Vrije Universiteit Amsterdam

Master Thesis

Revisiting Vine Copula Imputation

Author: Neel Shah (nsh580)

1st supervisor: dr. Paulo Jorge de Andrade Serra
2nd reader: dr. Eduard Belitser

A thesis submitted in fulfillment of the requirements for the VU Master of Science degree in Mathematics

November 28, 2022

Contents

1 Introduction

Dealing with missing values is an important consideration in data analysis. Imputation (filling in the missing values) can improve the quality of statistical estimates and reduce the burden of data collection. Recently, Zhao and Udell [13] developed a parsimonious imputation method by fitting Gaussian Copulas over incomplete datasets.

Gaussian Copulas can be viewed as special cases of Vine Copulas [1], which model multivariate distributions as hierarchies of bivariate distribution functions. Because Vine Copulas are inherently appealing as a model for high-dimensional data, they have already been studied in an imputation setting [12, 10, 3]. However, existing methods have limited practical applicability because they are not optimized for imputation under a fixed time budget.

For continuous-data matrices, we want to study whether Vine Copula imputation procedures provide enough of a gain in accuracy relative to time spent compared to simpler methods. In this thesis, we compare Vine Copula imputation [12, 10] to Gaussian Copula imputation, which is benchmarked against other popular methods in [13]. To facilitate the comparison, we re-implement the Vine-based imputation algorithms using a modern Vine Copula package [7].

The comparison leads to two key questions: (1) What data-generating processes favor Vine Copulas for imputation accuracy? (2) What properties of the data challenge the computational efficiency of Vine Copula imputation? For the first, we created conditions that are expected to favor Vine Copula imputation, e.g. we consider incomplete datasets from hierarchical, non-elliptically distributed and/or high-dimensional models. And, for the second, we profiled the algorithmic bottlenecks in Vine Copula imputation by varying dimension, missingness rate and missingness structure.

This document is organized as follows: Section 2 collects the minimum preliminary material necessary to understand Copula-based imputation, Section 3 reproduces results from selected research articles and forms a basis of comparison, Section 4 assesses Vine Copula imputation against Gaussian Copula imputation with experiments on simulated data. Finally, we answer the two key questions in Section 5.

2 Background

Section 2.1 defines a formal framework to study statistical inference over incomplete datasets. Section 2.2 introduces two complementary sampling techniques: the Inverse Transform and the Rosenblatt Transform, which are used to sample from distribution functions. Section 2.3 derives bivariate copulas from first principles and relates them to distribution functions via Sklar's Theorem. Then it introduces conditional copulas and shows how they can be used to decompose multivariate joint distribution functions into products of bivariate conditional copulas. This leads to Regular (R-)Vine distributions, which are central to Vine Copula imputation.

2.1 Missing Data Analysis

In this sub-section, we introduce three concepts: missingness pattern, mechanism and structure. Missingness patterns group units with exactly the same missing variables. Missingness mechanism describes how the availability of a variable depends on other variables. Missingness structures are labels for collections of missingness patterns.

Missingness pattern

For $n, d \in \mathbb{Z}$, let $\mathrm{X} \in \mathbb{R}^{n\times d}$ be a data matrix with one-or-more missing entries.
The matrix X corresponds to n observations of a d-dim random vector $\boldsymbol{X} = (X_1, \dots, X_d)$. Columns of X are random variables and denoted as $X_j,\ j \in \{1, \dots, d\}$. Rows of X are units and denoted as $\boldsymbol{x}_i,\ i \in \{1, \dots, n\}$.

Definition 2.1 (Complete case matrix, Chapter 1 [6])**.** The complete case matrix of X is the maximal submatrix of X such that no row has a missing value in any column. □

Let $\mathrm{M} = (m_{ij}) \in \{0, 1\}^{n\times d}$ be the missingness indicator matrix corresponding to X, with $m_{ij} = 1$ if x_{ij} missing and $m_{ij} = 0$ otherwise.

Definition 2.2 (Missingness pattern, Chapter 1 [6])**.** Two rows $\boldsymbol{x}_i, \boldsymbol{x}_j$ have the same missingness pattern when $m_{ik} = 1 \iff m_{jk} = 1,\ \forall k \in \{1, \dots, d\}$. □

Missingness patterns are indexed by a subset of columns $S \subset \{1, \dots, d\}$, which distinguishes a *unique* subset of rows $I_S = \{i \in \{1, \dots, n\} \mid m_{ik} = 1 \text{ if } k \in S \text{ o.w. } m_{ik} = 0\}$ that are missing entries for exactly the columns in S.

Definition 2.3 (Available case matrix, Chapter 1 [6])**.** The available case matrix of X with respect to missingness pattern S is the complete case matrix of $\mathrm{X}_{S^C} = (X_j), j \in \{1, \dots, d\} \setminus S$. □

Example 2.1. Given 3×3 data matrix X and missingness indicator matrix M

$$\mathrm{X} = \begin{bmatrix} 1 & 0.5 & 100 \\ 2 & 0.25 & - \\ 3 & - & - \end{bmatrix} \qquad \mathrm{M} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

For missingness pattern $S = \{3\}$, $I_S = \{2\}$ and $\mathrm{X}_{S^C} = \begin{bmatrix} 1 & 0.5 \\ 2 & 0.25 \\ 3 & - \end{bmatrix}$

Complete case matrix $\overline{X}$ and available case matrix $\overline{\mathrm{X}}_{S^C}$ are

$$\overline{\mathrm{X}} = \begin{bmatrix} 1 & 0.5 & 100 \end{bmatrix} \quad \overline{\mathrm{X}}_{S^C} = \begin{bmatrix} 1 & 0.5 \\ 2 & 0.25 \end{bmatrix}$$

Observe that row 3 has missingness pattern $\{2, 3\}$ and not missingness pattern $\{3\}$. □

Missingness mechanism

Assume that the rows of X and M, $(\boldsymbol{x}_i, \boldsymbol{m}_i)$ are iid over i. Let $f_{M|X}(\boldsymbol{m}_i|\boldsymbol{x}_i, \phi)$ be the conditional distribution of $\boldsymbol{m}_i$ given $\boldsymbol{x}_i$ with unknown parameters ϕ.

Definition 2.4 (MCAR missingness mechanism, Chapter 1 [6])**.** If the missingness does not depend on observed or missing values of the data then it is said to be Missing Completely at Random (MCAR), which is implied by

$$f_{\boldsymbol{M}|\boldsymbol{X}}(\boldsymbol{m}_i|\boldsymbol{x}_i, \phi) = f_{\boldsymbol{M}|\boldsymbol{X}}(\boldsymbol{m}_i|\boldsymbol{x}_i^*, \phi)$$

for any distinct members $\boldsymbol{x}_i, \boldsymbol{x}_i^*$ of the sample space of $\boldsymbol{X}$. □

Under the MCAR mechanism, since the probability that a variable is missing is the same for all units, statistical estimates over complete cases will be unbiased relative to the same estimates supposing no incomplete cases.
Let $\boldsymbol{x}_{(0)i}$ denote the observed variables of unit $\boldsymbol{x}_i$ and $\boldsymbol{x}_{(1)i}$ the unobserved.

Definition 2.5 (MAR missingness mechanism, Chapter 1 [6])**.** If the missingness does not depend on the unobserved variables, but possibly on the observed, then it is said to be Missing at Random (MAR), which is implied by

$$f_{\boldsymbol{M}|\boldsymbol{X}}(\boldsymbol{m}_i|\boldsymbol{x}_{(0)i}, \boldsymbol{x}_{(1)i}, \phi) = f_{\boldsymbol{M}|\boldsymbol{X}}(\boldsymbol{m}_i|\boldsymbol{x}_{(0)i}, \boldsymbol{x}_{(1)i}^*, \phi)$$

for any distinct members $\boldsymbol{x}_{(1)i}, \boldsymbol{x}_{(1)i}^*$ of the sample space of $\boldsymbol{x}_{(1)i}$. □

Let $f_{\boldsymbol{X}_{(1)}|\boldsymbol{X}_{(0)}}$ denote the conditional distribution of unobserved variables $\boldsymbol{X}_{(1)} = (X_i)_{i \in S}$ given observed variables $\boldsymbol{X}_{(0)} = (X_i)_{i \in S^C}$. Under both MCAR and MAR mechanisms, $f_{\boldsymbol{X}_{(1)}|\boldsymbol{X}_{(0)}}$ is independent of missingness pattern S. Thus, we can sample from conditional distributions to fill in any missing values.

Example 2.2. In a survey, for a question, if the rate of non-response depends only on the participant's gender then the missingness mechanism cannot be MCAR. Furthermore, if gender is always observed when there is non-response then the missingness mechanism is MAR. □

Missingness structure

When a data matrix has has exactly one variable with missing values, it is said to have *univariate* missingness structure. Collections of missingness patterns can also form a *monotone* missingness structure.

Definition 2.6 (Monotone missingness structure, 4.1.1 [11])**.** The data has a monotone missingness structure if the variables can be ordered such that for all rows with missing values in variable $X_j, j \in \{1, \dots, d\}$, variables $X_k, \forall k > j$ are also missing. □

From Definition 2.6 it can be seen that *univariate* is a special case of *monotone*. All other missingness structures are labeled as *general*.

Example 2.3. X has missingness patterns $\{\{3\}, \{2, 3\}\}$, which lends it a monotone missingness structure.

$$\mathrm{X} = \begin{bmatrix} 1 & 0.5 & 100 \\ 2 & 0.25 & - \\ 3 & - & - \end{bmatrix} \quad \mathrm{Y} = \begin{bmatrix} - & 0.5 & 100 \\ 2 & 0.25 & - \\ 3 & - & - \end{bmatrix}$$

whereas Y has a general missingness structure, with missingness patterns $\{\{1\}, \{3\}, \{2, 3\}\}$. □

2.2 Sampling Techniques

Once we fit a copula model, we have access to the distribution function governing the data. If we can sample from this distribution function, then we can impute missing values. The Inverse Transform generates iid samples from a univariate distribution function and is used iteratively in the Rosenblatt Transform. The Inverse Rosenblatt transform generates samples from a d-dim distribution function and can be applied on partially observed units.

Inverse Transform

Definition 2.7 (Generalized Inverse, Definition 2.3 [9])**.** For a non-decreasing function F on $\mathbb{R}$, the generalized inverse of F, F^-, is the function defined by

$$F^-(u) = \inf\{x : F(x) \geq u\}$$

□

Lemma 2.1 (Inverse Transform, Lemma 2.4 [9])**.** If $U \sim \text{Uniform}[0, 1]$ then the random variable $F^-(U) \sim F$. □

Example 2.4. If $X \sim F$ is an univariate continuous random variable and $u_1, u_2, \dots$ are iid samples of $U \sim \text{Uniform}[0, 1]$ then $x_i = F^-(u_i),\ i = 1, 2, \dots$ are iid samples of X. □

Rosenblatt Transform

For the following, let $\boldsymbol{x} = (x_1, \dots, x_d) \sim F_{1:d}$ be a d-dim random vector. Let $F_{j|1:(j-1)}, j = 2, \dots, d$ be the associated conditional distributions functions.

Theorem 2.1 (Rosenblatt Transform, Theorem 6.1 [2])**.** If we compute

$$\begin{aligned}
u_1 &= F_1(x_1) \\
u_2 &= F_{2|1}(x_2|x_1) \\
&\vdots \\
u_d &= F_{d|1:(d-1)}(x_d|x_1, \dots, x_{d-1})
\end{aligned}$$

then u_i are iid Uniform$[0, 1]$ distributed for $i = 1, \dots, d$. □

We sample from a d-dim random vector when we apply the transform in reverse.

Example 2.5 (Inverse Rosenblatt Transform)**.** Let $u_i \sim$ Uniform$[0, 1], i = 1, \dots, d$ be iid samples. If we compute

$$\begin{aligned}
x_1 &= F_1^{-}(u_1) \\
x_2 &= F_{2|1}^{-}(u_2|x_1) \\
&\vdots \\
x_d &= F_{d|1:(d-1)}^{-}(u_d|x_1, \dots, x_{d-1})
\end{aligned}$$

then $(x_1, \dots, x_d)$ is a random sample from $F_{1:d}$. □

Sampling with Partial Data

On the surface, sampling with partial data seems trivial. For a unit, if $S \subset \{1, \dots, d\}$ observations are missing, then generate $|S|$ uniform random samples and apply the Inverse Rosenblatt Transform (Example 2.5). During the Inverse Rosenblatt Transform, substitute observed coordinates $x_j,\ j \in \{1, \dots, d\} \setminus S$ and simulate missing coordinates $x_k \sim F_{k:1:(k-1)}^{-}(u_k|x_1, \dots, x_{k-1}),\ k \in S$ to obtain a complete $(x_1, \dots, x_d)$ random sample.
However, when working conditional distributions expressed in terms of copulas, the arguments, here $x_1, \dots, x_d$, are non-trivial to calculate. We encounter this difficulty later in Example 2.11 and continually as we discuss Vine Copula imputation algorithms in Section 3.2.

2.3 Copula Theory

In this sub-section, we cover the minimal theory required to understand Copula based imputation. Key results are Copula (Definition 2.11), Sklar's theorem (Theorem 2.4), Gaussian copula (Example 2.7), Conditional copulas (Lemma 2.2, Definition 2.16), Regular (R-)Vine distributions (Definition 2.21) and the Recursion formula (Theorem 2.7).

Bivariate Copulas

Let $C : [0, 1]^2 \to [0, 1]$.

Definition 2.8 (C volume of rectangle B, $V_C(B)$, Definition 2.1.1 [8])**.** Rectangle $B = [x_1, x_2] \times [y_1, y_2] \subseteq [0,1]^2$ has C volume

$$V_C(B) = C(x_2, y_2) - C(x_2, y_1) - C(x_1, y_2) + C(x_1, y_1)$$

□

Definition 2.9 (2-increasing, Definition 2.1.2 [8])**.** C is 2-increasing if

$$V_C(B) \geq 0$$

for all rectangles $B \subseteq [0,1]^2$. □

Definition 2.10 (Grounded, Chapter 2 [8])**.** C is grounded when

$$C(x, 0) = 0 = C(0, y)$$

for all $(x, y) \in [0,1]^2$. □

Definition 2.11 (Copula, Definition 2.2.1-2 [8])**.** C is a copula if C is 2-increasing, grounded and satisfies

$$C(x, 1) = x \quad \text{and} \quad C(1, y) = y$$

for all $(x, y) \in [0,1]$. □

Example 2.6. For $(x, y) \in [0,1]^2$, the following are copulas

$$\Pi(x,y) = xy \qquad M(x,y) = \min(x,y) \qquad W(x,y) = \max(x+y-1, 0)$$

□

Theorem 2.2 (Derivative of copula, Theorem 2.2.7 [8])**.** For any $x \in [0,1]$, $\partial C(x,y)/\partial x$ exists for almost all y and

$$0 \leq \frac{\partial}{\partial x} C(x,y) \leq 1$$

almost surely. The equivalent statement can be made about $\partial C(x,y)/\partial y$. □

Theorem 2.3 (Copula density, Theorem 2.2.8 [8])**.** For all $(x, y) \in [0,1]^2$, the copula density

$$c(x,y) = \frac{\partial}{\partial x \partial y} C(x,y)$$

exists and $c(x,y) \in [0,1]$ almost surely. □

Sklar's Theorem

Define $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$ and let $F, H : \overline{\mathbb{R}} \to [0,1]$.

Definition 2.12 (Distribution function, Definition 2.3.1 [8])**.** $F(x)$ is a distribution function if it is is non-decreasing, $F(-\infty) = 0$ and $F(\infty) = 1$. □

Definition 2.13 (Joint distribution function, Definition 2.3.2 [8])**.** $H : \overline{\mathbb{R}}^2 \to \mathbb{R}$ is a joint distribution function if it is 2-increasing, $H(x, -\infty) = 0 = H(-\infty, y)$ and $H(\infty, \infty) = 1$.
H has marginal distributions $F(x) = H(x, \infty)$ and $G(y) = H(\infty, y)$. □

Copulas can be interpreted in two equivalent ways: (1) functions that link multivariate joint distributions to univariate marginal distributions, or (2) multivariate distributions on the unit hypercube with uniform $[0, 1]$ univariate marginals.

Theorem 2.4 (Sklar's Theorem, Theorem 1.9 [2])**.** Let $\boldsymbol{X} = (X_1, X_2)$ be a random vector with joint distribution F and marginals F_1, F_2.
The joint distribution function can be expressed as

$$F(x_1, x_2) = C(F_1(x_1), F_2(x_2))$$

with associated density or probability mass function

$$f(x_1, x_2) = c(F_1(x_1), F_2(x_2))f_1(x_1)f(x_2)$$

for some copula C with copula density c.
Conversely, the copula corresponding to multivariate distribution function F with marginals F_1, F_2 can be expressed as

$$C(u_1, u_2) = F(F_1^{-1}(u_1), F_2^{-1}(u_2))$$

and its copula density or probability mass function is determined by

$$c(u_1, u_2) = \frac{f(F_1^{-1}(u_1), F_2^{-1}(u_2))}{f_1(F_1^{-1}(u_1))f_2(F_2^{-1}(u_2))}$$

□

Example 2.7 (Gaussian copula, Example 1.11 in [2])**.** The bivariate Gaussian copula is defined as

$$C(u_1, u_2; \rho) = \Phi_2(\Phi^{-1}(u_1), \Phi^{-1}(u_2); \rho)$$

where $\Phi(\cdot)$ is the standard normal distribution function and $\Phi(\cdot, \cdot; \rho)$ is the bivariate normal distribution function with zero mean, unit variance and correlation ρ.
The copula density is

$$c(u_1, u_2; \rho) = \frac{1}{\phi(x_1)\phi(x_2)} \frac{1}{\sqrt{1-\rho^2}} \exp\left\{ -\frac{\rho^2(x_1^2 + x_2^2) - 2\rho x_1 x_2}{2(1-\rho^2)} \right\}$$

where $x_i := \Phi^{-1}(u_i),\ i = 1, 2$ and ϕ is the standard normal density. □

Conditional Copulas

Lemma 2.2 (Conditional distributions and densities, Lemma 1.15 [2])**.** For $\boldsymbol{X}$ and accompanying quantities as in Sklar's Theorem, the conditional distribution function is

$$F_{1|2}(x_1|x_2) = \frac{\partial}{\partial F_2(x_2)} C_{12}(F_1(x_1), F_2(x_2))$$

where $C_{12}(u_1, u_2)$ is a copula from Theorem 2.4, and the conditional density is

$$f_{1|2}(x_1|x_2) = c_{12}(F_1(x_1), F_2(x_2)) f_1(x_2)$$

where $c_{12}(u_1, u_2)$ is the associated copula density.
Conversely,

$$C_{1|2}(F_1(x_1)|F_2(x_2)) = \frac{\partial}{\partial F_2(x_2)} C_{12}(F_1(x_1), F_2(x_2))$$

and

$$F^{-1}_{1|2}(u_1|x_2) = F_1^{-1}(C^{-1}_{1|2}(u_1|F_2(x_2)))$$

Equivalent formulas for $F_{2|1}, f_{2|1}, C_{2|1}$ and $F^{-1}_{2|1}$ follow naturally. □

Definition 2.14 (h-functions of bivariate copulas, Definition 1.16 [2])**.** The conditional distribution function from Lemma 2.2 are also known as h-functions corresponding to copula C_{12}

$$h_{1|2}(u_1|u_2) = \frac{\partial}{\partial u_2} C_{12}(u_1, u_2)$$
$$h_{2|1}(u_2|u_1) = \frac{\partial}{\partial u_1} C_{12}(u_1, u_2)$$

for all $(u_1, u_2) \in [0, 1]^2$ □

Algorithm 2.1: Simulation from a Copula, Chapter 2.9 [8]

```
input  : Copula $C_{12}$, $u_1, v_2$ independent uniform $[0,1]$ samples
output: $(u_1, u_2)$ random sample from $C_{12}$
set $u_2 \leftarrow h^{-1}_{2|1}(v_2|u_1)$
```

Higher-dimensional Copulas

Let $\boldsymbol{X} = (X_1, \ldots, X_d)$ be a d-dim random variable. Sklar's Theorem gives an expression for $\boldsymbol{X}$'s joint density. The joint density can be decomposed into products of conditional densities. Lemma 2.2 relates conditional densities and copula conditional densities. Thus, we can represent $\boldsymbol{X}$'s joint density using only bivariate copulas. We illustrate this for $d = 3$ shortly.
Let $\boldsymbol{x}_{1:j}$ denote the vector with components $x_1, \ldots, x_j$.

Definition 2.15 (Joint density decomposition, Equation 4.16 [2])**.** Every joint density can be factorized into a product of conditionals

$$\begin{aligned} f_{1:d}(\boldsymbol{x}_{1:d}) &= f_{d|1:(d-1)}(x_d|\boldsymbol{x}_{1:(d-1)})f_{1:(d-1)}(\boldsymbol{x}_{1:(d-1)}) \\ &= f_{d|1:(d-1)}(x_d|\boldsymbol{x}_{1:(d-1)})f_{d-1|1:(d-2)}(x_{(d-1)}|\boldsymbol{x}_{1:(d-2)})f_{1:(d-2)}(\boldsymbol{x}_{1:(d-2)}) \\ &= \dots \\ &= \left[\prod_{j=2}^{d} f_{j|1:(j-1)}(x_j|\boldsymbol{x}_{1:(j-1)})\right] f_1(x_1) \end{aligned}$$

□

Example 2.8 (Section 4.1 in [2]). Suppose $\boldsymbol{X} = (X_1, X_2, X_3)$.
By Definition 2.15

$$f_{123}(x_1, x_2, x_3) = f_{3|12}(x_3|x_1, x_2)f_{2|1}(x_2, x_1)f_1(x_1)$$

By Bayes' theorem

$$f_{3|12} = \frac{f_{123}}{f_{12}} = \frac{f_{123}}{f_{1|2}f_2} = \frac{f_{13|2}f_2}{f_{1|2}f_2} = \frac{f_{13|2}}{f_{1|2}}$$

By Sklar's theorem

$$f_{13|2}(x_1, x_3|x_2) = c_{13;2}(F_{1|2}(x_1|x_2), F_{3|2}(x_3|x_2); x_2)f_{1|2}(x_1|x_2)f_{3|2}(x_3|x_2)$$

Notice that copula density $c_{13;2}$ depends on given x_2.
Combining the last two statements

$$f_{3|12}(x_3|x_1, x_2) = c_{13;2}(F_{1|2}(x_1|x_2), F_{3|2}(x_3|x_2); x_2)f_{3|2}(x_3|x_2)$$

By Lemma 2.2

$$\begin{aligned} f_{3|2}(x_3|x_1) &= c_{23}(F_2(x_2), F_3(x_3))f_3(x_3) \\ f_{2|1}(x_2|x_1) &= c_{12}(F_1(x_1), F_2(x_2))f_2(x_2) \end{aligned}$$

Thus

$$\begin{aligned} f(x_1, x_2, x_3) =& c_{13;2}(F_{1|2}(x_1|x_2), F_{3|2}(x_3|x_2); x_2) \times c_{23}(F_2(x_2), F_3(x_3)) \\ & \times c_{12}(F_1(x_1), F_2(x_2))f_3(x_3)f_2(x_2)f_1(x_1) \end{aligned}$$

□

Definition 2.16 (Copulas associated with bivariate conditional distributions, Definition 4.5 [2]). Let $D = \{i_1, \dots, i_k\} \subset \{1, \dots d\}$ be a set of indices not including i, j with $i < j$ and $i_1 < \dots < i_k$. $C_{ij;D}(\cdot, \cdot; \boldsymbol{x}_D)$ is the copula associated with the bivariate conditional distribution of (X_i, X_j) given $\boldsymbol{X}_D = \boldsymbol{x}_D$ and

$$c_{ij;D} = c_{ij;D}(F_{i|D}(x_i|\boldsymbol{x}_D), F_{j|D}(x_j|\boldsymbol{x}_D); \boldsymbol{x}_D)$$

is the corresponding bivariate density function.

In contrast, the conditional distribution function of (U_i, U_j) given $\boldsymbol{U}_D = \boldsymbol{u}_D$ is $C_{ij|D}(\cdot,\cdot;\boldsymbol{u}_D)$ with corresponding bivariate density function $c_{ij|D}(\cdot,\cdot;\boldsymbol{u}_D)$. □

Definition 2.17 (Simplifying assumption, Definition 4.9 [2]). The simplifying assumption is satisfied when

$$c_{ij;D}(F_{i|D}(x_i|\boldsymbol{x}_D), F_{j|D}(x_j|\boldsymbol{x}_D); \boldsymbol{x}_D) = c_{ij;D}(F_{i|D}(x_i|\boldsymbol{x}_D), F_{j|D}(x_j|\boldsymbol{x}_D))$$

holds for all $\boldsymbol{x}_D$, for any i, j, D as in Definition 2.16. □

Since the joint density decomposition is not unique, each decomposition differs in their selection of bivariate conditional distributions.
We present two basic decompositions (C- and D-vines), before we show how to specify all decompositions (R-vines).
Assume the simplifying assumption holds throughout.

Theorem 2.5 (Drawable vines, D-vines, Theorem 4.7 [2]). Every joint density $f_{1:d}$ can be decomposed as

$$f_{1:d}(\boldsymbol{x}_{1:d}) = \left[\prod_{j=1}^{d-1}\prod_{i=1}^{d-j} c_{i(i+j);(i+1):(i+j-1)}\right] \cdot \left[\prod_{k=1}^{d} f_k(x_k)\right]$$

and the associated distribution is called a drawable vine or D-vine. □

Theorem 2.6 (Canonical vines, C-vines, Theorem 4.8 [2]). Every joint density $f_{1:d}$ can be decomposed as

$$f_{1:d}(\boldsymbol{x}_{1:d}) = \left[\prod_{j=1}^{d-1}\prod_{i=1}^{d-j} c_{j(j+i);1:(j-1)}\right] \cdot \left[\prod_{k=1}^{d} f_k(x_k)\right]$$

and the associated distribution is called a canonical vine or C-vine. □

Regular (R-)Vines

We assume the reader is familiar with the following graph-theoretic concepts: graph $G = (N, E)$, node $v \in N$, edge $e \in E$, connectedness and tree T.

Definition 2.18 (Regular vine (R-vine) tree sequence, Definition 5.4 [2]). The set of trees $\mathcal{V} = (T_1, \dots, T_{d-1})$ is a regular vine tree sequence on d elements if

1. $T_j = (N_j, E_j)$ is connected, $\forall j = 1, \dots, d-1$
2. T_1 has node set $N_1 = \{1, \dots, d\}$ and edge set E_1
3. For $j \geq 2$, T_j is a tree with node $N_j = E_{j-1}$ and edge set E_j.
4. For $j = 2, \dots, d-1$ and $\{a, b\} \in E_j$ it must hold that $|a \cap b| = 1$

□

Condition (4) above is called the *Proximity condition.* For $j \in \{2, \dots, d-1\}$ and edge $e = \{a, b\}$ in tree T_j, it guarantees that a and b (edges in tree T_{j-1}) share a common node in T_{j-1}.

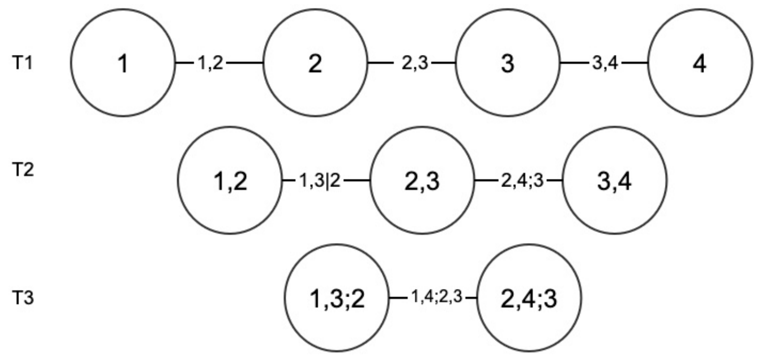

Figure 1: Four-dimensional D-vine

Definition 2.19 (Complete Union, Definition 5.7 [2])**.** For edge $e \in E_i$, the complete union of the edge e is the set

$$A_e = \{j \in N_1 \mid \exists e_1 \in E_1, \dots, e_{i-1} \in E_{i-1} \text{ s.t. } j \in e_1 \in \dots \in e_{i-1} \in e\}$$

□

Definition 2.20 (Conditioning Set and Conditioned Set, Definition 5.7 [2])**.** The conditioning set of an edge $e = \{a, b\}$ is

$$D_e = A_a \cap A_b$$

while the conditioned sets are

$$C_{e,a} = A_a \setminus D_e \quad \text{and} \quad C_{e,b} = A_b \setminus D_e$$

□

Note that $A_e \subset \{1, \dots, d\}$ and $C_{e,a}, C_{e,b}$ are singletons. The edges of a R-vine tree sequence are abbreviated as $e = (C_{e,a}, C_{e,b}; D_e)$. These edges are linked to copulas associated with bivariate conditional distributions (Definition 2.16).

Example 2.9 (Four-dimensional R-vine Tree Sequence)**.** The R-vine sequence $\mathcal{V} = (T_1, T_2, T_3)$ defined as

$$T_1 : N_1 = \{1, 2, 3, 4\} \quad E_1 = \{\{1, 2\}, \{2, 3\}, \{3, 4\}\}$$

$$T_2 : N_2 = E_1 = \{\{1, 2\}, \{2, 3\}, \{3, 4\}\} \quad E_2 = \{\{\{1, 2\}, \{2, 3\}\}, \{\{2, 3\}, \{3, 4\}\}\}$$

$$T_3 : N_3 = E_2 = \{\{\{1, 2\}, \{2, 3\}\}, \{\{2, 3\}, \{3, 4\}\}\}$$

$$E_3 = \{\{\{\{1, 2\}, \{2, 3\}\}, \{\{2, 3\}, \{3, 4\}\}\}\}$$

is also a D-vine. See Theorem 2.5 for the density.
The edge $e = \{a, b\} = \{\{2, 3\}, \{3, 4\}\} \in E_2$ has the complete unions $A_a = \{2, 3\}, A_b = \{3, 4\}$, conditioning set $D_e = \{3\}$, conditioned sets $C_{e,a} = \{2\}, C_{e,b} = \{4\}$ and can be abbreviated as $2, 4; 3$. See Figure 1 for a visual representation. □

Definition 2.21 (R-vine distribution, Definition 5.11 [2])**.** The joint distribution F of a d-dim random vector $\boldsymbol{X} = (X_1, \dots, X_d)$ has a R-vine distribution, if we can specify a triplet $(\mathcal{F}, \mathcal{V}, \mathcal{B})$ such that

1. $\mathcal{F} = (F_1, \dots, F_d)$ is a vector of continuous, invertible distribution functions representing the marginal distributions of the random variable X_i, $i = 1, \dots, d$
2. $\mathcal{V}$ is a R-vine tree sequence on d elements.
3. $\mathcal{B} = \{C_e \mid e \in E_i;\ i = 1, \dots, d-1\}$ where C_e is a symmetric bivariate copula where E_i is edge set of tree T_i in R-vine tree sequence $\mathcal{V}$.
4. For each $e \in E_i$, $i = 1, \dots, d-1, e = \{a, b\}$, C_e is the copula associated with the conditional distribution of $X_{C_{e,a}}$ and $X_{C_{e,b}}$ given $\boldsymbol{X}_{D_e} = \boldsymbol{x}_{D_e}$. By the simplifying assumption C_e is independent of the specific value $\boldsymbol{x}_{D_e}$.

□

We use an upper-triangular matrix to store the vine tree sequence to simplify likelihood calculations and structure-selection algorithms.

Definition 2.22 (R-vine matrix, Definition 5.23 [2])**.** Let R be an upper triangular matrix with entries $r_{i,j}$ for $i \leq j$. The elements $r_{i,j}$ have values 1 to d. Such a matrix R is called a R-vine matrix if it satisfies the following conditions

1. $\{r_{1,i}, \dots, r_{i,i}\} \subset \{r_{1,j}, \dots, r_{j,j}\}$ for $1 \leq i < j \leq d$ (The entries of a column are contained in all columns right of this column)
2. $r_{i,i} \notin \{r_{1,i-1}, \dots, r_{i-1,i-1}\}$ (The diagonal entry of a column does not appear in any column further to the left)
3. For $i = 3, \dots d$ and $k = 1, \dots, i-1$ there exists (j, l) with $j < i$ and $l < j$ such that

$$\{r_{k,i}, \{r_{1,i}, \dots, r_{k-1,i}\}\} = \{r_{j,j}, \{r_{1,j}, \dots, r_{l,j}\}\}$$

or

$$\{r_{k,i}, \{r_{1,i}, \dots, r_{k-1,i}\}\} = \{r_{l,j}, \{r_{1,j}, \dots, r_{l-1,j}, r_{j,j}\}\}$$

where the last condition is the Proximity condition from Definition 2.18. □

Example 2.10 (Four-dimensional R-vine matrix)**.** The four-dimensional R-vine tree sequence from the last example has R-vine matrix $\begin{pmatrix} 3 & 3 & 3 & 2 \\ & 4 & 4 & 3 \\ & & 2 & 4 \\ & & & 1 \end{pmatrix}$ that specifies the bivariate copulas in the density decomposition $\begin{pmatrix} - & 3,4 & 2,3 & 1,2 \\ & - & 2,4;3 & 1,3;2 \\ & & - & 1,4;2,3 \\ & & & - \end{pmatrix}$ because the diagonal entry $r_{k,k}, k = 2, \dots d$

and any entry above the diagonal $r_{j,k}, j < k$ form a conditioned set $C_e = \{r_{k,k}, r_{j,k}\}$ with associated conditioning set $D_e = \{m_{j-1,k}, \dots, m_{1,k}\}$. □

Recursion Formula

To simulate from Vine copulas with partially known values, we need conditional distributions (e.g. $F_{1|2:d}$), which we will invert to apply Inverse Transform Sampling. This is exactly what we did in Algorithm 2.1. The following theorem lets us express conditional distributions in terms of bivariate copulas.

Theorem 2.7 (Recursion for conditional distribution functions, Theorem 4.10, [2])**.** For X random variable and $\boldsymbol{Y}$ random vector, such that $(X, \boldsymbol{Y})$ has an absolutely continuous joint distribution. If $\boldsymbol{Y}_{-j}$ denotes the subvector of $\boldsymbol{Y}$ with component Y_j removed then

$$F_{X|\boldsymbol{Y}}(\cdot \mid y) = \frac{\partial C_{X,Y_j;\boldsymbol{Y}_{-j}}(F_{X|\boldsymbol{Y}_{-j}}(x \mid \boldsymbol{y}_{-j}), F_{Y_j|\boldsymbol{Y}_{-j}}(y \mid \boldsymbol{y}_{-j}))}{\partial F_{Y_j|\boldsymbol{Y}_{-j}}(y \mid \boldsymbol{y}_{-j})}$$

where $C_{X,Y_j;\boldsymbol{Y}_{-j}}(\cdot,\cdot|\boldsymbol{y}_{-j})$ is the copula corresponding to (X, Y_j) given $\boldsymbol{Y}_{-j} = \boldsymbol{y}_{-j}$. □

We will assume the simplifying assumption (Definition 2.17) holds when we apply Theorem 2.7.

Example 2.11. Suppose we believe the R-vine matrix R (featured in the last two-examples) is the best model for a particular data set.

$$\mathrm{R} = \begin{pmatrix} 3 & 3 & 3 & 2 \\ & 4 & 4 & 3 \\ & & 2 & 4 \\ & & & 1 \end{pmatrix}$$

Since this model is a D-vine, we apply Theorem 2.5 to derive the density

$$\begin{aligned} f_{1:4}(\boldsymbol{x}_{1:4}) = {} & c_{1,4;2,3}(F_{1|2,3}, F_{4|2,3}) \times c_{1,3;2}(F_{1|2}, F_{3|2}) \times c_{2,4;3}(F_{2|3}, F_{4|3}) \times \\ & c_{1,2}(F_1, F_2) \times c_{2,3}(F_2, F_3) \times c_{3,4}(F_3, F_4) \times f_1(x_1) f_2(x_2) f_3(x_3) f_4(x_4) \end{aligned}$$

where we omitted the distribution function arguments, e.g. $F_{1|2,3}$ is short-hand for $F_{1|2,3}(x_1 \mid (x_2, x_3))$, see Definition 2.16.

We can choose bivariate copula families (e.g. Gaussian, Clayton, Frank) to specify the functional form of $c_{1,3;2}$ and the other copula conditional densities. Then we can maximize the likelihood over the data to derive the copula parameters.

Referring back to Sklar's theorem (Theorem 2.4) we can rewrite the density as

$$\begin{aligned} F_{1:4}(\boldsymbol{x}_{1:4}) = {} & C_{1,4;2,3}(F_{1|2,3}, F_{4|2,3}) \times C_{1,3;2}(F_{1|2}, F_{3|2}) \times C_{2,4;3}(F_{2|3}, F_{4|3}) \times \\ & C_{1,2}(F_1, F_2) \times C_{2,3}(F_2, F_3) \times C_{3,4}(F_3, F_4) \end{aligned}$$

Suppose we seek $F_{1|2:4}$ so that we can simulate x_1 given (x_2, x_3, x_4).

First apply Theorem 2.7 with $X = x_1$, $\boldsymbol{Y} = (x_2, x_3, x_4)$ and $Y_j = x_4$ to obtain

$$F_{1|2:4} = \frac{\partial C_{1,4;2,3}(F_{1|2,3}(\cdot|\cdot), F_{4|2,3}(\cdot|\cdot))}{\partial F_{4|2,3}(\cdot|\cdot)} := h_{1|4;2,3}(\cdot|\cdot)$$

then apply Theorem 2.7 with $X = 1, \boldsymbol{Y} = (x_2, x_3), Y_j = x_3$ to obtain

$$F_{1|2,3} = \frac{\partial C_{1,3;2}(F_{1|2}, F_{3|2})}{\partial F_{3|2}} := h_{1|3;2}$$

repeat with $X = 4, \boldsymbol{Y} = (x_2, x_3), Y_j = x_3$ to obtain

$$F_{4|2,3} = \frac{\partial C_{2,4;3}(F_{2|3}, F_{4|3})}{\partial F_{2|3}} := h_{4|2;3}$$

Notice above that we reuse the previously derived copulas $C_{1,4;2,3}, C_{1,3;2}, C_{2,4;3}$.
To derive $F_{1|2} = h_{1|2}, F_{2|3} = h_{2|3}, F_{4,3} = h_{4|3}$, refer back to Lemma 2.2 and Definition 2.14. Putting it all together

$$F_{1|2:4} = h_{1|4;2,3}\Bigg(h_{1|3;2}\big(h_{1|2}(x_1|x_2) \mid h_{3|2}(x_3|x_2)\big) \Bigg| h_{4|2;3}\big(h_{4|3}(x_4|x_3) \mid h_{2|3}(x_2|x_3)\big)\Bigg)$$

The above h-functions can be derived analytically for some copula families, otherwise they are evaluated numerically.
By Inverse Transform Sampling, $F^{-1}_{1|2:4}(u|(x_2, x_3, x_4))$ has the same distribution as X_1 where u is a Uniform $[0, 1]$ random sample. $\square$

3 Copula-based Imputation

In this section, two competing models are presented. The first set of models is based on Vine Copulas, which specify a hierarchical distribution over the data, while the second set is based on Gaussian Copulas, which map observations to a latent normal space. Recently, Gaussian Copula model-based imputation was shown [13] to outperform the start-of-the-art in a variety of settings. Thus, it will be good to be familiar with the working details of both model's assumptions before a comparison is made.

3.1 Copula Imputation Workflow

To fit a copula model, we need to map real-valued observations onto the unit hypercube. Given a copula model, if we use Inverse Transform or Inverse Rosenblatt Transform sampling, then only select missing data patterns can be imputed. When we sample to impute, we may need multiple copula models to cover all missingness patterns. In contrast, with conditional mean imputation, we always only need one copula model. Furthermore, because the copula model parameters were estimated over an incomplete data set, we might benefit from re-estimating the parameters as we complete the data.

Change of scale

For $n, d \in \mathbb{Z}$, let $\mathrm{X} \in \mathbb{R}^{n \times d}$ be a data matrix with one-or-more missing entries.
Copulas are supported over the unit-hypercube, but data matrices X have real-valued columns. If we have the marginal distribution functions, $F_j,\ j \in \{1, \dots, d\}$ then we can transform from original scale to copula scale as $U_j = F_j(X_j)$. If we don't know the true marginal distribution function, we can use the empirical distribution function $\hat{F}_j$.

Definition 3.1 (Original and copula scales, Definition 3.11 in [2])**.** The original scale is $(X_1, \dots, X_d)$ where $X_j \in \mathbb{R}^{n \times 1}$ with marginal distribution functions F_j whereas the copula scale is $(U_1, \dots, U_d)$ where $U_j := F_j(X_j) \in [0, 1]^{n \times 1}$. □

Definition 3.2 (Empirical Distribution Function, Definition 1.1 in [2])**.** Given $x_{1j}, \dots, x_{nj}$, the Empirical Distribution Function $\hat{F}_j$ is defined as

$$\hat{F}_j(x) = \frac{1}{n+1} \sum_{i=1}^{n} \mathbb{1}_{\{x_{ij} \leq x\}}$$

□

Evaluation criteria

These criteria are used to evaluate accuracy, run time and estimation bias respectively.

Definition 3.3 (Scaled Mean Absolute Error, SMAE, [13])**.** For columns with missing data $I \subset$

$\{1,\dots,d\}$, the scaled mean absolute error (SMAE) is a measure of imputation error, defined as

$$\text{SMAE} = \frac{1}{|I|}\sum_{j\in I}\frac{\|X'_j - X_j^{\text{true}}\|_1}{\|X_j^{\text{med}} - X_j^{\text{true}}\|_1}$$

where X'_j imputes missing values using the chosen method, X_j^{med} imputes using the observed median of the j^{th} column and X_j^{true} are the true values. □

SMAE values less than 1 imply that the chosen method outperforms column median imputation.

Definition 3.4 (Elapsed, Spent or Run time)**.** We take the difference between consecutive calls to python's `time.process_time_ns()`, which returns the sum of the system and user CPU time of the current process throughout a process in nanoseconds excluding the time elapsed during sleep. □

Elapsed time per method is converted into the second-scale, where higher values indicate worse performance.

Definition 3.5 (Bias or Relative Error, [13])**.**

$$\text{Bias} = \frac{\|\hat{\Sigma} - \Sigma\|_F}{\|\Sigma\|_F}$$

where $\hat{\Sigma}$ is the estimated correlation matrix calculated over the imputed dataset. □

Bias is reported as a float, but is interpreted as a percentage, i.e. a value of 0.01 means 1% relative error, which is negligible.

3.2 Vine Copula Imputation

Vine Copula Imputation was first described in Zeisberger's Masters Thesis [12]. We describe two of his algorithms: CopReg and CopFit. Later, another Master's Thesis by Sakuth [10] expanded the Vine Copula Imputation methods. We describe his main algorithm, MDPFit at the end of this sub-section.

CopReg: Vine Copula Regression Imputation

Let $\mathrm{U} = (u_{ij}) \in [0,1]^{n\times d}$ be a continuous data matrix in the d-dim unit hypercube. The rows of U are $\boldsymbol{u}_i = (u_{i1},\dots u_{id}),\ i = 1,\dots,n$ and the columns are $U_j = (u_{1j},\dots,u_{nj}),\ j = 1,\dots,d$. Assume MAR missingness mechanism. The complete case matrix is $\overline{\mathrm{U}}$ and the available case matrix is $\overline{\mathrm{U}}_{SC}$ for missingness pattern S.

If we use Inverse Transform Sampling for simulation, then we need to know when we can simulate a missing variable given values of other variables. We'd like to use as many of the other variables as possible. However, for some combinations of vine structures and non-response variables, Inverse Transform Sampling is not possible because we would have to condition on a missing value. The following theorem characterizes when its possible to use Inverse Transform Sampling.

Theorem 3.1 (Inverse Transform Sampling of missing values, Theorem 5 [12], Lemma 4.5 [10])**.** Given non-response variable $j \in \{1,\dots,d\}$ and $D \subset \{1,\dots,d\}\setminus\{j\}$, and R-vine matrix R fit over

$\{j\} \cup D$, the following are equivalent (1) we can simulate $U_j|U_D$ via inverse transform sampling (2) in every tree of the R-vine tree sequence, i is in the conditioned set of a leaf node (3) in every tree of the R-vine tree sequence, i is not in the conditioning set of any bivariate copula $\square$

The first algorithm is Vine Copula Regression Imputation (CopReg), see Section 4.1 in [12] for more details.

Algorithm 3.1: CopReg: Single column missing data for many rows

```
input  : $\mathrm{U}, \overline{\mathrm{U}}$ with only $U_1$ missing data for the first $r$ rows.
output: Imputed values $u^*_{11}, \dots, u^*_{r1}$.
1 Fit R-vine model to $\overline{\mathrm{U}}$ such that $U_1|U_2, \dots U_d$ can be simulated.
2 Simulate $u^*_{11} \sim U_1|U_2 = u_{12}, \dots, U_d = u_{1d}, \ \dots, \ u^*_{r1} \sim U_1|U_2 = u_{r2}, \dots, U_d = u_{rd}$.
```

Algorithm 3.2: CopReg: Many columns missing data for one row

```
input  : $\mathrm{U}, \overline{\mathrm{U}}$ with $U_1, \dots, U_{m-1}$ missing data for the first row.
output: Imputed values $u^*_{11}, \dots, u^*_{1(m-1)}$.
1 Fit a R-vine model $\mathrm{R}_1$ over $(U_1, U_m, \dots, U_d)$ over $\overline{\mathrm{U}}$ such that $U_1|U_m, \dots, U_d$ can be
    simulated.
2 Simulate $u^*_{11} \sim U_1|U_m = u_{1m}, \dots, U_d = u_{1d}$ from $\mathrm{R}_1$.
3 Extend $\mathrm{R}_1$ by adding non-response variable 2 to fit $\mathrm{R}_2$ over $(U_1, U_2, \dots, U_d)$ over $\overline{\mathrm{U}}$. The
    model $\mathrm{R}_2$ is constrained such that $U_2|U_1, U_m, \dots, U_d$ and $U_1|U_m, \dots, U_d$ can be
    simulated.
4 Simulate $u^*_{12} \sim U_2|U_1 = u^*_{11}, U_m = u_{1m}, \dots, U_d = u_{1d}$ from $R_2$.
5 Repeat last two-steps to fit models $\mathrm{R}_3, \dots, \mathrm{R}_{m-1}$ and impute $u^*_{13}, \dots, u^*_{1(m-1)}$.
```

Algorithm 3.3: CopReg: Many columns missing data for many rows

```
input  : $\mathrm{U}, \overline{\mathrm{U}}$ with $S_1, \dots, S_k \subset \{1, \dots, d\}$ missingness patterns.
output: Imputed values $u^*_{ij}$ for $\forall i \in I_S, \forall j \in S, S \in \{S_1, \dots, S_k\}$
1 for missingness pattern $S$ do
2     Label rows $1, \dots, m \in I_S$ with missing values in exactly columns $S$.
3     Apply algorithm 3.2 on the row 1 associated with $S$.
4     Collect fitted R-vine models $(\mathrm{R}_1, \dots, \mathrm{R}_{|S|})$.
5     for row $i = 2, \dots m$ do
6         Apply algorithm 3.2 on row $i$ using prefitted models.
```

CopFit: Vine Copula Fitting Imputation

The second algorithm is Vine Copula Fitting Imputation (CopFit), see Section 4.2 in [12] for more details. The last algorithm incrementally builds up a Vine matrix starting with known variables and adding unknown variables. In contrast, the following algorithm starts with the best fitting Vine matrix (over complete cases) and searches for all sub-matrices within this matrix. In either algorithm, we apply Theorem 3.1 and Inverse Transform Sampling to sequentially impute missing values per missingness pattern.

Algorithm 3.4: CopFit: Find sub-vine structures

```
input  : R-vine matrix R, non-response variable $j$
output: $r_j$ sub-vines, $L = (\mathrm{R}_1, \dots, \mathrm{R}_{r_j})$ satisfying Theorem 3.1.
L ← ()
for column $l \in \{d, \dots, 1\}$ do
    if $r_{il} = j$ and $i \le l - 2$ then
        copy R as R'
        delete(column $l$ from R') and delete(all entries $r_{ll}$ from R')
        if R' only contains variable $j$ then
            continue
        else if R' has $j$ in any conditioning set then
            append(call(Algorithm 3.4 on R') to $L$)
        else
            append(R' to $L$)
    if exists row $k$ such that $r_{kl} = j$ then
        copy R as R'
        for $m \in \{r_{(k+1)l}, \dots, r_{kk}\}$ do
            delete(columns where $m$ is on the diagonal) and
            delete(all entries $m$ from R')
        if R' only contains variable $j$ then
            continue
        else if R' has $j$ in any conditioning set then
            append(call(Algorithm 3.4 on R') to $L$)
        else
            append(R' to $L$)
```

Example 3.1 (Algorithm 3.4, Example 15 in [12])**.** The following R-vine matrix has exactly 3 sub-vine structures ($L = (R_1^{(1)}, R_{00}^{(1)}, R_{01}^{(1)})$) where non-response variable 1 is not in the conditioning set of any tree.

$$\mathrm{R} = \begin{bmatrix} 1 & 1 & 2 & 2 & 1 \\ & 2 & 1 & 1 & 2 \\ & & 3 & 3 & 4 \\ & & & 4 & 3 \\ & & & & 5 \end{bmatrix}, \begin{bmatrix} - & 12 & 23 & 24 & 15 \\ & - & 13|2 & 14|2 & 25|1 \\ & & - & 34|12 & 45|12 \\ & & & - & 35|124 \\ & & & & - \end{bmatrix}$$

$$\mathrm{R}_1^{(1)} = \begin{bmatrix} 1 & 1 \\ & 5 \end{bmatrix}, \mathrm{R}_{00}^{(1)} = \begin{bmatrix} 1 & 1 & 2 \\ & 2 & 1 \\ & & 3 \end{bmatrix}, \mathrm{R}_{01}^{(1)} = \begin{bmatrix} 1 & 1 & 2 \\ & 2 & 1 \\ & & 4 \end{bmatrix}$$

□

Algorithm 3.5: CopFit: Impute one missingness pattern

```
input  : U, missingness pattern $S = \{1, \ldots, n\}$, $\overline{\mathrm{U}}_{S^C}$ available cases.
output: Imputed values $u^*_{i1}, \ldots, u^*_{in}, \forall i \in I_S$ incomplete rows.
1  Fit R-vine model R to $\overline{\mathrm{U}}_{S^C}$.
2  for non-response column $j \in S$ do
3      Apply algorithm 3.4 to find $r_j$ sub-vine structures in R.
4  for incomplete row $i \in I_S$, column $j \in S$ do
5      for sub-vine $k \in \{1, \ldots, r_j\}$ do
6          Define conditioning set $D_k \subset \{U_{n+1} = u_{i(m+1)}, \ldots, U_d = u_{id}\}$
7          if $j = 1$ then
8              Simulate $u^{*(k)}_{ij} | D_k$.
9          else
10             Simulate $u^{*(k)}_{ij} | U_1 = u^*_{i1}, \ldots, U_{j-1} = u^*_{i(j-1)}, D_k$.
11     Set $u^*_{ij} = \frac{1}{r_j} \sum_{l=1}^{r_j} u^{*(k)}_{ij}$.
```

MDPFit: Missing Data Pattern Imputation

Definition 3.6 (Diagonal R-vine matrix, Lemma 4.1 [10])**.** Given fitted R-vine matrix R of dimension d with diagonal $(r_{11}, \ldots, r_{dd})$, we can always relabel variables such that the super diagonal is $(r_{11}, \ldots, r_{(d-1)(d-1)})$. This relabeling is always possible and does not change the R-vine distribution. □

Let R, R' denote diagonal R-vine matrices of dimensions $d, (d-1)$ respectively.

Definition 3.7 (Ad-hoc imputable variables, Lemma 4.5 [10])**.**] Given R in diagonal form, the variables $\{r_{dd}, r_{(d-1)d}\}$ are ad-hoc imputable. When $j \in \{r_{dd}, r_{(d-1)d}\}$ then Theorem 3.1 applies with $D = \{1, \ldots, d\} \setminus \{j\}$. □

Ad-hoc imputable variables have the advantage of being simulatable given values of all other variables in the vine. But, every diagonal R-vine matrix only has 2 such variables. We gain 1 additional ad-hoc imputable variable when we use Algorithm 3.6 to reduce the R-vine matrix's dimension. See Theorem 4.7 in [10] for details.

Algorithm 3.6: Reduce R-vine matrix

```
input  : R in diagonal form, $j \in \{r_{dd}, r_{(d-1)d}\}$
output: $\mathrm{R}' = R \setminus \{j\}$
if $j = r_{dd}$ then
    delete(row $d$) and delete(column $d$)
else if $j = r_{(d-1)d}$ then
    delete(row $d-1$) and delete(column $d-1$)
```

Example 3.2. Start with an arbitrary 5-dim R-vine matrix and tree sequence, R_1.

$$R_1 = \begin{pmatrix} 5 & 5 & 4 & 3 & 2 \\ & 4 & 5 & 4 & 3 \\ & & 3 & 5 & 4 \\ & & & 2 & 5 \\ & & & & 1 \end{pmatrix}, \begin{pmatrix} - & 4,5 & 3,4 & 2,3 & 1,2 \\ & - & 3,5|4 & 2,4|3 & 1,3|2 \\ & & - & 2,5|3,4 & 1,4|2,3 \\ & & & - & 1,5|2,3,4 \\ & & & & - \end{pmatrix}$$

R_1 can be written in diagonal form, which utilizes the same copulas (up to ordering of conditioning and conditioned set).

$$R_2 = \begin{pmatrix} 3 & 3 & 3 & 2 & 4 \\ & 2 & 2 & 3 & 3 \\ & & 4 & 4 & 2 \\ & & & 1 & 1 \\ & & & & 5 \end{pmatrix}, \begin{pmatrix} - & 2,3 & 4,3 & 1,2 & 5,4 \\ & - & 4,2|3 & 1,3|2 & 5,3|4 \\ & & - & 1,4|2,3 & 5,2|3,4 \\ & & & - & 5,1|2,3,4 \\ & & & & - \end{pmatrix}$$

Both R_1, R_2 specify the same Vine structure. Variables $j \in \{1, 5\}$ sastisfy the conditions of Theorem 3.1 because they appear in the special $\{r_{dd}, r_{(d-1)d}\}$. When we use Algorithm 3.6 to reduce R_2 with variable 1 then we get the following R-vine matrix and tree sequence.

$$R_2' = \begin{pmatrix} 3 & 3 & 3 & 4 \\ & 2 & 2 & 3 \\ & & 4 & 2 \\ & & & 5 \end{pmatrix}, \begin{pmatrix} - & 2,3 & 4,3 & 5,4 \\ & - & 4,2|3 & 5,3|4 \\ & & - & 5,2|3,4 \\ & & & - \end{pmatrix}$$

When we diagonalize R_2', we find that $j \in \{2, 5\}$ are ad-hoc imputable variables since they are in the special $\{r_{(d-1)(d-1)}, r_{(d-2)(d-1)}\}$ positions.

$$R_3 = \begin{pmatrix} 4 & 4 & 4 & 3 \\ & 3 & 3 & 4 \\ & & 5 & 5 \\ & & & 2 \end{pmatrix}, \begin{pmatrix} - & 3,4 & 5,4 & 2,3 \\ & - & 5,3|4 & 2,4|3 \\ & & - & 2,5|4,3 \\ & & & - \end{pmatrix}$$

□

Definition 3.8 (Ad-hoc imputable missingness pattern, Definition 4.10 [10])**.** Given R, a missingness pattern $S = \{s_1, \dots, s_n\} \subset \{1, \dots, d\}$ is ad-hoc imputable when for all $s_i \in S$, s_i is an ad-hoc imputable variable in R $\setminus \{s_1, \dots, s_{i-1}\}$. □

When we impute an ad-hoc missingness pattern, we use the full joint distribution $F_{1:d}$ distribution. Most missingness patterns are only ad-hoc imputable for joint distribution F_I where $I \subsetneq \{1, \dots, d\}$. See Theorem 4.11 in [10] for details.

Algorithm 3.7: Ad-hoc missingness pattern imputation

```
input  : Data matrix U, R, ad-hoc imputable missingness pattern $S = \{s_1, \dots, s_n\}$
output: Imputed values for incomplete rows $I_S$
1 set $D = \{1, \dots, d\} \setminus S$ observed variables.
2 for non-response column $j \in \{s_1, \dots, s_n\}$ do
3     for incomplete row $i \in I_S$ do
4         simulate $u^*_{ij} \sim U_j | D$
5     set $D \leftarrow D \cup \{j\}$.
```

We have seen that we can generate more ad-hoc imputable variables by reducing a R-vine matrix. Analogously, we can fit a R-vine on available cases (relative to a non-response variable) and extend the R-vine matrix by adding the non-response variable such that it is ad-hoc imputable. See Algorithm 3.8 (adapted from Section 4.2.2 in [10]).

Algorithm 3.8: Extend R-vine matrix

```
input  : $R'$ without variable $d$
output: R with $r_{dd} \leftarrow d$
init R as a $d-$dim matrix.
copy $r_{ij} \leftarrow r'_{ij}$, $\forall i, j \in \{1, \dots, d-1\}$.
set $r_{dd} = d$.
for tree $i \in \{1, \dots, d-1\}$ do
    set conditioning set $D = \{r_{1d}, \dots, r_{(i-1)d}\}$
    set available variables $I = \{1, \dots, d-1\} \setminus D$
    set eligible variables $J = \{j \in I \mid (j, d|D)$ satisfies (4) in definition 2.18$\}$
    set $r_{id} = \arg\max_{j \in J} \tau(j, d)$ where $\tau(\cdot, \cdot)$ is Kendall's Tau.
    fit-copula $C_{j,d|D}$
```

Finally, we describe Sakuth's main procedure, Missing Data Pattern Imputation (MDPfit), in Algorithm 3.9, which sequentially imputes missing values for data matrices with genreal missingness structure. see Algorithm 4.19 in [10] for more details.

Algorithm 3.9: MDPfit: General Vine Copula Imputation

```
Input: Data matrix U, procedure tmis-pat(U) $\to \mathcal{S}$ list of missingness patterns, denoted
       by $\mathcal{S}$, procedure adhoc-mis-pat(R) $\to \mathcal{S}_A$ list of ad-hoc missingness patterns
output: Imputed data matrix $U^*$
1  copy $U^*$ from U.
2  fit R-vine matrix R on complete cases $\overline{U}$.
3  for adhoc missingness pattern $S \in$ adhoc-mis-pat(R) do
4      apply Algorithm 3.7 to impute $U^*$
5  while $U^*$ has any missing data do
6      for missingness pattern $S \in$ sort-asc(mis-pat($U^*$)) do
7          fit R-vine matrix R on available cases of $U^*_S$.
8          for non-response variable $j \in S$ do
9              apply Algorithm 3.8 to extend $R = R \cup \{j\}$.
10         for adhoc missingness pattern $S_A \in$ adhoc-mis-pat(R) do
11             apply Algorithm 3.7 to impute $U^*$.
```

3.3 Gaussian Copula Imputation

The following results are taken from [13].

Definition 3.9 (Gaussian Copula)**.** $X \in \mathbb{R}^d$ follows the Gaussian copula $X \sim GC(\Sigma, f)$ if there exists a correlation matrix Σ and elementwise strictly monotonic function $f : \mathbb{R}^d \to \mathbb{R}^d$ such that $f(Z) = f(X)$ for $Z \sim \mathcal{N}_d(0, \Sigma)$. □

Lemma 3.1 (Transform Continuous Random Vector to Standard Normals)**.** For $X \in \mathbb{R}^d$ continuous random vector, with CDF F_j per coordinate, there exists a unique coordinate-wise strictly monotonic function $f(Z) = (f_1(Z_1), \dots f_d(Z_d))$ such that

$$X_j = f_j(Z_j) \quad \text{and} \quad f_j^{-1}(X_j) = Z_j \quad \text{and} \quad f_j = F_j^{-1} \circ \Phi \quad j \in \{1, \dots, d\}$$

where Φ is the standard normal CDF. □

Suppose $X \in \mathbb{R}^{n \times d}$ is a data matrix sampled from a random vector with strictly continuous components. Assuming MCAR missingness mechanism.

Algorithm 3.10: GCImpute: Gaussian Copula Imputation

```
input  : $\overline{X}$ complete case matrix.
output: Imputed data matrix $X^*$.
1  compute monotone mappings to latent space $\hat{f}^{-1}$ via Lemma 3.1
2  estimate correlation matrix $\hat{\Sigma}$ via EM algorithm (Section 6.3 [13])
3  for row $i \in \{1, \dots, n\}$ do
4      for missingness pattern $S \subset \{1, \dots, d\}$ such that $x_{ij}$ missing, $\forall j \in S$ do
5          impute $\tilde{z}_{iS} = \mathbb{E}[z_{iS} \mid z_{iS^c}]$ (Section 6.4-5 [13])
6          transform $\tilde{x}_{iS} = \hat{f}_S(\hat{z}_{iS})$
```

4 Comparison

Referring back to Section 1, we were interested in two key questions, which we can answer if we characterize the variables that influence the accuracy and computational efficiency of Vine Copula imputation. In each of the following sections, we introduce a specific variable-of-interest, present additional theory and illustrative examples as necessary, state a hypothesis, conduct experiments on simulated data and interpret results.

4.1 Data generating process

Hierarchical data

Definition 4.1 (Hierarchically generated data)**.** Given a d-dim R-vine distribution (Definition 2.21) with bivariate Gaussian copulas (Example 2.7). We generate hierarchical data when we apply the Inverse Rosenblatt transform (Example 2.5) to sample $(U_1, \dots, U_d)$. $\square$

Definition 4.2 (Non-hierarchically generated data)**.** Given a d-dim multivariate normal distribution F with mean-zero and random correlation matrix [4]. We generate non-hierarchical data when we sample $(X_1, \dots, X_d)$ from F and transform each of the marginals X_i to a $[0, 1]$ space, i.e. $(u_{i1}, \dots, u_{id}) = (\Phi(x_{i1}), \dots, \Phi(x_{id})),\ i = \{1, \dots, n\}$ where Φ is the standard normal distribution function. $\square$

Example 4.1. Given three variables (X_1, X_2, X_3) consider the correlation matrix

$$\Sigma = \begin{bmatrix} 1 & 0.5 & 0.5 \\ 0.5 & 1 & 0.9 \\ 0.5 & 0.9 & 1 \end{bmatrix}$$

When we generate data non-hierarchically, we sample from the multivariate normal distribution $\mathcal{N}(\mathbf{0}, \Sigma)$ to get data matrix $\mathrm{X} \in \mathbb{R}^{n \times d}$, which we transform to $[0, 1]$ space $\mathrm{U} = [\Phi(X_1), \Phi(X_2), \Phi(X_3)]$. Instead, when we generate data non-hierarchically, we specify any valid 3-dim R-vine structure, for example

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 2 \\ 0 & 0 & 3 \end{bmatrix}$$

and corresponding bivariate Gaussian copulas $C_{12}, C_{13}, C_{3,2|1}$ with respective parameters $\rho = \{0.5, 0.5, 0.9\}$. We generate data hierarchically when we sample v_1, v_2, v_3 iid Uniform$[0, 1]$ and set $U_1 = v_1$, $U_2 = h^{-1}_{2|1}(v_2|u_1)$, and $U_3 = h^{-1}_{3|2,1}(v_3|h_{2|1}(u_2|u_1))$. $\square$

Hypothesis Both methods specify identical distributions once we fix a covariance structure. Thus, we should not expect hierarchically generated data to favor Vine- over Gaussian-Copula imputation.

Experiment Hierarchically (hier) and non-hierarchically (non) generate 100 datasets with $d = 10$, $n = 1000$, univariate missingness pattern and missingness rate 10%. Compare SMAE, bias and elapsed time for each of the four methods: Gaussian-copula imputation, CopReg, CopFit and MDPFit.

Results Surprisingly, both methods perform 1.5x-3x better under hierarchical data generation rather than non-hierarchical.

	smae (hier)	smae (non)	bias (hier)	bias (non)	elapsed (s)
copfit	0.32	0.46	0.01	0.01	5.93
copreg	0.14	0.39	0.00	0.00	5.74
gcimpute	0.14	0.28	0.00	0.00	0.24
mdpfit	0.15	0.40	0.00	0.00	16.62

Non-elliptical copulas

Sklar's theorem (Theorem 2.4) states that any d-dim distribution function can be expressed in terms of its marginal distributions and a copula (not necessarily Gaussian). This begs the question: when is the Gaussian copula an inappropriate model for the data?
To answer the above question, we introduce the two statistics: Kendall's Tau and Tail Dependence Coefficient. The former allows us to standardize parameters between different parametric bivariate copulas, while the latter allows us to characterise the joint probability of extremely small or large standardized observations in a bivariate setting.
For the following let $\boldsymbol{X} = (X_1, X_2)$ be a 2-dim random vector with marginal F_1, F_2 and let $C(u_1, u_2)$ be the copula from Sklar's theorem such that $F(X_1, X_2) = C(F_1(X_1), F_2(X_2))$. Let P be the probability function associated with the copula density and distribution functions.

Definition 4.3 (Kendall's Tau (τ), Definition 2.3 in [2]).

$$\tau(X_1, X_2) = P((X_{11} - X_{21})(X_{12} - X_{22}) > 0) - P((X_{11} - X_{21})(X_{12} - X_{22}) < 0)$$

where (X_{11}, X_{12}) and (X_{21}, X_{22}) are i.i.d copies of (X_1, X_2). □

Kendall's Tau is a scale-invariant way to measure bivariate (rank) correlation. For the parametric copula families we will shortly introduce, there are formulas to convert that family's parameter into Kendall's Tau, e.g. $\tau = \frac{2}{\pi}\sin^{-1}(\rho)$, $\tau \in [-1, 1]$ (Table 3.2 in [2]) for the Gaussian copula (Example 2.7).

Definition 4.4 (Tail dependence coefficients, Definition 2.12 in [2]).

$$\lambda^{upper} = \lim_{t\to 1^-} P(X_2 > F_2^{-1}(t) \mid X_1 > F_1^{-1}(t)) = \lim_{t\to 1^-} \frac{1 - 2t + C(t,t)}{1-t}$$
$$\lambda^{lower} = \lim_{t\to 0^+} P(X_2 \leq F_2^{-1}(t) \mid X_1 \leq F_1^{-1}(t)) = \lim_{t\to 0^+} \frac{C(t,t)}{t}$$

$C(t,t) \to 0$ as $t \to 0^+$ and $C(t,t) \to 1$ as $t \to 1^-$ (Definition 2.11). □

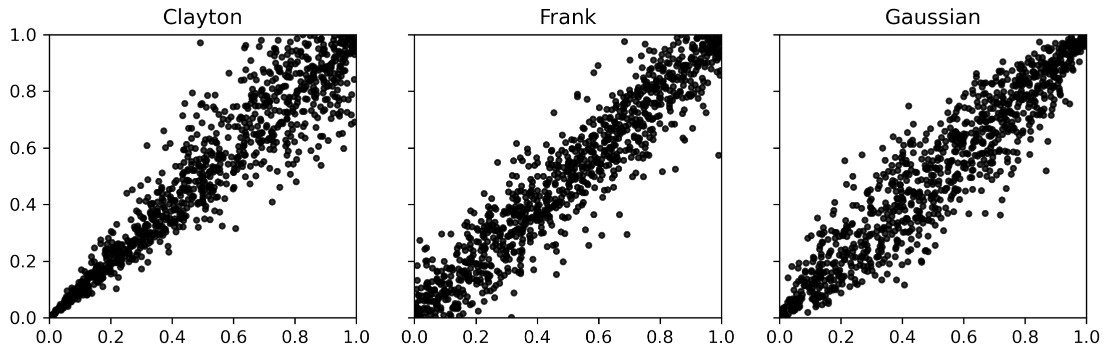

Figure 2: Clayton copula has higher correlation in the lower tail, while the Frank copula has a tubular shape in contrast to the Gaussian copula, all for the same Kendall's $\tau = 0.8$

Tail dependence coefficients are non-negative, with higher values reflecting higher conditional dependence in the bivariate tails.

The Gaussian copula is reflection symmetric, i.e. (X_1, X_2) has the same distribution as $(1 - X_1, 1 - Y_2)$, and has no tail dependence $\lambda = \lambda^{lower} = \lambda^{upper} = 0$. In contrast, the Clayton copula is asymmetric and has a lower tail dependence $\lambda = 2^{-1/\delta}$, while the Frank copula is non-elliptical with no tail dependence. See [5] for more details.

Definition 4.5 (Clayton copula, Example 3.1 in [2])**.**

$$C(u_1, u_2; \delta) = (u_1^{-\delta} + u_2^{-\delta} - 1)^{-\frac{1}{\delta}}$$

for $0 < \delta < \infty$ □

Definition 4.6 (Frank copula, Example 3.1 [2])**.**

$$C(u_1, u_2; \delta) = \frac{-1}{\delta} \ln \left(\frac{1}{1 - e^{-\delta}} \left[(1 - e^{-\delta}) - (1 - e^{-\delta u_1})(1 - e^{-\delta u_2}) \right] \right)$$

for $\delta \in [-\infty, \infty]$ □

Figure 2 illustrates the differences between the Clayton, Frank and Gaussian copula families, for $\tau = 0.8$.

Hypothesis Based on the presented theory, it stands to reason: if we generate data from a d-dim R-vine distribution with bivariate copulas that are purely Clayton and/or Frank, then we might find that Vine Copula based imputation model (allowed to fit Clayton and/or Frank copulas) better recovers missing observations.

Experiment Generate 5000 hierarchical datasets per copula type (Clayton, Frank and Gaussian) with $n = 1000$ samples, $d = 10$ dimensions, univariate missingness structure and missingness rate 10%. Per copula type, there will be 5000 datasets, each with one column missing. Calculate the average of absolute empirical Kendall $|\tau|$ between the missing column and the rest of the columns, denoted by $\overline{|\tau|}$. For higher $\overline{|\tau|}$ we expect easier recovery of the missing data.

Results Overall the test reveals that CopReg and MDPFit have comparable accuracy (smae) to GCImpute on a Gaussian copula type dataset. However, GCImpute outperforms all Vine copula based methods on Clayton and Frank copula type dataset, even though the latter fit Clayton and Frank copulas, respectively. Finally, GCImpute is the fastest method across all components of the test. As expected, the accuracy of each method increases as $\overline{|\tau|}$ increases.
Clayton copula type dataset

	smae				elapsed (s)			
$\overline{\lvert\tau\rvert}$ bucket	(0.059, 0.7]	(0.7, 0.8]	(0.8, 0.9]	(0.9, 1.0]	(0.059, 0.7]	(0.7, 0.8]	(0.8, 0.9]	(0.9, 1.0]
copfit	0.20	0.08	0.05	0.04	0.69	0.69	0.69	0.68
copreg	0.19	0.09	0.06	0.05	0.66	0.66	0.67	0.63
gcimpute	0.12	0.05	0.03	0.02	0.27	0.28	0.28	0.27
mdpfit	0.17	0.07	0.06	0.04	1.83	2.05	2.12	2.13

Frank copula type dataset

	smae				elapsed (s)			
$\overline{\lvert\tau\rvert}$ bucket	(0.069, 0.5]	(0.5, 0.6]	(0.6, 0.7]	(0.7, 0.9]	(0.069, 0.5]	(0.5, 0.6]	(0.6, 0.7]	(0.7, 0.9]
copfit	0.24	0.14	0.11	0.09	0.55	0.56	0.56	0.56
copreg	0.21	0.13	0.10	0.08	0.49	0.49	0.49	0.49
gcimpute	0.15	0.08	0.06	0.05	0.22	0.23	0.23	0.23
mdpfit	0.20	0.11	0.09	0.08	1.40	1.62	1.61	1.59

Gaussian copula type dataset

	smae			elapsed (s)		
$\overline{\lvert\tau\rvert}$ bucket	(0.049, 0.2]	(0.2, 0.3]	(0.3, 0.6]	(0.049, 0.2]	(0.2, 0.3]	(0.3, 0.6]
copfit	0.35	0.32	0.27	6.52	6.45	6.45
copreg	0.18	0.11	0.07	6.28	6.30	6.34
gcimpute	0.18	0.11	0.08	0.25	0.27	0.27
mdpfit	0.18	0.11	0.07	16.69	19.69	21.85

For this test, we omitted bias results, as they are nearly 0 across all methods and dataset combinations. Since bias measures how well we recover the covariance structure, we expect it to be low since we only omit 1% of total observations (1 column missing 10% out of a 10-dim dataset with 1000 units) in this test.
Also, Vine copula based methods are much faster imputing non-Gaussian copulas, but not faster than Gaussian copula imputation by [13].

4.2 Missingness pattern and structure

In the last experiments, we focused on single column missingness and incidentally found an empirical lower bound on elapsed time. For Gaussian Copula type datasets, the empirical lower bounds are roughly (CopReg, CopFit, GCImpute, MDPFit) $= (6, 6, 0.25, 17)$ seconds per dataset for the chosen dimension, sample size and missingness rate of $(10, 1000, 10\%)$.
One of the weaknesses of Vine Copula imputation methods that will soon become apparent is that its run-time scales as a function of number of missingness patterns (Definition 2.2). When we have a univariate missingness structure, we have exactly 1 missingness pattern. When we have a monotone missingness structure (Definition 2.6), we have upto $d-1$ missingness patterns. Finally, for all other missingness structures, with no empty rows, we can have upto $\sum_{k=1}^{d-1}\binom{d}{k}$ missingness patterns.

Example 4.2. For $d = 5$, there are upto $\binom{5}{1}+\binom{5}{2}+\binom{5}{3}+\binom{5}{4} = 5+10+10+5 = 30$ missingness patterns, corresponding to combinations of 1, 2, 3, 4 variables missing, respectively. □

For each of the methods, there are two key operations: copula parameter estimation (fit) and calculating samples or conditional expectations (inference). With GCImpute, regardless of number of missingness patterns, we have only 1 fit, but we always have 1 inference per missingness pattern (Algorithm 3.10). With CopReg, per missingness pattern, we have 1 fit and 1 inference because we sequentially impute and refit (Algorithm 3.2). With CopFit, like with GCImpute, we always have only 1 fit, however, we have upto d inferences per missingness pattern because of the sub-vine inferences (Algorithm 3.5). Finally, with MDPFit, we have upto 1 fit per missingness pattern and always have 1 inference per missingness pattern since a model can be used to impute more than one missingness pattern (Algorithm 3.9).

	num fits	num inferences	speed fit	speed inference
copfit	1	dimension $*$ (num mdps)	slow	fast
copreg	num mdps	num mdps	slow	fast
gcimpute	1	num mdps	fast	fast
mdpfit	O(num mdps)	num mdps	slow	fast

Hypothesis Of the Vine copula based imputation methods, CopFit will scale best on general and monotonic missingness structures, whereas CopReg will scale worst.

General missingness structure

Experiment Generate 1000 non-hierarchical datasets from multivariate normal distributions with $n = 1000$ samples, $d = 5$ dimensions, general missingness structure and missingness rate 10%. Fix the number of missingness patterns to exactly 20.

Results Compared to Section 4.1 the runtime of CopFit and MDPFit are higher by 30% while the runtime of GCImpute is unchanged. CopReg has a 600% higher runtime, though it has the best accuracy results of the Vine Copula-based methods.

method	mdps	smae	bias	elapsed_s	num fits	num inferences
copfit	20	0.84	0.02	8.06	1.00	81.15
copreg	20	0.79	0.02	43.16	40.00	40.00
gcimpute	20	0.58	0.03	0.26	1.00	20.00
mdpfit	20	0.87	0.03	21.96	19.40	20.00

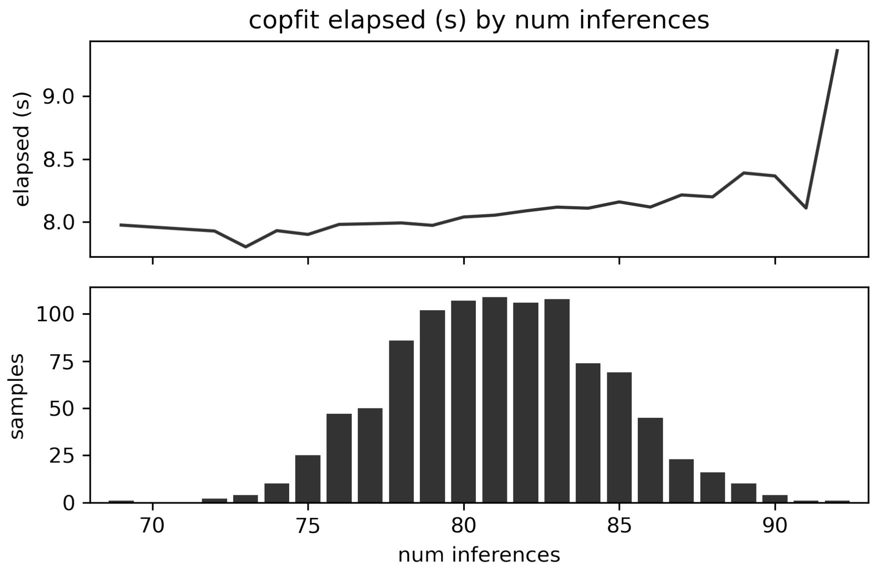

Figure 3: CopFit elapsed time very weakly linearly increases in number of missingness patterns, however, this effect is negligible relative to the per-fit time.

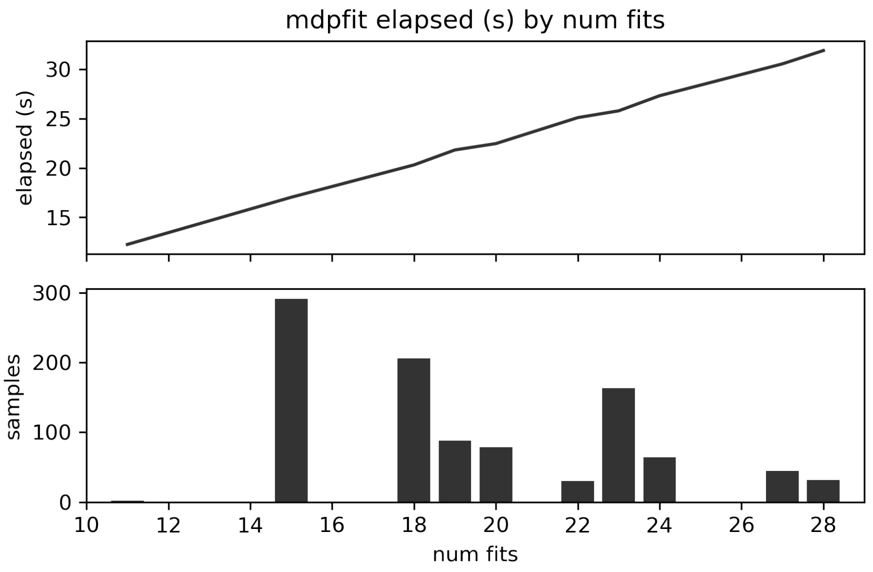

Figure 4: MDPFit has a variable number of copulas fit for a fixed number of missingness patterns and its elapsed time depends strongly linearly in the number of fits.

Monotonic missingness structure

Experiment Generate 1000 hierarchical datasets with bivariate Gaussian copulas with $n = 1000$ samples, $d = 10$ dimensions, monotonic missingness structure with between 1 to $d - 1$ columns missing (num. cols. mis.) and missingness rate of 10% per. column. Note that monotonic missingness structures have (number of columns missingness) missingness patterns per dataset.

Results Imputation accuracy decreases as number of missing columns increases, and the accuracy of Vine copula-based methods drops much faster. Values of SMAE above 1 indicate that median imputation is a better method.

smae by num. cols. mis.	1	2	3	4	5	6	7	8	9
copfit	0.55	0.69	0.77	0.85	0.91	0.97	1.03	1.08	1.15
copreg	0.34	0.49	0.60	0.72	0.84	0.95	1.06	1.17	1.27
gcimpute	0.26	0.34	0.40	0.47	0.55	0.63	0.71	0.79	0.87
mdpfit	0.35	0.46	0.57	0.68	0.80	0.91	1.03	1.16	1.25

We start to see observable increases in bias as number of missing columns grows, however, even the largest bias (6%) is negligible.

bias by num. cols. mis.	1	2	3	4	5	6	7	8	9
copfit	0.01	0.03	0.04	0.05	0.06	0.06	0.06	0.06	0.06
copreg	0.00	0.01	0.01	0.02	0.02	0.03	0.03	0.03	0.03
gcimpute	0.00	0.01	0.01	0.02	0.02	0.03	0.03	0.04	0.04
mdpfit	0.00	0.01	0.01	0.02	0.02	0.02	0.03	0.03	0.04

Gaussian copula imputation has a constant computation time in number of missingness patterns because it fits only once whereas Vine copula imputation has a computation time that increases in number of missingness patterns.

elapsed (s) by num. cols. mis.	1	2	3	4	5	6	7	8	9
copfit	5.17	6.03	6.71	7.19	7.56	7.79	7.94	8.02	8.05
copreg	4.97	9.65	14.10	18.32	22.40	26.35	30.24	34.01	37.71
gcimpute	0.34	0.34	0.34	0.34	0.34	0.34	0.35	0.35	0.34
mdpfit	15.05	29.20	44.63	61.60	77.82	92.06	98.14	96.94	90.62

4.3 Dimension, sample size and effective missingness rate

We expect imputation error to reduce as we increase sample size (all else being the same) because it should be easier to estimate the copula parameters. We also expect imputation error to reduce as dimension grows, because we have a higher chance that a well-correlated variable is available. However, due to the curse of dimensionality, we expect copula parameter estimation to be worse in increasing dimension. Finally, as the ratio of available-to-missing data reduces and we expect a higher imputation error.
We summarize our expectations as

$$\text{SMAE} \propto \frac{\#\text{ Missing samples}}{\text{Dimension} \times \#\text{ Units}} := \text{Effective missingness rate}$$

Effective missing rate is agnostic of missingness structure, unlike the previously mentioned missingness rate.

Example 4.3 (Missingness rate to effective missingness rate)**.** For univariate missingness structure with a 20% missingness rate, the effective missingness rate is $20\%/d$. For monotonic missingness structure with 20% missingness over 4 columns, we have an effective missingness rate of $20\% \cdot (4/d)$. For general missingness structure, the concept of missingness rate and its effective counterpart are equivalent. □

In reading [13], the authors demonstrate that their Gaussian copula model outperforms on “tall-skinny” datasets. These have significantly more units than variables and are hypothesized to originate from low-rank models.

Hypothesis As dimension and missingness rate grow, for fixed sample size, we expect Vine copula models to outperform Gaussian copula models.

Because of the previous experiments on run-time, to keep the computational budget limited, we restrict our attention to univariate and monotonic missingness structures. We expect that the concept of effective missing rate should tie these missingness structures to the general one.

Univariate missingness structure

Experiment Generate upto 100 hierarchical datasets with Gaussian bivariate copulas, $n = 1000$ samples, univariate missingness structure, missingness rate 10% and dimension varying $d = 10, \ldots, 100$. The effective missingness rate of $10\%/d$ decreases in dimension.

Results For $d = 50$, it takes approximately 24 hours of computation time to generate 100 datasets, mask them and run all four methods (GCImpute, CopReg, CopFit and MDPFit). The majority of this time is spent on the Copula based imputation methods. Nonetheless, results from the SMAE comparison upto dimension 100 show that CopReg and MDPFit remain competitive (in accuracy and relative error) with GCImpute in the univariate setting in increasing dimension. This generalizes the results found in the Hierarchical Data experiment (Section 4.1). See Figure 5.

Monotone missingness structure

In the last experiment, we increased dimension, but did not control effective missingness rate, which was decreasing to 0. We would like to know if Copula based methods can remain competitive in higher dimensions for a floored effective missingness rate. To keep the total experiment runtime reasonable, we only compare CopReg versus GCImpute, since CopReg has comparable performance to MDPFit, but is the fastest of three Copula based methods.

Experiment Generate 20 hierarchical datasets with Gaussian bivariate copulas, $n = 1000$ samples, monotone missingness structure, 20 columns missing data, missingness rate 50% per column and $d \in \{50, 60, 70, 80, 90, 100\}$. Effective missingness rate is floored at 10%.

Results The final head-to-head comparison shows that CopReg is not competitive against GCImpute in accuracy nor elapsed time, but has comparable bias results.

	smae		bias		elapsed (min)		effective missingness	
tag	copreg	gcimpute	copreg	gcimpute	copreg	gcimpute	copreg	gcimpute
d								
50	0.82	0.59	0.11	0.14	36.41	0.02	0.20	0.20
60	0.76	0.55	0.11	0.12	54.43	0.03	0.17	0.17
70	0.73	0.53	0.11	0.11	76.08	0.03	0.14	0.14
80	0.65	0.47	0.10	0.09	101.18	0.04	0.12	0.12
90	0.64	0.46	0.10	0.08	129.90	0.05	0.11	0.11
100	0.62	0.44	0.10	0.08	162.05	0.06	0.10	0.10

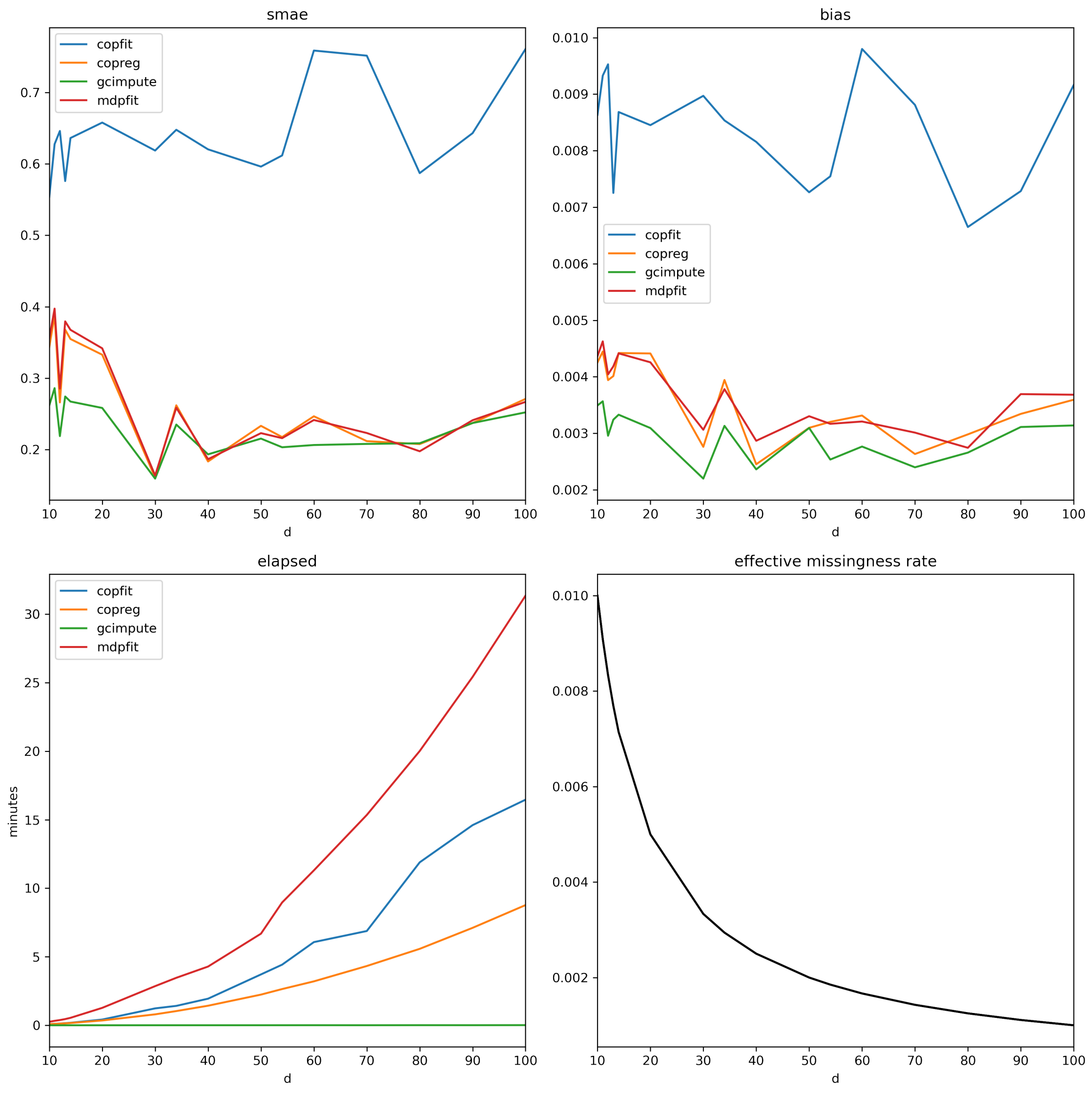

Figure 5: SMAE and Bias (Top; Orange/Red vs. Green): CopReg and MDPFit match GCImpute as dimension grows and effective missingness (Bottom Right) drops. However, total runtime of Vine-based methods grows exponentially per dataset in dimension (Bottom Left), making it prohibitively expensive to test on 100 datasets per dimension. CopFit has very poor accuracy and relative error results (Top; Blue)

5 Conclusion

Gaussian copula imputation [13] has better accuracy and speed compared to Vine copula imputation [12, 10] when in the MAR and MCAR settings. Vine copula-based algorithms have uncompetitive accuracy relative to median imputation and are slow. Vine- and Gaussian-methods both have similarly low biases if our task is to estimate a correlation matrix. But, this can be attributed to the completely-at-random missingness mechanism and low effective missingness rate of less than 10%. Of the Vine-copula based algorithms, CopReg is better than MDPFit and CopFit because it offers comparable accuracy at a lower runtime.

Vine copula-based algorithms trade-off between using the best fitting Vine copula model and using all available data to inform each imputation. If we choose the best fit model, the constraints of the tree structure and inverse Rosenblatt transform restrict the class of imputable missing data patterns (Definition 3.8). Thus, for most missing data patterns, we either use a lower-likelihood model with the right tree structure or a smaller model with fewer available variables to inform the imputed value. Vine-copula based algorithms are disadvantaged because they are sensitive to imputation order and the variables included in the conditioned set for the imputed sample draw. We hypothesize that this disadvantage can be a benefit in the missing not-at-random setting, where we might prioritize specific relationships between variables when imputing. Based on results in [12], we do not believe conditional mean imputation (used in GCImpute) gives GCImpute the edge over the draw-from-distribution approach (used in CopReg, CopFit, and MDPFit). Further experimentation is required to determine if multiple imputation can boost the accuracy of Vine-copula based methods.

Vine copula-based implementations require many copula fits. The time cost of copula fitting grows linearly in number of missing data patterns and polynomially in sample-size. Even though we used the highly-performant C++ package [7], multiprocessing and multithreading, we could not match the near-constant sub-second runtime of gcimpute [13] on datasets with 1000 samples and dimensions ranging between 5 and 200. gcimpute has an edge over the present implementations of Vine-based methods because of the EM algorithm and because it uses optimized operations on the multivariate normal distribution. Vinecopulib [7] does not treat a Vine-tree sequence with all pair copulas restricted to the Gaussian copula, i.e. the multivariate Gaussian copula, as a special-case. Thus, it takes about $5 - 20s$ to fit a single Gaussian copula using [7] compared to the $0.1 - 0.5s$ implementation in [13]. We hypothesize that mini-batching can make copula fitting faster in increasing sample size, but this will not improve fit times in increasing dimension. Because Vine-copula based algorithms gain accuracy when they sequentially impute and refit, we will have to trade-off increased speed with lower accuracy if we decide to parallel process different missing data patterns.

References

[1] Tim Bedford and Roger M Cooke. Vines–a new graphical model for dependent random variables. *The Annals of Statistics*, 30(4):1031–1068, 2002.

[2] Claudia Czado. Analyzing dependent data with vine copulas. *Lecture Notes in Statistics, Springer*, 2019.

[3] Caren Hasler, Radu V Craiu, and Louis-Paul Rivest. Vine copulas for imputation of monotone non-response. *International Statistical Review*, 86(3):488–511, 2018.

[4] Harry Joe. Generating random correlation matrices based on partial correlations. *Journal of multivariate analysis*, 97(10):2177–2189, 2006.

[5] Harry Joe. *Dependence modeling with copulas.* CRC press, 2014.

[6] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

[7] Thomas Nagler and Thibault Vatter. vinecopulib: High performance algorithms for vine copula modeling in c++, 2017.

[8] Roger B Nelsen. *An introduction to copulas.* Springer Science & Business Media, 2007.

[9] Christian P Robert, George Casella, and George Casella. *Monte Carlo statistical methods*, volume 2. Springer, 1999.

[10] Alexander Sakuth. Identification of directly imputable missing data patterns using r-vine copulas and application to multiple imputation. 2016.

[11] Stef Van Buuren. *Flexible imputation of missing data.* CRC press, 2018.

[12] Stephan Zeisberger. Vine copula imputation. 2014.

[13] Yuxuan Zhao and Madeleine Udell. Missing value imputation for mixed data via gaussian copula. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 636–646, 2020.