from a copula C. That is, suppose

$(U_1, U_2) \sim C$, meaning

$$P(U_1 \le u_1, U_2 \le u_2) = C(u_1, u_2). \tag{3}$$

Recall that a copula has uniform marginals. This means that $U_1 \sim U[0,1]$ and $U_2 \sim U[0,1]$. Therefore, following the argument above, we may let $X_1 = F_1^{-1}(U_1)$ and $X_2 = F_2^{-1}(U_2)$, and see that the marginal distribution of X_1 is F_1 and the marginal distribution of X_2 is F_2. This means $P(X_i \le x_i) = F_i(x_i)$ for $i = 1, 2$, and does not say anything about the dependence between X_1 and X_2. We now show that (X_1, X_2) has the distribution as in eq. (1), i.e., $P(X_1 \le x_1, X_2 \le X_2) = C(F_1(x_1), F_2(x_2))$. Using the definition of X_1, X_2 and applying the functions F_1 and F_2 respectively on the two inequalities, we have

$$\begin{aligned} P(X_1 \le x_1, X_2 \le x_2) &= P(F_1^{-1}(U_1) \le x_1, F_2^{-1}(U_2) \le x_2) \\ &= P(U_1 \le F_1(x_1), U_2 \le F_2(x_2)) \\ &= C(F_1(x_1), F_2(x_2)) \end{aligned}$$

where the last equality uses eq. (3).

A.1.3. *Bivariate copula simulation.* Let us now review how to simulate (U_1, U_2) from C. We follow a general method described in Nelsen (2007, Section 2.9), which uses the so-called multivariate quantile transform. In the univariate case, the central step in simulating from F_1 was to compute the function F_1^{-1}. In the bivariate case, the central step is to compute a function $h_{u_1}^{-1}(u_2)$, which will later be shown to be the inverse of the distribution function of U_2 conditional on U_1. After having written code which can evaluate this function, the details of which will be covered shortly, we may simulate from C as follows: Let $U_1 \sim U[0,1]$ and $V \sim U[0,1]$ be independent. Then set $U_2 = h_{U_1}^{-1}(V)$. The pair (U_1, U_2) is then distributed according to the copula C (Nelsen, 2007, Section 2.9).

We now define $h_{u_1}^{-1}(u_2)$. For each value $0 < u_1 < 1$, let

$$h_{u_1}(u_2) = \frac{\partial}{\partial u_1} C(u_1, u_2)$$

Then, for each $0 < u_1 < 1$, the function $h_{u_1}^{-1}$ is the (generalized) inverse of $h_{u_1}(u_2)$ as a function of u_2. Recall that for a function $H(x)$, its generalized inverse is defined as $H^{-1}(y) = \inf\{x : H(x) > y\}$, a definition which agrees with the standard inverse when H is invertible. As in the univariate case, where it was simpler to work with the case when F_1 was continuous and strictly increasing, we will again assume that h_{u_1} is continuous and strictly increasing for all u_1. We now show that this follows if C has a density c which is non-zero on $(0,1)^2$ and continuous in each coordinate. Both assumptions on c are fulfilled for all popular copula classes, and will therefore be assumed also in the following. To see that these two assumptions imply that h_{u_1} is continuous and strictly increasing for any u_1, notice that since

$$C(u_1, u_2) = \int_0^{u_1} \int_0^{u_2} c(x_1, x_2)\, dx_1 dx_2,$$

we have $h_{u_1}(u_2) = \int_0^{u_2} c(u_1, x_2)\, dx_2$ by the fundamental theorem of calculus. Since $c(u_1, u_2) > 0$, and since the function $x_2 \mapsto c(u_1, x_2)$ is continuous for a given u_1, the integral is strictly increasing and continuous in u_2 and hence $h_{u_1}(u_2)$ is strictly increasing and continuous in u_2. Further, we have $h_{u_1}(0) = 0$ and $h_{u_1}(1) = 1$, and $h_{u_1} : [0,1] \mapsto [0,1]$ is therefore a bijection for each u_1. That $h_{u_1}(0) = 0$ and

$h_{u_1}(1) = 1$ can be seen from noticing that $h_{u_1}(0) = \frac{\partial}{\partial u_1} C(u_1, 0) = \frac{\partial}{\partial u_1} 0 = 0$ since $C(u_1, 0) = P(U_1 \le u_1, U_2 \le 0) = 0$, and that since $C(u_1, 1) = P(U_1 \le u_1, U_2 \le 1) = P(U_1 \le u_1) = u_1$ we also have $h_{u_1}(1) = \frac{\partial}{\partial u_1} C(u_1, 1) = \frac{\partial}{\partial u_1} u_1 = 1$.

Finally, we show that $(U_1, U_2) \sim C$. We have not found an elementary presentation of this result in the literature (see Rüschendorf (2009, Section 3) for an authoritative account), and include it for completeness and since the following argument is short, and will be generalized progressively in the following. For compactness, we assume $h_{u_1}(u_2)$ is invertible with respect to u_2 for all $0 < u_1 < 1$. We have

$$\begin{aligned}
&P(U_1 \le u_1, U_2 \le u_2) = P(U_1 \le u_1, h_{U_1}^{-1}(V) \le u_2) \\
&\overset{(a)}{=} P(U_1 \le u_1, V \le h_{U_1}(u_2)) \overset{(b)}{=} \mathsf{E}\mathsf{E}[I\{U_1 \le u_1\} I\{V \le h_{U_1}(u_2)\} | U_1] \\
&\overset{(c)}{=} \mathsf{E} I\{U_1 \le u_1\} \mathsf{E}[I\{V \le h_{U_1}(u_2)\} | U_1] \overset{(d)}{=} \mathsf{E} I\{U_1 \le u_1\} h_{U_1}(u_2) \\
&= \int_0^1 I\{x_1 \le u_1\} h_{x_1}(u_2)\, dx_1 = \int_0^{u_1} h_{x_1}(u_2)\, dx_1 = \int_0^{u_1} \frac{\partial}{\partial x_1} C(x_1, u_2)\, dx_1 \\
&\overset{(e)}{=} C(u_1, u_2) - C(0, u_2) \overset{(f)}{=} C(u_1, u_2)
\end{aligned}$$

Explanations: *(a)* Apply h_{U_1} to both sides of the second inequality. *(b)* $P(A) = \mathsf{E} I\{A\}$. Double expectation. Also, $I\{A, B\} = I\{A\} I\{B\}$. *(c)*: Use $\mathsf{E}[H(X)Y|X] = H(X)\mathsf{E}[Y|X]$. *(d)*: Since V is independent to U_1, we have $\mathsf{E}[I\{V \le h_{U_1}(u_2)\}|U_1] = \mathsf{E}_V I\{V \le h_{U_1}(u_2)\}$ where the expectation is with respect to V only, and U_1 is fixed. Recalling that for a random variable X with CDF F_X we have $F_X(x) = P(X \le x) = \mathsf{E} I\{X \le x\}$ we have that $\mathsf{E}_V I\{V \le h_{U_1}(u_2)\} = h_{U_1}(u_2)$ since V is uniform on $[0, 1]$, i.e., V has CDF $F_U(u) = u$ for $0 \le u \le 1$. *(e)* Fundamental theorem of calculus. *(f)* Since $0 \le U_1 \le 1$ and continuous we have $C(0, u_2) = P(U_1 \le 0, U_2 \le u_2) = P(U_1 = 0, U_2 \le u_2) = 0$.

Practically speaking, we may therefore simulate from any copula by computing $h_{u_1}^{-1}(u_2)$, which requires the computation of a partial derivative and the inversion of a function. This may be done analytically in some cases, but in general, numerical approximation is required.

A.1.4. *Identifying correlations from a bivariate distribution.* We now consider how we may identify a θ_0 such that the distribution $F(x_1, x_2; \theta_0) = C(F_1(x_1), F_2(x_2); \theta_0)$ has a Pearson correlation ρ. The Höffding (1940) formula for correlation $\rho(\theta)$ gives

$$\begin{aligned}
\rho(\theta) = \mathrm{sd}(F_1)^{-1}\, \mathrm{sd}(F_2)^{-1} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} & C(F_1(z_1), F_2(z_2); \theta) \\
& - F_1(z_1) F_2(z_2)\, dz_1 dz_2, \qquad (4)
\end{aligned}$$

where $\mathrm{sd}(F_1), \mathrm{sd}(F_2)$ are the standard deviations of F_1, F_2. Most bivariate copula classes are such that $C(u_1, u_2; \theta)$ is increasing in θ, a property fulfilled by all copulas catalogued in Section 5.1 of Joe (1997). This implies (Grønneberg & Foldnes, 2017, Theorem 1) that $\rho(\theta)$ is an increasing function. It is therefore easy to solve for θ_0 via numerical root finding methods. The function $\rho(\theta)$ then has to be evaluated through numerical integration.

A.2. **How trivariate vine simulations are performed on a computer.** We here provide more technical details on how to simulate from the three dimensional vine distribution used as an example in the main text. As is generally the case, we only need to simulate from the vine copula (which by definition has uniform marginals), as the marginals are easily transformed to any desired marginal distributions in the same way as explained in the univariate and bivariate case, i.e., by applying the quantile functions of the marginals to each of the coordinates of the simulated vector from a copula. When this is done, the resulting distribution has the desired marginals, and the same copula as before transformation. That is, if C is a d dimensional copula, meaning C is a d dimensional cumulative distribution function with uniform marginals, and $(U_1, U_2, \dots, U_d) \sim C$, then $X = (F_1^{-1}(U_1), F_2^{-1}(U_2), \dots, F_d^{-1}(U_d))$ will have a distribution of the form

$$F(x_1, x_2, \dots, x_d) = C(F_1(x_1), F_2(x_d), \dots, F_d(x_d))$$

meaning X has marginals $F_1, F_2, \dots, F_d$ and C as its copula.

Let us first consider how to simulate from a three dimensional copula in general. Constructing multivariate distributions can be difficult, and vines provide a general construction which is often useful. A useful feature of this class is that some of the properties of the resulting distribution are well suited for computation, and in spite of the flexibility and simplicity of constructing vine distributions, simulating from them is straightforward.

We first consider how to simulate from a general three dimensional copula $(U_1, U_2, U_3)' \sim C$, which does not have to be a vine distribution. We first simulate (U_1, U_2) from the bivariate copula $C_{1,2}$ as described in the bivariate section above. Recall that $C_{1,2}$ can be easily computed from C, since $0 \leq U_3 \leq 1$ implies that $C_{1,2}(u_1, u_2) = P(U_1 \leq u_1, U_2 \leq u_2) = P(U_1 \leq u_1, U_2 \leq u_2, U_3 \leq 1) = C(u_1, u_2, 1)$. We may now simulate U_3 from the distribution of (U_1, U_2, U_3) after "conditioning away" the values of U_1, U_2. Again we may use the multivariate quantile transform. We simulate V which is uniform on $[0, 1]$ and independent from previously generated variables, and then let

$$U_3 = h^{-1}_{U_1,U_2}(V),$$

where $h^{-1}_{U_1,U_2}$ is the generalized inverse of h_{U_1,U_2}, which is the distribution of U_3 conditional on (U_1, U_2). That is,

$$h_{u_1,u_2}(u_3) = C_{3|12}(u_3|u_1, u_2).$$

Conditional distributions are conceptually complex, and will only be covered superficially here. While we will give some more needed technical details for conditional distributions in the next subsection, we follow common text-book treatments (e.g., Rice, 2006), and use the fact that if C has a density $c_{1,2,3}$, which we will assume, $C_{3|12}$ has density given by

$$c_{3|12}(u_3|u_1, u_2) = \frac{c_{1,2,3}(u_1, u_2, u_3)}{c_{1,2}(u_1, u_2)},$$

where

$$c_{1,2}(u_1, u_2) = \int_0^1 c_{1,2,3}(u_1, u_2, x_3)\, dx_3.$$

We then have that

$$C_{3|12}(u_3|u_1,u_2) = \int_0^{u_3} c_{3|12}(u_3|u_1,u_2)\,dx_3 = \int_0^{u_3} \frac{c_{1,2,3}(u_1,u_2,x_3)}{c_{1,2}(u_1,u_2)}\,dx_3$$
$$= \frac{\int_0^{u_3} c(u_1,u_2,x_3)\,dx_3}{c_{1,2}(u_1,u_2)}.$$

As in the bivariate case, $h_{u_1,u_2}(u_3)$ is seen to be invertible if we assume that c is continuous and non-zero on $(0,1)^3$, and generalized inverses are not needed when computing U_3. If a simple formula for the copula CDF is available, which we note is often not the case, we may avoid integration when computing h_{u_1,u_2}, since

$$C_{3|12}(u_3|u_1,u_2) = \frac{\int_0^{u_3} c(u_1,u_2,x_3)\,dx_3}{c_{1,2}(u_1,u_2)} = \frac{\frac{\partial^2}{\partial u_1 \partial u_2} C_{1,2,3}(u_1,u_2,u_3)}{\frac{\partial^2}{\partial u_1 \partial u_2} C_{1,2,3}(u_1,u_2,1)}. \tag{5}$$

When only the joint density of C is available, integration is in general needed for computing $C_{3|12}$, and this is achieved by numerical approximations.

Following this recipe gives $(U_1,U_2,U_3) \sim C_{1,2,3}$ by a similar argument as in the bivariate case: Again we assume $h_{u_1,u_2}(u_3)$ is invertible as a function of u_3 for all $0 < u_1, u_2 < 1$. We then have

$$\begin{aligned}
P(U_1 \le u_1, U_2 \le u_2, U_3 \le u_3) &= \mathsf{E} I\{U_1 \le u_1, U_2 \le u_2\} I\{h_{U_1,U_2}^{-1}(V) \le u_3\} \\
&= \mathsf{E}\mathsf{E}[I\{U_1 \le u_1, U_2 \le u_2\} I\{V \le h_{U_1,U_2}(u_3)\} | U_1, U_2] \\
&= \mathsf{E} I\{U_1 \le u_1, U_2 \le u_2\} \mathsf{E}[I\{V \le h_{U_1,U_2}(u_3)\} | U_1, U_2] \\
&\overset{(a)}{=} \mathsf{E} I\{U_1 \le u_1, U_2 \le u_2\} \mathsf{E}_V[I\{V \le h_{U_1,U_2}(u_3)\}] \\
&\overset{(b)}{=} \mathsf{E} I\{U_1 \le u_1, U_2 \le u_2\} h_{U_1,U_2}(u_3) \\
&= \int_0^{u_1} \int_0^{u_2} c_{12}(x_1,x_2) h_{x_1,x_2}(u_3)\,dx_1 dx_2 \\
&= \int_0^{u_1} \int_0^{u_2} \frac{\partial^2}{\partial x_1 \partial x_2} C_{1,2,3}(x_1,x_2,u_3)\,dx_1 dx_2 \\
&= C_{1,2,3}(u_1,u_2,u_3).
\end{aligned}$$

Explanations: *(a)* V is independent to U_1, U_2. *(b)* V is uniform.

Let us now apply this to three dimensional regular vines. The idea behind vines is described in Joe (1996) and (Joe, 2014, Sections 3.8 and 3.9), and is based on expressing cumulative distribution functions as mixtures of conditional distributions. The motivation for its construction will be sketched in the next sub-section, and we here simply state the distribution for a trivariate copula in terms of its CDF.

The three dimensional vine copula illustrated in Figure 4 has cumulative distribution

$$C_{1,2,3}(u_1,u_2,u_3) = \int_0^{u_2} C_{1,3;2}(C_{1|2}(u_1|z_2), C_{3|2}(u_3|z_2))\,dz_2 \tag{6}$$

where $C_{1,3;2}$ is a chosen bivariate copula to bind marginals 1 and 3 together, when given 2, and where

$$C_{1|2}(u_1, u_2) = \frac{\partial}{\partial u_2} C_{1,2}(u_1, u_2) = \int_0^{u_2} c_{1,2}(u_1, x_2)\, dx_2,$$
$$C_{3|2}(u_3, u_2) = \frac{\partial}{\partial u_2} C_{3,2}(u_3, u_2) = \int_0^{u_2} c_{3,2}(u_3, x_2)\, dx_2,$$

where $C_{1,2}$ and $C_{3,2}$ are chosen copulas, directly giving the copula of marginals $1, 2$ and $3, 2$ respectively. Notice $C_{1,3;2}$ is a standard bivariate copula, and does not depend on the value x_2 being integrated over in the above display. This is an important point which will be discussed further in the next sub-section.

There is an important distinction between $C_{1,3|2}$, which is the conditional distribution of (U_1, U_3) given U_2, and $C_{1,3;2}$, which as we will discuss later is the *copula* of $C_{1,3|2}$. The objects $C_{1,3|2}$ and $C_{1,3;2}$ need not be the same, and while $C_{1,2,3}$ have all uniform marginals, the marginals of the conditional distribution $C_{1,3|2}$ need *not* be uniform, which in turn implies that $C_{1,3|2}$ need not be a copula. We will return to this issue in the following. In order to further separate the two further, we will keep the C notation for functions such as $C_{1,3;2}$ – since these *are* actually copulas, but rather write $F_{1,3|2}$ to refer to the conditional distribution of (U_1, U_3) given U_2. That is, we will from now on write $F_{1,3|2} = C_{1,3|2}$.

Consider now how to simulate from this vine. Since (U_1, U_2) is simulated from $C_{1,2}$ which is directly specified in the lowest tree, its simulation procedure follows from the already described bivariate case. We now need to compute $F_{3|12}$. Recalling eq. (5), we have

$$\begin{aligned}
F_{3|12}(u_3|u_1, u_2) &= \frac{\frac{\partial^2}{\partial u_1 \partial u_2} C_{1,2,3}(u_1, u_2, u_3)}{c_{12}(u_1, u_2)} \\
&= \frac{\frac{\partial^2}{\partial u_1 \partial u_2} \int_0^{u_2} C_{1,3;2}(C_{1|2}(u_1|z_2), C_{3|2}(u_3|z_2))\, dz_2}{c_{12}(u_1, u_2)} \\
&= \frac{\frac{\partial}{\partial u_1} C_{1,3;2}(C_{1|2}(u_1|u_2), C_{3|2}(u_3|u_2))}{c_{12}(u_1, u_2)}.
\end{aligned}$$

A notable feature is that this expression only depends on bivariate distributions, which are usually computationally well-behaved.

A.3. **More details on the vine construction in the trivariate case.**

We here provide a sketch of the vine construction of Joe (1996). We are unaware of an elementary presentation of this material in the literature, and presentations such as those in Bedford & Cooke (2002); Joe (1996, 2014) require considerable technical training to read. We therefore include an elementary presentation of this material here, restricted to the trivariate case.

Using operations similar to the derivation on the validity of the general trivariate simulation method, we see that for any trivariate continuous copula C, we have for variables $(U_1, U_2, U_3) \sim C$ that

$$\begin{aligned}
&C(u_1, u_2, u_3) \\
&= P(U_1 \leq u_1, U_2 \leq u_2, U_3 \leq u_3) \\
&= \mathsf{E} I\{U_1 \leq u_1, U_2 \leq u_2, U_3 \leq u_3\} \\
&= \mathsf{E} I\{U_2 \leq u_2\} I\{U_1 \leq u_1, U_3 \leq u_3\} \\
&= \mathsf{E}\mathsf{E}[I\{U_2 \leq u_2\} I\{U_1 \leq u_1, U_3 \leq u_3\} | U_2] \\
&= \mathsf{E} I\{U_2 \leq u_2\} \mathsf{E}[I\{U_1 \leq u_1, U_3 \leq u_3\} | U_2] \\
&= \mathsf{E} I\{U_2 \leq u_2\} P(U_1 \leq u_1, U_3 \leq u_3 | U_2) \\
&= \int_0^1 I\{x_2 \leq u_2\} P(U_1 \leq u_1, U_3 \leq u_3 | U_2 = x_2) \, dx_2 \\
&= \int_0^{u_2} P(U_1 \leq u_1, U_3 \leq u_3 | U_2 = x_2) \, dx_2.
\end{aligned}$$

This calculation provides an expansion of the full distribution of (U_1, U_2, U_3) in terms of the conditional distribution of (U_1, U_3) given U_2. This conditional bivariate distribution $F_{1,3|2}(u_1, u_3|x_2) = P(U_1 \leq u_1, U_3 \leq u_3 | U_2 = x_2)$ has marginals $F_{1|2}(u_1|x_3)$ and $F_{3|2}(u_3|x_2)$, which can be derived using properties of conditional distributions. A non-rigorous heuristic argument for the formula for $F_{1|2}(u_1|x_2)$ is that

$$\begin{aligned}
F_{1|2}(u_1|x_2) &= P(U_1 \leq u_1 | U_2 = x_2) \\
&= \lim_{h \to 0} \frac{P(U_1 \leq u_1, x_2 \leq U_2 \leq x_2 + h)}{P(x_2 \leq U_2 \leq x_2 + h)} \\
&= \lim_{h \to 0} \frac{C_{1,2}(u_1, u_2 + h) - C_{1,2}(u_1, u_2)}{x_2 + h - x_2} \\
&= \lim_{h \to 0} \frac{C_{1,2}(u_1, u_2 + h) - C_{1,2}(u_1, u_2)}{h} \\
&= \frac{\partial}{\partial u_2} C_{1,2}(u_1, u_2),
\end{aligned}$$

using the uniformity of U_2. Similarly, $F_{3|2}(u_3|x_2) = \frac{\partial}{\partial u_2} C_{3,2}(u_3, u_2)$. A formal argument justifying the formulas for $F_{1|2}$ and $F_{3|2}$ requires the general and rather complex mathematical framework of conditional probability, as developed by Kolmogorov, see Kallenberg (2002). A nice feature flowing from our focus on simulation is that an alternative justification for the formula for conditional distributions is provided by its successful application in simulation.

Sklar's theorem applied for each given x_2 value to the conditional distribution $F_{1,3|2}(u_1, u_3|x_2) = P(U_1 \leq u_1, U_3 \leq u_3 | U_2 = x_2)$ shows that there is a class of copulas $C_{13;2}(u_1, u_3; x_2)$ varying with x_2, which is such that

$$F_{1,3|2}(u_1, u_3|x_2) = C_{13;2}(F_{1|2}(u_1|x_2), F_{3|2}(u_3|x_2); x_2).$$

Using the formulas we identified for $F_{1|2}(u_1|x_2), F_{3|2}(u_3|x_2)$ and recalling that we started with an expansion for $C(u_1, u_2, u_3)$, we have shown that

$$C(u_1, u_2, u_3) = \int_0^{u_2} C_{13;2}\left(\frac{\partial}{\partial u_2} C_{1,2}(u_1, u_2), \frac{\partial}{\partial u_2} C_{3,2}(u_3, u_2); x_2\right) dx_2. \tag{7}$$

which holds in general. This expression can also be used to construct multivariate distributions from bivariate distributions: Based on bivariate copulas $C_{1,2}$ and $C_{3,2}$ we may compute $\frac{\partial}{\partial u_2} C_{1,2}(u_1, u_2)$, $\frac{\partial}{\partial u_2} C_{3,2}(u_3, u_2)$, and they may be combined using a family of copulas $C_{13;2}(u_1, u_3; x_2)$ for every x_2. For each x_2, the Sklar theorem implies that

$$C_{13;2}\left(\frac{\partial}{\partial u_2} C_{1,2}(u_1, u_2), \frac{\partial}{\partial u_2} C_{3,2}(u_3, u_2); x_2\right)$$

is a proper distribution. However, the family of copulas $C_{13;2}(u_1, u_3; x_2)$ has to be linked together via their x_2 dependence in such that the resulting C in eq. (7) is a proper CDF. This may be challenging, and does not have a simple solution.

The vine copula construction assumes that the family $C_{13;2}(u_1, u_3; x_2)$ is constant in x_2, i.e., does not depend on x_2. This is known as the simplifying assumption (Joe, 2014). We therefore write

$$C_{13;2}(u_1, u_3; x_2) = C_{13;2}(u_1, u_3), \tag{8}$$

and see that we re-gain the vine CDF of eq. (6). Since $C_{13;2}(u_1, u_3)$ does not vary with x_2, the combination from eq. (6) always results in a valid CDF, as may be seen as follows. We may consider the algorithm for simulating from $C_{1,2,3}$. After having simulated from (U_1, U_2) using previously described bivariate techniques, we define $U_3 = h^{-1}_{U_1,U_2}(V)$ from an independent $V \sim U[0, 1]$. Clearly, U_3 is a random variable, and by the above argument, the joint distribution of (U_1, U_2, U_3) is precisely $C_{1,2,3}$ from eq. (6), and hence $C_{1,2,3}$ is indeed a valid distribution function since it is the CDF of a random vector. By eq. (7) the constructed distribution has $C_{13;2}$ as the copula of the conditional distribution of (U_1, U_3) given U_2.

A.4. **The density of a four-dimensional vine.** In the main text, we gave the the density of the three dimensional vine in Figure 4 without a complete technical description. We here rectify this by deriving the density of the more general four-dimensional vine as depicted in Figure 10, and sketch how to form such densities in general. How to simulate from this four dimensional vine will be the topic of the next section. Our discussion of this four dimensional example ought to be sufficient to prepare the reader to understand general descriptions of simulation in e.g., Dissmann, Brechmann, Czado & Kurowicka (2013); Joe (2014), as well as fully understanding the vine based VITA simulation methodology of Grønneberg & Foldnes (2017).

The copula density of a vine is found by multiplying all copulas that are chosen as edge copulas. These copulas are evaluated at rather specific points, which will be discussed in the following. The edge copulas are the copulas of bivariate conditional distributions from the resulting full copula, with conditioning set indicated by the edge names. For example, the top-most edge connects (U_1, U_4) and conditions on (U_2, U_3), and represents the copula of $F_{1,4|2,3}$. Its contribution to the full density therefore includes a multiplicative factor $c_{1,4;2,4}$, where the use of a semi-colon indicates that this is the copula of a conditional distribution. As explained in the previous section, we write all conditional distributions of c such as the actual conditional distribution (here, a density) of (U_1, U_4) conditional on (U_2, U_3) using the notation $f_{1,4|2,3}$ for the density and $F_{1,3|2,3}$ for the CDF.

Conditional marginals are the key to the general description of writing down the density of a vine copula based on its vine, such as Figure 10, as they are included in the multiplicative contribution from each edge. For any edge on the vine, the edge may be denoted by $(i, j|\mathbf{v})$ where i, j are the marginals connected by this edge, and $\mathbf{v}$ may contain several indices (or none, as is the case at the lowest tree) which are conditioned on. For example, the top-most tree in Figure 10 contains only one edge, where $i = 1, j = 4$ and $\mathbf{v} = \{2, 3\}$. In contrast, the second tree in Figure 10 contains two edges. For the left-most edge, we have $i = 1, j = 3$ and $\mathbf{v} = \{2\}$. For the right-most edge, we have $i = 2, j = 4$ and $\mathbf{v} = \{3\}$. For the lowest tree, we do not condition on any indices, and so $\mathbf{v}$ is always empty. Going from left to right, the first edge has $i = 1, j = 2$, the next edge has $i = 2, j = 3$, and the final edge has $i = 3, j = 4$.

The multiplicative contribution of every edge $i, j|\mathbf{v}$ is $c_{i,j|\mathbf{v}}(F_{i|\mathbf{v}}(u_i|u_{\mathbf{v}}), F_{j|\mathbf{v}}(u_j|u_{\mathbf{v}}))$, where $u_{\mathbf{v}} = (u_k : k \in \mathbf{v})$, and where $F_{i|\mathbf{v}}(u_i|u_{\mathbf{v}})$ is the conditional cumulative distribution of U_i given $\{U_k : k \in \mathbf{v}\}$.

When $\mathbf{v}$ is empty, $F_{i|\mathbf{v}}(u_i|u_{\mathbf{v}})$ is the actual cumulative distribution of U_i, which is uniform since c is a copula. In these cases, we have $F_{i|\mathbf{v}}(u_i|u_{\mathbf{v}}) = u_i$. Therefore, the contributions of the lowest tree is simply $c_{1,2}(u_1, u_2)c_{2,3}(u_2, u_3)c_{3,4}(u_3, u_4)$.

Combining this description for the vine in Figure 10, we see that

$$\begin{aligned} c_{1,2,3,4}(u_1, u_2, u_3, u_4) =& c_{1,2}(u_1, u_2)c_{2,3}(u_2, u_3)c_{3,4}(u_3, u_4) \\ & c_{1,3;2}(F_{1|2}(u_1|u_2), F_{3|2}(u_3|u_2)) \\ & c_{2,4;3}(F_{2|3}(u_2|u_3), F_{4|3}(u_4|u_3)) \\ & c_{1,4;2,3}(F_{1|2,3}(u_1|u_3, u_3), F_{4|2,3}(u_4|u_2, u_3)) \end{aligned}$$

Now each of the bivariate copulas, i.e., each $c_{i,j;\mathbf{v}}$ are chosen by us, and therefore does not require further calculation to be evaluated. In contrast, the conditional marginal distributions have to be computed, and we now explain how this is done. We note that for general regular vines, there is a simple recursive method to calculate the required conditional densities, see Section 2.4 of Dissmann et al. (2013).

For the lowest tree, the marginals are uniform, and we do not need to deal with them. For the second tree, use

$$F_{i|j}(u_i|u_j) = \frac{\partial}{\partial u_j} C_{i,j}(u_i, u_j)$$

as discussed above. For the third, and here highest tree, we need to compute $F_{1|2,3}$ and $F_{4|2,3}$. Since we assume the vine copula has a density, $F_{1|2,3}$ is the cumulative distribution function of the density

$$f_{1|23}(u_1|u_2, u_3) = \frac{c_{1,2,3}(u_1, u_2, u_3)}{c_{2,3}(u_2, u_3)}. \tag{9}$$

Now, (U_1, U_2, U_3) is also generated from a vine, and this vine can be found by removing everything that has to do with the other variables. Here, this sub-vine results in exactly the three dimensional vine we used in earlier examples, see Figure 4. Therefore, we know that the joint distribution of (U_1, U_2, U_3) has joint density

$$c_{1,2,3}(u_1, u_2, u_3) = c_{1,2}(u_1, u_2)c_{2,3}(u_2, u_3)c_{1,3;2}(F_{1|2}(u_1|u_2), F_{3|2}(u_3|u_2)).$$

Inserting this into eq. (9) gives

$$\begin{aligned} f_{1|23}(u_1|u_2,u_3) &= \frac{c_{1,2,3}(u_1,u_2,u_3)}{c_{2,3}(u_2,u_3)} \\ &= c_{1,2}(u_1,u_2)c_{1,3;2}(F_{1|2}(u_1|u_2),F_{3|2}(u_3|u_2)) \\ &= c_{1,2}(u_1,u_2)c_{1,3;2}\left(\frac{\partial}{\partial u_2}C_{1,2}(u_1,u_2),\frac{\partial}{\partial u_2}C_{2,3}(u_2,u_3)\right) \end{aligned}$$

Recalling that we wish to identify not the density $f_{1|23}$ but instead the cumulative distribution function $F_{1|23}$, we integrate with respect to u_1. Now for $u_1 < 0$ or $u_1 > 1$ we have $c_{1,2}(u_1,u_2) = 0$, and therefore we start integrating at 0, and get, for $u_1 \leq 1$, that

$$F_{1|23}(u_1|u_2,u_3) = \int_0^{u_1} c_{1,2}(x_1,u_2)c_{1,3;2}\left(\frac{\partial}{\partial u_2}C_{1,2}(x_1,u_2),\frac{\partial}{\partial u_2}C_{2,3}(u_2,u_3)\right)dx_1.$$

Using the substitution

$$y = \frac{\partial}{\partial u_2}C_{1,2}(x_1,u_2),$$

which has derivative

$$\frac{d}{dx_1}y = \frac{\partial^2}{\partial x_1 \partial u_2}C_{1,2}(x_1,u_2) = c_{1,2}(x_1,u_2),$$

integration with substitution gives

$$\begin{aligned} (10) \qquad & F_{1|23}(u_1|u_2,u_3) \\ &= \int_0^{y(u_1)} c_{1,3;2}\left(y,\frac{\partial}{\partial u_2}C_{2,3}(u_2,u_3)\right)dy \\ &= \int_0^{y(u_1)} \frac{\partial^2}{\partial u_1 \partial u_2}C_{1,3;2}\left(y,\frac{\partial}{\partial u_2}C_{2,3}(u_2,u_3)\right)dy \\ &= D_2C_{1,3;2}\left(y(u_1),D_2C_{2,3}(u_2,u_3)\right) \\ &= D_2C_{1,3;2}\left(D_2C_{1,2}(u_1,u_2),D_2C_{2,3}(u_2,u_3)\right). \end{aligned}$$

where we, to avoid notational ambiguity, use the notation $D_iH(x_1,x_2) = (\partial/\partial x_i)H(x_1,x_2)$.

By a similar argument, we identify $F_{4|23}$. We have that

$$f_{4|23}(u_4|u_2,u_3) = \frac{c_{2,3,4}(u_2,u_3,u_4)}{c_{2,3}(u_2,u_3)}$$

where the density of the sub-vine (U_2,U_3,U_4) is deduced by the previously described general technique, giving

$$c_{2,3,4}(u_2,u_3,u_4) = c_{2,3}(u_2,u_3)c_{3,4}(u_3,u_4)c_{2,4;3}(F_{2|3}(u_2|u_3),F_{4|3}(u_4|u_3))$$

and therefore

$$f_{4|23}(u_4|u_2,u_3) = c_{3,4}(u_3,u_4)c_{2,4;3}(D_2C_{2,3}(u_2,u_3),D_1C_{3,4}(u_3,u_4)).$$

It then follows that

$$(11) \qquad F_{4|2,3}(u_4|u_2,u_3) = D_1C_{2,4;3}(D_2C_{2,3}(u_2,u_3),D_1C_{3,4}(u_3,u_4)).$$

Combining the above derivations gives a complete expression for the density of the vine in Figure 10. A notable feature is that numerical integration is avoided, at least when the densities and cumulative distribution functions of the bivariate copulas chosen by the user can be evaluated without numerical integration, which is usually the case for commonly used bivariate copulas. In cases where numerical integration is required, only bivariate numerical integration is needed, which is considerably less complex than general high dimensional integration.

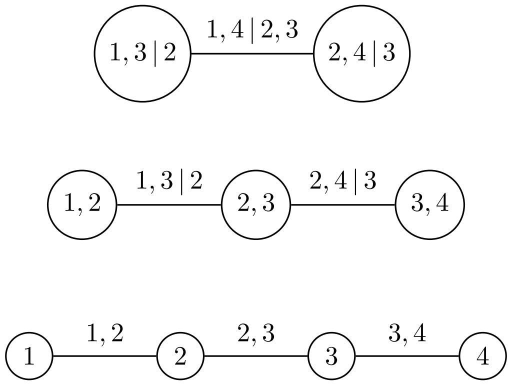


FIGURE 10. A four-dimensional regular vine.

A.5. **How to simulate from a vine.** Since the three-dimensional case considered earlier is too simple to easily see the general pattern of how to simulate from a general multivariate regular vine, we here consider the four-dimensional vine of Figure 10. The four dimensional case is sufficiently complex for the general case to be within reach after having studied it.

A.5.1. *Simulation from a general p dimensional copula.* We start by providing a general algorithm for simulating from an arbitrary p dimensional copula. This method, while general, will in high dimensions often be numerically infeasible, as there are no closed form expressions for the quantities required for applying the algorithm and numerical approximations have to be employed. In contrast, we will see that simulating from vines are computationally simpler to simulate from, mainly since simulation in most cases does not require high dimensional numerical integration. Vine simulation, illustrated via a four dimensional example, will be explained in the next section.

The general simulation method we now present extends the bivariate and trivariate examples given above, and continues to use the multivariate quantile transformation. For p variables this transform takes the following form (Rüschendorf, 2009, Section 3). We want to simulate from a p dimensional copula CDF C. We simulate $V_1, V_2, \ldots, V_p$ which are independent and uniform on $[0, 1]$. Then we let $U_1 = V_1$ (the marginals are already uniform), and recursively define

$$U_j = F^{-1}_{j|1,2,\ldots,j-1}(V_j|U_1, U_2, \ldots, U_{j-1})$$

where $F^{-1}_{j|1,2,\ldots,j-1}$ is the generalized inverse of the conditional distribution function

$$F_{j|1,2,\ldots,j-1}(u_j|u_1, u_2, \ldots, u_{j-1}).$$

For simplicity, we will assume that C has a density. Let $C_{1,2,\ldots,j}$ denote the distribution of $(U_1, U_2, \ldots, U_j)$, given by $C_{1,2,\ldots,j}(u_1, u_2, \ldots, u_j) = C(u_1, u_2, \ldots, u_j, 1, 1, \ldots, 1)$, and let $c_{1,2,\ldots,j}$ denote its density. For simplicity, we assume that the density $c_{1,2,\ldots,j}$ is strictly positive for all inner points in the unit cube $[0, 1]^j$. Recall that the density of a subset of the variables, such as the density $c_{1,2,\dots,j}$ can be found by $c_{1,2,\dots,j}(u_1,u_2,\dots,u_j) = \partial^j C(u_1,u_2,\dots,u_j,1,1,\dots,1)/(\partial u_1\cdots\partial u_j)$. We have

$$\begin{aligned}
&F_{j|1,2,\dots,j-1}(u_j|u_1,u_2,\dots,u_{j-1})\\
&= \int_0^{u_j} \frac{c_{1,2,\dots,j}(u_1,\dots,x_j)}{c_{1,2,\dots,j-1}(u_1,\dots,u_{j-1})}\,dx_j = \frac{\int_0^{u_j} c_{1,2,\dots,j}(u_1,\dots,x_j)\,dx_j}{c_{1,2,\dots,j-1}(u_1,\dots,u_{j-1})}\\
&= \frac{\partial^{j-1}}{\partial u_1\partial u_2\cdots\partial u_{j-1}} C_{1,2,\dots,j}(u_1,u_2,\dots,u_j)\left(c_{1,2,\dots,j-1}(u_1,\dots,u_{j-1})\right)^{-1}
\end{aligned}$$

Notice that if, say, only the density c of C is known, the computation of $F_{j|1,2,\dots,j-1}$ requires numerical integration routines in order to approximate the integral $\int_0^{u_j} c_{1,2,\dots,j}(u_1,\dots,x_j)\,dx_j$.

The bivariate and trivariate simulation methods also follow the above pattern, and we have shown earlier that they work as intended. We may therefore conclude that the general method is valid using a proof by induction: Supposing this works for generating $(U_1,U_2,\dots,U_{j-1})$, we prove that it also works for $(U_1,U_2,\dots,U_j)$. By the induction hypothesis, $(U_1,U_2,\dots,U_{j-1})$ is already generated as required. Then we generate

$$U_j = h^{-1}_{U_1,U_2,\dots,U_{j-1}}(V_j)$$

where $h^{-1}_{u_1,u_2,\dots,u_{j-1}}(u_j)$ is the (generalized) inverse function of $F_{j|1,2,\dots,j-1}(u_j|u_1,u_2,\dots,u_{j-1})$. Again we restrict attention to the cases where h is in invertible in the regular sense. We have

$$\begin{aligned}
&P(U_1\le u_1,\dots,U_{j-1}\le u_{j-1},U_j\le u_j)\\
&= \mathsf{E}\big(I\{U_1\le u_1,\dots,U_{j-1}\le u_{j-1}\}\mathsf{E}[I\{h^{-1}_{U_1,U_2,\dots,U_{j-1}}(V_j)\le u_j\}|U_1,\dots,U_{j-1}]\big)\\
&= \mathsf{E}\big(I\{U_1\le u_1,\dots,U_{j-1}\le u_{j-1}\}\mathsf{E}[I\{V_j\le h_{U_1,U_2,\dots,U_{j-1}}(u_j)\}|U_1,\dots,U_{j-1}]\big)\\
&= \mathsf{E}I\{U_1\le u_1,\dots,U_{j-1}\le u_{j-1}\}h_{U_1,U_2,\dots,U_{j-1}}(u_j)\\
&= \mathsf{E}I\{U_1\le u_1,\dots,U_{j-1}\le u_{j-1}\}F_{1,\dots,j-1}(u_j|U_1,U_2,\dots,U_{j-1})\\
&= \int_0^{u_1}\cdots\int_0^{u_{j-1}} c_{1,\dots,j-1}(x_1,\dots,x_{j-1})F_{1,\dots,j-1}(u_j|x_1,x_2,\dots,x_{j-1})\,dx_1\cdots dx_{j-1}\\
&= \int_0^{u_1}\cdots\int_0^{u_{j-1}} \frac{\partial^{j-1}}{\partial u_1\partial u_2\cdots\partial u_{j-1}}C_{1,2,\dots,j}(x_1,x_2,\dots,x_{j-1},u_j)\,dx_1\cdots dx_{j-1}\\
&= C_{1,2,\dots,j}(u_1,u_2,\dots,u_j).
\end{aligned}$$

as required.

A.5.2. *Simulating from a four-dimensional vine.* Let us now see how to apply this general technique to the four dimensional vine copula distribution represented in Figure 10. Again, simulation can be performed without needing numerical integration. The general simulation approach from the multivariate quantile transform always simulate from $F_{j|1,2,\dots,j-1}$ with j starting at 1 and increasing up to p. This will always work, but for vines, we may simulate directly from the bivariate conditional distributions specified in the vine. Simulating from a bivariate conditional distribution will amount to simulate from two conditional distributions that are connected via bivariate copulas. This will in total lead to the same steps as the multivariate quantile transform, but has the advantage of having computations that are simpler to follow, as we follow the structure of the vine. A general simulation algorithm for regular vines is given in Algorithm 2.2 of Dissmann et al. (2013).

The main insight we need is that we have direct knowledge of certain conditional distributions from how the vine distribution is specified. We have easy access to the following conditional (and unconditional) distributions

$$\begin{aligned}&F_{1,4|2,3},\\&F_{1,3|2}, F_{2,4|3},\\&F_{1,2}, F_{2,3}, F_{3,4}.\end{aligned}$$

These distributions are all bivariate, and, as we have seen from constructing the joint density of (U_1, U_2, U_3, U_4), can be joined to produce the full joint distribution of (U_1, U_2, U_3, U_4).

We already know how to simulate from bivariate distributions. Let us see how this can be extended to simulating from bivariate conditional distributions.

Suppose therefore, that we have simulated (U_2, U_3) in such a way that it has the required bivariate distribution, i.e., it has the cumulative distribution function $C_{2,3}(u_2, u_3) = C(1, u_2, u_3, 1)$. We may do this directly using previously described techniques, since the copula and marginals of U_2, U_3 are known and directly specified.

To simulate the remaining coordinates U_1, U_4 we start by simulating from the conditional distribution of U_1, U_4 when conditioning on U_2, U_3, whose conditional CDF is denoted by $F_{1,4|2,3}$. By the simplifying assumption, the copula $C_{1,4;2,3}(u_1, u_4; u_2, u_3)$ of $F_{1,4|2,3}$ does not depend on u_2, u_3, and we therefore write $C_{1,4;2,3}(u_1, u_4; u_2, u_3) = C_{1,4;2,3}(u_1, u_4)$. Due to the simplifying assumption, $C_{1,4;2,3}$ is further a bivariate copula that we have chosen, which connects U_1, U_4 when conditioning on U_2, U_3. This implies that

$$F_{1,4|2,3}(u_1, u_3|u_2, u_3) = C_{1,4;2,3}(F_{1|2,3}(u_1|u_2, u_3), F_{4|2,3}(u_4|u_2, u_3))$$

How should we simulate from $F_{1,4|2,3}$? We will show that if u_2 and u_3 are fixed to the already simulated U_2 and U_3 respectively, we may treat $F_{1,4|2,3}$ as if it is a standard (non-conditional) distribution, and use already described techniques to simulate from this bivariate distribution. The resulting variables will be valid simulations of the remaining U_1, U_4.

Since $F_{1,4|2,3}$ is a bivariate distribution with non-uniform marginals, we will as before split this simulation into two steps. Firstly, we simulate W_1, W_4 from its copula, which is $C_{1,4;2,3}$. By the simplifying assumption, this copula is a standard bivariate copula which does not depend on variables simulated earlier. We will then transform W_1, W_4 using univariate quantile transforms so that they have distributions $F_{1|2,3}$ and $F_{4|2,3}$ respectively.

Let us first simulate W_1, W_4 from the bivariate copula $C_{1,4;2,3}$. How this is done has been explained earlier: We simulate independent V_1, V_4 from $U[0, 1]$. Then we set $W_4 = V_4$ and

$$W_1 = h_{W_4}^{(1,4;2,3)}(V_1)$$

where

$$h_{w_4}^{(1,4;2,3)}(v_1) = D_1 C_{1,4;2,3}(v_1, w_4).$$

Again note that due to the simplifying assumption, this step does not depend on the already simulated U_2, U_3.

We next transform W_1, W_4 so that they are $F_{1|2,3}$ and $F_{3|2,3}$ distributed respectively. An important point here is that this is where dependence from U_2 and U_3

is introduced. We again use the univariate quantile transform and set

$$U_1 = F^{-1}_{1|2,3}(W_1|U_2, U_3), \quad U_4 = F^{-1}_{4|2,3}(W_4|U_2, U_3)$$

where $F_{1|2,3}$ is given in eq. (10) (p.33) and $F_{4|2,3}$ is given in eq. (11) (p.33), considering the already simulated U_2, U_3 as fixed. It is not immediately apparent that U_1, U_4 have uniform marginals. This can be shown directly, but we will now show the more general fact that $(U_1, U_2, U_3, U_4) \sim C$, and since all marginals in C are uniform, this also implies that U_1, U_4 have uniform marginals.

Let us for completeness (and since we do not know an elementary reference for this fact) show the formal validity of this simulation method. That is, let the variables generated in this fashion be denoted U_1, U_2, U_3, U_4. We will show that the joint CDF of these variables is C. Again, all inverse functions are assumed to be traditional inverse functions. Since $(W_1, W_4) \sim C_{1,4;2,3}$ is independent to (U_2, U_3) we have

$$\begin{aligned}
&P(U_1 \le u_1, U_2 \le u_2, U_3 \le u_3, U_4 \le u_4) \\
=&\mathsf{E} I\{U_2 \le u_2, U_3 \le u_3\}\mathsf{E}[I\{F^{-1}_{1|2,3}(W_1|U_2, U_3) \le u_1, F^{-1}_{4|2,3}(W_4|U_2, U_3) \le u_4\}|(U_2, U_3)] \\
=&\mathsf{E} I\{U_2 \le u_2, U_3 \le u_3\}\mathsf{E}[I\{W_1 \le F_{1|2,3}(u_1|U_2, U_3), W_4 \le F_{4|2,3}(u_4|U_2, U_3)\}|(U_2, U_3)] \\
\overset{(a)}{=}&\mathsf{E} I\{U_2 \le u_2, U_3 \le u_3\}\mathsf{E}_{W_1,W_2}[I\{W_1 \le F_{1|2,3}(u_1|U_2, U_3), W_4 \le F_{4|2,3}(u_4|U_2, U_3)\}] \\
=&\mathsf{E} I\{U_2 \le u_2, U_3 \le u_3\}P_{W_1,W_2}\big(W_1 \le F_{1|2,3}(u_1|U_2, U_3), W_4 \le F_{4|2,3}(u_4|U_2, U_3)\big) \\
\overset{(b)}{=}&\mathsf{E} I\{U_2 \le u_2, U_3 \le u_3\}C_{1,4;2,3}(F_{1|2,3}(u_1|U_2, U_3), F_{4|2,3}(u_4|U_2, U_3)) \\
=&\int_0^{u_2}\int_0^{u_3} c_{2,3}(x_2, x_3)C_{1,4;2,3}(F_{1|2,3}(u_1|x_2, x_3), F_{4|2,3}(u_4|x_2, x_3))\, dx_2 dx_3
\end{aligned}$$

Explanations: *(a)* Use that (W_1, W_4) is independent to (U_2, U_3). E_{W_1,W_2} and P_{W_1,W_2} means that we integrate only over W_1, W_2. *(b)* Use that $(W_1, W_4) \sim C_{1,4;2,3}$.

Since

$$C_{1,4;2,3}(F_{1|2,3}(u_1|x_2, x_3), F_{4|2,3}(u_4|x_2, x_3)) = F_{1,4|2,3}(u_1, u_4|x_2, x_3)$$

where $F_{1,4|2,3}$ is the conditional distribution of (U_1, U_4) conditional on (U_2, U_3), we have that

$$\begin{aligned}
C_{1,4;2,3}(F_{1|2,3}(u_1|x_2, x_3), F_{4|2,3}(u_4|x_2, x_3)) &= \int_0^{u_1}\int_0^{u_4} c_{1,4|2,3}(x_1, x_4|x_2, x_3)\, dx_1 x_4 \\
&= \int_0^{u_1}\int_0^{u_4} \frac{c_{1,2,3,4}(x_1, x_2, x_3, x_4)}{c_{2,3}(x_2, x_3)}\, dx_1 x_4.
\end{aligned}$$

Therefore,

$$\begin{aligned}
&P(U_1 \le u_1, U_2 \le u_2, U_3 \le u_3, U_4 \le u_4) \\
&= \int_0^{u_2}\int_0^{u_3} c_{2,3}(x_2, x_3)C_{1,4;2,3}(F_{1|2,3}(u_1|x_2, x_3), F_{4|2,3}(u_4|x_2, x_3))\, dx_2 x_3 \\
&= \int_0^{u_2}\int_0^{u_3} c_{2,3}(x_2, x_3)\int_0^{u_1}\int_0^{u_4} \frac{c_{1,2,3,4}(x_1, x_2, x_3, x_4)}{c_{2,3}(x_2, x_3)}\, dx_1 x_4 dx_2 x_3 \\
&= \int_0^{u_1}\int_0^{u_2}\int_0^{u_3}\int_0^{u_4} c_{1,2,3,4}(x_1, x_2, x_3, x_4) dx_1 dx_2 dx_3 dx_4 \\
&= C(u_1, u_2, u_3, u_4),
\end{aligned}$$

showing the validity of the simulation.

References

Aas, K., Czado, C., Frigessi, A. & Bakken, H. (2009). Pair-copula constructions of multiple dependence. *Insurance: Mathematics and economics* **44**, 182–198.

Bedford, T. & Cooke, R. M. (2002). Vines–a new graphical model for dependent random variables. *The Annals of Statistics* **30**, 1031–1068.

Bentler, P. (2006). Eqs 6 structural equations program manual.

Boomsma, A. (2013). Reporting monte carlo studies in structural equation modeling. *Structural Equation Modeling: A Multidisciplinary Journal* **20**, 518–540.

Box, G. E. P. (1954). Some theorems on quadratic forms applied in the study of analysis of variance problems, i. effect of inequality of variance in the one-way classification. *The annals of mathematical statistics* **25**, 290–302.

Cain, M. K., Zhang, Z. & Yuan, K.-H. (2017). Univariate and multivariate skewness and kurtosis for measuring nonnormality: Prevalence, influence and estimation. *Behavior research methods* **49**, 1716–1735.

Cario, M. C. & Nelson, B. L. (1997). Modeling and generating random vectors with arbitrary marginal distributions and correlation matrix. Tech. rep., Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois.

Christoffersson, A. (1977). Two-step weighted least squares factor analysis of dichotomized variables. *Psychometrika* **42**, 433–438.

Curran, P. J., West, S. G. & Finch, J. F. (1996). The robustness of test statistics to nonnormality and specification error in confirmatory factor analysis. *Psychological Methods* **1**, 16–29.

Dimitrov, D. M. (2002). Reliability: Arguments for multiple perspectives and potential problems with generalization across studies. *Educational and Psychological Measurement* **62**, 783–801.

Dissmann, J., Brechmann, E. C., Czado, C. & Kurowicka, D. (2013). Selecting and estimating regular vine copulae and application to financial returns. *Computational Statistics & Data Analysis* **59**, 52–69.

Embrechts, P., Lindskog, F. & McNeil, A. (2001). Modelling dependence with copulas. *Rapport technique, Département de mathématiques, Institut Fédéral de Technologie de Zurich, Zurich* **14**.

Flora, D. B. & Curran, P. J. (2004). An empirical evaluation of alternative methods of estimation for confirmatory factor analysis with ordinal data. *Psychological methods* **9**, 466–491.

Foldnes, N. & Grønneberg, S. (2020a). *covsim: Simulate from Distributions with Given Covariance Matrix and Marginal Information*. R package version 0.1.0.

Foldnes, N. & Grønneberg, S. (2020b). *discnorm: Test for Discretized Normality in Ordinal Data*. R package version 0.1.0.

Foldnes, N. & Grønneberg, S. (2015). How general is the Vale–Maurelli simulation approach? *Psychometrika* **80**, 1066–1083.

Foldnes, N. & Grønneberg, S. (2017a). Approximating test statistics using eigenvalue block averaging. *Structural Equation Modeling: A Multidisciplinary Journal* , 1–14.

Foldnes, N. & Grønneberg, S. (2017b). The asymptotic covariance matrix and its use in simulation studies. *Structural Equation Modeling: A Multidisciplinary Journal* , 1–16.

Foldnes, N. & Grønneberg, S. (2019a). On identification and non-normal simulation in ordinal covariance and item response models. *Psychometrika* **84**, 1000–1017.

Foldnes, N. & Grønneberg, S. (2019b). Pernicious polychorics: The impact and detection of underlying non-normality. *Structural Equation Modeling: A Multidisciplinary Journal* , 1–19.

Foldnes, N. & Grønneberg, S. (2021). The sensitivity of structural equation modeling with ordinal data to underlying non-normality and observed distributional forms. *Psychological Methods* .

Foldnes, N. & Olsson, U. H. (2015). Correcting too much or too little? The performance of three chi-square corrections. *Multivariate behavioral research* **50**, 533–543.

Foldnes, N. & Olsson, U. H. (2016). A simple simulation technique for nonnormal data with prespecified skewness, kurtosis, and covariance matrix. *Multivariate behavioral research* **51**, 207–219.

Fouladi, R. (2000). Performance of modified test statistics in covariance and correlation structure analysis under conditions of multivariate nonnormality. *Structural Equation Modeling: A Multidisciplinary Journal* **7**, 356–410.

Frees, E. W. & Valdez, E. A. (1998). Understanding relationships using copulas. *North American actuarial journal* **2**, 1–25.

Genest, C. & Favre, A.-C. (2007). Everything you always wanted to know about copula modeling but were afraid to ask. *Journal of hydrologic engineering* **12**, 347–368.

Grimm, K. J. & Widaman, K. F. (2010). Residual structures in latent growth curve modeling. *Structural Equation Modeling: A Multidisciplinary Journal* **17**, 424–442.

Grønneberg, S. & Foldnes, N. (2017). Covariance model simulation using regular vines. *Psychometrika* **82**, 1035–1051.

Grønneberg, S. & Foldnes, N. (2019a). A problem with discretizing Vale-Maurelli in simulation studies. *Psychometrika* **84**, 554–561.

Grønneberg, S. & Foldnes, N. (2019b). Testing model fit by bootstrap selection. *Structural Equation Modeling: A Multidisciplinary Journal* **26**, 182–190.

Grønneberg, S., Foldnes, N. & Marcoulides, K. M. (2022). covsim: An R package for simulating non-normal data for structural equation models using copulas. *Journal of Statistical Software* Forthcoming.

Grønneberg, S. & Hjort, N. L. (2014). The copula information criteria. *Scandinavian Journal of Statistics* **41**, 436–459.

Hofert, M., Kojadinovic, I., Maechler, M. & Yan, J. (2013). *copula: Multivariate Dependence with Copulas.* R package version 0.999-7.

Höffding, W. (1940). Masstabinvariante korrelationstheorie. *Schriften des Mathematischen Instituts und Instituts fur Angewandte Mathematik der Universitat Berlin* **5**, 181–233.

Joe, H. (1996). Families of m-variate distributions with given margins and m (m-1)/2 bivariate dependence parameters. *Lecture Notes-Monograph Series* , 120–141.

Joe, H. (1997). *Multivariate models and multivariate dependence concepts*, vol. 73. Chapman & Hall/CRC.

Joe, H. (2014). *Dependence modeling with copulas.* CRC Press.

Joe, H. & Kurowicka, D. (2011). *Dependence modeling: vine copula handbook.* World Scientific.

Jöreskog, K. G. (1967). Some contributions to maximum likelihood factor analysis. *Psychometrika* **32**, 443–482.

Jöreskog, K. & Sörbom, D. (2006). Lisrel version 8.8. lincolnwood, il: Scientific software international.

Kallenberg, O. (2002). *Foundations of modern probability.* Springer-Verlag, 2nd ed.

Laenen, A., Alonso, A., Molenberghs, G. & Vangeneugden, T. (2009). Reliability of a longitudinal sequence of scale ratings. *Psychometrika* **74**, 49.

Li, C.-H. (2016). The performance of ml, dwls, and uls estimation with robust corrections in structural equation models with ordinal variables. *Psychological methods* **21**, 369.

Mair, P., Satorra, A. & Bentler, P. M. (2012). Generating Nonnormal Multivariate Data Using Copulas: Applications to SEM. *Multivariate Behavioral Research* **47**, 547–565.

Marcoulides, K. M. (2019). Reliability estimation in longitudinal studies using latent growth curve modeling. *Measurement: Interdisciplinary Research and Perspectives* **17**, 67–77.

Marcoulides, K. M., Foldnes, N. & Grønneberg, S. (2019). Assessing model fit in structural equation modeling using appropriate test statistics. *Structural Equation Modeling: A Multidisciplinary Journal* , 1–11.

Micceri, T. (1989). The unicorn, the normal curve, and other improbable creatures. *Psychological bulletin* **105**, 156.

Muthén, B. (1984). A general structural equation model with dichotomous, ordered categorical, and continuous latent variable indicators. *Psychometrika* **49**, 115–132.

Nagler, T. & Vatter, T. (2019). *rvinecopulib: High Performance Algorithms for Vine Copula Modeling*. R package version 0.5.1.1.0.

Nelsen, R. B. (2007). *An introduction to copulas*. Springer-Verlag.

Pearson, K. (1895). Contributions to the mathematical theory of evolution. ii. skew variation in homogeneous material. *Philosophical Transactions of the Royal Society of London. A* , 343–414.

Pornprasertmanit, S., Miller, P., Schoemann, A. & Jorgensen, T. D. (2020). *simsem: SIMulated Structural Equation Modeling*. R package version 0.5-15.

Qu, W., Liu, H. & Zhang, Z. (2019). A method of generating multivariate non-normal random numbers with desired multivariate skewness and kurtosis. *Behavior research methods* , 1–8.

Qu, W. & Zhang, Z. (2020). *mnonr: A Generator of Multivariate Non-Normal Random Numbers*. R package version 1.0.0.

Quiroga, A. M. (1994). *Studies of the polychoric correlation and other correlation measures for ordinal variables*. Ph.D. thesis, Uppsala University.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rhemtulla, M., Brosseau-Liard, P. É. & Savalei, V. (2012). When can categorical variables be treated as continuous? a comparison of robust continuous and categorical SEM estimation methods under suboptimal conditions. *Psychological methods* **17**, 354.

Rice, J. A. (2006). *Mathematical statistics and data analysis*. Cengage Learning.

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. *Journal of Statistical Software* **48**, 1–36.

Rüschendorf, L. (2009). On the distributional transform, Sklar's theorem, and the empirical copula process. *Journal of Statistical Planning and Inference* **139**, 3921–3927.

Ruscio, J. & Kaczetow, W. (2008). Simulating multivariate nonnormal data using an iterative algorithm. *Multivariate Behavioral Research* **43**, 355–381.

Satorra, A. & Bentler, P. (1988). Scaling corrections for statistics in covariance structure analysis (UCLA statistics series 2). *Los Angeles: University of California at Los Angeles, Department of Psychology* .

Schepsmeier, U., Stoeber, J., Brechmann, E. C., Graeler, B., Nagler, T. & Erhardt, T. (2018). *VineCopula: Statistical Inference of Vine Copulas*. R package version 2.1.8.

Shapiro, A. (1983). Asymptotic distribution theory in the analysis of covariance structures. *South African Statistical Journal* **17**, 33–81.

Shorack, G. R. & Wellner, J. A. (2009). *Empirical processes with applications to statistics*, vol. 59. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM). Originally published in 1986 by John Wiley & Sons Inc., New York.

Sklar, M. (1959). *Fonctions de repartition a n dimensions et leurs marges.* Université Paris 8.

Takane, Y. & De Leeuw, J. (1987). On the relationship between item response theory and factor analysis of discretized variables. *Psychometrika* **52**, 393–408.

Tarka, P. (2018). An overview of structural equation modeling: its beginnings, historical development, usefulness and controversies in the social sciences. *Quality & quantity* **52**, 313–354.

Touloumis, A. (2016). Simulating correlated binary and multinomial responses under marginal model specification: The simcormultres package. *The R Journal* **8**, 79–91.

Vale, C. D. & Maurelli, V. A. (1983). Simulating multivariate nonnormal distributions. *Psychometrika* **48**, 465–471.

Van De Schoot, R., Sijbrandij, M., Winter, S. D., Depaoli, S. & Vermunt, J. K. (2017). The grolts-checklist: guidelines for reporting on latent trajectory studies. *Structural Equation Modeling: A Multidisciplinary Journal* **24**, 451–467.

Wu, H. & Lin, J. (2016). A scaled F distribution as an approximation to the distribution of test statistics in covariance structure analysis. *Structural Equation Modeling: A Multidisciplinary Journal* **23**, 409–421.

Yan, J. et al. (2007). Enjoy the joy of copulas: with a package copula. *Journal of Statistical Software* **21**, 1–21.

Appendix B. Supplementary material: Complete R-code for examples

```
## Journal of Statistical Software
##
## covsim: An R package for simulating non-normal data for Structural
    ↪ Equation Models Using Copulas

## S. Gr?nneberg, N. Foldnes, K. Marcoulides
##
## August 2021

library("MASS")
library("lavaan")
library("copula")
library("rvinecopulib")
library("covsim")
library("discnorm")
library("ggExtra")
library("GGally")
library("psych")
library("VineCopula")

## ---- Figure 1 and 2---- ##

set.seed(1234)
n=1000
#Figs 1a and 2a
tmp <- rCopula(n, claytonCopula(param=3.4))
p <- ggplot(data.frame(x=tmp[, 1], y=tmp[, 2]), aes(x,y))+
  geom_point()+xlab(expr(U[1]))+ylab(expr(U[2]))
ggExtra::ggMarginal(p, type="histogram", bins=15)

tmp <- qnorm(tmp)# add N(0,1) marginals
p <- ggplot(data.frame(x=tmp[, 1], y=tmp[, 2]), aes(x,y))+
  geom_point()+xlab(expr(Z[1]))+ylab(expr(Z[2]))
ggExtra::ggMarginal(p, type="histogram", bins=15)

#Figs 1b and 2b
set.seed(1234)
n=1000
tmp = rCopula(n, normalCopula(param=0.8))# 14.1: corr 0.8 with chi-ssquare
    ↪ marginals
p <- ggplot(data.frame(x=tmp[, 1], y=tmp[, 2]), aes(x,y))+
  geom_point()+xlab(expr(U[1]))+ylab(expr(U[2]))
ggExtra::ggMarginal(p, type="histogram", bins=15)

tmp <- qnorm(tmp)# add N(0,1) marginals
p <- ggplot(data.frame(x=tmp[, 1], y=tmp[, 2]), aes(x,y))+
  geom_point()+xlab(expr(Z[1]))+ylab(expr(Z[2]))
ggExtra::ggMarginal(p, type="histogram", bins=15)
```

```
## --- VITA calibration for the clayton copula parameter 3.4 used above ---
    ↪ ##
mnorm <- list(list(distr="norm"),list(distr="norm"))
sigma.target <- matrix(c(1,0.8,0.8,1),2)
set.seed(1)
calibrated.vita <- vita(mnorm, sigma.target, family_set="clayton")
summary(calibrated.vita)
#simulate from vita and test accuracy
library(rvinecopulib)
cov(rvine(10^5, calibrated.vita))


## --- Fig 5: VITA calibration in the 3-dimensional growth curve
    ↪ illustration --- ##
sigma.target <- matrix(c(1,0.4,0.3, 0.4,1,0.4,0.3,0.4,1),3)

margins <- list(list(distr="norm"),
              list(distr="chisq", df=1),
              list(distr="t", df=5))
pcs <- list(list(bicop_dist("clayton"), bicop_dist("joe")), list(bicop_dist
    ↪ ("frank")))
vine_cop<- vinecop_dist(pcs, structure=dvine_structure(1:3))

#vita vector has these marginal variances:
margin.variances <- c(1,2,5/3)
#the covariance matrix to be attained by vita must be scaled, therefore
pre <- diag(sqrt(margin.variances/diag(sigma.target)))
vita.target <- pre %*% sigma.target %*% pre

set.seed(1)
calibrated.vita<- vita(margins, vita.target, vc=vine_cop, verbose=T)
# the vita vector must be post-multiplied to have unit variances
post <- diag(1/diag(pre))

vita.sample <- rvine(10^5, calibrated.vita)%*%post
round(cov(vita.sample)-sigma.target,3)# approximately equal

set.seed(1234)
GGally::ggpairs(data.frame(rvine(10^3, calibrated.vita)%*%post),
                   diag=list(continuous="barDiag"),
                   columnLabels = c("delta[1]", "delta[2]","delta[3]"),
                   labeller = "label_parsed")


## Fig 6 IG ##
## --- Subsection 3.1 on independent generator approach
set.seed(1234)
ig.sample<-rIG(N=10^3,sigma.target=sigma.target, reps=1,
             skewness=c(0,sqrt(8),0), excesskurtosis =c(0, 12, 6))

GGally::ggpairs(data.frame(ig.sample[[1]]),
              diag=list(continuous="barDiag"),
              columnLabels = c("delta[1]", "delta[2]","delta[3]"),
              labeller = "label_parsed")

## simulation for growth model
growth.mod <- "
i =~ 1*t1 + 1*t2 + 1*t3
s =~ 0*t1 + 1*t2 + 2*t3
i~~1*i; s~~1*s; i~~start(0)*s;
t1~~t2+t3;
t2~~t3;
t1~~c*t1; t2~~c*t2; t3~~c*t3;
"
set.seed(1)
n <- 1000; reps =1000
par <- 9; parvalue <- 0
sim.df <- replicate(reps, {
  errors <- data.frame(MASS::mvrnorm(n, mu=rep(0,3), Sigma=sigma.target))
  i <- rnorm(n); s <- rnorm(n)
  t1 <- i+errors[,1]
  t2 <- i+s+errors[, 2]
  t3 <- i+2*s+errors[,3]
  f <- sem(growth.mod, data.frame(t1,t2,t3))
  cover.norm <- (parameterestimates(f)[par, "ci.lower"]- parvalue)*
    (parameterestimates(f)[par, "ci.upper"]- parvalue) < 0
  errors <- data.frame(rvine(n, calibrated.vita)%*% post)
  i <- rnorm(n); s <- rnorm(n)
  t1 <- i+errors[,1]
  t2 <- i+s+errors[, 2]
  t3 <- i+2*s+errors[,3]
  f <- sem(growth.mod, data.frame(t1,t2,t3))
  cover.vita <- (parameterestimates(f)[par, "ci.lower"]- parvalue)*
    (parameterestimates(f)[par, "ci.upper"]- parvalue) <0
  c(cover.norm, cover.vita)
})

sim.df <- data.frame(t(sim.df))
colMeans(sim.df, na.rm=T )



## --- Fig 7: VITA calibration in the 6-dimensional growth curve
    ↪ illustration --- ##

residual.covariance <- toeplitz(1:6)
residual.covariance[residual.covariance > 3] <-0
residual.covariance[residual.covariance == 2] <- 0.5
residual.covariance[residual.covariance == 3] <- 0.2

margins.nonnorm <- list(list(distr="norm"), list(distr="chisq", df=5),
                     list(distr="chisq", df=4), list(distr="chisq", df=3),
                     list(distr="chisq", df=2), list(distr="chisq", df=1))
margins.norm <- list(list(distr="norm"), list(distr="norm"),
                     list(distr="norm"), list(distr="norm"),
                     list(distr="norm"), list(distr="norm"))
```

```
margin.variances <- c(1, 10, 8, 6, 4, 2)

sigma.target <- diag(sqrt(margin.variances))%*% residual.covariance%*%
  diag(sqrt(margin.variances))
#calibration
set.seed(1)
vita1 <- vita(margins.norm, residual.covariance, family_set="clayton")
set.seed(1)
vita2 <- vita(margins.nonnorm, sigma.target, family_set="gauss")
set.seed(1)
vita3 <- vita(margins.nonnorm, sigma.target, family_set="clayton")

# post multiplication matrix
pre <- diag(sqrt(margin.variances/diag(residual.covariance)))
post <- diag(1/diag(pre))
#approximately equal at N=10^5
round(cov(rvine(10^5, vita1))-residual.covariance,2)
round(cov(rvine(10^5, vita2)%*%post)-residual.covariance,2)
round(cov(rvine(10^5, vita3)%*%post)-residual.covariance,2)

# DATA GENERATOR
get_data <- function(n, cv){
    residuals <- rvine(n, cv, cores=parallel::detectCores())
    if(cv$margins[[6]]$distr != "norm")
      residuals <- residuals %*% post
    i <- rnorm(n); s <- rnorm(n)
    t1 <- i + residuals[,1]
    t2 <- i+s+ residuals[,2]
    t3 <- i+2*s+ residuals[,3]
    t4 <- i+3*s+ residuals[,4]
    t5 <- i+4*s+ residuals[,5]
    t6 <- i+5*s+ residuals[,6]
    data.frame(t1=t1, t2=t2, t3=t3, t4=t4, t5=t5, t6=t6)
}

growth.model.toeplitz <- "
i =~ 1*t1 + 1*t2 + 1*t3 + 1*t4 + 1*t5 + 1*t6
s =~ 0*t1 + 1*t2 + 2*t3 + 3*t4 + 4*t5 + 5*t6
i~~start(0)*s;
t1~~b*t2+c*t3; t2~~b*t3+c*t4;
t3~~b*t4+c*t5; t4~~b*t5+c*t6;
t5~~b*t6
t1~~a*t1;t2~~a*t2; t3~~a*t3; t4~~a*t4; t5~~a*t5; t6~~a*t6;
"

# ASYMPTOTIC POWER
set.seed(1)
big <- get_data(10^6, vita1 )
f <- sem(growth.model.toeplitz, big, estimator="MLM")
eigs1 <- Re(eigen(lavInspect(f, "UGamma"))$values)
library(CompQuadForm)
```

```
imhof(qchisq(0.95, 15), eigs1[1:15])$Qq# RR= 7.3%

set.seed(1)
big <- get_data(10^6, vita2 )
f <- sem(growth.model.toeplitz, big, estimator="MLM")
eigs2 <- Re(eigen(lavInspect(f, "UGamma"))$values)
imhof(qchisq(0.95, 15), eigs2[1:15])$Qq# RR= 29.4%

set.seed(1)
big <- get_data(10^6, vita3 )
f <- sem(growth.model.toeplitz, big, estimator="MLM")
eigs3 <- Re(eigen(lavInspect(f, "UGamma"))$values)
imhof(qchisq(0.95, 15), eigs3[1:15])$Qq# RR= 50.6%

#produce graphs
get_density <- function(egs,x){
  CDF.T <- NULL
  for (i in x) {
    CDF.T <- c(CDF.T, 1-imhof(i, egs)$Qq)
  }
  dens.val <- NULL
  for (i in 2:length(x)) {
    dens.val <- c(dens.val, CDF.T[i]- CDF.T[i-1])
  }
  dens.val <- dens.val/(x[2]-x[1])
}

Tvalues <- seq(0, 60, length.out = 500); x <- Tvalues[-1]
my.df <- data.frame(x=Tvalues[-1],
                    VITA1 = get_density(eigs1, Tvalues),
                    VITA2 = get_density(eigs2, Tvalues),
                    VITA3 = get_density(eigs3, Tvalues),
                    nominal = dchisq(x, df=15))

m <- reshape2::melt(my.df, id.vars="x");m$Distribution <- m$variable

ggplot(m, aes(x, value, color=Distribution, linetype=Distribution))+
    ↪ geom_line()+
  geom_vline(xintercept=qchisq(0.95, df=15))+
  xlab(expr("T"["NTML"]))+ylab("Density")+
  theme(legend.position = c(0.8,.6))

##MODERATE SAMPLE SIZE, n=500 SIMULATIONS, confirm the asymptotics
set.seed(1)
f1 <- replicate(1000, {sem(growth.model.toeplitz, data=get_data(500, vita1))
    ↪ })
f2 <- replicate(1000, {sem(growth.model.toeplitz, data=get_data(500, vita2))
    ↪ })
f3 <- replicate(1000, {sem(growth.model.toeplitz, data=get_data(500, vita3))
    ↪ })

#Test of fit
pval.df <- data.frame(p1=sapply(f1, function(f) fitmeasures(f, "pvalue")),
                    p2=sapply(f2, function(f) fitmeasures(f, "pvalue")),
                    p3=sapply(f3, function(f) fitmeasures(f, "pvalue")))

colMeans(pval.df <0.05)


## --- SECTION 5 --- ###
## --- running times of original code from 2017 vs vita() based on
    ↪ rvinecopulib --- ###
####
## Auxiliary functions needed for original VITA implementation in Gronneberg
    ↪  and Foldnes (2017)
#####

index.par <- function(A) {
  a <- A[1]
  b <- A[2]
  d <- dim(Matrix)[1]

  index.i <- NULL
  index.j <- NULL
  for(i in(1:d)) {
    if(Matrix[i,i] %in% c(a,b)) {
      ind <- Matrix[,i] %in% c(a,b)
      #cat("i = ", i, " result: ", ind, "\n")
      if (sum(ind) == 2) {
        index.i <- which(ind[-i] == TRUE)+1
        index.j <- i
        # print(c(index.i, index.j))
      }
    }
  }
  return(c(index.i,index.j))
}


parameter.valid = function (family, par.value1, par.value2){#check if
    ↪ parameter value is valid
  if ((family == 1 || family == 2) &&
      abs(par.value1) >= 1)
    return(FALSE)
  if (family == 2 && par.value2 <= 2)
    return(FALSE)
  if ((family == 3 || family == 13) &&
      par.value1 <= 0)
    return(FALSE)
  if ((family == 4 || family == 14) &&
      par.value1 < 1)
    return(FALSE)
  if ((family == 6 || family == 16) &&
      par.value1 <= 1)
    return(FALSE)
  if (family == 5 && par.value1 == 0)
```

```
    return(FALSE)
  if ((family == 7 || family == 17) &&
      par.value1 <= 0)
    return(FALSE)
  if ((family == 7 || family == 17) &&
      par.value2 < 1)
    return(FALSE)
  if ((family == 8 || family == 18) &&
      par.value1 <= 0)
    return(FALSE)
  if ((family == 8 || family == 18) &&
      par.value2 < 1)
    return(FALSE)
  if ((family == 9 || family == 19) &&
      par.value1 < 1)
    return(FALSE)
  if ((family == 9 || family == 19) &&
      par.value2 <= 0)
    return(FALSE)
  if ((family == 10 || family == 20) &&
      par.value1 < 1)
    return(FALSE)
  if ((family == 10 || family == 20) &&
      (par.value2 <= 0 || par.value2 > 1))
    return(FALSE)
  if ((family == 23 || family == 33) &&
      par.value1 >= 0)
    return(FALSE)
  if ((family == 24 || family == 34) &&
      par.value1 > -1)
    return(FALSE)
  if ((family == 26 || family == 36) &&
      par.value1 >= -1)
    return(FALSE)
  if ((family == 27 || family == 37) &&
      par.value1 >= 0)
    return(FALSE)
  if ((family == 27 || family == 37) &&
      par.value2 > -1)
    return(FALSE)
  if ((family == 28 || family == 38) &&
      par.value1 >= 0)
    return(FALSE)
  if ((family == 28 || family == 38) &&
      par.value2 > -1)
    return(FALSE)
  if ((family == 29 || family == 39) &&
      par.value1 > -1)
    return(FALSE)
  if ((family == 29 || family == 39) &&
      par.value2 >= 0)
    return(FALSE)
  if ((family == 30 || family == 40) &&
      par.value1 > -1)
    return(FALSE)
  if ((family == 30 || family == 40) &&
      (par.value2 >= 0 || par.value2 < (-1)))
    return(FALSE)
  if ((family == 104 || family == 114 ||
       family == 204 || family == 214) &&
      par.value1 < 1)
    return(FALSE)
  if ((family == 104 || family == 114 ||
       family == 204 || family == 214) &&
      (par.value2 < 0 || par.value2 > 1))
    return(FALSE)
  if ((family == 124 || family == 134 ||
       family == 224 || family == 234) &&
      par.value1 > -1)
    return(FALSE)
  if ((family == 124 || family == 134 ||
       family == 224 || family == 234) &&
      (par.value2 < 0 || par.value2 > 1))
    return(FALSE)
  return(TRUE)
}


### This function identifies the appropriate sub-vine and uses the R-vine
    ↪ array representation from the VineCopula package.
create.submatrix <- function(I) {
  # I <- unique(I)
  d <- dim(Matrix)[1]
  l <- length(I)

  for(i in (d:1)) {
    if(sum(Matrix[,i] %in% I) == l)
      break
  }
  sub.matrix <- Matrix[(i:d),(i:d)]
  sub.par <- par[(i:d),(i:d)]
  sub.family <- family[(i:d),(i:d)]

  remove.ind <- which(!(diag(sub.matrix) %in% I))
  if(length(remove.ind) != 0) {
    sub.par <- sub.par[,-remove.ind]
    sub.family <- sub.family[,-remove.ind]
    sub.matrix <- sub.matrix[,-remove.ind]
  }

  new.sub.matrix <- NULL
  new.sub.par <- NULL
  new.sub.family <- NULL

  for(i in (1:(dim(sub.matrix)[2]))) {
    indx <- which(sub.matrix[,i] %in% I)
    tmp.mat <- sub.matrix[indx,i]
    new.sub.matrix <- cbind(new.sub.matrix, c(rep(0, length(I)-length(tmp.
        ↪ mat)), tmp.mat))

    tmp.par <- sub.par[indx,i]
    new.sub.par <- cbind(new.sub.par, c(rep(0, length(I)-length(tmp.mat)),
        ↪ tmp.par))

    tmp.family <- sub.family[indx,i]
    new.sub.family <- cbind(new.sub.family, c(rep(0, length(I)-length(tmp.
        ↪ mat)), tmp.family))
  }

  sub.matrix <- new.sub.matrix
  sub.par <- new.sub.par
  sub.family <- new.sub.family

  ## add zeros to diagonals on par and family:
  diag(sub.par) <- rep(0, l)
  diag(sub.family) <- rep(0, l)

  return(list(sub.matrix=sub.matrix, sub.par=sub.par, sub.family=sub.family)
      ↪ )
}


##############
# Specify parametric R-vine in five dimensions.
#############

Matrix <- c(5, 2, 3, 1, 4,
           0, 2, 3, 4, 1,
           0, 0, 3, 4, 1,
           0, 0, 0, 4, 1,
           0, 0, 0, 0, 1)
Matrix <- matrix(Matrix, 5, 5)

family <- c(0, 3, 3, 3, 3,
           0, 0, 3, 3, 3,
           0, 0, 0, 3, 3,
           0, 0, 0, 0, 3,
           0, 0, 0, 0, 0) #3=Clayton copula

family <- matrix(family, 5, 5)

par=family #parameter matrix for copulas. VITA fills in parameters
    ↪ sequentially in this matrix

RVM = RVineMatrix(Matrix, family, par) #the R vine specification.

#define an order in which to run through the vine structure, edge by edge
pair.index <- NULL
```

```
index.set <- list()
d <- dim(Matrix)[1]
for(i in ((d-1):1)) {
  for(j in (d:(i+1))) {
    cond.index <- NULL

    pair.index <- c(Matrix[i,i], Matrix[j,i])
    if(j+1<=d) {
      cond.index <- Matrix[((j+1):d),i]
    }
    index.set <- c(index.set, list(pair.index=pair.index, cond.index=cond.
        ↪ index))
  }
}

###############
# Define target covariance matrix
###############

model.true <- '
# latent variables
F1 =~ load*x1 +load* x2
F2 =~ load*y1 + load*y2+load*y3

x1 ~~ resvar *x1
x2 ~~ resvar *x2
y1 ~~ resvar *y1
y2 ~~ resvar *y2
y3 ~~ resvar *y3

F1 ~~ phi *F2
F1 ~~ 1 *F1
F2 ~~ 1 *F2
'

load=0.95
resvar=1-load^2
phi=.9

model.true <- gsub("resvar", as.character(resvar), model.true, fixed=TRUE)
model.true <- gsub("load", as.character(load), model.true, fixed=TRUE)
model.true <- gsub("phi", as.character(phi), model.true, fixed=TRUE)


f=cfa(model.true, data=NULL)
sigma.target = fitted(f)$cov


########
## Original VITA for given sigma.target and standard normal marginals
#######
```

```
# the numerical routine that searches for copula parameter for a given edge=
    ↪ pair.index in the vine
solve.param <- function(pair.index, cond.index) {
  I <- c(pair.index,cond.index)

  curr.family = family[index.par(pair.index)[1], index.par(pair.index)[2]]
  cat(" family: ", curr.family, "\n")

  sub.matrices <- create.submatrix(I) #the sub-vine we need for simulation

  ## Rename the RVineMatrix, to get the names from 1 and up:
  sub.matrix <- sub.matrices$sub.matrix
  recoded.index <- sort(unique(as.vector(sub.matrix)))[-1]
  recoded.sub <- as.vector(sub.matrix)
  recoded.pair.index <- pair.index
  for(i in (1:length(recoded.index))) {
    recoded.sub[which(recoded.sub==recoded.index[i])] <- i
    recoded.pair.index[which(recoded.pair.index==recoded.index[i])] <- i
  }
  sub.matrix <- matrix(recoded.sub, dim(sub.matrix)[1], dim(sub.matrix)[2])

  RVM.sub <- RVineMatrix(Matrix=sub.matrix, par=sub.matrices$sub.par, par2=
      ↪ sub.matrices$sub.par*0, family=sub.matrices$sub.family)

  root.function <- function(theta) { # this function returns the difference
      ↪ between target covariance and vine-implied covariance for theta
    #cat(theta, "\n")
    if(!parameter.valid(curr.family, theta, NA)){
      cat("not valid parameter: ", theta , "\n")
      return(10^3)
    }

    RVM.sub$par[2, 1] <<- theta
    simdata <- RVineSim(N,RVM.sub)[,recoded.pair.index] #Simulate large
        ↪ sample of size N

    return(sigma.target[pair.index[1], pair.index[2]]-cov(qnorm(simdata[,1])
        ↪ , qnorm(simdata[,2])))#standard normal marginals
  }

  est <- uniroot(root.function, lower=0.001, upper=20)$root
  cat("Approximation:", est, "\n")
  par[index.par(pair.index)[1], index.par(pair.index)[2]] <<- est #replace
      ↪ VITA result into global vine structure

}


#Original implementation timing
N=10^6
set.seed(1)
system.time({
  for(i in (1:(length(index.set)/2))) {
    pair.index <- index.set[[2*i-1]]
    cond.index <- index.set[[2*i]]
    cat("Optimizing for:", pair.index, "|", cond.index)

    solve.param(pair.index, cond.index)
  }
})
RVM=RVineMatrix(Matrix, family, par)

#Timing of covsim implementation of VITA

bicop <- bicop_dist("clayton")
pcs <- list(list(bicop, bicop, bicop, bicop),
            list(bicop, bicop, bicop),
            list(bicop, bicop), list(bicop))

# set up vine copula model with Gaussian margins
tmat <- Matrix[5:1, ]
vc <- vine_dist(list(distr = "norm"), pcs, tmat)
marginsnorm <- lapply(X=sqrt(diag(sigma.target)),function(X) list(distr="
    ↪ norm", sd=X) )

set.seed(1)
system.time(calvita <- vita(marginsnorm, sigma.target, vc=vc, family_set="
    ↪ clayton"))

## Simulation runtimes at n=1000
system.time(replicate(10000, qnorm(RVineSim(10^3, RVM))))123.364
system.time(replicate(10000, rvine(10^3, calvita)))34.186




## --- SECTION 6 --- ###
## --- Table 1 --- ###

test_func <- function(d){
  set.seed(1)
  ncores <- parallel::detectCores()
  marginsnorm <- lapply(X=rep(1,d),function(X) list(distr="norm", sd=X) )
  mat <- diag(d)+0.3
  diag(mat) <- 1
  time.cal <- tryCatch(system.time(vita <- vita(marginsnorm, mat)), error=
      ↪ function(w) {-1},
                    warning=function(w) {-2})
  if(length(time.cal) == 5){
    time.sim <- tryCatch(system.time(replicate(10^3, rvine(500, vita))),
                      error=function(w) {-1},
                      warning=function(w) {-2})
  }

  return(list(time.cal=time.cal, time.sim=time.sim, vita=vita, ncores=ncores
      ↪ ))
```

```
}

#RAN FOLLOWING CODE. With 40 dimension it took too long. 15++ hours.
for(d in c(5, 10, 15)){#}, 20, 25, 30, 40, 50)){
  res <- test_func(d)
  print(res)
}

## --- SECTION 6 --- ##

## --- Continuous SEM data example --- ##

sem.pop <- '
        Ksi1 =~ start(.8)*x1+start(.7)*x2+start(.6)*x3+start(.5)*x4
        Ksi2 =~ start(.8)*x5+start(.7)*x6+start(.6)*x7+start(.5)*x8
        Eta1 =~ start(.8)*y1+start(.7)*y2+start(.6)*y3+start(.5)*y4
        Eta2 =~ start(.8)*y5+start(.7)*y6+start(.6)*y7+start(.5)*y8
        Eta3 =~ start(.8)*y9+start(.7)*y10+start(.6)*y11+start(.5)*y12
        Eta1 ~ start(.4)*Ksi1+start(.6)*Ksi2
        Eta2 ~ start(.4)*Ksi1+start(.2)*Ksi2+start(.3)*Eta1
        Eta3 ~ start(.1)*Ksi1+start(.1)*Ksi2+start(.2)*Eta1+start(.5)*Eta2
        Ksi1~~start(.3)*Ksi2; Eta1~~start(.5)*Eta1; Eta2~~start(.5)*Eta2
        Eta3~~start(.5)*Eta3; x1~~start(.5)*x1; x2~~start(.5)*x2; x3~~start
            ↪ (.5)*x3
        x4~~start(.5)*x4; x5~~start(.5)*x5; x6~~start(.5)*x6; x7~~start(.5)*
            ↪ x7
        x8~~start(.5)*x8; y1~~start(.5)*y1; y2~~start(.5)*y2; y3~~start(.5) *
            ↪ y3
        y4~~start(.5)*y4; y5~~start(.5)*y5; y6~~start(.5)*y6; y7~~start(.5) *
            ↪ y7
        y8~~start(.5)*y8; y9~~start(.5)*y9; y10~~start(.5)*y10
        y11~~start(.5)*y11; y12~~start(.5)*y12'

sigma.target <- lavInspect(sem(sem.pop, data=NULL), "sigma.hat")

marginsnorm <- lapply(X=sqrt(diag(sigma.target)),
                    function(X) list(distr="norm", sd=sqrt(X)) )
vitadist <- vita(marginsnorm, sigma.target)
randomsamples <- replicate(10^3, rvine(10^3, vitadist))



## --- Ordinal data simulation --- ##

sigma.target <- matrix(c(1,0.5,0.5,1),2)
set.seed(1)
vita_clayton <- vita(list(list(distr="norm"), list(distr="norm")),sigma.
    ↪ target, family_set="clayton")
set.seed(1)
vita_joe <- vita(list(list(distr="norm"), list(distr="norm")),sigma.target,
    ↪ family_set="joe")

#generate data and discretize
```

```
clayton.disc <- apply(rvine(10^3, vita_clayton), 2, cut, breaks=c(-Inf,0, 1,
    ↪  Inf),labels=FALSE)


## --- Run bootstrap test --- ##
bootTest(clayton.disc)
```

Department of Economics, BI Norwegian Business School, Oslo, Norway 0484
Email address: steffeng@gmail.com

Department of Economics, BI Norwegian Business School, Stavanger, Norway 4014
Email address: njal.foldnes@bi.no

Department of Psychology, University of Minnesota, Elliott Hall 75 East River Rd., Minneapolis, MN 55455, USA
Email address: kmarcoul@umn.edu