
Erasmus University Rotterdam

Bachelor Thesis

*Bsc. International Econometrics and Operations Research*

# Dynamic C-vine Copula and its Application

Student Name: Yixun Zeng

Student Number: 446385

supervised by

**dr. Martina Zaharieva**

co-assessed by

**dr. Xiao xiao**

July 7, 2019

**ABSTRACT**

We tackled the challenge of modelling dependence amongst high-dimensional time series. We first studied a type of bivariate stochastic autoregresive copula and used pair-copula construction technique to build high-dimensional copula models which serve to capture time-varying dependence. We showed that the maximum likelihood estimation of this high-dimensional model can be decomposed into sequential estimations of bivariate models. We applied the dynamic copula models to study the dependence amongst four major Western Stock market indices and showed the advantages of the proposed model in-sample. We then used the estimated model to construct risk-managed portfolios to showed the usefulness of this model in portfolio management based on density forecasting. Finally, we proposed a method to obtain probability integral transformation(PIT) of high-dimensional copula models and showed that the high-dimensional time series can be studied block by block with the use of this technique. To illustrate the use thereof, we applied it to the major stock market indices of the Western and Eastern markets and studied the holistic dependence between the West and the East.

# Contents

# I. Introduction

Modelling multivariate distribution has been a crucial yet challenging task for risk management and portfolio construction. The difficulty lies mainly in characterising the dependence between variables, which often bears irregular patterns such as tail dependency. Previous attempts at multivariate dependency modelling, namely the multivariate GARCH(Bauwens, Laurent, and Rombouts (2006)) and stochastic volatility(Harvey and Shephard (1996)), presumed (conditional) multivariate normality. The resulting models are only able to capture elliptical(linear) dependence and as a consequence, often prove insufficient in the presence of richer dependence patterns. Copulas, on the other hand, offer a promising alternative. This class of models, after decades of development, possesses a great extent of variety and is thereby agile in modelling different situations of tail dependencies. Amongst the early literature on copula models, Patton (2009) and U. Cherubini (2004) provided a summary on their applications in finance. In general, there are two shortcomings of the early copula-based models. First of all, the fact that most researches revolved around bivariate copulas imposes great limitations on the use of this class of model for high-dimensional analysis. Secondly, as opposed to time-varying correlations amongst variables observed in reality, early copula models made unrealistic assumption on the constant multivariate dependence.

In recent research, both two drawbacks have been tackled. In respect of high-dimensional modelling with copulas, Okhrin, Okhrin, and Schmid (2013) and Savu and Trede (2010) developed the idea of hierarchical Archimedean copulas. Oh and Patton (2017) brought forward a class of factor-based copula models. Of other alternatives, modelling proposed in Bedford and Cooke (2002) and later developed in Aas, Czado, Frigessi, and Bakken (2009) has gained great popularity thanks to its flexible structure and ease in estimation. Their proposed models, also known as Vine copula, make use of a so-called "pair-copula construction" technique, allowing for constructing and estimating a large number of copulas sequentially. Following the work of Aas et al. (2009), Loran, Andreas, and Alfonso (2008) provided an extensive discussion on financial applications of the vine copulas. Studies on bivariate time-varying copulas include break-point-based approaches(Giacomini, Hardle, and Spokoiny (2009) and Dias (2002)), regime-switching models(Fink, Klimova, Czado, and Stober (2017) and Stober and Czado (2014)). Beside these, we also notice approaches that model the underlying process of parameter movement, to wit, Hafner and Manner (2012) and Almeida and Czado (2012) treated the copula dependence parameters as driven by some autoregressive latent process while Creal, Koopman, and Lucas (2013) by an observation-driven scaled score drives. An overview and applications regarding (bivariate) time-varying copulas can be found in Manner and Reznikova (2012).

The goal of our paper is to model the time-varying dependence of high-dimensional distribution. This has previously been discussed in various studies with the use of differing approaches. Heinen and Valdesogo Robles (2009) and So and Yeung (2014) suggest modelling the dependence using Dynamic Conditional Correlation(DCC) proposed in Engle (2002). In the realm of factor copula models, Creal and Tsay (2015) proposed a class of model with stochastic factor loadings; Oh and Patton (2018), on the other hand, model the time-dependence by means of the general autoregres-

sive score(GAS). Besides these, some extended the method of modelling the dependence parameter as some underlying process to the high-dimensional case. This includes the non-parametric approach proposed in Acar, Czado, and Lysy (2019) as well as Almeida, Czado, and Manner (2016), a follow-up study of the bivariate stochastic copula brought forward in Hafner and Manner (2012) as mentioned earlier. In our paper, we propose a high-dimensional dynamic vine-copula that utilises the same bivariate copula as in Almeida et al. (2016) but deviates from theirs in terms of the copula structure. We show that the model proposed can be estimated using sequential estimation. We also address how to obtain the probability integral transformation of the proposed model using an importance-sampling-based simulation and accordingly propose a block-based method of constructing vine-copula. Lastly, we make use of the proposed model to construct risk-managed portfolios and study the performance thereof. The remainder of this paper is structured as follows. II discusses the specification of canonical vine copula, stochastic autoregressive copula as well as how they can be combined to create the dynamic C-vine copula models using pair-copula construction. III discusses the estimation of the proposed model and IV presents the empirical applications. Finally, V outlines the findings and potential further research.

# II. C-vine-based Dynamic Copula Models

Consider a g-dimensional time series $\{\mathbf{X}_t\}_{t=1}^T = \{(X_{1t}, \dots, X_{gt})\}_{t=1}^T$. Each individual series is assumed to follow the Stochastic Volatility process proposed by Taylor (1986). This process[1], in its simplest form, reads as follow:

$$x_{it} = exp(h_{it}/2)\epsilon_{it} \tag{1}$$

$$h_{it} = \alpha_i + \beta_i h_{it-1} + \kappa_i \eta_{it} \tag{2}$$

where the innovations $\epsilon_{it} \overset{i.i.d}{\sim} N(0,1)$ and $\eta_{it} \overset{i.i.d}{\sim} N(0,1)$ are assumed to be mutually independent. With the supposition of the above marginal process, we introduce a copula-based model to characterise the joint (conditional) distribution of this high dimensional time series. Beginning with the overall design of dependence structure, II.A introduces the *Pair-copula Construction* technique and the C-vine copula model. Based on this, II.B extends this class model into a dynamic specification. Putting everything together, II.C discusses a C-vine-based alternative to model high-dimensional density through clustering the variables into several blocks.

## A. *Pair-Copula Construction of C-vine Copula*

Consider a g-dimensional random vector $X = (X_1, \dots, X_g)$ with the joint density $f(x_1, \dots, _g)$. Following the Theorem of Conditional Probability, the density can be decomposed as follows:

$$f(\mathbf{X}) = f(x_1, \dots, x_g) = f_1(x_1) f_{2|1}(x_2|x_1) f_{3|1,2}(x_3|x_1, x_2) \dots f_{d|1,\dots,d-1}(x_d|x_1, \dots, x_{d-1}). \tag{3}$$

[1]The stationary conditions are assume to hold (i.e $|\beta_i| < 1$).

According to Aas et al. (2009), every high-dimensional joint distribution can essentially be viewed as a composition of two components — the marginal behaviour of individual variables and the dependence structure for the involved variables — and the role of Copulas is to enable explicit modelling of the later component. It is a multivariate distribution(usually represented by $C(\cdot)$) with U(0,1) marginal distributions. According to Sklar's Theorem(Sklar (1957)), the relation between Copula distribution and the joint distribution reads as follows:

$$F(x_1, \ldots, x_g) = C(F_1(x_1), \ldots, F_g(x_g)) \tag{4}$$

where $F_i$ is the marginal distribution of variable $i$. Now denote $c_{1\ldots g}$ as the density function of Copula $C(\cdot)$(copula density hereafter). Applying the chain rule to this equation, we obtain that

$$f(x_1, \ldots, x_g) = c_{1\ldots g}(F_1(x_1), \ldots, F_g(x_g)) \times f_1(x_1) \ldots f_g(x_g). \tag{5}$$

## A.1. Pair-Copula Construction

A *pair-copula density* is defined as the copula density given by (5) in the bivariate case(Aas et al. (2009)). Under this setting, (5) can be simplified to

$$f(x_1, x_2) = c_{12}(F_1(x_1), F_2(x_2)) f(x_1) f(x_2) \tag{6}$$

where $c(\cdot, \cdot)$ denotes a bivariate copula — referred to as *pair-copula* throughout the paper. Following (6), the conditional probability density between these two variables can be written as:

$$f_{2|1}(x_2|x_1) = f(x_2) c_{12}(F_1(x_1), F_2(x_2)). \tag{7}$$

Now consider the three-variate term in (3), it can likewise be written using some appropriate pair-copula, for example[2]

$$\begin{aligned} f_{3|12}(x_3|x_1, x_2) &= f_{3|1}(x_3|x_1) c_{23|1}(F_{2|1}(x_2|x_1), F_{3|1}(x_3|x_1)) \\ &= f_3(x_3) c_{13}(F_1(x_1), F_3(x_3)) c_{23|1}(F_{2|1}(x_2|x_1), F_{3|1}(x_3|x_1)) \end{aligned} \tag{8}$$

Through the previous two examples, we notice that every term in (3) can be written as the product of some pair-copula and a conditional marginal density. The formal expression of this idea is derived in Aas et al. (2009). Denote the vector of conditioning variables as $\mathbf{v}$ and $\mathbf{v}_{-j} := \mathbf{v} \setminus \{v_j\}$, then for any given pivot variable $x$, the formula for pair-copula construction reads as follows:

$$f(x|\mathbf{v}) = f(x|\mathbf{v}_{-j}) c_{x v_j}(F(x|\mathbf{v}_{-j}), F(v_j|\mathbf{v}_{-j})), \quad v_j \in \mathbf{v} \tag{9}$$

Using this formula, we can perform iterative factorisation for (3) and break it down to the product of some conditional densities and the corresponding pair-copulas. Since each pair-copula in (9)

[2]or alternatively $f_{3|12}(x_3|x_1, x_2) = f_3(x_3) c_{23}(F_2(x_2), F_3(x_3)) c_{23|1}(F_{2|1}(x_2|x_1), F_{3|1}(x_3|x_1))$

takes the conditional distribution $F(x|\mathbf{v})$ as its input, Joe (1996) proposes that such a distribution can be obtained as follows:

$$F(x|\mathbf{v}) = \frac{\partial C_{xv_j|\mathbf{v}}(F(x|\mathbf{v}_{-j}), F(v_j|\mathbf{v}_{-j}))}{\partial F(v_j|\mathbf{v}_{-j})} \tag{10}$$

where $C_{xv_j|\mathbf{v}}$ is the CDF of the conditional bivariate copula. Given a univariate $\mathbf{v}$, the author shows that this formula can be simplified as:

$$F(x|v) = \frac{\partial C_{xv}(F(x), F(v))}{\partial F(v)} \tag{11}$$

Further, let us denote $u_i := F_i(x_i)$ and introduce the dependence parameter $\theta^{ij}$ to the previous pair copula ($C_{ij}(u_i, u_j) = C(u_i, u_j; \theta_{ij})$), we then define the following function(Aas et al. (2009)):

$$h(x_j|x_i, \theta^{ij}) = F(x_j|x_i) = \frac{\partial C_{ij}(u_i, u_j; \theta^{ij})}{\partial u_j} \tag{12}$$

It will be seen in the later section that this function is particularly important for the sequential maximum likelihood estimation regarding (3), as well as for sampling observations from C-vine.

## A.2. C-vine

In the case of high-dimensional distribution, the number of possible pair-copula constructions is undesirably large due to the flexibility in selecting the dependence structure. Bedford and Cooke (2002) propose a graphical model, the regular(R-) vine, to organise and visualise the choice of pair-copulas. However, this class of models is still very general and inherits the great flexibility in modelling dependence structure. For example, as discussed in Aas et al. (2009), a five-dimensional density can have as many as 240 different R-vine constructions.

In order to reduce the sheer amount of possibilities, Kurowicka and Cooke (2004) brought forward two specific structures in decomposing the density function, namely the Canonical(C-) Vine and D-vine model. For both models, the structure of dependence is determined by the order of variables. This results in great reduction of the number of potential decompositions(e.g. 60 decompositions in total for the five-dimensional case). In this paper, we will mainly concentrate on C-vine copula. This class of copulas implies the existence of some variable governing the dependence and interaction in the dataset(Aas et al. (2009)). We deem this phenomenon is particularly common in financial markets and as such, the C-vine setting is opted for our study. Mathematically, the C-vine-based density decomposition, given a certain variable order, can be formulated as follows(Aas et al. (2009)):

$$f(x_1, \ldots, x_g) = \prod_{k=1}^{g} f_k(x_k) \prod_{j=1}^{g-1} \prod_{i=1}^{g-j} c_{j,j+i|1,\ldots,j-1} \tag{13}$$

where $c_{j,j+i|1,\ldots,j-1} := c_{j,j+1|1,\ldots,j-1}(F(x_j|x_1, \ldots, x_{j-1}), F(x_{j+i}|x_1, \ldots, x_{j-1}))$. The underlying assumption for the C-vine model is that there is no further dependence between variable $x_i$ and $x_{i+1}$

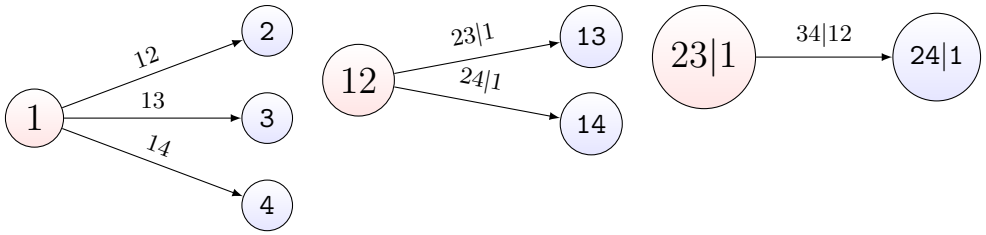



***Figure 1. Tree representation of a C-vine with 4 variables, 3 trees and 6 edges.*** *Each edge is associated with a pair copula, connecting two nodes that are taken as its input. Each component forms a 'tree'.*

except through the pair-copula $c_{i,i+1|1,\ldots,j-1}$ (see Aas et al. (2009)). Furthermore, for the simplicity of notation, denote $u_{j+i|1,\ldots,j-1} := F(x_{j+i}|x_1,\ldots,x_{j-1})$ such that $u_{j+i|1,\ldots,j-1} \sim U(0,1)$. And since $u_{\cdot|\cdot}$ is the input for copulas, it is designated as *copula data* throughout the rest of our paper. As an example, suppose we have a 4-dimensional vectors $\mathbf{X} = (X_1,\ldots,X_4)$, with the joint density $f(x_1,x_2,x_3,x_4)$. With the use of C-vine, we assume the dependence structure presented in Figure 1. Each edge in the graph represents a pair-copula, connecting two nodes taken as its input. Each component represents a *tree*. Also, for the sake of clearer notation, denote $E$ as the set of all edges(or equivalently, pair-copulas) involved in a C-vine copula. This designation will be used in the remaining of this paper. Following the theorem of conditional probability and the dependence structure given above, the joint density of the involved variables can be decomposed as such:

$$\begin{aligned} f(x_1,x_2,x_3,x_4) = {} & f_1(x_1)\cdot f_2(x_2)\cdot f_3(x_3)\cdot f_4(x_4) \\ & \times c_{12}\cdot c_{13}\cdot c_{14}\cdot c_{23|1}\cdot c_{23|1}\cdot c_{24|1}\cdot c_{24|1}\cdot c_{34|12} \end{aligned} \tag{14}$$

where $c_e(\cdot), e \in E$ is a binary copula, $E$ the edges in Figure 1.

## *B. Dynamic C-vine Copulas*

The last section has shown that the C-vine copula can best be understood as a decomposition of high-dimensional density into the products of some pair-copulas. In light of this, one can establish dynamic C-vine copula through incorporating dynamic pair-copulas in the pair-copula construction. Suppose we would like to model the time-varying dependence of time series $\{\mathbf{X}_t\}_{t=1}^T$ defined at the beginning of this section. Based on the pre-defined marginal process given by (1)(2), we first transform it into the copula data:

$$\{\mathbf{u}_t\}_{i=1}^T := \{(u_{1t},\ldots,u_{gt})\}_{t=1}^T = \{(F_1(x_{1t}),\ldots,F_g(x_{gt})\}_{t=1}^T$$

In the remainder of this section, we table a class of pair-copulas with time-varying dependence parameter and then integrate the dynamic pair-copulas into the C-vine model setting. The two actions are discussed in the following two sections (II.B.1 and II.B.2) respectively.

## B.1. A Pair-Copula Setting with Dynamic Latent Process

Now consider the bivariate copula data $\mathbf{u}_t = (u_t, v_t)$, whose joint distribution is given by a time-dependent parametric copula $C(\cdot, \cdot; \theta_t)$, or

$$(u_t, v_t) \sim C(u_t, v_t; \theta_t), \quad \theta_t \in \Theta \subset \mathbf{R}^k \tag{15}$$

with $k$ the number of parameters. In our paper, we will only consider $k = 1$ for the dynamic copulas. Hafner and Manner (2012) propose that the dependent parameter vector $\theta_t$ be driven by an unobserved stochastic process $\lambda_t$ through the mapping $\theta_t = \Phi(\lambda_t), \Phi : \mathbf{R} \to \Theta$. Note that the functional form of $\Phi$ is determined on the basis of the associated copula, for the domain of dependent parameter varies across different copula settings. The latent process $\lambda_t$ is modelled as a Gaussian autoregressive process, which, in its simplest form, is given as follows:

$$\lambda_t = \alpha + \beta\lambda_{t-1} + \nu\epsilon_t \tag{16}$$

where $\epsilon_t \overset{i.i.d}{\sim} N(0, 1)$. In the same paper, the authors argue that, despite the possibility of using higher order AR processes, the one given by (16) suffices to capture the persistence and autocorrelation of the dependence parameter. For the sake of less computational burden, we will make use of the simplest AR(1) process. Integrating the process into the copula dependence parameter, we obtain a new class of dynamic pair-copula model(short SCAR hereafter). Besides the autocorrelation effects on the dependence parameter, we are also interested in how certain economic variable will affect its movement(as will be seen in Section IV.C). The resulting new ARX(1,1) process can be formulated as follows:

$$\lambda_t = \alpha + \beta\lambda_{t-1} + \rho x_t + \nu\epsilon_t \tag{17}$$

and accordingly we have another dynamic pair-copula, SCARX. It is important to note that both AR processes are assumed to be strictly stationary, i.e. $|\gamma| < 1$. In addition to this, $\nu$ is restricted to be positive for the identification reason(Hafner and Manner (2012)).

## B.2. Towards Dynamic C-vine Copula Models

With the dynamic pair-copula clearly defined, we can now integrate the dynamic process into the multivariate C-vine copula. To elaborate on this, we revisit (13). This density, after adopting the time-varying dependence copula parameters, is given as follows:

$$f(x_{1t}, \ldots, x_{gt}; \Theta_t) = \prod_{k=1}^{g} f_k(x_{kt}) \times \prod_{j=1}^{g-1} \prod_{i=1}^{g-j} c_{j,j+i|1\ldots j-1,t} \tag{18}$$

where $c_{j,j+i|1\ldots j-1,t} := c_{j,j+i|1\ldots j-1,t}(u_{j|1\ldots j-1,t}, u_{j+i|1\ldots j-1,t}; \theta_t^{j,j+i})$ and $\Theta_t := \{\theta_t^e | e \in E\}$ denotes the set of time-varying dependence parameters of all involved dynamic pair-copulas. For each $e \in E$, the process is given by $\{\theta_t^e\}_{t=1}^T := \{\Phi_e(\lambda_t^e), \lambda_t^e = \alpha^e + \beta^e\lambda_{t-1}^e + \nu^e\epsilon_t^e | t = 1, \ldots, T\}$.

## C. Block-based Copula Construction

We now discuss the block-based construction of copula. In this context, a *block* designates a (multivariate) copula joining the variables within one cluster[3] and the *block-based construction* means using a certain technique to establish relations between blocks. The reasons why such a specification will useful are as follows.(i)Consider the construction of a certain $g$-dimensional mixed C-vine copula, the total number of pair-copulas involved is $\frac{g(g-1)}{2}$(Aas et al. (2009)). Thus, this figure will grow quadratically as the number of involved variables increase, such that when $g$ is big, the actual implementation becomes greatly stymied due to the sheer amount of pair-copulas to estimate.(ii)Sometimes the studied variables come in clusters by nature. For example, the stock returns of major American stock indices should be classified as one cluster as opposed to the other group that consists of the indices of European markets. In this case, the cross-cluster dependence[4] order is often-times vague. Although there exist algorithms that serve to determine such an order(e.g. see Czado, Schepsmeier, and Min (2012)), its interpretability tends to be equivocal. As a result, in place of explicitly specifying the cross-cluster dependence relations, one may be more interested in studying the interactions amongst clusters as a whole.

Consider a set of random vectors $(\mathbf{B}_1, \ldots, \mathbf{B}_n)$, with $\mathbf{B}_i = (X_{i1}, \ldots, X_{ig_i})$ the vector of variables from block $i$. For $i = 1, \ldots, n$, suppose there exists a transformation $H_i : \mathbb{R}^{g_i} \to \mathbb{R}$ such that

$$H_i(\mathbf{B}_i) = U_{B_i} \sim U(0, 1), \tag{19}$$

Provided that the variables within each block are joined by the C-vine copula[5], the transformation is then the probability integral transformation(PIT) thereof. Following the Sklar's Theorem(Sklar (1957)), it is not hard to derive that the joint pdf of $(\mathbf{B}_1, \ldots, \mathbf{B}_n)$ can be written as follows:

$$f(\mathbf{b}_1, \ldots, \mathbf{b}_n) = c(u_{B_1}, \ldots, u_{B_n}) \tag{20}$$

In order to delineate the dynamic feature of the dependence between blocks, $c(\cdot)$ needs to be decomposed into pair-copulas that follow (15). Put formally, for any $t = 1, \ldots, T$, we have

$$f(\mathbf{b}_{1t}, \ldots, \mathbf{b}_{nt}) = c(u_{B_{1t}}, \ldots, u_{B_{nt}}; \mathbf{\Theta}_t) \tag{21}$$

where $\mathbf{\Theta}_t$ is the set of dependence parameters for all involved pair-copulas. The remaining difficulty in the actual implementation is how to obtain the transformed copula data $U_{X_i}$, or in other words, how to estimate the cdf of a mixed C-vine copula given the dependence parameters. Sobering enough, we notice there exists no analytical formula for such estimation due to the complexity of C-vine copula specification. As a makeshift, we propose an IS-based Monte-Carlo simulation to work around the issue. The details regarding this method will be discussed in III.B.

[3]For example, geographically, the stocks listed in the American markets form a cluster and those in the Chinese markets another; conceptually, tech-company stocks form one cluster and financial-service companies another.
[4]For example, the dependence between one US stock and one European stock
[5]The choice of vine can of course be varied

# III. Model Estimation

## A. *Maximum Likelihood Estimation of Dynamic C-vine Copula*

Model estimation of the dynamic C-vine copula consists of estimating the parameters of (1) the (conditional) marginal distributions of the involved time series and (2) the dynamic copulas that define the linkages between the variables. To further elaborate on this, consider the copula-based joint density of time series $(X_{1,t}, \ldots, X_{g,t})$:

$$f(x_{1,t}, \ldots, x_{g,t}) = c(F_1(x_{1t}), \ldots, F_g(x_{gt}); \Theta_t) \times \prod_{i=1}^{g} f_i(x_{it}) \tag{22}$$

It is evident that the log-likelihood of the above density consists of two parts, the copula density and the marginal density. In terms of estimation, a two-step method labelled as the inference function for margins(Joe (2005)) is often used. The details are as follows. First, we maximise the marginal log-likelihood functions. Then, given the estimated marginal functions, we transform $X_t$ into copula data using probability integral transformation($u_{it} = F_i(x_{it})$) and maximise the copula log-likelihood function subsequently. In the same paper, it is shown that, under weak regularity condition, the resulting estimator is consistent(although not fully efficient). Furthermore, it is worth noting that the choice of marginal is crucial for good estimation of the copula data, as problems may occur given misspecified marginal functions(Kim, Kim, Liao, and Jung (2013)).

The next task is to estimate the C-vine copula density given the transformed sample. By the virue of the structure of the C-vine density(which is the product of an array of pair-copulas shown in the previous section), we can break the estimation of copula parameters down to sequential estimations of pair-copula parameters, instead of estimating the copula parameters all at once. In effect, the latter is barely feasible in practice due to the computational complexity of this estimation.

Following the idea of sequential estimation, the remainder of this section will elaborate on how to obtain parameter estimates for the dynamic pair-copula(III.A.1) and how to conduct sequential estimations using the estimates from the pair-copula(III.A.2).

### A.1. Estimation of Bivariate SCAR Models

For now, we consider the bivariate copula with one time-varying dependence parameter — the specification thereof given by (15)(16). Define the time series of the latent process $\Lambda = \{\lambda_t\}_{t=1}^{T}$ and the hyper-parameter vector(see (16)) to be optimised $\omega := (\alpha, \beta, \nu)$. Let $f(U, V; \Lambda, \omega)$ be the joint pdf of the observable copula variable $U$ and $V$, then the likelihood function of the parameter $\omega$ can be formulated as follows(Hafner and Manner (2012)):

$$\mathscr{L}(\omega; U, V) = \int f(U, V, \Lambda; \omega) d\Lambda \tag{23}$$

As $\lambda_t$ is dependent on $\lambda_{t-1}$, using conditional probability, the integrand in (23) can be decomposed as such:

$$\mathscr{L}(\omega, U, V) = \int \prod_{t=1}^{T} f(u_t, v_t, \lambda_t | \lambda_{t-1}, \omega) d\Lambda \tag{24}$$

This $T$-dimensional integral in general does not bear analytical or numerical solutions(Hafner and Manner (2012)). As a workaround, the authors resort to the Monte-Carlo(MC) integration in conjunction with a sampling technique known as *Efficient Importance Sampling* — brought forward by Richard and Zhang (2007). This is a variance-reduction technique with the use of an auxiliary sample $m(\lambda_t; \lambda_{t-1}, \mathbf{a}_t)$ which exploits the information contained in the observed variable $U$ and $V$ known by time $t$ by means of optimising the choice of parameter $\mathbf{a}_t$ — as will be discussed later. In the presence of the auxiliary sampler, the likelihood function is adjusted as:

$$\mathscr{L}(\omega, U, V) = \int \prod_{t=1}^{T} \left[ \frac{f(u_t, v_t, \lambda_t | \lambda_{t-1}, \omega)}{m(\lambda_t; \lambda_{t-1}, \mathbf{a}_t)} \right] \prod_{t=1}^{T} m(\lambda_t; \lambda_{t-1}, \mathbf{a}_t) d\Lambda \tag{25}$$

Drawing $N$ trajectories of $\hat{\Lambda}^{(i)}$ from the importance sampler $m(\cdot)$, the MC estimator of the likelihood function is calculated as such:

$$\hat{L}(\omega, U, V) = \frac{1}{N} \sum_{i=1}^{N} \left( \left[ \prod_{t=1}^{T} \frac{f(u_t, v_t, \lambda_t | \lambda_{t-1}, \omega)}{m(\lambda_t; \lambda_{t-1}, \mathbf{a}_t)} \right] \prod_{t=1}^{T} m(\lambda_t; \lambda_{t-1}, \mathbf{a}_t) \right) \tag{26}$$

The remaining question is how to properly choose the importance sampler to minimise the variance of the MC estimator. One possibility is proposed in Hafner and Manner (2012), which breaks the importance sampler into the product of two densities:

$$m(\lambda_t; \lambda_{t-1}, \mathbf{a}_t) = \frac{k(\lambda_t, \lambda_{t-1}; \mathbf{a}_t)}{\chi(\lambda_{t-1}; \mathbf{a}_t)}, k = p(\lambda_t; \lambda_{t-1}, \omega) \zeta(\lambda_t, \mathbf{a}_t) \tag{27}$$

where $p(\lambda_t; \lambda_{t-1}, \omega)$ is the conditional density of $\lambda_t$ and $\zeta(\lambda_t, \mathbf{a}_t) = exp(a_{1t}\lambda_t + a_{2t}\lambda_t^2)$ is a Gaussian kernel and $\chi(\lambda_{t-1}; \mathbf{a}_t) := \int k(\lambda_t, \lambda_{t-1}; \mathbf{a}_t) d\lambda_t$ is the normalising constant. Note that the choice of kernel $\zeta(\lambda_t, \mathbf{a}_t)$ can vary and for non-Gaussian latent process particularly, a different kernel is entailed(Hafner and Manner (2012)). Richard and Zhang (2007) propound that, given the specifications of $p(\cdot)$ following (16) and $m(\cdot)$ parameterised by some $\mathbf{a}_t = (a_{1t}, a_{2,t})$, the importance sampler has a normal density with the following mean and variance:

$$\mu_t = \sigma_t^2 \left( \frac{\delta + \gamma \lambda_{t-1}}{\nu^2} + a_{1t} \right), \sigma_t^2 = \frac{\nu^2}{1 - 2\nu^2 a_{2t}} \tag{28}$$

Note that for a ARX(1,1) latent process, we then have

$$\mu_t = \sigma_t^2 \left( \frac{\delta + \gamma \lambda_{t-1} + \rho X_t}{\nu^2} + a_{1t} \right). \tag{29}$$

In minimising the variance of (26), the optimal choice of EIS-specific parameter $\mathbf{A} = \{\mathbf{a}_t\}_{t=1}^T$ can be obtained. Richard and Zhang (2007) prove that, on the ground of the choice of density from the exponential family, the optimisation problem is simplified down to solving an Ordinary Least Squares(OLS) regression, which, for $t = 1, \ldots, T$, is defined as follows:

$$log(c(u_t, v_t; \theta_t(\omega))) + log\chi(\tilde{\lambda}_{t-1}^{(i)}(\omega)) = c_t + a_{1t}\tilde{\lambda}_t^{(i)}(\omega) + a_{2t}[\tilde{\lambda}_t^{(i)}]^2 + \eta_t^{(i)}, \quad \lambda_t^{(i)} \in \Lambda^{(i)} \tag{30}$$

where $\Lambda^{(i)}, i = 1, \ldots, N$ is a set of trajectories drawn from a certain sampler. Based on this, Hafner and Manner (2012) provide an algorithm(see Algorithm 1) to obtain the optimal $\mathbf{A}$ and use the resulting auxiliary sampler to perform Monte-Carlo Integration. Under the proposed scheme of computing the MC estimator for the likelihood function, we can obtain the optimal $\omega$ through maximum likelihood estimation. This will be implemented on MATLAB(MATLAB (2010)) with the use of the package provided by Hafner and Manner (2012). As a spin-off of EIS maximum likelihood estimation, the smoothed estimate of dependence parameter $\{\theta_t\}_{t=1}^T$ can be obtained by

$$\theta_t = \Phi(\hat{\lambda}_{t|T}) = \frac{1}{N}\sum_{i=1}^{N} \Phi(\tilde{\lambda}_t^{(i)}(\hat{\mathbf{a}}_t)), \quad t = 1, \ldots, T, \tag{31}$$

**Algorithm 1:** Procedures for obtaining MC estimator of log-likelihood function using Efficient Importance Sampling

1. Draw $N$ trajectories of $\Lambda^{(i)}$ from the natural sampler p
2. For $T = 1, \ldots, T$, solve the back-recursive OLS regression given by (30)
3. Draw $N$ trajectories of $\Lambda^{(i)}$ from the estimated importance sampler and solve the problem in step again.
4. Repeat steps 2 and 3 until $\hat{\mathbf{A}}$ converges.
5. Draw $N$ trajectories of $\Lambda^{(i)}$ from the importance sample obtained in step 4 and evaluate the MC estimator for the likelihood function given in (26)

## A.2. Sequential Estimation of dynamic C-vine Models

The sequential estimation for C-vine copula is brought forward in Czado et al. (2012). In their paper, the authors show that the sequential estimation for C-vine is close to the $d$-variate Maximum Likelihood estimation. From (13) and the tree representation(Figure 1), it can be seen that for a given tree level, the corresponding pair-copulas make use of the copulas in the precedent tree to transform the inputs for its predecessor into copula data and subsequently use these copula data as its pseudo observations. Specifically, this transformation is done through the $h(\cdot)$ function defined in (10). Here, we accommodate this transformation scheme to the dynamic setting of our C-vine model according to Almeida et al. (2016). Sighting along the first tree, for any $t = 1, \ldots, T$, the $h$ function of the copula $c_{ij}(u_{it}, u_{jt}; \theta_t^{ij})$ reads as follows:

$$u_{j|i,t} = h(u_{it}|u_{jt}, \theta_t^{ij}) \tag{32}$$

where $\theta_t^{ij}$ is the 'true' dependence parameter of the given copula at time $t$. However, through Maximum Likelihood estimation, one can only obtain the estimates for the hyper-parameters $\omega$, yet the latent copula parameter remains stochastic and unobservable. Consequently, (32) is not directly applicable. To get around this issue, an approximated $\hat{u}_{j|i,t}$ — obtained on the basis of $N$ EIS-generated trajectories $\theta_t^{(i)}$ — will be used as the pseudo observations[6]:

$$\hat{u}_{j|i,t} = \frac{1}{N}\sum_{i=1}^{N} h(u_{it}|u_{jt}, \tilde{\theta}_t^{(i)}) \tag{33}$$

For higher tree orders, the corresponding pseudo observations can be obtained recursively using the same scheme shown in (33). To help better understanding this approach, we consider an example in which we estimate the parameter of copula $c_{23|1}$. First, estimate the pair-copulas in the first tree, namely $c_{12}, c_{13}$. Then, use (33) to obtain pseudo observations $\hat{u}_{2|1}$ and $\hat{u}_{3|1}$ for the next tree. Based on these two inputs, the latent process of copula $c_{23|1}$ can be estimated.

Furthermore, to implement the sequential estimation method, we first make use of the SCAR model estimation package mentioned in III.A.1 to obtain estimation results of the dynamic pair-copulas and then channel the estimation outcomes to R(R Core Team (2013)). With the use of the CDVine package, we calculate $\hat{u}_{j|i,t}$(see (33)) and then pass the results back to the MATLAB(MATLAB (2010)) package to estimate the next pair-copula. We repeat this until all the pair-copulas are estimated.

## *B. Three-step Estimation of Dynamic Block-based Copula Construction*

As can be inferred from the title of this section, the estimation of a block-based copula consists of three steps. First, we estimate the copula model for each block using the sequential estimation approach detailed in III.A. With the dependence parameter estimates for each in-block copula, we utilise an algorithm proposed in the same section that performs probability integral transformation(PIT)[7] for a given C-vine copula. The resulting copula data associated to each block will then be passed into the block-specific copula — defined in (20)(21). Since the block-specific copula is nothing more than a (dynamic) copula, the estimation of such an object has already been thoroughly discussed in section III.A.

Next, we discuss the PIT for a C-vine copula. Due to the complexity of C-vine setting — which takes the transformations(partial derivatives particularly) of results from the previous layer as the input for the next layer — an analytical solution to the probability integration is all but in-feasible. The repercussion is again the demand for the Monte-Carlo integration. To illustrate this, we consider a C-vine copula for which we know the dependence structure and the dependence parameters. Suppose this copula involves random variables $\mathbf{X} = (X_1, \ldots, X_g)$ that are presumed

H

[6]For the ease of notation, we omit the superscript $ij$

[7]or put differently, to transforms the regular input data into copula data

to be *i.i.d* Normal. The joint density of these variables is $f(x_1, \ldots, x_g; \Theta)$ and its cdf:

$$F(x_1, \ldots, x_g) = \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_g} f(x_1, \ldots, x_g; \Theta) d\mathbf{x} \tag{34}$$

A naive approach to obtaining the Monte-Carlo estimator for this integral makes use of the fact that for standard normal density $f(\cdot)$, given a $c$ that is small enough, $f(c) \approx 0$. Therefore, (34) can be decomposed as follows:

$$F(x_1, \ldots, x_g) = \underbrace{\int_{-\infty}^{c} \cdots \int_{-\infty}^{c} f(x_1, \ldots, x_g; \Theta) d\mathbf{x}}_{o(\mathbf{x}) \approx 0} + \int_{c}^{x_1} \cdots \int_{c}^{x_g} f(x_1, \ldots, x_g; \Theta) d\mathbf{x} \tag{35}$$

Omitting the term $o(\mathbf{x})$, the Naive MC estimator is therefore

$$\hat{F}(\mathbf{x}) = \frac{1}{N} \sum_{s=1}^{N} f(\mathbf{x}^{(s)}) \tag{36}$$

where $N$ is the number of replications and $x_i^{(s)}$ is sampled element by element, with $X_i^{(s)} \sim U(c, x_i)$ for $i = 1, \ldots, g$. While this method might be efficient for univariate Monte-Carlo integration, it has two lethal shortcomings that render it highly inefficient in higher-dimensional sampling. First of all, for each variable, the entire variable space is treated equally, ignoring the fact that the bulk of standard normal density is distributed densely in a small area. Secondly, the method fails to take into account the dependencies between variables. This results in an exponential increase of the replication number $N$ as the dimensionality increases, rendering this method remotely viable for simulating higher-dimensional integrals. In the response to these issues, we propose a simulation method with the use of importance sampling(Kahn and Harris (1957)). Denote $w(\mathbf{X})$ as the auxiliary sampler. Following (26), given this importance sampler, the Monte-Carlo estimator for the integral can be calculated as such:

$$\hat{F}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} \frac{f(\mathbf{x}^{(i)})}{w(\mathbf{x}^{(i)})} \tag{37}$$

with $x^{(s)}$ sampled from the importance sampler $w$. We would like to choose the auxiliary sampler in such a way that it is easy to obtain and can largely reflect the feature of the joint distribution of the variables. This invites the use of truncated multi-normal distribution, as it guarantees that for each individual variable, the corresponding simulated values are sampled mostly from the important area; meanwhile, it models the dependencies between variables, which, although not precise, will greatly reduce the variance of the MC estimator. Moreover, estimating the density of this sampler is as simple as estimating the correlation matrix for the involved variables since that is the only parametric requirement for a truncated multivariate distribution. Given a truncated multi-normal distribution with the upper-bound vector $U$(a vector that contains the upper-bounds

of all variables) and the correlation matrix $\Sigma$, we can generate sample from $w(\mathbf{X}; U, \Sigma)$ with the use of the method proposed in Kotecha and Djuric (1999). This method employs the so-called GIBBS sampling(Casella and George (1992)), a Monte-Carlo Markov Chain(MCMC) algorithm for sampling from some multivariate distribution. Now, with all the ingredients ready, we can finally present the algorithm for the PIT of a dynamic C-vine copula:

**Algorithm 2:** Monte-Carlo Estimator for C-vine PIT

Estimate the correlation matrix of the involved variables, denoted as $\hat{\Sigma}$;
**for** $t = 1, \ldots, T$ **do**
  Generate $U_t \leftarrow \mathbf{X}_t$;
  Sample $X_t^{(1)}, \ldots, X_t^{(N)}$ from $w(\mathbf{X}; \hat{\Sigma}, U_t)$ ;
  Generate $\hat{F}_t \leftarrow \frac{1}{N} \sum_{s=1}^{N} \frac{f(\mathbf{X}^{(s)}; \Theta_t)}{m(\mathbf{X}^{(s)}; \hat{\Sigma}, U_t)}$;

The simulation is done on R(R Core Team (2013)) with the use of the 'tmvtnorm' package to generate truncated multivariate normal random numbers the 'CDVine' package to calculate the pdf of a given C-vine copula. We notice that for a three-dimensional C-vine copula, setting $N = 2000$ gives an estimator with decent variance($\sigma_{MC} \approx 0.03$).

# IV. Application

## *A. Modelling Dependence Between Major American Stock indices with bivariate SCAR*

This first application will be dedicated to illustrating the bivariate dynamic copula model — discussed in II.B.2 — as this class of models is the cornerstone of our proposed C-vine based dynamic copula models and the ensuing block-based copula. The dataset for this application is the same as the one used in Hafner and Manner (2012), which consists of daily observations of the Dow Jones Industrial Average(DJI) and Nasdaq Composite, ranging from 26 March 1990 until 23 March 2000. Based on this dataset, we reserve the last 250 observations for examining out-of-sample fit of the model and use the rest for estimation and in-sample fit — the same train-test split as in Hafner and Manner (2012). We expect the outcomes to be highly similar to the correspondings in their paper, but foresee tiny discrepancies brought by simulation-related factors(e.g. the use of different seeds of random number generator).

### A.1. Estimation and In-Sample Fit

First, for each index, we fit the demeaned returns to the stochastic volatility(SV) model. The estimation of such a model is conducted by means of EIS, using the Matlab(MATLAB (2010)) package provided by Hafner and Manner (2012). In this paper, the details of the SV model estimation will not be discussed, but it can be found in Richard and Zhang (2007) if interested. Following this model, the parameter estimate results are shown in Table IV in Appendix B, together with

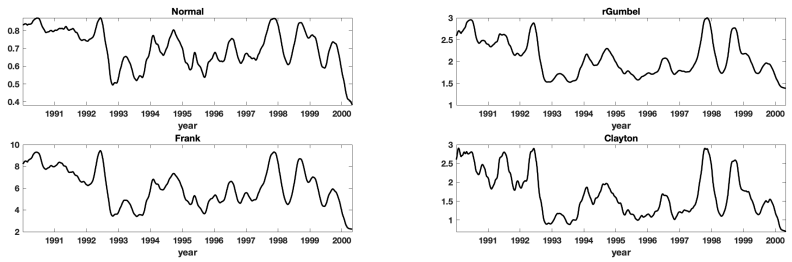



***Figure 2. Smoothed time paths of dependence parameters corresponding to four different copulas.*** *The calculation is based on (31)*

.

the estimated volatility (displayed in Figure 6b). Using the results from SV models, we obtain the marginal distribution of each return series and can thereby transform them into $U(0,1)$ random variables(shown in Figure 6a in Appendix B) — the copula data — with the use of PIT. The data will then be fed into the dynamic SCAR model and the estimation of the underlying process of the dynamic copula be done using the approach discussed in III.A.1. Following Hafner and Manner (2012), we consider four different copulas, namely the Gaussian copula, rotated Gumbel copula, Frank Copula and Clayton Copula and see which copula gives the best overall performance. Also note that from Figure 6a, a certain extent of tail dependency can be observed amongst the copula data. Given the use of the aforementioned copulas, the estimation results are presented in Table VI in Appendix B. In comparison with the results in Hafner and Manner (2012), we observe only minute differences which can be attributed to simulation-related disturbance. The underlying dependence processes with all copulas show a strong persistence, echoing findings in earlier studies. Of all the models, the Gaussian copula shows the best in-sample fit in terms of log-likelihood value. Considering the potential presence of tail dependence, we can suggest that the time-varying setting of dependence parameter enable the Gaussian copula to overcome its widely-acknowledged inability of capturing such a feature. The smoothed time paths of dependence parameters of the involved copula models are displayed in Figure 2. Despite differences in scale, the dependence parameter paths for all copulas resemble each other. Noteworthy is the drastic drop in dependency in 2000. The explanation regarding this abnormal observation can be found in Engle (2002).

## A.2. Out-of-sample Comparison

The details regarding how to obtain the r-step ahead forecasts are discussed in Appendix E. As mentioned before, the last 250 observations of the dataset will be used for the purpose of out-of-sample comparison. We transform these observations into copula data using the SV model obtained from IV.A.1 and re-estimate the four copula models in order to obtain the smoothed time paths of dependence parameter for the test period. So in doing, we have the proxy for the realised dependence parameter within the test period, with which we can compare the performance of the r-step forecasts from different models. Also note that (i) with the use of this proxy, the first realised observations, obtained from different estimated sample than the previous, will likely be far-

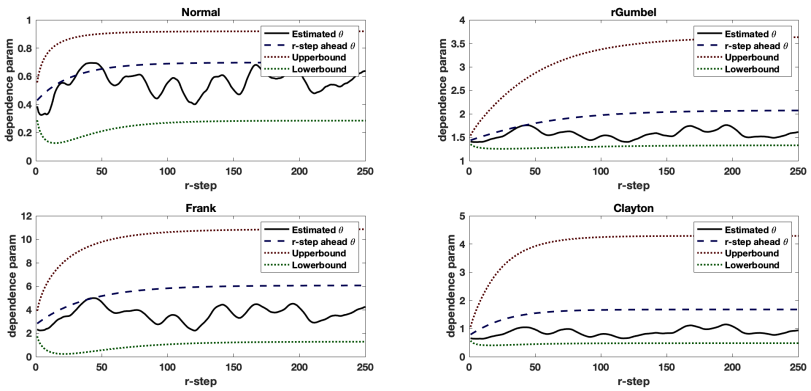



***Figure 3. Out-of-sample forecasts of the dependence parameter[Dow Jones and Nasdaq]****. The dash lines denote the out-of-sample* $\hat{\theta}_{T+r}$*; the dotted lines the* 95% *confidence bands; the normal lines the 'true'* $\theta$

off from the one-step forecasts; (ii)as the underlying is an AR(1) process by design, the r-step ahead forecasts will tend to the unconditional mean of the associated process as the step size r increases. The forecast results, along with the associated 95% confidence bands and the realised dependence parameter are shown in Figure 3. From these graphs, we can see that the r-step ahead forecasts for all copula models reflect closely the realisations. Next to this, the realisations for all copulas also fall within the given confidence interval. These observations suggest that the forecasting results be reasonably accurate, contextualising the difficulty of r-step ahead forecasting.

## *B. C-vine Models and Portfolio Management*

In this section, we discuss the use of the C-vine copula models — both dynamic and static — in depicting the dependencies between asset returns, constructing portfolios accordingly and how portfolios from different models compare to each other in terms of performance. Besides the C-vine models, a DCC GARCH model will also be treated in our analysis for comparison purposes. Specifically, we emulate an interesting context in which investors minimise draw-down risks while maintaining a certain level of portfolio returns and see which models best help achieve such a goal.

The data used for this section consists of daily returns of Dow Jones Industry Index, Nasdaq Index, Cotation Assistée en Continu 40(CAC 40), Deutscher Aktienindex 30(DAX 30), ranging from 1 January 2001 up to and including 31 December 2018. All returns are multiplied by 100. The sample from 1 January 2016 to 31 December 2018 will be used for out-of-sample analysis.

### B.1. Model Estimation and In-sample Fit

First, we convert the original data into copula data using the SV model. We then employ an order-selection technique proposed in Czado et al. (2012) — an iterative process which maximises the sum of the pair-wise Kendall's $\tau$ w.r.t each tree sequentially — to determine the variable order for C-vine copula models. The resulting order(obtained using 'CDVine' package in R(R Core Team (2013))) is {DAX 30, DJI, Nasdaq, CAC 40}. With the variable order, the next step is to

***Table I*** **In-sample fit Comparison amongst C-vine models and DCC-GARCH**

| Model | logl | AIC | BIC | Number of Parameters |
|---|---|---|---|---|
| SCAR C-vine | **-17500.01** | **35056.02** | **35008.49** | 28 |
| Static C-vine | -19550.25 | 39136.51 | 39108.97 | **18** |
| DCC-GARCH | -19675.46 | 39399.04 | 39553.99 | 24 |

*Note: In-sample fit comparisons in terms of log-likelihood, the Akaike Information Criteria and Bayesian Information Criteria, with the bald font highlighting the best fitting model.*

choose copulas for the involved pairs. With regard to the static C-vine model, the copulas are selected according to sequential estimation of pair-copulas and their associated Akaike Information Criteria(AIC) — a method proposed in Hans (2007). As for the dynamic C-vine model, two decisions have to be made, namely the choice between dynamic and static copula and the choice of copula family. Following the observations from the in-sample fit outcomes in Hafner and Manner (2012) and in Section III.A.1, we decide to apply Gaussian Copula to all dynamic pairs on the ground that the time-varying feature effectively equips the copula the agility to capture different tail-dependencies. In terms of whether or not the dynamic feature should be enabled, we employ the LR test against time-varying parameters(see Hafner and Manner (2012)). It turns out that, except for the last copula-pair, all the copulas appear to be time-varying.

The results of model selection and estimation of the C-vine copula models are displayed in Table VII in Appendix H. Furthermore, contour plots regarding all copulas selected for the static C-vine can be found in Appendix F if interested. In regard to the DCC GARCH, a model without specification on the mean process is chosen[8]. Table I shows the in-sample fit results of the three models. As can be seen, the dynamic C-vine model, despite having the most parameters, shows the best in-sample fit in terms of all criteria used. Following the dynamic C-vine model is its static counterpart — the most parsimonious choice amongst the three yet manifests higher log-likelihood relative to DCC model.

## B.2. Out-of-Sample Portfolio Performance Analysis

Given the broad applications of copula models in the realm of risk management (thanks to its ability to capture tail dependency), one natural usage of C-vine models is to construct risk-managed portfolios with certain expected returns. Interested, we opt to build the portfolios in such a way that the Conditional Value-at-Risk(C-VaR) is optimised. The reasons for choosing such a risk metric instead of the more widely-used Value-at-Risk(VaR) can be found in Rockafellar and Uryasev (2000), in which the authors argue that CVaR is a more coherent risk measure in the context of portfolio positions. In the same paper, it is shown that optimising the C-VaR portfolio is, in essence, a stochastic optimisation problem and can be approximated by a linear programming problem, considering the use of Monte-Carlo Integration. Given a set of candidate assets $\mathbf{X} = \{\mathbf{x}_t\}_{t=1}^{T}$, for any given point in time $t$, we sample $N$ observations[9] from some predicted

[8]Due to the size limit of our paper, we will not report the estimation results. But it is worth mentioning that the results of estimation and prediction are obtained using 'mvgarch' package in R(R Core Team (2013)).

[9]Sampling from C-vine copula is enabled by 'CDVine' package in R(R Core Team (2013))

distribution $f(\mathbf{X};\Theta_t)$ and solve the following linear programming[10] proposed by Rockafellar and Uryasev (2000) to minimise C-VaR given a return level $\mathbf{R}$:

$$\min_{(\mathbf{w_t},\alpha_t)} \mathscr{F}_\alpha(\mathbf{w_t},\alpha_t) = \alpha_t + \frac{1}{N(1-\beta)}\sum_{k=1}^{N}[-\mathbf{w}^T\mathbf{x}_t^{(k)} - \alpha_t]^+ \tag{38}$$

$$s.t. \qquad \mu(\mathbf{w_t}) \leq -\mathbf{R} \tag{39}$$

$$\mathbf{w_t}^T\mathbf{1} = 1 \tag{40}$$

where $\mathbf{w}_t$ is the portfolio weight which we aim to optimise, $\alpha_t$ is the Value-at-Risk, $\beta$ is the threshold probability — often set to be 0.90, 0.95 or 0.99 in practice — and $x_t^{(k)}$ is the $k$-element of the simulated returns. Since the distributions characterised by all the involved models are time-varying[11], one should, in theory, update the optimal portfolio weights every day. But this is not the least viable in practice. As an alternative, weekly re-balancing is to be adopted. This amounts to an exercise of $h$-step ahead forecasts of the distributional parameters. In practice, we estimate the full sample in order to obtain the realisations in the test sample[12]. With regard to the C-vine models, at the beginning of each week, we conduct out-of-sample forecasting of the dependence parameters and the conditional volatility up to 5-step ahead and use the average of our forecasts as the predicted results for the given week. Then we solve the linear programme problem thereof to update the optimal portfolio weight. In respect of the DCC-GARCH, the predicted covariance matrix of asset returns for the coming week is approximated by the one-step ahead forecasting value[13]. For all the portfolios, the expected daily portfolio return $\mathbf{R}$ in (39) is set to be $\mathbf{R} = 0.04$. In addition, another dynamic C-vine-based portfolio, with the sole aim of controlling portfolio risk, is also constructed. Such a portfolio is obtained by dispensing the minimal expected return constraint and restrict the largest position(both long and short) on any asset $i$ to $|\mathbf{w}_t^i| = 2$.

Figure 4 summarises the performance of cumulative returns, monthly returns, Conditional VaR(shorthand Historic ES in the graphs) and draw-down risk of each portfolio involved, juxtaposed with the market average *1/N* portfolio[14]. We observe that the dynamic C-vine model performs exceptionally well in terms of cumulative returns, almost double that of the second highest portfolio(the DCC-GARCH portfolio). This, however, comes at the expense of a much higher volatility and draw-down risk relative to the DCC-GARCH portfolio and the *1/N* portfolio. This issue is also visible in the static C-vine portfolio, though the compensation for high risk is much lower for this portfolio compared to the former. In stark contrast to the previous two C-vine portfolios, the risk-controlled dynamic C-vine portfolio manifests a notably lower draw-down risk due to the relaxation of return target. But a deep and long-lasting plummet occurred in late 2018, rendering the previous profits all but non-existent. Interestingly, while the headwind of recession in

[10]The linear programming optimisation is done using 'lpsolveAPI' package in R(R Core Team (2013))

[11]Even Static C-vine is no exception since the marginal processes are modelled by SV models, which assumes changing conditional volatility

[12]In this sense, we perform pseudo out-of-sample study

[13]This is obtained with the use of 'rmgarch' Package in R(R Core Team (2013))

[14]The results are obtained using the 'PerformanceAnalytics' package in R(R Core Team (2013)).

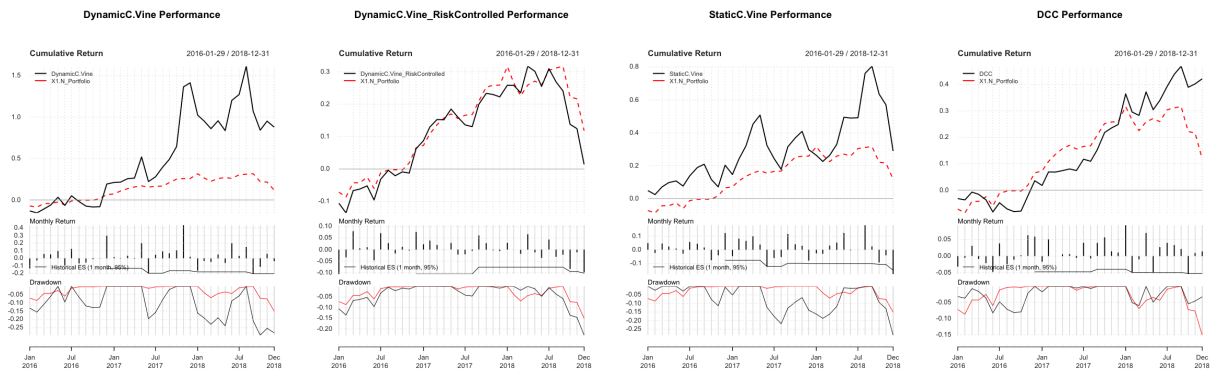



***Figure 4.*** *Summary of performance of cumulative returns, monthly returns, Conditional VaR(shorthand Historic ES in the graphs) and draw-down risk of each portfolio involved, juxtaposed with the market average 1/N portfolio(red lines). The involved assets(indices) are as follow: {DAX 30, DJI, Nasdaq, CAC 40}. All portfolios are re-balanced on a weekly basis, according to the respective pseudo out-of-sample density predictions. Due to the page limit, we can only put a condensed version here. For the graphs in full size, please see Appendix A.*

***Table II*** **Risk-Adjusted Portfolio Performance**

| **Risk Adjusted Metrics** | Methods | | | |
|---|---|---|---|---|
| | Sharpe Ratio | Sortino Ratio | Omega Ratio | VaR(99%) |
| D. C-vine | 0.195 | 0.375 | 0.732 | 0.206 |
| D. C-vine RC | 0.031 | 0.043 | 0.085 | 0.112 |
| S. C-vine | 0.131 | 0.201 | 0.393 | 0.169 |
| DCC-GARCH | **0.290** | **0.552** | **1.097** | **0.065** |
| 1/N | 0.111 | 0.150 | 0.352 | 0.088 |

*Note: Risk-adjusted performance comparison amongst all portfolios. The bald font highlights the best performing model for the given risk metric.*

this period blows the hunky-dory situations of all three C-vine-based portfolios, the DCC portfolio sails through the market hiccup. In general, this portfolio shows no notable return dip and remains profiting almost constantly. Thanks to this feature, despite having a much lower absolute return than dynamic C-vine portfolio, it completely out-duels its competitors in terms of risk-adjusted performance measures, as shown in table II. In effect, the large scale fluctuations of the C-vine portfolios is due to the fact that the use of this type of model in characterising joint distribution of candidate assets tend to yield radical weight choice. As a consequence of high leverage, even a slight downturn of the market indices will likely cause a deep draw-down of portfolio return. Unfortunately, appropriate account for this interesting phenomenon has not been discovered in this study and hence, we will leave this to further research.

In sum, there is no overall best performing portfolio. Nonetheless, in the sense of balancing portfolio downside risks and returns (which is the very goal of constructing C-VaR optimised portfolio), DCC-GARCH clearly stands out amongst all competing models.

## *C. C-vine Block and Holistic Dependence*

In the last application, we study the holistic dependence between the Western and Eastern Stock markets with the use of Block-based Copula construction discussed in II.C. The Western

***Table III*** **Parameter Estiamtes**

| | $\alpha$ | $\beta$ | $\phi$ | $\nu$ | log-likelihood |
|---|---|---|---|---|---|
| Gaussian SCAR | 0.4627 | 0.1187 | — | 0.1757 | 823.918 |
| Gaussian SCARX | 0.4616 | 0.1185 | 0.0043 | 0.1750 | 824.074 |
| Gaussian Copula | 0.4647 | — | — | — | 817.260 |

*Note: the log-likelihood denote the one corresponding to copula density function. For static Gaussian copula, $\alpha$ denotes the correlation parameter thereof.*

block consists of daily returns of stock market indices used in IV.B and the Eastern of the Shanghai Composite Index(SSE) and the Nikkei Stock Average(Nikkei 225), both sampled from 1 January 2001 to 12 December 2018 and re-scaled by multiplying 100. In addition, we are also interested in studying whether the sentiment of risk-aversion will influence the dependence between these two markets. Such sentiment is reflected by a gold-price-based dummy variable $D = \{d_t\}_{t=1}^{T}$ obtained recursively using a moving-window method(implemented on R(R Core Team (2013))):

$$d_t = I_{\{s_t \geq 2\}} \quad with \quad s_t = \sum_{i=1}^{h} [r_{t-h}]^+ \tag{41}$$

where $I_{\{A\}}$ is the indicator function which equals to 1 if $A$ is true and 0 otherwise; $h$ is the size of moving window — set to be 22 in practice — and $r_t$ is the daily return of some gold price index. Specifically, we make use of the London Gold Fixing sampled from the same period as other data used in this application.

In a step to studying the holistic dependence of the Western and Eastern market, we make use of dynamic Copula models to characterise the distribution within each block. For the Western block, the estimation results using the dynamic C-vine model has already been displayed in II.C. For the Eastern block, we transformed the return data into copula data using the SV model and then fit it to a bivariate Gaussian SCAR model. Next, using the IS-based PIT proposed in III.B, we obtain the copula data for each block. The scatter plot regarding the results is presented in Figure 9 in Appendix G. With this data at hand, we are particularly interested in studying (i)whether the dependence between these two blocks is time-varying; (ii)whether the risk-aversion sentiment indicator has any explanatory power for the behaviour of the dependence parameter. As such, we estimate three bivariate models, namely the bivariate Gaussian SCAR, the static Gaussian copula and the Gaussian SCARX model(see (17)) using the risk-aversion sentiment indicator as external explanatory variable for the latent process. Moreover, the estimated sample skips the observations of the first months in order to obtain the first value of the risk-aversion sentiment indicator. The estimation results for the chosen models are shown in Table III. Based on the obtained log-likelihood of these three models, we then conduct two LR-tests to verify whether the parameter is indeed time-varying and whether the risk-aversion indicator is significant. The p-value for the these tests is $p = 0.00$ and $p = 0.5767$, respectively. Therefore, we conclude that the dependence parameter of Gaussian Copula is indeed time-varying and, surprisingly, the risk-aversion sentiment — represented by our indicator — has little influence on the dependence between the Western and Eastern market.

# V. Conclusions and Further Research

In an attempt to address the great need of flexible time-varying multivariate distribution, we developed the dynamic C-vine model that fuses the SCAR model into the C-vine copula. In conjunction with EIS and the sequential estimation technique, we estimated the model using the traditional approach, maximum likelihood. The in-sample study showed its superiority over the competing multivariate models such as static C-vine and DCC-GARCH. Several limitations should be noticed in this study. For one thing, only a relatively low number of variables were considered in the modelling. This dodges the potential issues with higher-dimensional dynamic vine models, particularly the decision between truncated and complete dynamic vine model — an exercise studied extensively in Brechmann, Czado, and Aas (2012). Furthermore, we applied Gaussian copula to all involved pair-copulas in the constructed dynamic C-vine models based on the study of in-sample fit of SCAR. As an improvement therein, copula families fall beyond the list in our study can also be experimented. Another potential problem lies in the choice of marginal processes of the C-vine based models. Although, we suggested the SV model be the appropriate proxy for the 'true' marginal distributions, evidence towards the validity of this decision is generally missing. This can lead to certain problems associated with model mis-specification(Kim et al. (2013)). One notable application with the dynamic C-vine model pertains portfolio management using density forecasting thereof. To this end, we applied the proposed model and other competing models for constructing C-VaR optimised portfolios. The performance comparisons between the resulting portfolios evinced its usefulness in the realm of the risk-managed portfolios. Through this application, we notice several interesting questions for further research. Firstly, the use of C-vine class models to construct C-VaR portfolios always brings about high-leverage portfolio strategies. This can lead to less immunity to the caprice of the financial market. Next, we notice for all C-vine-based portfolios, a great recession in cumulative returns occurred in 2018, scuppering the once-extraordinary performance. Accounts for this happening is also of great interest. Lastly, the asset pool for portfolio construction only contains four assets. In the future, one should investigate inclusions of different numbers of assets, whereby gaining insights into the efficacy of portfolio diversification using the (dynamic) C-vine copula models. As another contribution of this paper, we proposed a simulation-based PIT to obtain the cdf of a dynamic C-vine copula and subsequently employed the proposed method to construct the block-based copulas. With the use of such, we studied the holistic dependence between the Western stock market and Eastern stock market. However, shackled by the page limit, we failed to deliver the research in its full depth. It generally lacks the due amplitude of the risk-aversion sentiment indicators and the copula models choice. Owing to the same reason, the discussion on this regard also came with great haste. As for the algorithm *per se*, problems occur when the number of variables in a C-vine copula grows higher, as the resulting transformation will then centre mostly in the vicinity of 0 due to the curse of dimensionality(Bellman, Corporation, and Collection (1957)). The potential solution to this problem will again be left to future study. Lastly, as a general remark regarding the sequential estimation, the absence of an appropriate approach to obtaining the standard error for parameter estimates should be coped with in the future.

# REFERENCES


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

Acar, Elif F., Claudia Czado, and Martin Lysy, 2019, Flexible dynamic vine copula models for multivariate time series data, *Econometrics and Statistics* .

Almeida, Carlos, and Claudia Czado, 2012, Efficient bayesian inference for stochastic time-varying copula models, *Computational Statistics  Data Analysis* 56, 1511 – 1527.

Almeida, Carlos, Claudia Czado, and Hans Manner, 2016, Modeling high-dimensional time-varying dependence using dynamic d-vine models 32, 621–638.

Bauwens, Luc, S*é*bastien Laurent, and Jeroen V. K. Rombouts, 2006, Multivariate garch models: a survey, *Journal of Applied Econometrics* 21, 79–109.

Bedford, Tim, and Roger M. Cooke, 2002, Vines–a new graphical model for dependent random variables, *Ann. Statist.* 30, 1031–1068.

Bellman, R., Rand Corporation, and Karreman Mathematics Research Collection, 1957, *Dynamic Programming*, Rand Corporation research study (Princeton University Press).

Brechmann, Eike C., Claudia Czado, and Kjersti Aas, 2012, Truncated regular vines in high dimensions with application to financial data.

Casella, George, and Edward I. George, 1992, Explaining the gibbs sampler, *The American Statistician* 46, 167–174.

Creal, Drew, Siem Jan Koopman, and Andr Lucas, 2013, Generalized autoregressive score models with applications, *Journal of Applied Econometrics* 28, 777–795.

Creal, Drew D., and Ruey S. Tsay, 2015, High dimensional dynamic stochastic copula models, *Journal of Econometrics* 189, 335 – 345, Frontiers in Time Series and Financial Econometrics.

Czado, Claudia, Ulf Schepsmeier, and Aleksey Min, 2012, Maximum likelihood estimation of mixed c-vines with application to exchange rates, *Statistical Modelling* 12, 229–255.

Demarta, Stefano, and Alexander J. McNeil, 2005, The t copula and related copulas, *International Statistical Review* 73, 111–129.

Dias, Alexandra, 2002, Change-point analysis for dependence structures in finance and insurance .

Engle, Robert, 2002, Dynamic conditional correlation: A simple class of multivariate generalized autoregressive conditional heteroskedasticity models, *Journal of Business  Economic Statistics* 20, 339–350.

Fink, Holger, Yulia Klimova, Claudia Czado, and Jakob Stober, 2017, Regime switching vine copula models for global equity and volatility indices, *Econometrics* 5.

Giacomini, Enzo, Wolfgang Hardle, and Vladimir Spokoiny, 2009, Inhomogeneous dependence modeling with time-varying copulae, *Journal of Business & Economic Statistics* 27, 224–234.

Hafner, Christian M., and Hans Manner, 2012, Dynamic stochastic copula models: estimation, inference and applications, *Journal of Applied Econometrics* 27, 269–295.

Hamilton, J., 1994, *Time Series Analysis* (Princeton University Press: Princeton, NJ).

Hans, Manner, 2007, Estimation and model selection of copulas with an application to exchange rates .

Harvey, Andrew C., and Neil Shephard, 1996, Estimation of an asymmetric stochastic volatility model for asset returns, *Journal of Business & Economic Statistics* 14, 429–434.

Heinen, Andras, and Alfonso Valdesogo Robles, 2009, Asymmetric capm dependence for large dimensions: The canonical vine autoregressive copula model .

Joe, Harry, 1996, *Families of $m$-variate distributions with given margins and $m(m-1)/2$ bivariate dependence parameters*, volume Volume 28 of *Lecture Notes–Monograph Series*, 120–141 (Institute of Mathematical Statistics, Hayward, CA).

Joe, Harry, 2005, Asymptotic efficiency of the two-stage estimation method for copula-based models, *Journal of Multivariate Analysis* 94, 401 – 419.

Joe, Harry, and Taizhong Hu, 1996, Multivariate distributions from mixtures of max-infinitely divisible distributions, *Journal of Multivariate Analysis* 57, 240 – 265.

Kahn, H., and T. E. Harris, 1957, *National Bureau of Standards applied mathematics series* .

Kim, Daeyoung, Jong-Min Kim, Shu-Min Liao, and Yoon-Sung Jung, 2013, Mixture of d-vine copulas for modeling dependence, *Computational Statistics  Data Analysis* 64, 1–19.

Kotecha, J. H., and P. M. Djuric, 1999, Gibbs sampling approach for generation of truncated multivariate gaussian random variables 3, 1757–1760 vol.3.

Kurowicka, and Cooke, 2004, Distribution-free continuous bayesian belief nets, *Modern Statistical and Mathematical Methods in Reliability* 309–322.

Loran, CHOLLETTE, HEINEN Andreas, and VALDESOGO Alfonso, 2008, Modelling international financial returns with a multivariate regime switching copula, Discussion Papers (ECON - Dpartement des Sciences Economiques) 2008011, Universit catholique de Louvain, Dpartement des Sciences Economiques.

Manner, Hans, and Olga Reznikova, 2012, A survey on time-varying copulas: Specification, simulations, and application, *Econometric Reviews* 31, 654–687.

MATLAB, 2010, *version 9.3.0 (R2017b)* (The MathWorks Inc., Natick, Massachusetts).

Oh, Dong Hwan, and Andrew J. Patton, 2017, Modeling dependence in high dimensions with factor copulas, *Journal of Business & Economic Statistics* 35, 139–154.

Oh, Dong Hwan, and Andrew J. Patton, 2018, Time-varying systemic risk: Evidence from a dynamic copula model of cds spreads, *Journal of Business & Economic Statistics* 36, 181–195.

Okhrin, Ostap, Yarema Okhrin, and Wolfgang Schmid, 2013, On the structure and estimation of hierarchical archimedean copulas, *Journal of Econometrics* 173, 189–204.

Patton, Andrew J., 2009, *Copula–Based Models for Financial Time Series*, 767–785 (Springer Berlin Heidelberg, Berlin, Heidelberg).

R Core Team, 2013, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.

Richard, Jean-Francois, and Wei Zhang, 2007, Efficient high-dimensional importance sampling, *Journal of Econometrics* 141, 1385–1411.

Rockafellar, R. Tyrrell, and Stanislav Uryasev, 2000, Optimization of conditional value-at-risk, *Journal of Risk* 2, 21–41.

Savu, Cornelia, and Mark Trede, 2010, Hierarchies of archimedean copulas, *Quantitative Finance* 10, 295–304.

Sklar, M., 1957, Fonction de repartition a n dimensions et leurs marges, *Publications de l'Institut de Statistique de L'Universite de Paris 8* 229–231.

So, Mike K.P., and Cherry Y.T. Yeung, 2014, Vine-copula garch model with dynamic conditional dependence, *Computational Statistics  Data Analysis* 76, 655–671.

Stober, Jakob, and Claudia Czado, 2014, Regime switches in the dependence structure of multidimensional financial data, *Computational Statistics  Data Analysis* 76, 672 – 686, CFEnetwork: The Annals of Computational and Financial Econometrics.

Taylor, Stephen J, 1986, *Modelling Financial Time Series* (World Scientific Publishing Co. Pte. Ltd.).

U. Cherubini, W. Vechiatto, E. Luciano, 2004, *Copula methods in finance* (JOHN WILEY).

# Appendix A. C-VaR Portfolio Performance Summary

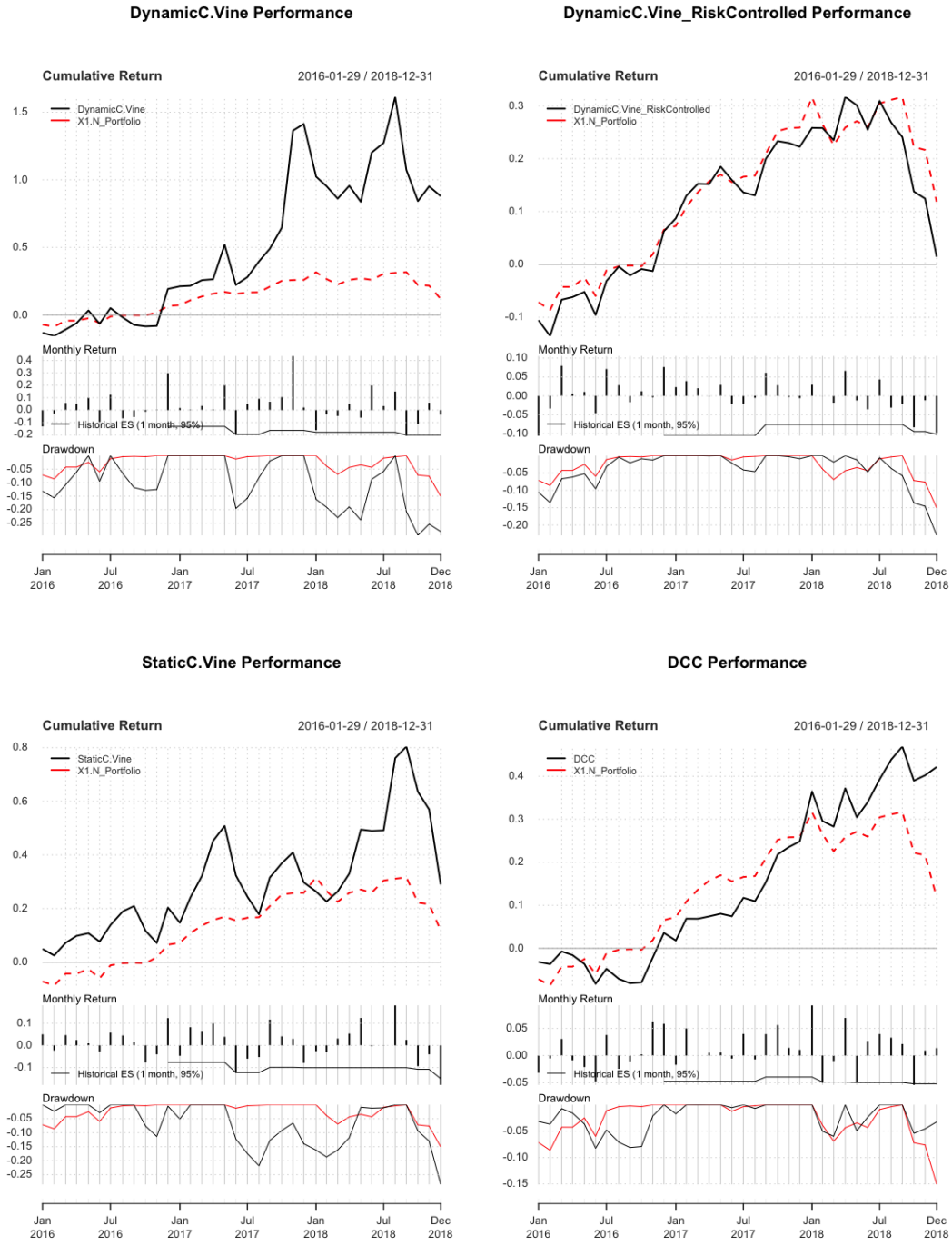


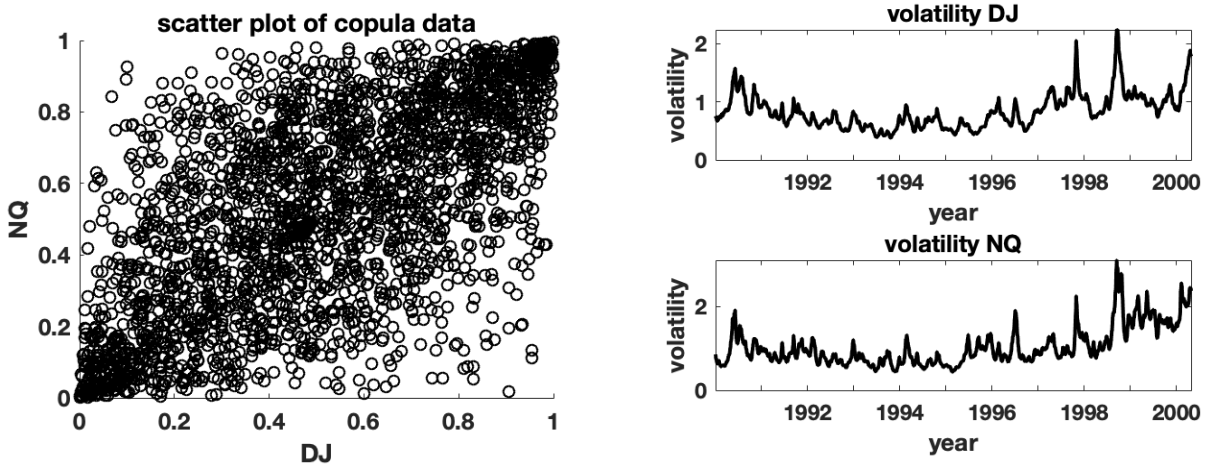

***Figure 5.*** *Magnified version of portfolio performance summary in IV.B.2.*

# Appendix B. SV Estimation

***Table IV*** **SV model Estimates**

| | DJ | NQ |
|---|---|---|
| $\alpha$ | -0.0111 | -0.0054 |
| | (0.0050) | (0.0042) |
| $\beta$ | 0.9787 | 0.9732 |
| | (0.0076) | (0.0078) |
| $\nu$ | 0.1566 | 0.2077 |
| | (0.0254) | (0.0268) |
| logl | -3137.48 | -3638.17 |

*Note: the figures in the parentheses report the corresponding standard deviations*




***(a)*** *Scatter plot of Copula Data* ***(b)*** *Volatility of DJ and NQ*

***Figure 6. SV Estimation Results.*** *(a) shows the scatter plot of the copula data respective to the return of Dow Jone(DJ) and Nasdaq(NQ). (b) plots the SV-estimated volatility of these two indices.*

# Appendix C. Copula Density Summary

***Table V*** **Copula Density Summary**

| | Density | $\theta$ | $\delta$ | $\Phi$ |
|---|---|---|---|---|
| Gaussian | $\frac{1}{\sqrt{(1-\theta^2)}} exp(\frac{2\theta xy - x^2 - y^2}{2(1-\theta^2)} + \frac{x^2+y^2}{2})$ | $\theta \in [-1, 1]$ | - | $\Phi(x) = \frac{exp(2x)-1}{exp(2x)+1}$ |
| Gumbel | $\frac{(log(u)log(v))^{\{(\theta-1)}[(-log(u))^\theta + (-log(v)^\theta]^{1/\theta} + \theta - 1\}}{[(-log(u))^\theta + (-log(v)^\theta]^{2-1/\theta} uv}$ $\times exp\{[(-log(u))^\theta + (-log(v))^\theta]^{1/\theta}\}$ | $\theta \in [1, \infty]$ | - | $\Phi(x) = exp(x) + 1$ |
| Clayton | $u^{(-1-\theta)} v^{(-1-\theta)} (u^{-\theta} + v^{-\theta} - 1)^{(-2-1/\theta)} (1 + \theta)$ | $\theta \in (0, -\infty)$ | - | $\Phi(x) = exp(x)$ |
| Frank | $\frac{exp((1+u+v)\theta)(exp(\theta)-1)\theta}{\{exp(\theta)+exp((u+v)\theta)-exp(\theta+u\theta)-exp(\theta+v\theta)\}^2}$ | $\theta \in (-\infty, \infty)$ | - | $\Phi(x) = x$ |
| BB1 | $\{1 + [(u^{-\theta} - 1)^\delta + (v^{-\theta} - 1)^\delta]^{1/\delta}\}^{-1/\theta}$ | $\theta \in (0, \infty)$ | $\delta \in [1, \infty)$ | - |
| BB8 | $\frac{1}{\delta}(1 - \{1 - \theta\}^{-\theta})\{1 - (1 - \theta u\}^\theta\}\{1 - (1 - \theta v\}^\theta\}$ | $\theta \in [1, \infty)$ | $\delta \in [0, 1]$ | - |
| $t$ | $f_{\delta,\theta}(t_\nu^{-1}(u), t_\nu^{-1}(v)) / f_\nu(t_\nu^{-1}(u)) f_\nu(t_\nu^{-1}(v))$ | $\theta \in [-1, 1]$ | $\delta \in (0, \infty)$ | - |

*Note: A summary of all the copulas used in this paper. The Gaussian Copula takes the inverse cdf of $u$ and $v$ as its input(i.e. $x = F^{-1}(u)$ and $y = F^{-1}(y)$). For those involved in the SCAR model, the number of dependence parameter is 1. BB1 and BB8 copulas are the joe-copulas given by Joe and Hu (1996). Both of them are used for the static C-vine copula only, therefore, the transformation function is not needed. The t-copula density is derived in Demarta and McNeil (2005). Notation-wise, $\theta$ is the off-diagonal element of the correlation matrix and $\delta$ is the degree of freedom; $f_{\delta,\theta}$ denotes the joint density of a bivariate $t_2(\delta, \mathbf{0}, \theta)$-distributed random vector and $f_\delta$ is the density of the univariate standard t-distribution.*

# Appendix D. SCAR Estimation Results

***Table VI*** **SCAR parameter estimates**

| | $\alpha$ | $\beta$ | $\nu$ | logl | p-value(LR test) |
|---|---|---|---|---|---|
| Gaussian | 0.0312 | 0.9668 | 0.0841 | **901.2194** | 0.00 |
| | (0.0101) | (0.0105) | (0.0146) | | |
| Rot. Gumbel | -0.0014 | 0.9796 | 0.1064 | 865.7110 | 0.00 |
| | (0.0023) | (0.0085) | (0.0238) | | |
| Frank | 0.1564 | 0.9742 | 0.5512 | 789.3049 | 0.00 |
| | (0.0589) | (0.0094) | (0.1181) | | |
| Clayton | 0.0146 | 0.9594 | 0.1576 | 747.9096 | 0.00 |
| | (0.0071) | (0.0166) | (0.0389) | | |

*Note: Parameter estimate results together with the log-likelihood of the copula model associated. p-value corresponds to the LR-test for time-varying dependence parameter discussed in Hafner and Manner (2012). The rotate Gumbel copula density is defined as* $c_{rGumbel}(u,v) = c_{Gumbel}(1-u, 1-v)$. *The densities of the above copula are given in Appendix C.*

# Appendix E. SCAR Out-of-Sample Forecasting In-depth

We provide the derivation of out-of-sample forecasting of SCAR models given by Hafner and Manner (2012). The AR(1) specification of the latent process[15] is typically conducive to forecasting. Using the theory derived in Hamilton (1994), the r-step-ahead forecast given an estimate $\hat{\lambda}_T$ can be written as

$$\hat{\lambda}_{T+r} = \mu + \beta^r(\hat{\lambda}_T - \mu) \tag{E1}$$

where $\mu = \alpha/(1-\beta)$. The mean squared of the forecast error is

$$\sigma^2_{T+r} = \nu^2(1-\beta^{2r})/(1-\beta^2). \tag{E2}$$

Note that eventually, we would like to obtain the forecasting of $\theta = \Phi(\lambda)$. From table **??** we see that only in the context of the Frank copula are the transformed parameter and $\lambda$ are identical. In the case of the Gumbel and Clayton copulas, we use the results given in Hafner and Manner (2012) that for $\lambda_t|I_{t-1} \;\; N(\mu_t, \sigma_t)$, the r-step-ahead forecast of $\theta_t = exp(\lambda)_t)$ is as such:

$$\theta_{T+r} = exp(\hat{\lambda}_{T+r} + \frac{\sigma^2_{T+r}}{2}) \tag{E3}$$

The confidence intervals for these forecasts are obtained using the associated quantiles of the log-normal distribution $logN(\hat{\lambda}_{T+r}, \sigma^2_{T+r})$.

Hafner and Manner (2012) argue that due to the non-linearity of the inverse Fisher transformation corresponding to the normal copula, a second-order Taylor expansion can be used to

[15]See (16)

approximate the transformation results. Given a $\lambda_t$ around $\mu_t$, we have

$$\Phi(\lambda_t) \approx \Phi(\mu_t) + \Phi^{'}(\mu_t)(\lambda_t - \mu_t) + \frac{1}{2}\Phi^{''}(\mu_t)(\lambda_t - \mu_t)^2 \tag{E4}$$

The conditional expectation thereof is then

$$\begin{aligned} E(\Phi(\lambda_t)|I_{t-1}) &\approx E\{\Phi(\mu_t) + \Phi^{'}(\mu_t)(\lambda_t - \mu_t) + \frac{1}{2}\Phi^{''}(\mu_t)(\lambda_t - \mu_t)^2|I_{t-1}\} \\ &= \Phi(\mu_t) + \Phi^{'}(\mu_t)E(\lambda_t - \mu_t|I_{t-1}) + \frac{1}{2}\Phi^{''}(\mu_t)E((\lambda_t - \mu_t)^2|I_{t-1}) \\ &= \Phi(\mu_t) + \frac{\Phi^{''}(\mu_t)}{2}\nu^2 \end{aligned} \tag{E5}$$

The r-step-ahead forecast is therefore

$$\hat{\theta}_{T+r} = \Phi(\hat{\lambda}_{T+r}) + \frac{-4(exp(2\hat{\lambda}_{T+r})exp(2\hat{\lambda}_{T+r}))}{(exp(2\hat{\lambda}_{T+r} + 1)^3}\sigma^2_{T+r} \tag{E6}$$

# Appendix F. Contour Plot of Static C-vine Copula

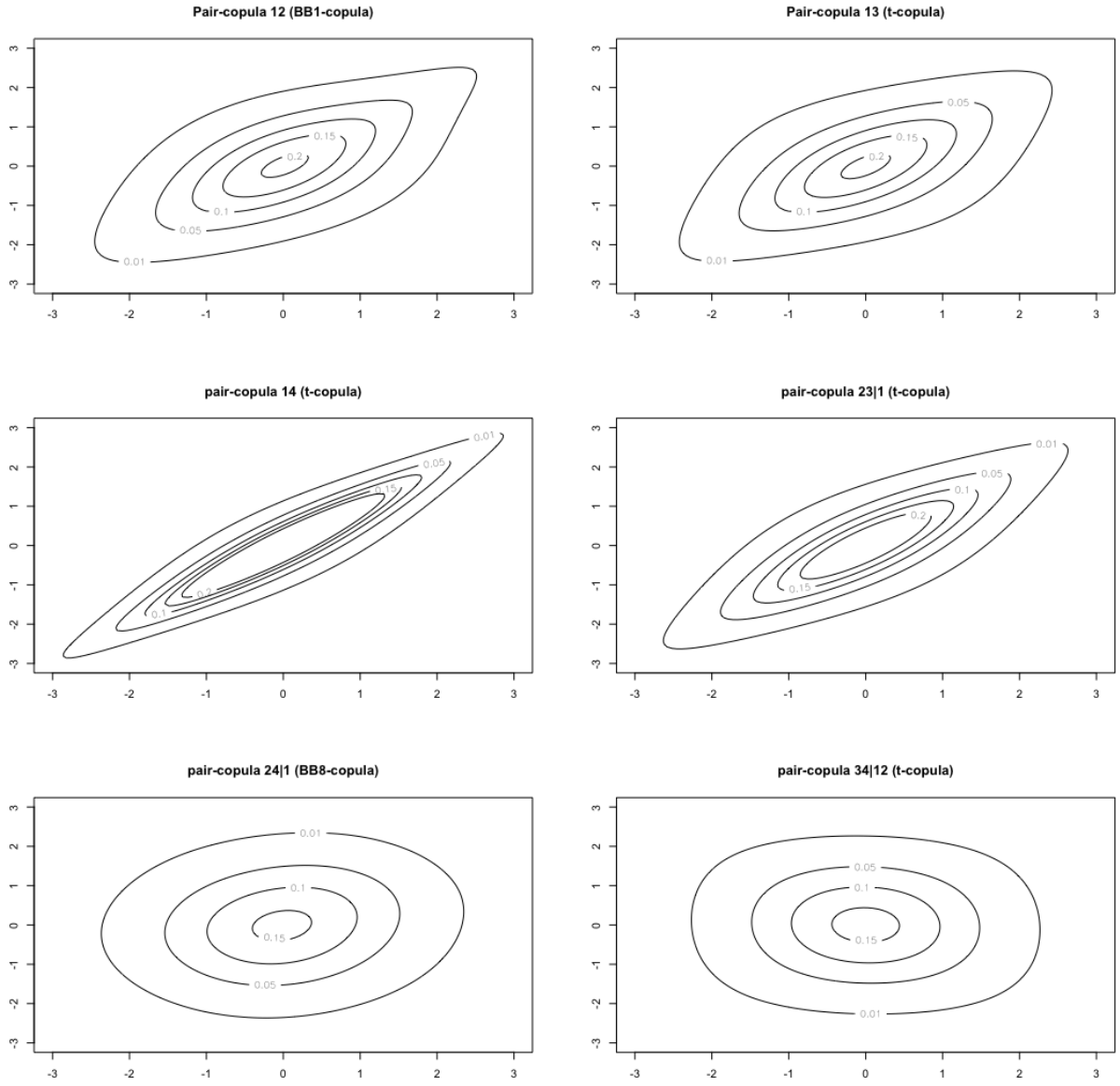



***Figure 7. Cotour Plot of Pair-Copulas involved in the Static C-vine.*** *The involved variables are ordered as follows: {DAX 30, DJ, NQ, CAC 30}*

# Appendix G. Dynamic C-vine Parameter Time Paths

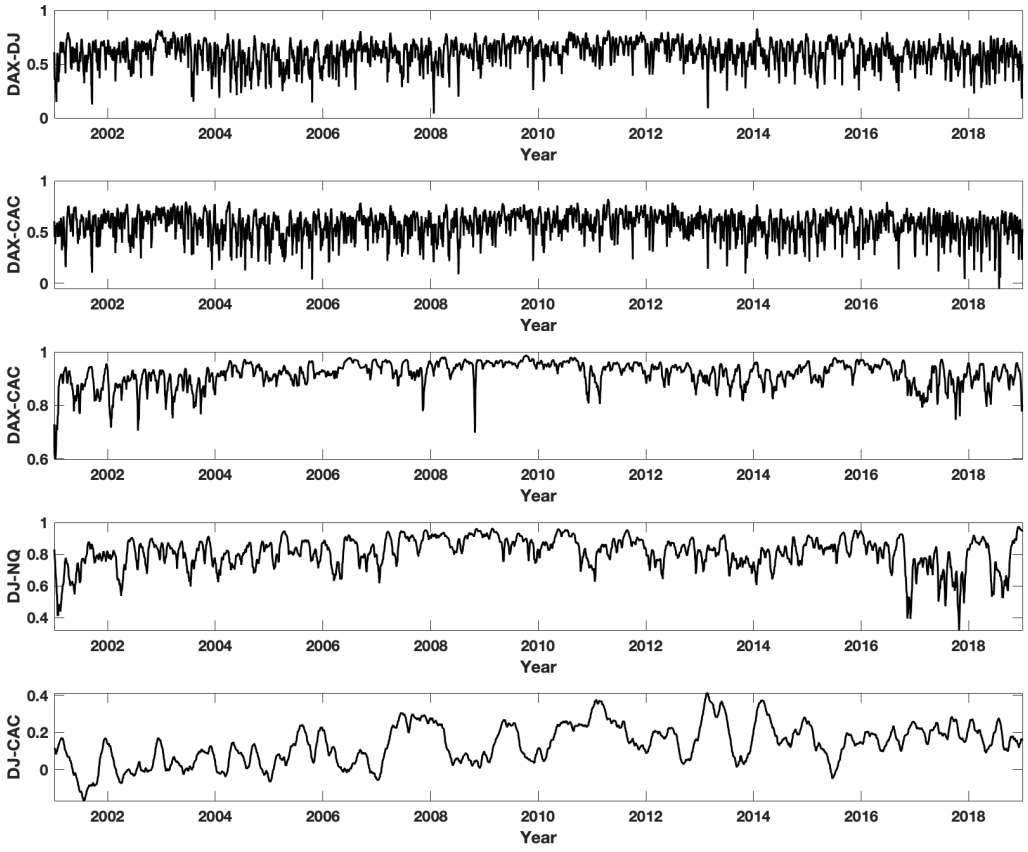



***Figure 8. Smoothed time paths of dependence parameters of all dynamic pair-copulas.*** *Each title of the y-axis indicates the associated pair-copula*

# Appendix H. C-Vine Models Estimation Results

***Table VII*** **Vine Copulas Estimation Results**

| | Dynamic C-vine | | | | Static C-vine | | |
|---|---|---|---|---|---|---|---|
| Pair | Type | $\alpha$ | $\beta$ | $\nu$ | Copula | $\theta_1$ | $\theta_2$ |
| 12 | Time-varying | 0.2337 | 0.6965 | 0.2601 | BB1 copula | 0.3427 | 1.445 |
| 13 | Time-varying | 0.2653 | 0.6336 | 0.2956 | t-copula | 0.5689 | 5.515 |
| 14 | Time-varying | 0.1445 | 0.9135 | 0.1481 | t-copula | 0.9206 | 4.888 |
| 23‖1 | Time-varying | 0.0677 | 0.9448 | 0.1202 | t-copula | 0.7964 | 5.672 |
| 24‖1 | Time-varying | 0.0033 | 0.9752 | 0.0391 | BB8 copula | 1.8672 | 0.518 |
| 34‖12 | Constant | -0.0401 | — | — | t-copula | -0.0397 | 10.763 |

*Note: Results of copula model selection and the corresponding parameter estimates. The involved indices are ordered as follow: {DAX 30, DJI, Nasdaq, CAC 40}.The choice of copula family for Dynamic C-vine model is set to be Normal copula. The standard errors for the dynamic C-vine parameter estimates cannot be obtained. The accounts for this can be found in Almeida et al. (2016). As a result, we did not report the standard errors in the table. Also note that the estimated $\alpha$ associated with the time-constant is equivalent to the estimated correlation parameter. Furthermore, the copula families chosen for static C-vine copula all have two dependence parameters.*

# Appendix I. Scatter Plot of Block Copula Data

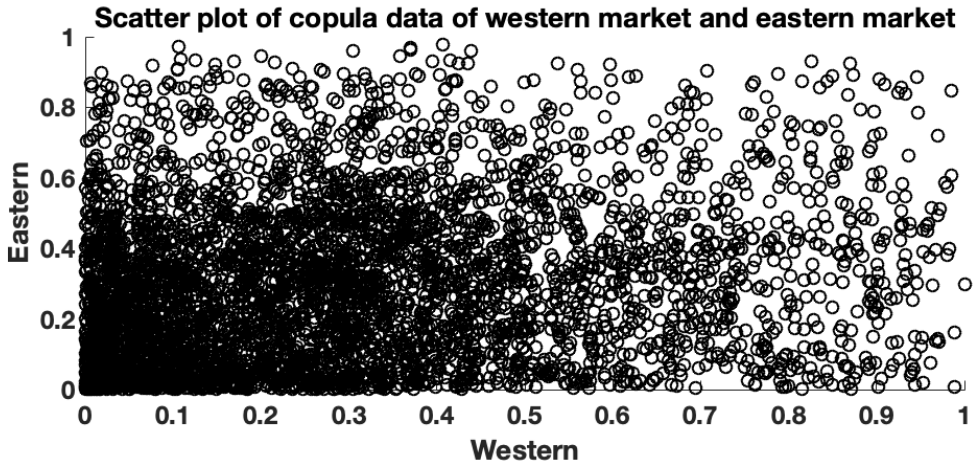



***Figure 9. Copula Data of the Western and Eastern Block***

## Appendix J. Code Usage Description

In this section, we go down to the nitty-gritty about the code usage in this paper. As have been hinted throughout the paper, we make use of two softwares, Matlab(MATLAB (2010)) and R(R Core Team (2013)), for our analyses. On each platform, we were provided with code and packages that would be greatly conducive for our study. Below, we make references to the external help in this regard.

**MATLAB(MATLAB (2010))** We make use of the code package provided by Hafner and Manner (2012). This package is used for the estimation of the SV model and SCAR model specifically. Key functions that we capitalised from this package is as follows.

- Stochastic_Copula_MLE.m: Main function used to estimate the SCAR model.
- SV_MLE_EIS.m: Main function used to estimate the SV model
- LL_Normal_EIS.m: log-likelihood function for the Normal Copula with EIS
- LL_Gumbel_EIS.m: log-likelihood function for the Gumbel Copula with EIS
- LL_Frank_EIS.m: log-likelihood function for the Frank Copula with EIS
- LL_Clayton_EIS.m: log-likelihood function for the Clayton Copula with EIS
- PdfGumbel.m: pdf of the Gumbel copula
- PdfNormal.m: pdf of the Normal copula
- PdfClayton.m: pdf of the Clayton copula
- PdfFrank.m: pdf of the Frank copula

**R(R Core Team (2013))** We export different R packages for our study. This includes:

- **PerformanceAnalytics**: portfolio performance analysis
  Source: https://cran.r-project.org/web/packages/PerformanceAnalytics/index.html
- **CDVine**: all the analyses revolving around Vine copula and the $h$ function.
  Source: https://cran.r-project.org/web/packages/CDVine/index.html
- **rmgarch**: estimation and one-step-ahead forecasting for the DCC GARCH.
  Source: https://cran.r-project.org/web/packages/rmgarch/index.html
- **lpSolverAPI**: solving linear programming problem associated with C-VaR portfolio construction.
  Source: https://cran.r-project.org/web/packages/lpSolveAPI/index.html
- **tmvtnorm**: sampling from the truncated multivariate normal using the GIBBS sampling(Kotecha and Djuric (1999)).
  Source: https://cran.r-project.org/web/packages/tmvtnorm/index.html
- **xts**: convert data into zoo-type time series.
  Source: https://cran.r-project.org/web/packages/xts/index.html
- **mvtnorm**: fit the data into the multivariate normal distribution.
  Source: https://cran.r-project.org/web/packages/mvtnorm/index.html

- **mvnmle**: maximum-likelihood estimation in general.
  Source: https://cran.r-project.org/web/packages/mvnmle/index.html
- **stats**: statistical computations(correlation, covariance, kendall's $\tau$) in general.
  Source: https://cran.r-project.org/web/packages/mvnmle/index.html

In the last two sections, we list the code we have written that makes use of the above-mentioned packages to achieve the goal of our study. Note that the 'LL_Normal_EIS_X.m' function, the adjusted version for the SCARX estimation, bases off of the original version 'LL_Normal_EIS.m'(Hafner and Manner (2012)).

# Appendix K. R Programming Code

***Listing 1.*** **DATA Preparation**

```
##DATA PREPARATION
##Merge and clean the data
range <- "2001/2018"
setxts<-function(df){
  d <- df$Date
  f <- df$Close
  c <- as.character(f)
  p <- as.numeric(c)
  x <- as.xts(p,d)
  x <- rmna(x)
  return(x[range])
}

setxts_alt <- function(df){
  d <- df$Date
  p <- df$Price
  x<-as.xts(p,d)
  x<- rmna(x)
  return(x[range])
}

setxts_nik <- function(df){
  d <- df$Date
  f <- df$Price
  c <- as.character(f)
  p <- as.numeric(gsub(",", "", c))
  x <- as.xts(p,d)
  x <- rmna(x)
  return(x)
}

setxts_gold <- function(df){
  d<- df[,1]
  c <- as.character(df[,2])
  p <- as.numeric(c)
  x <- as.xts(p,d)
  x<- rmna(x)
  return(x[range])
}

rmna <- function(x){
  xn <- subset(x,is.na(x)==FALSE)
  return(xn)
}

p_cac <- setxts(df_cac)
p_dax <- setxts(df_dax)
p_dji <- setxts(df_dji)
p_nas <- setxts(df_nas)
p_sse <- setxts(df_sse)

```

```
p_fx1 <- setxts_alt(df_fx1)
p_fx2 <- setxts_alt(df_fx2)
p_nik <- setxts_nik(df_nikkei)
p_gold <- setxts_gold(df_gold)

A <- merge(p_dax,p_cac)
B <- merge(p_fx1,p_fx2, p_gold)
C <- merge(p_dji,p_nas)
D <- merge(p_sse, p_nik)

df_P <- merge(A,B)
df_P <- merge(df_P,C)
df_P <- merge(df_P,D)

df_P <- df_P["2001-01-02/2018"]
df_P <- na.locf(df_P,fromLast = FALSE)
df_P["2001/2001-01-04","p_nik"]<-df_P["2001-01-05","p_nik"]

matrix.P <- as.matrix(df_P)
n <- length(matrix.P[,1])
matrix.R <- log(matrix.P[2:n,]/matrix.P[1:n-1,]) * 100
df.R <- as.xts(matrix.R)


plot(df.R)
cor(df.R)

# l <- list()
# for (i in 2:9){
#   vec <- as.matrix(b[,i])
#   tree <- missForest(vec,10,100)
#   vec_update <- tree$ximp
#   l[i]<- vec_update
# }

U.wes = U[,c(8,4,5,3,2)]
U.est = U[,c(9,6,7)]
```

***Listing 2.*** **Pipeline**

```
###----------------------This file reads results from and export results to Matlab ---------------------------###


##output: results obtained from bivariate SCARs for a given layer
##path: theta generated from importance samplers
##Calculating the mean of h and pass it to the next iteration

#obtain the hfunction!!!! powerhouse here lol
U.list <- list(marginal = U)
U.adhoc <- read.csv("U.csv",header = FALSE)
path.adhoc <- read.csv("paths.csv",header = FALSE)
getHList(U.adhoc,path.adhoc)


##read in files for estimating scarVineCDF
EPS <- read.csv("EPS.csv")
path_layer_1_est <- read.csv("paths_layer_1_est.csv")
path_layer_2_est <- read.csv("paths_layer_2_est.csv")
path_layer_1_wes <- read.csv("paths_layer_1_wes.csv")
path_layer_2_wes <- read.csv("paths_layer_2_wes.csv")
path_layer_3_wes <- read.csv("paths_layer_3_wes.csv")
path_layer_4_wes <- read.csv("paths_layer_4_wes.csv")

##joining paths together
PATH_est <- data.frame(path_layer_1_est,path_layer_2_est)
PATH_wes <- data.frame(path_layer_1_wes,path_layer_2_wes,path_layer_3_wes,path_layer_4_wes)

##partition the EPS datafrome into two, Asian and Western
EPS_est <- EPS[,c("r_fx2","r_sse","r_nik")]
EPS_wes <- EPS[, c("r_fx1","r_dji","r_nas","r_cac","r_dax")]


##get the CDF of those two vine copulas!!!
F_wes <- scarVineCDF(EPS = EPS_wes,PATHS = PATH_wes)
F_est <- scarVineCDF(EPS = EPS_est, PATHS = PATH_est)
```

```

F_est <- matrix(F_est, nrow = length(F_est), ncol = 1)
F_wes <- matrix(F_wes, nrow = length(F_wes), ncol = 1)
U_ew <- matrix(cbind(F_wes,F_est),nrow = length(F_wes),ncol = 2)
U_ew <- as.data.frame(U_ew)
colnames(U_ew) <- c("c_wes","c_est")
write.csv(U_ew, "U_ew.csv")

#preparing data for
path_est <- read.csv("path_est.csv",header = FALSE)
cdf.est <- data.frame()
for (i in 1:4753){
  path <- path_est[i,]
  u1 <- U.est$V1[i]
  u2 <- U.est$V2[i]
  cdf <- BiCopCDF(u1 = u1,u2 = u2,family = 1,par = path)
  cdf.est <- rbind(cdf.est,cdf)
}
colnames(cdf.est)<- c("cop_sse_nik")

EPS_estSansFx <- EPS_est[,c(3,2)]
cdf.est.alt <- scarVineCDF(EPS = EPS_estSansFx,PATHS = path_est,N = 1000,corr = cor(EPS_estSansFx))

df.interagir <- cbind(cdf.est,returns.cdf)
xts.df.interagir <- xts(df.interagir,order.by = time(xts.r.gold))
riskAversMetric <- rollapply(xts.r.gold, 22, function(x){colMeans(x[x>0,])})
xts.df.interagir <- merge(riskAversMetric,xts.df.interagir,join = "right")
xts.df.interagir <- xts.df.interagir["2001-02/2018"]

estBic <- BiCopEst(u1 = xts.df.interagir$cop_sse_nik,u2 = xts.df.interagir$returns.cdf,family = 4)
CDVineLogLik(merge(xts.df.interagir$cop_sse_nik,xts.df.interagir$returns.cdf),family = 4,par = 1.57906,type = 1)
write.csv(xts.df.interagir,"df_interagir.csv")
```

***Listing 3.*** **Functions for C-VaR portfolio Construction**

```

##Function for predicting volatility(SV model)
getPredVol <- function(par,sigma,N = 22){
  sigma <- as.data.frame(sigma)
  nVar = length(par[1,])
  alpha = par[1,]
  beta = par[2,]
  nu = par[3,]
  mu = alpha/(1-beta)
  h = log(sigma)
  pred.h <- data.frame()
  pred.sig <- data.frame()
  pred.vol <- data.frame()
  for (i in 1:N){
    pred.h <- rbind(pred.h, mu + (beta^i)*(h - mu))
    pred.sig = rbind(pred.sig, sqrt((nu^2)*(1-beta^(2*i))/(1 - beta^2)));
  }
  pred.vol <- exp(pred.h + (pred.sig^2)/2)
  return(colMeans(pred.vol))
}

#Function for conducting out-of-sample forecasting for the dependence parameter of Dynamic C-vine
getPredTheta  <- function(par,theta,N = 22){
  theta <- as.data.frame(theta)
  lam <- 0.5*log((1+theta)/(1-theta))
  alpha = par[1,]
  beta = par[2,]
  nu = par[3,]
  mu = alpha/(1-beta)
  pred.lam <- data.frame()
  pred.sig <- data.frame()
  pred.theta <- data.frame()
  for (i in 1:N){
    pred.lam <- rbind(pred.lam, mu + (beta^i)*(lam - mu))
    pred.sig = rbind(pred.sig, sqrt((nu^2)*(1-beta^(2*i))/(1 - beta^2)));
  }
  add1 = (exp(2*pred.lam) - 1)/(exp(2*pred.lam)+1);
  den = -4*(exp(2*pred.lam)-1)*(exp(2*pred.lam))*(pred.sig^2);
  nom = (exp(2*pred.lam) + 1)^3;
  add2 = den/nom;
  pred.theta = add1 + add2;
```

```
  return (colMeans(pred.theta))
}


#Function for obtaining the porfolio weight(CVaR portfolio!!!!!)
getOptWeight <- function(parSV, parDCV = NULL, sigma, theta = NULL, family = NULL,
                         N = 22, isDynamic = !is.null(parDCV), simNum = 10000, R = 0.003,
                         par2 = NULL,shortLimit = rep(-Inf,4),longLimit = rep(-Inf,4)){
  pred.vol <- getPredVol(parSV,sigma,N)
  pred.theta <- NULL
  if (isDynamic){
    pred.theta <- getPredTheta(par = parDCV,theta = theta, N = N)
    family = rep(1,length(theta))
    par2 <- rep(0,length(family))
  }else{
    pred.theta <- theta
    if (is.null(par2)){
      par2 <- rep(0,length(family))
    }
  }
  simU <- CDVineSim(simNum,par = pred.theta,par2 = par2,family = family, type = 1)
  simR <- simU
  for (i in 1:4){
    u <- simU[,i]
    simR[,i] <- qnorm(u,0,pred.vol[i])
  }
  v <- solveCVaRPortfolio(simR = simR,simNum = simNum,shortLimit = shortLimit,longLimit = longLimit, R = R)
  res <- matrix(v,nrow = 1,dimnames = list(NULL,c("VaR","w_dax","w_dji","w_nas","w_cac")))
  thetaOutput <- matrix(pred.theta,nrow = 1)
  res <- cbind(res,thetaOutput)
  return(res)
}


##-----------------BLOCK-----------------------------------##
#below are the function input for 'apply' function
#(used to conduct Portfolio Rebalancing)
#optimal weight Dynamic C-vine monthly rebalanced
optWDCV <- function(x){
  df1 <- x[,1:6]
  df2 <- x[,7:10]
  res <- getOptWeight(parSV = Param.SV,parDCV = param_DCV,sigma = df2,
                      theta = df1,simNum = 2000,R = 0.01)
  return(res)
}
#Optimal weight Dynamic C-vine weekly rebalanced
optWDCV.weekly <- function(x){
  df1 <- x[,1:6]
  df2 <- x[,7:10]
  res <- getOptWeight(parSV = Param.SV,parDCV = param_DCV,sigma = df2,
                      theta = df1,simNum = 2000,R = 0.04, N = 5)
  return(res)
}
#Optimal weight Static C-vine monthly rebalanced
optWSCV <-function(x){
  res <- getOptWeight(parSV = Param.SV,parDCV = NULL,sigma = x,simNum = 20000,
                      family = selectRes$family,theta = selectRes$par,
                      par2 = selectRes$par2,R = 0.01)
  return(res)
}
#Optimal weight Static C-vine weekly rebalanced
optWSCV.weekly <-function(x){
  res <- getOptWeight(parSV = Param.SV,parDCV = NULL,sigma = x,simNum = 2000,
                      family = selectRes$family,theta = selectRes$par,
                      par2 = selectRes$par2,R = 0.04)
  return(res)
}
#optimal weight Dynamic C-vine risk-controlled
optWDCV.weekly.shortLimited <- function(x){
  limitS <- rep(-2,4)
  limitL <- rep(2,4)
  df1 <- x[,1:6]
  df2 <- x[,7:10]
  res <- getOptWeight(parSV = Param.SV,parDCV = param_DCV,
                      sigma = df2,theta = df1,simNum = 1000,R = -1,
                      N = 5,shortLimit = limitS,longLimit = limitL)
```

```
  return(res)
}
#optimal weight DCC-GARCH
optWDCC <- function(vector.covMat,simNum = 2000, beta = 0.99, R = 0.04){
  limitS <- rep(-2,4)
  limitL <- rep(2,4)
  covMat <- matrix(vector.covMat,nrow = 4)
  simR <- mvrnorm(n = simNum,mu = rep(0,4),Sigma = covMat)
  v <- solveCVaRPortfolio(simR = simR,shortLimit = limitS,longLimit = limitL)
  res <- matrix(v,nrow = 1,dimnames =
                  list(NULL,c("VaR","w_dax","w_dji","w_nas","w_cac")))
  return(res)
}
##-----------------------------------------------------------------------------------------------------------##


#Get portfolio returns given portfolio weight
getPortfolioReturn <- function(weight, R = xts.returns[testRange],
                               f = "weeks", portfolioName = "V1"){
  list.return <- split(R, f = f)
  list.w <- split(weight, f=f)
  N <- length(list.return)
  xts.port <- data.frame()
  for (i in 1:N){
    w = matrix(list.w[[i]],nrow = 4)
    subR <- as.matrix(list.return[[i]])
    r <- subR %*% w
    xts.port <- rbind(xts.port,r)
  }
  xts.port <- xts(xts.port,order.by = time)
  colnames(xts.port) <- portfolioName
  return(xts.port)
}


#Solve the linear programming problem associated with CVaR portfolio
solveCVaRPortfolio <- function(simR,beta = 0.99,simNum = length(simR[,1]),
                               shortLimit = NULL, longLimit = NULL, R){
  m =-colMeans(simR)
  A = matrix(nrow = simNum + 2, ncol = simNum + 5)
  A[1,] <- c(0,t(m),rep(0,simNum))
  A[2,] <- c(0,rep(1,4),rep(0,simNum))
  A[3:(simNum+2),1:5] <- cbind(rep.row(1,n = simNum),simR)
  A[3:(simNum+2),6:(simNum+5)] = diag(simNum)

  B = c(-R,1,rep(0,simNum))
  constranints_direction = c("<=","=",rep(">=",simNum))

  C <- c(1,rep(0,4),rep((1/(simNum*(1-beta))),simNum))

  port.lp <- make.lp(0,simNum+5)
  for (i in 1:(simNum+2)){
    add.constraint(port.lp,xt = A[i,],type = constranints_direction[i],
                   rhs = B[i])
  }
  set.objfn(port.lp,C)
  set.type(port.lp,1,"real")
  set.type(port.lp,c(2,3,4,5),"real")
  set.bounds(port.lp,lower = c(-Inf,shortLimit,rep(0,simNum)),
             upper = c(Inf, longLimit,rep(Inf,simNum)))
  solve(port.lp)
  v <- get.variables(port.lp)
  v <- v[1:5]
  # res <- list(VaR = v[1], weights = matrix(v[2:5],nrow = 1,
  #    dimnames = list(NULL,c("w_dax","w_dji","w_nas","w_cac"))))
  return(v)
}
```

***Listing 4.*** **C-VaR Portfolio Construction**

```
#Portfolio based on static C-vine
order <- c("r_dax","r_dji","r_nas","r_cac")

STD <- read.csv("STD.csv")
Mean.ordered <- Mean[,order]
```

```
STD.ordered <- STD[,order]
CVinwRN <- CDVineSim(N = 1000, family = selectRes$family
                     ,par = selectRes$par,par2 = selectRes$par2,type = 1)

PATHS_DCVP<-read.csv("PATHS_DCVP.csv",header = FALSE)
param_DCV <- read.csv("param_CV.csv")
dfR <- read.csv("dfR.csv",header = TRUE)
dfR$Date <- as.Date(dfR$Date)
Param.SV <- read.csv("PARAM_SV.csv",header = TRUE)
Param.SV <- Param.SV[,order]
param_DCV$Static <- c(PATHS_DCVP[1,6],0,0)
#train test split
xts.returns <- as.xts(dfR[,2:5],dfR$Date)
xts.paths <- as.xts(PATHS_DCVP,dfR$Date)
xts.STD <- as.xts(STD,dfR$Date)
xts.weight.DCV <- xts(rep(NaN,4753),order.by = dfR$Date)
xts.weight.SCV <- xts(rep(NaN,4753),order.by = dfR$Date)
xts.predVol.DCV <- xts(rep(NaN,4753),order.by = dfR$Date)
xts.predVol.SCV <- xts(rep(NaN,4753),order.by = dfR$Date)
xts.predTheta <- xts(rep(NaN,4753),order.by = dfR$Date)
xts.paths.test <- xts.paths["2016/2018"]
xts.paths.train <- xts.paths["2001/2015"]

trainRange = "2001/2015"
testRange = "2016/2018"
#solve for the best weights for static C-vine Model
res.SCV <- selectRes

asset.name <- c("dax","dji","nas","cac")


#predict the volatility of next month

#----------------------rebalance monthly-----------------------------
xts.lag.path <- apply.monthly(xts.paths.test, last)
xts.lag.path["2016-02/2018-12"] <- xts.lag.path["2016-01/2018-11"]
xts.lag.path["2016-01"] <- last(xts.paths["2015-12"])

xts.lag.std <- apply.monthly((xts.STD[testRange]),last)
xts.lag.std["2016-02/2018-12"] <- xts.lag.std["2016-01/2018-11"]
xts.lag.std["2016-01"] <- last(xts.STD["2015-12"])
xts.lag.BIG = merge(xts.lag.path,xts.lag.std)

xts.returns.dcvp = data.frame()
xts.returns.scvp = data.frame()

xts.w.dcvp <- apply.monthly(xts.lag.BIG,optWDCV)
xts.w.scvp <- apply.monthly(xts.lag.std,optWSCV)
xts.w.dcvp <- xts.w.dcvp[,2:5]
xts.predPaths.monthly <- xts.w.dcvp[,6:11]
xts.paths.big.monthly <- na.locf(merge(xts.paths.test,xts.predPaths.monthly),na.rm = FALSE)
xts.w.scvp <- xts.w.scvp[,2:5]
list.R <- split(xts.returns[testRange],f="months")
list.w.dcvp <- split(xts.w.dcvp, f="months")
list.w.scvp <- split(xts.w.scvp, f="months")
N = length(list.R)
for (i in 1:N){
  w.dcvp = matrix(list.w.dcvp[[i]],nrow = 4)
  w.scvp = matrix(list.w.scvp[[i]],nrow = 4)
  subR <- as.matrix(list.R[[i]])
  r.dcvp <- subR %*% w.dcvp
  r.scvp <- subR %*% w.scvp
  xts.returns.dcvp <- rbind(xts.returns.dcvp,r.dcvp)
  xts.returns.scvp <- rbind(xts.returns.scvp,r.scvp)
}

time = time(xts.returns[testRange])
xts.returns.dcvp <- apply.monthly(0.01*xts(xts.returns.dcvp,order.by = time(xts.returns[testRange])),Return.cumulative)
xts.returns.scvp <- apply.monthly(0.01*xts(xts.returns.scvp,order.by = time(xts.returns[testRange])),Return.cumulative)
xts.returns.naive <- apply.monthly(0.01*xts(rowMeans(xts.returns[testRange]),order.by = time(xts.returns[testRange])),
     Return.cumulative)
colnames(xts.returns.dcvp) <- c("dynamicC-vinePortfolio_MonthlyRebalanced")
colnames(xts.returns.scvp) <- c("staticC-vinePortfolio_MonthlyRebalanced")
colnames(xts.returns.naive) <- c("1/N_Portfolio")

#----------------------rebalance Weekly-----------------------------
```

```
xts.lag.path.weekly = apply.weekly(xts.paths.test,last)
xts.lag.path.weekly <- lag.xts(xts.lag.path.weekly)
xts.lag.path.weekly[1,] <- xts.lag.path.weekly[2,]
xts.lag.std.weekly <- lag.xts(apply.weekly(xts.STD[testRange],last))
xts.lag.std.weekly[1,] <- xts.lag.std.weekly[2,]
xts.lag.BIG.weekly = merge(xts.lag.path.weekly,xts.lag.std.weekly)

xts.w.dcvp.weekly <- apply.weekly(xts.lag.BIG.weekly,optWDCV.weekly)
xts.w.scvp.weekly <- apply.weekly(xts.lag.std.weekly,optWSCV.weekly)
xts.w.dcvp.weekly.limited <- apply.weekly(xts.lag.BIG.weekly,optWDCV.weekly.shortLimited)
xts.w.dcvp.weekly <- xts.w.dcvp.weekly[,2:5]
xts.w.scvp.weekly <- xts.w.scvp.weekly[,2:5]
xts.w.dcvp.weekly.limited <- xts.w.dcvp.weekly.limited[,2:5]
xts.predPaths.weekly <- xts.w.dcvp.weekly.limited[,6:11]
xts.paths.big.weekly <- na.locf(merge(xts.paths.test,xts.predPaths.weekly),na.rm = FALSE)
list.R.weekly <- split(xts.returns[testRange],f="weeks")
list.w.dcvp.weekly <- split(xts.w.dcvp.weekly, f="weeks")
list.w.scvp.weekly <- split(xts.w.scvp.weekly, f="weeks")
list.w.dcvp.weekly.limited <- split(xts.w.dcvp.weekly.limited, f="weeks")

N.weekly = length(list.R.weekly)
xts.returns.dcvp.weekly = data.frame()
xts.returns.scvp.weekly = data.frame()

xts.returns.dcvp.weekly.limited = getPortfolioReturn(xts.w.dcvp.weekly.limited,portfolioName = "dynamicC-vinePortfolio_
    WeeklyRebalanced_riskControl")

# for (i in 1:N.weekly){
#   w.dcvp.weekly = matrix(list.w.dcvp.weekly[[i]],nrow = 4)
#   w.scvp.weekly = matrix(list.w.scvp.weekly[[i]],nrow = 4)
#   subR.weekly <- as.matrix(list.R.weekly[[i]])
#   r.dcvp.weekly <- subR.weekly %*% w.dcvp.weekly
#   r.scvp.weekly <- subR.weekly %*% w.scvp.weekly
#   xts.returns.dcvp.weekly <- rbind(xts.returns.dcvp.weekly,r.dcvp.weekly)
#   xts.returns.scvp.weekly <- rbind(xts.returns.scvp.weekly,r.scvp.weekly)
#   xts.returns.dcvp.weekly.limited <- rbind(xts.returns.dcvp.weekly.limited,r.scvp.weekly)
# }

xts.returns.dcvp.weekly <- apply.monthly(0.01*xts(xts.returns.dcvp.weekly,order.by = time(xts.returns[testRange])),Return.
    cumulative)
xts.returns.scvp.weekly <- apply.monthly(0.01*xts(xts.returns.scvp.weekly,order.by = time(xts.returns[testRange])),Return.
    cumulative)
xts.returns.dcvp.weekly.limited <- apply.monthly(0.01*xts.returns.dcvp.weekly.limited,Return.cumulative)
colnames(xts.returns.dcvp.weekly) <- c("dynamicC-vinePortfolio_WeeklyRebalanced")
colnames(xts.returns.scvp.weekly) <- c("staticC-vinePortfolio_WeeklyRebalanced")
charts.PerformanceSummary(xts.returns.dcvp.weekly)
charts.PerformanceSummary(xts.returns.scvp.weekly)

#---------Monthly returns of the originals-------------
xts.returns.test <- xts.returns[testRange]
xts.returns.test <- apply.monthly(0.01*xts.returns.test,Return.cumulative)

#-------------------Benchmark DCC model----------------------
uspec <- ugarchspec(mean.model = list(armaOrder = c(0,0),include.mean = TRUE))
mspec <- multispec(replicate(4,uspec))
spec <- dccspec(mspec,model = "DCC",distribution = "mvnorm",VAR = FALSE)
mdcc <- dccfit(spec,xts.returns,out.sample = 841)
mdcc <- dccfit(spec,xts.returns)
mdcc.forecast <- dccforecast(mdcc,n.roll = 841)
mdcc.forecast.fit <- fitted(mdcc.forecast)
goodStuff <- mdcc.forecast@mforecast$H
# covMatDCC <- mdcc.forecast@mforecast$H
# list.covMatDCC <- list()
# for (i in 1:4753){
#   cov = covMatDCC[,,i]
#   list.covMatDCC[[i]]<-cov
# }
# list.covMatDCC.test <- list.covMatDCC[(4753-841) : 4752]
df.vectorize.covMatDCC <- data.frame()
for (i in 1:841){
  cov <- goodStuff[[i]]
  df.vectorize.covMatDCC <- rbind(df.vectorize.covMatDCC,matrix(cov,nrow = 1))
}
xts.vectorise.covMat <- xts(df.vectorize.covMatDCC,order.by = time)
#Montly rebalanced
xts.vectorise.predCov.monthly <- apply.monthly(xts.vectorise.covMat,first)
```

```
xts.w.DCC.monthly <- apply.monthly(xts.vectorise.predCov.monthly, optWDCC)
xts.w.DCC.monthly<-xts.w.DCC.monthly[,2:5]

#Weekly rebalanced
xts.vectorise.predCov.weekly <- apply.weekly(xts.vectorise.covMat,first)
xts.w.DCC.weekly <- apply.weekly(xts.vectorise.predCov.weekly, optWDCC)
xts.w.DCC.weekly <- xts.w.DCC.weekly[,2:5]
xts.returns.DCC <- getPortfolioReturn(xts.w.DCC.weekly, portfolioName = c("DCCPortfolio_WeeklyRebalanced"))
xts.returns.DCC <- apply.monthly(0.01*xts.returns.DCC,Return.cumulative)
chart.RelativePerformance(xts.returns.DCC,xts.returns.dcvp.weekly)


xts.returns.dcvp.best <- apply.monthly(0.01*xts.returns.dcvp.best,Return.cumulative)
##performance analysis
charts.PerformanceSummary(merge(xts.returns.dcvp.weekly.limited,xts.returns.naive))

attach(mtcars)
par(mfrow=c(1,1))
charts.PerformanceSummary(merge(xts.returns.DCC,xts.returns.naive),methods = c("HistoricalES"))
charts.PerformanceSummary(merge(xts.returns.dcvp.weekly.limited,xts.returns.naive))

colnames(xts.returns.dcvp.weekly) <- c("DynamicC-Vine")
colnames(xts.returns.dcvp.weekly.limited) <- c("DynamicC-Vine_RiskControlled")
colnames(xts.returns.DCC) <- c("DCC")
colnames(xts.returns.scvp.weekly) <- c("StaticC-Vine")
charts.PerformanceSummary(merge(xts.returns.dcvp.weekly,xts.returns.naive),methods = c("HistoricalES"))
charts.PerformanceSummary(merge(xts.returns.dcvp.weekly.limited,xts.returns.naive),methods = c("HistoricalES"))
charts.PerformanceSummary(merge(xts.returns.scvp.weekly,xts.returns.naive),methods = c("HistoricalES"))
charts.PerformanceSummary(merge(xts.returns.DCC,xts.returns.naive),methods = c("HistoricalES"))

SharpeRatio(xts.returns.dcvp.weekly, FUN = "StdDev")
SharpeRatio(xts.returns.dcvp.weekly.limited,FUN = "StdDev")
SharpeRatio(xts.returns.scvp.weekly,FUN = "StdDev")
SharpeRatio(xts.returns.DCC,FUN = "StdDev")
SharpeRatio(xts.returns.naive,FUN = "StdDev")

SharpeRatio(xts.returns.DCC,FUN = "StdDev")
SortinoRatio(xts.returns.dcvp.weekly, FUN = "StdDev")
SortinoRatio(xts.returns.dcvp.weekly.limited,FUN = "StdDev")
SortinoRatio(xts.returns.scvp.weekly,FUN = "StdDev")
SortinoRatio(xts.returns.DCC,FUN = "StdDev")
SortinoRatio(xts.returns.naive,FUN = "StdDev")

OmegaSharpeRatio(xts.returns.dcvp.weekly, FUN = "StdDev")
OmegaSharpeRatio(xts.returns.dcvp.weekly.limited,FUN = "StdDev")
OmegaSharpeRatio(xts.returns.scvp.weekly,FUN = "StdDev")
OmegaSharpeRatio(xts.returns.DCC,FUN = "StdDev")
OmegaSharpeRatio(xts.returns.naive,FUN = "StdDev")
```

***Listing 5.*** **Static C-vine Order Selection**

```
df.wes = U.wes[,c('r_dji','r_nas','r_cac','r_dax')]

#The method is based on Czado et al.(2012)
#estimate pair-wise Kendall's tau
mat.cor <- cor(df.wes,method = "kendall")
S <- rowSums(mat.cor)
index_1 <- which(S == max(S))

pivot1 <- df.wes[,index_1]
df2 <- df.wes[,-index_1]
df2.cond <- df2
for (i in 1:3){
  newcommer <- df2[,i]
  par <- BiCopEst(pivot1,newcommer,family = 1)$par
  H1 <- BiCopHfunc(pivot1,newcommer,family = 1,par = par)
  df2.cond[,i] <- H1$hfunc1
}
mat.cor1 <- cor(df2.cond,method = "kendall")
S1 <- rowSums(mat.cor1)
index_2 <- which(S1 == max(S1))

df3 <- df2.cond[,-index_2]
pivot2 <- df2.cond[,index_2]
df3.cond<-df3
```

```
for (i in 1:2){
  newcommer <- df3[,i]
  par <- BiCopEst(pivot2,newcommer,family = 1)$par
  H2 <- BiCopHfunc(pivot2,newcommer,family = 1,par = par)
  df3.cond[,i] <- H2$hfunc1
}

rmdn <- colnames(df3)
order = cbind(names(index_1),names(index_2),rmdn[1],rmdn[2])
df.wes.ordered <- U.wes[,order]


#here we construct a static C-vine model which will later be used for conparison
selectRes <- CDVineCopSelect(df.wes.ordered,type = 1)
BiCopMetaContour(u1 = df.wes.ordered[,1],u2 = df.wes.ordered[,2],family = 7,par = 0.3427,par2 = 1.4453)
write.csv(df.wes.ordered,"U_wes_ordered.csv")
```

***Listing 6.*** **Functions for C-Vine PIT**

```
#Obtain cdf for dynamic SCAR
#Based on Algorithm 2
scarVineCDF <- function(EPS, PATHS, IS = TRUE, N = 100, is.fixed.path = FALSE, fix.val = NULL,corr = NULL, lb = -10){
  nVar = length(EPS[1,])
  nr <- length(EPS[,1])
  if (is.null(corr)){
  fit <- mlest(EPS)
  sigmahat <- fit$sigmahat
  corr <- cov2cor(sigmahat)
  }
  CDF <- array(dim = nr)
  for (i in 1:nr){
    # print(i)
    if (!is.fixed.path){
      path <- as.vector(as.matrix(PATHS[i,]))
      path.rep <- rep.row(path,N)
    }else{
      path.rep <- rep.row(fix.val,N)
    }
    eps <- as.vector(as.matrix(EPS[i,]))
    if (IS){
      rnd <- rtmvnorm(N,sigma = corr,upper = eps,algorithm = "gibbs")
      d <- dtmvnorm(rnd,sigma = corr,upper = eps)
    }else{
      rnd <- draw.d.variate.uniform(no.row = N,d = nVar, cov.mat = corr)
      rep.lb <- rep(lb,nVar)
      f1 <- rep.row(eps - rep.lb,N)
      rnd <- f1*rnd + rep.lb
      d <- 1
    }
    f <- scarVinePDF(rnd,path.rep)
    cdf <- mean(f/d)
    cdf <- min(cdf,1)
    CDF[i] <- cdf
    # print(cdf)
  }
  return(CDF)
}


#obtain pdf of the dynamic C-vine density
#pdf is obtained through an iterative approach based on the h(‘) function
scarVinePDF <- function(EPS,PATHS, family = 1, is.fixed.path = FALSE, fix.val = NULL){
  nr <- length(EPS[,1])
  nVar <- length(EPS[1,])
  nLayer <-  nVar - 1
  tC <- (nVar+1)*nVar/2
  MM <- matrix(nrow = 1,ncol = tC)
  FF <- matrix(nrow = nr, ncol = 1)
  if (is.fixed.path){
    PATHS <- rep.row(fix.val,nr)
  }
  for (i in 1:nr){
    f<-1
    eps = as.matrix(EPS[i,])
    path <- PATHS[i,]
```

```
    u <- pnorm(eps)
    MM[,1:nVar] <- as.matrix(u)
    for (nn in 1: nVar){
      f <- f*dnorm(EPS[i,nn])
    }
    count1 = 0
    count2 = 0
    for (j in 1:nLayer){
      n1 <- nLayer - j + 2
      n2 <- n1 -1
      H <- matrix(MM[(count1+1):(count1+n1)],nrow = 1,ncol = n1)
      param <- matrix(path[(count2+1) : (count2+n2)],nrow = 1,ncol = n2)
      pairs <- getPairs(H)
      nPairs <- length(pairs[1,])/2
      H_new <- matrix(nrow = 1,ncol = nPairs)
      for (k in 1:nPairs){
        h1 <- pairs[,2*k-1]
        h2 <- pairs[,2*k]
        par <- param[,k]
        h <- BiCopHfunc(h1,h2,family = family,par = par)$hfunc1
        f <- f * BiCopPDF(h1,h2,family = family, par = par)
        H_new[,k] <- h
      }
      count1 = count1 + n1
      count2 = count2 + n2
      MM[,(count1 + 1) : (count1 + n2)] <- H_new
    }
    FF[i,] <- f
  }
  return(FF)
}

#Auxiliary functions
getPairs <- function(X){
  nc <- length(X[1,])
  nr <- length(X[,1])
  pairs <- matrix(nrow = nr,ncol = 2*(nc-1))
  for (i in 1:(nc-1)){
    pairs[,(2*i -1):(2*i)] <- cbind(X[,1],X[,i+1])
  }
  return(pairs)
}
rep.row<-function(x,n){
  matrix(rep(x,each=n),nrow=n)
}


#Study variance of the cdf estimator
#Long running time
EPS.ordered <- EPS[,order]
returns.cdf <- scarVineCDF(EPS = EPS.ordered,N = 5000, PATHS = PATHS_DCVP,corr = cor(xts.returns))
valL <- c(50,seq(100,1000, by = 100),1500,2000,3000,5000,10000,20000,50000)
arr.std = array()
count = 0
for (i in valL){
  count = count + 1
  rep.N = 1000
  vessel <- scarVineCDF(N = i,EPS = matrix(rep.row(c(0.1,0.1,0.1,0.1),rep.N),nrow = rep.N),PATHS = PATHS_DCVP,corr = cor(
      xts.returns))
  arr.std[count] <- std(vessel)
}
```

***Listing 7.*** $h(\cdot;\theta_t)$ **in action**

```
##Crucial Function for Sequential Estimation of Dynamic C-vine
##Make use of CDVine package to perform h() function(see (33) in our paper)
##Then save the results to a certain directory
##They will then be fed to Matlab
getHList <- function(U, path, name = NULL){
  N = 200
  TT = length(U[,1])
  ncol <- length(U[1,])
  nPair = 0
  H.list = matrix(nrow = TT, ncol = ncol/2)
  for (pair in seq(1,ncol-1,2)){
```

```
    H = array(dim = TT)
    nPair = nPair + 1
    U.pair <- U[,pair:(pair+1)]
    path.pair <- path[,(1+(nPair-1)*N):(N*nPair)]
    # path.pair <- path[,pair:(pair+1)]
    nPaths = length(path.pair[1,])
    for (i in 1:TT){
      p <- array(path.pair[i,])
      u <- U.pair[i,1]
      v <- U.pair[i,2]
      h = 0
      for (j in 1:nPaths){
        par1 = p[[j]]
        h = h + BiCopHfunc(u,v,family = 1,par = par1,par2 = 0)$hfunc1
      }
      H[i] <- h/nPaths
    }
    # H.list[paste("pair",nPair)] <- H
    H.list[,nPair] <- H
  }
  if (is.null(name)){
    write.csv(H.list,"H.csv")
    return(H.list)
    }else{
    colnames(H.list)<-name
    write.csv(H.list,"H.csv")
    return(H.list)
  }
}
```

***Listing 8.*** *C-vine In-sample Validation*

```
##In-sample Validation for C-vine models

U.ordered <- read.csv("Uordered.csv",header = TRUE)

ll.scv<-CDVineLogLik(U.ordered,family = selectRes$family,par = selectRes$par,par2 = selectRes$par2,type = 1)
bic.scv <- CDVineBIC(U.ordered,family = selectRes$family,par = selectRes$par,par2 = selectRes$par2,type = 1)
aic.scv <- CDVineAIC(U.ordered,family = selectRes$family,par = selectRes$par,par2 = selectRes$par2,type = 1)
ll.dcv <- array()
for (i in 1:4753){
  path <- PATHS_DCVP[i,]
  u <- U.ordered[i,]
  ll <- CDVineLogLik(u,family = rep(1,6),par = path, type = 1)
  ll.dcv[i] <- ll$loglik
}
ll.dcv <- sum(ll.dcv)

aic.dcv <- 2*28 - 2 * ll.dcv
aic.scv <- 2*18 - 2* ll.scv

bic.dcv <- log(4753) - 2 * ll.dcv
bic.scv <- log(4753) - 2 * ll.scv
```

***Listing 9.*** *SCAR estimation R version*

```
##Reference to Hafner et al.(2012)
##The computational time is so long that we did not implement
##SCAR estimation using this function
##We use the Matlab package from Hafner et al.(2019) instead


path <<- data.frame()
LL <- function(u,v,ga,de,nu,N,fam, par2 = 0){
  ## choice of copulas:
  # 1 = Gumbel Copula
  # 2 = Clayton
  # 3 = Normal
  # 4 = Frank
  # 5 = Rotated Gumbel
  # 6 = Rotated Clayton
  # so this is not valid anymore
  iterations <- 5
```

```
  seed <- 100
  ga <- ga
  de <- de
  nu <- nu
  print(list(ga = ga, de = de, nu = nu))
  T = length(u)

  ap = matrix(nrow =  T, ncol = 1)
  q = matrix(nrow = T, ncol = 1)
  av = matrix(nrow = T, ncol = 1)


  if (abs(de)>0.99 | nu < 0.001 | nu > 1){
    f = 10000;
    return(f)
  }else{
    la = matrix(nrow = T+1, ncol = N)
    set.seed(seed = seed)
    crn <- matrix(rnorm((T+1)*N),nrow = T+1, ncol = N)
    # crn <- as.matrix(RN)
    eps = nu*crn
    la[1,] = rep(ga/(1-de),N)

    for (j in 2:(T+1)){
      la[j,] <- rep(ga,N) + de*la[j-1,] + eps[j,]
    }

    for (jj in 1 : iterations){
      lnx = matrix(0, nrow = N,ncol = 1)
      a1 = matrix(nrow = T,ncol = 1)
      a2 <- matrix(nrow = T, ncol = 1)
      la1 <- distort(la,family = fam)

      for (t in T:1){
        ones <- matrix(rep(1,N),nrow = N,ncol = 1)
        c2 <- matrix((la[t+1,]), nrow = N, ncol = 1)
        c3 <- matrix((la[t+1,]^2), nrow = N, ncol = 1)
        x <- cbind(ones,c2,c3)
        uu = u[t]
        vv = v[t]
        l = la1[t+1,]
        y <- logd(u = uu, v = vv, lambda = l ,family = fam, iftranspose = FALSE)
        y <- y + lnx
        est <- solve(t(x)%*%x)%*%t(x)%*%y
        a1[t] <- est[2]
        a2[t] <- est[3]
        E1 = ga + de*la[t,]
        V2 = (nu^2)/(1-2*(nu^2)*est[3])
        E2 = V2 * ((E1/(nu^2))+ est[2])

        lnx = matrix(0.5*(((E2^2)/V2)-((E1^2)/(nu^2))),nrow = N, ncol = 1)

        ap[t]=V2*((ga/(nu^2))+est[2])
        q[t]=(V2/(nu^2))*de
        av[t]=V2
      }

      for (j in 2:(T+1)){
        la[j,] = ap[j-1] + q[j-1]*la[j-1,] + sqrt(av[j-1])*crn[j,1:N]
      }

      # if (mean(mean(la,na.rm = TRUE))>0){
      #   f = 10000
      #   return(f)
      # }
    }
  }

  ones <- matrix(1,nrow = 1,ncol = N)
  av.mat <- av%*%ones
  mu1s <- (la[2:(T+1),] - sqrt(av.mat)*crn[2:(T+1),])^2
  mu0s <- (ga + de*la[1:T,])^2
  lnx <- 0.5*((mu1s/(av.mat)) - (mu0s/nu^2))

  CL = matrix(0,nrow = T, ncol = N)
  la1 = distort(lambda = la, fam = fam)
```

```
94
95    for (i in 1:N){
96      CL1 <- logd(u,v,family = fam, lambda = la1[2:(T+1),i])
97      CL2 <- (a1) * la[2:(T+1),i]
98      CL3 <- (a2) * la[2:(T+1),i]^2
99      CL4 <- lnx[,i]
100     CL5 <- matrix(0.5*log(nu^2),nrow = T,ncol = 1)
101     CL6 <- 0.5*log(av)
102     CL[,i] <- CL1 - CL2 - CL3 + CL4 - CL5 + CL6
103   }
104   scal1 = mean(CL)
105   scal2 = mean(scal1)
106   lngs = -scal2 + CL
107   tg1 = log(mean(exp(colSums(lngs))))
108   f = T*scal2 + tg1
109
110   f = -f
111
112   path <<- la1[2:(T+1),]
113   print(f)
114   return(f)
115
116 }
117
118 distort <- function(lambda,family){
119
120     if (family==1) distort<- (exp(2*lambda) - 1) / (exp(2*lambda) + 1) #normal copula
121     else if (family == 2) distort <- (exp(2*lambda) - 1) / (exp(2*lambda) + 1) #t copula
122     else if (family == 3) distort <- exp(lambda)                    #Clayton copula
123     else if (family == 4) distort<- exp(lambda)+1            #Gumbel Copula
124     else if (family ==5)  distort <- lambda    #Frank
125     else if (family == 13) distort = exp(lambda)
126     else if (family == 14) distort = exp(lambda)+1
127     return(distort)
128     }
129
130 logd <- function(u,v,lambda,family, par2 = NULL, iftranspose = FALSE){
131   logd <- matrix(nrow = length(u),ncol = 1)
132   if (length(u) == length(lambda)){
133     for (k in 1:length(u)){
134       logd[k,1]<- BiCopPDF(u[k],v[k],family = family, par = lambda[k], par2 = par2)
135     }
136     return(log(logd))
137   }else{
138     logd <- sapply(lambda, function(x) BiCopPDF(u,v,family = family, par = x, par2 = par2))
139     logd <- log(logd)
140     if (iftranspose){
141     logd<- t(logd)
142     }
143     }
144   return(logd)
145 }
```

# Appendix L. Matlab Programming Code

***Listing 10.*** *Transformation into Copula DATA using SV models*

```
1  europe = ["r_dax","r_cac"];
2  % others = ["r_fx1","r_fx2","r_gold"];
3  us = ["r_dji","r_nas"];
4  % asia = ["r_sse","r_nik"];
5  % varlist = [europe,us,asia,others];
6
7  varlist = [us,europe]
8
9  N = length(varlist);
10 model = 1;
11 U = NaN(size(returns));
12 EPS = NaN(size(returns));
13 RES = cell(size(returns));
14 STD = NaN(size(returns));
```

```
count = 0;
PARAM_SV = NaN(3,N);
logl_SV = NaN(N,1);
for i = varlist
    count = count + 1;
    r = returns{:,i};
    res = SV_MLE_EIS(r,model);
    std = sqrt(res.path);
    STD(:,count) = std;
    eps = r./std;
    u = normcdf(eps);
    U(:,count) = u;
    EPS(:,count) = eps;
    PARAM_SV(:,count) = res.theta;
    logl_SV(count) = res.logl;
end

% U(:,10) = [];
% EPS(:,10) = [];
% U = array2table(U);
% U.Properties.VariableNames = varlist;
% U.date = date;
% EPS.date = date;
STD = array2table(STD(:,1:4));
STD.Properties.VariableNames = varlist;
Mean = array2table(mean(returns{:,varlist}));
Mean.Properties.VariableNames = varlist;
EPS = array2table(EPS);
EPS.Properties.VariableNames = varlist;
PARAM_SV = array2table(PARAM_SV);
PARAM_SV.Properties.VariableNames = varlist;
writetable(PARAM_SV,"PARAM_SV.csv");


plot(EPS)

wes = ["r_fx1","r_dji","r_nas","r_cac","r_dax"];
est = ["r_fx2","r_sse","r_nik"];
```

***Listing 11.*** *SCAR Estimation and Out-of-Sample Forecasting*

```
% %train test split
% n = numel(dji);
% trainRange = 1:(n-250);
% testRange = (n-250+1):n;
% r1 = dji(trainRange);
% r2 = nas(trainRange);
% rt1 = dji(testRange);
% rt2 = nas(testRange);
%
% model = 1;
%
% res_dji = SV_MLE_EIS(r1,model);
% res_nas = SV_MLE_EIS(r2,model);
%
% sigma_dji = sqrt(res_dji.path);
% sigma_nas = sqrt(res_nas.path);
%
figure(1);
subplot(2,1,1)
plot(bDates,sigma_dji,'k','LineWidth',2)
xlabel("year")
ylabel("volatility")
title("volatility DJ")
set(gca,"FontSize",12,"FontWeight",'bold')
subplot(2,1,2)
plot(bDates,sigma_nas,'k','LineWidth',2)
xlabel("year")
ylabel("volatility")
title("volatility NQ")
set(gca,"FontSize",12,"FontWeight",'bold')
%
% %probability integral transform
% eps_1 = r1./sigma_dji;
% eps_2 = r2./sigma_nas;
```

```
% u_1 = normcdf(eps_1);
% u_2 = normcdf(eps_2);

figure()
scatter(u_1,u_2,'k')
xlabel("DJ")
ylabel('NQ')
title("scatter plot of copula data")
set(gca,"FontSize",12,"FontWeight",'bold')
copList = [3,5,4,2];

% copList = 3;
matlogl = NaN(numel(copList),1);
matstd = NaN(numel(copList),3);
matpara = NaN(numel(copList),3);
paths = NaN(numel(u_1),numel(copList));
matpvars = NaN(1,numel(copList));
matloglres = NaN(numel(copList),1);
names = [];
count = 0;
for i = copList
    count = count + 1;
    res = Stochastic_Copula_MLE(u_1,u_2,i);
    matstd(count,:) = res.stderr;
    matpara(count,:) = res.theta;
    matlogl(count) = abs(res.logl);
    paths(:,count) = mean(res.path,2);
    matpvars(count) = res.LRpval;
    matloglres(count) = abs(res.loglres);
    name = [name,res.model];
end
paths = array2table(paths);
paths.Properties.VariableNames = ["Normal","rGumbel", "Frank", "Clayton"];

paths_sm = smoothdata(paths);
paths_sm.Properties.VariableNames = ["Normal","rGumbel", "Frank", "Clayton"];
paths_sm.date = bDates;
figure(1)
for i = 1:4
    subplot(2,2,i)
    p = paths_sm{:,i};
    nom = paths_sm.Properties.VariableNames{i};
    plot(paths_sm.date,p,'k',"LineWidth",2)
    xlabel('year');
    set(gca,'FontSize',12, 'FontWeight','Bold')
    title(nom)
end

%out-of-sample forecasting

disp("phase 1 done");
%obtain the 'true' paths
res_dji1 = SV_MLE_EIS(rt1,model);
res_nas1 = SV_MLE_EIS(rt2,model);

sigma_dji1 = sqrt(res_dji1.path);
sigma_nas1 = sqrt(res_nas1.path);

eps_t1 = rt1./sigma_dji1;
eps_t2 = rt2./sigma_nas1;
u_t1 = normcdf(eps_t1);
u_t2 = normcdf(eps_t2);
paths_t = NaN(numel(testRange),numel(copList));
count = 0;
for i = copList
    count = count + 1;
    res_t = Stochastic_Copula_MLE(u_t1,u_t2,i);
    paths_t(:,count) = res_t.path;
end
paths_t = array2table(paths_t);
paths_t.Properties.VariableNames = ...
    ["Normal","rGumbel", "Frank", "Clayton"];
disp("phase 2 done")

nCop = numel(copList);
nTest = numel(testRange);
```

```
theta0 = paths{end,:};
lamt0 = NaN(1,nCop);
lamt0(1) = 0.5*log((1+theta0(1))/(1-theta0(1)));
lamt0(2) = log(theta0(2)-1);
lamt0(3) = theta0(3);
lamt0(4) = log(theta0(4));
mu = matpara(:,1)./(1-matpara(:,2));
nu = matpara(:,3);
beta = matpara(:,2);
lamt = NaN(nTest,nCop);
sigt = NaN(nTest,nCop);
for i = 1:numel(testRange)
    lamt(i,:) = mu' + (beta'.^(i)).*(lamt0 - mu');
    sigt(i,:) = sqrt((nu.^2).*(1-beta.^(2*i))./(1 - beta.^2));
end

thetat = NaN(nTest,nCop);
upper = NaN(nTest,nCop);
lower = NaN(nTest,nCop);
for i = 1:4
    lam = lamt(:,i);
    sig = sigt(:,i);
    if i == 2
        thetat(:,i) = exp(lam + (sig.^2)/2) + 1;
        upper(:,i) = logninv(0.975,lam,sig) + 1;
        lower(:,i) = logninv(0.025,lam,sig) + 1;
    elseif i == 4
        thetat(:,i) = exp(lam + (sig.^2)/2);
        upper(:,i) = logninv(0.975,lam,sig);
        lower(:,i) = logninv(0.025,lam,sig);
    elseif i == 1
        add1 = (exp(2*lam) - 1)./(exp(2*lam)+1);
        den = -4*(exp(2*lam)-1).*(exp(2*lam)).*(sig.^2);
        nom = (exp(2*lam) + 1).^3;
        add2 = den./nom;
        thetat(:,i) = add1 + add2;
        nql = norminv(0.975,lam,sig);
        nqu = norminv(0.025,lam,sig);
        upper(:,i) = (exp(2*nql)-1)./(exp(2*nql)+1);
        lower(:,i) = (exp(2*nqu)-1)./(exp(2*nqu)+1);
    else
        thetat(:,i) = lam;
%         upper(:,i) = norminv(0.975,lam,sig);
%         lower(:,i) = norminv(0.025,lam,sig);
        upper(:,i) = lam + 1.96*sig;
        lower(:,i) = lam - 1.96*sig;
    end
end

figure;
for i = 1:4
    switch i
        case 1
            titre = "Normal";
        case 2
            titre = "rGumbel";
        case 3
            titre = "Frank";
        case 4
            titre = "Clayton";
    end
    subplot(2,2,i)
    plot(smoothdata(paths_t{:,i}),'k','LineWidth',2,...
        'DisplayName',"Estimated \theta");
    hold on
    plot(thetat(:,i),'Color',[0 0 0.3],'LineStyle','--','LineWidth',2,...
        'DisplayName',"r-step ahead \theta");
    hold on
    plot(upper(:,i),'Color',[0.3 0 0],'LineStyle',':','LineWidth',2,...
        'DisplayName',"Upperbound");
    hold on
    plot(lower(:,i),'Color',[0 0.3 0],'LineStyle',':','LineWidth',2,...
        'DisplayName','Lowerbound');
    xlabel("r-step")
    ylabel("dependence param")
    title(titre)
```

```
    legend()
    set(gca,'FontSize',12,'FontWeigh','bold')
    hold off
end
```

***Listing 12.*** *Estimation of Dynamic C-vine Copula*

```
%Sequential estimation of Dynamic C-vine copula
%After executing each block, switch to R and use 'pipeline.r'
%to obtain pseudo-observations for the next block
%(i.e. to perform h(') function

PATH_wes = table();
N = 100;

copulas_layer_1_wes = ["fx1_dji","fx2_nas","fx1_cac","fx1_dax"];
U_paired_wes = getPairedArgument(U,wes);
[res_layer_1_wes, U_layer_1_wes,paths_layer_1_wes] = biSCAR(U_paired_wes);
PATH_wes.layer_1 = getAvgPath(paths_layer_1_wes,N,copulas_layer_1_wes);

%moving to the next layer
copulas_layer_2_wes = ["dji_nasCfx1","dji_cacCfx1","dji_daxCfx1"];
H = csvread("H.csv", 1, 1);
H = array2table(H);
H_paired_1_wes = getPairedArgument(H,[]);
[res_layer_2_wes, U_layer_2_wes,paths_layer_2_wes] = biSCAR(H_paired_1_wes);
PATH_wes.layer_2 = getAvgPath(paths_layer_2_wes,N,copulas_layer_2_wes);

%moving to the third layer
copulas_layer_3_wes = ["nas_cacCfx1_dji","nas_daxCfx1_dji"]
H = csvread("H.csv", 1, 1);
H = array2table(H);
H_paired_2_wes = getPairedArgument(H,[]);
[res_layer_3_wes, U_layer_3_wes,paths_layer_3_wes] = biSCAR(H_paired_2_wes);
PATH_wes.layer_3 = getAvgPath(paths_layer_3_wes,N,copulas_layer_3_wes);

%moving to the last layer
copulas_layer_4_wes = "cac_daxCfx1_dji_nas"
H = csvread("H.csv", 1, 1);
H = array2table(H);
H_paired_3_wes = getPairedArgument(H,[]);
[res_layer_4_wes, U_layer_4_wes,paths_layer_4_wes] = biSCAR(H_paired_3_wes);
PATH_wes.layer_4 = getAvgPath(paths_layer_4_wes,N,copulas_layer_4_wes);
```

***Listing 13.*** *SCAR estimation(Eastern Block)*

```
%Estimation the Eastern Market Block
%using SCAR model
varlist = ["r_sse","r_nik"];

U_est = U(:,1:2);
csvwrite("U_est.csv",U_est);

res_est = Stochastic_Copula_MLE(U_est(:,1),U_est(:,2),3);

df_interagir = csvread("df_interagir.csv");

u_est = dfinteragir.Asian;
u_wes = dfinteragir.Western;
RAM = dfinteragir.RAM;
RAM_alt = double(RAM>2);
res_interaction = Stochastic_Copula_MLE(u_est,u_wes,3)
res_interaction_alt = Stochastic_Copula_MLE(u_est,u_wes,7,"xx",RAM_alt);
res_interaction_gumbel = Stochastic_Copula_MLE(u_est,u_wes,8,"xx",RAM_alt);

paths = res_interaction_alt.path;
mean_paths = mean(paths,2);

%Scatter plot of the copula data of the two blocks
scatter(u_wes,u_est,'k');
title("Scatter plot of copula data of western market and eastern market");
xlabel("Western")
ylabel("Eastern")
```

```
set(gca,"FontSize",12,"FontWeight",'bold')
```

***Listing 14.*** *Holistic Dependence*

```
%two models are involved in this file
% 1. SCARX
% 2. Dynamic-C-vine

%---------SCAR-CVine for markets interaction-----------------------------

%Evaluate bivariates SCAR seperately
%Number of invovled financial assets:
% r_gold: London gold fixing
% c_wes: Western market condition
% c_est: Estern market condition

%----------------BLOCK 1--------------
interagir = ["RA","c_wes","c_est"];
U_ew = csvread("U_ew.csv",'R1',2,'C1',2);
U_ew = array2table(U_ew);
U_ew.Properties.VariableNames = ["c_wes","c_est"];
U_ew.RA = r_gold;
PATH = table();
N = 100;
copulas_layer_1 = ["gold_wes","gold_est"];
U_paired_est = getPairedArgument(U,[]);
[res_layer_1_est, U_layer_1_est,paths_layer_1] = biSCAR(U_paired_est);
PATH.layer_1 = getAvgPath(paths_layer_1,N,copulas_layer_1);

%moving to the next layer
%----------------------BLOCK 2------------------------
copulas_layer_2 = "wes_estCRA";
H = csvread("H.csv", 1, 1);
H = array2table(H);
H_paired = getPairedArgument(H,[]);
[res_layer_2, U_layer_2,paths_layer_2] = biSCAR(H_paired);
PATH.layer_2 = getAvgPath(paths_layer_2,N,copulas_layer_2);


%NEXT MODEL
%-------------------------SCARX--------------------------------
model = 7;
xx = U_ew.RA;
u = U_ew.c_est;
v = U_ew.c_wes;
res_ew = Stochastic_Copula_MLE(u,v,7,"xx",xx);
```

***Listing 15.*** *SCAR under alternative Hypothesis of constant dependence parameter*

```

%-------For obtaining the log-likelihood under alternative hypothesis------------
%Reference to Hafner et al.(2012), performing LR test

function [f path]=MLEFnNormal(u,v,theta)

la1 = theta;

%now calculate the average of the log-likelihood

f = log(PdfNormal(u,v,la1));

f = -sum(f);

end

function [f path]=MLEFnGumbel(u,v,theta)

la1 = theta;

%now calculate the average of the log-likelihood

f = log(PdfGumbel(u,v,la1));

```

```
f = mean(f);

f=-f;   %this is the output
path=la1;

end

function [f path]=MLEFnClayton(u,v,theta)

la1 = theta;

%now calculate the average of the log-likelihood

f = log(PdfFrank(u,v,la1));

f = -sum(f);

end

function [f path]=MLEFnClayton(u,v,theta)

la1 = theta;

%now calculate the average of the log-likelihood

f = log(PdfClayton(u,v,la1));

f = -sum(f);

end
```

```
function [RES, t_U,t_paths] = biSCAR(U)
%Estimating SCAR pair-copula
%U: pair-wise arguments

[nrow,ncol] = size(U);
model = 3;
nPair = 0;
t_U = [];
t_paths = [];
RES = cell(8,ncol/2);
for i = 1:2:ncol
    nPair = nPair + 1;
    U.adhoc = U{:,i:i+1};
    u = U.adhoc(:,1);
    v = U.adhoc(:,2);
    results = Stochastic_Copula_MLE(u,v,model);
    path = results.path;
    t_paths = [t_paths path];
    t_U = [t_U U.adhoc];
    RES(:,nPair) = struct2cell(results);
end

%Save the results to a given directory
%they will later be read into R
csvwrite("U.csv",t_U);
csvwrite("paths.csv",t_paths);
end

%Obtain the smoothed time path
function [avgPaths] = getAvgPath(paths,N,name)
[nR,nC] = size(paths);
nVar = nC/N;
avgPaths = NaN(nR,nVar);
for i = 1:nVar
   range = (i-1)*N +1 : i*N;
   avgPaths(:,i) = mean(paths(:,range),2);
end
avgPaths = array2table(avgPaths);
avgPaths.Properties.VariableNames = name;
end

function [U_new] = getPairedArgument(U,order)
%Obtain Pair-wise argument for pair copula
if (~isempty(order))
```

```
    U = U(:,order);
end
[~,nc] = size(U);
U_new = [];
u_pivot = U{:,1};
for i = 2:nc
    U_adhoc = [u_pivot, U{:,i}];
    U_new = [U_new U_adhoc];
end
U_new = array2table(U_new);
end
```

***Listing 16.*** *File Preperation*

```
%creating csv files that will be passed to R for later analyses

%creating residual files(which will be used to estimate multivariate normal
%distribution)
writetable(EPS(:,2:end),"EPS.csv");
%creating path files
writetable(PATH_est.layer_1, "paths_layer_1_est.csv");
writetable(PATH_est.layer_2,"paths_layer_2_est.csv");

writetable(PATH_wes.layer_1, "paths_layer_1_wes.csv");
writetable(PATH_wes.layer_2, "paths_layer_2_wes.csv");
writetable(PATH_wes.layer_3, "paths_layer_3_wes.csv");
writetable(PATH_wes.layer_4, "paths_layer_4_wes.csv");
```

***Listing 17.*** *Log-likelihood of Normal SCARX*

```
function [f path]=LL_Normal_EIS_X(u,v,xx,theta,N)

%This function is built upon 'LL_Normal_EIS' from Hafner et al.(2012)

%computes the log-likelihood of the stochastic Gaussian copula by efficient
%importance sampling using results from Liesenfeld and Richard (2003)
%seed is the variable for the random number generation, so as to use the
%same one always

%xx: risk-aversion measure
%the latent process follows an ARX process
iterations=5;  %number of iterations used to draw trajectories

seed=100;

ga=theta(1);     %set parameters ("ga" is the intercept, "de" the AR coefficient and "nu" the error standard deviation
de=theta(2);
phi = theta(3);
nu=theta(4);
%ga=thetabar-de*thetabar;   %this is for an alternative specification restricting the unconditional estimate to equal it's
      ML value

T=rows(u);


ap=zeros(T,1);     %initialize parameters for importance sampler
q=zeros(T,1);
b=zeros(T,1);
av=zeros(T,1);

%penalty for parameter values at the boundary or rather extrem values to
%ensure stability of numerical procedures
if abs(de)>0.999||nu<0.001||nu>1
    f=10000;
    path=1;
    return
else


    %Step 0: generate trajectories of theta_t from the natural sampler
    la=zeros(T+1,N);               %"la" is the underlying process for the copula dependence parameter
    randn('state',seed);
    crn=randn(T+1,N);              %always use the same random number when optimizing. otherwise it will not be a smooth
         function
```

```
    eps=nu*crn;
    la(1,:)=(ga+mean(xx))/(1-de);
    xx = [mean(xx);xx];
    xx = repmat(xx,1,N);
    for j=2:T+1
        la(j,:)=ga+de*la(j-1,:)+ phi * xx(j-1,:) + eps(j,:);       %generate the first lambda series from
                                                                   %a natural sampler
    end;


    for jj=1:iterations
        %Step t: solve back-recursive problem by LS
        lnxsi=zeros(N,1);
        a1=zeros(T,1);
        a2=zeros(T,1);
        la1=(exp(2*la)-1)./(exp(2*la)+1);         %inverse Fisher transform, alternative may be considered
        for t=T:-1:1

            x=[ones(N,1) la(t+1,:)' la(t+1,:).^2'];

            y=log(PdfNormal(u(t),v(t),la1(t+1,:)))'+lnxsi;     %here I simply inserted the log of the Normal pdf as
                 dependent variable

            est=inv(x'*x)*x'*y;
            a1(t)=est(2);
            a2(t)=est(3);
            E1=ga+de*la(t,:)+phi*xx(t,:);              %parameters for the xsi function(E1: the entire nominator
            V2=(nu^2)/(1-2*(nu^2)*est(3));
            E2=V2*((E1/(nu^2))+est(2));%-0.5

            lnxsi=(0.5*(((E2.^2)/V2)-((E1.^2)/(nu^2))))'; %update the chi-term

            ap(t)=V2*((ga/(nu^2))+est(2));%-0.5
            q(t)=(V2/(nu^2))*de;
            b(t) = (V2/(nu^2))*phi;
            av(t)=V2;
            %     %calculate the R^2
            %     yhat = x*est;
            %     resid = y - yhat;
            %     sigu = resid'*resid;
            %     ym = y - mean(y);
            %     rsqr1 = sigu;
            %     rsqr2 = ym'*ym;
            %     R2(t) = 1.0 - rsqr1/rsqr2; % r-squared  (seems to be always one)

        end;
        % mean(R2)

        %Step T+1: draw N trajectories of lambda from m-function (including
        %iteration steps)
        for j=2:T+1
            la(j,:)=ap(j-1)+q(j-1)*la(j-1,:)+b(j-1)*xx(j-1,:)+sqrt(av(j-1)).*crn(j,1:N);    %data drawn from the
                 importance sampler m
        end;
        if mean(mean(isnan(la)))>0   %activate this part when there are numerical problems creating many warnings
            f=10000;
            path=1;
            return
        end;
    end;%(here the iteration ends)


    %now calculate the average of the log-likelihood

    mu1s=(la(2:T+1,:)-sqrt(av*ones(1,N)).*crn(2:T+1,:)).^2;   %parameters for xsi
    mu0s=(ga+de*la(1:T,:)+phi*xx(1:T,:)).^2;

    lnxsi=0.5*((mu1s./(av*ones(1,N)))-(mu0s/(nu^2)));



    %now calculate the average of the log-likelihood
    CL=zeros(T,N);
    la1=(exp(2*la)-1)./(exp(2*la)+1);

    for i= 1:N
```

```
117
118         CL(:,i) = log(PdfNormal(u,v,la1(2:T+1,i)))-...
119             (a1).*la(2:T+1,i)-(a2).*(la(2:T+1,i).^2)+...
120             lnxsi(:,i)-0.5*log(nu^2)+0.5*log(av);
121
122     end;
123     scal1=mean(CL);    %these lines because mean of log is not equal to log of mean
124     scal2=mean(scal1);
125     lngs=-scal2+CL;
126     tg1=log(mean(exp(sum(lngs))));
127     f=T*scal2+tg1;
128
129     f=-f  %this is the output
130 %     path=mean(la1(2:T+1,:)')';
131     path = la1(2:T+1,:);
132
133 end;
```