VINE COPULA BASED QUANTILE REGRESSION

Yitong Lin

Supervisor: Dr Donna Saplopek

School of Mathematics and Statistics
UNSW Sydney

June 2018

Submitted in partial fulfilment of the requirements of the degree of
Master of Financial Mathematics

Plagiarism statement

I declare that this thesis is my own work, except where acknowledged, and has not been submitted for academic credit elsewhere.

I acknowledge that the assessor of this thesis may, for the purpose of assessing it:

- Reproduce it and provide a copy to another member of the University; and/or,
- Communicate a copy of it to a plagiarism checking service (which may then retain a copy of it on its database for the purpose of future plagiarism checking).

I certify that I have read and understood the University Rules in respect of Student Academic Misconduct, and am aware of any potential plagiarism penalties which may apply.

By signing this declaration I am agreeing to the statements and conditions above.

Signed: _____ Date: _____

Abstract

Quantile regression, which allows for predicting conditional quantiles has gained importance in statistical learning. Copula that defines the dependence structure between variables can help to evaluate the conditional quantile. Two semiparametric quantile regression methods are investigated. The first method is a variation of traditional quantile regression, by utilizing R-vine to model the dependencies between the data, and solve for conditional quantile through methods of least squares. The second methods fits the data into a D-vine and the quantiles can be solved analytically. Both models are highly flexible, and in simulation studies, have comparable or better accuracy than established quantile regression methods. An application on the Australian equity market is used to demonstrate the performance of the two copula based quantile regression methods.

Contents

Chapter 1

Introduction

The multivariate Gaussian distribution is the most popular statistical dependence model. However, there is a growing demand for non-Gaussian models especially in finance (Dissmann et al., 2013). With large samples, we are able to model non-Gaussian dependencies using *vine copula*. The Sklar's theorem(Sklar, 1973) is the bases of such models, which allows us to construct multivariate distribution from the marginal distribution and copulae. It allows for the copulae parameters to be specified independently from the margins. Generally, copulae can be classified into two categories: elliptical (Fang et al., 2002) and Archimedean (McNeil, 2008). Elliptical copulae include the symmetric Gaussian and Student t-copula, and Archimedean copulae include Clayton copula.

For practical applications, Gaussian copula does not allow for heavy tail and the approach of using Gaussian copula to value collateral debt obligations (CDO) (see Li (1999)) has been criticized for contributing to the Global Financial Crisis in between mid 2007 and early 2009 (Salmon, 2012). Thus more flexible copulae were needed for applications.

Bedford and Cooke (2002) presented an new approach and class to model multivariate data with copulae. This approach allowed for utilization of larger class of copulae by constructing the multivariate dependence structure using only bivariate copulae. The bivariate variables and its conditioning variables are stored in a sequence of trees. These trees are called *regular vines* (R-vine). Aas et al. (2009) then popularized two sub type of R-vine, *canonical vines* (C-vines) and *drawable vines* (D-vine). C-vine has a star like tree structure, while D-vine has a path like tree structure.

Based on the flexibility of copula, we investigate two quantile regression methods that utilize vine copula. The first is a semiparametric method of Noh et al. (2014) and it rewrites the characterization of a regular quantile regression in terms of copula and marginal distribution. This method requires numerical optimization. The second approach was proposed by Kraus and Czado (2017), and it fits an D-vine to the data which allows the conditional quantile to be calculated empirically. Simulations are conducted on the two copula based quantile regression and its performance is tested against other well established quantile regression.

Finally, we apply the two copula based quantile regression to financial data.

Chapter 2

Vine Copula

This chapter begin with the theoretical background and few key definition of vine copulas.

2.1 Regular vine

Bedford and Cooke (2002) introduced a graphic model denoted as *Regular vines (R-vine)*. The class of regular vines is very general and encompass a large number of pair-copula decomposition. The definition of R-vine is given as:

Definition 2.1.1 (R Vine). *$V = (T_1, ..., T_n)$ is an R-vine on n elements if:*

1. *T_1 is a tree with nodes $N_1 = \{1, ..., n\}$ and set of edges denoted E_1.*
2. *For $i = 2, ..., n-1, T_i$ is a tree with nodes $N_i = E_{i-1}$ and edge set E_i.*
3. *For $i = 2, ..., n-1$ and $a, b \in E_i$ with $a = a_1, a_2$ and $b = b_1, b_2$ it must hold that $\#(a \cap b) = 1$ (proximity condition), where $\#$ denotes the cardinality of a set.*

Next, we need to define complete union, conditioning sets and conditioned sets.

Definition 2.1.2 (Complete Union, Conditioning Sets and Conditioned Sets). *For edge $e_i = a, b \in E_i, a, b \in N_i, i = 1, ..., n-1$:*

1. *Complete union: $U_{e_i} = n_1 \in N_1 | n_1 \in e_1 \in ... \in e_i \subset N_1$*
2. *Conditioning set: $D_{e_i} = U_a \cap U_b$.*
3. *Conditioned sets: $C_{e_i,a} = U_a \backslash D_{e_i}$ and $C_{e_i,b} = U_b \backslash D_{e_i}$ and $C_{e_i} = C_{e_i,a} \cup C_{e_i,b}$.*

The collection of conditioning sets and conditioned are collected in a set called *constrained set*. This set comprises of pair of conditioned sets and a conditioning set.

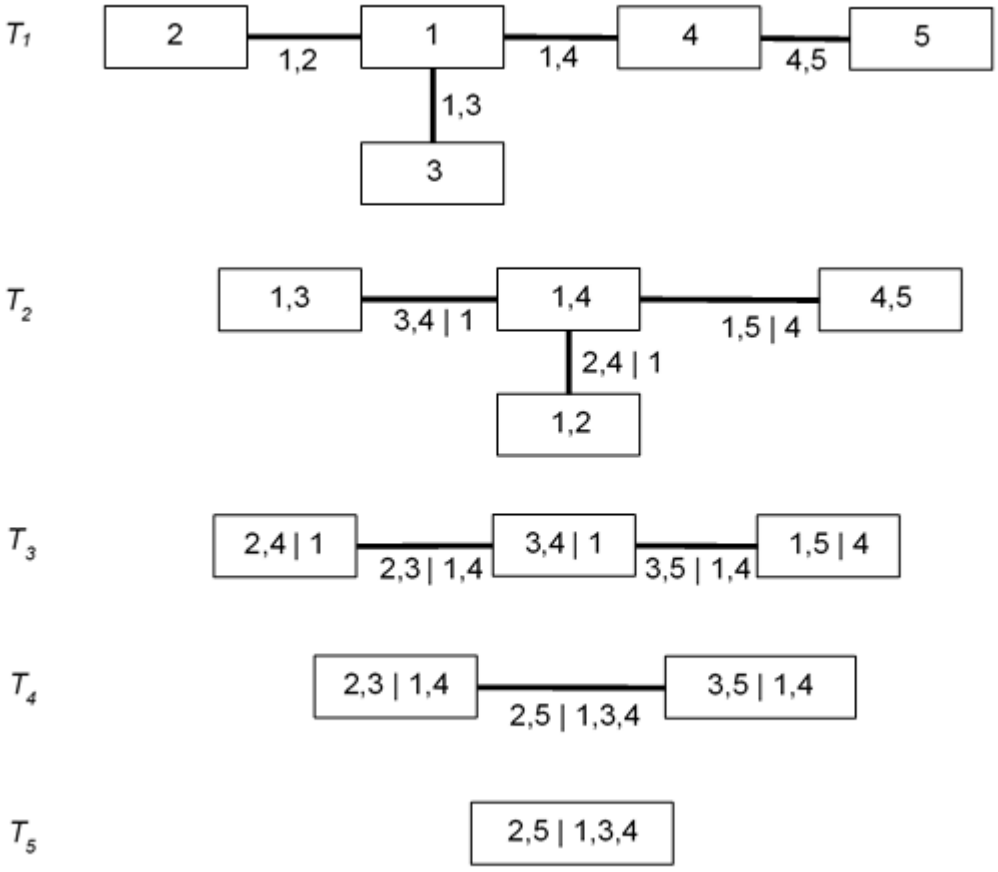
Definition 2.1.3 (Constrained set). *For a vine V, the constrained set is a set:*

$$CV = \{(\{C_{e,a}, C_{e,b}\}, D_e) | e \in E_i, i = 1, ..., n-1\}$$

The constrained set defines the bivariate copula in each edges and subsequently the nodes in the next tree. We will illustrate this through an five dimensional example in Figure 2.1. The first tree comprise of nodes $N_1 = \{1, 2, 3, 4, 5\}$ and edges $E_1 = \{(\{1, 2\}), (\{1, 3\}), (\{1, 4\}), (\{4, 5\})\}$. Notice since it's the first tree it does not have conditioning set $D_e = \emptyset, e \in E_1$. For the second tree, $N_2 = E_1$ and $E_2 = \{(\{3, 4\}, 1), (\{2, 4\}, 1), (\{1, 5\}, 4)\}$

For Figure 2.1, we require four unconditioned copulae $c_{4,5}, c_{1,4}, c_{1,3}, c_{1,2}$ in T_1 and three conditioned copulae $c_{4,2|1}, c_{3,4|1}, c_{1,5|4}$ in T_2, etc. Every copula in the tree can be specified independently from each other.

Figure 2.1: R-vine with 5 dimensions. At each edge $e = \{a, b\} \in E_i$, $C_{e,a}$ and $C_{e,b}$ are separated by comma and are on the left of "|", while D_e appears on the right.



After defining the three structure and edge values, the density of an R-vine copula can be specified (Bedford and Cooke, 2002). This is equivalent to the product of all the copulae in in in the R-vine:

$$f_{1,\dots,d}(\mathbf{x}) = \prod_{k=1}^{n} f_k(x_k) \prod_{i=1}^{n-1} \prod_{e \in E_i} c_{C_{e,a},C_{e,b}|D_e}(F_{C_{e,a}|D_e}(x_{C_{e,a}}|\mathbf{x}_{D_e}), F_{C_{e,b}|D_e}(x_{C_{e,b}}|\mathbf{x}_{D_e})) \tag{2.1.1}$$

where $\mathbf{x} = (x_1, ..., x_d)$ and $e = \{a, b\}$, and $\mathbf{x}_{D_e} = \{x_i | i \in D_e\}$ and f_i is the density function for x_i. The conditional copulas are called *h-functions* and can be obtained recursively (Joe, 1996):

Definition 2.1.4 (Conditional distribution (h-functions))**.** *Let $e = \{a, b\} \in E_i$ and $a = \{a_1, a_2\}$ and $b = \{b_1, b_2\}$ to be the edge which connects $C_{e,a}$ with $C_{e,b}$ given the variable D_e*

$$F_{C_{e,a}|D_e}(x_{C_{e,a}|D_e}) = \frac{\partial C_{C_a|D_a}(F_{C_{a,a_1}|D_e}(x_{C_{a,a_1}}|\mathbf{x}_{D_a}), F_{C_{a,a_2}|D_e}(x_{C_{a,a_2}}|\mathbf{x}_{D_a}))}{\partial F_{C_{a,a_2}|D_e}(x_{C_{a,a_2}}|\mathbf{x}_{D_a})} \tag{2.1.2}$$

where $F_{C_{a,a_1}|D_e}(x_{C_{a,a_1}}|\mathbf{x}_{D_a})$ *and* $F_{C_{a,a_2}|D_e}(x_{C_{a,a_2}}|\mathbf{x}_{D_a})$ *are obtained recursively. For simplicity, we will use the* h-function *notation:*

$$h(F_{C_{a,a_1}|D_e}(x_{C_{a,a_1}}|\mathbf{x}_{D_a}), F_{C_{a,a_2}}|(x_{C_{a,a_2}}|\mathbf{x}_{D_a})) \tag{2.1.3}$$

This definition is the key for D-vine quantile regression which we will discuss later. This shows us how to find the inverse and conditional quantile empirically.

2.2 D-vine

The *Drawable vine (D-vine)* is a special class of vine structure within the R-vine, and was introduced by Kurowicka and Cooke (n.d.)

The D-vine is a R-Vine where the first tree has nodes with degree two or less and all of the trees have a path like structure without any branches. The formal definition of its density is given by the following:

Definition 2.2.1 (Density of D-vine)**.** *The density of* $\mathbf{x} = (x_1, ..., x_n)$ *of a D-vine is:*

$$f(\mathbf{x}) = \prod_{k=1}^{n} f(x_k) \prod_{j=1}^{n-1} \prod_{i=1}^{n-j} c_{i,i+1|i+1,...,i+j-1}(F(x_i|x_{i+1}, ..., x_{i+j-1}, F(x_{i+j}|x_{i+1,...,x_{i+1-1}})) \tag{2.2.1}$$

where index j identifies the trees, and index i iterate through the edges in each tree.

Figure 2.2 illustrate a D-vine in five dimensions. An advantage of D-vine is that the resulting correlation matrix is always positive definite without imposing restrictions on the parameters (Hofmann and Czado, 2010).

As you see from Figure 2.2 and 2.3, the D-vine and C-vine are very different structure.

2.3 C-vine

The *canonical vine (C-vine)* is also an R-vine which contains a node with maximal degree and all of the trees have a star like structure. Figure 2.3 shows a possible C-vine in five dimensions. The formal definition of its density is given by the following:

Definition 2.3.1 (Density of C-Vine)**.** *The density of* $\mathbf{x} = (x_1, ..., x_n)$ *of a C-vine is:*

$$f(x) = \prod_{k=1}^{n} f(x_k) \prod_{j=1}^{n-1} \prod_{i=1}^{n-j} c_{j,j+1|1,...,j-1}(F(x_j|x_1, ..., x_{j-1}, F(x_{j+1}|x_{1,...,x_{j-1}}))$$

Fitting a C-vine could be advantageous when a particular variable is a key variable that controls the interactions in the data.

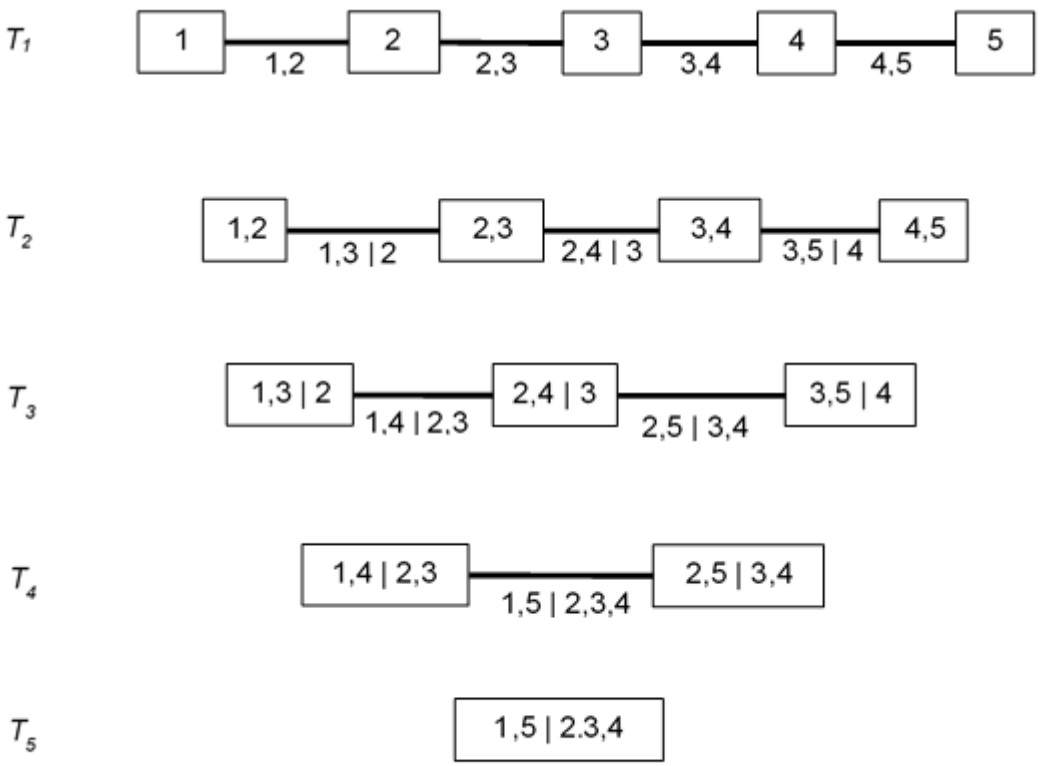


Figure 2.2: D-vine with five dimensions

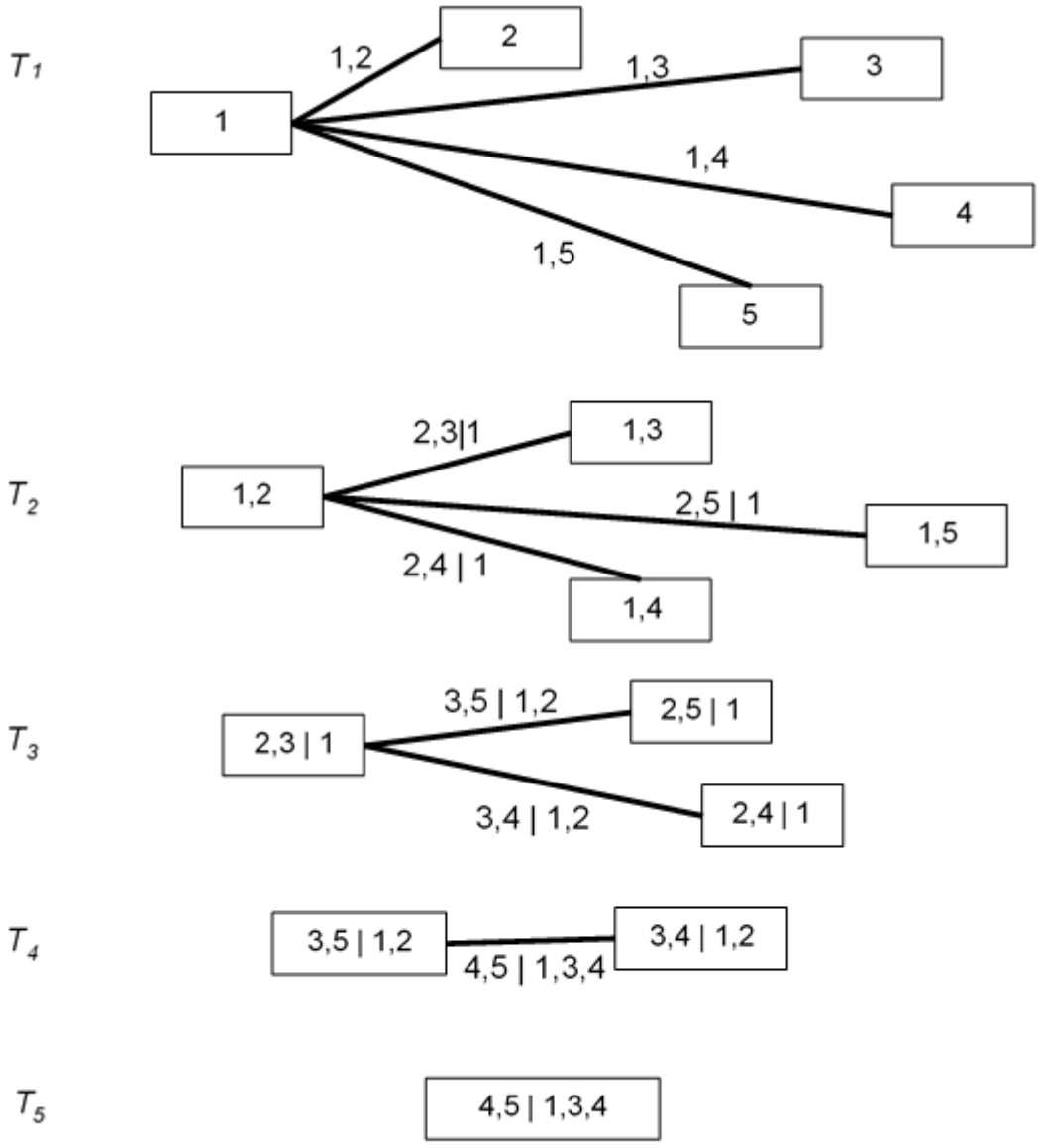


Figure 2.3: C-vine with five dimensions

Chapter 3

Copula based quantile regression

Quantile regression, which is the prediction of conditional quantiles, has attracted interest and application in various field, especially in finance. As mentioned by Buchinsky (1998), quantile regression models have a number of useful properties: (i) with non-Gaussian error terms, quantile regression estimators may be more efficient than least-square estimators, (ii) the entire conditional distribution can be characterized, (iii)different relationships between the response and the predictor variable may arise at different quantiles. In this chapter, we will discuss three methods of copula and vine copula based quantile regression techniques.

3.1 Semiparametric quantile regression

Noh et al. (2014) proposed semiparametric this regression to fit the response variable Y and predictor variable $\mathbf{X} = (X_1, ..., X_d)$ to a R-vine. They used a check function to perform weighted quantile regression. This approach of nonlinear quantile regression modelling is based on the copula function that defines the dependency structure between response variable Y and predictor variable X. Their approach is an extension of the principle of linear quantile regression.

3.1.1 Quantile regression estimator

Let the loss function be given as $\rho_\alpha(y) = y(\alpha - I_{(y<0)})$ where I is an indicator function. Koenker and Bassett Jr (1978) defined the quantile estimation as the following:

Definition 3.1.1 (Quantile estimation)**.** *A quantile $\alpha \in (0, 1)$ can be found by minimizing the expected loss of $Y - u$ with respect to u:*

$$\arg\min_u E(\rho_\alpha(Y - u)) \tag{3.1.1}$$

and the estimated sample quantile can be found by solving the minimizing problem:

$$\hat{q}_\alpha(x_1, ...x_d) = \arg\min_{q\in\mathbb{R}} \sum_{i=1}^{n} \rho_\alpha(y_i - q) \tag{3.1.2}$$

Noh et al. (2014) suggested modelling Y and $\mathbf{X}$ using copula, thus referring to R-vine density function (2.1.1), the conditional density of Y given $\mathbf{x} = (x_1, ..., x_d)$

is expressed as:

$$\begin{aligned} f(y|x_1,...,x_d) &= \frac{c(F(y),F_1(x_1),...,F_d(x_d))f(y)f(x_1),...,f(x_d)}{c(F_1(x_1)...F_d(x_d))f(x_1)...f(x_d)} \\ &= f_Y(y)\frac{c(F(y),F_1(x_1),...,F_d(x_d))}{c(F_1(x_1),...,F_d(x_d))} \end{aligned} \tag{3.1.3}$$

Hence the conditional quantile can be estimated using copula quantile regression models. We can achieve this by modifying the quantile function (3.1.1), by transforming the expected loss from expectation with respect to $Y|\mathbf{X}$ to Y:

$$\begin{aligned} q_\alpha(x_1,...,x_d) &= \arg\min_u E(\rho_\alpha(Y-u)|x_1,...,x_d) \\ &= \arg\min_u E(\rho_\alpha(Y-u)c(F(Y),F(x_1,..,x_d))) \end{aligned} \tag{3.1.4}$$

and the conditional quantile estimate becomes:

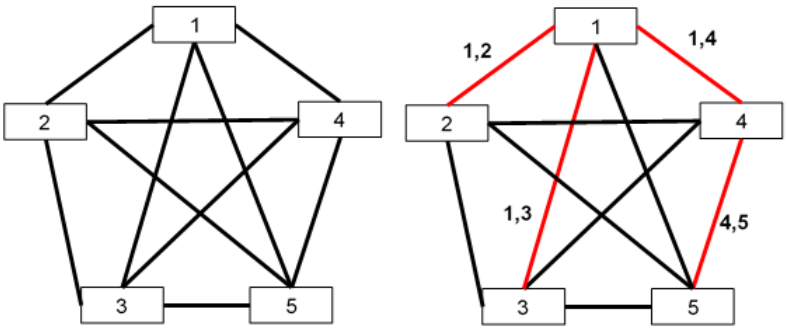
$$\hat{q}_\alpha(x_1,...,x_d) = \arg\min \sum_{i=1}^{n} \rho_\alpha(Y_i-u)\widehat{c}(\hat{F}(Y_i),\hat{F}(x_1,..,x_d)) \tag{3.1.5}$$

Next, we move on to estimating $\widehat{c}()$ with an R-vine.

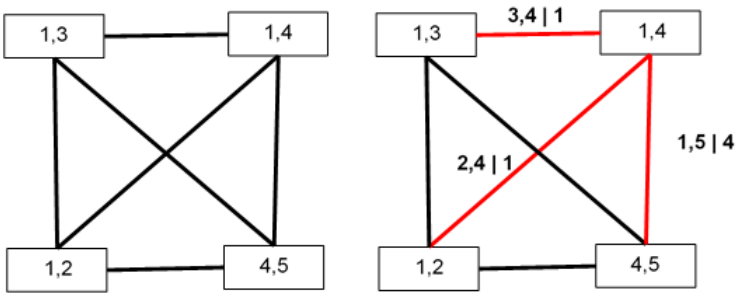
3.1.2 Estimating R-vine

To perform this quantile regression, we first need to estimate the copula on Y and $X_1,...,X_d$. Kraus and Czado (2017) suggested selecting a R-vine to the data. The algorithms for fitting an R-vine is discussed in Dissmann et al. (2013). It aims at sequentially selecting an R-vine model with most explanatory power. For each iteration, it aims to find the tree that maximize the sum of Kendall's tau of the of its edges, which can be found using minimum spanning tree algorithms such as Prim's algorithm. The process of the algorithm is shown in the example below.

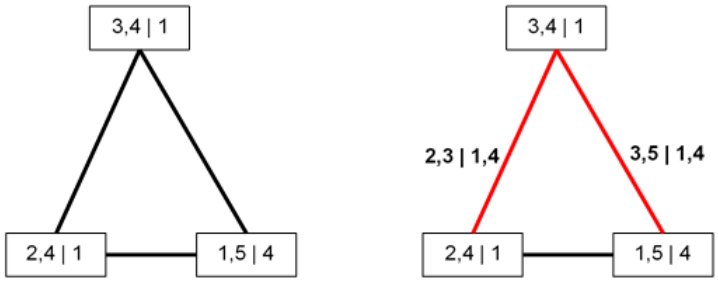
Example 3.1.2 (R-vine selection)**.** *For a five dimension data, the steps are the following:*
Iteration 1: *Map out all the the possible edges for T_1 and select the optimal path that maximize the sum of Kendall's tau. The sample tree is shown in red. T_1 is selected.*



Iteration 2: *Use the edges of the sample tree from T_1 as the nodes for T_2. Repeat the tree selection process. T_2 is selected.*



Iteration 3: *Use the edges of the sample tree from T_2 as the nodes for T_3. Repeat the tree selection process. T_3 is selected.*



Iteration 4: *Connect the two edges in T_3 to form T_4. Algorithm terminates.*

The pseudo code is shown in A.1. The R-vine selection methods is implemented in `RVineStructSelect` copula density $\widehat{c}(F(Y_i), F(X_1, ..X_d))$ is calculated with `RVinePDF` from the package `VineCopula`. The conditional quantile estimate (3.1.5) is solved using `optimize`. For source code of implementation see B.1

3.2 D-vine based quantile regression

Kraus and Czado (2017) introduced the D-vine quantile regression method. The purpose of D-vine quantile regression is to predict the quantile of a response variable Y given predictor variable $\mathbf{X} = (X_1, ..., X_d), d \geq 1$, where the distribution is defined as $Y \sim F_Y$ and $X_j \sim F_j$.

3.2.1 Conditional quantile function

We need to model the relationship between Y and $\mathbf{X}$ and the *conditional quantile function* for $\alpha \in (0, 1)$:

$$q_\alpha(x_1, ..., x_d) = F^{-1}_{Y|X_1,...,X_d}(\alpha|x_1, ..., x_d) \tag{3.2.1}$$

Applying probability integral transform (PIT) $V = F_Y(Y)$ and $U_j = F_j(X_j)$ with respective PIT value $v = F_Y(y)$ and $u_j = F_j(x_j)$, then:

$$\begin{aligned} F_{Y|X_1,...,X_d} &= P(Y \leq y|X_1 = x_1, ..., X_d = x_d) \\ &= P(F_Y(Y) \leq v|F_1(X_1) = u_1, ..., F_d(X_d) = u_d) \\ &= C_{V|U_1,...,U_d}(v|u_1, ..., u_d) \end{aligned}$$

Therefore, the inverse becomes

$$F^{-1}_{Y|X_1,...,X_d}(\alpha|x_1, ..., x_d) = F^{-1}_y\big(C^{-1}_{V|U_1,...,U_d}(\alpha|u_1, ..., u_d)\big) \tag{3.2.2}$$

The conditional quantile function can be expressed in terms of the inverse distribution F^{-1}_Y response variable Y, and the *conditional copula quantile function* $C^{-1}_{V|U_1,...,U_d}$ conditioned on the PIT values of $\mathbf{x}$.

Finally, we can obtain an estimate of the conditional quantile function by estimating the marginals F_Y and F_j and substituting into (3.2.2):

$$\hat{q}_\alpha(x_1, ..., x_d) = \hat{F}^{-1}_Y\big(C^{-1}_{V|U_1,...,U_d}(\alpha|u_1, ..., u_d)\big) \tag{3.2.3}$$

3.2.2 Estimating D-vine

With regards to the estimating the multivariate copula $C_{V,U_1,...U_d}$, Kraus and Czado (2017) proposed fitting a D-vine copula to $(V, U_1, ..., U_d)$, such that V is the starting node in the first tree. This allows for a flexible class of copulas since each bivariate copula of the pair-construction can be estimated separately. Finally, this allows $C^{-1}_{V|U_1,...,U_d}$ to be easily calculable using a recursive method. The example below illustrate this method.

Example 3.2.1. *D-vine with structure $V - U_1 - ... - U_j$, the conditional distribution of V given $(U_1, ..., U_j)$ for $j = 1, 2, 3, ...$ can be expressed recursively as:*

$j = 1$ $\quad C_{V|U_1}(v|u_1) = h_{V|U_1}(v|u_1)$

$j = 2$ $\quad C_{U_2|U_1}(u_2|u_1) = h_{U_2|U_1}(u_2|u_1)$

$\quad C_{U_1|U_2}(u_1|u_2) = h_{U_1|U_2}(u_1|u_2)$

$\quad C_{V|U_1,U_2}(v|u_1, u_2, u_3) = h_{V|U_2;U_1}(C_{V|U_1}(v|u_1)|C_{U_2|U_1}(u_2|u_1))$

$j = 3$ $\quad C_{U_3|U_2}(u_3|u_2) = h_{U_3|U_2}(u_3|u_2)$

$\quad C_{U_3|U_1,U_2}(u_3|u_1, u_2) = h_{U_3|U_1,U_2}(C_{U_3|U_2}(u_3|u_2)|C_{U_1|U_2}(u_1|u_2))$

$\quad C_{V|U_1,U_2,U_3}(v|u_1, u_2, u_3) = h_{V|U_3;U_1,U_2}(C_{V|U_1,U_2}(v|u_1, u_2)|C_{U_3|U_1,U_2}(u_3|u_1, u_2))$

The inverse of this yields the conditional copula quantile functions:

$j = 1$ $\quad C^{-1}_{V|U_1}(v|u_1) = h^{-1}_{V|U_1}(\alpha|u_1)$

$j = 2$ $\quad C^{-1}_{V|U_1,U_2}(v|u_1, u_2) = C^{-1}_{V|U_1}(h^{-1}_{V|U_2;U_1}(\alpha|h_{U_2|U_1}(u_2|u_1))|u_1)$

$j = 3$ $\quad C^{-1}_{V|U_1,U_2,U_3}(v|u_1, u_2, u_3) = C^{-1}_{V|U_1,U_2}(\alpha|h_{U_3|U_1;U_2}(h_{U_3|U_2}(u_3|u_2))|u_1, u_2)$

...

Note that $C^{-1}_{V|U_1,...,U_d}(v|u_1, ..., u_d)$ is monotonically increasing with respect to α, thus crossing of quantile functions corresponding to different levels of quantile is not possible. This can be a problem for other linear and non-linear quantile regression.

We than fit a D-vine with order $V - U_{m_1} - ... - U_{m_d}$ to the pseudo copula data. We want to select the ordering $M = (m_1, ..., m_d)$ which has the highest explanatory power. If we compare all $d!$ possible permutations of ordering it would be inefficient and takes very long time. Therefore, (Kraus and Czado, 2017) proposes a forward sequential selection methods that starts with zero covariates and in each step add the covaraite to the model that improves the model fit the most. The measure of model fit chosen is the conditional log-likelihood(cll), which given parameters $\hat{\theta}$ and pseudo copula data u is defined as:

$$cll = \sum_{i=1}^{n} ln\ c_{v|u}\big(\hat{v}^{(i)}|\hat{\mathbf{u}}^{(\mathbf{i})},\big) \tag{3.2.4}$$

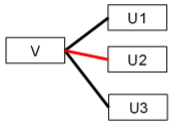
where the conditional copula density $c_{v|\mathbf{u}}$ is defined as:

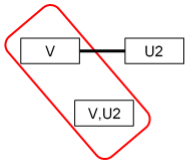
$$
\begin{aligned}
c_{v|\mathbf{u}}(\hat{v}^{(i)}|\hat{\mathbf{u}}^{(\mathbf{i})}) = & c_{vu_{m_1}}(\hat{v}^{(i)}, \hat{u}^{(i)}_{m_1}) \\
& \prod_{j=2}^{d} c_{vu_{m_j}|u_{m_1},\ldots,u_{m_{j-1}}}\big(C_{v|u_{m_1},\ldots,u_{m_{j-1}}}(\hat{v}^{(i)}_{m_j}|\hat{u}^{(i)}_{m_1},\ldots,\hat{u}^{(i)}_{m_{j-1}}), \\
& C_{u_{m_j}|u_{m_1},\ldots,u_{m_{j-1}}}(\hat{u}^{(i)}_{m_j}|\hat{u}^{(i)}_{m_1},\ldots,\hat{u}^{(i)}_{m_{j-1}})\big)
\end{aligned}
$$

We will now demonstrate the D-vine quantile regression algorithm with a three dimensional example.

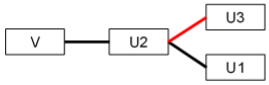
Example 3.2.2 (D-vine selection)**.** *For a three dimension data, the steps are the following:*

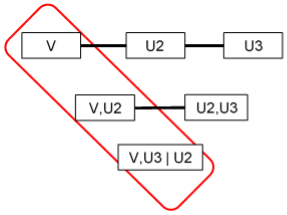
Iteration 1: *From the predictor set $\{u_1, u_2, u_3\}$, add u_2 to the D-vine which maximizes cll. The cll is calculated by taking the product of copula density in the red circle. The ordering is now $M = (2)$*



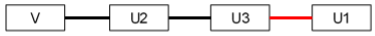


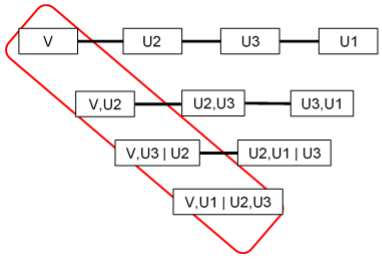
Iteration 2: *From the predictor set $\{u_1, u_3\}$, add u_3 to the D-vine. The ordering is now $M = (2, 3)$*





Iteration 3: *From the predictor set $\{u_1\}$, add the only predictor left to the D-vine. The ordering is now $M = (2, 3, 1)$. Since no predictors are left, algorithm terminates.*





The pseudo code is shown on A.2. The D-vine selection method is not implemented in any packages, so we implemented this method independently, using `BiCopSelect` from `VineCopula` to select the bivariate copula within each tree. For detailed implementation, see B.2.

3.3 Estimation process

This sections explains how to estimate $\widehat{q}_\alpha(\mathbf{x})$ using a two step estimation procedures to estimate the (1)marginals F_Y and $F_j, j = 1, ..., d$ and (2)R-Vine or D-Vine and their conditional copula quantile function.

Let $\mathbf{y} = (y^{(i)})_{i=1,...,d}$ and $(X) = (x_j^{(i)})_{j=1,..,d,i=1,...,n}$ be n independent and identically distributed sample from random vector $(Y, X_1, ..., X_d)$.

First step: estimation of marginals

The estimation of univarate CDF $\hat{F}_j$ can be done in a multiple of ways such as kernel density estimation or logspline density estimation. We have two choices of how to fit the marginal distributions, either parametrically or nonparametrically. Depending on the choice, this could either result in a fully parametric or semiparametric estimate of $q_\alpha(\mathbf{x})$. Noh et al. (2014) pointed out the marginals modelling marginals and copula parametrically might cause the estimator to be biased and inconsistent if one of the parametric models is misspecified. Therefore, we implement the semiparametric approach and estimate the marginals nonparametrically. We choose continuous kernel smoothing estimator (Parzen, 1962), which for a given sample $(x^{(i)})_{i=1,...,n}$ is defined as:

$$\hat{F}(x) = \frac{1}{n}\sum_{i=1}^{n} K(\frac{x - x^{(i)}}{h}), \quad x \in \mathbb{R} \tag{3.3.1}$$

where $K(x)$ is a probability distribution function. Therefore we marginal distribution estimates $\hat{F}_Y$ and $\hat{F}_j$. We then perform PIT on the data using the above marginal distribution estimates to obtain pseudo copula data. i.e. $v^{(i)} = \hat{F}_Y(y^{(i)}), i = 1, ..., n$ and $u_j^{(i)} = \hat{F}_j(x_j^{(i)}), j = 1, ..., d, i = 1, ...n$.

Second step: estimation of copula

Once the marginals estimates $\widehat{F}_Y$ and $\widehat{F}_1, ..., \widehat{F}_d$ and their pseudo copula data v and $u_1, ..., u_d$ are estimated, we will estimate either an R-vine or D-vine using the methods and algorithms discussed in section 3.1 or 3.2, depending on which quantile regression methods we choose. Finally, we find the estimated conditional quantile $\widehat{q}_\alpha(\mathbf{x})$ either through numerical optimization for semi parametric method, or calculate empirically in the case of D-vine methods.

Chapter 4

Simulation

In this chapter, motivated by (Kraus and Czado, 2017) we perform simulation study on the three copula based quantile regression describe above as well as two additional quantile regression methods for comparison:

Linear quantile regression (LQR)

Linear quantile regression is first introduced by Koenker and Bassett Jr (1978). Let $(y^{(i)}), i = 1, ..., n$ and $(x_j^{(i)}), j = 1, ..., d, i = 1, ..., n$ be sample from Y and X respectively. Then the αth quantile function is defined as:

$$\hat{q}_\alpha(x_1, ..., x_d) = \hat{\beta}_0 + \sum_{j=1}^{d} \hat{\beta}_j x_j$$

where the parameter $\hat{\beta}_j, j = 0, ..., d$ is the solution to the linear programming problem

$$\min_{\beta_j \in \mathbb{R}, j=1,..,d} \left[\alpha \sum_{i=1}^{n} \left(y^{(i)} - \hat{q}_\alpha(x_1^{(i)}, ..., x_d^{(i)}) \right)^+ + (1-\alpha) \sum_{i=1}^{n} \left(\hat{q}_\alpha(x_1^{(i)}, ..., x_d^{(i)}) - y^{(i)} \right)^+ \right]$$

Boosting additive quantile regression (BAQR)

This methods relaxes the linearity assumption in LQR by utilizing additive models for quantile regression. The quantile regression method proposed by Koenker (2011) is defined as:

$$\hat{q}_\alpha(x_1^{(i)}, ..., x_d^{(i)}, z_1^{(i)}, ..., z_J^{(i)}) = \hat{\beta}_0 + \sum_{j=1}^{d} \hat{\beta}_j x_j^{(i)} + \sum_{j=1}^{J} \hat{g}_j(z_j^{(i)})$$

where $\hat{g}_j$ is a smooth function on the continuous variable $z_j^{(i)}, j = 1, ..., J$

Semi parametric quantile regression(SPQR) Discussed in Chapter 3 section 1 (Noh et al., 2014).

D-vine quantile regression(DVQR) Discussed in Chapter 3 section 2 (Kraus and Czado, 2017).

Case	Copula parameter	Marginals			
CL3	$\delta_1 = 0.8$, $\delta_2 = 4.67$		Y	X_1	X_2
		M_1	$N(0,1)$	$t_4(0,1)$	$N(1,4)$
		M_2	$st_4(0,1,2)$	$sN(-2,0.5,3)$	$st_3(1,2,5)$
t5	S_1, S_2		Y	X_1	X_2
		M_1	$N(0,1)$	$t_4(0,1)$	$N(1,4)$
		M_2	$st_4(0,1,2)$	$sN(-2,0.5,3)$	$st_3(1,2,5)$
			X_3	X_4	
			$t_4(0,1)$	$N(1,4)$	
			$sN(-2,0.5,3)$	$st_3(1,2,5)$	

Table 4.1: Copula parameters and and marginal setting for **CL3** and **t5**

We follow the same simulation setup as Kraus and Czado (2017) with slight modification:

1. **CL3** : (Y, X_1, X_2) follows a three-dimensional Clayton copula with parameter δ_1 and δ_2 and margin set M_1 or M_2
2. **t5** : $(Y, X_1, ..., X_2)$ follows a five-dimensional t-copula with 3 degrees of freedom, correlation matrix S_1 or S_2 and margin set M_1 or M_2. The correlation matrix is given by:

$$S_1 = \begin{bmatrix} 1 & 0.6 & 0.5 & 0.5 & 0.4 \\ 0.6 & 1 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 1 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & 1 & 0.5 \\ 0.4 & 0.5 & 0.5 & 0.5 & 1 \end{bmatrix} \quad S_2 = \begin{bmatrix} 1 & 0.27 & 0.74 & 0.72 & 0.41 \\ 0.27 & 1 & 0.28 & 0.29 & 0.27 \\ 0.74 & 0.28 & 1 & 0.74 & 0.42 \\ 0.72 & 0.29 & 0.74 & 1 & 0.40 \\ 0.41 & 0.27 & 0.42 & 0.4 & 1 \end{bmatrix}$$

3. **N5** : $X \sim N(0, \Sigma)$, with $\Sigma_{ij} = 0.5^{|i-j|}$ and
$Y = \sqrt{3X_1 + 0.5X_2 + 2} + (-0.7X_3 + 2)(0.2X_4^3) + \sigma\epsilon$, $\epsilon \sim N(0,1), \sigma \in \{0.5, 1\}$

Table 4.1 gives details to regarding parameters and marginal for **CL3** and **t5**. The true quantiles are known for these scenarios (see B.3,B.4,B.5). The measure the performance and error of the four quantile regression methods mentioned above, we use out-of-sample mean integrated square error(MISE). The MISE of method m is defined as

$$MISE_m = \frac{1}{R}\sum_{r=1}^{R}\left[\frac{1}{n_{eval}}\sum_{i=1}^{n_{eval}}\{\hat{q}_{m,\alpha}^{(r)}(\mathbf{x}_{r,i}^{eval}) - q_\alpha(\mathbf{x}_{r,i}^{eval})\}^2\right]$$

where we simulate for $r = 1, ..., R = 100$ replications. For each replication r,the simulation is divided into three steps:

1. Simulate training data $(y_{r,i}^{train}, \mathbf{x}_{r,i}^{train}), i = 1, ..., n_train$ from the distribution $(Y, \mathbf{X})$. Estimate $\hat{q}_{m,\alpha}^{(r)}$ from the training set.
2. Simulate evaluation data set $\mathbf{x}_{r,i}^{train}, i = 1, ..., n_eval$ from the distribution of $\mathbf{X}$. Predict $\hat{q}_{m,\alpha}^{(r)}$ using evaluation set.
3. Calculate MISE for each methods

Marginals	Parameters	α	SPQR	DVQR	LQR	BAQR
M1	δ_1	0.5	0.0245	0.0246	0.0590	0.0339
		0.95	0.0420	0.0410	0.1146	0.0627
	δ_1	0.5	0.0209	0.0244	0.0419	0.0169
		0.95	0.0464	0.0594	0.1356	0.0492
M2	δ_1	0.5	0.0195	0.0179	0.0892	0.0592
		0.95	0.1214	0.0855	0.5056	0.2186
	δ_1	0.5	0.0264	0.0288	0.0836	0.0366
		0.95	0.2005	0.1562	0.4819	0.2736

Table 4.2: MISE for **CL3**

Marginals	Parameters	α	SPQR	DVQR	LQR	BAQR
M1	R_1	0.5	0.0782	0.0577	0.0297	0.0539
		0.95	0.2811	0.2260	0.2944	0.2543
	R_2	0.5	0.0619	0.0472	0.0160	0.0389
		0.95	0.1978	0.1686	0.2002	0.1727
M2	R_1	0.5	1.9064	1.8895	1.8176	1.9393
		0.95	2.9410	1.3998	1.2937	0.9686
	R_2	0.5	0.9067	0.8850	0.9339	0.9488
		0.95	3.3226	3.2512	0.9113	1.6017

Table 4.3: MISE for **t5**

4.1 Result for CL3

From table Table 4.2, we can see that the MISE for semi parametric quantile regression and d-vine quantile regression perform better than their counterparts for all parameters and quantile level. We observe that the 50% quantile has better fit than 95% since the estimator for 50% quantile is more robust (Kraus and Czado, 2017). We also observe that when we change parameter from low dependence (δ_1) to high dependence (δ_2), the error drops. When the marginal distributions are skewed, the results are larger since the estimation of marginal distributions are more imprecise.

Compared to linear and boosting additive methods, we see that copula based quantile regression outperform them in this scenario. From the quantile plot of this three-dimensional Clayton copula, we see that it clearly fails the linearity assumption of linear quantile regression. Although boosting additive methods allows for modelling nonlinear parameters, it still under performs compared with copula based models. For detailed implementation, see B.3.

4.2 Result for t5

From Table 4.3, we see that the semi parametric and quantile regression has similar performance compared with linear and boosting additive methods at 50% quantile level. Copula based models was more accurate on non skewed normal and t distributions. For detailed implementation, see B.4

σ	Parameters	SPQR	DVQR	LQR	BAQR
σ_1	0.5	0.2689	0.2672	0.3648	0.1645
	0.95	0.5972	0.5296	0.7608	0.4339
σ_2	0.5	0.3052	0.2923	0.3757	0.2150
	0.95	0.3627	0.3598	0.4687	0.2951

Table 4.4: MISE for **N5**

4.3 Result for N5

This scenario simulates the non-monotonic dependence between the response and predictor variables. Dette et al. (2014) argues that non-monotonic relationship between the response and predictor variables cannot be modeled by a parametric copula. From Table 4.4, the simulation confirms the results where both semiparametric and D-vine models underperformed compared with boosting additive models. However, it still produces less error overall than linear model. For detailed implementation, see B.5.

Chapter 5

Application in Finance

For application of semiparametric and D-vine quantile regression, we will attempt to model the interdpendencies in the Australian equity market. We want to be able to forecast the median return of a stock as well as the risk of extreme quantiles. For this application, we consider a time period between 01/01/2006 - 01/01/2018, since the time span includes the Global Financial Crisis and allow us to consider scenario of different economic circumstances. We sample 7 equity from top 30 equity from the S&P/ASX 200 index, ignoring discarding the equities without complete price data within our time frame. The result is data set consisting 7 equities with 3032 daily observations of log returns calculated as:

$$R_t = 100 * (ln(P_t) - ln(P_{t-1})) \tag{5.0.1}$$

We will divide our data set into two sets: evaluation set and validation set. We will consider a roughly a quarter of the sample to be in the evaluation set, so that $n_{train} = 2274$ and $n_{eval} = 758$. We will also test the data set at the quantile level $\alpha = \{0.05, 0.5, 0.95\}$. We will consider two scenarios and measure the performance of each of the quantile regression with an out-of-sample test. The accuracy is measured using *averaged tick loss* L_α^m for quantile regression method m, defined as:

$$L_\alpha^m = \frac{1}{n_{eval}} \sum_{i=1}^{n_eval} \rho_\alpha(y^{(i)} - \widehat{q}_{\alpha,m}^{(i)}) \tag{5.0.2}$$

where $\rho_\alpha(y) = y(\alpha - I_{(y<0)})$ is the same loss function defined in Section 3.1.

5.1 Within industry

We now examine how the return is affected by the other companies belonging in the same industry. The first case we choose is the real estate industry with response Y:Cromwell Property Group Stapled Securities(CMW) and predictor X_1:Charter Hall Retail Real Estate Investment Trust(CQR) and X_2:Investa Office Fund Stapled Securities(IOF).

From Table 5.1 we can see the semiparametric and D-vine methods perform worse than linear and boosting additive methods for all quantile level. This is not surprising since companies in the same sector will usually have the same movement, and thus their quantile dependencies will be more linear like, thus linear and boosting additive models will have more advantage.

α	SPQR	DVQR	LQR	BAQR
0.05	0.2622	0.2470	0.1721	0.1336
0.50	0.4660	0.4582	0.3880	0.3814
0.95	0.1714	0.1686	0.1566	0.1182

Table 5.1: Tick loss for within industry

α	SPQR	DVQR	LQR	BAQR
0.05	1.2223	1.1886	1.4300	1.3767
0.50	0.7510	0.7405	0.7867	0.7715
0.95	0.2129	0.2122	0.3650	0.2732

Table 5.2: Tick loss for between industries

5.2 Between industries

First, we examine how the return of a company's share price is affected by other companies from other industry. We choose the response Y:Fletcher Building Limited(FBU) from the industry goods sector, X_1:Brickworks Limited(BKW),X_2:Newcrest Mining Limited(NCM) from basic material sector and X_3:Oil Search Limited(OSH) from energy sector.

From Table 5.2, we can see that in this case, copula based models outperformed their counter parts. The linear model cannot sufficiently model the dependencies between the variables since they are unlikely to exhibit linear dependencies. However, the vast majority of the copulae selected in R-vine and D-vine were t-copula, which isn't surprising for financial data. This might explain why linear model performed competitively in this case. Furthermore, moving away from the median to the tail quantiles linear model's error becomes apparent as it can't effectively describe the tail behaviour of t-copula. Overall, this real data application shows copula based semiparametric and D-vine quantile regression is a reasonable tool for estimating conditional quantile.

Chapter 6

Conclusion

We investigated two types of copula based quantile regression. We observe that copula based quantile regression are able to estimate and predict the quantiles of simulate data with monotonic quantiles, with comparable or better accuracy than other linear or non-linear quantile regressions.We have also seen the the usage of copula based quantile regression allow for relatively accurate estimation of risk of extreme return of an equity, using equities from other industries as predictors. Copula based quantile regressions are flexible tools that allows us to measure multiple non-linear dependencies accurately.

Furthermore, there are still options for further research. Currently, we are unable to model non-monotonic dependencies between a response and its predictor as pointed out by Dette et al. (2014). We only considered continuous margins, and the implementation of copula based quantile regression with discrete or a mix between discrete and continuous data can be investigated in the future.

Overall, using simulation and real data, we conclude that copula based regression models is a valid approach to estimating conditional quantiles.

Appendix A

A.1 Algorithm 1: R-vine selection

Algorithm 1: R-vine selection algorithm based on Kendall's tau

input: $(\hat{v}^{(i)})_{i=1,..,n}, (\hat{u}_j^{(i)})_{i=1,...,n,\ j=1,...,d}$

Calculate the empirical Kendall's tau $\hat{\tau}_{j,k}$ for all possible pair $j, k, 1 \leq j < k \leq n$.

Select the spanning tree that miximizes the sum of absolute empirical Kendall's taus, i.e:

$$\max \sum_{e=\{j,k|D\} \text{in spanning tree}} |\hat{\tau}_{j,k}|$$

For each edge $\{j, k\}$ in the selected spanning tree, select a copula and estimate the corresponding parameter(s). Then transform $\widehat{F}_{j|k}(x_{lj}|x_{lk})$ and $\widehat{F}_{k|j}(x_{lk}|x_{lj})$,$l = 1, ..., N$ using fitted copula $\widehat{C}_{jk}$.

for *i=2,...,n-1* **do**

Calculate the empirical Kendall's tau $\widehat{\tau}_{j,k|D}$ for all conditional variable pairs $\{j, k|D\}$ that can be part of tree T_i.

Among these edges, slect the spanning tree that maximizes the sum of absolute empirical Kendall's taus, i.e.

$$\max \sum_{e=\{j,k|D\} \text{in spanning tree}} |\hat{\tau}_{j,k}|$$

.

For each edge $\{j, k|D\}$ in the selected spanning tree, select a conditional copula and estimate the corresponding parameter(s). Then transform $\widehat{F}_{j|k\cup D}(x_{lj}|x_{lk}, \mathbf{x}_{lD})$ and $\widehat{F}_{k|j\cup D}(x_{lk}|x_{lj}, \mathbf{x}_{lD})$,$l = 1, ..., N$ using fitted copula $\widehat{C}_{jk|D}$.

end

A.2 Algorithm 2: D-vine sequential selection

A.3 List of companies for application

Algorithm 2: D-vine sequential selection

input: $(\hat{v}^{(i)})_{i=1,..,n}$, $(\hat{u}_j^{(i)})_{i=1,...,n,\ j=1,...,d}$
global.max.cll $= -\infty$;
$M = 1, ..., d$;
for *step* $= 1$ **to** d **do**
  if *step == 1* **then**
    for $i \in M$ **do**
      Estimate C_{V,U_i} the cll-optimal pair-copula;
    end
    loglik.max $= \min_{i \in M} loglik(VU_i)$
  else
    for $i \in M$ **do**
      Estimate $C_{V,U_i|U_{m_1},...,U_{m_{step-1}}}$ and $C_{U_{m_{step}},U_i|U_{m_1},...,U_{m_{step-1}}}$;
    end
    loglik.max $= \min_{i \in M} loglik(VU_i|U_{m_1}, ..., U_{m_{step-1}})$;
  end
  cll = cll+loglik.max;
  if *cll* $<$ *global.max.cll* **then**
    return D-vine $U - V_{m_1} - ... - V_{m_{step}}$
  end
  global.max.cll = cll ;
  M = M/m;
end
return D-vine $U - V_{m_1} - ... - V_{m_d}$

ASX code	Name	Industry
ALU	Altium Limited	Technology
BEN	Bendigo and Adelaide Bank Limited	Financial Services
BKL	Blackmores Limited	Consumer Defensive
BKW	Brickworks Limited	Basic Material
CMW	Cromwell Property Group Stapled Securities	Real Estate
CQR	Charter Hall Retail Real Estate Investment Trust	Real Estate
FBU	Fletcher Building Limited	Industrial Goods
FLT	FleetCor Technologies, Inc.	Consumer Cyclical
FXJ	Fairfax Media Limited	Consumer Cyclical
HVN	Harvey Norman Holdings Limited	Consumer Cyclical
IAG	Insurance Australia Group Limited	Financial Services
IOF	Investa Office Fund Stapled Securities	Real Estate
IRE	IRESS Limited	Technology
NCM	Newcrest Mining Limited	Basic Material
NVT	Navitas Limited	Consumer Defensive
OSH	Oil Search Limited	Energy
QAN	Qantas Airways Limited	Industrial
SRX	Sirtex Medical Limited	Healthcare
SUL	Super Retail Group Limited	Consumer Cyclical
TNE	Technology One Limited	Technology
TPM	TPG Telecom Limited	Communication

Table A.1: Source:Bloomberg

Appendix B

Code listing

B.1 SPqreg.r

```
library(VineCopula)
library(ks)

SPvine = function(data){
  n = nrow(data); d = ncol(data)-1

  U = matrix(nrow=n, ncol=d)
  for (i in 1:d){
    U[,i] = kcde(data[,i+1], eval.points=data[,i+1])$estimate
  }
  V = kcde(data[,1], eval.points=data[,1])$estimate

  RV = RVineStructureSelect(cbind(V,U), indeptest=TRUE)
  return(RV)
}

SPQR.quantile = function(newdata, obj, Y, tau){
  n.train = length(Y); d=ncol(newdata); n = nrow(newdata)
  U_mat = matrix(nrow=n, ncol=d)
  for (i in 1:d){
    U_mat[,i] = kcde(newdata[,i], eval.points=newdata[,i])$estimate
  }
  V = kcde(Y, eval.points=Y)$estimate

  obj_func = function(q, tau, y, pdf){
    n = length(y)
    rho = rep(NA,n)
    for (i in 1:n){
      if ((y[i]-q) < 0){rho[i] = (y[i]-q)*(tau-1)}else{rho[i] = (y[i]-q)*tau}
    }
    return(sum(rho*pdf))
  }

  est_quantile = rep(NA, n)
  for (i in 1:n){
    U = matrix(U_mat[i,], nrow=n.train, ncol=d, byrow=TRUE)
    cop_pdf = RVinePDF(cbind(V,U),RVM=obj)
    est_quantile[i] = optimize(obj_func, interval=c(min(Y), max(Y)), tau=tau, y=Y, pdf=cop_pdf)
  }
  return(est_quantile)
}
```

B.2 dvineqreg.r

```
library(VineCopula)
library(logspline)
library(ks)

DVine = function(data){
  d = dim(data)[2]-1; n = dim(data)[1]
  U = matrix(nrow=n, ncol=d)

  # PIT with continous kernel smoothing estimator
  for (i in 1:d){
```

```
  U[,i] = kcde(data[,i+1], eval.points=data[,i+1])$estimate
}
#V = kcde(data[,1], eval.points = data[, 1])$estimate

# PIT with logspline
ls = logspline(data[,1], error.action=2)
V = plogspline(data[,1], fit=ls)
# for (i in 1:d){
#   ls_x = logspline(data[,i+1], error.action=2)
#   if (!is.null(ls_x)){
#     U[,i] = plogspline(data[,i+1], fit=ls_x)
#   }
# }

# Allocate empty vector and matrix
cop.VU.cll = vector(length=d)
cop.VU = vector('list', length=d)
cop.UU = vector('list', length=d)
U1 = vector(length=n)
U2 = matrix(nrow=n, ncol=d)
order = vector(mode='integer', length=d)

# Initialize global variable
I = 1:d
global.max.cll = -Inf
cll = 0
for (step in 1:d){
  if (step == 1){
    loglik_max = -Inf
    for (i in 1:d){
      cop.VU_temp = BiCopSelect(V, U[,i], indeptest=TRUE)
      if (cop.VU_temp$logLik > loglik_max){
        loglik_max = cop.VU_temp$logLik
        l = i
        cop.VU[[step]] = cop.VU_temp
        U2[,1] = U[,i]
      }
    }
    U1 = V
  }else{
    U2_prev = U2; U1_prev = U1

    U1 = BiCopHfunc2(U1_prev, U2_prev[,step-1], obj=cop.VU[[step-1]])
    U2_temp = matrix(nrow=n, ncol=d)

    loglik_max = -Inf
    cop.UU_temp = vector('list', length=d)
    for (i in I){
      h1 = U[, i]; h2 = U[,l]
      for (k in 2:step){
        cop.UU_temp[[k]] = BiCopSelect(h1, h2, indeptest=TRUE)
        U2_temp[, k] = BiCopHfunc1(h1, h2, cop.UU_temp[[k]])
        h1 = BiCopHfunc2(h1, h2, obj=cop.UU_temp[[k]])
        h2 = U2_prev[, k]
      }
      cop.VU_temp = BiCopSelect(U1, h1, indeptest=TRUE)
      if (cop.VU_temp$logLik > loglik_max){
        loglik_max = cop.VU_temp$logLik
        U2 = U2_temp
        cop.UU[[step]] = cop.UU_temp
        cop.VU[[step]] = cop.VU_temp
        l = i
      }
    }
  }

  cll = cll+loglik_max
  if (cll <= global.max.cll){
    order = order[1:step-1]
    return(list(ord = order, copVU = cop.VU, copUU = cop.UU, logSpline=ls))
  }
```

```
    global.max.cll = cll
    I = I[! I %in% l]
    order[step] = l
  }
  return(list(ord = order, copVU = cop.VU, copUU = cop.UU, logSpline=ls))
}


DVQR.quantile = function(obj, newdata, tau){
  order = obj$ord; cop.VU = obj$copVU; cop.UU = obj$copUU

  d = length(order); n = dim(newdata)[1]
  U = matrix(nrow=n, ncol=d); U2 = matrix(nrow=n, ncol=d)

  # Probability integral transform
  j=1
  for (i in order){
    U[,j] = kcde(newdata[,i], eval.points=newdata[,i])$estimate
    j = j+1
  }

  # Calculate conditioned distribution
  UU = matrix(nrow=n, ncol=d); U2 = matrix(nrow=n, ncol=d)
  for (step in 1:d){
    if (step == 1){
      UU[,1] = U[,1]
      U2[,step] = UU[,1]
    }else{
      UU_prev = UU
      h1 = U[, step]; h2 = U[,step-1]; UU[,1] = U[,step-1]
      for (k in 2:step){
        UU[, k] = BiCopHfunc1(h1, h2, obj=cop.UU[[step]][[k]])
        h1 = BiCopHfunc2(h1, h2, obj=cop.UU[[step]][[k]])
        h2 = UU_prev[, k]
      }
      U2[,step] = h1
    }
  }

  # Calculate conditional quantile
  cqf = rep(tau,n)
  for (step in d:1){
    cqf = BiCopHinv2(cqf, U2[,step], obj=cop.VU[[step]])
  }
  quantile = qlogspline(cqf, fit=obj$logSpline)
  return(quantile)
}
```

B.3 C3simulation.r

```
# SIMULATION: SCENARIO C3

rm(list = ls()) # clear workspace variable

library(parallel)
library(doSNOW)
library(foreach)

num_core = detectCores()
cl = makeCluster(num_core)
registerDoSNOW(cl)

# Simulation parameter
R = 100
d = 2
D = d+1

n.train = 300
alpha = 0.95 # quantile level
```

```
# True quantile function
quantile_true = function(data, tau, dist, par, theta){
  env = new.env()
  u1 = do.call(paste(c('p', dist[2]), collapse=''), c(list(data[,1]), par[[2]]), envir=env)
  u2 = do.call(paste(c('p', dist[3]), collapse=''), c(list(data[,2]), par[[3]]), envir=env)
  prob = ((tau^(-theta/(1+2*theta))-1)*(u1^(-theta)+u2^(-theta)-1)+1)^(-1/theta)
  return(do.call(paste(c('q', dist[1]), collapse=''), c(list(p=prob), par[[1]]), envir=env))
}

# Scenario parameters
#delta = 0.86;
delta = 4.67
#dist = c('norm', 't', 'norm')
#para = list(list(mean=0, sd=1), list(df=4), list(mean=1, sd=sqrt(4)))
dist = c('st', 'sn', 'st')
para = list(list(xi=0, omega=1, alpha=2, nu=4), list(xi=-2, omega=sqrt(0.5), alpha=3),
            list(xi=1, omega=sqrt(2), alpha=5, nu=3))

clusterEvalQ(cl, list(library(sn), library(quantreg), library(mboost), library(np), library(cop
clusterExport(cl, list('R', 'd', 'D', 'n.train', 'alpha', 'quantile_true', 'delta', 'dist', 'pa

start_time = proc.time()
ISE <- foreach(r=1:R) %dopar% {
  # Create copula object
  copula = claytonCopula(param=delta,dim=D)
  mv <- mvdc(copula, margins=dist,paramMargins=para)

  # Genearte training data
  sim.train = data.frame(rMvdc(n.train, mv))
  # Genearte evaulation data
  n.eval = n.train/2
  sim.eval = data.frame(rMvdc(n.eval, mv))

  # Linear quanitle regression (LQR)
  LQR = rq(X1~., tau=alpha, data=sim.train)

  # Boosting additive (BAQR)
  it = 100
  bc = boost_control(mstop = it, nu=0.25, trace = TRUE, risk = "oob")
  BAQR = gamboost(X1~., data=sim.train, control=bc, family=QuantReg(tau=alpha))

  # Semiparametric quantile regression (SPQR)
  source('R/SPqreg.R')
  SPQR = SPvine(sim.train)

  # D-vine quantile regression (DVQR)
  source("R/dvineqreg.R")
  DVQR = Dvine(sim.train)

  # Predict quanile/calculate true quantile
  q.LQR = predict.rq(LQR, newdata=sim.eval[,2:D])
  q.BAQR = predict(BAQR, newdata=sim.eval[,2:D])
  q.DVQR = DVQR.quantile(obj=DVQR, newdata=sim.eval[,2:D], tau=alpha)
  q.SPQR = SPQR.quantile(obj=SPQR, newdata=sim.eval[,2:D], Y=sim.train[,1], tau=alpha)
  q = quantile_true(sim.eval[,2:D], tau=alpha, dist=dist, par=para, theta=delta)

  # Integrated square error
  ISE.LQR = sum((q-q.LQR)^2)/n.eval
  ISE.BAQR = sum((q-q.BAQR)^2)/n.eval
  ISE.DVQR = sum((q-q.DVQR)^2)/n.eval
  ISE.SPQR = sum((q-q.SPQR)^2)/n.eval
  output = list(ISE.SPQR, ISE.DVQR, ISE.LQR, ISE.BAQR)
  output
}

# Mean Integrated Square Error
MISE.SPQR = 0; MISE.DVQR = 0; MISE.LQR = 0; MISE.BAQR = 0
for (i in 1:R){
  MISE.SPQR = MISE.SPQR+ISE[[i]][[1]]/R
  MISE.DVQR =  MISE.DVQR+ISE[[i]][[2]]/R
  MISE.LQR =  MISE.LQR+ISE[[i]][[3]]/R
```

```
  MISE.BAQR = MISE.BAQR+ISE[[i]][[4]]/R
}
MISE.SPQR
MISE.DVQR
MISE.LQR
MISE.BAQR

# Stop parallel
stopCluster(cl)
proc.time()-start_time
```

B.4 t5simulation.r

```
# SIMULATION; SCENARIO t5

rm(list = ls()) # clear workspace variable

library(parallel)
library(doSNOW)
library(foreach)

num_core = detectCores()
cl = makeCluster(num_core)
registerDoSNOW(cl)

# Simulation parameter
R = 50
d = 4; D = d+1
n.train = 300
alpha = 0.95 # quantile level

# True quantile function
quantile_true = function(data, tau, df, corr, sd, mu){
  n = dim(data)[1]
  d = dim(data)[2]
  D = d+1
  df_c = df+d

  S = diag(sd)%*%corr%*%diag(sd)
  S_XX = as.matrix(S[2:D,2:D])
  S_YX = as.matrix(S[1,2:D])
  x_bar = as.matrix(sweep(data,2,mu[2:D]))

  mu_c = rep(NA, n); var_c = rep(NA, n)
  for (i in 1:n){
    mu_c[i] = mu[1]+t(S_YX)%*%solve(S_XX)%*%x_bar[i,]
    var_c[i] = (df+t(x_bar[i,])%*%solve(S_XX)%*%x_bar[i,])*
                (S[1,1]-t(S_YX)%*%solve(S_XX)%*%S_YX)/df_c
  }
  return(qt.scaled(tau, df=df_c, mean=mu_c, sd=sqrt(var_c)))
}


# Scenario paramters
v = 3 # df
#R_vec = c(0.6, 0.5, 0.5, 0.4, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5)
R_vec = c(0.27, 0.74, 0.72, 0.41, 0.28, 0.29, 0.27, 0.74, 0.42, 0.40)
#dist = c('norm', 't', 'norm', 't', 'norm')
#para = list(list(mean=0), list(df=4), list(mean=1,sd=2), list(df=4), list(mean=1,sd=2))
dist = c('st', 'sn', 'st', 'sn', 'st')
para = list(list(alpha=2,nu=4), list(xi=-2,omega=sqrt(0.5),alpha=3), list(xi=1,omega=sqrt(2),al
            list(xi=-2,omega=sqrt(0.5),alpha=3), list(xi=1,omega=sqrt(2),alpha=5,nu=3))
mu = c(0, 0, 1, 0, 1)
sd = c(1, sqrt(2), 2, sqrt(2),2)

clusterEvalQ(cl, list(library(sn), library(quantreg), library(mboost), library(np), library(cop
clusterExport(cl, list('R', 'n.train', 'alpha', 'd', 'D', 'quantile_true', 'dist', 'para', 'mu'

start_time = proc.time()
ISE <- foreach(r=1:R) %dopar% {
```

```
  copula = tCopula(R_vec, dim=D, dispstr='un', df=v)
  mv <- mvdc(copula, margins=dist, paramMargins=para)

  # Genearte training data
  sim.train = data.frame(rMvdc(n.train, mv))

  # Genearte evaulation data
  n.eval = n.train/2
  sim.eval = data.frame(rMvdc(n.eval, mv))

  # Linear quanitle regression (LQR)
  LQR <- rq(X1~., tau=alpha, data=sim.train)

  # Boosting additive (BAQR)
  it <- 100
  bc <- boost_control(mstop = it, nu=0.25, trace = TRUE, risk = "oob")
  BAQR <- gamboost(X1~., data=sim.train, control=bc, family=QuantReg(tau=alpha))

  # Semiparametric quantile regression (SPQR)
  source('R/SPqreg.R')
  SPQR = SPvine(sim.train)

  # D-vine quantile regression (DVQR)
  source("R/dvineqreg.R")
  DVQR = Dvine(sim.train)

  # Predict quanile/calculate true quantile
  q.LQR = predict.rq(LQR, newdata=sim.eval[,2:D])
  q.BAQR = predict(BAQR, newdata=sim.eval[,2:D])
  #q.DVQR = DVQR.quantile(DVQR, newdata=sim.eval[,2:D], tau=alpha)
  q.DVQR = DVQR.quantile(obj=DVQR, newdata=sim.eval[,2:D], tau=alpha)
  q.SPQR = SPQR.quantile(obj=SPQR, newdata=sim.eval[,2:D], Y=sim.train[,1], tau=alpha)
  q = quantile_true(data=sim.eval[,2:D], tau=alpha, df=v, corr=getSigma(copula), sd=sd, mu=mu)

  # Integrated square error
  ISE.LQR = sum((q-q.LQR)^2)/n.eval
  ISE.BAQR = sum((q-q.BAQR)^2)/n.eval
  ISE.DVQR = sum((q-q.DVQR)^2)/n.eval
  ISE.SPQR = sum((q-q.SPQR)^2)/n.eval
  output = list(ISE.SPQR, ISE.DVQR, ISE.LQR, ISE.BAQR)
  output
}

# Mean Integrated Square Error
MISE.SPQR = 0; MISE.DVQR = 0; MISE.LQR = 0; MISE.BAQR = 0
for (i in 1:R){
  MISE.SPQR = MISE.SPQR+ISE[[i]][[1]]/R
  MISE.DVQR =  MISE.DVQR+ISE[[i]][[2]]/R
  MISE.LQR =  MISE.LQR+ISE[[i]][[3]]/R
  MISE.BAQR = MISE.BAQR+ISE[[i]][[4]]/R
}
MISE.SPQR
MISE.DVQR
MISE.LQR
MISE.BAQR

# Stop parallel
stopCluster(cl)
proc.time()-start_time
```

B.5 N5simulation.r

```
# SIMULATION: SCENARIO N5

rm(list = ls()) # clear workspace variable

library(parallel)
library(doSNOW)
library(foreach)
```

```
num_core = detectCores()
cl = makeCluster(num_core)
registerDoSNOW(cl)

# Simulation parameters
R = 100
d = 4
D = d+1

n.train = 300
alpha = 0.95 # quantile level

# True quantile function
quantile_true = function(newdata, tau, func, sd){
  return(qnorm(tau, mean=func(newdata), sd=sd))
}

# Scenario paramters
sigma = 1
fY = function(x){
  y = sqrt(abs(2*x[,1]-x[,2]+0.5))+(-0.5*x[,3]+1)*(0.1*x[,4]^3)
  return(y)
}
S_X = matrix(NA, 4, 4)
for (i in 1:d){
  for (j in 1:d){
    S_X[i,j] = 0.5^abs(i-j)
  }
}
mu_X = rep(0, 4)

clusterEvalQ(cl, list(library(sn), library(quantreg), library(mboost), library(MASS)))
clusterExport(cl, list('n.train', 'alpha', 'quantile_true', 'S_X', 'mu_X', 'd', 'D','fY', 'sigm

start_time = proc.time()
ISE <- foreach(r=1:R) %dopar% {
  # Genearte training data
  X.train = mvrnorm(n.train, mu_X, S_X)
  Y.train = fY(X.train)+sigma*rnorm(n.train)
  sim.train = data.frame(cbind(Y.train, X.train))

  # Genearte evaulation data
  n.eval = n.train/2
  X.eval = mvrnorm(n.eval, mu_X, S_X)
  Y.eval = fY(X.eval)+sigma*rnorm(n.eval)
  sim.eval = data.frame(cbind(Y.eval, X.eval))

  # Linear quanitle regression (LQR)
  LQR = rq(Y.train~., tau=alpha, data=sim.train)

  # Boosting additive (BAQR)
  it = 100
  bc = boost_control(mstop = it, nu=0.25, trace = TRUE, risk = "oob")
  BAQR = gamboost(Y.train~., data=sim.train, control=bc, family=QuantReg(tau=alpha))

  # Semiparametric quantile regression (SPQR)
  source('R/SPqreg.R')
  SPQR = SPvine(sim.train)

  # D-vine quantile regression (DVQR)
  source("R/dvineqreg.R")
  DVQR = Dvine(sim.train)

  # Predict quanile/calculate true quantile
  q.LQR = predict.rq(LQR, newdata=sim.eval[,2:D])
  q.BAQR = predict(BAQR, newdata=sim.eval[,2:D])
  #q.DVQR = DVQR.quantile(DVQR, newdata=sim.eval[,2:D], tau=alpha)
  q.DVQR = DVQR.quantile(obj=DVQR, newdata=sim.eval[,2:D], tau=alpha)
  q.SPQR = SPQR.quantile(obj=SPQR, newdata=sim.eval[,2:D], Y=sim.train[,1], tau=alpha)
  q = quantile_true(newdata=sim.eval[,2:D], tau=alpha, func=fY, sd=sigma)
```

```
  # Integrated square error
  ISE.LQR = sum((q-q.LQR)^2)/n.eval
  ISE.BAQR = sum((q-q.BAQR)^2)/n.eval
  ISE.DVQR = sum((q-q.DVQR)^2)/n.eval
  ISE.SPQR = sum((q-q.SPQR)^2)/n.eval
  output = list(ISE.SPQR, ISE.DVQR, ISE.LQR, ISE.BAQR)
  output
}

# Mean Integrated Square Error
MISE.SPQR = 0; MISE.DVQR = 0; MISE.LQR = 0; MISE.BAQR = 0
for (i in 1:R){
  MISE.SPQR = MISE.SPQR+ISE[[i]][[1]]/R
  MISE.DVQR = MISE.DVQR+ISE[[i]][[2]]/R
  MISE.LQR = MISE.LQR+ISE[[i]][[3]]/R
  MISE.BAQR = MISE.BAQR+ISE[[i]][[4]]/R
}
MISE.SPQR
MISE.DVQR
MISE.LQR
MISE.BAQR

# Stop parallel
stopCluster(cl)
proc.time()-start_time
```

References

Aas, K., Czado, C., Frigessi, A. and Bakken, H. (2009), 'Pair-copula constructions of multiple dependence', *Insurance: Mathematics and economics* **44**(2), 182–198.

Bedford, T. and Cooke, R. M. (2002), 'Vines: A new graphical model for dependent random variables', *Annals of Statistics* pp. 1031–1068.

Buchinsky, M. (1998), 'Recent advances in quantile regression models: a practical guideline for empirical research', *Journal of human resources* pp. 88–126.

Dette, H., Van Hecke, R. and Volgushev, S. (2014), 'Some comments on copula-based regression', *Journal of the American Statistical Association* **109**(507), 1319–1324.

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

Fang, H.-B., Fang, K.-T. and Kotz, S. (2002), 'The meta-elliptical distributions with given marginals', *Journal of Multivariate Analysis* **82**(1), 1–16.

Hofmann, M. and Czado, C. (2010), 'Assessing the var of a portfolio using d-vine copula based multivariate garch models', *Preprint* .

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

Koenker, R. (2011), 'Additive models for quantile regression: Model selection and confidence bandaids', *Brazilian Journal of Probability and Statistics* **25**(3), 239–262.

Koenker, R. and Bassett Jr, G. (1978), 'Regression quantiles', *Econometrica: journal of the Econometric Society* pp. 33–50.

Kraus, D. and Czado, C. (2017), 'D-vine copula based quantile regression', *Computational Statistics & Data Analysis* **110**, 1–18.

Kurowicka, D. and Cooke, R. (n.d.), Non-parametric continuous bayesian belief nets with expert judgement, *in* 'Probabilistic Safety Assessment and Management', Springer, pp. 2784–2790.

Li, D. X. (1999), 'On default correlation: A copula function approach'.

McNeil, A. J. (2008), 'Sampling nested archimedean copulas', *Journal of Statistical Computation and Simulation* **78**(6), 567–581.

Noh, H., Ghouch, A. E. and Van Keilegom, I. (2014), 'Semiparametric conditional quantile estimation through copula-based multivariate models', *Journal of Business & Economic Statistics* **33**(2), 167–178.

Parzen, E. (1962), 'On estimation of a probability density function and mode', *The annals of mathematical statistics* **33**(3), 1065–1076.

Salmon, F. (2012), 'The formula that killed wall street', *Significance* **9**(1), 16–20.

Sklar, A. (1973), 'Random variables, joint distribution functions, and copulas', *Kybernetika* **9**(6), (449)–460.