Factor Copulas : Modelling dependence in Higher Dimensions

End Term Report, Second Semester 2016-17

10th June 2017

Kushagra Gupta (2012MT50599)

Guided by
Dr. Aparna Mehra

Chapter 1

Introduction

1.1 Motivation/Problem at hand

When dealing with a financial market, we often come across problems where we have to deal with a high number of interdependent assets. We need to model the interdependence of assets for better risk evaluation among other things. One of the most famous examples of consequences of poor modelling of interdependence is the 2008 crisis. The companies had high negative tail dependence and failure of some firms triggered failures of others.

The method of copula modelling is one of the most frequently used method to model the interdependence. The method has a lot of applications especially in the area of credit risk where it is where important to model the negative tail dependence.

Copulas are of various types and vine-copulas are one of the most frequently used models for fitting copulas on large number of assets. We'll see in the coming sections that the process of Copula fitting is a computationally intensive process. It is computationally not feasible to model the interdependence of more than 50 assets at once, which has been proven to be a major setback. The typical market index could contain anywhere between 50-300 stocks, so forming a portfolio using copulas is a difficult task. Same goes for number of assets in CDOs. So we are not able to model the market as a whole but rather model only a representative subset which is sometimes not a good approximation to use.

One of the newer improvement is the Factor Copulas where the assets are assumed to be dependent on some common factors and given these common factors the assets are assumed to be independent. These methods are fast and authors claim to have modelled 100 assets at once.Through the means of this project, we would like to investigate the validity of claims, compare them with existing methods, analyze the tradeoff between the amount of information loss vs time saved, implement the method as a R library and come up with further enhancement of speed.

We found 2 independent and fairly distinct approach to Factor Copulas. The first one is *Factor copula models for multivariate data* by Pavel Krupskii, Harry Joe and other one is *Modelling Dependence in High Dimensions with Factor Copulas* by Oh and Patton. We'll be going to implement the former. Both the papers present the similar idea through different approaches

1.2 Mathematical Prelimnaries

1.2.1 Copulas

A copula is a multivariate probability distribution for which the marginal probability distribution of each variable is uniform. Copulas are used to describe the dependence between random variables. Copulas have been used widely in quantitative finance to model and minimize tail risk and portfolio optimization applications.

Definition 1.1. Given n random variables $X_1, X_2, ..., X_n$ with cdf $F_1, F_2, ..., F_n$ respectively, the copula C is defined as the joint cumulative distribution of $F_1(X_1), F_2(X_2), \ldots, F_n(X_n)$.

Given n random variables $X_1, X_2, ..., X_n$, we apply the probability integral transform to each component to achieve the random vector with uniformly distributed marginals.

$$(U_1, U_2, ..., U_n) = (F_1(X_1), F_2(X_2), ..., F_n(X_n))$$

The copula of $(X_1, X_2, \ldots, X_d)$ is defined as the joint cumulative distribution function of $(U_1, U_2, \ldots, U_d)$

$$\begin{aligned} C(u_1, u_2, \ldots, u_n) &= P[U_1 \leq u_1, U_2 \leq u_2, \ldots, U_n \leq u_n] \\ &= P[F_1(X_1) \leq u_1, F_2(X_2) \leq u_2, \ldots, F_n(X_n) \leq u_n] \\ &= P[X_1 \leq F_1^{-1}(u_1), X_2 \leq F_2^{-1}(u_2), \ldots, X_n \leq F_n^{-1}(u_n)] \end{aligned}$$

Differentiating the above w.r.t. every factor we get the cumulative joint distribution:

$$c(u_1, u_2, \ldots, u_n) = c(F_1(x_1), \ldots, F_d(x_n)) \prod_{i=1}^{n} f_i(x_i)$$

1.2.2 Sklar's Theorem

Sklar's theorem provides the theoretical foundation for the application of copulas. Sklar's theorem states that every multivariate cumulative distribution function

$$H(x_1, \ldots, x_d) = P[X_1 \leq x_1, \ldots, X_d \leq x_d]$$

of a random vector $(X_1, X_2, \dots, X_d)$ can be expressed in terms of its marginals $F_i(x) = P[X_i \leq x]$ and a copula C. Indeed:

$$H(x_1, \dots, x_d) = C(F_1(x_1), \dots, F_d(x_d))$$

In case that the multivariate distribution has a density h, and this is available, it holds further that

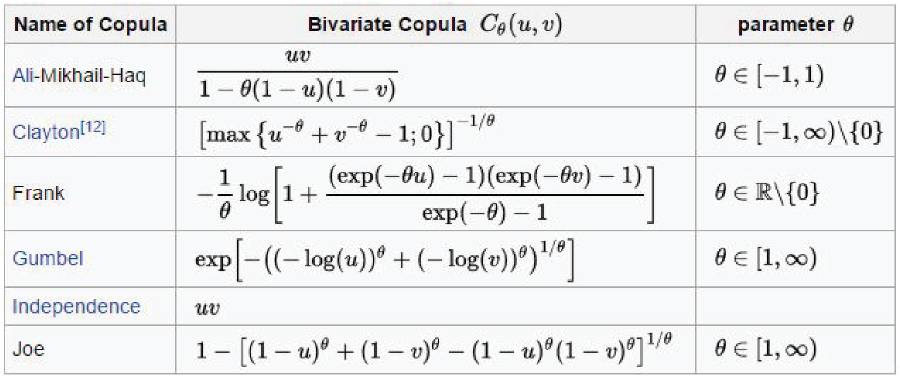
$$h(x_1, \dots x_d) = c(F_1(x_1), \dots F_d(x_d)) \cdot f_1(x_1) \cdot \dots \cdot f_d(x_d),$$

where c is the density of the copula.
The theorem also states that that the copula is unique if the marginals F_i are continuous.

Conversely we get: given a copula $C : [0,1]^d \to [0,1]$ and margins $F_i(x)$ then $C(F_1(x_1), \dots, F_d(x_d))$ defines a d-dimensional cumulative distribution function.

1.2.3 Some common copula families

Figure 1.1: Some common copula families



Name of Copula	Bivariate Copula $C_\theta(u,v)$	parameter θ
Ali-Mikhail-Haq	$\dfrac{uv}{1-\theta(1-u)(1-v)}$	$\theta \in [-1,1)$
Clayton[12]	$\left[\max\left\{u^{-\theta}+v^{-\theta}-1;0\right\}\right]^{-1/\theta}$	$\theta \in [-1,\infty)\backslash\{0\}$
Frank	$-\dfrac{1}{\theta}\log\left[1+\dfrac{(\exp(-\theta u)-1)(\exp(-\theta v)-1)}{\exp(-\theta)-1}\right]$	$\theta \in \mathbb{R}\backslash\{0\}$
Gumbel	$\exp\left[-\left((-\log(u))^\theta+(-\log(v))^\theta\right)^{1/\theta}\right]$	$\theta \in [1,\infty)$
Independence	uv	
Joe	$1-\left[(1-u)^\theta+(1-v)^\theta-(1-u)^\theta(1-v)^\theta\right]^{1/\theta}$	$\theta \in [1,\infty)$

1.2.4 Vine Copulas

The number of parametric multivariate copula families with flexible dependence is limited, there are many parametric families of bivariate copulas. Moreover, standard multivariate copulas do not allow for different dependency structures between pairs of variables and become inflexible in high dimensions. Vines leverage from bivariate copulas and enable extensions to arbitrary dimensions. This makes them very useful when dealing with high dimensional data.

Definition 1.2. A vine V on n variables is a nested set of connected trees where the edges in the first tree are the nodes of the second tree, the edges of the second tree are the nodes of the third tree, etc.

We can represent a density $f(x_1, \ldots, x_d)$ as a product of pair copula densities and marginal densities. For example when $d = 3$,

$$f(x_1, x_2, x_3) = f_{3|12}(x_3|x_1, x_2)f_{2|1}(x_2|x_1)f_1(x_1) \tag{1.1}$$
$$f_{2|1}(x_2|x_1) = c_{12}(F_1(x_1), F_2(x_2))f_2(x_2) \tag{1.2}$$
$$f_{3|12}(x_3|x_1, x_2) = c_{13|2}(F_{1|2}(x_1|x_2), F_{3|2}(x_3|x_2))f_{3|2}(x_3|x_2) \tag{1.3}$$
$$f_{3|2}(x_3|x_2) = c_{23}(F_2(x_2), F_3(x_3))f_3(x_3) \tag{1.4}$$

This gives us

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

So in general, we can write:

$$f(x_1, \ldots, x_d) = \prod_{j=1}^{d-1}\prod_{i=1}^{d-1} c_{i,(i+j)|(i+1),\ldots,(i_j-1)} \cdot \prod_{k=1}^{d} f_k(x_k)$$

where,

$$c_{i,j|i_1,\ldots,i_k} := c_{i,j|i_1,\ldots,i_k}(F(x_i|x_{i_1}, \ldots, x_{i_k}), F(x_j|x_{i_1}, \ldots, x_{i_k}))$$

So we are able to write the joint distribution function in terms of bivariate copulas. The above formula is represented using vine structure.

Some fo the common vine copulas include C-vine copulas where each tree has a unique node that is connected to all other nodes and D-vine copulas where each tree is a path.

In vine copulas we have 3 things to determine: the structure, copula families, copula parameters. Determining these things is computationally intensive and proves to be bottleneck when dealing with large number of assets. An improvement is truncated vine copulas where the vines are reduced only to a given depth.

1.3 Factor Copula Models

Given a random vector $(X_1, X_2, ..., X_d)$, factor copula model assumes that given a few latent factors, the given random variables are independent. The number of bivariate (conditional or marginal) copulas used in the vine copula construction is $d(d-1)/2$ for d variables, so typically vine copulas involve $O(d^2)$ number of parameters.

An important advantage of factor models is that they can be nicely interpreted. In case of stocks in a common sector, the current state of this sector can affect all of their change of prices, but the sector index, if measured, might not contain all of the latent information that explains the dependence. Similarly for market data, the state of the economy as a whole can determine the latent dependence structure. The state variables are aggregated from many exogenous variables (such as interest rate, refinancing rate, political instabilities, etc.) and cannot be easily measured, therefore factor copula models based on latent variables might be a good choice.

Let $U = (U_1, U_2, \dots, U_d)$ be d uniformly distributed random variables. (They can be attained by applying probability integral transform to given random variables). The joint cdf of the vector U is then given by $C(u_1, \dots, u_d)$ where C is a d-dimensional copula. In the p-factor copula model, $U_1, \dots, U_d$ are assumed to be conditionally independent given p latent variables $V_1, \dots, V_p$. Without loss of generality, we can assume V_i are independent and identically distributed (i.i.d.) $U(0, 1)$. Let the conditional cdf of U_j given $V_1, \dots, V_p$ be denoted by $F_{j|V_1,\dots,V_p}$. Then,

$$C(u_1, \dots, u_d) = \int_{[0,1]^p} \prod_{j=1}^{d} F_{j|V_1,\dots,V_p}(u_j|v_1, \dots, v_p)dv_1 \dots dv_p \tag{1.5}$$

Now $F_{j|V_1,\dots,V_p}(u_j|v_1, \dots, v_p)$ are expressed appropriately in terms of a sequence of bivariate copulas that link the observed variables U_j to the latent variables V_k. So now instead of $d(d-1)/2$ bivariate copulas, we'll only have to fit $p \cdot d$ copulas. If p is small as compared to d, we have reduced the complexity form $O(d^2)$ to $O(d)$.

1.3.1 1-factor Model

Let number of factors $p = 1$. For $j = 1, \dots, d$, denote the joint cdf and density of (U_j, V_1) by C_{j,V_1} and c_{j,V_1} respectively. Since U_1, V_j are $U(0, 1)$ random variables, then $F_{j|V_1}$ is just a partial derivative of the copula C_{j,V_1} with respect to the second argument. That is, $F_{j|V_1}(u_j|v_1) = C_{j|V_1}(u_j|v_1) = \partial C_{j,V_1}(u_j, v_1)/\partial v_1$. So using this, we get

$$C(u_1, \dots, u_d) = \int_0^1 \prod_{j=1}^{d} F_{j|V_1}(u_j|v_1)dv_1 \tag{1.6}$$

$$= \int_0^1 \prod_{j=1}^{d} C_{j|V_1}(u_j|v_1)dv_1 \tag{1.7}$$

We know,

$$\frac{\partial}{\partial u} C_{j|V_1}(u|v_1) = \frac{\partial^2}{\partial u} C_{j,V_1}(u, v_1) \tag{1.8}$$

$$= c_{j,V_1}(u, v_1) \tag{1.9}$$

So we get,

$$c(u_1, \ldots, u_d) = \frac{\partial^d C(u_1, \ldots, u_d)}{\partial u_1 \ldots \partial u_d} \tag{1.10}$$

$$= \int_0^1 \prod_{j=1}^{d} c_{j,V_1}(u_j, v_1) dv_1 \tag{1.11}$$

So we need to fit a total of d bivariate copulas.

1.3.2 2-factor Model

We get results similar to the results in 2 factor models

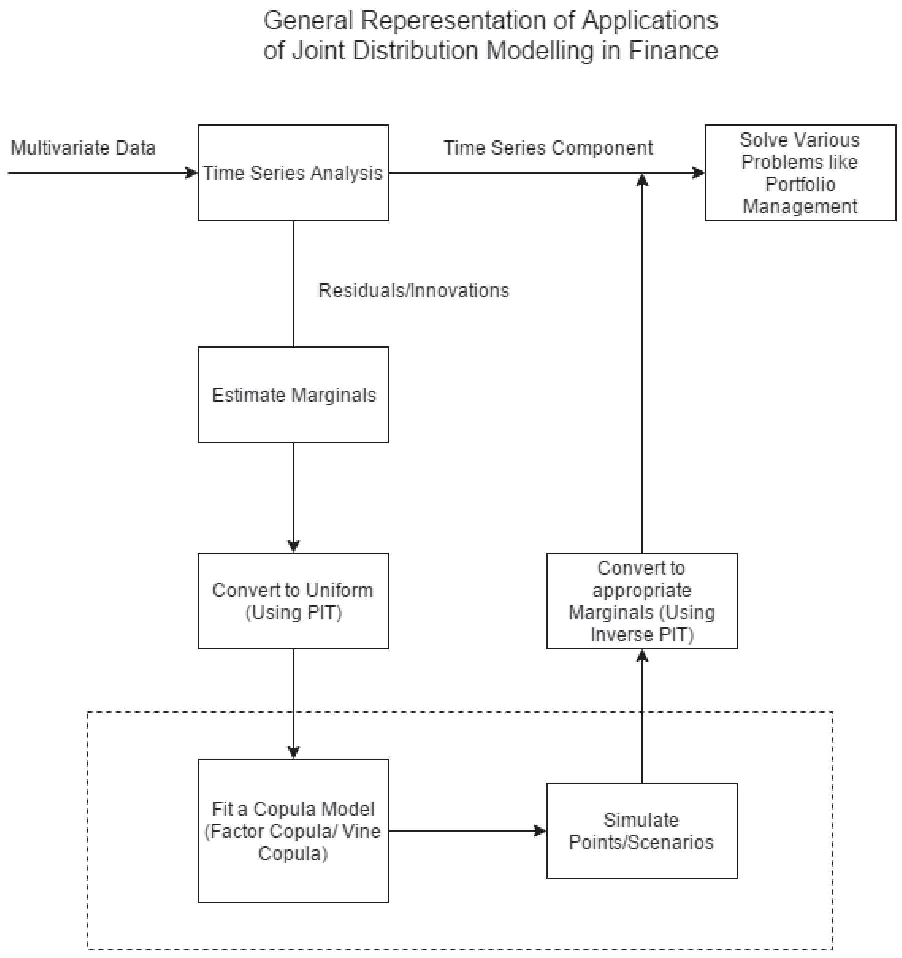
$$c(u_1, \ldots, u_d) = \int_0^1 \int_0^1 \prod_{j=1}^{d} \{c_{j,V_2;V_1}(C_{j|V_1}(u_j|v_1), v_2) \cdot c_{j,v_1}(u_j, v_1)\} dv_1 dv_2 \tag{1.12}$$

Here we need to fit $2 \cdot d$ copulas.

1.3.3 Computational Details

A parametric copula family is used for each linking copula, and θ is the vector of all of the dependence parameters. Appropriate choices of the bivariate linking copulas can be made based on bivariate normal scores plots for the data and the properties. With parameters of univariate margins estimated first followed by conversion to $U(0,1)$ data, the parameter vector θ of the joint density $c(\cdot;\theta)$ can then be estimated using maximum likelihood. That is, the parameters of uniform margins are estimated at the first step and dependence parameters at the second step with parameters of the univariate margins fixed at the estimates obtained from the first step. The two-step estimation procedure significantly simplifies the computation process. The second stage optimization of the log-likelihood of the copula model is done via a quasi-Newton or modified Newton-Raphson numerical method and this outputs a Hessian matrix from which standard errors (conditional on the first step) are obtained.
For many commonly used parametric copula families, the density $c_{j,V_1}(u_{ij}, v_1; \theta)$ approaches infinity as $(u_{ij}, v_1) \to (0,0)$ or $(u_{ij}, v_1) \to (1,1)$, therefore the integrand could be unbounded. So we can transform the variable of integration with $v_1 \to \phi(t1)$, where ϕ is the standard normal cdf and then use Gauss-Hermite or Gauss-Legendre quadrature.

Figure 1.2: Flow Chart of Applications


General Reperesentation of Applications
of Joint Distribution Modelling in Finance
Multivariate Data
Time Series Analysis
Time Series Component
Solve Various Problems like Portfolio Management
Residuals/Innovations
Estimate Marginals
Convert to Uniform (Using PIT)
Convert to appropriate Marginals (Using Inverse PIT)
Fit a Copula Model (Factor Copula/ Vine Copula)
Simulate Points/Scenarios

1.4 Project Target

The following points were decided as the project target:

1. Implement the Factor Copula model (Both Parameter Learning and Simulating points). Parameter Learning is discussed in Chapter 2 while simulation is discussed in Chapter 3.

2. Compare Factor Copulas to existing methods

3. Use a real-life application and compare the results. (Chapter 4)

4. Verify claims that around 100 assets could be processed easily.

Chapter 2

Learning the Model Parameters

In this chapter If would like to discuss the computational process in detail and see a small working examle of the performance. It was one of the target of project to publish a R library for fitting Factor Copula Models.

2.1 Overview of the Process

For fitting factor copula model, we need to perform the following steps:

1. Convert the given marginals to uniform distribution.
2. Select the number of hidden factors.
3. Use numerical methods to calculate integration in likelihood formula and thus calculate overall likelihood of the data.
4. Use gradient descent or quasi-Newton methods to get maximum likelihood estimation of the parameters.

2.2 Maximum Likelihood Estimation

We would use maximum likelihood estimation to estimate the model parameters. We'll experiment with 2 methods. Firstly we'll see the gradient descent method and then the BFGS Method. Here I'll present basic theories rearding these methods. In the following section, all formulas would be presented assuming that the linking copulas have chosen to be of Gumbel Family.

2.2.1 Likelihood Formula

1-Factor Model

Likelihood for 1 point,

$$c(u_1, \dots, u_d, \theta) = \int_0^1 \prod_{j=1}^{d} c_{j,V_1}(u_j, v_1, \theta_j) dv_1 \tag{2.1}$$

Likelihood for complete data

$$L = \prod_{i=1}^{n} c(u_{i1}, \dots, u_{id}, \theta) \tag{2.2}$$

$$= \prod_{i=1}^{n} \int_0^1 \prod_{j=1}^{d} c_{j,V_1}(u_{ij}, v_1, \theta_j) \tag{2.3}$$

So we get the log likelihood to be:

$$log(L) = \sum_{i=1}^{n} \int_0^1 \prod_{j=1}^{d} c_{j,V_1}(u_{ij}, v_1, \theta_j) \tag{2.4}$$

If we use Gauss-Legendre method to evaluate integrals with k points, the formula becomes:

$$log(L) = \sum_{i=1}^{n} \sum_{k=1}^{k} w_k \cdot \prod_{j=1}^{d} c_{j,V_1}(u_{ij}, g_k, \theta_j) \tag{2.5}$$

where g_k are the gauss-legendre evaluation points and w_k are the gauss-legendre weights.

Here we can see that if we take d assets , n datapoints and k points for integral evaluation, we have $n * d * k$ arithmetic operations excluding the calculation of copula functions. So we can see that just the likelihood calculation becomes very heavy.

2-factor Models

In this, the log likelihood comes out to be:

$$log(L) = \sum_{i=1}^{n} \int_0^1 \int_0^1 \prod_{j=1}^{d} \{c_{j,V_2;V_1}(C_{j|V_1}(u_{ij}|g_{k1}), g_{k2}) \cdot c_{j,v_1}(u_{ij}, g_{k1})\} dv_1 dv_2 \tag{2.6}$$

If we use Gauss-Legendre method to evaluate integrals with k points, the formula becomes:

$$log(L) = \sum_{i=1}^{n} \sum_{k2=1}^{k} \sum_{k1=1}^{k} w_{k1} \cdot w_{k2} \prod_{j=1}^{d} \{ c_{j,V_2;V_1}(C_{j|V_1}(u_{ij}|g_{k1}), g_{k2}) \cdot c_{j,v_1}(u_{ij}, g_{k1}) \} \tag{2.7}$$

where g_k are the gauss-legendre evaluation points and w_k are the gauss-legendre weights.

Here we have $n*d*k^2$ arithmetic operations excluding the calculation of copula functions. So we can see that just the likelihood calculation becomes very heavy.

2.2.2 Estimation Methods

So I chose 2 estimation methods to estimate the parameters in the model. They are discussed in detail below.

Gradient Descent

Gradient descent is a first-order iterative optimization algorithm. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or of the approximate gradient) of the function at the current point. If instead one takes steps proportional to the positive of the gradient, one approaches a local maximum of that function; the procedure is then known as gradient ascent. For the likelihood optimzation, gradient descent was used with finite difference used as the gradient. Gradient was also calculated using the analytical expression. The expression comes out to be:

$$\frac{dlog(L)}{d\theta_p} = \sum_{i=1}^{n} \sum_{k=1}^{k} \Big\{ \Big(\prod_{j=1, j\neq p}^{d} c_{j,V_1}(u_{ij}, g_k, \theta_j) \Big) \cdot \frac{dc_{p,V_1}(u_{ip}, g_k, \theta_p)}{d\theta_p} \Big\} \tag{2.8}$$

So the gradients w.r.t. all the parameters were calculated and thus the local optimal was calculated. Further multiple starting values were taken to ensure that global maxima was achieved. The drawback of this method is found to be the slow linear convergence and inability to predict whether the global maxima have been achieved or not. The second problem could be cuntered using different starting points. So as a improvement we use quasi-newton methods.

Broyden-Fletcher-Goldfarb-Shanno Algorithm

In numerical optimization, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm is an iterative method for solving unconstrained nonlinear optimization problems.

The BFGS method approximates Newton's method, a class of hill-climbing optimization techniques that seeks a stationary point of a (preferably twice continuously differentiable) function. For such problems, a necessary condition for optimality is that the gradient be zero. In quasi-Newton methods, the Hessian matrix of second derivatives doesn't need to be evaluated directly. Instead, the Hessian matrix is approximated using updates specified by gradient evaluations (or approximate gradient evaluations). Quasi-Newton methods are generalizations of the secant method to find the root of the first derivative for multidimensional problems. In multi-dimensional problems, the secant equation does not specify a unique solution, and quasi-Newton methods differ in how they constrain the solution. The BFGS method is one of the most popular members of this class.

From an initial guess $\mathbf{x}_0$ and an approximate Hessian matrix B_0 the following steps are repeated as $\mathbf{x}_k$ converges to the solution.

1. Obtain a direction $\mathbf{p}_k$ by solving: $B_k\mathbf{p}_k = -\nabla f(\mathbf{x}_k)$.
2. Perform a line search to find an acceptable stepsize α_k in the direction found in the first step, then update $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k$.
3. Set $\mathbf{s}_k = \alpha_k\mathbf{p}_k$.
4. $\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$.
5. $B_{k+1} = B_k + \frac{\mathbf{y}_k\mathbf{y}_k^{\mathrm{T}}}{\mathbf{y}_k^{\mathrm{T}}\mathbf{s}_k} - \frac{B_k\mathbf{s}_k\mathbf{s}_k^{\mathrm{T}}B_k}{\mathbf{s}_k^{\mathrm{T}}B_k\mathbf{s}_k}$.

$f(\mathbf{x})$ denotes the objective function to be minimized. Convergence can be checked by observing the norm of the gradient, $|\nabla f(\mathbf{x}_k)|$. Practically,B_0 can be initialized with $B_0 = I$, so that the first step will be equivalent to a gradient descent, but further steps are more and more refined by B_k, the approximation to the Hessian.

The first step of the algorithm is carried out using the inverse of the matrix B_k, which can be obtained efficiently by applying the Sherman-Morrison formula to the fifth line of the algorithm, giving

$$B_{k+1}^{-1} = \left(I - \frac{s_k y_k^T}{y_k^T s_k}\right) B_k^{-1} \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) + \frac{s_k s_k^T}{y_k^T s_k}. \tag{2.9}$$

This can be computed efficiently without temporary matrices, recognizing that B_k^{-1} is symmetric, and that $\mathbf{y}_k^{\mathrm{T}}B_k^{-1}\mathbf{y}_k$ and $\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k$ are scalar, using an expansion such as

$$B_{k+1}^{-1} = B_k^{-1} + \frac{(\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k + \mathbf{y}_k^{\mathrm{T}}B_k^{-1}\mathbf{y}_k)(\mathbf{s}_k\mathbf{s}_k^{\mathrm{T}})}{(\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k)^2} - \frac{B_k^{-1}\mathbf{y}_k\mathbf{s}_k^{\mathrm{T}} + \mathbf{s}_k\mathbf{y}_k^{\mathrm{T}}B_k^{-1}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k}. \tag{2.10}$$

In statistical estimation problems (such as maximum likelihood or Bayesian inference), credible intervals or confidence intervals for the solution can be estimated from the inverse of the final Hessian matrix. However, these quantities are technically defined by the true Hessian matrix, and the BFGS approximation may not converge to the true Hessian matrix.

2.3 Platform Details

The parameter estimation of Factor Copula models using likelihood was implemented for 1-factor and 2-factor models using both gradient descent and BFGS Algorithm.

Machine Details

All the programs were run on intel i5-3317U CPU at 1.7GHz.

Programming Languages

The algorithms are implemented and tested in both R and $C++$. While internal libraries are used in R, everything is implemented from scratch in $C++$.

2.4 Model Testing Parameters

So once the model is established we would need to compare it with the existing models and implementations to compare the running time and efficiency. It would help us determine the viablity of this model. So we would compare our given model with C-Vine and D-vine Copulas. And the model accuracy parameters will be the following:

Log Likelihood

The cumulative log likelihood of the data. It is the most basic measure of accuracy. But it may be biased towards an over-fitted model.

AIC

The Akaike information criterion (AIC) is a measure of the relative quality of statistical models for a given set of data. Suppose that we have a statistical model of some data. Let $\hat{L}$ be the maximum value of the likelihood function for the model; let k be the number of estimated parameters in the model. Then the AIC value of the model is the following:

$$AIC = 2k - 2\ln\hat{L}$$

where k is the number of free parameters. Given a set of candidate models for the data, the preferred model is the one with the minimum AIC value. AIC rewards goodness of fit (as assessed by the likelihood function), but it also includes a penalty that is an increasing function of the number of estimated parameters. The penalty discourages overfitting, because increasing the number of parameters in the model almost always improves the calculated goodness of the fit.

BIC

In statistics, the Bayesian information criterion (BIC) or Schwarz criterion (also SBC, SBIC) is a criterion for model selection among a finite set of models; the model with the lowest BIC is preferred. It is based, in part, on the likelihood function and it is closely related to the Akaike information criterion (AIC). Formally it is given by:

$$BIC = \ln(n)k - 2\ln(\hat{L})$$

where n is the number of data points and k is the number of parameters.

2.5 Code Properties

The salient features of current code are:

1. The code is completely written in C++ without any additional dependencies.
2. It currently features 4-bivariate copula families.
3. Gauss-Legendre method is implemented for numerical integration.
4. Gradient descent and BFGS methods are implemented for the Maximum Likelihood Estimation.
5. Finite difference methods are used for calculating gradients.
6. Supports 1,2,3-factor models. However 3-factor models take huge amount of time.
7. Integrated with RCpp, so that the code can be used in R programs.

There are a lot of minor improvements possible in the code like modularizing the code, using memoization techniques, analytical formulas to increase efficiency. Support more bivariate families however the major families have been already incorporated.

2.6 Testing the code

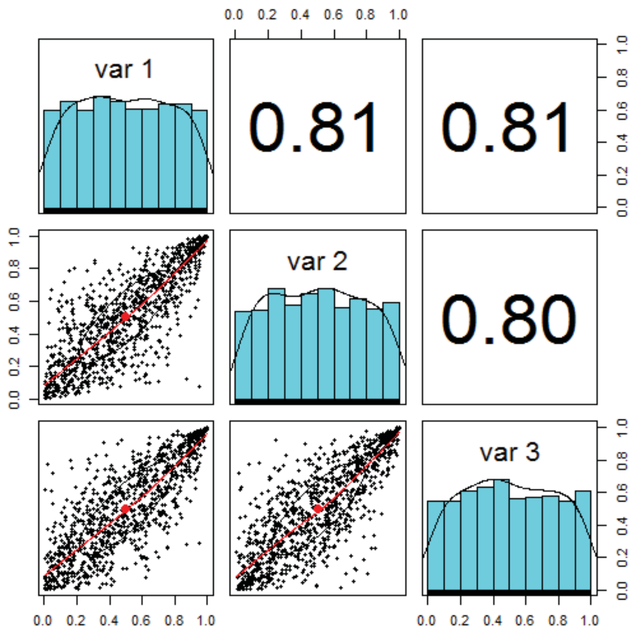
In this chapter we'll discuss the small experiments done to analyze the performance and efficiency of codes.

2.6.1 Testing against simulated data

Data

I simulated some data using R copula library. The data generated was for 10 variables(assets) with 250 distinct points for each variable. The marginals of variables was uniform but the joint distribution was given by the gumbel copula with the parameter set to be 2.75. A smaller dataset is used to get some visualisation.

Figure 2.1: Pairwise Analysis of Simulated Data

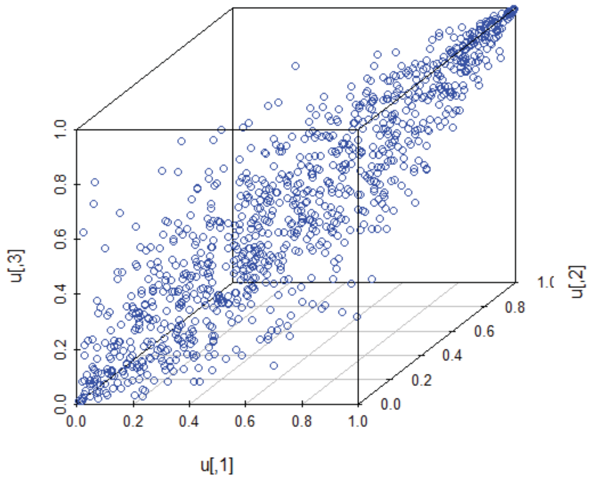


From the images we can see that the data has a positive tail dependence and the value was found to be 0.71.

Code:

```
u <- rCopula(1000,gumbelCopula(2.75, dim = 3))
scatterplot3d(u,color=4)
pairs.panels(u)
```

Figure 2.2: Visualization of Simulated Data



Experiment

Now this data is next fitted using various models such as R-Vine copulas, C-Vine Copulas, Factor Copulas (1-factor,2-Factor) and the various values of AIC, BIC etc. are calculated. Futher the approximate time is also measured and observations are made. For R-Vines library VineCopula of R is used and for C-Vine, D-Vine CDVine library for R is used. Factor Copula implementation is completely implemented in C++ from scratch.

Further the data set was extended to 300 points and 200 points were used for model fitting and the rest 100 points were taken to test the model fiitting.

Observations

The following observations were made from this small experiment.

	LogLik	AIC	BIC	Running Time
C-Vine	1988.72	-3887.44	-3673.75	11.39
D-Vine	1947.93	-3805.86	-3647.39	10.47
GD 1-Factor	1671.64	-3323.28	-3288.06	15.31
BFGS 1-Factor	1673.54	-3327.08	-3291.86	3.4
BFGS 2-Factor	1821.92	-3603.84	-3533.41	25.23

	Training LogLik	Testing LogLik
D-Vine	1525.40	722.41
BFGS 1-Factor	1284.05	658.846

So from these observations, we could conclude that the 1-Factor and 2-Factor Models could be used as approximation to the Vine Copula Models. But currently the 2 factor model takes a higher fitting time than the C-Vine or D-Vine models. The code used for reference in this is highly optimized and industry-level efficient. So we need to improve the code as much as required. Further, in D-Vine around 22 bivariate copula families are present while currently I have only incorporated 4 basic families.

From the testing data we can see that 1-factor slightly underfits the model but it can be used where time is of the essence.

Chapter 3

Simulating Factor Copula Models

As we have seen in the flow chart that after fitting the model, the next thing needed is to be able to generate points from the distribution. It is essential for application involving predictions etc. So now I studied the approaches to multivariate statistical simulation.

3.1 Common Simulation Techniques

In the following paragraphs we'll discuss common statistical simulation techniques which would prove to be useful in our case.

3.1.1 Probability Integral Transform for Univariate Generation

In this method, given an arbitrary cumulative distribution function $F(x)$, we can generate sample points from this distribution. The probability integral transform states that if X is a continuous random variable with cumulative distribution function F_X, then the random variable $Y = F_X(X)$ has a uniform distribution on $[0, 1]$. The inverse probability integral transform is just the inverse of this: specifically, if Y has a uniform distribution on $[0, 1]$ and if X has a cumulative distribution F_X, then the random variable $F_X^{-1}(Y)$ has the same distribution as X .

Steps to generate sample points is as follows:

1. Generate a random number u from the standard uniform distribution in the interval $[0, 1]$.
2. Compute the value x such that $F(x) = u$.
3. Take x to be the random number drawn from the distribution described by F.

Proof of Correctness

We know that, if $x = T(y)$ then

$$\begin{aligned} f_X(x)dx &= f_Y(y)dy \\ f_X(x) &= \frac{f_Y(y)}{\frac{dy}{dx}} \\ &= \frac{f_Y(y)}{T'(y)} \end{aligned}$$

Now let T be the cumulative distribution function F. Then $T'(y) = f_Y(y)$. So,

$$\begin{aligned} f_X(x)dx &= \frac{f_Y(y)}{f_Y(y)} \\ &= 1 \end{aligned}$$

Hence the distribution of X is uniform and its domain could also be found as 0 to 1.

But this approach is only limited to univariate distribution. We cannot generate multivariate distributed variables using this approach. Other Methods for univariate distribution involves composition, convolution, Acceptance-Rejection etc.

3.1.2 Conditional Distribution Approach for Multivariate Distributions

The conditional distribution approach approach reduces the problem of generating p-dimensional random vector into a series of p univariate generation problems. This strategy can utilie the vast body of techniques available for univariate generation. Let the random vector $X = (X_1, X_2, \cdots, X_p)'$ be the p-dimensional random vector of interest. The conditional distribution approach involves the following steps:

1. Generate $X_1 = x_1$ from the marginal distribution of X_1.

2. Generate $X_2 = x_2$ from the conditionall distribution of X_2 given $X_1 = x_1$.

3. Generate $X_3 = x_3$ from the conditionall distribution of X_2 given $X_1 = x_1$ and $X_2 = x_2$.

And so on.

However this approach requires the knowledge of all the conditional distributions and coputing their inverses which may be cumbersome to find for complex joint distributions.

3.1.3 Maximum Likelihood Approach

This approach generates X_i using the marginal distribution. Rest of the $p-1$ variables are generated such that the the likelihood is maximum. This approach is computationally expensive and not very feasible.

Other approaches include transformation of variables or using some special properties of the distribution.

3.2 Simulation of Factor Copula Distribution

The cdf of 1-factor copula model is given by

$$C(u_1, \dots, u_d) = \int_0^1 \prod_{j=1}^{d} C_{j|V_1}(u_j|v_1) dv_1$$

And the pdf is given by

$$c(u_1, \dots, u_d) = \int_0^1 \prod_{j=1}^{d} c_{j,V_1}(u_j, v_1) dv_1$$

Because of complex nature of the pdf and cdf, it is difficult to get the conditional distributions in closed form.

But we can use the basic definition of factor copulas for generation. It says that given the hidden variables which are distributed as $U[0,1]$ all the other variables are independent to each other. So we generate the hidden variables and then use the conditional distribution $C_{j|V_1}(u_j|v_1)$ to generate u_j. So given the parameter vector θ, the steps for generation for 1 factor model are as follows:

1. Generate a random point from $U[0,1]$. Call it v_1.
2. Get the conditional distribution $C_{j|V_1}(u_j|v_1)$.

   $$C_{j|V_1}(u_j|v_1) = \frac{\partial C(u_j, v_1)}{\partial v_1}$$

   . This function could be computed numerically or analytically.
3. Use the probability integral transform to compute the value of u_j given v_1 i.e. for $j = 1, 2, \cdots, d$ generate a random point y_j from $U[0,1]$. Compute u_j such that $C_{j|V_1}(u_j|v_1) = y_j$.

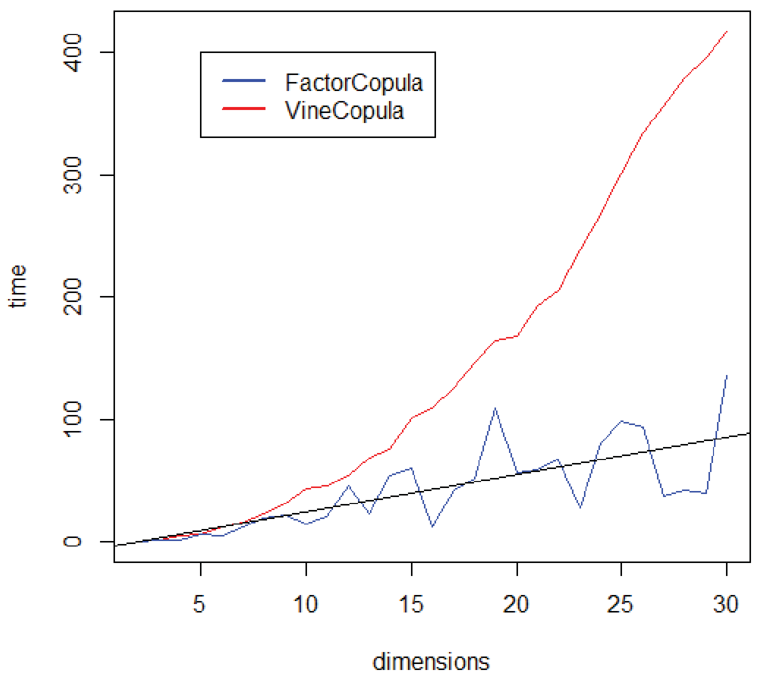
After suggesting the approach, similar approach was also seen in this.

If we compare to the vine copula approach where one has to compute $O(d^2)$ inverse, this approach only requires $O(d)$ inverse and is thus computationally faster.

3.2.1 Some Observations

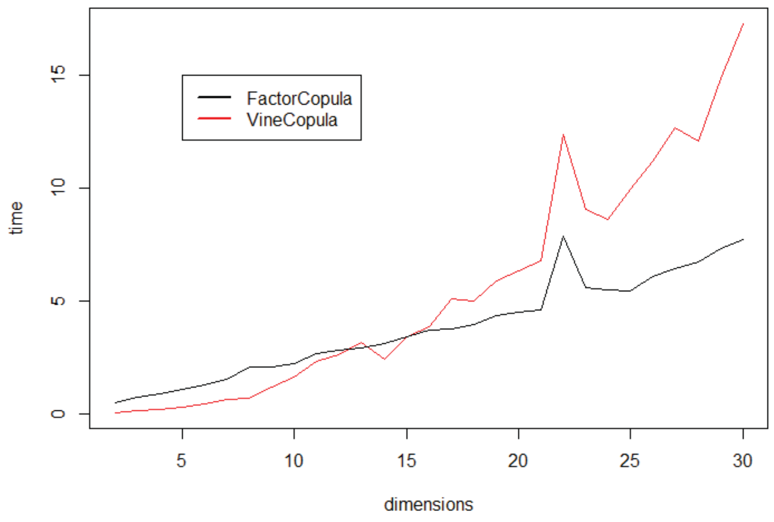
I increased the dimension of data from 2 to 30 and measured the simulation and fitting times. Here are the results:

Figure 3.1: Dimensions Vs Running Time



We could easily see that 1-Factor Copula Model is much more efficient than the traditional Vine Copula Models.

Figure 3.2: Dimensions Vs Simulation Time



Chapter 4

Application of Factor Copulas

In this chapter, we explored different avenues for applications of Factor Copulas. Finally we did a robust portfolio optimization using factor copulas where we minimized the worst case CVaR. In the following sections we shall read in detail about the same.

4.1 Mathematical Preliminaries

4.1.1 VaR

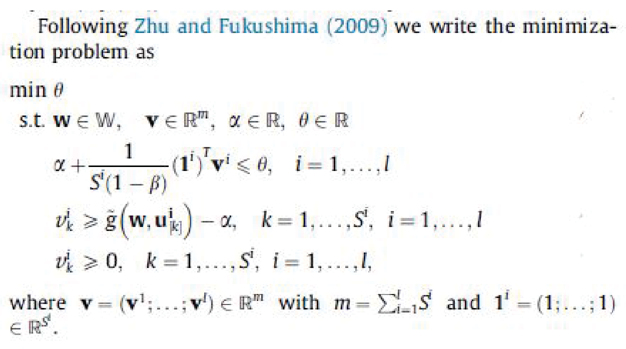
Value at Risk (VaR) is a measure of the risk of investments. VaR is defined as: for a given portfolio, time horizon, and probability p, the p VaR is defined as a threshold loss value, such that the probability that the loss on the portfolio over the given time horizon exceeds this value is p.

4.1.2 CVaR or Expected Shortfall

The "expected shortfall at q % level" is the expected return on the portfolio in the worst q% of cases. It is a coherent risk measure.

4.1.3 WCVaR for Portfolio Optimization

Zhu and Fukushima (2009) proposed a worst-case CVaR approach for portfolio optimization where the distribution of underlying asset is not assumed to belong to a certain family but rather to different families and WCVaR is defined as maximum CVaR among those families. Kakouris and Rustem further used the same approach for copulas. We have picked up the same formulation and substituted factor copulas instead of copulas. The formulation is as follows



Following Zhu and Fukushima (2009) we write the minimization problem as

$$
\begin{aligned}
&\min \theta \\
&\text{s.t. } \mathbf{w} \in \mathbb{W}, \quad \mathbf{v} \in \mathbb{R}^m, \ \alpha \in \mathbb{R}, \ \theta \in \mathbb{R} \\
&\qquad \alpha + \frac{1}{S^i(1-\beta)} \left(\mathbf{1}^i\right)^T \mathbf{v}^i \leqslant \theta, \quad i = 1, \ldots, l \\
&\qquad v_k^i \geqslant \tilde{g}\left(\mathbf{w}, \mathbf{u}^i_{[k]}\right) - \alpha, \quad k = 1, \ldots, S^i, \ i = 1, \ldots, l \\
&\qquad v_k^i \geqslant 0, \quad k = 1, \ldots, S^i, \ i = 1, \ldots, l,
\end{aligned}
$$

where $\mathbf{v} = (\mathbf{v}^1; \ldots; \mathbf{v}^l) \in \mathbb{R}^m$ with $m = \sum_{i=1}^{l} S^i$ and $\mathbf{1}^i = (1; \ldots; 1) \in \mathbb{R}^{S^i}$.

Here S(i) are the number of simulated values for ith family which we have to simulate. and g is the loss function.

4.1.4 Process Summary for Portfolio Optimization

The complete portfolio optimization process could be split in following major tasks:

1. Fit appropriate models on returns time series of each model.
2. Calculate the residuals and obtain the marginal distributions.
3. Convert the residuals to the uniform distribution and fit facctor copulas on the same.
4. Simulate S values using the learned model.
5. Use the simulated values in the LPP model discussed above and obtain the weights.
6. Use the weights to compare the portfolio returns.

Chapter 5

Numerical Example

We used the WCVaR formulation to verify the usefulness of Factor Copulas.

5.1 Data

I took S&P 100 data from January 1 1998 to 31 January 2013. Out of the 100 stocks, 18 had incomplete data during the period so they were removed. The closing prices were already adjusted for the dividends etc. The data had a total of 4349 data points.

5.2 Fitting time-series models

I used log of returns of indivisual assets and used ARMA-GJR-GARCH model for time-series fitting. GJR-GARCH is used to capture the leverage effect. Since the parameters of ARMA come to be different for different time periods and different assets. I used a common ARMA(1,1) model. GARCH parameters were learnt indivisually.

$$\sigma_t^2 = \omega + (\alpha + \gamma I_{t-1})\varepsilon_{t-1}^2 + \beta\sigma_{t-1}^2$$

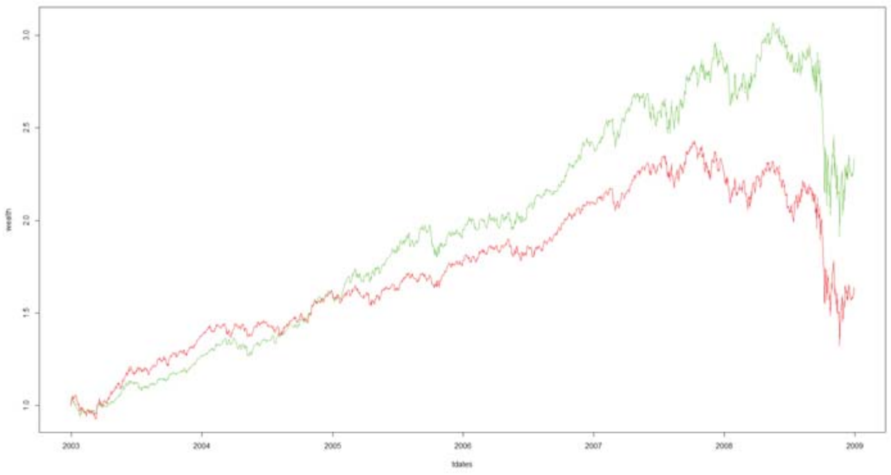
$$I_{t-1} = \begin{cases} 0 & \text{if } r_{t-1} \geq \mu \\ 1 & \text{if } r_{t-1} < \mu \end{cases}$$

5.3 Static Portfolio

In this kind of portfolio, I used starting 1255 points (1998-2002) as the training points. Then the portfolio weights were obtained using the LPP. The weights were not changed again for a span of 6 years.

In this case I skipped the time series analysis but rather assumed the indivisual returns to be lognormally distributed. So normal distribution was fitted on the log of the returns. Then the returns were converted to uniform and copulas were used.

5.3.1 Observations



In this figure it was assumed that 1 unit of currency was invested on 1st Jan 2003 and its value is shown as various times. The green line represents our portfolio and the red line is equal weights portfolio.

Here are some more observations:

	Naive Model	WCVaR Model
Average Return(Per Day)	0.04%	0.06%
Sharpe Ratio	0.036	0.056
Sortino Ratio	0.053	0.082

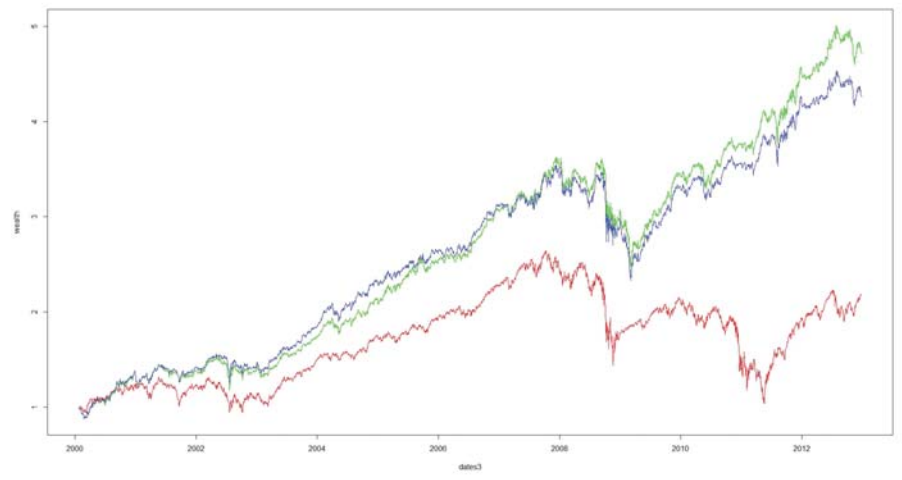
We can see from the graph that at the nd of tenure return of Naive Portfolio is 56% while that of WCP is 124%. If we assume a risk free rate of 6% per annnum compunded annually, the return after 6 years is around 42%.Many other statistics such as average drawdown are also significant for WCP.

5.4 Dynamic Portfolio

In this type of portfolio, the weights are recalibrated after every 130 working days (appx. 6 months). The data used to recalibrate the model is of previous 520 days (appx 2 years). I thus used a rolling window model. The data is available for around 15 years. Hence we have 25 rolling windows. In this section I also included a CVaR model based on gaussian copula. We'll also comare our performance with that.

5.4.1 Observations

Figure 5.1: Returns on Dynamic Portfolio



In this figure it was assumed that 1 unit of currency was invested on 1st Jan 2000 and its value is shown as various times. The green line represents our portfolio and the red line is equal weights portfolio while the blue line is the GP. It is clearly evident from the graph that the returns are much better in GP and WCP than the EWP.

Here are the window wise comparison of the same.

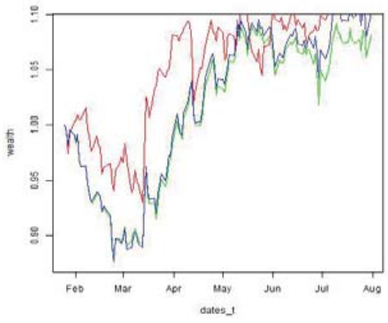
When I performed the t-test on the return series, I was unable to reject the null hypothesis that the returns are equal. But when we consider parameters like Sortino Ratio which describes returns per unit risk we could see that worst case portfolio is much better than the others.

Here are the observations:

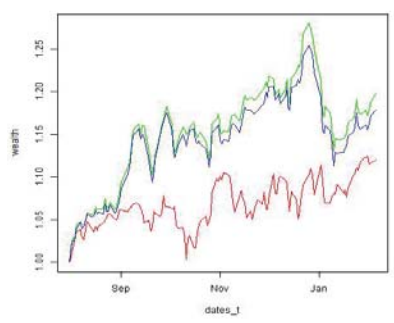
	Naive Model	Gaussian Model	WCVaR Model
Average Return(Per Day)	0.034%	0.058%	0.061%
Sharpe Ratio	0.03	0.048	0.051
Sortino Ratio	0.053	0.077	0.082

5.5 Summary and Conclusions

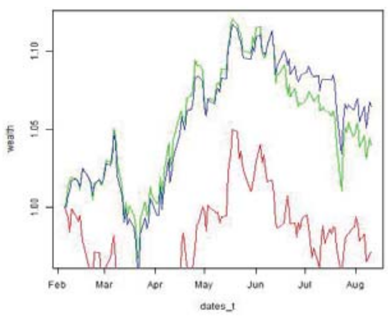
Using this numerical experiment we could conclude various things. Firstly, Factor Copulas are pretty fast to learn and we were able to meodel 82 assets at once in same time as Gaussian Copula Modelling. The results of WCP are also comparable and better than the GP. While there is some information loss due to the assumption of latent factors, the results are good enough and fast. However we could still improve the results as we work on mathematics of dependence relations in Factor Copulas and including more families in the Factor Copula code. (Currently only 4 families are supported)



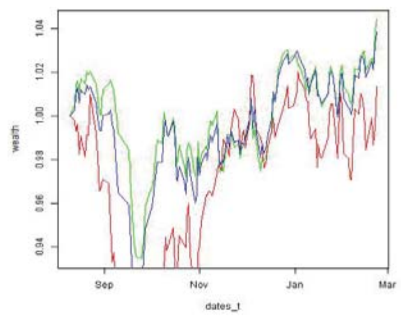
(a) Window 1



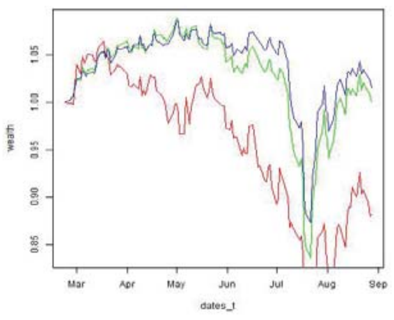
(b) Window 2



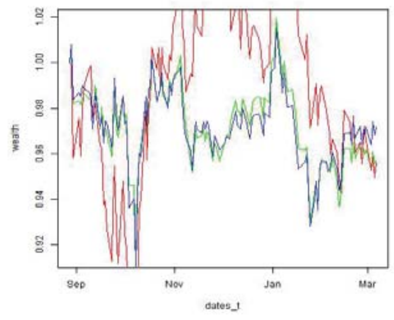
(c) Window 3



(d) Window 4

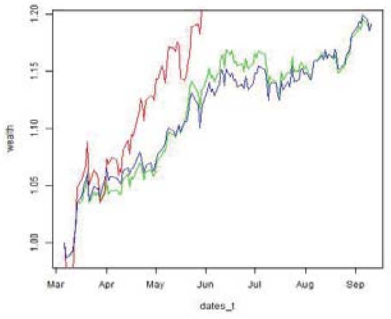


(e) Window 5

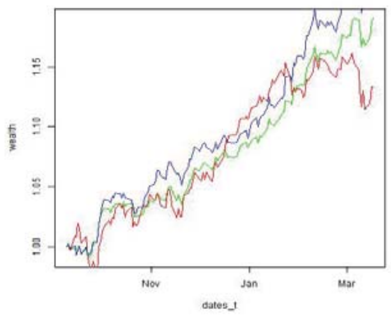


(f) Window 6

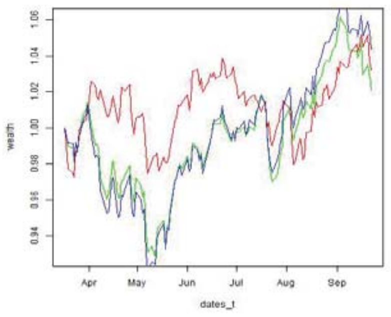
Figure 5.2: Window-wise Results



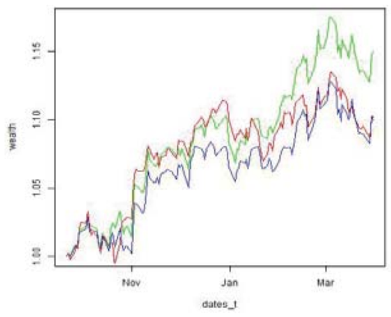
(a) Window 7



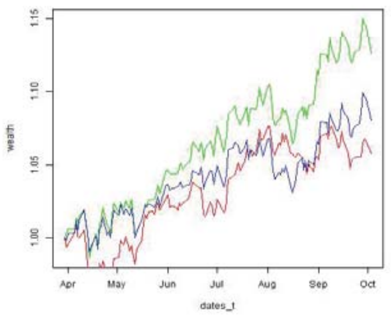
(b) Window 8



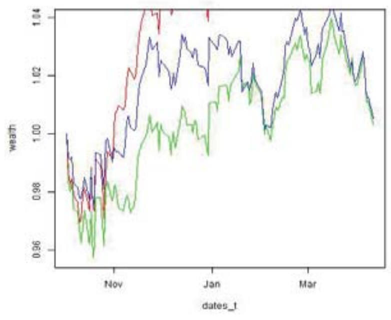
(c) Window 9



(d) Window 10

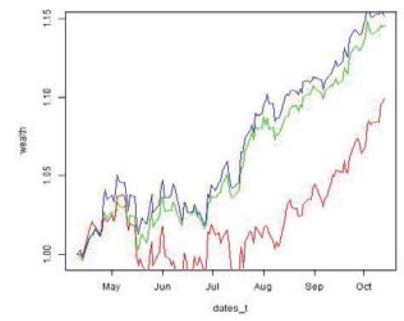


(e) Window 11

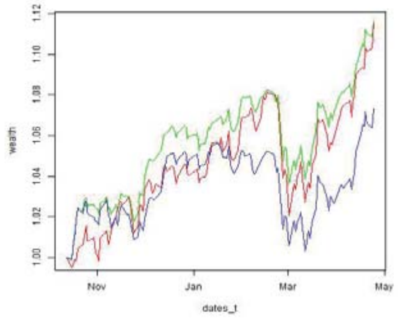


(f) Window 12

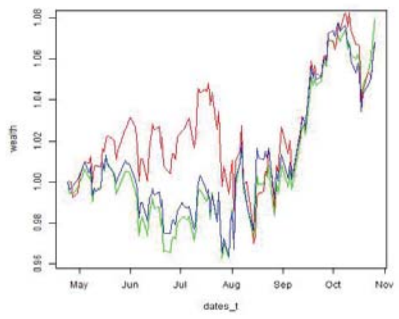
Figure 5.3: Window-wise Results



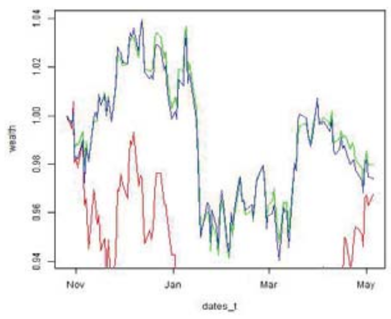
(a) Window 13



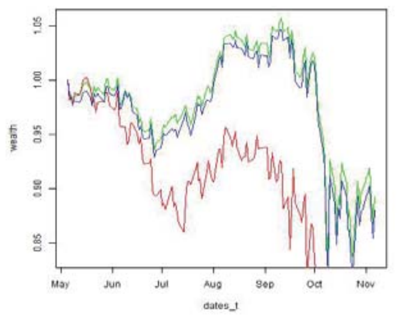
(b) Window 14



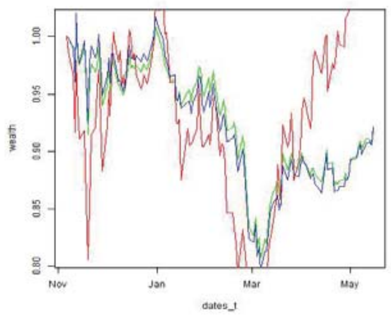
(c) Window 15



(d) Window 16

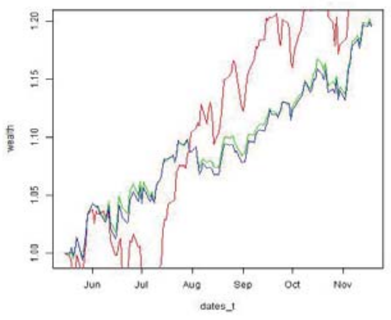


(e) Window 17

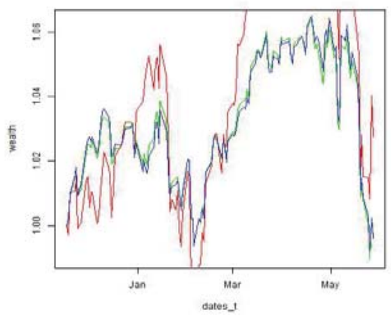


(f) Window 18

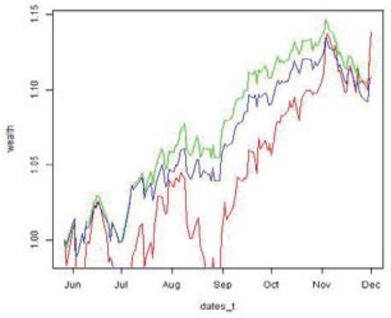
Figure 5.4: Window-wise Results



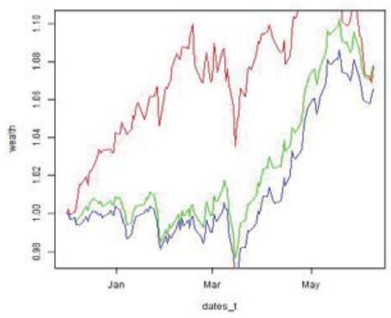
(a) Window 19



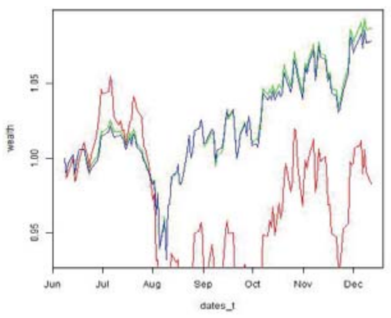
(b) Window 20



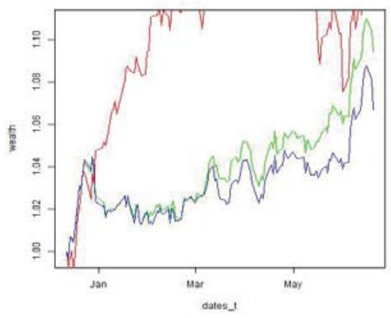
(c) Window 21



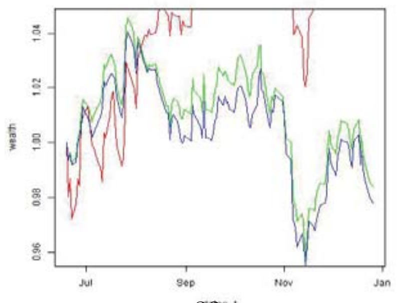
(d) Window 22



(e) Window 23



(f) Window 24



(g) Window 25

Figure 5.5: Window-wise Results

Project Summary

Aim of this project was to develop a library for Factor Copula nd investigate the claims by the author of the paper. Further we also wanted to compare the performance by existing methods and analyze the performance in a real-life financial application.

To achieve these goals RCpp Library was used to develop the code for model learning and simulation in C++ with integration in R for ease of use. The perormance (both parameter learning and simulation) was compared with Vine Copulas and was found to be very fast when dealing with large dimensions and fairly accurate.

Further we used a robust Worst-case CVaR model for portfolio optimization and used Factor Copulas to model dependencies. We used 82 assets at once and still the model was very fast. The results obtained were comparable to other models.

So we can conclude that Factor Copulas are really fast and could be used in applications that requires modelling dependence amond high-dimensional data where VineCopula Models become too slow to compute.

References

[1] P. J. Davis and P. Rabinowitz. *Methods of numerical integration*. Courier Corporation, 2007.

[2] J. C. Hull and A. D. White. Valuation of a cdo and an n-th to default cds without monte carlo simulation. *The Journal of Derivatives*, 12(2):8–23, 2004.

[3] M. E. Johnson. *Multivariate statistical simulation: A guide to selecting and generating continuous multivariate distributions*. John Wiley & Sons, 2013.

[4] I. Kakouris and B. Rustem. Robust portfolio optimization with copulas. *European Journal of Operational Research*, 235(1):28–37, 2014.

[5] P. Krupskii and H. Joe. Factor copula models for multivariate data. *Journal of Multivariate Analysis*, 120:85–101, 2013.

[6] P. Krupskii and H. Joe. Structured factor copula models: Theory, inference and computation. *Journal of Multivariate Analysis*, 138:53–73, 2015.

[7] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[8] G. Mazo, S. Girard, and F. Forbes. A flexible and tractable class of one-factor copulas. *Statistics and Computing*, 26(5):965–979, 2016.

[9] A. K. Nikoloulopoulos, H. Joe, and H. Li. Vine copulas with asymmetric tail dependence and applications to financial return data. *Computational Statistics & Data Analysis*, 56(11):3659–3673, 2012.

[10] D. H. Oh and A. J. Patton. Modelling dependence in high dimensions with factor copulas. *Journal of Business & Economic Statistics*, (just-accepted), 2015.

[11] S. Sriboonchitta, J. Liu, V. Kreinovich, and H. T. Nguyen. A vine copula approach for analyzing financial risk and co-movement of the indonesian, philippine and thailand stock markets. In *Modeling Dependence in Econometrics*, pages 245–257. Springer, 2014.

[12] S. Zhu and M. Fukushima. Worst-case conditional value-at-risk with application to robust portfolio management. *Operations research*, 57(5):1155–1168, 2009.