Modelling Portfolios using High-Dimensional Conditional Nonparametric Vine Copulae

AALBORG UNIVERSITET

Master's Thesis, 10. semester

Thorkild Bendsen
Nick Kruse

Aalborg University
Faculty of Engineering and Science
Skjernvej 4A
DK-9220 Aalborg Øst

AALBORG UNIVERSITET
STUDENTERRAPPORT

Faculty of Engineering and Science
Mathematics
Skjernvej 4A
9220 Aalborg Øst
http://math.aau.dk

Title:

Modelling Portfolios using High-Dimensional Conditional Nonparametric Vine Copulae

Subtitle:

Vine Copulae

Project:

Master's Thesis

Period:

February 2023 - June 2023

Group:

4.106c

Participant:

Thorkild Bendsen
Nick Kruse

Supervisor:

Orimar Arregui Sauri

Page Count: 62
Completed d. 2. June 2023

Abstract:

Non-parametric copula estimators present a flexible way of modelling multivariate distributions. However, they are vulnerable to higher dimensions where they become very inefficient.

A way of combating the so called 'curse of dimensionality' is by combining *bivariate nonparametric copula estimators* with the well established *vine copula* to produce an estimator capable of efficiently estimating higher dimensions.

This thesis successfully applies nonparametric vine copulae to the problem of portfolio allocation to obtain portfolios that minimize the expected shortfall, where they are shown to outperform Gaussian copulae using the same marginals.

It is also shown that the nonparametric vine copula can be applied to arbitrarily high dimensional distributions with only a small decrease in efficiency.

Preface

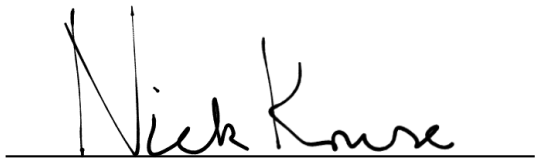
This master's thesis was written by Thorkild Bendsen & Nick Kruse for the final semester of *Matematics-Economics* at the Department of Mathematical Sciences, at Aalborg University. The thesis was made in the period from February to May 2023.

The thesis uses data from the Polygon API at `polygon.io`. The implementation was written in R and the code can be found in the authors Github `github.com/NickKruse18`. The two relevant repositories are `P10` which contains everything presented in the chapter, Data Analysis, and `kropula` which is an R package containing only the capability for modelling with the presented nonparametric vine copulae. The thesis is typeset in LaTeX and the figures are produced with `tikz`, `pgfplots`, `R` and `RStudio`.

Aalborg University, 2023.

Signature

Thorkild Bendsen tmbe21@student.aau.dk



Nick Kruse nkruse18@student.aau.dk

Contents

1 | Introduction

Copula models have become quite popular as a way of modelling the dependence between variables in a way that allows for more complexity than a conventional linear correlation. They also allow the variables to have atypical marginal distributions, which makes them a natural choice for modelling a portfolio of financial assets. The parametric copulae that are available, may however be too restrictive to match the observed data. Therefore, a kernel based nonparametric copula is considered as a remedy.

The challenge when using *kernel based estimators* is that more data is needed to obtain accurate estimates, when compared to parametric models. Additionally, kernel based estimators rapidly lose efficiency when modelling higher dimensional variables [Sto80]. This is especially a problem because portfolios tend to have a lot of assets. A remedy for this was proposed in [NC16], where the kernel based estimator was implemented in a vine copula.

The *vine copula* is a way of constructing a multidimensional copula as a large set of *bivariate copulae*. This reduces the flexibility of the copula to only consider the pairwise dependence between variables. The loss of flexibility has the benefit that only a set of bivariate copulae have to be estimated, which means it doesn't lose efficiency in higher dimensions. It is possible that limiting the copula to only pairwise dependencies may be unrealistic, but [NC16] shows that even in cases where the true copula can't be represented as a vine, a kernel based vine copula still outperforms its non-vine counterpart at higher dimensions for small sample sizes.

While copulae can be used to model the dependence between variables, they can also be used to model the auto-dependence as well, that is to say the dependence between a time series x_t and itself at a previous time, x_s when $s < t$, as can be seen in [GD22]. This is a case where nonparametric copulae are especially relevant, as the dependence tends to be far from linear.

The problem with the approach in [GD22] is that it only models a single asset. However, the inclusion of vines allows the modelling of an arbitrarily large amount of assets. A key feature of vines is also that with the right choice of vine it is possible to condition the assets on their previous values in a way that is efficient for higher dimensions.

Additionally, in [NC16] their tests didn't go beyond 10 dimensions. However, within areas like portfolio allocation there is need for dimensions much greater than 10. Thus the thesis also tests the efficiency of the nonparametric vine copula at dimensions up to 100.

1.1 Thesis Structure

The thesis consists of 5 chapters. *Chapter 2* starts with the basic theory of copulae, where Sklar's Theorem is the most important aspect. The focus is on the multidimensional case since portfolios require modeling multiple assets. The chapter also considers the probit functions use in kernel estimators for

copula density, which theoretically can handle the problem of boundary bias in the standard kernel estimator. It is also shown that kernel estimators become worse when estimating higher dimensional data.

Chapter 3 introduces vines as a way to combat kernel estimators worse performance in higher dimensions. This chapter begins with the definition of a vine and a regular vine. Later the canonical vine is specified since it is a vine that is easiest to implement. Additionally, the chapter presents how to estimate canonical vines as well, simulate them, both conditionally and unconditionally.

Chapter 4 applies the model to real one minute trading data from 2022 for a selection of twelve assets. The nonparametric vine copulae are applied to the data in a variety of ways along with the standard Gaussian copulae for comparison. The models are compared on the basis of minimizing the 5% expected shortfalls. Finally a synthetic test is conducted to see how well the nonparametric vine copula estimates higher dimensional Gaussian data up to a dimension of 100.

The thesis concludes on *Chapter 5* with a summary of the thesis and considerations for some of the shortfalls in application and potential extensions.

2 | Copula Theory

As Roger writes in [Nie06], copulae are functions that join or "couple" multivariate distribution functions to their one-dimensional marginal distribution functions. This can also be formulated as copulae are multivariate distribution functions whose one-dimensional margins are uniform on the interval $(0, 1)$. To give a formal definition of a copula, some preliminary results lead up to the definition of a copula. The focus is on the multivariate case.

2.1 Notation

The following is a list of Notations that need to be clarified.

- $\mathbb{R}$ denotes the ordinary real line $(-\infty, \infty)$, $\overline{\mathbb{R}}$ denotes the extended real line $[-\infty, \infty]$ and $\overline{\mathbb{R}}^n$ denote the extended n-space $\overline{\mathbb{R}} \times \overline{\mathbb{R}} \times \cdots \times \overline{\mathbb{R}}$.
- The vector notation for points in $\overline{\mathbb{R}}^n$ is for example $\mathbf{a} = (a_1, a_2, \cdots, a_n)$, and $\mathbf{a} \leq \mathbf{b}$ when $a_k \leq b_k$ for all k and $\mathbf{a} < \mathbf{b}$ when $a_k < b_k$ for all k.
- For $\mathbf{a} \leq \mathbf{b}$ let $[\mathbf{a}, \mathbf{b}]$ denote the n-box $B = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n]$, the Cartesian product of n closed intervals.
- The vertices of an n-box B are denoted as ver$B$ and contains all points $\mathbf{c} = (c_1, c_2, \cdots, c_n)$ where each c_k is equal to either a_k or b_k.
- $\mathbb{I}$ is the range $[0, 1]$ and $\mathbb{I}^n$ is the unit n-cube and is the product $\mathbb{I} \times \mathbb{I} \times \cdots \times \mathbb{I}$.
- An n-place real function H is a function whose domain, $\mathrm{Dom}(H)$, is a subset of $\overline{\mathbb{R}}^n$ and whose range, $\mathrm{Ran}(H)$, is a subset of $\mathbb{R}$.

2.2 Basic Definitions and Properties

The aspects presented in this section come from [Nie06] and will be relevant for defining the joint distribution functions for multivariate stochastic variables.

Definition 2.1
Let $S_1, S_2, \cdots S_n$ be nonempty subsets of $\overline{\mathbb{R}}$, and let H be an n-place real function such that $\mathrm{Dom}(H) = S_1 \times S_2 \times \cdots \times S_n$. Let $B = [\mathbf{a}, \mathbf{b}]$ be an n-box all of whose vertices are in $\mathrm{Dom}(H)$. Then the H volume of B is given by

$$V_H(B) = \sum_{\mathbf{c} \in \mathrm{ver} B} \mathrm{sgn}(\mathbf{c}) H(\mathbf{c}),$$

where the sum is taken over all vertices $\mathbf{c}$ of B, and $\mathrm{sgn}(\mathbf{c})$ is given by

$$\mathrm{sgn}(\mathbf{c}) = \begin{cases} 1, \text{ if } c_k = a_k \text{ for an even number of } k's, \\ -1, \text{ if } c_k = a_k \text{ for an odd number of } k' \text{ s.} \end{cases}$$

The H-volume of an n-box $B = [\mathbf{a}, \mathbf{b}]$ is equivalently the nth order difference of H on B such that

$$V_H(B) = \Delta_{\mathrm{a}}^{\mathrm{b}} H(\mathbf{t}) = \Delta_{a_n}^{b_n} \Delta_{a_{n-1}}^{b_{n-1}} \cdots \Delta_{a_2}^{b_2} \Delta_{a_1}^{b_1} H(\mathbf{t}),$$

where $\mathbf{t}$ is any element of $\mathbb{R}^n$ and where the first order difference of an n-place function is defined as

$$\Delta_{a_k}^{b_k} H(\mathbf{t}) = H\left(t_1, \cdots, t_{k-1}, b_k, t_{k+1}, \cdots, t_n\right) - H\left(t_1, \cdots, t_{k-1}, a_k, t_{k+1}, \cdots, t_n\right).$$

In the context of multivariate variables, if H was the joint distribution function of X then $V_H(B) = \mathbb{P}(X \in B)$. The volume $V_H(B)$ is here meant to describe a probability and for H to be a distribution function any volume $V_H(B)$ must always be non-negative.

Definition 2.2
An n-place real function H is *n-increasing* if $V_H(B) \geq 0$ for all n-boxes B whose vertices lie in $\mathrm{Dom}(H)$.

For H to be increasing is effectively a requirement for V_H to function like a measure for the multidimensional space $\mathrm{Dom}(H)$. Specifically, within the context of probability it should function as a probability measure for a multivariate stochastic variable. The relation to probability becomes more clear in the next definition.

Definition 2.3
Assume that H is an n-place real function with domain $\mathrm{Dom}(H) = S_1 \times S_2 \times \cdots \times S_n$, where each S_k has a least element a_k.

Then, H is said to be *grounded* if $H(\mathbf{t}) = 0$ for all $\mathbf{t}$ in $\mathrm{Dom}(H)$ such that $t_k = a_k$ for at least one k.

Additionally, if each S_k is nonempty and has a greatest element b_k, then H is said to have *margins*. Furthermore the *one-dimensional margins* of H are the functions H_k given by $\mathrm{Dom}(H_k) = S_k$ and

$$H_k(x) = H\left(b_1, \cdots, b_{k-1}, x, b_{k+1}, \cdots, b_n\right) \text{ for all } x \text{ in } S_k. \tag{2.1}$$

A nice consequence of H being grounded is that given a box $B = [\mathbf{a}, \mathbf{t}]$ where $\mathbf{a} = (a_1, \ldots, a_n)$. Then the volume $V_H([\mathbf{a}, \mathbf{t}]) = H(\mathbf{t})$, because the only vertex of B that doesn't contain at least one a_k is $\mathbf{t}$

The margins as defined in Definition 2.3 are directly related to the marginal distributions of some multivariate variable X. Such that, if H is the joint distribution of X, then the k'th variable X_k with support S_k has marginal distribution $F_k(x)$,

$$F_k(x) = H_k(x).$$

The concept of margins as presented also goes further than just one dimension as by fixing fewer places in H it is possible to define higher dimensional margins of H.

Lemma 2.4
Let $S_1, S_2, \cdots S_n$ be nonempty subsets of $\overline{\mathbb{R}}$, and let H be a *grounded n-increasing* function with $\mathrm{Dom}(H) = S_1 \times S_2 \times \cdots \times S_n$. Then H is nondecreasing in each argument, that is, if $B_x = (t_1, \cdots, t_{k-1}, x, t_{k+1}, \cdots, t_n)$ and $B_y = (t_1, \cdots, t_{k-1}, y, t_{k+1}, \cdots, t_n)$ are in $\mathrm{Dom}(H)$ and $x < y$, then

$$H(B_x) \leq H(B_y).$$

A formal proof will not be given, but the result is a straight forward consequence of H being n-increasing and $B_x \subsetneq B_y$.

The following lemma concerns a grounded n-increasing function with one-dimensional margins and is needed to show that n-copulae are uniformly continuous. This lemma is also important in the proof of the n-dimensional version of Sklar's theorem. For proof see [Sch82].

Lemma 2.5
Let $S_1, S_2, \cdots S_n$ be nonempty subsets of $\overline{\mathbb{R}}$, and let H be a grounded n-increasing function with one-dimensional margins whose domain is $\mathrm{Dom}(H) = S_1 \times S_2 \times \cdots \times S_n$. Let $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ and $\mathbf{y} = (y_1, y_2, \cdots, y_n)$ be any points in $\mathrm{Dom}(H)$. Then

$$|H(\mathbf{x}) - H(\mathbf{y})| \leq \sum_{k=1}^{n} |H_k(x_k) - H_k(y_k)|.$$

2.3 Multivariate Copulae

Using the theory related to the function H, it is possible to define n-dimensional subcopulae and n-dimensional copulae as

Definition 2.6 (Subcopula)
An *n-dimensional subcopula (or n-subcopula)* is a function C' with the following properties:

1. $\mathrm{Dom}(C') = S_1 \times S_2 \times \cdots \times S_n$, where each S_k is a subset of $\mathbb{I}$ containing 0 and 1;
2. C' is grounded and n-increasing;
3. C' has (one-dimensional) margins $C'_k, k = 1, 2, \cdots, n$, which satisfy

$$C'_k(u) = u \quad \text{for all } u \text{ in } S_k. \tag{2.2}$$

As a result of the third property, note that for every $\mathbf{u}$ in $\mathrm{Dom}(C'), 0 \leq C'(\mathbf{u}) \leq 1$, so that $\mathrm{Ran}(C')$ is also a subset of $\mathbb{I}$. Like the name implies a subcopula is exactly like a copula with the exception that its domain $\mathrm{dom}(C')$ is a subset of the domain of all copulae.

Definition 2.7 (Copula)
An *n-dimensional copula (or n-copula)* is an n-subcopula C whose domain is $\mathbb{I}^n$.

This definition of copulae uses several other definitions and they can all be reduced to the requirements:

1. C is a function from $\mathbb{I}^n$ to $\mathbb{I}$.
2. For every $\mathbf{u}$ in $\mathbb{I}^n$ where at least one coordinate of $\mathbf{u}$ is 0.
3. If all coordinates of $\mathbf{u}$ are 1 except u_k, then $C(\mathbf{u}) = u_k$;
4. For every $\mathbf{a}$ and $\mathbf{b}$ in $\mathbb{I}^n$ such that $\mathbf{a} \leq \mathbf{b}$,

$$V_C([\mathbf{a}, \mathbf{b}]) \geq 0.$$

It can be shown that for any n-copula C for $n \geq 3$, each k-margin of C is a k-copula for $2 \leq k < n$.

2.4 Sklar's Theorem

Sklar's theorem is the main result of copula theory and it says that a multivariate distribution function is represented as a composition of univariate marginal distribution functions and a copula, which describes the dependence structure between the variables.

First, the n-dimensional distribution function is defined.

Definition 2.8
An *n-dimensional distribution function* is a function H with domain $\overline{\mathbb{R}}^n$ such that

1. H is n-increasing,
2. $H(\mathbf{t}) = 0$ for all $\mathbf{t}$ in $\overline{\mathbb{R}}^n$ such that $t_k = -\infty$ for at least one k, and $H(\infty, \infty, \cdots, \infty) = 1$.

So H is grounded, and since $\mathrm{Dom}(H) = \overline{\mathbb{R}}^n$, it follows from Lemma 2.4 that the one-dimensional margins in (2.1) of an n-dimensional distribution function, H, are themselves the one-dimensional distribution functions $F_1, F_2, \cdots, F_n$ [Nie06].

Theorem 2.9 (Sklar's theorem)
Let H be an n-dimensional distribution function with one-dimensional margins $F_1, F_2, \cdots, F_n$. Then there exists an n-copula C such that for all $\mathbf{x}$ in $\overline{\mathbb{R}}^n$,

$$H(x_1, x_2, \cdots, x_n) = C(F_1(x_1), F_2(x_2), \cdots, F_n(x_n)). \tag{2.3}$$

If $F_1, F_2, \cdots, F_n$ are all continuous, then C is unique; otherwise, C is uniquely determined on $\mathrm{Ran}(F_1) \times \mathrm{Ran}(F_2) \times \cdots \times \mathrm{Ran}(F_n)$.
Conversely, if C is an n-copula and $F_1, F_2, \cdots, F_n$ are distribution functions, then the function H defined by (2.3) is an n-dimensional distribution function with one-dimensional margins $F_1, F_2, \cdots, F_n$.

The proof of Theorem 2.9 uses the two following lemmas. The first lemma is related to Theorem 2.9 but for subcopulae.

Lemma 2.10
Let H be an n-dimensional distribution function with one-dimensional margins $F_1, F_2, \cdots, F_n$. Then there exists a unique subcopula C' such that

1. $\mathrm{Dom}(C') = \mathrm{Ran}(F_1) \times \mathrm{Ran}(F_2) \times \cdots \times \mathrm{Ran}(F_n)$,
2. For all $\mathbf{x}$ in $\overline{\mathbb{R}}^n$, $H(\mathbf{x}) = C'\left(F_1\left(x_1\right), F_2\left(x_2\right), \cdots, F_n\left(x_n\right)\right)$.

The second lemma is the n-dimensional extension lemma, which says that every n-subcopula can be extended to a n-copula.

Lemma 2.11 (Extension Lemma)
Let C' be an n-subcopula. Then there exists a copula C such that,

$$C(\mathbf{u}) = C'(\mathbf{u})$$

for all $\mathbf{u}$ in $\mathrm{Dom}(C')$; i.e., any n-subcopula can be extended to a n-copula. The extension is generally non-unique.

See [Nie06] for proof of Lemma 2.10 and 2.11 in the 2-dimensional case.

Given Lemma 2.10 and 2.11, [Nie06] proves Sklar's theorem in Theorem 2.9 as follows:

Proof. The existence of a copula C such that (2.3) holds for all $\mathbf{x}$ in $\overline{\mathbb{R}}^n$ follows from Lemmas 2.10 and 2.11. If the one-dimensional margins $F_1, F_2, \cdots, F_n$ are continuous, then $\mathrm{Ran}(F_1) = \mathrm{Ran}(F_2) = \cdots = \mathrm{Ran}(F_n) = \mathbb{I}$, so that the unique n-subcopula in Lemma 2.10 is an n-copula. The converse is a matter of straightforward verification. □

Equation (2.3) can be inverted to express copulae in terms of a n-dimensional distribution function and the "inverses" of the one-dimensional margins. First one need to define "quasi-inverses" of distribution functions.

Definition 2.12 (Quasi-inverses)
Let F be a distribution function. Then a quasi-inverse of F is any function $F^{(-1)}$ with domain $\mathbb{I}$ such that

1. If t is in $\mathrm{Ran}(F)$, then $F^{(-1)}(t)$ is any number x in $\overline{\mathbb{R}}$ such that $F(x) = t$, i.e., for all t in $\mathrm{Ran}\, F$,
$$F\left(F^{(-1)}(t)\right) = t$$
2. If t is not in $\mathrm{Ran}(F)$, then
$$F^{(-1)}(t) = \inf\{x \mid F(x) \geq t\} = \sup\{x \mid F(x) \leq t\}$$
If F is strictly increasing, then it has a single quasi-inverse, which is the ordinary inverse which is notated. F^{-1}.

By using quasi-inverses from Definition 2.12 the following Corollary exists as an inversion of Theorem 2.9.

Corollary 2.13
Let $H, C, F_1, F_2, \cdots, F_n$ be as in Theorem 2.9 , and let $F_1^{(-1)}, F_2^{(-1)}, \cdots, F_n^{(-1)}$ be quasi-inverses

of $F_1, F_2, \cdots, F_n$, respectively. Then for any $\mathbf{u}$ in $\mathbb{I}^n$,

$$C(u_1, u_2, \cdots, u_n) = H\left(F_1^{(-1)}(u_1), F_2^{(-1)}(u_2), \cdots, F_n^{(-1)}(u_n)\right).$$

The next theorem is a repetition of Theorem 2.9, but instead in terms of random variables and their marginal distribution functions and their joint distribution function.

Theorem 2.14
Let $X_1, X_2, \cdots, X_n$ be random variables with distribution functions $F_1, F_2, \cdots, F_n$, respectively, and joint distribution function H. Then there exists an n-copula C such that (2.3) holds. If $F_1, F_2, \cdots, F_n$ are all continuous, C is unique. Otherwise, C is uniquely determined on $\mathrm{Ran}(F_1) \times \mathrm{Ran}(F_2) \times \cdots \times \mathrm{Ran}(F_n)$.

The n-copula C in Theorem 2.14 is called the n-copula of $X_1, X_2, \cdots, X_n$.

2.5 Basic Copulae

Three basic examples of copulae are,

Example 2.15 (The comonotonicity copula)
Let U be a random variable with a uniform distribution on $\mathbb{I}$ and consider the random vector consisting of n copies of U,

$$\mathbf{U} = (U, \ldots, U).$$

Then, for every $\mathbf{u} \in \mathbb{I}^n$,

$$\mathbb{P}(\mathbf{U} \leq \mathbf{u}) = \mathbb{P}(U \leq \min\{u_1, \ldots, u_n\}) = \min\{u_1, \ldots, u_n\}$$

and the distribution function $M^n : \mathbb{I}^n \to \mathbb{I}$ defined as

$$M^n(u_1, u_2, \ldots, u_d) := \min\{u_1, \ldots, u_d\}$$

is a copula, which is called the *comonotonicity copula.*

Example 2.16 (The independence copula)
Let $U_1, \ldots, U_n$ be independent random variables. Suppose each of U_i is uniformly distributed on $\mathbb{I}$ and consider the random vector, $\mathbf{U} = (U_1, \ldots, U_n)$. Then, for every $\mathbf{u} \in \mathbb{I}^n$,

$$\mathbb{P}(\mathbf{U} \leq \mathbf{u}) = \mathbb{P}(U_1 \leq u_1) \cdots \mathbb{P}(U_n \leq u_n) = \prod_{j=1}^{n} u_j$$

and the distribution function $\Pi^n : \mathbb{I}^n \to \mathbb{I}$ defined as

$$\Pi^n(u_1, u_2, \ldots, u_n) := \prod_{j=1}^{n} u_j$$

is a n-copula, which is called the *independence copula.*

Example 2.17 (The countermonotonicity copula)
Suppose U is uniformly distributed on $\mathbb{I}$ and consider the random vector $\mathbf{U} = (U, 1-U)$. Then, for every $\mathbf{u} \in \mathbb{I}^2$,

$$\mathbb{P}(\mathbf{U} \leq \mathbf{u}) = \mathbb{P}\left(U \leq u_1, 1-U \leq u_2\right) = \max\left\{0, u_1 + u_2 - 1\right\}.$$

and the distribution function $W^2 : \mathbb{I}^2 \to \mathbb{I}$ defined as

$$W^2\left(u_1, u_2\right) := \max\left\{0, u_1 + u_2 - 1\right\}$$

is a 2-copula, which is called the *countermonotonicity* copula.

Note that W^2 is only defined as a 2-copula and not as a n-copula. There is an analogous function $W^n : \mathbb{I}^n \to \mathbb{I}$ defined as

$$W^n(\mathbf{u}) := \max\left\{0, \sum_{j=1}^{n} u_j - (n-1)\right\}.$$

But this function fails to be an n-copula for any $n > 2$ because it is not necessarily n-increasing.

To summarize the three copulae, Π^n arises when the variables $X_1, \ldots X_n$ are all *mutually independent*.

M^n arises when $X_1, \ldots X_n$ are *perfectly positively correlated* in the sense that for any two variables X_k and $X_{k'}$, $k \neq k'$, then $X_k = f(X_{k'})$ for some strictly increasing deterministic function f.

W^n is only a valid copula for $n = 2$ and it arises when X_1, X_2 are *perfectly negatively correlated* in the sense that $X_2 = -f(X_1)$ for some strictly increasing deterministic function f. The intuitive reason for why W^n only works for $n = 2$ is that for $n = 3$ to be valid X_1, X_2 and X_3 would all have to be perfectly negatively correlated with each other at the same time.

The functions M^n and W^n have a special relation to copulae as can be seen by

Theorem 2.18 (Fréchet-Hoeffding bounds)
If C' is any n-subcopula, then for every $\mathbf{u}$ in $\mathrm{Dom}(C')$,

$$W^n(\mathbf{u}) \leq C'(\mathbf{u}) \leq M^n(\mathbf{u}). \tag{2.4}$$

A proof of Theorem 2.18 will not be given but it follows from Lemmas 2.4 and 2.5.

Since W^n represents variables that are perfectly negatively correlated (at least when $n = 2$) and M^n represents variables that are perfectly positively correlated, Theorem 2.18 can be seen as stating, that a copula (which specifies the dependence between variables) must be somewhere between perfect positive correlation and perfect negative correlation. Theorem 2.18 is therefore analogous to the Pearson correlation ρ having the bounds $-1 \leq \rho \leq 1$.

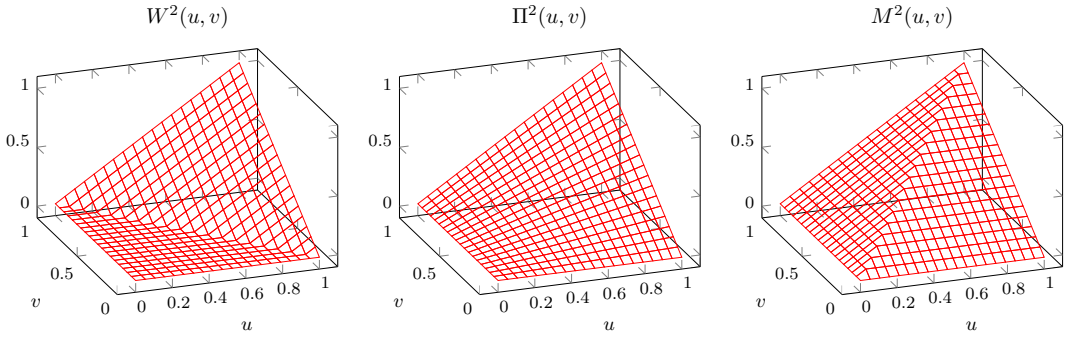


Figure 2.1: Plots of the W^2, Π^2 and M^2 copulae.

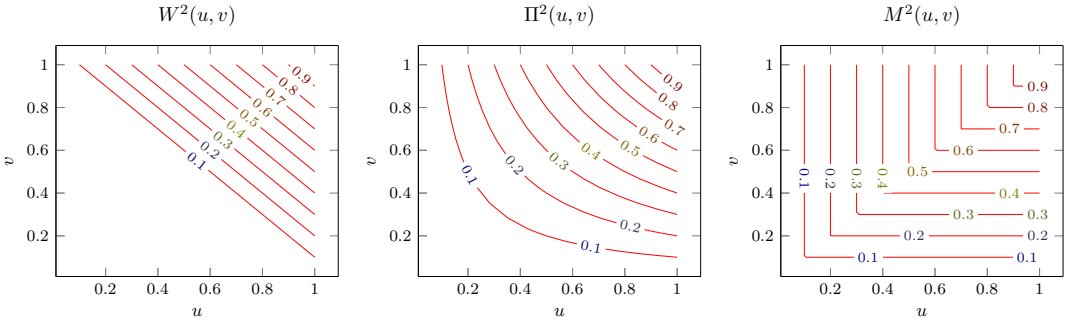


Figure 2.2: Contour plots of the W^2, Π^2 and M^2 copulae.

2.6 Tail Dependence

Copulae present a useful way of extending how variables can depend on each other. Conventionally, dependence is limited to linear dependence as implied by covariance and correlation.

One type of nonlinear dependence is *tail dependence*, which measures the dependence between two variables in the upper-right quadrant, λ_U, or in the lower-left quadrant, λ_L, of $\mathbb{I}^2$. This can be especially useful in risk management, where one is primarily concerned with worst case scenarios. Tail dependence tells if extreme events tend to occur simultaneously or if they occur independently. This is referred to as the absence or presence of tail dependence [MS03]. [Nie06] defines tail dependence as,

Definition 2.19 (Tail dependence)
Let X and Y be continuous random variables with distribution functions F_X and F_Y, respectively. The upper tail dependence parameter λ_U is the limit (if it exists) of the conditional probability that Y is greater than the 100-th percentile of F_Y given that X is greater than the 100-th percentile of F_X as t approaches 1 , i.e.

$$\lambda_U = \lim_{t \to 1^-} P\left[Y > F_Y^{(-1)}(t) \mid X > F_X^{(-1)}(t)\right]. \tag{2.5}$$

Similarly, the lower tail dependence parameter λ_L is the limit (if it exists) of the conditional probability that Y is less than or equal to the 100-th percentile of F_Y given that X is less than

or equal to the 100-th percentile of F_X as t approaches 0 , i.e.

$$\lambda_L = \lim_{t \to 0^+} P\left[Y \leq F_Y^{(-1)}(t) \mid X \leq F_X^{(-1)}(t)\right]. \tag{2.6}$$

(2.5) essentially states, that if X is above the t percentile, then the probability that Y is also above the t percentile will converge on λ_U as $t \to 1$.

Because tail dependence is defined using the percentile t, it doesn't actually depend on the marginals of X and Y and only on their copula C. This is illustrated in the following theorem.

Theorem 2.20
Let $X, Y, F_X, F_Y, \lambda_U$, and λ_L be as in Definition 2.19, and let C be the copula of X and Y. If the limits in (2.5) and (2.6) exist, then

$$\lambda_U = 2 - \lim_{t \to 1^-} \frac{1 - C(t,t)}{1-t} \tag{2.7}$$

and

$$\lambda_L = \lim_{t \to 0^+} \frac{C(t,t)}{t}. \tag{2.8}$$

The proof of Theorem 2.20 can be found in [Nie06].

In (2.7) the copula C has an upper tail dependence coefficient equal to λ_U and in (2.8) the copula C has an lower tail dependence coefficient equal to λ_L.

If λ_U lies in $(0, 1]$, then the copula C has upper tail dependence and large events tend to occur simultaneously. If $\lambda_U = 0$, then the copula C has no upper tail dependence and large events tend to occur essentially independently. Similarly can be said for λ_L.

2.7 Gaussian Copula

One of the most well-known copula families is the *Gaussian family of copulae*. It is the traditional candidate for modelling dependence and it is the copula that comes from the multivariate Gaussian distribution. [MS03] defines the Gaussian copula as,

Definition 2.21
Let Φ denote the standard normal (cumulative) distribution function and $\Phi_{\rho,n}$ the n-dimensional Gaussian distribution funciton with correlation matrix ρ. Then, the Gaussian n-copula with correlation matrix ρ is

$$C_\rho(u_1, \ldots, u_n) = \Phi_{\rho,n}\left(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_n)\right), \tag{2.9}$$

The Gaussian n-copula has density

$$c_\rho(u_1, \ldots, u_n) = \frac{\partial C_\rho(u_1, \ldots, u_n)}{\partial u_1 \ldots \partial u_n} = \frac{1}{\sqrt{\det \rho}} \exp\left(-\frac{1}{2} y(u)^t \left(\rho^{-1} - \mathrm{I}\right) y(u)\right)$$

where $y_k(u) = \Phi^{-1}(u_k)$. Note that it is Theorem 2.9 and Corollary 2.13 that ensures the Gaussian n-copula in (2.9) is a copula.

When $n = 2$ the ρ simplifies down to the coefficient of correlation between $\Phi^{-1}(u_1)$ and $\Phi^{-1}(u_2)$. Additionally the Gaussian 2-copula becomes the countermonotonic copula, independence copula, and comonotonic copula are obtained when $\rho = -1$, 0 and 1, respectively.

The Gaussian copula has zero tail dependence $\lambda_U = \lambda_L = 0$ when $\rho \in (-1, 1)$. To see this consider a bivariate Gaussian copula $C_\rho(\cdot)$. The result will only consider the lower tail dependence λ_L. The proof for λ_U can then be argued from the symmetry of Gaussian distributions.

Assume that X, Y have uniform marginals on $\mathbb{I}$ with Gaussian copula C_ρ. Then by applying l'Hospital's rule to (2.8)

$$\lambda_L = \lim_{t \to 0^+} \frac{\partial C_\rho(t,t)}{\partial t} = \lim_{t \to 0^+} P\left(Y \leq t \mid X = t\right) + \lim_{t \to 0^+} P\left(X \leq t \mid Y = t\right) \tag{2.10}$$

[EMS02]. Since a Gaussian copula is an exchangeable copula, that is $C_\rho(X, Y) = C_\rho(Y, X)$, then

$$\lambda_L = 2 \lim_{t \to 0^+} P\left(Y \leq t \mid X = t\right)$$

This limit can be evaluated by applying the same quantile distribution to both marginals in order to obtain a bivariate distribution where the conditional distribution is known. Let,

$$(S, T) = \left(\Phi^{-1}(X), \Phi^{-1}(Y)\right)$$

So (S, T) has a bivariate normal distribution with standardized marginals and correlation ρ. Then,

$$\begin{aligned} \lambda_L &= 2 \lim_{t \to 0^+} P\left(\Phi^{-1}(Y) \leq \Phi^{-1}(t) \mid \Phi^{-1}(X) = \Phi^{-1}(t)\right) \\ &= 2 \lim_{\tau \to -\infty} P\left(T \leq \tau \mid S = \tau\right) \end{aligned}$$

Using the fact that $T \mid S \sim \mathcal{N}\left(\rho\tau, 1 - \rho^2\right)$ it can be calculated that,

$$\lambda_L = 2 \lim_{\tau \to -\infty} \Phi\left(\tau \sqrt{\frac{(1-\rho)}{(1+\rho)}}\right) = 0$$

Thus the Gaussian copula gives an asymptotic independence provided that $\rho \in (-1, 1)$. So regardless of how high the correlation is, there is still zero tail dependence. This does bring up an important distinction that tail dependence doesn't mean that X and Y are independent in the tails.

The Gaussian copula for d dimensions is entirely determined by the correlation matrix ρ. ρ is also simple to estimate regardless of how large d is, because it only depends on the correlation between each pair of variables.

2.8 Kernel Based Estimators

In copula modeling, there is a general problem of estimating the copula density in a nonparametrically way. The kernel estimator, which is a nonparametric density estimator, is not suitable for the support of unit-square copula densities. There are two major problems associated with the kernel estimator. One of the problems is that the kernel estimator is biased at the boundaries and the kernel estimator is also sensitive to both kernel and bandwidth.

Consider a bivariate copula function C, for the random variables X and Y with marginal cdf's F_X and F_Y. Assume that C is absolutely continuous and that the associated copula density of C is

$$c(u, v) = \frac{\partial^2 C}{\partial u \partial v}(u, v) \quad \text{for } (u, v) \in \mathbb{I}. \tag{2.11}$$

Let the standard kernel estimator for c be $\hat{c}^*$ at $(u,v) \in \mathbb{I}$. Given a sample $\{(U_i, V_i)\}_{i=1}^n$ where $U_i = F_X(X_i)$ and $V_i = F_Y(Y_i)$, then $\hat{c}^*$ is defined in [GCP17] as

$$\hat{c}^*(u,v) = \frac{1}{n\left|H_{UV}\right|^{1/2}} \sum_{i=1}^{n} K\left(H_{UV}^{-1/2} \begin{pmatrix} u - U_i \\ v - V_i \end{pmatrix}\right). \tag{2.12}$$

Where the kernel function $K : \mathbb{R}^2 \to \mathbb{R}$ has that $\int_{\mathbb{R}^2} K(x)dx = 1$ and where H_{UV} is a bandwidth matrix. Thus the density of $\hat{c}^*$ at a given point (u,v) will be determined by how many points from the sample are 'close' to it, with the bandwidth matrix, H_{UV}, specifying what is considered close.

There are however issues with the estimator in (2.12), as explained in [GCP17]. One is that kernel estimators are not consistent for unbounded densities, which means (2.12) will be inconsistent for many theoretical densities like the Gaussian kernel which for most choices of correlation ρ will have at least two corners with infinite density. This is however not necessarily an issue when estimating real data.

A much more significant problem with (2.12) is boundary bias. That is to say that along the boundaries of $\mathbb{I}^2$ the estimator will be inconsistent with $\mathbb{E}\left[\hat{c}^*(u,v)\right] = \frac{1}{2}c(u,v) + O(h)$ on the edges and $\mathbb{E}\left[\hat{c}^*(u,v)\right] = \frac{1}{4}c(u,v) + O(h)$ in corners. The reason for this is simply that when estimating $c(u,v)$ the kernel estimator uses how many points in the sample that are in area close to the point (u,v) to make its estimate. However at the edges of $\mathbb{I}^2$ the area that is would be considered close is only half what it would be closer to the center. Likewise at the corners the area is only a quarter.

2.8.1 Probit Transformation

The aim is to come up with a kernel-type copula density estimator that can handle the problems with the estimator in (2.12). The idea is to transform the uniform marginals of the copula density estimator into normal distributions by using the probit function Φ^{-1}. Then estimate the density in the transformed $\mathbb{R}^2$ domain and get an estimate of the copula density through back-transformation.

Define $S = \Phi^{-1}(U)$ and $T = \Phi^{-1}(V)$ and let F_{ST} be their joint cdf, where Φ is the standard normal cdf and Φ^{-1} is the quantile (probit) function. The main idea is summarized in the following way:

- If $c(u,v) > 0$ Lebesgue-almost everywhere over $\mathcal{I}$, then (S,T) has unconstrained support $\mathbb{R}^2$.
- Under mild assumptions the density of c and its first- and second-order partial derivatives can be seen as being uniformly bounded on $\mathbb{R}^2$. This is also the case with an unbounded copula density c.
- When estimating the density of c it can not suffer from boundary issues and because it has normal margins it is expected that the density of c is very smooth and well-behaved, which makes the estimation easy.

Now let C be the copula of F_{ST}. Given that both U and V are uniform on $\mathbb{I}$ then $S \sim \mathcal{N}(0,1)$ and $T \sim \mathcal{N}(0,1)$. For the vector (S,T) using Sklar's theorem one has that

$$F_{ST}(s,t) = \mathbb{P}(S \leq s, T \leq t) = C(\Phi(s), \Phi(t)) \quad \forall (s,t) \in \mathbb{R}^2 \tag{2.13}$$

Differentiating with respect to s and t, one gets

$$f_{ST}(s,t) = c(\Phi(s), \Phi(t))\phi(s)\phi(t) \tag{2.14}$$

Inverting (2.14) the following expression is obtained for any $(u,v) \in (0,1)^2$,

$$c(u,v) = \frac{f_{ST}\left(\Phi^{-1}(u), \Phi^{-1}(v)\right)}{\phi\left(\Phi^{-1}(u)\right)\phi\left(\Phi^{-1}(v)\right)}.$$

This shows that any estimator $\hat{f}_{ST}$ of f_{ST} on $\mathbb{R}^2$ does produce an estimator of the copula density on $\text{int}(\mathcal{I})$:

$$\hat{c}^{(\tau)}(u,v) = \frac{\hat{f}_{ST}\left(\Phi^{-1}(u), \Phi^{-1}(v)\right)}{\phi\left(\Phi^{-1}(u)\right)\phi\left(\Phi^{-1}(v)\right)}. \tag{2.15}$$

The superscript (τ) in (2.15) is referring to the idea of transformation. It is possible to define $\hat{c}^{(\tau)}$ at the boundaries of $\mathcal{I}$ by continuity when it is necessary.

The transformation-based estimator in (2.15) has some nice properties. The first one is that $\hat{c}^{(\tau)}$ can not allocate any probability outside of $\mathcal{I}$, because $\left(\Phi^{-1}(u), \Phi^{-1}(v)\right)$ is not defined for $(u,v) \notin \mathcal{I}$ Secondly, if $\hat{f}_{ST}$ is a bona fide density function, that is $\hat{f}_{ST}(s,t) \geq 0$ for all (s,t) and

$$\iint_{\mathbb{R}^2} \hat{f}_{ST}(s,t)dsdt = 1,$$

then by a change of variables $u = \Phi(s)$ and $v = \Phi(t)$, $\hat{c}^{(\tau)}(u,v) \geq 0$ for all $(u,v) \in \mathcal{I}$ and

$$\iint_{\mathcal{I}} \hat{c}^{(\tau)}(u,v)dudv = 1.$$

Lastly, if $\hat{f}_{ST}$ is uniformly weak or uniformly strong consistent estimator for f_{ST}, that is,

$$\sup_{(s,t)\in\mathbb{R}^2} \left|\hat{f}_{ST}(s,t) - f_{ST}(s,t)\right| \overset{\mathbb{P}-a.s.}{\longrightarrow} 0 \quad \text{as } n \to \infty,$$

So $\hat{c}^{(\tau)}$ has that uniform consistency on any compact proper subset of $\mathcal{I}$.

2.8.2 The Naive Estimator

The most basic application of the standard kernel estimator (2.12) and the probit transformation is to use (2.12) as the estimate $\hat{f}_{ST}$ in (2.15):

$$\hat{f}^{*}_{ST}(s,t) = \frac{1}{n\left|H_{ST}\right|^{1/2}} \sum_{i=1}^{n} K\left(H_{ST}^{-1/2}\begin{pmatrix} s - S_i \\ t - T_i \end{pmatrix}\right) \tag{2.16}$$

where (S_i, T_i) is the probit transformed sample $\left(\Phi^{-1}(U_i), \Phi^{-1}(V_i)\right)$. However, (U_i, V_i) are only available if the marginal cdfs F_X and F_Y are known. The more feasible case is that the samples are estimated as $\hat{U}_i = \hat{F}_X(X_i)$ and $\hat{V}_i = \hat{F}_Y(Y_i)$, which are then transformed into $\hat{S}_i = \Phi^{-1}(\hat{U}_i)$ and $\hat{T}_i = \Phi^{-1}(\hat{V}_i)$. The feasible version of (2.16) is therefore,

$$\hat{f}_{ST}(s,t) = \frac{1}{n\left|H_{ST}\right|^{1/2}} \sum_{i=1}^{n} K\left(H_{ST}^{-1/2}\begin{pmatrix} s - \hat{S}_i \\ t - \hat{T}_i \end{pmatrix}\right).$$

By using (2.15), this leads to the following *probit transformation kernel copula density estimator*:

$$\hat{c}^{(\tau)}(u,v) = \frac{1}{n\left|H_{ST}\right|^{1/2}\phi\left(\Phi^{-1}(u)\right)\phi\left(\Phi^{-1}(v)\right)} \sum_{i=1}^{n} K\left(H_{ST}^{-1/2}\begin{pmatrix} \Phi^{-1}(u) - \Phi^{-1}\left(\hat{U}_i\right) \\ \Phi^{-1}(v) - \Phi^{-1}\left(\hat{V}_i\right) \end{pmatrix}\right) \tag{2.17}$$

The estimator in (2.17) is in [Gee14] called the *Naive Estimator* because it fails to produce good results close to the borders although it was designed to fix boundary issues. This is illustrated later in an example.

2.8.3 Asymptotic Properties of the Naive Estimator

For simplicity it is assumed that K is a product Gaussian kernel, that is, $K(z_1, z_2) = \phi(z_1)\phi(z_2)$, and $H_{ST} = h^2\mathbf{I}$ for some $h > 0$. The naive estimator in (2.17) is then reduced to,

$$\begin{aligned}\hat{c}^{(\tau)}(u,v) =& \frac{1}{nh^2\phi\left(\Phi^{-1}(u)\right)\phi\left(\Phi^{-1}(v)\right)} \\ &\times \sum_{i=1}^{n} \phi\left(\frac{\Phi^{-1}(u) - \Phi^{-1}\left(\hat{U}_i\right)}{h}\right)\phi\left(\frac{\Phi^{-1}(v) - \Phi^{-1}\left(\hat{V}_i\right)}{h}\right)\end{aligned} \tag{2.18}$$

Given (2.15) the statistical properties does depend on those in (2.16), where

$$\hat{f}_{ST}(s,t) = \frac{1}{nh^2}\sum_{i=1}^{n}\phi\left(\frac{s-\hat{S}_i}{h}\right)\phi\left(\frac{t-\hat{T}_i}{h}\right). \tag{2.19}$$

If f_{ST} has continuous second-order partial derivatives, then expressions are known for the bias and the variance of the infeasible estimator $\hat{f}^*_{ST}$ in (2.16). This is also the case for this estimator's asymptotic normality.

In [GCP17] it is proven that when (u, v) is near one of the boundaries, then the bias and variance of $\hat{c}^{(\tau)}(u, v)$ tends to grow unboundedly. In fact along the boundaries, $\hat{c}^{(\tau)}$ is only consistent over areas where c approaches 0 smoothly. Instead $\hat{c}^{(\tau)}$ will have have a lot of volatility and a large positive bias, especially in the corners. To alleviate some of the boundary problems an adjustment can be made to (2.15) as

$$\begin{aligned}\hat{c}^{(\tau\ \mathrm{am})}(u,v) =& \frac{\hat{f}_{ST}\left(\Phi^{-1}(u), \Phi^{-1}(v)\right)}{\phi\left(\Phi^{-1}(u)\right)\phi\left(\Phi^{-1}(v)\right)} \\ &\times \frac{1}{1 + (h^2/2)\left(\left\{\Phi^{-1}(u)\right\}^2 + \left\{\Phi^{-1}(v)\right\}^2 - 2\right)}.\end{aligned} \tag{2.20}$$

This is the 'amended' version of $\hat{c}^{(\tau)}$ where the included factor will reduce the density at the boundaries. The factor will also disappear as h goes to 0. Even this doesn't fix the problems completely. This is illustrated by the example from [GCP17] where a sample size of $n = 1000$ from the Gaussian copula with correlation $\rho = 0.3$ is fitted using both the naive and the amended estimators, see Figure 2.3.

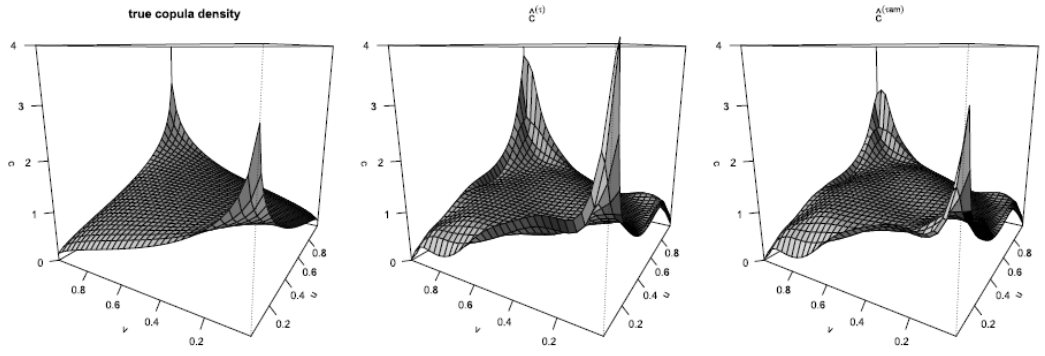


Figure 2.3: The true Gaussian copula with $\rho = 0.3$ on the left panel, the naive estimator is in the middle and the amended naive estimator is on the right panel [GCP17].

The figure shows that the naive estimator works decently in the center of $\mathbb{I}$, but towards the boundaries, the estimate becomes more erratic with way too high peaks at $(0,0)$ and $(1,1)$. The rightmost panel shows an improvement to the middle panel with the peaks at $(0,0)$ and $(1,1)$ being closer in level to the true ones in the left panel. But the wiggly appearance of the estimate along the boundaries remains because the large border variance isn't affected by the amendment.

In [Gee14] it was explained that the back-transformation to the (U,V)-domain through (2.15) that yields the result in (2.17) caused the poor fit of the naive estimator. Thus [GCP17] also includes an improved version which has much better results, though it won't be covered here.

Overall, kernel based estimators are an excellent way of giving the copula flexibility, which can also be very needed, however, they suffer greatly from the so called, *curse of dimensionality*, as seen in the next section.

2.9 Convergence Rate for Kernel Based Estimators

This section is based on [Sto80] and its purpose is to explain the best possible or *optimal* rate of convergence for nonparametric estimators.

Let Θ be a collection of probability density functions on a fixed subset of $\mathbb{R}^d$ with $\theta \in \Theta$. Consider a sequence of estimators, $\left\{\hat{T}_n(\theta)\right\}$ based on a random sample of size n from the distribution θ. Now $r > 0$ is called an *upper bound to the rate of convergence* if for every sequence of estimators $\left\{\hat{T}_n(\theta)\right\}$,

$$\liminf_n \sup_{\theta\in\Theta} P_\theta\left(\left|\hat{T}_n(\theta) - \theta\right| > cn^{-r}\right) > 0 \quad \text{for all c } > 0 \tag{2.21}$$

and

$$\lim_{c\to 0} \liminf_n \sup_{\theta\in\Theta} P_\theta\left(\left|\hat{T}_n(\theta) - \theta\right| > cn^{-r}\right) = 1. \tag{2.22}$$

Here (2.21) effectively states that the members of Θ that $\left\{\hat{T}_n(\theta)\right\}$ are worst at estimating are in the best case, still likely to be more than cn^{-r} away of the true value. Likewise (2.22) states that worst members of Θ will always be more than cn^{-r} away from the true value, when c approaches 0. Thus its an *upper bound* for what can be expected.

The parameter r is instead called an *achievable rate of convergence* if there is a sequence $\left\{\hat{T}_n(\theta)\right\}$ of estimators so that

$$\lim_{c\to\infty} \limsup_n \sup_{\theta\in\Theta} P_\theta\left(\left|\hat{T}_n(\theta) - \theta\right| > cn^{-r}\right) = 0. \tag{2.23}$$

Thus for r to be achievable the worst members of Θ for $\hat{T}_n$ to estimate are at worst within cn^{-1} when c goes to ∞. Notice the contrast with (2.22) where *at best* the estimator will be further than cn^{-1} from the true value when c goes to 0.

Furthermore, r is also called the *optimal rate of convergence* if it is both an upper bound to the rate of convergence and an achievable rate of convergence. It is worth noting that if r is an upper bound to the rate of convergence and q is an achievable rate of convergence, then $q \leq r$.

The optimal rate of convergence for an estimator for a probability density function θ requires some assumptions about θ that are based on Taylor polynomials. So, let $\alpha = (\alpha_1, \cdots, \alpha_d)$ be a d-tuple of

non-negative integers and set $|\alpha| = \alpha_1 + \cdots + \alpha_d$ and $\alpha! = \alpha_1! \cdot \cdots \cdot \alpha_d!$. For $x = (x_1, \cdots, x_d) \in \mathbb{R}^d$ set $x^\alpha = x_1^{\alpha_1} \cdot \cdots \cdot x_d^{\alpha_d}$. Then the differential operator, $D^{\alpha\alpha}$, is defined by

$$D^\alpha = \frac{\partial^{\alpha_1 + \cdots + \alpha_d}}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}} \tag{2.24}$$

Let k be a non-negative integer and let $\mathcal{C}_k$ be the collection of k times continuously differentiable real-valued functions g on $\mathbb{R}^d$. Given $g \in \mathcal{C}_k$, its Taylor polynomial g_k of degree k is defined by

$$g_k(x) = \sum_{|\alpha| \le k} \frac{1}{\alpha!} D^\alpha g(0) x^\alpha.$$

Now let $p > k$ and M be positive constants and let U be an open neighborhood of the origin of $\mathbb{R}^d$. Let $\mathcal{G}_p^k$ be the collection of functions $g \in \mathcal{C}_k$ so that

$$|g(x) - g_k(x)| \le M\|x\|^p, \quad x \in U, \tag{2.25}$$

where $\|x\| = \left(x_1^2 + \cdots + x_d^2\right)^{1/2}$.

Definition 2.22 (Unknown density function)
Let θ_0 be a fixed probability density function in $\mathcal{C}_m$ such that $\theta_0(0) > 0$ and $0 \le m \le k$. Set $\Theta = \left\{\theta_0(1+g) : g \in \mathcal{G}_p^k, |g| \le 1 \text{ on } \mathbb{R}^d \text{ and } \int \theta_0 g dx = 0\right\}$. Let $\hat{T}_n(\theta)$ be an estimator of $\theta \in \Theta$ based on $X_1,\ X_2, \cdots, X_n$ iid. $\mathbb{R}^d$-valued variables with density θ.

The optimal rate of convergence for a density function as described in Definition 2.22 is given in the following theorem,

Theorem 2.23
Let Definition 2.22 hold. Then $r = p/(2p + d)$ is the optimal rate of convergence.

The procedure to prove Theorem 2.23 is by showing that r is an upper bound to the rate of convergence. This is done by showing that (2.21) and (2.22) hold. Next, showing that r is an achievable rate of convergence is done by constructing a sequence $\left\{\hat{T}_n(\theta)\right\}$ of estimators of θ so that (2.23) holds. The details of the proof can be found in section 3 in [Sto80].

Theorem 2.23 shows the problem with kernel based estimators is that when they are estimating d-dimensional density functions, that follow Definition 2.22, the optimal convergence rate r will decrease. This is also ignoring that r is the best case scenario. Therefore, kernels become impractical for higher dimensions. However, the next chapter shows that by using vine it is possible to model higher dimensional variables using only 2-dimensional kernel based estimators.

3 | Vines

In the context of copulae, *vines* are used to specify a d-copula C as a collection of smaller 2-dimensional copulae. The idea of using bivariate copulae to construct a d-copula was first introduced in [Joe96] and the actual term *vine* was coined in [BC02].

The idea behind vines can be seen as a natural extension to the prevalence of covariance. In that the typical focus when regarding multivariate distributions is on the pairwise interaction between variables, with covariance abbreviating that interaction as a single number. The vine instead seeks to replace that single number as a descriptor for pairwise interaction with a bivariate copula capable of much greater nuisance.

Assume that $u = (u_1,\ u_2, \ldots, u_d)$ is some d-dimensional distribution with uniform marginals on $[0, 1]$. A simple way of constructing a valid d-copula C from 2-dimensional copulae, would be by finding the copulae between pairs of variables $C_{1,2},\ C_{2,3},\ C_{3,4}, \ldots,\ C_{d-1,d}$, where say, $C_{1,2}$ specifies the copula between u_1 and u_2. Then C can be defined as

$$C(u_1, \ldots,\ u_d) = \int_0^{u_1} \cdots \int_0^{u_d} \prod_{i=1}^{d-1} c_{i,i+1}(v_i,\ v_{i+1})\ dv_d \cdots dv_1$$

where $c_{i,i+1}$ is the pdf of $C_{i,i+1}$. Thus C will now accurately specify the dependence between u_1 and u_2; u_2 and u_3; u_3 and u_4; and so on. As can be seen by

$$\begin{aligned} C(u_1,\ u_2,\ 1, \ldots,\ 1) &= \int_0^{u_1} \int_0^{u_2} c_{1,2}(v_1,\ v_2) \int_0^1 \cdots \int_0^1 \prod_{i=2}^{d-1} c_{i,i+1}(v_i,\ v_{i+1})\ dv_d \cdots dv_3\ dv_2\ dv_1 \\ &= \int_0^{u_1} \int_0^{u_2} c_{1,2}(v_1,\ v_2) \int_0^1 \cdots \int_0^1 \prod_{i=2}^{d-2} c_{i,i+1}(v_i,\ v_{i+1})\ dv_{d-1} \cdots dv_3\ dv_2\ dv_1 \\ &= \int_0^{u_1} \int_0^{u_2} c_{1,2}(v_1,\ v_2)\ dv_2\ dv_1 = C_{1,2}(u_1,\ u_2) \end{aligned}$$

Where $\int_0^1 c_{i,i+1}(v_i,\ v_{i+1})\ dv_{i+1} = 1$, since that is the marginal pdf for u_i. This way of specifying C is a subset of what is known as a *bivariate tree specification* [BC02]. In this specification the bivariate copulae are visualized as the edges of a tree.

Definition 3.1 (Tree)
A *Tree* $T = \{N, E\}$ is an acyclic graph, where N is its set of nodes, and E is its set of edges (unordered pairs of nodes).

Here the set of nodes N will be a set of arbitrary elements $N = \{a, b, c, \ldots\}$. A single edge $e \in E$ is denoted as $e = \{a, b\}$, if it connects the elements a and b from the set of nodes N.

For a graph to be acyclic means that starting at any node $a \in N$, and then following a path along the edges, without following the same edge twice, it is impossible to end up at the starting node a. That is to say there are no 'cycles' or 'rings' in a tree.

The tree in the case of C has nodes representing each variable $u_1, \ldots, u_d$, $N = \{1, \ldots, d\}$ and edges representing each bivariate copula $C_{1,2}, C_{2,3}, \ldots, C_{d-1,d}$, with $E = \{\{1, 2\}, \{2, 3\}, \ldots, \{d-1, d\}\}$.

The issue with the tree specification is that it doesn't specify the dependence between u_1 and u_3, except implicitly through the dependence both have with u_2. The lacking dependence cannot be remedied by simply including $C_{1,3}(u_1, u_3)$ as the edge $\{1, 3\}$, because this would create a cycle in the tree T. The reason this is a problem is that $C_{1,3}$ will interfere with the already present dependencies $C_{1,2}(u_1, u_2)$ and $C_{2,3}(u_2, u_3)$, which will result in C having none of the three.

Instead the dependence between u_1 and u_3 can be specified such as to not interfere with $C_{1,2}$ and $C_{2,3}$ by using the copula $C_{1,3|2}(F(u_1|u_2), F(u_3|u_2))$, that is to say the dependence between u_1 and u_3 given u_2. To see this consider the joint pdf of $u_1, \ldots, u_3$ as

$$f(u_1, u_2, u_3) = f(u_3| u_2, u_1) \cdot f(u_2| u_1) \cdot f(u_1).$$

Since u_1 is uniform $f(u_1) = 1$. Additionally,

$$f(u_2| u_1) = \frac{f(u_2, u_1)}{f(u_2)} = c_{1,2}(u_1, u_2),$$

$$\begin{aligned} f(u_3| u_2, u_1) = \frac{f(u_3, u_1| u_2)}{f(u_1| u_2)} &= \frac{c_{1,3|2}(F(u_1|u_2), F(u_3|u_2))\, f(u_1| u_2)\, f(u_3| u_2)}{f(u_1| u_2)} \\ &= c_{1,3|2}(F(u_1|u_2), F(u_3|u_2))\, c_{2,3}(u_2, u_3). \end{aligned}$$

Thus in combination

$$f(u_1, u_2, u_3) = c_{1,3|2}(F(u_1|u_2), F(u_3|u_2))\, c_{2,3}(u_2, u_3)\, c_{1,2}(u_1, u_2).$$

This can also be used for d-dimensional variables to extend the standard bivariate tree specification given for C with $d-2$ additional bivariate dependencies on the form

$$C_{i,i+2|i+1}(F(u_i|u_{i+1}), F(u_{i+2}|u_{i+1})), \text{ for } i = 1, \ldots, d-2$$

If C is equipped with these conditional copulae then it is on a *bivariate vine specification*, where a *vine* is defined as

Definition 3.2 (Vine)
$\mathcal{V}$ is a *vine* on n elements if:

1. $\mathcal{V} = (T_1, \ldots, T_m)$.
2. T_1 is a tree with nodes $N_1 = \{1, \ldots, n\}$ and a set of edges denoted E_1.
3. For $i = 2, \ldots, m$, T_i is a tree with nodes $N_i \subset N_1 \cup E_1 \cup E_2 \cup \cdots \cup E_{i-1}$ and edge set E_i.

The vine in the case of C then consists of two trees $\mathcal{V} = \{T_1, T_2\}$, where $T_1 = \{N_1, E_1\}$ is the first tree used for C and contains the $d-1$ copulae $C_{1,2}, C_{2,3}, \ldots, C_{d,d-1}$.

The second tree $T_2 = \{N_2, E_2\}$, contains the $d-2$ copulae $C_{1,3|2}, C_{2,4|3}, \ldots, C_{d-2,d|d-1}$. The nodes are the edges of T_1, $N_2 = E_1$, and the set of edges are

$$E_2 = \{\{\{1, 2\}, \{2, 3\}\}, \{\{2, 3\}, \{3, 4\}\}, \ldots, \{\{d-2, d-1\}, \{d-1, d\}\}\}.$$

Thus the edge $\{\{1,\ 2\},\ \{2,\ 3\}\}$ corresponds to the copula $C_{1,3|2}$, where the conditioning on 2 comes from 2 being present in both the nodes that the edge connects. How exactly an edge relates to a bivariate copula is given later in Definition 3.5.

This vine is still lacking some pairwise dependencies like u_1 and u_4. Thus additional trees can be added to $\mathcal{V}$ to reach this pair along with every other missing pair. For these additional trees to properly represent the missing pairs, they will need to be specified in such a way that the vine $\mathcal{V}$ will turn into a *regular vine*.

Definition 3.3 (Regular Vine)
$\mathcal{V}$ is a *regular vine* on n elements if:

1. $\mathcal{V} = (T_1, \ldots, T_n)$.
2. T_1 is a tree with nodes $N_1 = \{1, \ldots, n\}$ and a set of edges denoted E_1.
3. For $i = 2, \ldots, n$, T_i is a tree with nodes $N_i = E_{i-1}$ and edge set E_i.
4. The proximity condition holds: for $i = 2, \ldots, n-1$, if $a = \{a_1,\ a_2\}$ and $b = \{b_1,\ b_2\}$ are two nodes in N_i connected by an edge, then $\#a \cap b = 1$.

Thus a regular vine can be constructed by starting with the n nodes $\{1, \ldots,\ n\}$. The first tree T_1 can then be made simply by connecting the n nodes together with the $n-1$ edges, $e_1^1, \ldots,\ e_{n-1}^1$, such that all nodes are connected without cycles.

The edges $e_1^1, \ldots,\ e_{n-1}^1$ are now used as nodes in the second tree T_2 and they are connected again with $n-2$ edges, $e_1^2, \ldots,\ e_{n-2}^2$ such that all nodes are connected. Item 4, in the definition, specifies that two nodes in T_2, e_{j_1} and e_{j_2} can only be connected if both connect to the same node in T_1.

Each edge in the regular vine will be representing a bivariate copula. Thus the fourth condition, the proximity condition, is necessary to make the edges translatable in to copulae.

The vine specification of the d-dimensional C needs $d-2$ trees to become a *regular vine specification*. Now the nodes for T_3 are the edges E_2, which means that the notation for the edges in E_3 will become quite large. Thus instead note that the copulae that correspond to the edges in E_3, could be

$$C_{i,i+3|i+1,i+2}(F(u_i|u_{i+1},\ u_{i+2}),\ F(u_{i+3}|u_{i+1},\ u_{i+2})),\ \text{for } i = 1, \ldots, d-3.$$

and in general the edges in E_3 will always correspond to copulae conditioned on 2 variables. Likewise the edges in E_i will correspond to copulae conditioned on $i-1$ variables. An example how every pairwise copulae could be defined in a d-dimensional vine copula C is

$$C_{i,i+j|i+1,\ldots,i+j-1}\left(F(u_i|u_{i+1}, \ldots,\ u_{i+j-1}),\ F(u_{i+j}|u_{i+1}, \ldots,\ u_{i+j-1})\right)$$

$$\text{for } i = 1, \ldots, d-j, \quad j = 1, \ldots, d-1.$$

where j corresponds to the tree T_j. That is the copulae between u_{i_1} and u_{i_2} is conditioned on all the u's in between them. The vine that corresponds to these copulae is known as a *drawable vine* or D-vine and it has that name because its pyramid like structure is easily drawn, see Figure 3.1.

The specific variables that a copula is conditioned on will vary depending on the shape of the vine and thus the next two definition describe how to an edge relates to a pairwise copula.

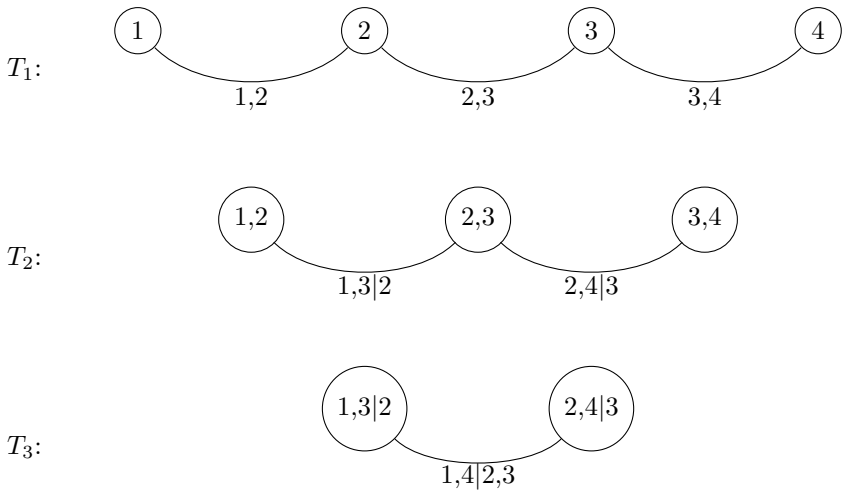


Figure 3.1: Graph of a drawable vine for a 4-copula. Note that the edges in the previous tree become the nodes of the next. Note that the dimensions the edges condition on are the set of dimensions that are shared between the nodes.

Definition 3.4 (Complete Union)
For a regular vine $\mathcal{V}$ and any $e_i \in E_i$ the complete union of e_i is the subset

$$A_{e_i} = \{j \in N_1 |\ \exists 1 \leq i_1 \leq i_2 \leq \cdots \leq i_r = i \text{ and } e_{i_k} \in E_{i_k} \text{ for } k = 1, \ldots, r,$$

$$\text{with } j \in e_{i_1}, e_{i_k} \in e_{i_{k+1}} \text{ for } k = 1, \ldots, r-1\}$$

For a regular vine and an edge $e_i \in E_i$ the j-fold union of e_i for $0 < j \leq i-1$ is the subset

$$U_{e_i}(j) = \{e_{i-j} \in E_{i-j} | \exists \text{ edges } e_k \in E_k \text{ for } k = i-j+1, ..., i-1,$$

$$\text{with } e_k \in e_{k+1} \text{ for } k = i-j, ..., i-1\}$$

For $j = 0$ define $U_{e_i}(0) = \{e_i\}$.

The complete union A_{e_i} is the set of nodes in N_1 that are indirectly connected to e_i. For example the edge $e = \{\{1,\ 2\},\ \{2,\ 3\}\}$ has $A_e = \{1,\ 2,\ 3\}$, because the two nodes it connects $\{1,\ 2\}$ and $\{2,\ 3\}$ themselves also connect $1,\ 2$ and 3. Thus if $e_i = \{e_{i_1}, e_{i_2}\}$, then A_{e_i} can be derived recursively as

$$A_{e_i} = A_{e_{i_1}} \cup A_{e_{i_2}}$$

And if e_i is an edge for the first tree T_1, then the complete union is simply the pair of nodes that e_i connect or

$$A_{e_i} = e_i, \quad \text{if } e_i \in E_1$$

Definition 3.5 (Constraint set)
For $e = \{j, k\} \in E_i, i = 1, \ldots, m-1$, the conditioning set associated with e is

$$D_e = A_j \cap A_k,$$

and the conditioned sets associated with e are

$$U_{e,j} = A_j \backslash D_e \text{ and } U_{e,k} = A_k \backslash D_e.$$

The constraint set for $\mathcal{V}$ is

$$\mathcal{CV} = \{(U_{e,j}, U_{e,k}, D_e) \mid i = 1, \ldots, m-1,\ e \in E_i,\ e = \{j, k\}\}$$

Thus the copula that corresponds to some edge $e = \{j, k\}$ is the copula

$$C_{U_{e,j}, U_{e,k}|D_e}\left(F\left(u_{U_{e,j}}|\ u_{D_e}\right),\ F\left(u_{U_{e,k}}|\ u_{D_e}\right)\right)$$

For example the copula $C_{1,4|2,3}$ could correspond to the edge $e = \{j,\ k\}$, where

$$j = \{\{1,\ 2\},\ \{2,\ 3\}\}, \quad k = \{\{2,\ 3\},\ \{3,\ 4\}\}$$

since the complete union for an edge is the set of nodes in N_1 that are connected directly or indirectly by that edge, then,

$$A_j = \{1,\ 2,\ 3\}, \quad A_k = \{2,\ 3,\ 4\}$$

$$D_e = \{2,\ 3\}, \quad U_{e,j} = \{1\}, \quad U_{e,k} = \{4\}$$

Note here that $u_{D_e} = u_{\{2,\ 3\}} = \{u_2,\ u_3\}$.

Given that a bivariate copula has to be between two variables, there is an implication that $\#U_{e,j} = 1$ and $\#U_{e,k} = 1$. This is ensured by the proximity condition from the definition of a regular vine.

We have used the concept regular vine specification to describe how C has been specified, however with conditioning sets a a formal definition can be given as

Definition 3.6 (Regular Vine Specification)
$(\mathbf{F}, \mathcal{V}, \mathcal{C})$ is a regular vine specification if:

1. $\mathbf{F} = (F_1, ..., F_n)$ is a vector of continuous invertible distribution functions.
2. $\mathcal{V}$ is a regular vine on n elements.
3. $\mathcal{C} = \{C_e|\ i = 1, \ldots, n-1;\ e \in E_i\}$ is a collection of bivariate copulae.

A joint distribution $F(x_1, \ldots, x_d)$ is said to realize a regular vine specification, $(\mathbf{F}, \mathcal{V}, \mathcal{C})$, if the marginal distributions of x_i is F_i and if for each $\{j,\ k\} = e \in E_i$ the copula of $x_{U_{e,j}}$ and $x_{U_{e,k}}$ given x_{D_e} is the copula $C_e \in \mathcal{C}$.

Theorem 3.7
Let $(\mathbf{F}, \mathcal{V}, \mathcal{C})$ be a d-dimensional regular vine specification. Let f_i be the density of $F_i \in \mathbf{F}$ and let $c_{U_{e,j}, U_{e,k}|D_e}$ be the copula density of C_e for $e = \{j,\ k\}$, then the joint density that realises the R-vine specification is

$$f_{1,\ldots,d}(x) = \prod_{i=1}^{d} f_i(x_i) \prod_{i=1}^{d-1} \prod_{e \in E_i} c_{U_{e,j}, U_{e,k}|D_e}\left(F_{U_{e,j}|D_e}(x_{U_{e,j}}|x_{D_e}),\ F_{U_{e,k}|D_e}(x_{U_{e,k}}|x_{D_e})\right)$$

where $F_{U_{e,j}|D_e}$ is the cdf of $x_{U_{e,j}}$ conditional on x_{D_e}.

The theorem is presented without proof though one is provided in [BC01]. The theorem provides a straight forward approach to transforming a vine into a joint density, take for example the 4-dimensional drawable vine from Figure 3.1. The theorem states that its joint distribution will be the product of

its bivariate copulae corresponding to every edge of every tree, as

$$\begin{aligned} f_{1,2,3,4}(x) = f_1(x_1) \cdot f_2(x_2) \cdot f_3(x_3) \cdot f_4(x_4) \\ \cdot \underbrace{c_{1,2}\left(F_1(x_1),\ F_2(x_2)\right) \cdot c_{2,3}\left(F_2(x_2),\ F_3(x_3)\right) \cdot c_{3,4}\left(F_3(x_3),\ F_4(x_4)\right)}_{T_1} \\ \cdot \underbrace{c_{1,3|2}\left(F_{1|2}(x_1|x_2),\ F_{3|2}(x_3|x_2)\right) \cdot c_{2,4|3}\left(F_{2|3}(x_2|x_3),\ F_{4|3}(x_4|x_3)\right)}_{T_2} \\ \cdot \underbrace{c_{1,4|2,3}\left(F_{1|2,3}(x_1|x_2,x_3),\ F_{4|2,3}(x_4|x_2,x_3)\right)}_{T_3} \end{aligned}$$

The way of obtaining the conditional cdfs like $F_{1|2}$ or $F_{1|2,3}$ and in general any cdf the form $F_{U_{e,j}|D_e}$, will be presented later on in the section on how to estimate vines.

3.1 Canonical Vine

The types of regular vines considered up until now have been what is known as drawable vines and as can be seen in Figure 3.1 they have some nice properties like their inverted pyramid shape. However, though they looks nice, the vines shape doesn't make it easy to implement as seen in [ACFB09].

Instead, the *canonical vine* or C-vine will be presented. In a canonical vine, each tree T_i focuses on a single node from N_i and every edge $e \in E_i$ is connected to that node, see Figure 3.2. This focus on a single node, for each tree, has the effect that for example, the first tree T_1 could focus on the dependency between x_1 and every other variable $x_2, \ldots, x_d$. The second tree T_2 could then effectively focus on the dependency between x_2 and every other variable $x_3, \ldots, x_d$, not including x_1, because T_1 already exhausted x_1's dependency.

This continues for T_3 and the dependency between x_3 and $x_4, \ldots, x_d$ up until the final tree T_d. This structure of focusing on a single variable in every tree makes practical implementation such as estimation and simulation much simpler as compared to the drawable vine [ACFB09].

Definition 3.8 (Canonical Vine)
$\mathcal{V}$ is a *canonical vine* on n elements if it is a regular vine and

$$\# \left(\bigcap_{j=1}^{d-i} e_j^i \right) = 1, \quad e_j^i \in E_i, \quad i = 1, \ldots, d.$$

That is to say that for a given tree T_i, its set of edges E_i have to all connect to the same node.

In the context of copulae, a d-copula can be constructed from a canonical vine, by choosing some permutation of $1, \ldots, d$. For simplicity let the permutation P be the identity $P = 1, \ldots, d$. The i'th element of P now implies which node that all edges in E_i should connect to.

The first set of edges E_1 give the first $d-1$ copulae as $C_{j,1}$ for $j = 2, \ldots, d$. The second set of edges E_2 give the next $d-2$ copulae as $C_{j,2|1}$, for $j = 3, \ldots, d$. E_3 gives the next $d-3$ copulae as $C_{j,3|1,2}$, for $j = 4, \ldots, d$ and so on.

In general, E_i gives the $d-i$ copulae as $C_{j,i|1,\ldots,i-1}$, for $j = i+1, \ldots, d$. If the permutation P is different from the identity, then the generalization uses $C_{P_j,P_i|P_1,\ldots,P_{i-1}}$ instead. In total this gives $\sum_{i=1}^{d} d - i = \frac{(d-1)d}{2}$ bivariate copulae.

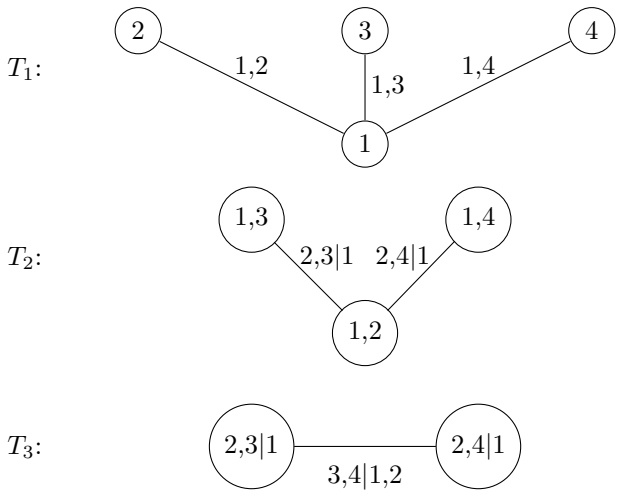


Figure 3.2: Graph of a canonical vine for a 4-copula. Note that the edges in the previous tree become the nodes of the next. Note that the dimensions the edges condition on are the set of dimensions that are shared between the nodes.

While vine copula are a great way of simplifying high dimensional copulae. It should be noted that the vine copula is limited in what multivariate copulae, that it can express. Consider for example the 3-dimensional copula density

$$c(u_1,\ u_2,\ u_3) = 2 \cdot \mathbb{1}\left\{ \mathbb{1}_{\{u_1 \leq 0.5\}} + \mathbb{1}_{\{u_2 \leq 0.5\}} + \mathbb{1}_{\{u_3 \leq 0.5\}} = \{1, 3\} \right\} \tag{3.1}$$

where $\mathbb{1}_{\{A\}}$ is an indicator function for whether A is true. Thus when

$$\begin{aligned} &u_1 \leq 0.5,\ u_2 \leq 0.5 \text{ then, } u_3 \leq 0.5, \qquad && u_1 \leq 0.5,\ u_2 > 0.5 \text{ then, } u_3 > 0.5 \\ &u_1 > 0.5,\ u_2 \leq 0.5 \text{ then, } u_3 > 0.5, \qquad && u_1 > 0.5,\ u_2 > 0.5 \text{ then, } u_3 \leq 0.5 \end{aligned}$$

However it is also the case that there isn't any pairwise dependence between the three variables,

$$\int_0^1 c(u_1, u_2, u_3) du_i = 1, \quad \text{for: } i = 1,\ 2,\ 3$$

that is, $c_{1,2}(u_1,\ u_2) = c_{1,3}(u_1,\ u_3) = c_{2,3}(u_2,\ u_3) = 1$. This also means that the conditional functions $F(u_i|u_j) = u_i$ and that the dependence in c can't be represented by any vine copula.

Take the canonical vine copula that uses the three bivariate copulae, $c_{1,2}$, $c_{1,3}$ and $c_{3,2|1}$. As mentioned $c_{1,2} = c_{1,3} = 1$ and therefore

$$c_{3,2|1}\left(F(u_3|u_1), F(u_2|u_1)\right) = c_{3,2|1}\left(u_3, u_2\right)$$

Because both $F(u_3|u_1)$ and $F(u_2|u_1)$ reduce to u_3 and u_2, then if c could be represented as a vine, $c_{3,2|1}$ should also reduce to $c_{3,2}$. However since $c_{3,2} = 1$, then $c_{3,2|1} = c_{3,2}$ would imply that $c = 1$, which is false. The discrepancy here comes from $c_{3,2|1}$ being dependent on the value of u_1, where a vine requires that $c_{3,2|1}$ is independent of u_1.

The reason vines don't work for some copulae like (3.1) is because vines can only model *simplified Pair Copula Constructions*, simplified PCC, [HAF10]. The simplified PCC requires that all conditional copulae as in those of the form $c_{a,b|C}$ are independent of the conditioning set C. Where as in the case of (3.1) the copula c requires that $c_{3,2|1}$ is dependent on the actual value of u_1. Common examples of simplified PCC's include Gaussian copulae, Student-t copulae and Clayton copulae [SJC13].

3.2 Estimation

This section will show how to estimate a canonical vine copula, C, with permutation $P = 1, \ldots, d$, from a set of vectors $\mathbf{u} = u_1, \ldots, u_d$ that each are uniformly distributed on $[0, 1]$. This is done by estimating its set of bivariate copulae $C_{i,j|1,\ldots,j-1}$, for every $j = 1, \ldots, d-1, \quad i = j+1, \ldots, d$. Where $C_{i,j|1,\ldots,j-1}$ explains the dependence between $F(u_i|u_1, \ldots, u_{j-1})$ and $F(u_j|u_1, \ldots, u_{j-1})$.

The conditional cdfs can be derived using the recursive relation

$$F(u_i|u_1, \ldots, u_j) = \frac{\partial C_{i,j|1,\ldots,j-1}(F(u_i|u_1, \ldots, u_{j-1}), F(u_j|u_1, \ldots, u_{j-1}))}{\partial F(u_j|u_1, \ldots, u_{j-1})}.$$

[ACFB09]. By letting $u_{i,j} = F(u_i|u_1, \ldots, u_j)$ the above equation simplifies to

$$u_{i,j} = \frac{\partial C_{i,j|1,\ldots,j-1}(u_{i,j-1}, u_{j,j-1})}{\partial u_{j,j-1}}, \tag{3.2}$$

where $u_{i,0} = u_i$ is the original unconditional uniform vector. The easiest copulae to estimate are $C_{i,1}$, since they specify the dependence between u_i and u_1 for $i = 2, \ldots, d$. Thus whatever bivariate copula estimation is desired like a kernel based estimator can then be applied on u_i and u_1 to obtain $\hat{C}_{i,1}$. With $\hat{C}_{i,1}$, the set of vectors $u_{i,1}$ for $i = 2, \ldots, d$ can be estimated as a transformation of u_i using the relation in (3.2)

$$\hat{u}_{i,1} = \frac{\partial \hat{C}_{i,1}(u_{i,0}, u_{1,0})}{\partial u_{1,0}}.$$

Thus we now have access to estimates of $u_{i,1}$ and we can use them to estimate $C_{i,2|1}$ with $\hat{u}_{i,1}$ and $\hat{u}_{2,1}$ for $i = 3, \ldots, d$, since $C_{i,2|1}$ specifies the relationship between $u_{i,1}$ and $u_{2,1}$. The estimated copulae can then again be used to estimate the transformations $u_{i,2}$ for $i = 3, \ldots, d$ as

$$\hat{u}_{i,2} = \frac{\partial \hat{C}_{i,2|1}(u_{i,1}, u_{2,1})}{\partial u_{2,1}}.$$

The general process for estimating a canonical vine copula switches between using the variables pairwise to estimate the bivariate copulae of tree T_j and then using those copulae to transform the variables with (3.2) into conditional versions, which can then be used to estimate the bivariate copulae in T_{j+1} for $j = 1, \ldots, d-1$. The entire process is written in Algorithm 1. The algorithm uses the notation

$$u_{i,j} = h_{i,j}(u_{i,j-1}) = \frac{\partial C_{i,j|1,\ldots,j-1}(u_{i,j-1}, u_{j,j-1})}{\partial u_{j,j-1}}$$

to denote the function that transforms $u_{i,j-1}$ into $u_{i,j}$.

3.3 Simulation

Let $\mathbf{u} = u_1, \ldots, u_d$ be a multivariate uniform distribution with some canonical vine copula C with permutation $P = 1, \ldots, d$. Then a sample from $\mathbf{u}$ can be simulated by starting with a simulation of iid. $w_1, \ldots, w_d \sim U[0, 1]$, and then deriving $u_1, \ldots, u_d$ as

$$\begin{aligned} u_1 &= w_1 \\ u_2 &= F^{-1}(w_2|u_1) \\ &\vdots \\ u_d &= F^{-1}(w_d|u_1, \ldots, u_{d-1}) \end{aligned}$$

Algorithm 1 Estimating a Canonical Vine Copula

```
u_1, ..., u_d
for i ← 1, ..., d do
    û_{i,0} ← u_i
end for
for j ← 1, ..., d − 1 do
    for i ← j + 1, ..., d do
        Ĉ_{i,j|1,...,j−1} ← Estimate Copula from û_{i,j−1} and û_{j,j−1}
        û_{i,j} ← ĥ_{i,j}(u_{i,j−1})
    end for
end for
```

[ACFB09]. Because C is a canonical vine copula, the various F's can be derived using (3.2), restated here as

$$u_{i,j} = \frac{\partial C_{i,j|1,\dots,j-1}(u_{i,j-1}, u_{j,j-1})}{\partial u_{j,j-1}}$$

Where $u_{i,j} = F(u_i|u_1, \dots, u_j)$. We can now use that in a canonical vine, $C_{i,j|1,\dots,j-1}$ is known, for every $j = 1, \dots, \ d-1, \quad i = j+1, \dots, \ d$. Additionally the fact that

$$w_j = F(u_j|u_1, \dots, u_{j-1}) = u_{j,j-1}$$

can be leveraged to simplify (3.2) to

$$u_{i,j} = \frac{\partial C_{i,j|1,\dots,j-1}(u_{i,j-1}, w_j)}{\partial w_j}.$$

By again using the notation $h_{i,j}(\cdot) = \frac{\partial C_{i,j|1,\dots,j-1}(\cdot, w_j)}{\partial w_j}$ we have that

$$u_{i,j-1} = h_{i,j}^{-1}\left(u_{i,j}\right).$$

This shows a recursive relationship starting at $w_i = u_{i,i-1}$ and going until $u_{i,0} = F(u_i) = u_i$. The recursion to transform w_i into u_i can be written as the single equation

$$u_i = \left(h_{i,1}^{-1} \circ \cdots \circ h_{i,i-1}^{-1}\right)(w_i)$$

Thus if the inverted h functions are available then u_i can be found using a simple recursive relationship. An implementation of this can be seen in Algorithm 2.

Algorithm 2 Simulating a Canonical Vine Copula

```
Sample w_1, ..., w_d
u_1 ← w_1
for i ← 2, ..., d do
    u_{i,i−1} ← w_i
    for j ← i − 1, ..., 1 do
        u_{i,j−1} ← h^{−1}_{i,j}(u_{i,j})
    end for
    u_i ← u_{i,0}
end for
```

3.3.1 Conditional Simulation

Let again $\mathbf{u} = u_1, \ldots, u_d$ follow some canonical vine copula C with permutation $P = 1, \ldots, d$. Additionally, assume that some of the variables in $\mathbf{u}$ already have known values. Then the distribution of the remaining variables will change accordingly and the method of simulation has to change.

In this section, the set of variables from $\mathbf{u}$ that are assumed to be known will be $u_1, u_2, \ldots, u_D$ for some index $0 < D < d$. The reason for assuming these variables in particular is that it is well suited for a canonical vine with permutation P.

Let the known values be $u_i = v_i$ for $i = 1, \ldots, D$. To simulate the remaining unknown values of $\mathbf{u}$, sample from a uniform distribution on $[0, 1]$ the values w_i for $i = D + 1, \ldots, d$. That is, sample one value w_i for each unknown value in $\mathbf{u}$.

Like for unconditional simulation, the goal is to obtain the unknown variables u_i using the relationship

$$u_i = F^{-1}(w_i | u_1, \ldots, u_{i-1}), \tag{3.3}$$

and like in unconditional simulation this relationship was obtained using the inverted $h_{i,j}$, where the non-inverted $h_{i,j}$ is given as

$$h_{i,j}(\cdot) = \frac{\partial C_{i,j|1,\ldots,j-1}(\cdot, w_j)}{\partial w_j}.$$

The added complication is that to use the h functions, $w_i = u_{i,i-1}$ are required for $i = 1, \ldots, d$. These are known in unconditional simulation. However, here the w_i are missing for $i \leq D$. Instead we have the known values v_i. Thus we have to convert the known values v_i into their w_i counterparts using the inverse version of (3.3)

$$w_i = F(u_i | u_1, \ldots, u_{i-1})$$

where $u_i = v_i$. This can again be obtained using a recursive application of $h_{i,j}$, that is

$$w_i = \left(h_{i,i-1} \circ \cdots \circ h_{i,1}\right)(v_i).$$

However, this also reveals that to obtain w_i for $i = 1, \ldots, D$, all w_j for $j = 1, \ldots, i - 1$ are needed, because again $h_{i,j}$ needs w_j. Thus starting at $i = 1$, set $w_1 = v_1$, then for $i = 2$, set $w_2 = h_{2,1}(v_2)$, then for $i = 3$, set $w_3 = h_{3,2}\left(h_{3,1}(v_3)\right)$ and so on, until and including $i = D$. After which the remaining $u_{D+1}, \ldots, u_d$ can be simulated like in the unconditional case. This whole process is implemented in Algorithm 3.

3.4 Vine Selection

In practice, the choice of vine impacts the fit and finding the optimal vine is a nontrivial problem and not least of which because it is uncertain what metric should be used to judge optimal. [DBCK13] considers maximizing the absolute Kendall's Tau $|\tau|$ for each edge as optimal. However, τ measures the ratio of concordant pairs, which isn't the only way dependence can manifest.

An alternative could be testing the uniformity of $\hat{u}_{i,i-1}$ for $i = 1, \ldots, d$ from Algorithm 1 [ACFB09], since in theory, they should be uniformly distributed. So the optimal vine should have the most uniform $\hat{u}_{i,i-1}$. However, testing uniformity is nontrivial when using the empirical distribution functions to transform the original data, as they essentially forces the uniformity of $\hat{u}_{i,0}$.

The amount of possible vines also becomes an issue as even when restricted to canonical vines, there are still $d!/2$ possible d-dimensional canonical vines [ACFB09]. Thus [DBCK13] approaches the problem greedily by choosing the vine's first tree T_1, as the one whose edges have the highest sum

Algorithm 3 Simulating a Canonical Vine Copula conditional on some values $u_j = v_j$

Given v_j for $j = 1, \ldots, D$, with $D < d$
$w_1 \leftarrow v_1$
for $i \leftarrow 1, \ldots, D$ **do**
  $u_{i,0} \leftarrow v_i$
  for $j \leftarrow 1, \ldots, i-1$ **do**
    $u_{i,j} \leftarrow h_{i,j}(u_{i,j-1})$
  end for
  $w_i \leftarrow u_{i,i-1}$
end for

Sample w_i for $i = D+1, \ldots, d$
for $i \leftarrow D+1, \ldots, d$ **do**
  $u_{i,i-1} \leftarrow w_i$
  for $j \leftarrow i-1, \ldots, 1$ **do**
    $u_{i,j-1} \leftarrow h_{i,j}^{-1}(u_{i,j})$
  end for
  $u_i \leftarrow u_{i,0}$
end for

of absolute Kendall's Tau $|\tau|$. The second tree T_2 is chosen in the same way and so on up to and including T_{d-1}.

It may also be the case that selecting the right vine is more important when considering parametric copulae, since they are inherently going to be more restrictive. That is, it is important to find the vine that can formulate the dependence in terms of the parametric copulae. Thus in the nonparametric version it may not be as important.

3.5 Gaussian Copula using a Vine

To finish on nonparametric vines and their estimation, their versatility is demonstrated by using it to estimate a 3-dimensional Gaussian copula with correlation matrix

$$\begin{bmatrix} 1 & -0.5 & 0 \\ -0.5 & 1 & 0.8 \\ 0 & 0.8 & 1 \end{bmatrix}.$$

The comparison of the Gaussian copula and its nonparametric vine estimation result can be seen Figure 3.3. The most noticeable flaw in the vine is in the plot between *Dim 2* and *Dim 3*, where the vine estimation has a few undesirable points in the bottom right corner.

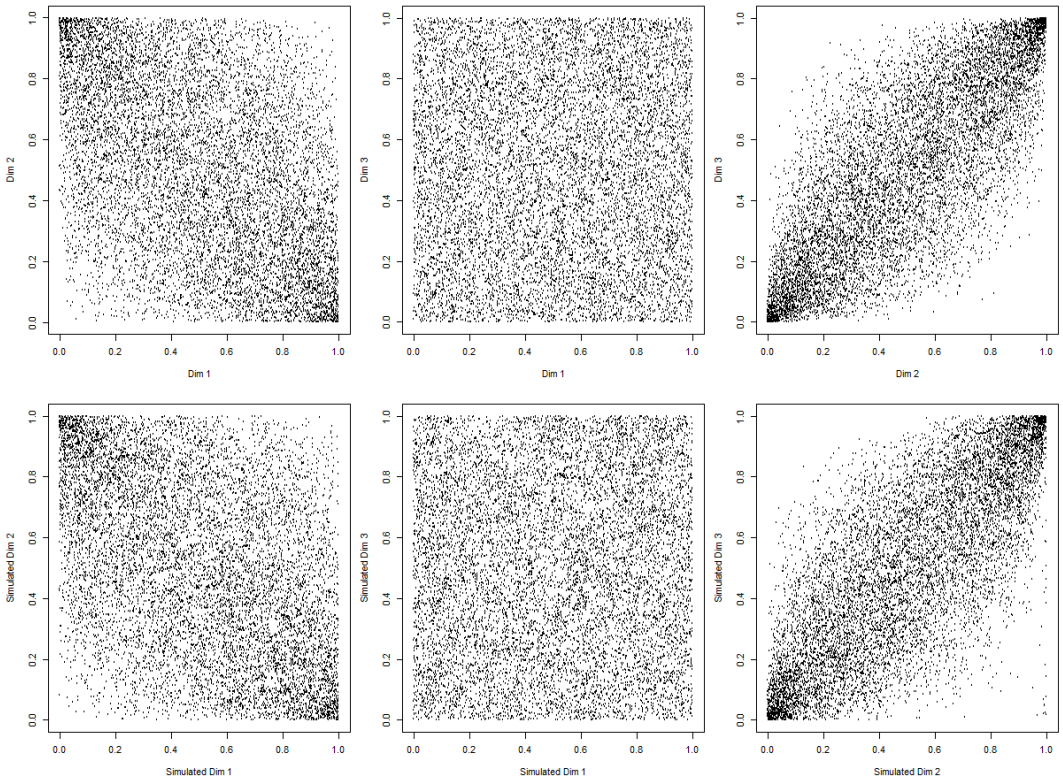


Figure 3.3: The top row of plots shows 10000 samples from a 3-dimensional Gaussian copula. The bottom row of plots shows 10000 samples from a nonparametric canonical vine estimation of the top row.

4 | Data Analysis

The goal here is to apply copula models to training data from financial assets, in order to construct an optimal portfolio. The optimized portfolio is then used on real test data to gauge its performance. The twelve assets considered are

SPY, NDAQ, AMD, AAPL, TSLA, NVO, NVDA, BA, LMT, GD, JPM, WFC

The assets represent a variety of different sectors and were chosen for personal reasons. The asset data was retrieved using the Polygon API at `polygon.io` and spans the entire year of 2022. The data is given in minutes and the modelling will be conducted on its log-returns. Data for all assets is not available at every minute, and the minutes where all assets aren't available are removed. The total amount of data for each month from January to December is, in order,

5744, 5456, 5770, 5492, 6301, 5761, 4799, 5208, 6368, 6495, 5742, 5164

The copula models will be used to find a portfolio of the twelve assets that minimize the *expected shortfall*, ES. The portfolio is assumed to be held for *ten minutes*, which means there are two separate ways of creating a model for this, either by creating a model for the ten minute returns and then finding a portfolio from that. Alternatively, creating a model for the one minute returns and then finding a portfolio for the sum of ten one minute returns.

The functions used can be found in the authors Github repositories `github.com/NickKruse18` where `P10` contains everything presented in this chapter and `kropula` is an R package containing only the capability for modelling with the presented nonparametric vine copulae without the application to portfolio allocation.

As is convention the modelling will be done on the log-returns and they are assumed to be strictly stationary.

4.1 Model Construction

4.1.1 Marginals

Let the one minute log-returns be denoted by ${}_kx_t$, where k refers to the asset and t the minute time. Additionally, let x_t be a 12 dimensional vector of all assets at time t. As mentioned, modelling will be done both on the one minute log-returns and on the ten minute log-returns. Thus define the ten minute log-returns as

$$ {}_k\underset{10}{x}_t = \sum_{\tau=t}^{t+9} {}_kx_\tau, \quad \forall t = 1,\ 11,\ 21, \dots $$

Notice that t is only defined every tenth integer.

The minute log-returns for different assets tend to vary in both mean, variance, skewness and kurtosis, as seen in both Table 4.1 and Table 4.2, with the majority of variation being in variance and kurtosis. Thus modelling them requires a distribution with a significant amount of freedom in its shape. The > 5 kurtosis for all assets also shows that they are all far from normal.

	SPY	NDAQ	AMD	AAPL	TSLA	NVO	NVDA	BA	LMT	GD	JPM	WFC
Mean	-0.04	-0.05	-0.22	-0.01	-0.12	0.09	-0.14	0.01	0.10	0.05	0.02	0.06
Std. Dev	19.13	8.27	16.31	8.70	18.01	7.49	17.46	11.15	6.57	6.57	7.65	8.85
Skewness	0.05	0.07	0.16	-0.02	0.13	-0.06	-0.04	-0.21	0.41	-0.07	0.39	0.23
Kurtosis	20.34	11.56	6.74	59.72	7.80	72.49	9.45	11.01	19.22	12.47	17.93	9.13

Table 4.1: Table of sample shape parameters for all minute log-returns ${}_k x_t$, over the period January to February 2022, 11200 data points. Mean and standard deviation are multiplied by 1000.

	SPY	NDAQ	AMD	AAPL	TSLA	NVO	NVDA	BA	LMT	GD	JPM	WFC
Mean	-0.47	-0.51	-2.22	-0.12	-1.32	0.88	-1.52	0.07	1.02	0.50	0.19	0.59
Std. Dev	30.15	28.98	60.20	29.28	66.10	22.70	62.82	39.44	22.92	25.15	26.09	29.96
Skewness	0.07	0.43	0.00	0.20	-0.10	0.59	-0.04	-0.29	0.14	0.43	0.27	0.17
Kurtosis	6.35	9.63	5.29	6.13	6.84	7.25	5.35	6.16	6.81	6.99	5.21	5.11

Table 4.2: Table of sample shape parameters for all ten minute log-returns ${}_k \underset{10}{x}_t$, over the period January to February 2022, 1120 data points. Mean and standard deviation are multiplied by 1000.

In this project the marginal modelling is done nonparametrically with a *sample cdf*. The sample cdf uses the sample, $X_k = \{{}_k x_t\}_{t=1}^n$, for asset k and is defined as the step function

$$\tilde{F}_k(x) = n^{-1} \sum_{i=1}^{n} \mathbb{1}_{\{{}_k x_{(i)} \leq x\}},$$

where ${}_k x_{(i)}$ is the i'th smallest element in X_k. The sample cdf for the ten minute returns, ${}_k \underset{10}{x}_t$, is denoted as $\underset{10}{F}_k(x)$.

By its definition, $\tilde{F}_k(x)$ is the proportion of elements in X_k that are smaller than x. One issue with $\tilde{F}_k$ as defined here is that the cdf will also be used for inverse transform sampling when making simulations. Since $\tilde{F}_k$ as defined is a step function, it will be the cdf of a discrete random variable and will only give values that are elements of X_k.

Instead, to create a continuous variable, the sample cdf will be defined with linear interpolation as

$$\hat{F}_k(x) = \frac{i_x + \left(x - {}_k x_{(i_x)}\right) / \left({}_k x_{(i_x+1)} - {}_k x_{(i_x)}\right)}{n},$$

where

$$i_x = \#\{y \in X_k | x \geq y\}$$

is the number of elements in X_k that are less than or equal to x. $\hat{F}_k$ is defined such that

$$\hat{F}_k(x) = \tilde{F}_k(x), \quad \forall x \in X_k,$$

but when $\hat{F}_k$ is between two elements, ${}_k x_{(i)}$ and ${}_k x_{(i+1)}$, then it gives a linear interpolation of $\tilde{F}_k\left({}_k x_{(i)}\right)$ and $\tilde{F}_k\left({}_k x_{(i+1)}\right)$, instead of just $\tilde{F}_k\left({}_k x_{(i)}\right)$. Figures 4.1 and 4.2 showcase two linear cdfs $\tilde{F}$ as compared to the samples they are derived from. They seem to match well with the samples difference in tail

heaviness showing up in their respective cdfs.

Another benefit of $\hat{F}_k$ being piecewise linear, is that its inverse $\hat{F}_k^{-1}$ will also be piecewise linear, which makes inverse transform sampling easier.

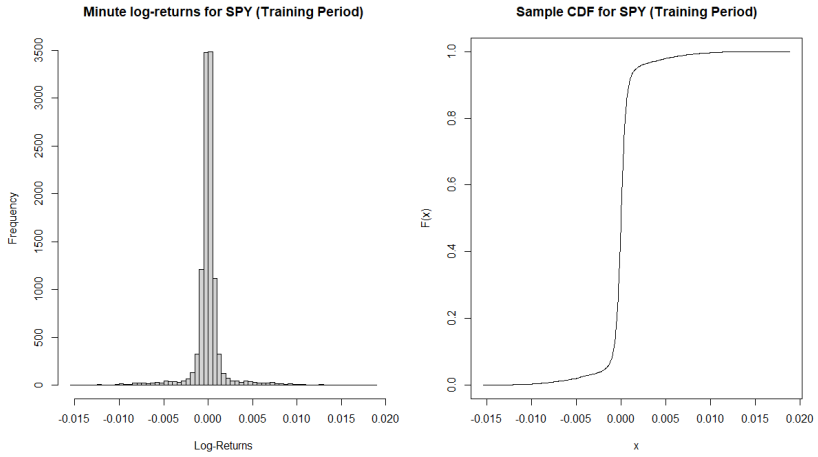


Figure 4.1: A linear sample cdf $\hat{F}_{SPY}$ for the minute log-returns ${}_{SPY}x_t$ from SPY in the period of January to February 2022, it uses 11200 points.

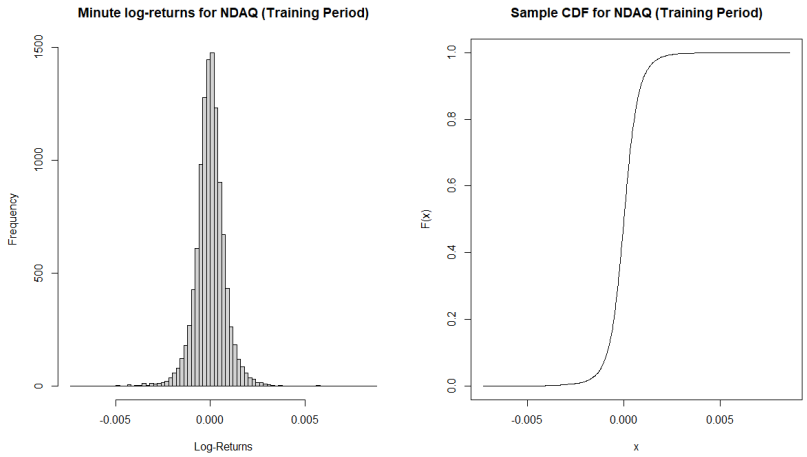


Figure 4.2: A linear sample cdf $\hat{F}_{NDAQ}$ for the minute log-returns ${}_{NDAQ}x_t$ from NDAQ in the period of January to February 2022, it uses 11200 points.

The marginals for the minute log-returns in the test period of March to December have the sample shape parameters given in Table 4.3. The modelling assumes stationarity which implies that the marginals for the test period should have the same shape parameters as the marginals in the training period. However, when comparing Table 4.3 to Table 4.1, there seems to be a general drop in standard deviation and an increase in kurtosis.

	SPY	NDAQ	AMD	AAPL	TSLA	NVO	NVDA	BA	LMT	GD	JPM	WFC
Mean	-0.01	0.01	-0.05	-0.01	-0.14	0.03	-0.03	0.01	-0.01	-0.02	-0.01	-0.05
Std. Dev	24.11	6.68	13.98	8.40	15.16	6.54	14.87	11.27	7.22	6.47	7.61	8.63
Skewness	0.02	-0.15	0.13	0.14	0.13	-0.81	0.16	0.02	0.03	0.01	-0.08	-0.17
Kurtosis	40.62	18.78	15.90	19.92	14.31	80.75	16.26	13.61	17.79	11.19	28.89	22.37

Table 4.3: Table of sample shape parameters for all minute log-returns ${}_k x_t$, over the period March to December 2022, 57100 data points. Mean and standard deviation are multiplied by 1000.

A bootstrapped test is applied to test if the training period and test period are generated by the same underlying distributions. The bootstrap samples 11200 minute log-returns from the test period with replacement and calculates a new set of shape parameters. This is done 1000 times and if a given shape parameter from Table 4.1 is between the 25'th smallest and the 25'th largest of the 1000 replications, that is to say the 5% level, it passes the test. The results of the test can be seen in Table 4.4. The reason a bootstrapped sample has 11200 log-returns is to match the size of the training data.

From Table 4.4 it seems that both the standard deviation and kurtosis only passes 50% of the time which is far below the expected rate of 95%. Both the mean and skewness pass for all twelve marginals, however, this simply reflects that the mean and skewness are naturally close to zero, which is expected in log-returns and doesn't prove that the training and test marginals are the same. Thus the test data fails to prove that it is generated by the same marginal distributions as the training data. Nonetheless, modelling will continue as this doesn't mean the modelling can't be successful.

	SPY	NDAQ	AMD	AAPL	TSLA	NVO	NVDA	BA	LMT	GD	JPM	WFC
Mean												
Std. Dev	-	+	+		+	+	+		-			
Skewness												
Kurtosis	-		-	+	-							-

Table 4.4: Table of whether the shape parameters from Table 4.1 fall within the bootstrapped confidence intervals for the test periods shape parameters at the 5% level. An empty entry means it falls within the confidence interval, a plus means it is too high and a minus means it is too low. It is based on 1000 replications.

4.1.2 Copulae

The sample cdfs, $\hat{F}_k$, defined in the previous section are used to transform the log-returns into uniformly distributed variables. Denote the uniform transforms as ${}_k u_t = \hat{F}_k({}_k x_t)$ and ${}_k \underset{10}{u}_t = \underset{10}{\hat{F}}_k({}_k \underset{10}{x}_t)$ for the one minute and ten minute log-returns respectively.

The copula models used to model the uniform transformations will be both a nonparametric vine copula and a Gaussian copula. The Gaussian copula will be used to benchmark the performance of the vine copula as it is a commonly used copula model in finance. The Gaussian copula is entirely defined by the lower triangle of its correlation matrix, which can be seen in Table 4.5 and Table 4.6 for ${}_k u_t$ and ${}_k \underset{10}{u}_t$, respectively.

	SPY	NDAQ	AMD	AAPL	TSLA	NVO	NVDA	BA	LMT	GD	JPM
NDAQ	0.50	·	·	·	·	·	·	·	·	·	·
AMD	0.61	0.48	·	·	·	·	·	·	·	·	·
AAPL	0.65	0.52	0.70	·	·	·	·	·	·	·	·
TSLA	0.59	0.46	0.73	0.67	·	·	·	·	·	·	·
NVO	0.37	0.35	0.38	0.41	0.36	·	·	·	·	·	·
NVDA	0.64	0.50	0.85	0.73	0.75	0.39	·	·	·	·	·
BA	0.56	0.43	0.56	0.55	0.54	0.31	0.57	·	·	·	·
LMT	0.31	0.27	0.16	0.24	0.17	0.19	0.18	0.38	·	·	·
GD	0.40	0.37	0.25	0.34	0.25	0.25	0.26	0.46	0.55	·	·
JPM	0.51	0.43	0.41	0.45	0.42	0.30	0.43	0.57	0.37	0.47	·
WFC	0.46	0.36	0.40	0.42	0.39	0.27	0.41	0.53	0.33	0.43	0.68

Table 4.5: The correlation matrix for the Gaussian copula with one minute log-returns.

	SPY	NDAQ	AMD	AAPL	TSLA	NVO	NVDA	BA	LMT	GD	JPM
NDAQ	0.58	·	·	·	·	·	·	·	·	·	·
AMD	0.61	0.53	·	·	·	·	·	·	·	·	·
AAPL	0.68	0.58	0.73	·	·	·	·	·	·	·	·
TSLA	0.60	0.51	0.78	0.69	·	·	·	·	·	·	·
NVO	0.42	0.44	0.44	0.46	0.44	·	·	·	·	·	·
NVDA	0.65	0.58	0.89	0.75	0.77	0.46	·	·	·	·	·
BA	0.59	0.53	0.61	0.62	0.63	0.38	0.62	·	·	·	·
LMT	0.34	0.30	0.16	0.27	0.17	0.25	0.15	0.39	·	·	·
GD	0.46	0.47	0.27	0.41	0.29	0.34	0.30	0.50	0.66	·	·
JPM	0.56	0.51	0.45	0.52	0.45	0.37	0.48	0.63	0.40	0.54	·
WFC	0.49	0.42	0.42	0.47	0.43	0.36	0.42	0.58	0.39	0.52	0.67

Table 4.6: The correlation matrix for the Gaussian copula with ten minute log-returns.

Both correlation matrices show a positive correlation between all assets with the weakest correlation in both matrices being between AMD and LMT at 0.16. The strongest correlation is also the same for both matrices and is between AMD and NVDA at 0.85 and 0.89, for one minute- and ten minute log-returns, respectively. Interestingly, the correlations for ${}_{k}\underset{10}{u}_{t}$ are higher almost across the board compared to ${}_{k}u_{t}$.

Both ${}_{k}u_{t}$ and ${}_{k}\underset{10}{u}_{t}$ are modelled using nonparametric vine copulae and both will use kernel estimates for their bivariate copulae. Since bivariate copulae are only supported on the unit square $[0,1] \times [0,1]$, kernel estimators need to be considerate of the squares boundaries.

The probit transformation presented in Section 2.8 and used by [NC16] and [GCP17] is a way of dealing with the boundary problem. This method is implemented in the `kdevine` package. However, its implementation takes 30 seconds to estimate a 12-dimensional vine from 1000 samples and an additional 13 minutes to sample 10000 samples from its estimate. Where as the implementation in `kropula` for the following kernel estimator takes around a second to do both. From limited testing both also seem to have similar performances in regards to expected shortfall, though a proper comparison is outside the scope of the thesis.

The reason boundaries are problematic is that while more central coordinates will be effected by points surrounding them on all sides, any coordinate on the boundary will have fewer surroundings to draw from. This has the noticeable effect that boundary areas can have an unrealistically lower estimated density. Thus instead of using a probit transformation, the proposed way of solving this problem is using *edge correction*, where the points on the boundaries are increased in weight to compensate for the reduced area to draw from. That is, the bivariate copula density c will be

$$\hat{c}(x,y) = \sum_{i=1}^{n} \frac{K\left(\max\left\{|x - x_i|, |y - y_i|\right\}\right)}{E_i n},$$

where E_i is the edge correction

$$E_i = \int_{[0,1]} \int_{[0,1]} K\left(\max\left\{|x - x_i|, |y - y_i|\right\}\right) dx \; dy,$$

and $\max\{|x - x_i|, |y - y_i|\}$ is the Chebyshev distance between $(x,\ y)$ and $(x_i,\ y_i)$. This distance metric is chosen because the area that $(x_i,\ y_i)$ will influence will resemble a square. This matches well with the copula being defined on the $[0, 1]^2$ square. As well, it makes computation simpler.

The kernel K is chosen as

$$K(d) = \max\left\{\frac{b-d}{4b^3/3}, 0\right\},$$

where d denotes the distance between $(x_i,\ y_i)$ and $(x,\ y)$. Thus the impact, the sample point $(x_i,\ y_i)$ has on the point $(x,\ y)$ will decrease linearly as $(x,\ y)$ moves further away. The use of the Chebyshev distance combined with the linearly decreasing kernel, means the influence of $(x_i,\ y_i)$ resembles a pyramid and thus the denominator in K is the volume of a pyramid with height b and side lengths $2b$. b is the *bandwidth* of K and here is chosen as 0.02 for the one minute log-returns and 0.03 for the ten minute log-returns. The influence of a single point under the chosen kernel can be seen in Figure 4.3.

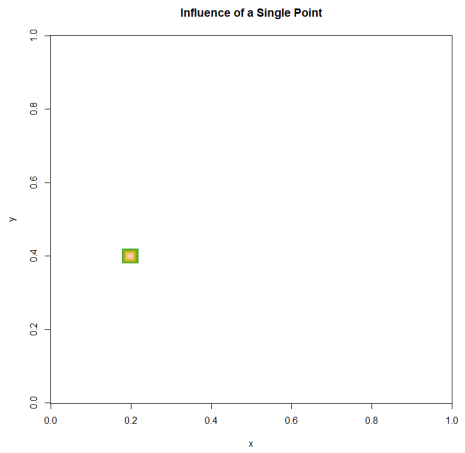


Figure 4.3: The area of influence of the data point $(0.2,\ 0.4)$ using the presented kernel with $b = 0.02$. Yellow represents a large influence and green represents a small influence. White means it has no influence.

Choosing the bandwidth as such can seem arbitrary and a more theoretically robust approach to choosing the bandwidth b is as the function of sample size n,

$$b = c \cdot n^{-1/5},$$

for some $c > 0$, see [FS03]. This approach accounts for the fact that as data increases the bandwidth must decrease at an appropriate rate to allow the estimator to convergence on the true copula. The decision to make the bandwidths 0.02 and 0.03 is a limitation of the implementation where these bandwidths are much easier to work with. Though they do scale approximately with $n^{-1/5}$, since there is 10 times as much data for the one minute log-returns and $0.03 \cdot 10^{-1/5} \approx 0.03 \cdot 0.631 \approx 0.02$.

Both models will use the same canonical vine. As mention earlier, the canonical vine is specified by the ordering of the variables and the chosen order for the assets will be the same as when first introduced, which is,

SPY, NDAQ, AMD, AAPL, TSLA, NVO, NVDA, BA, LMT, GD, JPM, WFC.

It will be demonstrated later that the ordering can be potentially significant, however, no consideration has been put into this chosen order, and no consideration will be put into finding an optimal ordering.

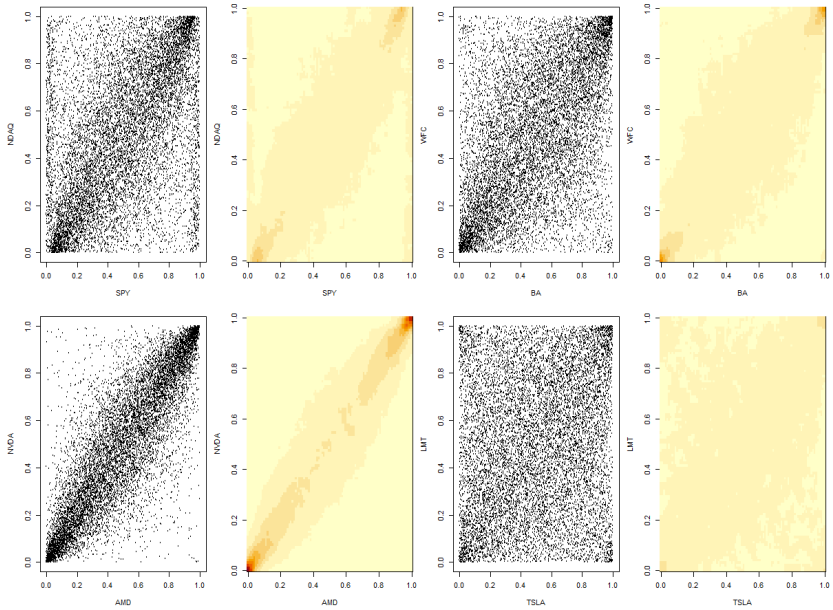


Figure 4.4: Each pair of plots shows: On the left, the uniform transforms of minute log-returns for two assets plotted against each other. On the right, the corresponding nonparametric copula density. The correlations for the estimated Gaussian copulae is from top-left to bottom-right: 0.50, 0.53, 0.85, 0.17. All four plots use 11200 points.

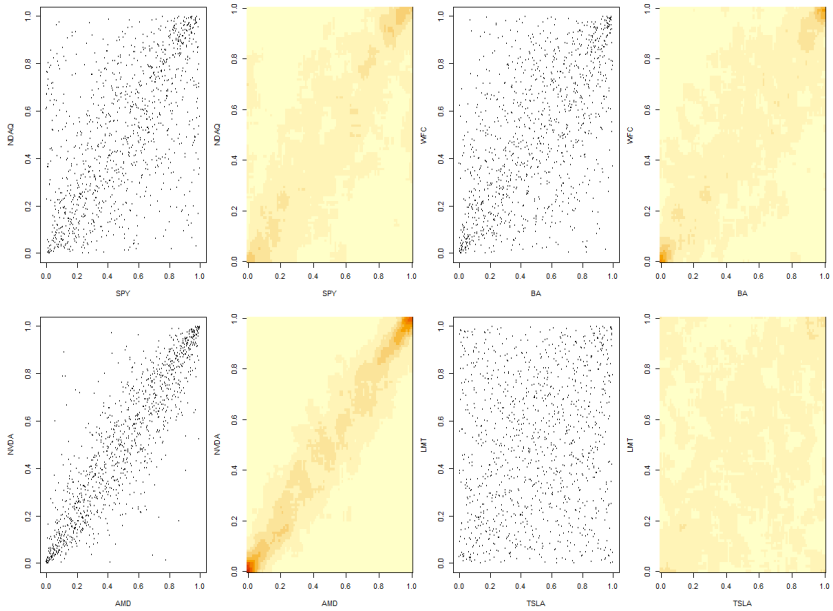


Figure 4.5: Each pair of plots shows: On the left, the uniform transforms of ten minute log-returns for two assets plotted against each other. On the right, the corresponding nonparametric copula density. The correlations for the estimated Gaussian copulae is from top-left to bottom-right: 0.58, 0.58, 0.89, 0.17. All four plots use 1120 points.

Figure 4.4 shows four comparisons between two one minute uniform transformations ${}_k u_t$ and ${}_{k'} u_t$ where $k \neq k'$, plotted against each other. Along with a corresponding nonparametric kernel estimate for its copula density. The copula densities match quite well. However, the dependence also doesn't seem to far from a Gaussian copula as it seems to be very linear, especially the dependence between AMD and NVDA. This suggests that a Gaussian copula could be sufficient.

The few differences there are to a Gaussian copula can be seen in the higher densities in all four corners of TSLA against LMT and the clusters along the right and left sides of SPY against NDAQ. Though it is not guaranteed that they will have a significant impact on the model.

Figure 4.5 shows the same four comparisons as in Figure 4.4, however, this time using $\underset{10}{u_t}$. The nonparametric copula densities again match quite well. However, the heavy reduction in data are really seen here with the densities appearing much rougher than with u_t. This is a clear advantage for the one minute models, especially in cases with much less data.

Copulae are conventionally considered when modelling the dependence between variables, however, they can also be used to model the auto-dependence of a single variable, like the dependence between $(u_{t-1},\ u_t)$. This could be relevant when modelling the one minute log-returns, ${}_k x_t$, since the model needs to sum up to ten one minute returns, which requires an accurate model of the joint distribution of

$$(x_t,\ x_{t+1}, \ldots,\ x_{t+9})\,.$$

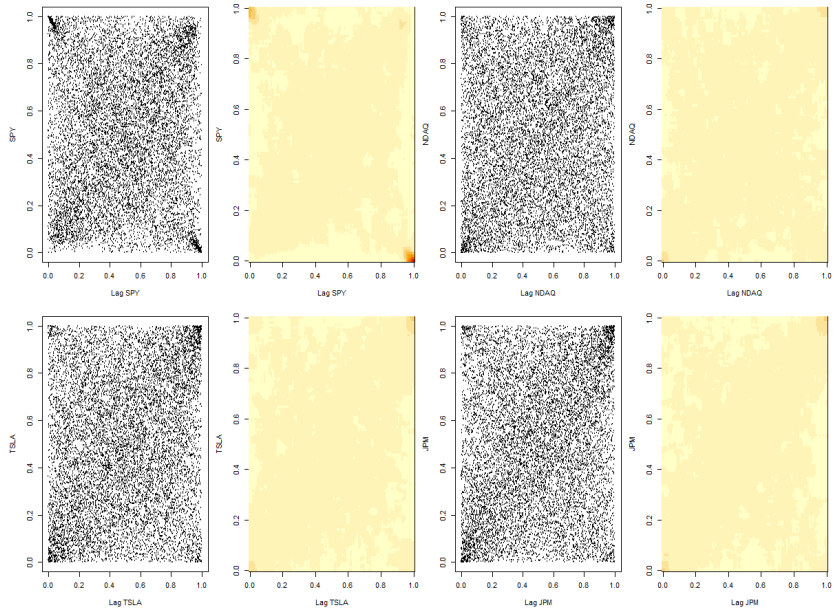


Figure 4.6: Each pair of plots shows: On the left, the uniform transforms of minute log-returns for a single asset plotted against itself lagged by a minute. On the right, the corresponding nonparametric copula density. The estimated Spearman's rho is from top-left to bottom-right: -0.01, 0.10, 0.17, 0.17. All four plots use 11199 points.

Looking at Figure 4.6 shows that there is some auto-dependence in the assets when plotting ${}_k u_{t-1}$ against ${}_k u_t$, which means it could be very necessary to include this in the model. Additionally their dependence seems to be non-Gaussian with the corners having the highest densities. Along with the

auto dependence in SPY being characteristically concentrated in two corners. The high corner densities is probably a result of asset volatility have a significant amount of auto correlation.

Thus a new nonparametric vine copula model is considered specifically for u_t that uses the same twelve assets in the same ordering. Additionally, they are included one more time lagged by one, u_{t-1}, for a total of 24 dimensions, with the lagged assets appearing first, to make prediction easier. The inclusion of u_{t-1} should hopefully be enough to account for the auto-dependence present in the log-returns.

Moving forward, the two Gaussian copula models will be denoted as GC and GC-10 for x_t and $\underset{10}{x}_t$, respectively. The nonparametric models will be denoted as NPC and NPC-10 for the two nonparametric models that don't account for auto-dependence. The one model that does will be denoted as NPC-AR. All vines for NPC, NPC-10 and NPC-AR are canonical with the identity permutation.

4.2 Simulation of the Vine Copula models

One very important consideration in regards to vine copulae is that most dependencies between two variables aren't directly modelled. Instead these dependencies are represented as the dependence between two conditional versions of the original two variables. For example, the dependence between the second variable, NDAQ, and the third variable, AMD, is given as

$$C_{NDAQ,AMD}(F(x_{NDAQ}), F(x_{AMD})),$$

but the canonical vine will capture it as

$$C_{AMD,NDAQ|SPY}(F(x_{AMD}|x_{SPY}), F(x_{NDAQ}|x_{SPY})).$$

Here both $F(x_{AMD}|x_{SPY})$ and $F(x_{NDAQ}|x_{SPY})$ have to be estimated which means some of the dependence could be lost. This could be compounded by the fact that some pairwise dependencies go through more than one transformation. In this specific case, the modelling is done with canonical vines, which means the pairwise dependency between the i'th and j'th variables is subject to $\min\{i, j\} - 1$ transformations before it is modelled. This all has the effect that the dependence seen in Figures 4.4, 4.5 and 4.6 could get lost in the transformation.

In fact this loss of dependence can be seen directly between the upper right plots of Figure 4.7 and Figure 4.8. The first figure shows the simulated version of SPY against NDAQ when using the NPC model and the higher density along the left and right sides are captured. However, The second figure shows simulated versions of SPY and NDAQ when using the NPC-AR model and the same feature isn't as prominent.

This is likely because SPY and NDAQ appear as the first and second variables in the NPC model, which means their dependence is modelled directly. For convenience the asset ordering is given again

SPY, NDAQ, AMD, AAPL, TSLA, NVO, NVDA, BA, LMT, GD, JPM, WFC.

In the NPC-AR model, however, the unlagged version of SPY and NDAQ appear as the 13'th and 14'th variables, which means their dependence is modelled indirectly after transforming both twelve times. So some of their features are lost. The simulated versions of AMD and NVDA also seem to not have as smooth corners in both Figure 4.7 and Figure 4.8.

The dependence of the NPC-10 model is shown between the same assets in Figure 4.9 and the simulations seem close to their real counterparts in all four cases. The only noticeable fault is that the

simulated versions of AMD and NVDA aren't clustered close enough together.

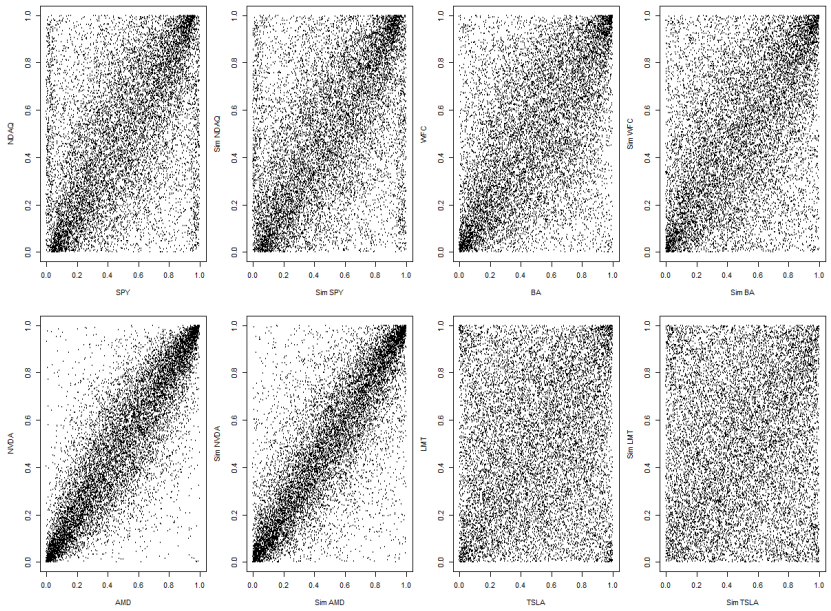


Figure 4.7: Each pair of plots shows: On the left, the uniform transforms of one minute log-returns for two assets plotted against each other. On the right, simulated versions from the NPC model. All four plots use 11200 points.

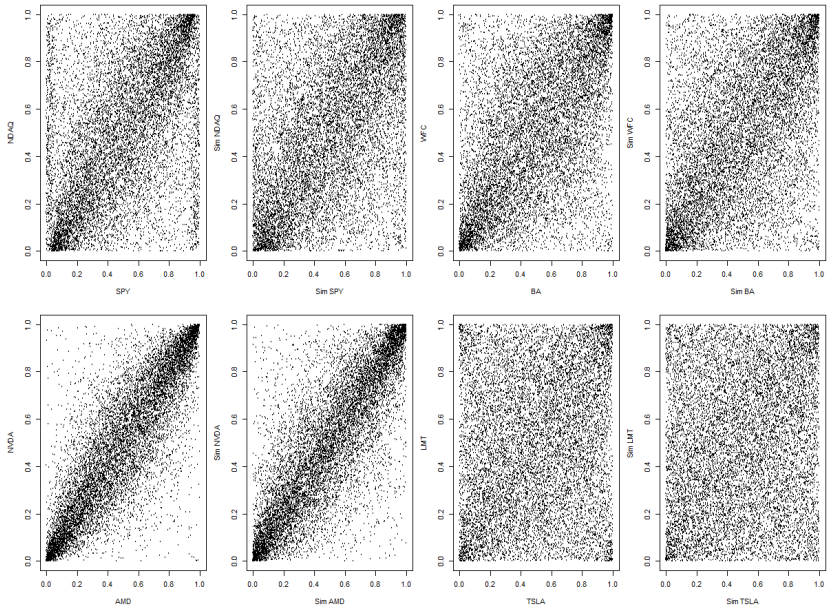


Figure 4.8: Each pair of plots shows: On the left, the uniform transforms of one minute log-returns for two assets plotted against each other. On the right, simulated versions from the NPC-AR model. All four plots use 11200 points.

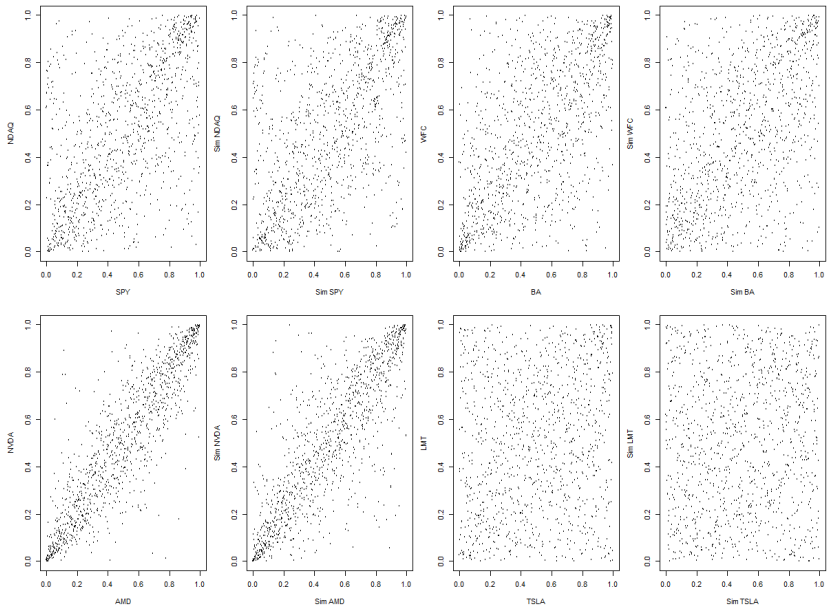


Figure 4.9: Each pair of plots shows: On the left, the uniform transforms of ten minute log-returns for two assets plotted against each other. On the right, simulated versions from the NPC-10 model. All four plots use 1120 points.

Another important ability for the NPC-AR model specifically is the ability to simulate the log-returns, ${}_kx_t$, based on its previous value ${}_kx_{t-1}$. To investigate how well it does this, the distribution of both ${}_{SPY}x_t$ and ${}_{TSLA}x_t$ given specific values of ${}_{SPY}x_{t-1}$ and ${}_{TSLA}x_{t-1}$, respectively, are plotted in Figures 4.10 and 4.11. The figures are both generated using the NPC-AR model.

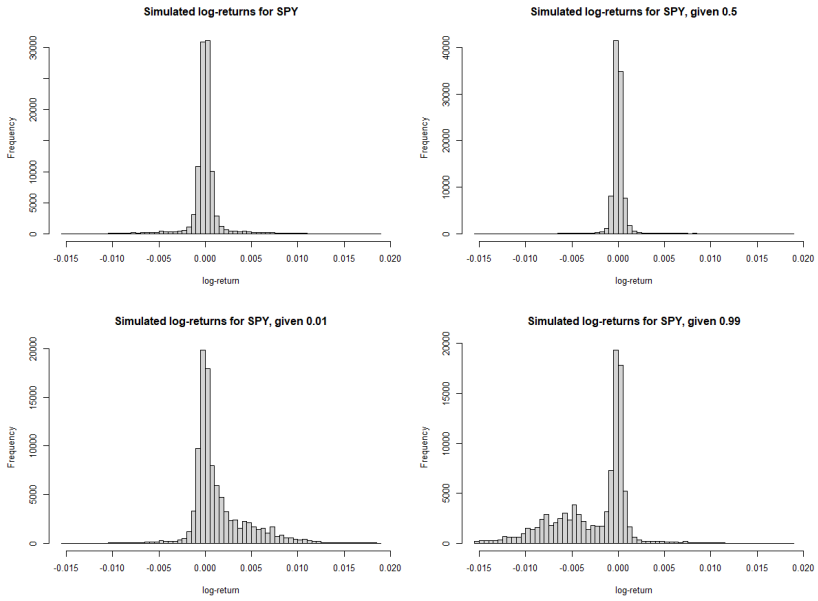


Figure 4.10: The conditional distribution of SPY given its previous minute value, as predicted by the NPC-AR model. The previous value of SPY is given in percentile. The top-left distribution is the unconditional distribution of SPY.

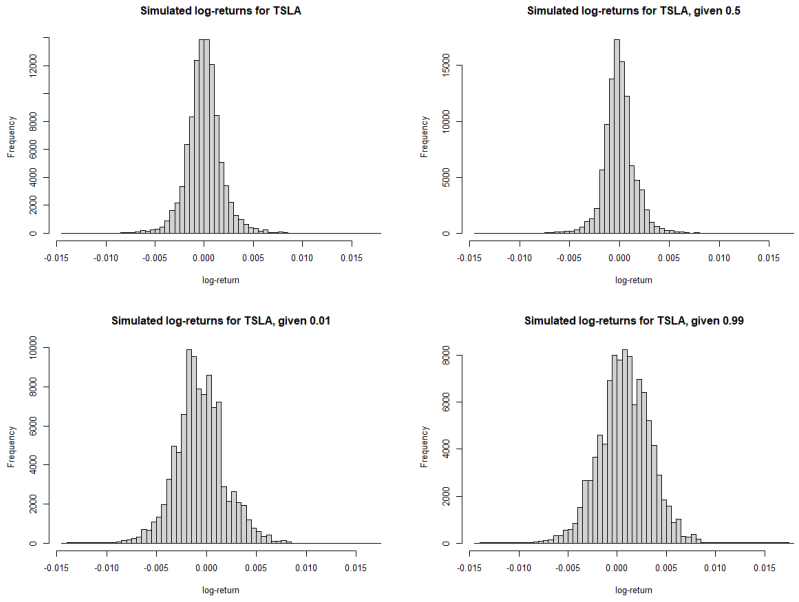


Figure 4.11: The conditional distribution of TSLA given its previous minute value, as predicted by the NPC-AR model. The previous value of TSLA is given in percentile. The top-left distribution is the unconditional distribution of TSLA.

The conditional distributions of SPY have a characteristic increase in positive values when conditioned on a very negative previous value (1'st percentile) and vice-versa, when conditioned on a very

positive value (99'th percentile). This is expected because of the two high density corners in SPY's auto dependence from Figure 4.6. The conditional distribution of TSLA shows that conditioned on a median previous value, the volatility reduces noticeably and when conditioned on either a very negative or very positive previous value, the volatility increases noticeably. This agrees with the higher density present in all four corners of TSLA's auto dependence.

4.3 Results

In this section, each of the five models will be compared to each other, GC, GC-10, NPC, NPC-10, NPC-AR. The models will be benchmarked using the 5% *expected shortfall*. That is, each model will construct a portfolio of the twelve assets that optimize expected shortfall. The optimization will assume that each portfolio is held for ten minutes. The increased duration is done to mimic holding a portfolio for an extended period, and practically a longer period than ten can be considered.

The data is from 2022 and it is split into training and test data, with January to February, two months or 11200 minutes, being used for training the models and March to December, ten months or 57100 minutes, being used to test the effectiveness of the portfolios. The minute log-returns from the training period will be denoted by ${}_k x_t$ for the k asset and the minute log-returns from the test period will be denoted by ${}_k y_t$. The ten minute log-returns will be denoted ${}_k \underset{10}{x}_t$ and ${}_k \underset{10}{y}_t$. The entire testing period from March to December will also be referred to as the *test year*.

Because there isn't a simple equation to calculate shortfall, it will be estimated using Monte Carlo simulations. That is all five models will be used to simulate multiple paths of ten minute log-returns, ${}_k \underset{10}{\tilde{x}}_t$, for all twelve assets. The portfolios are optimized for these paths.

The method for simulating $\underset{10}{\tilde{x}}_t$ differs between models. The GC-10 and NPC-10 models are the simplest as they are specifically made to simulate ten minute log-returns directly. The GC and NPC models are set up to simulate one minute returns, $\tilde{x}_t$, however the models assume by construction that $\tilde{x}_t$ are independent across time. Thus 10 one minute log-returns $\tilde{x}_t,\ \tilde{x}_{t+1}, \ldots,\ \tilde{x}_{t+9}$ are independently simulated and a simulation of a single ten minute log return is then calculated as

$$\underset{10}{\tilde{x}}_t = \sum_{\tau=t}^{t+9} \tilde{x}_t, \quad t \in \{1,\ 11,\ 21, \ldots\}.$$

The NPC-AR model is more complicated to simulate. Like the NPC model it is set up to simulate $\tilde{x}_t$, unlike the NPC model however, it doesn't assume temporal independence. Thus simulation of the 10 minute log-returns must be done jointly as

$$(\tilde{x}_t,\ \tilde{x}_{t+1}, \ldots,\ \tilde{x}_{t+9}), \quad t \in \{1,\ 11,\ 21, \ldots\}.$$

In practice, this is accomplished by first simulating $\tilde{x}_t$ unconditionally, then simulating $\tilde{x}_{t+1}|\tilde{x}_t$, then $\tilde{x}_{t+2}|\tilde{x}_{t+1}$ and so on until $\tilde{x}_{t+9}|\tilde{x}_{t+8}$. This should give the desired joint distribution of $(\tilde{x}_t,\ \tilde{x}_{t+1}, \ldots,\ \tilde{x}_{t+9})$, which can then be summed together.

The return of a portfolio after ten minutes, P_i, is the percentage increase in the asset prices ${}_k p_t$ between the buy date $t-1$ and the sell date $t+9$ multiplied by the portfolio weightings w_k,

$$P_i = \sum_{k=1}^{12} w_k \left(\frac{{}_k p_{t+9} - {}_k p_{t-1}}{{}_k p_{t-1}} \right).$$

However, given that the returns will be very small after ten minutes the approximation $e^x \approx 1 + x$ can be applied to use the log-asset returns ${}_k x_t$ directly as

$$P_i \approx \sum_{k=1}^{12} w_k \sum_{\tau=t-1}^{t+9} {}_k x_\tau = \sum_{k=1}^{12} w_k \cdot {}_k \underset{10}{x}_t. \tag{4.1}$$

The expected shortfall of the portfolio, $\widehat{ES}$. is estimated using n simulations of portfolio returns $\tilde{P}_i$,

$$\widehat{ES} = -\sum_{i=1}^{n} \chi\left(\tilde{P}_i < \widehat{VaR}_P(5\%)\right) \frac{\tilde{P}_i}{5\% \cdot n}, \tag{4.2}$$

where

$$\tilde{P}_i \approx \sum_{k=1}^{12} w_k \cdot {}_k \underset{10}{\tilde{x}}_i, \quad \widehat{VaR}_{\tilde{P}}(5\%) = \tilde{P}_{(5\% \cdot n)},$$

and $x_{i,k}$ is the i'th sample return from the k'th asset. Thus, $\tilde{P}_i$ is the payout from the i'th set of asset returns given the portfolio weights w_k for $k = 1, \ldots, 12$. Taken together, $\tilde{P}_i$ for $i = 1, \ldots, n$ is the entire sample distribution of portfolio returns.

The estimated 5% Value-at-Risk, $\widehat{VaR}_{\tilde{P}}(5\%)$, is the lowest 5'th percentile portfolio return, which is just the $(0.05 \cdot n)$'th smallest value from the entire sample distribution. The estimated expected shortfall, $\widehat{ES}$, is then the mean of all P_i conditional on them being lower than the 5% VaR.

The weightings w_k for the portfolio are chosen such that

$$w_k = \arg\min_{w_k} \widehat{ES}, \quad \text{subject to: } \sum_k w_k = 1.$$

The minimization is performed using the Nelder-Mead method from the `optim` function in R. The minimization is done with a simulated sample size of $n = 100000$ for each of the five models. Each model will be using their own simulations of ${}_k \underset{10}{\tilde{x}}_i$ when calculating expected shortfall.

The optimal portfolio weights for all five models are given in Table 4.7 and they are somewhat consistent across models, with all having particularly large positions on NVO and LMT. The large position in NVO and LMT along with the low positions on AMD and TSLA reflect that they are the assets with some of the lowest and some of the highest standard deviations, respectively, as seen in Table 4.2. It should also be noted that these weights are somewhat stochastic, since they are based on Monte Carlo estimates.

The OP model is a control portfolio that optimizes a portfolio to the *test* data. Thus its predicted shortfall represents the *lowest possible shortfall* that a fixed portfolio can have over the test period. It can thus be observed that all but one of the five models have lower predicted shortfalls. This isn't because their shortfalls are better than the optimal it is simply because they aren't good at predicting it. Additionally, the weightings for the OP portfolio also show that all five models are close to the optimal portfolio.

Model	SPY	NDAQ	AMD	AAPL	TSLA	NVO	NVDA	BA	LMT	GD	JPM	WFC	ES
OP	0.05	0.10	0.00	0.12	-0.04	0.37	-0.07	0.00	0.25	0.02	0.15	0.05	3.54
GC-10	0.01	0.08	-0.02	0.25	-0.03	0.41	-0.12	-0.04	0.25	0.03	0.19	0.00	3.25
NPC-10	0.08	0.14	-0.03	0.15	-0.01	0.31	-0.04	0.01	0.14	0.16	0.03	0.06	3.32
GC	-0.05	0.13	-0.02	0.15	-0.03	0.22	-0.05	0.00	0.31	0.09	0.14	0.11	3.02
NPC	0.00	0.12	-0.01	0.12	-0.01	0.18	-0.01	-0.04	0.26	0.20	0.08	0.11	2.94
NPC-AR	0.00	0.13	0.00	0.14	-0.03	0.29	-0.07	-0.01	0.25	0.13	0.12	0.03	3.66

Table 4.7: The optimal portfolios for minimizing expected shortfall as estimated using 100000 sample paths of the five copula models. ES is the 5% expected shortfall as predicted by each model. The shortfalls are multiplied by 1000.

There is significant difference between the portfolios predicted ES. GC-10 and NPC-10 give a predicted shortfall of around ≈ 3.3, where as, GC and NPC gives it at around ≈ 3 and NPC-AR predicting it to be around ≈ 3.7. This is likely not because GC and NPC have found better portfolios and instead because GC-10, NPC-10 and NPC-AR all predict that the distribution of portfolio returns will have fatter tails. This is especially apparent since the OP portfolio represents the best possible portfolio for the test year.

The higher predicted shortfalls for the NPC-AR model suggests that it predicts the ten minute log-returns will have fatter tails. The fatter tails are possibly because it predicts that extreme one minute log-returns increase the probability of future extreme log-returns as seen in Figures 4.6 and 4.11, which means some simulated ten minute log-returns, ${}_k\tilde{x}_{10}{}_t$, will also be more extreme. The fatter tails in GC-10 and NPC-10 are likewise also a result of auto-dependence within ${}_kx_t$ that is captured when modelling ${}_k x_{10}{}_t$ directly. It is unclear however why there still is a large discrepancy in the predicted shortfall of NPC-AR when compared to GC-10 and NPC-10, but it may be caused by a tenfold difference in sample sizes.

The five portfolios from Table 4.7 is used on the test data which are the ten minute returns from March 2022 to December 2022, ${}_k y_{10}{}_t$. It is assumed that the portfolio is purchased before the first minute at the price p_{t-1} and sold on the tenth minute at the price p_{t+9}. The approximation from (4.1) is used again to approximate the realized portfolio return directly with ${}_k y_{10}{}_t$ as

$$P_t \approx \sum_{k=1}^{12} w_k \cdot {}_k y_{10}{}_t.$$

The portfolio returns for each month is calculated separately. The portfolio returns for the entire test year are the returns from all ten months. The realized portfolio returns are used to estimate a 5% expected shortfall using (4.2).

Table 4.8 shows the sample expected shortfall at 5% for every model for every month and over the test year (March to December). The lowest shortfall for each month is highlighted in bold and the second lowest in italics. The NPC-10 model performed the best having the lowest shortfall on five out of ten months as well as the lowest shortfall over the entire test year.

the GC-10 and NPC-AR models seem to have performed comparably to each other, with GC-10 having better single month shortfalls and NPC-AR having a better test year shortfall being very close to the shortfall of NPC-10 and being close to the best possible shortfall from the OP portfolio. The two remaining models GC and NPC seem to share last place with both having identical year shortfalls.

While the NPC-10 model fares better in minimizing shortfall, however, it isn't much lower compared to the others, so maybe its better performance is pure chance. Perhaps more significant, is

in regards to the NPC-AR model. Because, the predicted shortfalls given in Table 4.7 shows that NPC-AR almost exactly predicted its realised shortfall from Table 4.8. All other models where very far off with NPC-10 being around 0.3 points off. This could suggest that the NPC-AR model was the only model to actually predict its shortfall.

Both GC and NPC had similar predicted shortfalls and performed comparably on the test data, which suggests there isn't a lot of difference between a Gaussian copula model and a nonparametric vine copula. This is compounded by the GC-10 and NPC-10 models also performing similarly both in prediction and realisation. Thus maybe a Gaussian copula is sufficient to explain the dependence between assets. Or at least that a nonparametric vine copula isn't much of an improvement.

However, one potential benefit of the nonparametric copula could come when regarding auto dependence as seen with the NPC-AR model having performed well on the test data and also potentially having been the only model to predict its shortfall.

Model	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Year
OP	3.61	3.53	3.95	3.68	2.88	3.98	3.69	3.29	4.00	2.99	3.54
GC-10	3.75	3.64	*4.10*	*3.77*	*2.94*	4.14	**3.67**	**3.29**	4.28	**3.02**	3.70
NPC-10	**3.59**	**3.49**	4.15	**3.58**	**2.85**	3.56	3.84	*3.44*	**4.10**	3.19	**3.66**
GC	3.91	3.75	4.22	4.08	3.31	*3.53*	4.03	3.72	4.26	3.44	3.87
NPC	3.90	3.70	4.32	3.99	3.30	**3.32**	4.04	3.84	4.25	3.55	3.87
NPC-AR	*3.72*	*3.57*	**4.06**	3.78	3.07	3.60	*3.79*	3.45	*4.14*	*3.18*	*3.67*
Data Points	577	549	630	576	479	520	636	649	574	516	5706

Table 4.8: The realized shortfalls for each month in 2022, excluding January and February, when using the six portfolios from Table 4.7. The lowest shortfall each month is in bold and the second lowest in italics (Not counting the OP portfolio). The shortfalls are multiplied by 1000.

4.3.1 Changing Vine

Three out of the five models uses vines, which means the choice of vine might have an effect on the final result. All three vine models use a canonical vine with an asset ordering of

$$\text{SPY, NDAQ, AMD, AAPL, TSLA, NVO, NVDA, BA, LMT, GD, JPM, WFC,} \tag{4.3}$$

where the NPC-AR model in particular features an extra copy of this ordering, but from the previous minute.

A full investigation into finding the optimal vine will not be conducted here, because at 12 assets there is simply to many possible vines. Instead to briefly explore how much a change of vines can impact the result, a new nonparametric vine copula model is setup again using a canonical vine, however, with the ordering from (4.3) being reversed. The model is derived from the NPC-AR model, because it uses the largest vine of all models, which would suggest that it is the most sensitive to changes in its vine. The version of NPC-AR with a different vine is denoted as R-NPC-AR.

The same procedure of constructing and testing a minimum expected shortfall portfolio, will be conducted using the same training and test data. The portfolio can be seen in Table 4.9 with comparison to the normal NPC-AR model. There seems to be a slight difference in portfolio weights and a bigger difference in its expected shortfall. How much difference exactly is obscured by the fact that the weights are based on Monte Carlo estimates and thus have some randomness to them.

Model	SPY	NDAQ	AMD	AAPL	TSLA	NVO	NVDA	BA	LMT	GD	JPM	WFC	ES
NPC-AR	0.00	0.13	0.00	0.14	-0.03	0.29	-0.07	-0.01	0.25	0.13	0.12	0.03	3.66
R-NPC-AR	-0.04	0.18	0.02	0.14	-0.02	0.24	-0.08	-0.07	0.29	0.11	0.16	0.07	3.80

Table 4.9: The optimal portfolios for minimizing expected shortfall as estimated using 100000 sample paths of the two copula models with different vines. The shortfalls are multiplied by 1000.

To test whether the differences in Table 4.9 are caused by Monte Carlo errors, the portfolio for the NPC-AR model is estimated 100 times using the same procedure as in Table 4.9. The sample mean and standard deviations for the portfolio weights are then calculated, seen in Table 4.10.

NPC-AR	SPY	NDAQ	AMD	AAPL	TSLA	NVO	NVDA	BA	LMT	GD	JPM	WFC	ES
Mean	0.003	0.156	-0.002	0.130	-0.033	0.275	-0.062	-0.011	0.282	0.106	0.109	0.048	3.696
Std. Dev	0.008	0.021	0.014	0.019	0.015	0.018	0.009	0.018	0.030	0.030	0.028	0.023	0.025
JB stat	1.662	1.358	5.641	0.322	2.467	1.371	0.046	1.502	1.725	1.267	0.844	2.790	3.190
p-value	0.436	0.507	0.060	0.851	0.291	0.504	0.977	0.472	0.422	0.531	0.656	0.248	0.203

Table 4.10: The sample mean and standard deviation for the optimal portfolio for NPC-AR, from Table 4.9, based on 100 replications. JB stat is a Jarque-Bera test on each weight and it indicates that all weights are Gaussian. The shortfall mean and standard deviation are multiplied by 1000.

The JB stats in Table 4.10 also shows that the weights are normally distributed, thus a chi squared test can be used to test if the portfolio generated by the R-NPC-AR model is identical to the one from the NPC-AR model. While there are 12 assets, the weights on each asset most sum to 1, which means 1 of the 12 weights is predetermined. Thus there is only 11 degrees of freedom and one asset can be removed before testing, which will be WFC. The result of the test can be seen in Table 4.11 and shows that R-NPC-AR is a different model to NPC-AR.

Chi Sq. test	Stat	df	p-value
	51.0	11	0.0000

Table 4.11: A chi squared test of whether the R-NPC-AR portfolio from Table 4.9 could be generated by the NPC-AR model based on the mean and standard deviations from Table 4.10.

Table 4.12 shows the comparative performances of R-NPC-AR and NPC-AR for every month. The R-NPC-AR uses the portfolio from Table 4.9 and it seems to have a slight increase in realized shortfall, which could suggest that the model is worse. However, the realized shortfall still manages to be close to its predicted shortfall. Thus it seems the vine used for modelling does impact its performance and consideration can be put into it finding the best one. This could especially be true since the R-NPC-AR model performs quite poorly when considering all models tested in Table 4.8.

Model	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Year
NPC-AR	3.72	3.57	4.06	3.78	3.07	3.60	3.79	3.45	4.14	3.18	3.67
R-NPC-AR	3.79	3.80	4.07	3.88	3.21	3.53	3.87	3.73	4.17	3.42	3.78
Data Points	577	549	630	576	479	520	636	649	574	516	5706

Table 4.12: The realized shortfalls for each month in 2022, excluding January and February, when using the two portfolios from Table 4.9. The shortfalls are multiplied by 1000.

Lastly the distribution from the realized portfolio returns for all six models, over the entire test year, are shown in Figure 4.12 and the realized means, Values-at-Risk and shortfalls are given in Table

4.13. It shows that the NPC-10 model and the NPC-AR model performed best, and the NPC model performed marginally worst in all aspects.

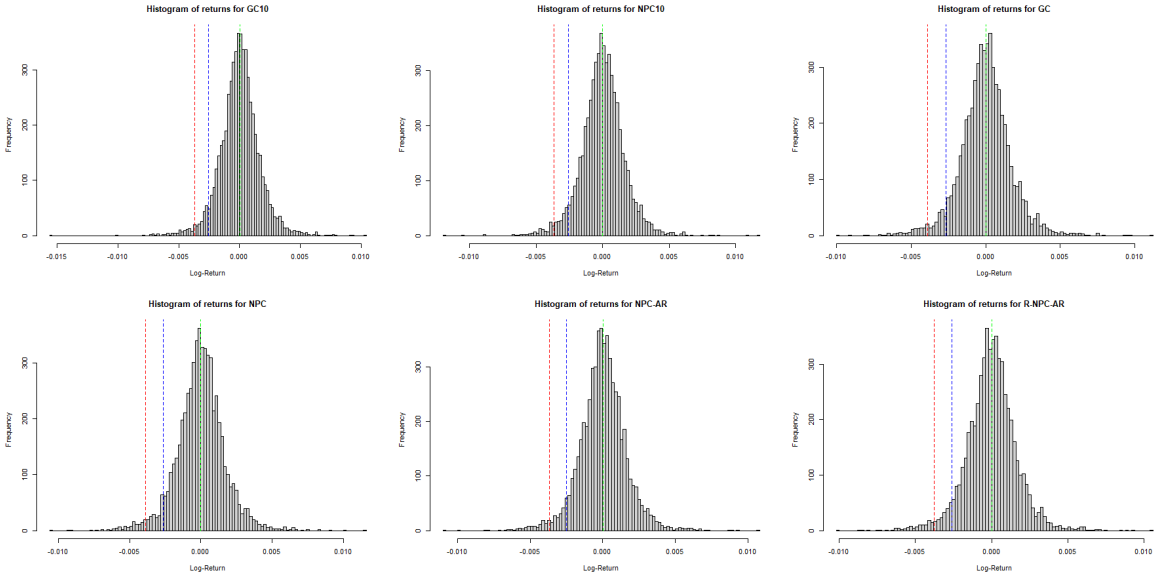


Figure 4.12: Histograms of the realized portfolio returns for the test year, when using the six portfolios from Tables 4.7 and 4.9. The green line is the mean return, the blue is the 5% VaR and the red is the 5% ES.

Model	GC-10	NPC-10	GC	NPC	NPC-AR	R-NPC-AR
Mean	0.02	0.01	0.00	-0.01	0.01	0.00
VaR	2.55	2.56	2.65	2.66	2.49	2.61
ES	3.70	3.66	3.87	3.87	3.67	3.78

Table 4.13: The realized mean, the 5% Value-at-Risk and the 5% expected shortfalls for the test year, when using the six portfolios from Tables 4.7 and 4.9. All values are multiplied by 1000.

4.4 Further on the NPC-AR Model

The previous section concluded with the NPC-AR model being considered one of the better models with a low expected shortfall and a close match between its predicted and realized shortfalls. Additionally the model uniquely has aspects to it that are open for further exploration. Thus this section will focus entirely on it, with further analysis and expansions.

4.4.1 Predicted vs. Realized Shortfall

The realized shortfall for the NPC-AR model in Table 4.8 was close to its predicted shortfall from Table 4.7. To test whether or not 'close' means that the NPC-AR model correctly predicted the expected shortfall, bootstrapping will be applied to the real portfolio returns. This will give an indication of how much variation there is in the realized shortfalls from Table 4.8

For each month in the test year and for the whole test year, the set of all log-returns will be pooled together and a new sample of returns will be drawn from the pool, with replacement. The size of the new sample will be as big as the original pool for the respective month or year. The estimated expected shortfall $\widehat{ES}$ is calculated for each of the new samples. This is done 1000 times.

To see if the predicted shortfall from Table 4.7 actually predicted the realized shortfall, it is tested whether the prediction falls between the 25'th smallest and 25'th largest of the 1000 bootstrapped shortfalls, or between the 2.5'th and 97.5'th percentiles. That is to say if the predicted shortfall passes at the 5% level.

It should be noted that bootstrapped samples disregards any autocorrelation there is likely to be present in the data, however this shouldn't be a problem as given that the portfolio weights are constant. The expected shortfall should only depend on the log-returns unconditional distribution.

Table 4.14 shows the two percentiles for every month and for the entire test year. The table shows that only the month of July failed the test with a particularly low expected shortfall. All other months including the whole year passed the test. A histogram of bootstrapped shortfalls for the test year are shown in Figure 4.13.

All together this suggests that the predicted shortfall is the same as the realized shortfall, because a single fail out of eleven isn't unexpected when working at the 5% level. It should be noted that the confidence intervals for every month are quite large, so the test isn't that powerful when it doesn't have a large amount of data. But nonetheless, it shows that the predicted shortfall is at least close enough to the realized shortfall to pass the test.

As discussed, the NPC-10 model generally performed better than the NPC-AR model in regards to minimizing realized shortfall. The issue with that model was that its predicted shortfall was far off its realized. Because of the large confidence interval when testing months, only the whole test year will be tested. The test for the NPC-10 model yields a confidence of (3.47, 3.83) which is far from its predicted shortfall of 3.32.

Shortfall	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Year
Data Points	577	549	630	576	479	520	636	649	574	516	5706
2.5%	3.16	3.08	3.61	3.28	2.59	2.86	3.25	2.88	3.34	2.59	3.49
97.5%	4.28	4.07	4.47	4.23	3.48	4.54	4.38	4.14	4.88	3.86	3.87
Pass	True	True	True	True	False	True	True	True	True	True	True

Table 4.14: The bootstrapped confidence intervals for the realized shortfall for each month. Pass is based on if the predicted shortfall of 3.67 falls within the interval. The shortfalls are multiplied by 1000.

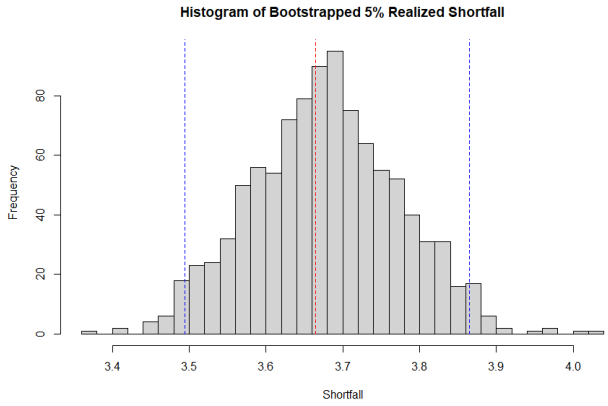


Figure 4.13: Histograms of the bootstrapped shortfalls over the test year. The blue lines are the confidence interval $(3.49,\ 3.87)$ and the red is the predicted shortfall 3.67.

4.4.2 Higher Order Lags

One of the defining features of the NPC-AR model is its auto dependence. This is accomplished by including lagged versions of each asset, ${}_k x_{t-1}$, together with the normal assets ${}_k x_t$ in its multivariate distributions. Given that the NPC-AR model has been successful, perhaps including more higher order lagged assets could improve performance. An i'th order lag here refer to including assets on the form ${}_k x_{t-i}$.

Additionally given that the modelling uses a vine, any number of combinations of lagged assets could be included based on how important they seem. So maybe ${}_{SPY} x_{t-2}$ has a uniquely high explanatory power compared to other 2'nd order lagged assets, ${}_k x_{t-2}$. Thus it alone can be included in the model.

The lagged assets can appear in any ordering that is desired. However, all lagged assets have to appear before the non-lagged assets, ${}_k x_t$, for the purposes of prediction. This is because when conditionally simulating new values, the canonical vine needs the previous values to appear earlier in the ordering.

Thus, there are many options when regarding adding higher order lags. To focus the scope, a model of order p includes all assets lagged at order p, x_{t-p}, and also all lags of lower order down to order 1. To end up with the joint distribution,

$$(x_{t-p}, \ldots,\ x_{t-1},\ x_t).$$

Three new models will be considered with $p = 2,\ 3,\ 4$, respectively, since the standard NPC-AR by definition is already of order 1. All three new models are tested using the same procedure as for the NPC-AR model. The summarised results of that test can be seen in Table 4.15. The realized ES is the shortfall over the entire test year. The order 1 model refers to the regular NPC-AR model. The shortfalls, both predicted and realized, for the higher order models seem to increase with order. The predicted and realized shortfalls also grow further away from each other.

Order	Predicted ES	Realized ES	Time Taken	Dimensions	Bi Copulae
1	3.67	3.66	30	24	276
2	3.70	3.76	59	36	630
3	4.00	3.84	113	48	1128
4	4.11	3.87	163	60	1770

Table 4.15: Predicted vs. Realized shortfall for higher orders of the NPC-AR model. Also the computation time, in seconds, to simulate 100000 ten minute sample paths, and the dimension of the copula and the amount of bivariate copulae used by the vine. The shortfalls are multiplied by 1000.

This all suggest that while adding auto dependence is beneficial, going above a lag of order 1 is detrimental to the performance. Table 4.15 also shows that the computation time for each model increases rapidly with higher orders. This is because of the increase in dimensions, which in the case of a vine copula means an increase in bivariate copulae. Specifically, for a dimension d the number of bivariate copulae in its vine is $\frac{d(d-1)}{2} \approx \frac{d^2}{2}$. The total number of bivariate copulae in each model can also be seen in the table and it seems to scale linearly with computation time.

A potential improvement to these results could be found by using the assumption of stationarity to reduce the high number of bivariate copulae. This can be done by recognising that e.g. the dependence between SPY and NDAQ at time t is identical to their dependence at time $t+1$ and the vine could be chosen to leverage that. An example of such a structure is the *Stationary Vine* found in [NKM22], which can reduce the number of unique bivariate copulae down to 210, 354, 498 and 642, respectively for orders 1, 2, 3 and 4. Though it should be noted that a reduction like this, while better for computation, doesn't necessarily translate to better predictions.

4.4.3 Greater Time Horizons

The model has been shown to perform well when holding a portfolio for ten minutes, however, it isn't clear that it works for periods longer than ten minutes. Realistically, it isn't possible to extend the time horizon up to a period where an investor would actually hold it, because the model needs to calculate every minute and the test period isn't large enough to give a meaningful comparison.

The time periods tested beyond 10 minutes, are 30, 60 and 120 minutes and the summarized results can be seen in Table 4.16. The tests are done using the NPC-AR model. These numbers are for the whole test year and as expected the shortfall increases with time period. The discrepancy between predicted and realized shortfall also increases, however that might be caused by the reduction in data points giving worse estimates for the realised shortfalls. The reduction in data points is because the test data is partitioned into periods that are 10, 30, 60 and 120 minutes long, respectively.

Time	Predicted ES	Realized ES	Time Taken	Data Points
10	3.66	3.67	30	5706
30	6.24	6.42	99	1900
60	8.55	9.18	203	948
120	11.62	12.86	383	472

Table 4.16: Predicted vs. Realized shortfall for greater time horizons of the NPC-AR model. Also the computation time, in seconds, to simulate 100000 sample paths, and the amount of tested data. The shortfalls are multiplied by 1000.

A unique portfolio was generated for each time period under the assumption that the optimal portfolio might change depending on how long it is held for. The portfolios generated can be seen in Table

4.17. There does seem to be trends in the weightings over increasing time. The position in AAPL, GD and JPM falls over time, while NVO, LMT and WFC increases.

Time	SPY	NDAQ	AMD	AAPL	TSLA	NVO	NVDA	BA	LMT	GD	JPM	WFC	ES
10	0.00	0.13	0.00	0.14	-0.03	0.29	-0.07	-0.01	0.25	0.13	0.12	0.03	3.66
30	0.01	0.18	0.02	0.14	-0.04	0.29	-0.07	-0.04	0.28	0.08	0.16	0.01	6.24
60	0.00	0.13	-0.01	0.13	-0.02	0.30	-0.07	-0.04	0.31	0.09	0.11	0.07	8.55
120	0.00	0.13	-0.02	0.11	-0.02	0.31	-0.07	-0.02	0.37	0.01	0.08	0.10	11.62

Table 4.17: The optimal portfolios for minimizing expected shortfall as estimated using 100000 sample paths of the four different time horizons using the same NPC-AR model. The shortfalls are multiplied by 1000.

These trends in weighting could however be purely coincidental given that there is some randomness at play when creating them. Thus Table 4.18 shows the chi squared test from Table 4.11 applied to each portfolio in Table 4.17. The results are that only the 120 minute portfolio is significantly different from the standard 10 minute one. This is perhaps not surprising given that the model only uses first order lags to give temporal variation, so the long term trends can't be much different from the short term trends.

Chi Sq. test	Stat	df	p-value
30	12.21	11	0.3483
60	9.55	11	0.5705
120	30.87	11	0.0012

Table 4.18: A chi squared test of whether the portfolios from Table 4.17 could be generated by the NPC-AR model based on the mean and standard deviations from Table 4.10.

Table 4.16 showed a significant difference between the predicted and realized shortfalls for longer time periods. To test if the difference is significant, the bootstrap test from Table 4.14 will be applied to the different time period, but only for the test year. The results of the test are given in Table 4.19 and all pass which suggests there isn't a significant difference and the little difference is a result of fewer data points.

Shortfall	10	30	60	120
Data Points	5706	1900	948	472
Predicted	3.67	6.24	8.55	11.62
2.5%	3.49	5.89	8.14	11.01
97.5%	3.87	6.96	10.18	14.57
Pass	True	True	True	True

Table 4.19: The bootstrapped confidence intervals for the realized shortfall for each time horizon. Pass is based on if the respective predicted shortfall falls within the interval. The shortfalls are multiplied by 1000.

4.5 Analysis of 2021

Up until now all data used was from 2022, so to verify that the results are consistent across time, the same models are fitted to data from the previous year, 2021. The data set consists of minute log-returns for the same twelve assets in the period of April to December 2021. Like with 2022, the

first two months April to May are used for training. The amount of data for each month starting from April is,

4742, 4291, 4711, 4600, 4124, 4591, 4568, 4146, 4108

The shape parameters of the marginal distributions can be seen in Table 4.20. The big separation from the marginals in 2022, see Table 4.1, is a sharp decrease in standard deviation across all assets.

	SPY	NDAQ	AMD	AAPL	TSLA	NVO	NVDA	BA	LMT	GD	JPM	WFC
Mean	-0.07	-0.07	-0.10	-0.06	-0.05	0.04	-0.02	-0.02	-0.01	-0.07	0.01	0.06
Std. Dev	14.98	5.39	9.03	5.72	12.06	3.80	8.74	7.22	4.34	4.61	5.56	7.35
Skewness	-0.21	0.02	0.04	0.25	0.34	-0.02	-0.14	0.14	0.07	-0.36	-0.09	0.34
Kurtosis	24.92	14.34	28.54	24.12	8.13	34.42	12.75	8.40	11.00	18.54	30.57	23.26

Table 4.20: Table of sample shape parameters for all minute log-returns ${}_k x_t$, over the period April to May 2021, 9034 data points. Mean and standard deviation are multiplied by 1000.

Figure 4.14 shows the assets plotted against each other, like seen in Figure 4.4 for 2022. However, unlike in 2022, the assets don't have as much co-dependence with the coefficients of correlation for a fitted Gaussian copula, being much lower than in 2022 and even slightly negative at times.

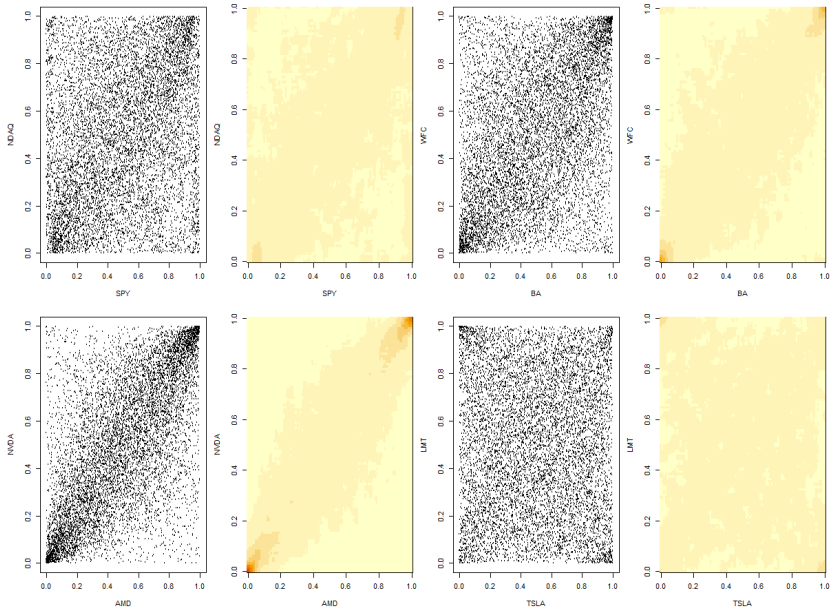


Figure 4.14: Each pair of plots shows: On the left, the uniform transforms of minute log-returns for two assets plotted against each other. On the right, the corresponding nonparametric copula density. The correlations for a Gaussian copulae is from top-left to bottom-right: 0.28, 0.42, 0.66, -0.02. All four plots use 9034 points.

The conclusion from fitting the model to 2022 was that the NPC-10 model and the NPC-AR model where the best. The NPC-10 model achieved the lowest realized shortfall, while the NPC-AR model accurately predicted its shortfall. Thus only those models will be considered for fitting and testing.

The portfolios for each model are generated using the same method as 2022 and can be seen in Table 4.21, including the optimal portfolio OP. Like in 2022, the NPC-10 model has a much lower

predicted shortfall compared to the NPC-AR model. It is relevant to determine whether the portfolios are actually different from those in 2022. Thus Table 4.22 shows the result of a chi squared test for the 2021 NPC-AR portfolio against the portfolio in Table 4.11. The test rejects the null hypothesis, so the 2021 portfolio for the NPC-AR is not the same as its 2022 counterpart.

Model	SPY	NDAQ	AMD	AAPL	TSLA	NVO	NVDA	BA	LMT	GD	JPM	WFC	ES
OP	0.05	0.02	0.04	0.15	0.04	0.31	-0.06	-0.01	0.24	0.06	0.16	0.00	2.39
NPC-10	0.01	0.12	0.04	0.11	-0.02	0.40	-0.01	-0.01	0.14	0.10	0.07	0.04	1.84
NPC-AR	0.00	0.11	0.05	0.11	-0.01	0.30	-0.01	-0.01	0.21	0.17	0.09	-0.01	2.21

Table 4.21: The optimal portfolios for minimizing expected shortfall as estimated using 100000 sample paths using the 2021 training data. The shortfalls are multiplied by 1000.

Chi Sq. test	Stat	df	p-value
	65.3	11	0.0000

Table 4.22: A chi squared test of whether the NPC-AR portfolio from Table 4.21 could be generated by a NPC-AR model using 2022 data, based on the mean and standard deviations from Table 4.10.

The portfolios are used on the remaining months, June to December seen in Table 4.23. It seems the same pattern emerges as in 2022. The NPC-10 model has a lower shortfall for five out of seven months and the NPC-AR model's predicted shortfall is much closer to its realized.

Model	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Year
OP	1.79	1.94	1.87	2.33	2.17	2.45	3.70	2.39
NPC-10	2.02	1.82	1.96	2.36	2.21	2.71	3.96	2.44
NPC-AR	2.07	1.91	2.01	2.38	2.33	2.64	3.55	2.43
Data Points	471	460	412	459	456	414	410	3082

Table 4.23: The realized shortfalls for each month in 2021, excluding April and May, when using the three portfolios from Table 4.21. The shortfalls are multiplied by 1000.

However, it seems the month of December has a much higher shortfall, in fact its comparable to the shortfalls in 2022. Thus it seems appropriate to remove the month from the test data. With December removed, the shortfall for the entire year, as seen in Table 4.23, is recalculated for the NPC-10 and the NPC-AR models as 2.19 and 2.23, respectively. Which like with 2022 puts the predicted shortfall for the NPC-AR model within 0.02 of the realized shortfall. Additionally, when using the bootstrapped test from Table 4.14, to test if the realized shortfall is close to the predicted shortfall. The NPC-AR model passes while the NPC-10 model fails, as seen in Table 4.24.

Thus the analysis of 2021 supports the results of 2022, that is the NPC-10 model produces marginally lower realized shortfalls, however, the NPC-AR model is better at predicting its realized shortfall.

Shortfall	NPC-10	NPC-AR
Predicted	1.84	2.21
2.5%	2.04	2.08
97.5%	2.34	2.39
Pass	False	True

Table 4.24: The bootstrapped confidence intervals for the NPC-10 and NPC-AR models using 2021 test data without December. Pass is based on if the respective predicted shortfall falls within the interval. The shortfalls are multiplied by 1000.

4.6 Synthetic Efficiency Test for Large Dimensions

The original paper on nonparametric vine copulae [NC16] conducted a few simulation tests to demonstrate the vines potential over the more conventional approach when modelling higher dimensions. However, the simulation didn't go beyond 10 dimensions, which doesn't reflect potential use cases in portfolio allocation, where they can be required to model hundreds of assets. Thus the test will be repeated for higher dimensions, but with out comparing it to the conventional nonparametric model. Additionally, only the Gaussian copula case will be considered as it provides an easily scalable model that can be represented as a vine [NC16].

This test simulates 10000 points from a multivariate Gaussian distribution as it is the simplest example of a Gaussian copula. The distributions will have zero mean and randomized covariance matrices to give the data more variety. A nonparametric vine copula is then fitted to the sample with a kernel bandwidth of 0.02. In [NC16], they compare the integrated absolute error of the joint densities of the real distribution f with its estimator $\hat{f}$,

$$\text{IAE} = \int_{\mathbb{R}^d} |f(x) - \hat{f}(x)| dx.$$

However, this metric becomes impractical in large dimensions. Instead since vine copulae are uniquely defined by the pairwise copulae between all its variables, a simpler approach would be to only compare the pairwise copulae of the real distribution and the estimator. Let c_{ij} be the bivariate copula between the i'th and the j'th variables and $\hat{c}_{ij}$ be its nonparametric vine estimator, then the performance of the estimator could be measured as

$$\frac{1}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \int_{[0,1]^2} |c_{ij}(u) - \hat{c}_{ij}(u)| du, \tag{4.4}$$

where n is the dimension in the multivariate distribution. Figure 4.15 shows (4.4) for Gaussian distributions at a variety of dimensions. The error is virtually constant, increasing slightly with number of dimensions. This result deviates from its counterpart in [NC16] where the error increased much more for higher dimensions. However, that may be a result of the difference in how error is measured. Nonetheless, Figure 4.15 suggests that the performance of the estimator doesn't decrease in higher dimensions.

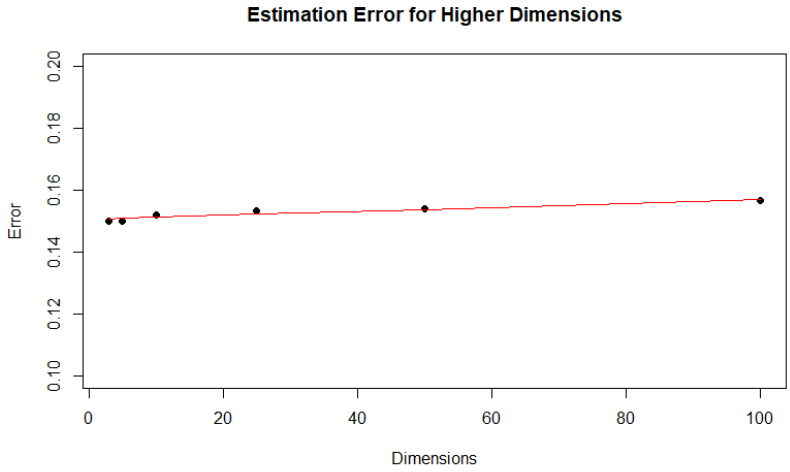


Figure 4.15: Equation (4.4) calculated for Gaussian distributions with dimensions 3, 5, 10, 25, 50 and 100. The shown error is the mean of 10 replications for each distribution.

5 | Discussion & Conclusion

The data analysis assumes strict stationarity however, it was shown that the marginals for the test period were different from the training period. This could call into question the conclusion that the NPC-AR model was able to predict the expected shortfall, since that should only be valid under strict stationarity. Thus the evidence that it predicted it in both 2021 and 2022, could be happenstance.

As mentioned, the data used for constructing the models often had missing entries. The approach to dealing with that was to simply ignore the missing data and stitch the data that was there together. However, this could have a negative impact on models like the NPC-AR model as it assumes that there is a one minute gap between all data.

In fact for 2022 only 9843 of the 11200 training data have a one minute gap between them or around 88%. This will impact performance and it likely means that the auto-dependence shown in Figure 4.6 actually includes a significant amount of noise.

Because of the way the NPC-AR model is specified it should be possible to condition on the previous minute, when constructing a portfolio. That is, instead of constructing a single portfolio to use over the entire test period, the portfolio weights could be made conditional to the last minute before each ten minute period. This may improve its performance, however likely only a slight improvement as the auto-dependence isn't incredible significant, and its significance will fall over the ten minute period.

As seen in Figure 4.11, the NPC-AR model used in data analysis seems to give the log-returns some heteroskedasticity. However, this effect is temporary and more conventional models like GARCH allow for the changing variance to persist for longer, which is also what is observed in practice.

Higher order AR components were also tried up to order 4, which concluded with order 1 being sufficient. Though this may just be because of too many variables or because it didn't reach far enough into the past. A potential solution to this could be that instead of modelling using the past four values, the modelling is done with the sum of the past four values. Periods of high variance would give a summed value further from zero, which would allow modelling in periods of changing variance.

It also doesn't need to be only the four past values and it may instead be the infinite past exponentially weighted in the sense of

$$\Sigma x_t = a x_{t-1} + a^2 x_{t-2} + a^3 x_{t-3} + \cdots ,$$

for some $0 < a < 1$. Σx_t can then be included as a variable in a regular vine specification in place of a past value like x_{t-1}.

Another potential extension would be to the vines. As vines are defined now, they only consider pairwise dependence. However, it is possible that in a d-dimensional variable, a group of three variables have a mutual dependency that need to be explained with a three-dimensional copula. For example,

given a four-dimensional variable $x = (x_1,\ x_2,\ x_3,\ x_4)$, it could be equipped with a copula such that its probability density would be given by

$$\begin{aligned} f(x) = f_1(x_1) \cdot f_2(x_2) \cdot f_3(x_3) \cdot f_4(x_4) \\ \cdot c_{123}\left(F_1(x_1), F_2(x_2), F_3(x_3)\right) \cdot c_{34}(F_3(x_3), F_4(x_4)) \\ \cdot c_{24|3}(F_{2|3}(x_2|x_3), F_{4|3}(x_4|x_3)) \cdot c_{14|23}(F_{1|23}(x_1|x_2, x_3), F_{4|23}(x_4|x_2, x_3)). \end{aligned}$$

The way this differs from the standard vine is the existence of the trivariate copula c_{123}, which under a standard vine would be expressed as $c_{12} \cdot c_{23} \cdot c_{13|2}$. This formulation would allow x_1, x_2 and x_3 more freedom in their mutual dependency.

This would need a new definition for a vine, since the edges that should represent trivariate copulae would need to connect three nodes, that is the edge e corresponding to a trivariate copula has to be expressed as $e = \{a,\ b,\ c\}$.

A goodness of fit test would also need to be considered, as what would justify using a trivariate copula over a set of bivariate ones. For parametric copulae that have a probability density, that could just be likelihood based. However, it would be challenging for kernel based copulae.

5.1 Conclusion

The goal of this project was to combine kernel based nonparametric copulae with vines. The nonparametric vine copulae were used to model the minute log-returns for a selection of twelve assets. The model was then used to construct a portfolio that minimized the 5% expected shortfall over a ten minute period. The modelling was done for the two years, 2021 and 2022.

The assets marginals where modelled using their sample marginals, to account for the large amount of variety in shape. Multiple copula models where tried on the assets uniform transforms, which includes Gaussian copulae and nonparametric vine copulae. The modelling was done on both the one minute and ten minute log-returns.

It was found that the dependency between assets were somewhat close to a Gaussian copula, as the models based on Gaussian copulae performed close to their nonparametric counterparts. However, a nonparametric vine copula for the ten minute log-returns, named NPC-10, was optimal at minimizing expected shortfall for both 2021 and 2022. However, its predicted shortfall underestimated the realized shortfall for both 2021 and 2022.

The nonparametric vine copulae showed their value when considering the auto-dependence within the same asset. The pairwise plots showed that the auto-dependence had a higher concentration in all four corners, which can't replicated by Gaussian copulae nor by other common parametric copulae. The high corner concentration shows GARCH-like behaviour, since it means extreme log-returns are more likely to be followed by more extreme log-returns. The nonparametric model that accounted for the auto-dependence within the one minute log-returns, named NPC-AR, only performed slightly worse than the NPC-10 model in terms of shortfall, however it managed to accurately predict its realized shortfall for both years.

Additional extensions to the NPC-AR model were explored. The standard model only considers the auto dependence for the log-returns and their values at the previous minute. The model was extended to also include the values for up to four minutes back, however it was found that adding more lagged values only decreased performance. A different extension was creating a portfolio that would be held for more than ten minutes into the future, going up to two hours. The NPC-AR model was found to

still be successful in this case, which implies that the NPC-AR model can be used to create a portfolio that can be held for an arbitrary length of time.

A synthetic test was conducted where the nonparametric vine copula was set to estimate multivariate Gaussian distributions. The Gaussian distributions had dimensions up to and including 100. The results where a slight but noticeable fall in performance that likely comes from the many variable transformations necessary in vines.

Bibliography

[ACFB09] Kjersti Aas, Claudia Czado, Arnoldo Frigessi, and Henrik Bakken. Pair-copula constructions of multiple dependence. *Insurance: Mathematics and Economics*, 44(2):182–198, 2009.

[BC01] Tim Bedford and Roger Cooke. Probability density decomposition for conditionally dependent random variables modeled by vines. *Ann. Math. Artif. Intell.*, 32:245–268, 08 2001.

[BC02] Tim Bedford and Roger M. Cooke. Vines–a new graphical model for dependent random variables. *The Annals of Statistics*, 30(4):1031 – 1068, 2002.

[DBCK13] J. Dißmann, E.C. Brechmann, C. Czado, and D. Kurowicka. Selecting and estimating regular vine copulae and application to financial returns. *Computational Statistics & Data Analysis*, 59:52–69, 2013.

[EMS02] Paul Embrechts, Alexander J. McNeil, and Daniel Straumann. *Correlation and Dependence in Risk Management: Properties and Pitfalls*, page 176–223. Cambridge University Press, 2002.

[FS03] Jean-David Fermanian and Olivier Scaillet. Nonparametric estimation of copulas for time series. *J. Risk*, 5, 03 2003.

[GCP17] Gery Geenens, Arthur Charpentier, and Davy Paindaveine. Probit transformation for nonparametric kernel estimation of the copula density. *Bernoulli*, 23(3):1848 – 1873, 2017.

[GD22] Gery Geenens and Richard Dunn. A nonparametric copula approach to conditional value-at-risk. *Econometrics and Statistics*, 21:19–37, 2022.

[Gee14] Gery Geenens. Probit transformation for kernel density estimation on the unit interval. *Journal of the American Statistical Association*, 109(505):346–358, 2014.

[HAF10] Ingrid Hobæk Haff, Kjersti Aas, and Arnoldo Frigessi. On the simplified pair-copula construction — simply useful or too simplistic? *Journal of Multivariate Analysis*, 101(5):1296–1310, 2010.

[Joe96] Harry Joe. Families of m-variate distributions with given margins and m(m-1)/2 bivariate dependence parameters. *Lecture Notes-Monograph Series*, 28:120–141, 1996.

[MS03] Y Malevergne and D Sornette. Testing the gaussian copula hypothesis for financial assets dependences. *Quantitative Finance*, 3(4):231–250, 2003.

[NC16] Thomas Nagler and Claudia Czado. Evading the curse of dimensionality in nonparametric density estimation with simplified vine copulas. *Journal of Multivariate Analysis*, 151:69–89, 2016.

[Nie06] Roger B. Nielsen. *An Introduction to Copulas*. Springer Series in Statistics. Springer, 2006.

[NKM22] Thomas Nagler, Daniel Krüger, and Aleksey Min. Stationary vine copula models for multivariate time series. *Journal of Econometrics*, 227(2):305–324, 2022.

[Sch82] B. (Berthold) Schweizer. *Probabilistic metric spaces / B. Schweizer and A. Sklar.* North Holland series in probability and applied mathematics. North Holland, New York, 1982.

[SJC13] Jakob Stöber, Harry Joe, and Claudia Czado. Simplified pair copula constructions—limitations and extensions. *Journal of Multivariate Analysis*, 119:101–118, 2013.

[Sto80] Charles J. Stone. Optimal Rates of Convergence for Nonparametric Estimators. *The Annals of Statistics*, 8(6):1348 – 1360, 1980.