Thomas Graf, Bakk. techn.

Vine Copulas und deren Anwendung in der Finanzmathematik

Diplomarbeit

zur Erlangung des akademischen Grades
Diplom-Ingenieur

Studium
Masterstudium Technische Mathematik

Alpen-Adria-Universität Klagenfurt
Fakultät für Technische Wissenschaften

Begutachter: Assoc.Prof.Mag.Dr. Gunter Spöck
Institut: Institut für Statistik

Juli/2014

Ehrenwörtliche Erklärung

Ich erkläre ehrenwörtlich, dass ich die vorliegende wissenschaftliche Arbeit selbstständig angefertigt und die mit ihr unmittelbar verbundenen Tätigkeiten selbst erbracht habe. Ich erkläre weiters, dass ich keine anderen als die angegebenen Hilfsmittel benutzt habe. Alle aus gedruckten, ungedruckten Quellen oder dem Internet im Wortlaut oder im wesentlichen Inhalt übernommenen Formulierungen und Konzepte sind gemäß den Regeln für wissenschaftliche Arbeiten zitiert und durch Fußnoten bzw. durch andere genaue Quellenangaben gekennzeichnet.

Die während des Arbeitsvorganges gewährte Unterstützung einschließlich signifikanter Betreuungshinweise ist vollständig angegeben.

Die wissenschaftliche Arbeit ist noch keiner anderen Prüfungsbehörde vorgelegt worden. Diese Arbeit wurde in gedruckter und elektronischer Form abgegeben. Ich bestätige, dass der Inhalt der digitalen Version vollständig mit dem der gedruckten Version übereinstimmt.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

_____ Klagenfurt, Juli 2014

(Unterschrift) (Ort, Datum)

Vorwort

Während meines Praxissemesters wurde ich mit dem Problem konfrontiert, Abhängigkeiten mehrerer Zufallsvariablen zu modellieren. Mein Arbeitskollege war der Meinung eine gute Lösung mit Hilfe von Vine Copulas zu bekommen. Dieses Thema weckte mein Interesse und ich entschloss mich in meiner Diplomarbeit mit Vine Copulas zu beschäftigen. Außerdem hat die Theorie über Copula Funktionen viele Anwendungsbereiche. Nicht nur in der Finanzmathematik, sondern auch im Versicherungsbereich, in der Meteorologie, in der Medizin und in vielen anderen Bereichen kann diese verwendet werden.

Ich möchte mich hiermit bei meinen Eltern bedanken, die mich während meiner gesamten Ausbildung unterstützt haben. Außerdem möchte ich mich noch bei Professor Spöck für die tolle Betreuung bedanken.

Inhaltsverzeichnis

1 Einleitung

Es stellt sich oft die Frage, wie man die Abhängigkeiten mehrerer Zufallsvariablen beschreiben kann. Es seien beispielsweise zwei Zufallsvariablen X und Y mit zugehörigen Verteilungen $F_x(x) = P(X \leq x)$ und $F_y(y) = P(Y \leq y)$ gegeben. Falls deren gemeinsame Verteilung $F(x, y) = P(X \leq x, Y \leq y)$ bekannt ist, so enthält diese auch die Information über die Abhängigkeiten der beiden Zufallsvariablen. Meistens ist die gemeinsame Verteilung aber unbekannt. Daher wird oft (lineare) Korrelation ρ verwendet, um die Abhängigkeit zu beschreiben. Die Korrelation als Abhängigkeitsmaß ist sehr beliebt, vor allem weil man diese leicht berechnen kann. Diese ist definiert als

$$\rho = \frac{cov(X, Y)}{\sigma_X \sigma_Y},$$

wobei σ_X und σ_Y die Standardabweichungen von X und Y sind und $cov(X, Y)$ die Kovarianz beschreibt. Ob die Korrelation ein gutes Maß für die Abhängigkeit ist, hängt von der gemeinsamen Verteilungsfunktion ab. Es werden nun elliptische Verteilungen und nicht-elliptische Verteilungen betrachtet.
Es lässt sich zeigen, dass im Falle von multivariaten elliptischen Verteilungen (z.B.: Normalverteilung oder t-Verteilung) die Korrelation ein gutes Maß für die Abhängigkeit ist. Mit Hilfe der Korrelation kann man aber nicht alle Eigenschaften von den Zufallsvariablen X und Y beschreiben. Sind X und Y unabhängig voneinander, dann folgt daraus, dass die Korrelation gleich Null ist ($\rho = 0$). Die Umkehrung gilt im Allgemeinen aber nicht. Ein weiterer Nachteil der Korrelation ist, dass diese nicht transformationsinvariant ist. Die Korrelation von X und Y ist im Allgemeinem nicht gleich der Korrelation von $\ln(X)$ und $\ln(Y)$.
Betrachtet man nicht-elliptische Verteilungen, so treten noch mehr Nachteile der Korrelation auf. Zum einen kann keine Korrelation definiert werden, wenn die Varianz nicht endlich ist (z.B.: Levy-Verteilung). Ein weiteres Problem ist, dass ρ nicht die Werte 1 oder -1 annehmen muss, aber die Variablen trotzdem perfekt positiv bzw. negativ abhängig sind. Außerdem reichen die Randverteilungen und Korrelation nicht aus, um multivariate nicht-elliptische Verteilungen zu bestimmen. Die Korrelation ist für nicht-elliptische Verteilungen daher ein schlechtes Maß für die Abhängigkeit.

Da die Korrelation nicht in allen Bereichen vernünftig eingesetzt werden kann, greift man oft auf andere Abhängigkeitsmaße zurück. Beispielsweise können Rangkorrelationskoeffizienten wie Kendall's Tau oder Spearman's Rho verwendet werden, welche transformationsinvariant sind. Anders als bei der Korrela-

tion, sind die Rangkorrelationskoeffizienten robust gegenüber Ausreißern. Der natürlichste Weg um Abhängigkeiten zu beschreiben, ist aber durch die Verwendung von Copula Funktionen gegeben. Mit diesen Funktionen ist es möglich den Zusammenhang mehrerer Zufallsvariablen sehr gut zu beschreiben. Mit der Copula Funktion ist es nämlich auch möglich nicht-symmetrische Abhängigkeiten zu modellieren und Ausreißer können besser beschrieben werden (*tail dependence*).

Das Interesse an Copula Funktionen in der Statistik ist in den letzten Jahren sehr gewachsen. Für den zweidimensionalen Fall sind bereits viele verschiedene Copulas und deren Eigenschaften bekannt. In höheren Dimensionen wird der Umgang mit Copulas schwieriger. Es sind zwar einige mehrdimensionale Copulas bekannt, aber diese reichen oft nicht aus, um bestimmte Abhängigkeiten zu modellieren. Um auch in höheren Dimensionen Copulas vernünftig verwenden zu können, wird oft das Konzept von Vine Copulas verwendet. Eine Vine Copula ist eine mehrdimensionale Copula, welche mit Hilfe von mehreren bivariaten Copulas aufgebaut wird. Wenn für diesen Aufbau die verschiedensten bivariaten Copulas verwendet werden, dann können viele unterschiedliche Abhängigkeiten modelliert werden. Vine Copulas sind daher sehr flexibel. In dieser Arbeit sollen die wichtigsten Kenntnisse über R-Vine Copulas zusammengefasst werden, welche in praktischen Anwendungen verwendet werden können. Da Vine Copulas nur für uniform verteilte Daten auf $[0,1]^n$ definiert sind, müssen in den Anwendungen die Daten zuerst auf diese Form transformiert werden.

Diese Arbeit ist folgend aufgebaut. Im ersten Abschnitt wird zuerst allgemein eine Copula Funktion definiert und es wird gezeigt wie man diese als statistisches Werkzeug verwenden kann. Dann werden die wichtigsten Eigenschaften von Copulas näher gebracht und es wird gezeigt wie man eine Copula schätzen kann. Am Ende des ersten Teils werden noch die wichtigsten bivaraten Copula Funktionen näher gebracht, welche dann später für eine R-Vine Konstruktion verwendet werden können.

Im zweiten Teil der Arbeit wird die Idee der Vine Copulas vorgestellt. Es werden auch spezielle Vine Strukturen wie C-Vine und D-Vine etwas näher gebracht, aber der Fokus in dieser Arbeit liegt bei den allgemeinen R-Vines. Es wird dann auch gezeigt, wie man mit Hilfe von Matrizen R-Vine Copulas beschreiben kann und somit alle weiteren Berechnungen mit Hilfe dieser Matrizen durchführen kann. Am Ende dieses Abschnittes soll es dann möglich sein, die "beste" R-Vine Copula an vorgegebenen Daten anzupassen.

Im letzten Teil wird mit Hilfe von R-Vine Copulas der VaR (*Value at Risk*) für ein Portfolio vorhergesagt. Dazu werden zuerst die Log-Renditen modelliert

und deren Residuen werden so transformiert, dass man uniformverteilte Zufallszahlen zwischen 0 und 1 erhält. An die transformierten Daten kann dann eine Vine Copula angepasst werden, welche die Abhängigkeiten beschreibt. Durch Generierung von Zufallszahlen dieser Vine Copula können dann die Residuen geschätzt werden, welche dann bei der Vorhersage des VaR verwendet werden können. [11][9][7]

2 Copulas und deren wichtigsten Eigenschaften

Der Begriff Copula wurde erstmals 1959 von Abe Sklar in der Mathematik bzw. in der Statistik verwendet. Das Wort Copula kommt aus dem Lateinischen und kann als Verknüpfung bzw. Verbindung übersetzt werden. Sklar konnte auch zeigen, dass eine Copula den Zusammenhang von einer mehrdimensionalen Verteilungsfunktion und deren Randverteilungen beschreibt. Copulas werden heutzutage vorallem dazu verwendet, um die Abhängigkeiten von Zufallsvariablen zu untersuchen. In der Literatur waren Schweizer und Wolff (1981) die ersten, die mit Hilfe von Copulas Abhängigkeiten beschrieben. [11]

2.1 Einführung in die grundlegende Theorie der Copulas

Copulas sind Funktionen die den Zusammenhang zwischen den Randverteilungen verschiedener Zufallsvariablen und deren gemeinsamer Verteilung angeben. Um allgemein eine n-dimensionale Copula definieren zu können, wird die Eigenschaft n-steigend verwendet.

Definition 1 (H-Volumen)**.** *Seien $S_1, S_2, ..., S_n$ nichtleere Teilmengen von $\overline{R} = [-\infty, \infty]$ und sei $H : S_1 \times S_2 \times ... \times S_n \to \mathbb{R}$. Weiters sei $B = [\underline{a}, \underline{b}] = [a_1, b_1] \times ... \times [a_n, b_n]$ ein Hyperrechteck mit Eckpunkten $\underline{c} = (c_1, ..., c_n)$ in Dom(H). Dann ist das **H-Volumen** für B gegeben durch:*

$$V_H(B) = \sum (-1)^{N(\underline{c})} H(\underline{c}),$$

wobei die Summe über alle Eckenpunkte $\underline{c}$ aus B läuft und

$$N(\underline{c}) = |\{k | c_k = a_k\}|.$$

Folgendes Beispiel soll die Notation etwas näher bringen.

Beispiel 1. *Sei $H : \overline{R}^3 \to \mathbb{R}$ und $B = [a_1, b_1] \times [a_2, b_2] \times [a_3, b_3]$ ein Hyperrechteck. Dann ist das H-Volumen für B gegeben durch*

$$V_H(B) = H(b_1, b_2, b_3) - H(b_1, b_2, a_3) - H(b_1, a_2, b_3) - H(a_1, b_2, b_3)$$

$$+H(b_1, a_2, a_3) + H(a_1, b_2, a_3) + H(a_1, a_2, b_3) - H(a_1, a_2, a_3).$$

Definition 2 (n-steigend)**.** *Die Funktion H aus Definition 1 heißt **n-steigend**, wenn $V_H(B) \geq 0$ für alle Hyperrechtecke mit Eckpunkten in Dom(H).*

Nun kann eine n-dimensionale Copula Funktion definiert werden.

Definition 3. *Eine* ***n-Copula*** *(n-dimensionale Copula) C ist eine Funktion mit folgenden Eigenschaften:*

1. $C : [0,1]^n \to [0,1]$
2. *Für jedes* $\underline{u} = (u_1, ..., u_n)$ *aus* $[0,1]^n$ *gilt:*
   (a) $C(\underline{u}) = 0$, *wenn mindestens eine Koordinate von* $\underline{u}$ *gleich 0 ist.*
   (b) $C(\underline{u}) = u_k$, *wenn alle Koordinaten bis auf* u_k *gleich 1 sind.*
3. *C ist n-steigend.*

In der Wahrscheinlichkeitstheorie ist die Funktion $C : [0,1]^n \to [0,1]$ eine n-dimensionale Copula, wenn C eine n-dimensionale Verteilungsfunktion mit eindimensionalen uniformverteilten Randverteilungen ist. Das wichtigste Resultat in der Copula Theorie liefert der Satz von Sklar.

Theorem 1 (Satz von Sklar)**.** *Sei H eine n-dimensionale Verteilungsfunktion mit Randverteilungen* $F_1, F_2, ..., F_n$. *Dann existiert eine n-Copula C, so dass für alle* $\underline{x} = (x_1, ..., x_n) \in \overline{R}^n$ *gilt:*

$$H(x_1, x_2, ..., x_n) = C(F_1(x_1), F_2(x_2), ..., F_n(x_n))$$

Wenn alle Randverteilungen stetig sind, dann ist C eindeutig. Umgekehrt, wenn C eine n-dimensionale Copula ist und $F_1, F_2, ..., F_3$ *eindimensionale Verteilungsfunktionen sind, dann ist* $H(x_1, x_2, ..., x_n)$ *eine n-dimensionale Verteilungsfunktion mit Randverteilungen* $F_1, F_2, ..., F_n$.

Zusammengefasst kann man also sagen, dass eine multivariate Verteilungsfunktion mittels Randverteilungen und einer Copula, welche die Abhängigkeiten beschreibt, dargestellt werden kann. Aus diesem Satz lässt sich die *Inverse Methode* herleiten. Mit dieser Methode ist es möglich Copula Funktionen zu konstruieren, wenn eine mehrdimensionale Verteilungsfunktion F mit zugehörigen invertierbaren Randverteilungen $F_1, F_2, ..., F_n$ bekannt ist.

$$C(\underline{u}) = F\left(F_1^{-1}(u_1), ..., F_n^{-1}(u_n)\right)$$

Jede Copula C lässt sich in einen absolut stetigen Teil A_C und einen singulären Teil S_C zerlegen.

$$C(\underline{u}) = A_C(\underline{u}) + S_C(\underline{u}),$$

wobei

$$A_C(\underline{u}) = A_C(u_1, ..., u_n) = \int_0^{u_1} ... \int_0^{u_n} \frac{\partial^n}{\partial v_1 ... \partial v_n} C(v_1, ..., v_n) dv_1 ... dv_n$$

und

$$S_C(\underline{u}) = C(\underline{u}) - A_C(\underline{u}).$$

Besitzt eine Copula C beide Komponenten A_C und S_C, so ist weder A_C noch S_C eine Copula. Eine Copula C heißt absolut stetig, wenn $C \equiv A_C$. In diesem Fall besitzt C eine Dichte, die durch

$$c(\underline{u}) = \frac{\partial^n C(u_1, ..., u_n)}{\partial u_1 ... \partial u_n}$$

gegeben ist.

Als nächstes werden einige Beispiele für Copulas näher gebracht. Zuerst betrachten wir für $\underline{u} \in [0,1]^n$ die Funktionen

- $M(\underline{u}) = \min(u_1, u_2, ..., u_n)$;
- $\Pi(\underline{u}) = \prod_{i=1}^n u_i$;
- $W(\underline{u}) = \max(u_1 + u_2 + ... + u_n - n + 1, 0)$.

Für $n \geq 2$ sind $M^n(\underline{u})$ und $\Pi^n(\underline{u})$ n-Copulas. $\Pi(\underline{u})$ wird Unabhängigkeitscopula (Independence Copula) genannt und $M^n(\underline{u})$ wird als obere Fréchet Hoeffding Schranke bezeichnet. Mit $W^n(\underline{u})$ wird die untere Fréchet Hoeffding Schranke bezeichnet, welche nur für $n = 2$ eine Copula ist. Diese drei Funktionen haben besondere Eigenschaften. Mit Hilfe der Unabhängigkeitscopula kann man Aussagen über die Unabhängigkeit von Zufallsvariablen treffen. Die Fréchet Hoeffding Schranken geben die extremsten Werte an, welche eine Copula annehmen kann. Die folgenden zwei Sätze sollen die Eigenschaften näher bringen.

Theorem 2. *Sei C eine n-Copula, dann gilt für jedes* $\underline{u} \in [0,1]^n$

$$W^n(\underline{u}) \leq C(\underline{u}) \leq M^n(\underline{u}).$$

Theorem 3. *Seien $X_1, X_2, ..., X_n$ stetige Zufallsvariablen und $n \geq 2$. Dann gilt:*

1. *$X_1, X_2, ..., X_n$ sind unabhängig genau dann, wenn die n-Copula von $X_1, X_2, ..., X_n$ gleich Π^n ist.*

2. *Jede der Zufallsvariablen $X_1, X_2, ..., X_n$ ist fast sicher eine streng ansteigende Funktion einer anderen genau dann, wenn die n-Copula von $X_1, X_2, ..., X_n$ gleich M^n ist.*

Im 2-dimensionalen Fall gilt für die Modellierung der Abhängigkeit: Wenn zwei Zufallsvariablen X und Y unabhängig sind, verwendet man die Unabhängigkeitscopula. Wenn die Zufallsvariablen positiv bzw. negativ abhängig sind, verwendet man die obere bzw. untere Fréchet Hoeffding Schranke als Copula.

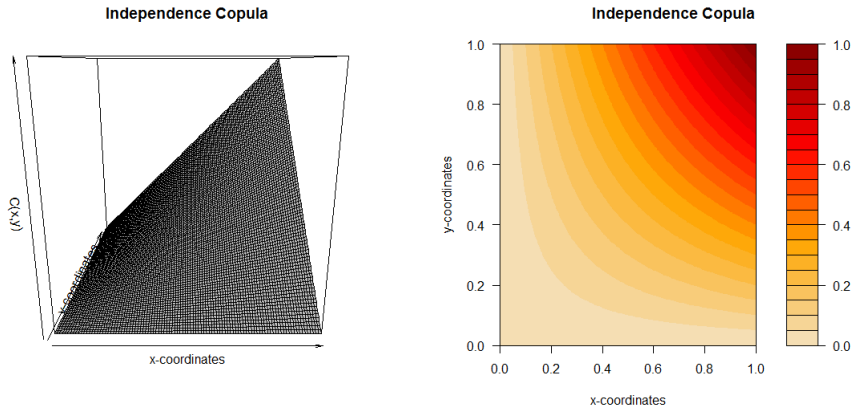

Abbildung 1: Unabhängigkeitscopula Π

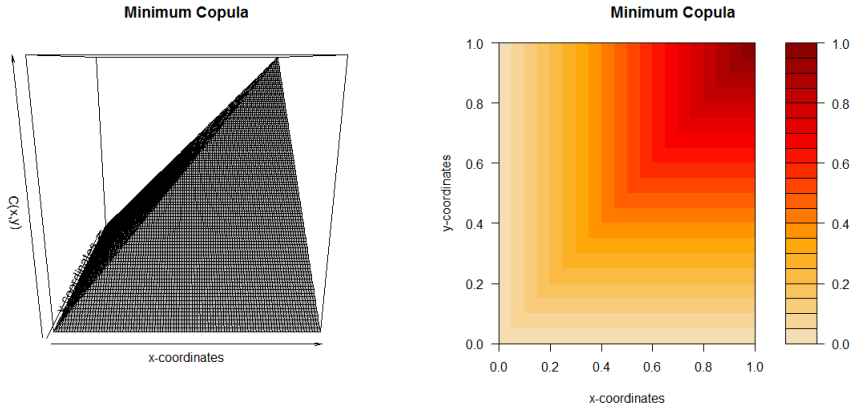

Abbildung 2: Minimum Copula M

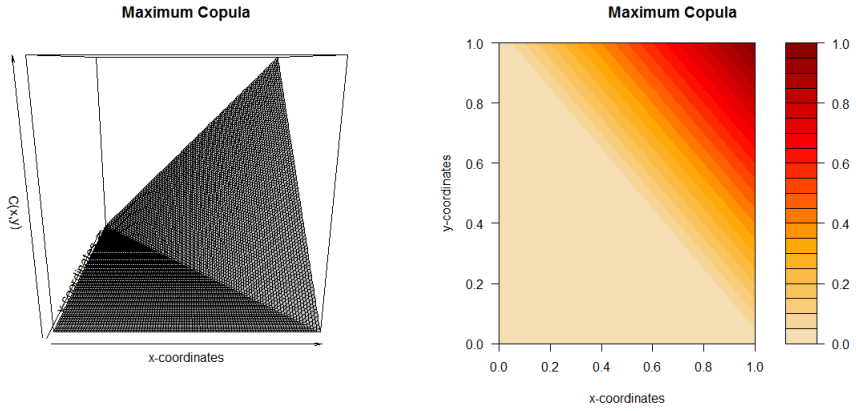

Abbildung 3: Maximum Copula W

Eine weitere wichtige Copula ist die Normal-Copula (Gauß-Copula). Für eine gegebene Korrelationsmatrix $K \in \mathbb{R}^{n \times n}$ lässt sich die Normal-Copula mit Hilfe der *Inversen Methode* folgend herleiten.

$$C_{\mathbf{K}}^{Gauss}(\underline{u}) = \Phi_{\mathbf{K}}(\Phi^{-1}(u_1), ..., \Phi^{-1}(u_n)),$$

wobei mit Φ^{-1} die Inverse der Standardnormalverteilung bezeichnet wird. Die multivariate Normalverteilung mit $\underline{\mu} = \underline{0}$ und Korrelationsmatrix $\mathbf{K}$ ist durch $\Phi_{\mathbf{K}}$ gegeben. Für die Berechnung dieser Copula müssen aber numerische Methoden herangezogen werden. Die t-Copula ist eine weitere elliptische Copula. Diese kann auch mit der *Inversen Methode* berechnet werden.

$$C_{\mathbf{K},\nu}^{t}(\underline{u}) = t_{\nu,\mathbf{K}}(t_{\nu}^{-1}(u_1), ..., t_{\nu}^{-1}(u_n)),$$

wobei mit $t_{\nu,\mathbf{K}}$ und t_ν die multivariate t-Verteilung mit Korrelationsmatrix $\mathbf{K}$ und die eindimensionale t-Verteilung bezeichnet werden. Die Anzahl der Freiheitsgrade ist durch ν gegeben. Man betrachtet auch Klassen von Copulas. Eine davon ist die Familie der Archimedischen Copulas. Die meisten Copulas dieser Klasse können in einer expliziten Form dargestellt werden. Eine Archimedische Copula hat die Form

$$C(\underline{u}) = \varphi^{[-1]}(\varphi(u_1) + ... + \varphi(u_n)),$$

wobei $\varphi : [0,1] \to [0,\infty)$ stetig, streng monoton fallend und konvex ist. Außerdem ist $\varphi(1) = 0$ und die Pseudo-Inverse von φ ist gegeben durch

$$\varphi^{[-1]}(t) = \begin{cases} \varphi^{-1}(t), & 0 \le t \le \varphi(0) \\ 0, & \varphi(0) \le t \le \infty \end{cases}$$

Mit der obigen Formel erhält man eine Copula genau dann, wenn $\varphi^{-1}(t)$ n-monoton ist.

Definition 4. *Eine Funktion g(t) ist* ***n-monoton*** *auf $[a,b]$, wenn g(t) bis zur (n-2) Ordnung differenzierbar ist und für die Ableitungen folgendes gilt:*

- $(-1)^k \frac{d^k}{dt^k} g(t) \ge 0$ *für alle* $t \in [a,b]$ *und k=1,...,(n-2)*
- $(-1)^{(n-2)} \frac{d^{(n-2)}}{dt^{(n-2)}} g(t)$ *ist fallend und konvex*

Die Funktion $\varphi(t)$ wird als Erzeuger (Generator) bezeichnet. Wählt man im 2-dimensionalen Fall als Erzeuger die Funktion $\varphi(t) = (-\log(t))^\theta$ so erhält man die Gumbel Copula mit Parameter $\theta \ge 1$

$$C_\theta(u_1,u_2) = \exp\left(-\left((-\log u_1)^\theta + (-\log u_2)^\theta\right)^{\frac{1}{\theta}}\right).$$

Diese Copula kann vor allem verwendet werden, um Extremwertverteilungen zu modellieren. Die Gumbel Copula erfüllt nämlich die Bedingung einer Extreme Value Copula. [11][7]

$$C(u_1,...,u_n)^k = C(u_1^k,...,u_n^k) \;\; \forall \underline{u} \in [0,1]^n, \;\; \forall k \ge 1$$

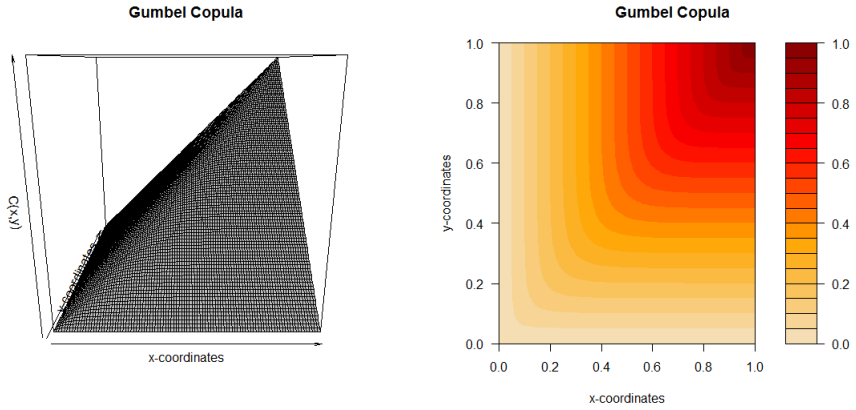

Abbildung 4: Gumbel Copula mit $\theta = 2$

2.2 Abhängigkeitsmaße

Bekannte Abhängigkeitsmaße wie Kendall's Tau τ oder Spearman's Rho ρ lassen sich mittels Copulas berechnen. Es kann oft auch umgekehrt mit Hilfe von solchen Abhängigkeitsmaßen der Parameter einer Copula geschätzt werden. In dieser Arbeit wird vor allem Kendall's Tau für einige Berechnungen verwendet. Um Kendall's Tau für Stichproben definieren zu können, benötigt man die Begriffe *konkordant* und *diskonkordant*. Es seien zwei Beobachtungen (x_i, y_i) und (x_j, y_j) von einem stetigen Zufallsvektor (X, Y) gegeben. Die zwei Paare (x_i, y_i) und (x_j, y_j) nennt man

- *konkordant*, wenn $x_i < x_j$ und $y_i < y_j$ oder wenn $x_i > x_j$ und $y_i > y_j$ $((x_i - x_j)(y_i - y_j) > 0)$.
- *diskonkordant*, wenn $x_i < x_j$ und $y_i > y_i$ oder wenn $x_i > x_j$ und $y_i < y_j$ $((x_i - x_j)(y_i - y_j) < 0)$.

Es sei nun eine Stichprobe $\{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ eines stetigen Zufallsvektors (X, Y) gegeben. Dann gibt es $\binom{N}{2}$ verschieden Paare (x_i, y_i) und (x_j, y_j), welche entweder *konkordant* oder *diskonkordant* sind. Kendall's Tau für Stichproben ist dann definiert durch

$$\tau = \frac{c - d}{c + d} = (c - d) \Big/ \binom{N}{2},$$

wobei mit c die Anzahl der *konkordanten* Paare und mit d die Anzahl der *diskonkordanten* Paare gegeben ist.

Für stetige Zufallsvariablen wird Kendall's Tau ähnlich definiert. Für zwei unabhängige, identisch verteilte Zufallsvektoren (X_1, Y_1) und (X_2, Y_2) ergib sich

$$\tau_{X,Y} = P[(X_1 - X_2)(Y_1 - Y_2) > 0] - P[(X_1 - X_2)(Y_1 - Y_2) < 0].$$

Wenn eine Copula bekannt ist, so kann Kendall's Tau direkt von der Copula berechnet werden. Ein zweites Abhängigkeitsmaß, welches unabhängig von einer Verteilungsannahme berechnet werden kann, ist Spearman's Rho. Dieses kann auch mittels einer Copula bestimmt werden.

Theorem 4. *Es seien X und Y zwei stetige Zufallsvariablen mit Copula C. Dann ist Kendall's Tau gegeben durch*

$$\tau_{X,Y} = \tau_C = 4 \int \int_{[0,1]^2} C(u, v) dC(u, v) - 1.$$

Das Integral kann natürlich auch als Erwartungswert der Funktion C(U,V) aufgefasst werden, wobei U und V (0,1)-uniformverteilte Zufallsvariablen mit gemeinsamer Verteilungsfunktion C sind. Somit ergibt sich

$$\tau_C = 4\mathbb{E}(C(U, V)) - 1.$$

Theorem 5. *Es seien X und Y zwei stetige Zufallsvariablen mit Copula C. Dann ist Spearman's Rho gegeben durch*

$$\rho_{X,Y} = \rho_C = 12 \int \int_{[0,1]^2} C(u, v) du dv - 3.$$

Ein weiteres wichtiges Abhängigkeitsmaß in der Copula Theorie ist *tail dependence*. Mit dieser ist es möglich die Ausreißer einer Verteilung besser zu beschreiben. Im zweidimensionalen Fall beschreibt *tail dependence* die Abhängigkeit zweier Zufallsvariablen im oberen-rechten bzw. unteren-linken Quadranten einer bivariaten Verteilung. Ist der Koeffizient der *tail dependence* gleich 0, so sind X und Y asymptotisch unabhängig.

Definition 5. *Seien X und Y stetige Zufallsvariablen mit Verteilungsfunktionen F und G. Dann ist die obere tail dependence λ_U von X und Y gegeben durch*

$$\lambda_U = \lim_{t \to 1^-} P[Y > G^{-1}(t) | X > F^{-1}(t)],$$

falls der Grenzwert existiert.
Analog ist auch die untere tail dependence λ_L gegeben durch

$$\lambda_L = \lim_{t \to 0^+} P[Y \leq G^{-1}(t) | X \leq F^{-1}(t)],$$

falls der Grenzwert existiert.

Die Koeffizienten λ_U und λ_L hängen nur von der Copula ab und es gilt:

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

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

Sind X und Y unabhängig, dann sind λ_L und λ_U gleich Null. Die Umkehrung gilt im Allgemeinem nicht. [11][10]

2.3 Parameterschätzung von Copulas

Bevor man die Parameter einer Copula schätzt, muss man wissen welcher Copulatyp an die Daten angepasst wird. Im zweidimensionalen Fall kann man dazu den Chi-Plot verwenden, welcher eine graphische Veranschaulichung über die Abhängigkeit zweier Zufallsvariablen gibt. Auch mit einem Streudiagramm kann man einen Einblick in die Daten bekommen, aber es ist schwer zu sehen ob zwei Zufallsvariablen unabhängig sind. Die Idee des Chi-Plots ist es daher, die empirische bivariate Verteilung mit der Nullhypothese, dass jeder Punkt in einem Scatterplot unabhängig ist, zu vergleichen. Um diesen Plot für die Punktepaare $(x_i, y_i), i = 1, ..., N$ zu konstruieren, wird die empirische bivariate Verteilung $\hat{H}$ und deren empirischen Randverteilungen $\hat{F}$ und $\hat{G}$ verwendet. Mit Hilfe der Indikatorfunktion $I(.)$ lassen sich

$$\hat{H}(s,t) = \frac{1}{N} \sum_{=1}^{N} I(x_i \leq s, y_i \leq t),$$

$$\hat{F}(t) = \frac{1}{N} \sum_{i=1}^{N} I(x_i \leq t) \quad \text{und} \quad \hat{G}(t) = \frac{1}{N} \sum_{i=1}^{N} I(y_i \leq t)$$

leicht berechnen. Der Chi-Plot umfasst dann die Chi-Statistik

$$\chi_i = \frac{\hat{H}(x_i, y_i) - \hat{F}(x_i)\hat{G}(y_i)}{\sqrt{\hat{F}(x_i)\left(1 - \hat{F}(x_i)\right)\hat{G}(y_i)\left(1 - \hat{G}(y_i)\right)}}$$

und die Lambda-Statistik

$$\lambda_i = 4 \cdot sign\left(\left(\hat{F}(x_i) - \frac{1}{2}\right)\left(\hat{G}(y_i) - \frac{1}{2}\right)\right) \cdot \max\left(\left(\hat{F}(x_i) - \frac{1}{2}\right)^2 \left(\hat{G}(y_i) - \frac{1}{2}\right)^2\right).$$

Dabei wird mit λ_i der Abstand zwischen (x_i, y_i) und dem Zentrum der Datenmenge gemessen und mit χ_i wird der Abstand zwischen einer Verteilung mit unabhängigen Zufallsvariablen (X, Y) und der Verteilung H gemessen. Unter der Annahme der Unabhängigkeit gilt $\chi_i \sim N\left(0, \frac{1}{N}\right)$ und $\lambda \sim U[-1, 1]$ (asymptotisch). Also Werte χ_i nahe bei 0 versprechen Unabhängigkeit. In einem Chi-Plot können auch Schranken für χ_i hinzugefügt werden. Zum Beispiel ergibt sich für die Schranken $\left(1.54/\sqrt{N}), -1.54/\sqrt{N})\right)$ ein approximatives Signifikanzniveau von 10%. Folgende Abbildung veranschaulicht drei Chi-Plots, welche mit der Funktion `BiCopChiPlot` aus dem Packet `VineCopula` erzeugt wurden. Dazu werden Zufallszahlen einer bivariaten Normalverteilung mit Erwartungswertvektor gleich dem Nullvektor, Varianzen gleich 1 und einer Korrelation ρ gezogen (Packet: `MASS`). Für ρ werden dabei die Werte -0.5, 0 und 0.5 verwendet, um unterschiedliche Abhängigkeiten zu erzeugen. Diese Zufallszahlen werden mit Hilfe der empirischen Verteilungsfunktion auf $[0, 1]$ transformiert, denn für die Funktion `BiCopChiPlot` dürfen nur Werte aus $[0, 1]$ verwendet werden. In Abbildung 5 sieht man in der ersten Grafik viele Werte unter 0 (negative Abhängigkeit) und in der dritten Grafik viele Werte über 0 (positive Abhängigkeit). In der zweiten Grafik liegen die Punkte nahe bei der 0 und somit kann man von unabhängigen Daten ausgehen. Man kann nun die Chi-Plots

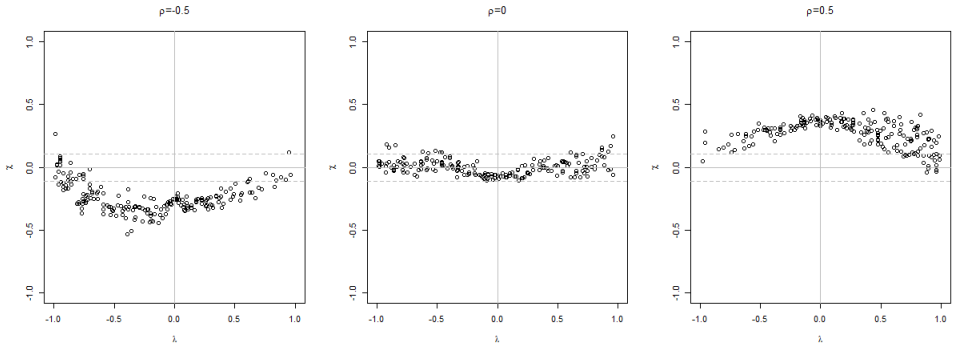

Abbildung 5: Chi-Plots für verschiedene Abhängigkeiten

der Daten mit den Chi-Plots der verschiedenen Copulatypen vergleichen, um so eine passende Copula zu finden. Im späteren Teil der Arbeit wird aber eine weitere Methode vorgestellt, um die Wahl der Copulatypen objektiver zu gestalten.

Nachdem man sich für einen Copulatyp entschieden hat, kann man mit der Copula Schätzung beginnen. Dafür gibt es im Wesentlichen drei verschiedene Vorgehensweisen.

1. parameterfreie Schätzung (empirische Schätzung der Copula)
2. parametrische Schätzung (Parameter der Copula und der Randverteilungen werden geschätzt)
3. semi-parametrische Schätzung (Parameter der Copula werden geschätzt)

Bei der parameterfreien Schätzung werden die Beobachtungen verwendet, um die Copula empirisch zu schätzen. Je größer die Stichprobe ist, desto besser ist die Schätzung.
Wenn die Randverteilungen bekannt sind, so kann die parametrische Schätzung verwendet werden. Dabei können mit Hilfe der Likelihood-Funktion die Parameter der Randverteilungen und Copula zugleich geschätzt werden. Es können aber auch zuerst die Parameter der Randverteilungen geschätzt werden und mit den geschätzten Parametern die Likelihood-Funktion für die Copula aufgestellt werden.
In Anwendungen sind die Randverteilungen meist nicht bekannt, dann wird die semi-parametrische Schätzung verwendet. Dieses Verfahren wird auch in dieser Arbeit verwendet und deshalb etwas näher gebracht. Die Idee dieser Schätzmethode ist es, die Verteilungsfunktionen in der Likelihood-Funktion durch die empirischen Verteilungsfunktionen zu ersetzen. Mit den Stichproben $x_{i,j}; i = 1, ..., N; j = 1, .., n$ ergeben sich die empirischen Randverteilungen

$$\hat{F}_j(t) = \frac{1}{N} \sum_{i=1}^{N} I(x_{i,j} \leq t).$$

Die Beobachtungen können dann auf $[0, 1]$ transformiert werden, wobei mit dem Faktor $\frac{n}{n+1}$ multipliziert wird, um numerische Probleme am Rand von $[0, 1]^n$ zu vermeiden.

$$u_{i,j} = \frac{n}{n+1} \hat{F}(x_{i,j})$$

Dann kann die Pseudo-Log-Likelihood-Funktion maximiert werden, um $\hat{\theta}$ zu schätzen.

$$l(\theta) = \sum_{i=1}^{N} \log c_\theta(u_{i,1}, ..., u_{i,n})$$

Mann kann mit Hilfe der $u_{i,j}$ bei gewissen Voraussetzungen den Parameter einer Copula auch ohne Likelihood-Funktion schätzen. Dazu muss für eine Copula Kendall's Tau (oder Spearman's Rho) in geschlossener Form berechnet werden können und diese Copula darf nur einen Parameter besitzen. Dann kann der Parameter der Copula mit einem geschätzten Kendall's Tau berechnet werden.

Diese Methode hat den großen Vorteil, dass keine Likelihood-Funktion verwendet wird und daher keine numerische Optimierung nötig ist. [11][2][15][8]

2.4 Bivariate Copulas

Die Theorie über bivariate Copulas ist schon weit entwickelt. Diese Copulas werden auch im nächsten Kapitel verwendet, um eine mehrdimensionale Copula zu konstruieren. Dazu benötigt man oft die partiellen Ableitungen.

Theorem 6. *Es sei C eine bivariate Copula. Für jedes $u_2 \in [0,1]$ existiert die partielle Ableitung für fast alle $u_1 \in [0,1]$ und es gilt*

$$0 \leq \frac{\partial}{\partial u_1} C(u_1, u_2) \leq 1.$$

Dasselbe gilt auch wenn man u_1 mit u_2 tauscht.

In der Literatur wird oft die sogenannte *h-Funktion* verwendet, um die Notation zu vereinfachen:

$$h(u_1, u_2) := \frac{\partial C_{u_1,u_2}(u_1, u_2)}{\partial u_2}$$

Bivarite Copulas können auch sehr flexiebel eingesetzt werden. Es sei beispielsweise eine bivariate Copula gegeben, welche nur untere *tail dependence* modellieren kann. Wenn es sich dabei um eine absolut stetige bivariate Copula handelt, kann diese aber rotiert werden. Das heißt, die zugehörige Dichte kann gedreht werden. So können auch andere Quadranten der bivariaten Verteilung besser modelliert werden.

$$\begin{aligned} 90° : &\quad c_{90}(u,v) := c(1-u, v) \\ 180° : &\quad c_{180}(u,v) := c(1-u, 1-v) \\ 270° : &\quad c_{270}(u,v) := c(u, 1-v) \end{aligned}$$

Die um 180° rotierte Copula wird auch Survival Copula genannt. Natürlich kann auch die zugehörige *h-Funktion* rotiert werden. Zum Abschluss dieses Kapitels werden noch einige der wichtigsten bivariaten Copulas zusammengefasst.

2.4.1 Bivariate Normal-Copula

Wie zuvor schon gezeigt wurde, wird die Normal-Copula mit Hilfe der *Inversen Methode* berechnet. Für $n = 2$ ergibt sich

$$C_\rho^{Gauss}(u_1, u_2) = \Phi_\rho(\Phi^{-1}(u_1), \Phi^{-1}(u_2)),$$

wobei Φ_ρ die bivariate Verteilungsfunktion mit Erwartungswertvektor gleich dem Nullvektor, Varianzen gleich 1 und einem Korrelationsparameter ρ ist. Mit Φ^{-1} wird wieder die Inverse der Standardnormalverteilung bezeichnet. Anders geschrieben gilt

$$C_\rho^{Gauss}(u_1, u_2) = \int_{-\infty}^{\Phi^{-1}(u_2)} \int_{-\infty}^{\Phi^{-1}(u_1)} \phi_\rho(x_1, x_2) dx_1 dx_2,$$

wobei

$$\phi_\rho(x_1, x_2) = \frac{1}{2\pi(1-\rho^2)^{\frac{1}{2}}} \cdot \exp\left(-\frac{x_1^2 - 2\rho x_1 x_2 + x_2^2}{2(1-\rho^2)}\right)$$

die Dichte von Φ_ρ ist. In Buch [10] wird gezeigt wie man die *h-Funktion* einer Normal-Copula mit Parameter $\rho \in (-1, 1)$ herleiten kann. Um den Schreibaufwand zu minimieren, wird $q_1 = \Phi^{-1}(u_1)$ und $q_2 = \Phi^{-1}(u_2)$ gesetzt.

$$\begin{aligned}
h(u_1, u_2|Gauss, \rho) &= \frac{\partial}{\partial u_2} C_\rho^{Gauss}(u_1, u_2) \\
&= \frac{\partial}{\partial u_2} \int_{-\infty}^{q_2} \int_{-\infty}^{q_1} \phi_\rho(x_1, x_2) dx_1 dx_2 \\
&= \frac{\partial q_2}{\partial u_2} \frac{\partial}{\partial q_2} \int_{-\infty}^{q_2} \int_{-\infty}^{q_1} \phi_\rho(x_1, x_2) dx_1 dx_2 \\
&= \frac{1}{\phi(q_2)} \int_{-\infty}^{q_1} \phi_\rho(x_1, q_2) dx_1 \\
&= \frac{1}{\phi(q_2)} \int_{-\infty}^{q_1} \frac{1}{2\pi(1-\rho^2)^{\frac{1}{2}}} \cdot \exp\left(-\frac{x_1^2 - 2\rho x_1 q_2 + q_2^2}{2(1-\rho^2)}\right) dx_1 \\
&= \frac{1}{\phi(q_2)} \int_{-\infty}^{q_1} \frac{1}{2\pi(1-\rho^2)^{\frac{1}{2}}} \cdot \exp\left(-\frac{(x_1 - \rho q_2)^2 + (q_2^2 - \rho^2 q_2^2)}{2(1-\rho^2)}\right) dx_1 \\
&= \frac{1}{\phi(q_2)} \cdot \frac{1}{(2\pi)^{\frac{1}{2}}} \cdot \exp\frac{-q_2^2}{2} \int_{-\infty}^{q_1} \frac{1}{(2\pi)^{\frac{1}{2}}(1-\rho^2)^{\frac{1}{2}}} \cdot \exp\left(-\frac{(x_1 - \rho q_2)^2}{2(1-\rho^2)}\right) dx_1 \\
&= \frac{1}{\phi(q_2)} \cdot \phi(q_2) \cdot \Phi\left(\frac{q_1 - \rho q_2}{(1-\rho^2)^{\frac{1}{2}}}\right) = \Phi\left(\frac{q_1 - \rho q_2}{(1-\rho^2)^{\frac{1}{2}}}\right)
\end{aligned}$$

Im späteren Teil der Arbeit wird auch die Inverse der *h-Funktion* verwendet. Für die Normal-Copula lässt sich diese mittels der Quantilfunktion der Standardnormalverteilung berechnen.

$$h^{-1}(u_1, u_2|Gauss, \rho) = \Phi\left(\Phi^{-1}(u_1) \cdot (1-\rho^2)^{\frac{1}{2}} + \rho q_2\right)$$

Die Dichte der bivarite Normal-Copula ist durch

$$c_\rho^{Gauss}(u_1, u_2) = \frac{1}{(1-\rho^2)^{\frac{1}{2}}} \cdot \exp\left(-\frac{\rho^2(q_1^2 + q_2^2) - 2\rho q_1 q_2}{2(1-\rho^2)}\right)$$

gegeben. [11][10][2][5]

2.4.2 Bivariate t-Copula

Die bivariate t-Copula kann auch mit Hilfe der *Inversen Methode* berechnet werden. Diese Copula besitzt zwei Parameter, den Korrelationsparameter $\rho \in (-1, 1)$ und die Freiheitsgrade $\nu > 0$. Die zur t-Copula

$$C^t_{\rho,\nu}(u_1, u_2) = t_{\rho,\nu}(t^{-1}_\nu(u_1), t^{-1}_\nu(u_2))$$

zugehörige Dichte ist gegeben durch

$$c^t_{\rho,\nu}(u_1, u_2) = \frac{\Gamma(\frac{\nu+2}{2})/\Gamma(\frac{\nu}{2})}{\nu\pi dt_\nu(x_1)dt_\nu(x_2)(1-\rho^2)^{\frac{1}{2}}} \left(1 + \frac{x_1^2 + x_2^2 - 2\rho x_1 x_2}{\nu(1-\rho^2)}\right)^{-\frac{\nu+2}{2}},$$

wobei $x_1 = t^{-1}_\nu(u_1)$ und $x_2 = t^{-1}_\nu(u_2)$. Dabei ist t_ν die eindimensionalen t-Verteilung mit ν Freiheitsgraden und dt_ν die zugehörige Dichte. Für $\nu \to \infty$ gilt $C^t_{\rho,\nu} = C^{Gauss}_\rho$. [11][2][5]
Als nächstes werden einige bivariate Archimedische Copulas vorgestellt.

2.4.3 Gumbel Copula

Die zuvor kennengelernte Gumbel Copula mit Parameter $\theta \geq 1$

$$C_\theta(u_1, u_2) = \exp\left(-\left((-\log u_1)^\theta + (-\log u_2)^\theta\right)^{\frac{1}{\theta}}\right)$$

besitzt die Dichte

$$c_\theta(u_1, u_2) = \frac{C(u_1, u_2)}{u_1 u_2}\left((-\log u_1)^\theta + (-\log u_2)^\theta\right)^{2/\theta-2}(\log u_1 \log u_2)^{\theta-1}$$
$$\cdot\left(1 + (\theta-1)\left((-\log u_1)^\theta + (-\log u_2)^\theta\right)^{-1/\theta}\right).$$

Mit der Gumbel Copula ist es nur möglich positive Abhängigkeiten zu modellieren. [2][5][1]

2.4.4 Clayton Copula

Mit dem Generator $\varphi(t) = \frac{1}{\theta}\left(t^{-\theta} - 1\right)$ ergibt sich die Clayton Copula mit Parameter $\theta \in (-1, \infty)\backslash\{0\}$. Für $\theta \in (0, \infty)$ kann wieder nur positive Abhängigkeit modelliert werden. Für $\theta \in [-1, 0)$ kann es bei den Berechnungen numerische Probleme geben. Deshalb empfiehlt es sich in den Anwendungen θ auf $(0, \infty)$

zu beschränken. [11][2][5] Die Clayton Copula

$$C_\theta(u_1, u_2) = (u_1^{-\theta} u_2^{-\theta} - 1)^{-\frac{1}{\theta}}$$

besitzt die Dichte

$$c_\theta(u_1, u_2) = (1+\theta)(u_1 u_1)^{-1-\theta}(u_1^{-\theta} u_2^{-\theta} - 1)^{-\frac{1}{\theta}-2}.$$

2.4.5 Frank Copula

Die Frank Copula wird mit dem Generator $\varphi(t) = -\log\left(\frac{e^{-\theta t}-1}{e^{-\theta}-1}\right)$ erzeugt. Mit dieser Copula ist es möglich sowohl negative als auch positive Abhängigkeiten zu modellieren. [11][2][5] Die Frank Copula mit $\theta \in \mathbb{R}\backslash\{0\}$

$$C_\theta^F(u_1, u_2) = -\frac{1}{\theta}\log\left(1 + \frac{(e^{-\theta u_1}-1)(e^{-\theta u_2}-1)}{e^{-\theta}-1}\right)$$

besitzt die Dichte

$$c_\theta^F(u_1, u_2) = \frac{\theta(e^{-\theta}-1)\cdot e^{-\theta(u_1+u_2)}}{\left(e^{-\theta}-1+(e^{-\theta u_1}-1)(e^{-\theta u_2}-1)\right)^2}$$

2.4.6 Weitere Archimedische Copulas

Eine weitere Archimedische Copula ist die Joe Copula mit dem Generator $\varphi(t) = -\log\left(1-(1-t)^\theta\right)$ und $\theta > 1$. Eine zweiparametrige Copula kann mit dem Generator $\varphi(t) = (t^{-\theta_1}-1)^{\theta_2}$ erzeugt werden, wobei $\theta_1 > 0$ und $\theta_2 \geq 1$. Diese wird als Clayton-Gumbel Copula bezeichnet, denn für $\theta_2 = 1$ erhält man die Clayton Copula und für $\theta_1 \to 0$ die Gumbel Copula. Eine weitere Sammlung von Archimedischen Copulas ist beispielsweise im Buch [11] aufgelistet. [11][10][2][5][1]

3 Vine Copulas

Der Begriff Vine ist in der Mathematik Ende 1990 zum ersten Mal publiziert worden. Dabei handelt es sich um eine graphische Struktur, welche eine mehrdimensionale Verteilungsfunktion beschreibt. Anfangs waren Vines nicht so sehr beliebt, aber in den letzten Jahren wuchs das Interesse speziell an Vine Copulas immer mehr.

3.1 Einführung in Vine Copulas

Um Abhängigkeiten von mehreren Zufallsgrößen zu modellieren, können mehrdimensionale Copulas verwendet werden. Als einfachste Wahl wird oft die Normal Copula verwendet. Diese hat aber den Nachteil, dass sie für die Modellierung von tail dependence unbrauchbar ist. Ein Ausweg hierfür liefert die t-Copula, welche größere Ausläufer beschreibt. Es können aber auch Archimedische Copulas für höher dimensionale Probleme verwendet werden. Solche multivariaten Copulas sind sehr beliebt, aber sie sind oft nicht so flexibel. Einen Ausweg hierfür geben Vine Copulas, welche auch PCC (Pair Copula Construction) genannt werden. Die Idee besteht darin ein Modell mit mehreren zweidimensionalen Copulas, welche durch eine Vine Struktur angeordnet werden, zu bestimmen.

Der Aufbau von Vine Copulas soll nun näher gebracht werden. Dazu werden die Zufallsvariablen $\underline{X} = (X_1, ..., X_n)$ mit gemeinsamer Dichtefunktion $f_{1:n}(x_1, ..., x_n) = f_{1:n}(\underline{x})$ betrachtet. Diese Dichte kann rekursiv in bedingte Dichten umgeschrieben werden.

$$f_{1:n}(x_1, ..., x_n) = f_1(x_1) \cdot f_{2|1}(x_2|x_1) \cdot f_{3|2,1}(x_3|x_1, x_2) \cdot ...$$

$$... \cdot f_{n|1:(n-1)}(x_n|x_1, ..., x_{n-1}) \quad (*)$$

Diese Darstellung ist bis auf Permutation der x_i, $i = 1, ..., n$ eindeutig. Die Idee ist es nun mit Hilfe vom Satz von Sklar die multivariate Dichte $f_{1:n}$ in bivariate Copula Dichten und die Dichten der eindimensionalen Randverteilungen zu zerlegen.
Mit dem Satz von Sklar wird gezeigt, dass jede mehrdimensionale Verteilungsfunktion F durch deren eindimensionalen Randverteilungen und eine Copula C beschrieben werden kann.

$$F_{1:n}(x_1, ..., x_n) = C_{1:n}(F_1(x_1), ..., F_n(x_n))$$

Mit partieller Differenzierung erhält man dann die Dichtefunktion

$$f_{1:n}(x_1, ..., x_n) = c_{1:n}(F_1(x_1), ..., F_n(x_n)) \cdot f_1(x_1) \cdot ... \cdot f_n(x_n).$$

Im zweidimensionalen Fall vereinfacht sich der Ausdruck auf

$$f_{1,2}(x_1, x_2) = c_{1,2}(F_1(x_1), F_2(x_2)) \cdot f_1(x_1) \cdot f_2(x_2).$$

Nun kann man die bedingte Dichte von X_1 bei gegebenen $X_2 = x_2$ folgend darstellen werden.

$$f_{1|2}(x_1|x_2) = \frac{f_{1,2}(x_1, x_2)}{f_2(x_2)} = c_{1,2}(F_1(x_1), F_2(x_2)) \cdot f_1(x_1)$$

Schlussendlich kann man Schritt für Schritt alle bedingten Dichten in der rechten Seite der Gleichung $(*)$ in bivariate Copula Dichten und Randdichten zerlegen.
Die bedingte Dichte $f_{3|2,1}(x_3|x_1, x_2)$ kann man zerlegen in

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

oder in

$$f_{3|2,1}(x_3|x_1, x_2) = c_{3,1|2}(F_{3|2}(x_3|x_2), F_{1|2}(x_1|x_2)) \cdot f_{3|2}(x_3|x_2).$$

Allgemein kann man zeigen, dass für $X \notin \underline{X} = (X_1, ..., X_n)$

$$f_{X|\underline{X}}(x|\underline{x}) = c_{X,X_j|\underline{X}_{-j}} \left(F_{X|\underline{X}_{-j}}(x|\underline{x}_{-j}), F_{X_j|\underline{X}_{-j}}(x_j|\underline{x}_{-j}) \right) \cdot f_{X|\underline{X}_{-j}}(x|\underline{x}_{-j})$$

gilt, wobei $\underline{X}_{-j}$ gleich dem Vektor $\underline{\mathrm{X}}$ ohne der j-ten Komponente X_j ist. Die Funktion $F_{X|\underline{X}_{-j}(x|\underline{x}_{-j})}$ ist dabei die bedingte Verteilungsfunktion von X bei gegebenem $\underline{X}_{-j} = \underline{x}_{-j}$, welche an der Stelle x ausgewertet wird.
Um die Notation etwas zu vereinfachen, werden in dieser Arbeit oft folgende Abkürzungen verwendet:

$$\begin{aligned}
&F_{i|i_1,...,i_r} := F_{i|i_1,...,i_r}(x_i|x_{i_1}, ..., x_{i_r}) \text{ bzw.} \\
&F(x_i|x_{i_1}, ..., x_{i_r}) := F_{i|i_1,...,i_r}(x_i|x_{i_1}, ..., x_{i_r}), \\
&f_{i|i_1,...,i_r} := f_{i|i_1,...,i_r}(x_i|x_{i_1}, ..., x_{i_r}) \text{ bzw.} \\
&f(x_i|x_{i_1}, ..., x_{i_r}) := f_{i|i_1,...,i_r}(x_i|x_{i_1}, ..., x_{i_r}),
\end{aligned}$$

wobei $i \neq j$ und $i_1, ..., i_r$ alle verschieden. Die Copula wird daher oft auch mit

$$C_{i,j|i_1,...,i_r} := C_{i,j|i_1,...,i_r}\left(F_{i|i_1,...,i_r}(x_i|x_{i_1}, ..., x_{i_r}), F_{j|i_1,...,i_r}(x_j|x_{i_1}, ..., x_{i_r})\right)$$

abgekürzt. Dasselbe gilt auch für die Dichte der Copula.

Beispiel 2. *Nun soll an Hand eines Beispiels die Zerlegung der Dichtefunktion näher gebracht werden. Um das Beispiel einfach zu halten, wird $n = 4$ gewählt.*

$$f(x_1, x_2, x_3, x_4) = f(x_1) \cdot f(x_2|x_1) \cdot f(x_3|x_1, x_2) \cdot f(x_4|x_1, x_2, x_3)$$

Die bedingten Dichtefunktionen können nun folgend zerlegt werden:

$$\begin{aligned}
f(x_2|x_1) &= c_{2,1} \cdot f(x_2) \\
f(x_3|x_1, x_2) &= c_{3,2|1} \cdot f(x_3|x_1) \\
&= c_{3,2|1} \cdot c_{3,1} \cdot f(x_3) \\
f(x_4|x_1, x_2, x_3) &= c_{4,2|1,3} \cdot f(x_4|x_1, x_3) \\
&= c_{4,2|1,3} \cdot c_{4,3|1} \cdot f(x_4|x_1) \\
&= c_{4,2|1,3} \cdot c_{4,3|1} \cdot c_{4,1} \cdot f(x_4)
\end{aligned}$$

Zusammengefasst erhält man Resultat (a)

$$\begin{aligned}
f(x_1, x_2, x_3, x_4) = \quad & c_{4,2|1,3} \cdot c_{4,3|1} \cdot c_{3,2|1} \cdot c_{4,1} \cdot c_{3,1} \cdot c_{2,1} \\
& \cdot f(x_4) \cdot f(x_3) \cdot f(x_2) \cdot f(x_1).
\end{aligned}$$

Man muss aber für $f(x_3|x_1, x_2)$ im ersten Schritt nicht $c_{3,2|1}$ wählen. Man könnte auch $c_{3,1|2}$ wählen. So lässt sich dann ein weiteres Resultat (b) herleiten.

$$\begin{aligned}
f(x_1, x_2, x_3, x_4) = \quad & c_{4,1|2,3} \cdot c_{4,2|3} \cdot c_{3,1|2} \cdot c_{4,3} \cdot c_{3,2} \cdot c_{2,1} \\
& \cdot f(x_4) \cdot f(x_3) \cdot f(x_2) \cdot f(x_1).
\end{aligned}$$

Mit der Anzahl der Variablen steigt auch die Anzahl der verschieden Wahlmöglichkeiten.

Die bedingten Verteilungen, welche in den Argumenten der Copulas vorkommen, können mittels Lower-Level-Copulas berechnet werden. Betrachtet man beispielsweise $c_{4,2|1,3}$, so sind die Lower-Level-Copulas für das Argument $F_{X_2|X_1,X_3}(x_2|x_1, x_3)$ gleich $c_{2,3|1}$, $c_{3|1}$ und $c_{2|1}$. Es gilt nämlich

$$F_{X_2|X_1,X_3}(x_2|x_1, x_3) = \frac{\partial C_{2,3|1}\left(F_{X_2|X_1}(x_2|x_1), F_{X_3|X_1}(x_3|x_1)\right)}{\partial F_{X_3|X_1}(x_3|x_1)},$$

wobei $F_{X_2|X_1}(x_2|x_1)$ und $F_{X_3|X_1}(x_3|x_1)$ ähnlich dargestellt werden können unter der Verwendung von $c_{3|1}$ und $c_{2|1}$. Die obige Gleichung kann auch allgemein aufgeschrieben werden, so dass alle Argumente der Copula rekursiv berechnet werden können,

$$F_{X|\underline{X}}(x|\underline{x}) = \frac{\partial C_{X,X_j|\underline{X}_{-j}}\left(F_{X|\underline{X}_{-j}}(x|\underline{x}_{-j}), F_{X_j|\underline{X}_{-j}}(x_j|\underline{x}_{-j})\right)}{\partial F_{X_j|\underline{X}_{-j}}(x_j|\underline{x}_{-j})}$$

für $X \notin \underline{X} = (X_1, ..., X_n)$ und $\underline{X}_{-j} = \underline{X} \backslash X_j$. Ziel ist es eine passende Zerlegung der Dichtefunktion zu finden, so dass die Argumente der bivariaten Copulas mittels der zuvor verwendeten Copulas berechnet werden können. Dieses Problem kann mit Hilfe von Graphen gelöst werden, welche eine sogenannte *proximity condition* erfüllen müssen. [10][9]

3.2 Graphische Darstellung von Vine Copulas

Vine Copulas haben den großen Vorteil, dass man die Abhängigkeiten graphisch darstellen kann. Betrachtet man das Resultat (b) von Beispiel 2, so erinnert diese Graphik sehr an Trauben bzw. an einen Traubenstock. Daher auch der Name Vine Copulas. In höheren Dimensionen wird solch eine Darstellung

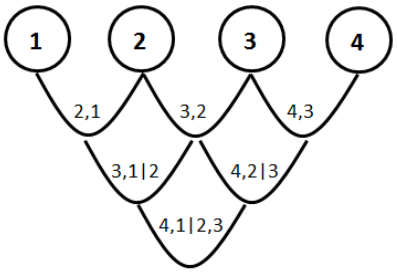

Abbildung 6: Resultat (b) von Beispiel 2

sehr unübersichtlich. Deshalb werden Folgen von Graphen (Bäumen) verwendet, um die Struktur von Vine Copulas darzustellen. Dazu benötigt man einige Hilfsmittel aus der Graphentheorie, welche jetzt wiederholt werden. Ein Tupel $G = (N, E)$ wird als ungerichteter Graph bezeichnet, wobei N eine Menge ist und E eine Teilmenge aller zweielementigen Teilmengen von N ist.

$$E \subset \{\{n_1, n_2\} | n_1, n_2 \in N\}$$

Die Elemente von N werden als Knoten und die Elemente aus E als Kanten bezeichnet. Ein Graph mit $N = \{n_1, ..., n_k\}$ und $E = \{\{n_1, n_2\}, \{n_2, n_3\}, ..., \{n_{k-1}, n_k\}\}$

für $k \geq 2$ wird als Pfad bezeichnet. Ein Pfad wird Kreis genannt, wenn der Anfangsknoten gleich dem Endknoten ist, $n_1 = n_k$. Ein Graph heißt zusammenhängend, wenn je zwei beliebigen Knoten mit einem Pfad verbunden werden können. Ein Baum ist dann ein zusammenhängender, kreisfreier und ungerichteter Graph.

Nun ist es möglich Folgen von Bäumen zu definieren, welche die Struktur von Vine Copulas darstellen. Später wird noch gezeigt, dass es spezielle Vine Strukturen (C Vine und D Vine) gibt, deshalbt wird die allgemeine Vine Struktur auch als Regular Vine (R-Vine) bezeichnet.

Definition 6 (Regular Vine Tree Sequence)**.** $\mathcal{V} = (T_1, ..., T_{n-1})$ *für n Elemente wird* ***Regular Vine Tree Sequence*** *genannt, wenn*

1. T_1 *ist ein Baum mit Knoten* $N_1 = \{1, ..., n\}$ *und einer Kantenmenge* E_1.
2. *Für* $j \geq 2$ *ist* T_j *ein Baum mit Knoten* $N_j = E_{j-1}$ *und Kantenmenge* E_j
3. *Für* $j = 1, ..., n-1$ *und* $\{a, b\} \in E_j$ *muss* $|a \cap b| = 1$ *gelten.*

Die 3. Bedingung wird *proximity condition* genannt. Diese garantiert, dass eine Kante e zwei Knoten a und b in $T_j, j > 2$ nur dann verbindet, wenn a und b in T_{j-1} einen gemeinsamen Knoten haben. Die Resultate (a) und (b) aus Beispiel 2 können nun als Folgen von Bäumen dargesetellt werden.

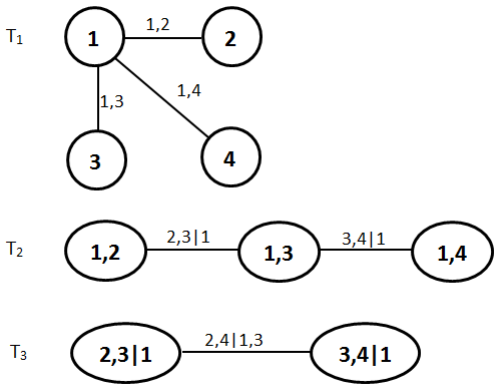

Abbildung 7: Resultat (a) von Beispiel 2 als Folge von Bäumen dargestellt

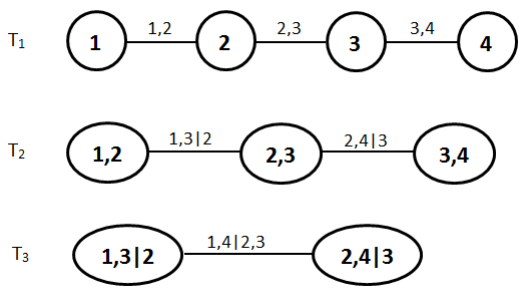

Abbildung 8: Resultat (b) von Beispiel 2 als Folge von Bäumen dargestellt

In den obigen Graphiken sind die Indizes der verwendeten Copulas in aufsteigender Reihenfolge gegeben, um die Notation einfach zu halten. Die Indizes für die Copulas können für jeden Baum leicht berechnet werden. Dazu werden in der Literatur folgende Mengen verwendet. Die Menge

$$A_e := \{j \in N_1 | \exists e_1 \in E_1, ..., e_{i-1} \in E_{i-1} : j \in e_1 \in ... \in e_{i-1} \in e\},$$

bei gegebener Kante e, wird *complete union* genannt. Für eine Kante $e = \{a, b\}$ wird

$$D_e := A_a \cap A_b$$

als *conditioning set* bezeichnet. Die Menge C_e für eine Kante e wird *conditioned set* genannt.

$$C_e := C_{e,a} \cup C_{e,b} \text{ mit } C_{e,a} := A_a \backslash D_e, \quad C_{e,b} := A_b \backslash D_e.$$

Als Beispiel ergeben sich für die Kante $e = 1, 3|2$ in T_2 vom Resultat (b) die Mengen:

$$A_e = \{1, 2, 3\}, \quad A_a = \{1, 2\} \quad A_b = \{2, 3\} \quad D_e = \{1, 3\} \quad C_e = \{2\}$$

Folgerung: Es sei $\mathcal{V} = (T_1, ..., T_{n-1})$ eine R-Vine Struktur, dann gilt:

1. Die Anzahl der Kanten ist gleich $n(n-1)/2$.
2. Jede *conditioned set* enthält genau zwei Elemente.
3. Wenn zwei Kanten dieselbe *conditioning set* besitzen, so sind diese gleich.

Bis jetzt ist nur die Struktur von Vine Copulas näher gebracht worden. Als nächstes wird allgemein eine R-Vine Verteilung definiert.

Definition 7. *Es seien $X_1, ..., X_n$ Zufallsvariablen mit gemeinsamer Verteilung F. Diese wird dann R-Vine Verteilung genannt, wenn es ein Tupel $(\mathcal{F}, \mathcal{V}, B)$ gibt, sodass:*

1. ***Marginal Distributions:*** *$\mathcal{F} = (F_1, ..., F_n)$ ist ein Vektor mit stetig, invertierbaren Verteilungsfunktionen, welche die Randverteilungen von $X_1, ..., X_n$ beschreiben.*

2. ***Regular Vine Tree Sequence:*** *$\mathcal{V}$ ist eine Folge von Bäumen, die eine R-Vine Struktur beschreibt.*

3. ***Bivariate Conditional Distributions:***

$$B = \{B_e | i = 1, ..., n-1; e \in E_i\},$$

   wobei B_e eine symmetrische bivariate Copula mit Dichte ist. Mit E_i werden die Kantenmengen der R-Vine Struktur des Baumes T_i bezeichnet.

4. ***Connection Between Tree Sequence And Bivariate (Conditional) Distributions:*** *Für jedes $e = \{a, b\} \in E_i, i = 1, ..., n-1$ ist B_e eine zu den bedingten Verteilungen von $X_{C_{e,a}}$ und $X_{C_{e,b}}$, bei gegebenen $\underline{X}_{D_e} = \underline{x}_{D_e}$, zugehörige Copula. Die Copula B_e hängt nicht von $\underline{x}_{D_e}$ ab.*

Betrachtet man nun wieder die Kante $e = 1, 3|2$ in T_2 vom Resultat (b) aus Beispiel 2, so ergibt sich die bedingte Verteilungsfunktion von (X_1, X_3) bei gegebenem $X_2 = x_2$ durch

$$F_{X_1, X_3|X_2}(x_1, x_2|x_3) = B_e \left(F_{X_1|X_2}(x_1|x_2), F_{X_3|X_2}(x_3|x_2) \right).$$

Von nun an wird die Copula B_e, wie schon im vorigen Abschnitt, mit $C_{C_{e,a}, C_{e,b}|D_e}$ bezeichnet. Dasselbe gilt auch für die zugehörige Dichte $c_{C_{e,a}, C_{e,b}|D_e}$.

Bemerkung: Die Vine Struktur ist mit Hilfe von ungerichteten Graphen erklärt worden. Deshalb ist die R-Vine Verteilung nur für symmetrische Copulas definiert worden. In der Literatur wird das üblich so gemacht, um die Notation zu vereinfachen. In den Anwendungen können aber auch nicht symmetrische Copulas verwendet werden, wie man dann später bei den R-Vine Matrizen sehen wird.

Wenn umgekehrt ein Tupel $(\mathcal{F}, \mathcal{V}, B)$ existiert, welches die ersten drei Bedingungen aus Definition 7 erfüllt, so kann man eine zugehörige Verteilungsfunktion F finden, wie folgender Satz zeigt.

Theorem 7. *Sei $(\mathcal{F}, \mathcal{V}, B)$ ein Tupel, welches die Bedingungen (1) − (3) aus Definition 7 erfüllt. Dann existiert eine eindeutige Verteilung mit Dichte*

$$f_{1,...,n} = f_1 \cdot ... \cdot f_n \cdot \prod_{i=1}^{n-1} \prod_{e \in E_i} c_{C_{e,a}, C_{e,b}|D_e} \left(F_{C_{e,a}|D_e}, F_{C_{e,b}|D_e} \right),$$

sodass für alle $e = \{a, b\} \in E_i, i = 1, ..., n-1$ die Verteilung von $X_{C_{e,a}}$ und $X_{C_{e,b}}$ bei gegebenen $\underline{X}_{D_e}$ durch

$$F(x_{C_{e,a}}, x_{C_{e,b}}|\underline{x}_{D_e}) = B_e \left(F(x_{C_{e,a}}|\underline{x}_{D_e}), F(x_{C_{e,b}}|\underline{x}_{D_e}) \right)$$

gegeben ist. Außerdem sind die eindimensionalen Randverteilungen durch $F(x_i) = F_i(x_i), i = 1, ..., n$ gegeben.

Morales-Nápoles (2010) konnte zeigen, dass es für den n-dimensionalen Fall $n!/2 \cdot 2^{\binom{n-2}{2}}$ verschiedene R-Vines gibt. In vielen Anwendungen werden aber meist nur zwei Klassen von R-Vines betrachtet.

Definition 8. *Sei $\mathcal{V} = (T_1, ..., T_{n-1})$ eine Vine Struktur, dann heißt $\mathcal{V}$*

- ***D-Vine**, wenn für alle Knoten $n \in N_i$*

  $$|\{e \in E_i | n \in e\}| \leq 2$$

  gilt.

- ***C-Vine**, wenn es in jedem Baum T_i ein $n \in N_i$ gibt, sodass*

  $$|\{e \in E_i | n \in e\}| = n - i$$

  gilt.

Das bedeutet, dass in einem D-Vine alle Bäume gleich Pfade sind. Das Resultat (b) aus Beispiel 2 ist daher ein D-Vine. Die Bäume von C-Vines nennt man in der Graphen Theorie auch Sterne. Das Resultat (a) aus Beispiel 2 ist ein C-Vine. [10][9] Die Dichte einer C-Vine Verteilung kann in der Form

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

angegeben werden. Eine Dichte der D-Vine Verteilung kann dargestellt werden in der Form

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

3.3 R-Vine Matrizen

Die graphische Darstellung von R-Vines gibt eine gute Übersicht, aber für Berechnungen empfiehlt sich eine kompaktere Schreibweise. Die Idee ist es alle Informationen von R-Vines in einer Dreiecksmatrix zu speichern. In der Literatur werden entweder untere oder obere Dreiecksmatrizen verwendet. In dieser Arbeit werden ausschließlich untere Dreiecksmatrizen verwendet, um die Notation einheitlich zu halten. Um eine R-Vine Matrix zu definieren, werden zwei Mengen benötigt, mit deren Hilfe die *proximity condition* gesichert werden kann. Für eine untere Dreiecksmatrix $\mathbf{M} = (m_{i,j})_{i,j=1,...,n}$ defininere für $j = 1, ..., n-1$ die Mengen

$$B_M(j) := \{(m_{j,j}, D) | i = j+1, ..., n; D = \{m_{i,j}, ..., m_{n,j}\}\}$$

und

$$\widetilde{B}_M(j) := \{(m_{i,j}, D) | i = j+1, ..., n; D = \{m_{j,j}\} \cup \{m_{i+1,j}, ..., m_{n,j}\}\}.$$

Definition 9 (R-Vine Matrix)**.** *Die untere Dreiecksmatrix $M = (m_{i,j})_{i \leq j}$ wird als **R-Vine Matrix** bezeichnet, wenn für jedes $j = 1, ..., n-1$ und $i = j+1, ..., n$ ein $k \in \{j+1, ..., n-1\}$ existiert, sodass*

$$(m_{i,j}, \{m_{i+1,j}, ..., m_{n,j}\}) \in B_M(k) \;\; oder \;\; \in \widetilde{B}_M(k).$$

Diese Bedingung ist das Gegenstück zur *proximity condition* der R-Vine Bäume. Mit dieser Bedingung kann man die zwei folgenden Eigenschaften herleiten.

1. Alle Elemente einer ausgewählten Spalte sind in jeder Spalte links davon enthalten.

$$\{m_{i,i}, ..., m_{n,i}\} \subset \{m_{j,j}, ..., m_{n,j}\}, \quad 1 \leq j < i \leq n$$

2. Das Diagonalelement einer Spalte ist in keiner Spalte rechts davon enthalten.

$$m_{i,i} \notin \{m_{i+1,j}, ..., m_{n,j}\}, \quad 1 \leq j < i \leq n$$

Außerdem können noch zwei weitere einfache Eigenschaften für R-Vine Matrizen mit der Definition 9 gezeigt werden.

- Alle Elemente einer Spalte sind verschieden

- Wenn man die erste Spalte und erste Zeile einer n-dimensionalen R-Vine Matrix löscht, so erhält man eine (n-1)-dimensionale R-Vine Matrix.

Im Buch [9] werden zwei Algorithmen vorgestellt. Der erste Algorithmus generiert eine R-Vine Matrix aus den vorgegebenen R-Vine Bäumen. Der zweite Algorithmus geht den umgekehrten Weg. An Hand des Resultats (a) aus Beispiel 2 sollen diese beiden Algorithmen etwas näher gebracht werden.

Man betrachte die *conditioned set* der Kante des letzten Baumes $T_n = T_3$ und wählt eines der beiden Elemente (2 oder 4). Das gewählte Element (z.B.:4) wird in der Matrix M an der Stelle $m_{1,1}$ gespeichert. Als nächstes betrachtet man in allen Bäumen alle Elemente, die mit dem ausgewählten Element (4) gemeinsam in einer *conditioned set* vorkommen. Die erste Spalte der Matrix wird dann mit diesen Elementen aufgefüllt. Dabei muss man beachten, dass Elemente aus dem Baum T_{n-i+1} in die i-te Zeile kommen. Das Zwischenergebnis wird in der Abbildung 9 dargestellt. Alle zuvor verwendeten Kanten und alle isolierten Knoten

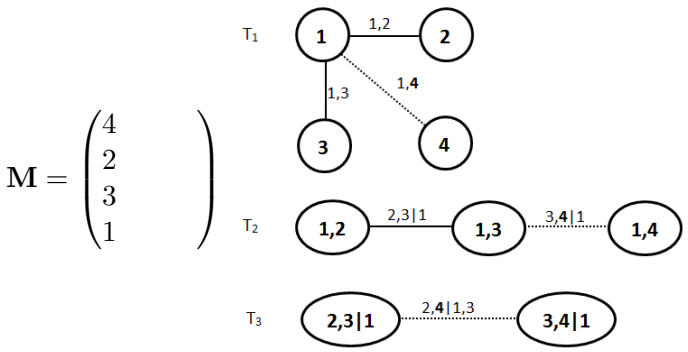

Abbildung 9: Resultat (a) von Beispiel 2 wird zu einer R-Vine Matrix umgeschrieben (Teil 1).

werden dann gelöscht. Das sind alle Knoten und Kanten, in denen das zuvor ausgewählte Element (4) vorkommt. Die R-Vine Struktur besitzt dann einen Baum weniger. In der neuen Struktur betrachtet man dann wieder die Kante vom letzten Baum. Man kann sich wieder ein Element der *conditioned set* dieser Kante aussuchen, welches in $m_{2,2}$ gespeichert wird. Nach den gleichen Regeln wie zuvor füllt man die zweite Spalte auf. Das zweite Zwischenergebnis wird in der Abbildung 10 dargestellt.

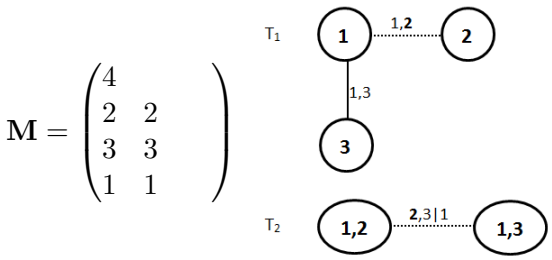

Abbildung 10: Resultat (a) von Beispiel 2 wird zu einer R-Vine Matrix umgeschrieben (Teil 2).

Man wiederholt diese Vorgänge bis zur vorletzten Spalte. Zum Schluss bleibt genau ein Element übrig, welches noch nicht als Diagonalelement ausgewählt wurde. Dieses wird in $m_{n,n}$ gespeichert. Das Endergebnis ist dann gegeben durch:

$$\mathbf{M} = \begin{pmatrix} 4 & & & \\ 2 & 2 & & \\ 3 & 3 & 3 & \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Aus dieser Matrix kann man alle $n(n-1)/2$ Copulas

$$C_{m_{j,j},m_{i,j}|m_{i+1,j},\ldots,m_{n,j}}$$

herauslesen, wobei $j = 1, ..., n-1$, $i > j$ und der bedingte Teil $m_{i+1,j}, ..., m_{n,j}$ für $i = n$ wegfällt. In der ersten Spalte kann man $C_{4,1}$, $C_{4,3|1}$ und $C_{4,2|1,3}$ herauslesen. Von nun an kann man auch mit nicht symmetrische Copulas arbeiten, indem man festlegt, dass ein Diagonalelement der Matrix M immer als erstes Argument in der Copula vorkommt.

Ein zweiter Algorithmus gibt den umgekehrten Weg an. Mit dessen Hilfe kann man mittels einer R-Vine Matrix die R-Vine Bäume konstruieren. Zuerst werden alle Elemente als Knoten in Baum T_1 gezeichnet. Man betrachtet dann die vorletzte Spalte der R-Vine Matrix. Die beiden Elemente dieser Spalte, Diagonalelement (3) und Element der letzten Zeile (1), werden dann in T_1 verbunden. Immer wenn eine Kante in einem Baum $T_i, i = 1, ..., n-1$ eingezeichnet wird, wird sofort ein Knoten, mit der selben Bezeichnung, in T_{i+1} hinzugefügt (d.h.: Baum T_2 erhält den Knoten $1, 3$). Das Zwischenergebnis wird in Abbildung 11 zusammengefasst.

$$\mathbf{M} = \begin{pmatrix} 4 & & & \\ 2 & 2 & & \\ 3 & 3 & \mathbf{3} & \\ 1 & 1 & \mathbf{1} & 1 \end{pmatrix}$$

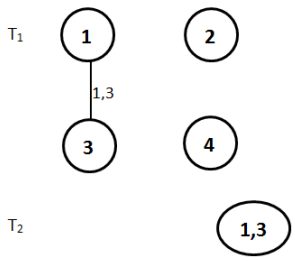

Abbildung 11: R-Vine Matrix wird in R-Vine Bäumen umgeschrieben (Teil 1).

Als nächstes wird die Spalte $(n-2)$ betrachtet. Wieder wird das Hauptdiagonalelement (2) mit dem Element in der letzten Zeile (1) verbunden und ein neuer Knoten (1,2) wird in T_2 hinzugefügt. Dann werden die zwei Knoten in T_2 mit der Kante $m_{2,2}, m_{3,2}|m_{4,2} = 2,3|1$ verbunden, siehe Abbildung 12.

$$M = \begin{pmatrix} 4 & & & \\ 2 & \mathbf{2} & & \\ 3 & \underline{\mathbf{3}} & 3 & \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

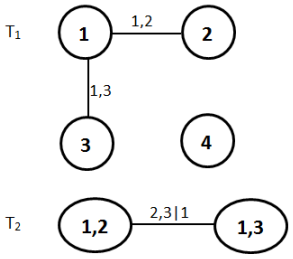

Abbildung 12: R-Vine Matrix wird in R-Vine Bäumen umgeschrieben (Teil 2).

Diese Schritte werden solange wiederholt, bis man schlussendlich dieselbe Folge von Graphen wie in Abbildung 6 erhält. Allgemein wird nämlich von jeder Spalte $j, j = 1, ..., n-1$ die Kante $m_{j,j}, m_{n-i+1,j}|m_{n-i+2,j}, ..., m_{n,j}$ für $i > j$ im Baum $T_i, i = 1, ..., n$ hinzugefügt, wobei der bedingte Teil $m_{n-i+2,j}, ..., m_{n,j}$ für $i = n$ wegfällt.

Die R-Vine Matrizen für die Spezialfälle, D-Vine und C-Vine, können durch Permutation von $(1, ..., n)$ auf folgende Formen gebracht werden. [10][9][6]

$$\mathbf{M}_{(C-Vine)} = \begin{pmatrix} n & & & & \\ n-1 & n-1 & & & \\ n-2 & n-2 & n-2 & & \\ \vdots & \vdots & \vdots & \ddots & \\ 1 & 1 & 1 & \dots & 1 \end{pmatrix}$$

$$\mathbf{M}_{(D-Vine)} = \begin{pmatrix} n & & & & & \\ 1 & n-1 & & & & \\ 2 & 1 & n-2 & & & \\ 3 & 2 & 1 & n-3 & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & \\ n-1 & n-2 & n-3 & n-4 & \dots & 1 \end{pmatrix}$$

3.4 Anzahl der R-Vines

In höheren Dimensionen erhält man eine große Anzahl an verschiedenen R-Vines. Mit Hilfe von R-Vine Matrizen kann man sich aber einen guten Überblick über diese R-Vines machen. Dazu wird eine natürliche Ordnung eingeführt. Das bedeutet, dass alle Variablen indiziert werden. Die beiden Variablen, die in der *condtioned set* in der Kante des letzten Baums vorkommen, bekommen die Indizes n und $n-1$. Die Variable mit dem Index $n-1$ muss noch in einer weiteren *condtioned set* vorkommen und die zweite Variable dieser Menge erhält dann den Index $n-2$. Diese Schritte werden iteriert, bis alle Variablen indiziert worden sind. Für eine R-Vine Matrix in natürlicher Ordnung gilt dann $m_{i,i} = n-i+1, i = 1, ..., n$ und $m_{i+1,i} = m_{i+1,i+1}, i = 1, ..., n-1$. Im Buch [9] wird ein Algorithmus vorgestellt, mit welchem es dann möglich ist die verschiedenen R-Vine Matrizen in natürlicher Ordnung zu konstruieren. Dieser Algorithmus erzeugt aber obere Dreiecksmatrizen. Der folgende `R`-Code, welcher den Algorithmus näher bringen soll, ist deshalb für untere Dreiecksmatrizen angepasst worden. Als Eingabe wird eine untere Dreiecksmatrix **B** benötigt, wobei nur die Einträge $b_{i,j}, j = 1, ..., n-3$ und $i = j+2, ..., n$ interessant sind. Diese Einträge nehmen nur die Werte 0 oder 1 an. Die restlichen Werte der unteren Dreiecksmatrix werden mit 1 aufgefüllt. Der leicht veränderte Algorithmus ist in `R` folgend implementiert worden.

```
> createM<-function(B){
+ n=nrow(B)
+ A=diag(seq(n,1,-1))
+ for(i in 2:n){A[i,i-1]=A[i,i]}
+ A[n,n-2]=A[n,n]
+ for(d in (n-3):1){
+ ac<-A[d+2,d+2]
+ a<-seq(1,ac,1)
+ for(j in (d+2):n){
+ if(B[j,d]==1){
```

```
+ A[j,d]=A[n-ac+1,n-ac+1]
+ a=setdiff(a,ac)
+ ac=max(a,0)
+ }else{
+ A[j,d]=A[j+1,n-ac+1]
+ a=setdiff(a,A[j+1,n-ac+1])
+ }
+ }
+ }
+ return(A)
+ }
```

Für $n = 3$ gibt es nur einen R-Vine in natürlicher Ordnung. Für $n = 4$ kann in der Matrix $\mathbf{B}$ der Eintrag $b_{3,1}$ entweder den Wert 0 oder 1 annehmen. Mit 1 erhält man eine C-Vine Struktur und mit 0 eine D-Vine Struktur. Für $n = 5$ können in der Matrix $\mathbf{B}$ mehrere Veränderungen vorgenommen werden. In der vierten Spalte kann wieder nur ein Element zwischen 0 und 1 gewählt werden, aber in der fünften Spalte sind es zwei Elemente. Man erhält also $2 \cdot 2 \cdot 2 = 2^3 = 8$ verschiedene R-Vines in natürlicher Ordnung für $n = 5$. Es lässt sich zeigen, dass es in der n-ten Dimension $2^{\binom{n-2}{2}}$ verschiedene R-Vines in natürlicher Ordnung gibt. Betrachtet man nun die natürliche Ordnung, so kann man erkennen, dass die Variablen der Indizes n und $n-1$ vertauscht werden können. Man hat also $\binom{n}{2}$ Möglichkeiten diese Variablen zu wählen. Für die restlichen $n-2$ Variablen gibt es dann $(n-2)!$ Möglichkeiten. Es gibt also insgesamt $\frac{n!}{2} \cdot 2^{\binom{n-2}{2}}$ verschiedene R-Vines.
Um die Verschiedenen R-Vines in natürlicher Ordnung zu unterscheiden, gibt es eine Möglichkeit diese zu benennen. Dazu wird die dazugehörige Matrix $\mathbf{B}$ verwendet und es werden die Spalten von rechts nach links betrachtet. Der Name der R-Vine Matrix für $n > 3$ beginnt mit C, falls $b_{n-1,n-3} = 1$ oder mit D, falls $b_{n-1,n-3} = 0$. Für jede weitere Spalte $j, j = n-4, ..., 1; n > 4$ wird der binäre Code $(b_{n,j}, ..., b_{j+3,j})$ in eine Dezimalzahl umgewandelt und zum Namen hinzugefügt. Außerdem werden Punkte zwischen den verschiedene Dezimalzahlen und dem ersten Buchstaben eingefügt. Für $n = 5$ erhält man $D.0$, $D.1$, $D.2$, $D.3$, $C.0$, $C.1$, $C.2$ und $C.3$. Allgemein werden für jede Dimension die D-Vines mit $D.0$, $D.0.0$, $D.0.0.0$ usw. benannt und die C-Vines werden mit $C.3$, $C.3.7$, $C.3.7.15$ usw. benannt. Wie man in dieser Notation schon vermuten kann, sind D-Vine und C-Vine die zwei Extremfälle einer R-Vine Struktur. Würde man obere Dreiecksmatrizen verwenden, müsste man die Spalten von B von links nach rechts lesen, um dieselben Namen zu bekommen. Der obige Algorithmus

wird nun verwendet, um zwei verschieden R-Matrizen in natürlicher Ordnung für $n = 5$ zu generieren.

```
> B1
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    1    1    0    0    0
[3,]    0    1    1    0    0
[4,]    1    0    1    1    0
[5,]    1    1    1    1    1
> B2
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    1    1    0    0    0
[3,]    0    1    1    0    0
[4,]    0    1    1    1    0
[5,]    1    1    1    1    1
> createM(B1)
     [,1] [,2] [,3] [,4] [,5]
[1,]    5    0    0    0    0
[2,]    4    4    0    0    0
[3,]    2    3    3    0    0
[4,]    3    1    2    2    0
[5,]    1    2    1    1    1
> createM(B2)
     [,1] [,2] [,3] [,4] [,5]
[1,]    5    0    0    0    0
[2,]    4    4    0    0    0
[3,]    2    3    3    0    0
[4,]    1    2    2    2    0
[5,]    3    1    1    1    1
```

Dabei sind die R-Vine Matrizen $D.2$ und $C.0$ generiert worden. Es handelt sich dabei um verschiedene R-Vine Matrizen, aber sie sind äquivalent. Zwei R-Vines V und U für n Variablen sind äquivalent, wenn es eine Permutation $\pi \in n!$ gibt, sodass $\pi(V) = U$. Das bedeutet $\{i, j|k, ..., m\}$ ist ein Knoten von V genau dann wenn $\{\pi(i), \pi(j)|\pi(k), ..., \pi(m)\}$ ein Knoten von U ist. Permutiert man in $D.2$ die Variablen $2 \rightleftarrows 3$ und $4 \rightleftarrows 5$, so erhält man dieselbe R-Vine Struktur wie für $C.0$. Im Buch [9] wird eine Formel angegeben, welche die Anzahl der

Äquivalenzklassen der R-Vines in der n-ten Dimension angibt.

$$E_n = (N_n + E_{1n})/2,$$

wobei

$$E_{1n} = \sum_{k=1}^{[n/2]-1} N_n \cdot l_k \cdot 2^{-k} \cdot 2^{-\sum_{i=0}^{k-1}(n-4-2i)}, N_n = 2^{(n-2)(n-3)/2}$$

und $l_k = 1$ für alle $k \neq [n/2] - 1$, denn $l_{[n/2]-1} = 2$. Für $n = 5$ ergeben sich 6 Äquivalenzklassen. Zwei Äquivalenzklassen sind immer D-Vine und C-Vine. Außerdem gibt es in jeder Dimension $\frac{n!}{2}$ D-Vines und $\frac{n!}{2}$ C-Vines. [9]

3.5 R-Vine Copula Spezifikation mittels Matrizen

In diesem Abschnitt soll nun gezeigt werden, wie es möglich ist eine R-Vine Copula mit Hilfe von Matrizen zu beschreiben. Wie bereits bekannt ist, kann eine R-Vine Struktur in einer Matrix $\mathbf{M}$ gespeichert werden. Aus dieser Matrix kann man auch die $n(n-1)/2$ bivariaten Copulas herauslesen. Nun werden noch zwei weitere Matrizen eingeführt, um die Parameter und Typen der zu verwendenden Copulas zu speichern. Hierfür werden wieder zwei untere Dreiecksmatrizen $\mathbf{T}$ und $\mathbf{P}$ verwendet.

$$\mathbf{T} = \begin{pmatrix} 0 & & & & & \\ t_{2,1} & 0 & & & & \\ t_{3,1} & t_{3,2} & 0 & & & \\ t_{4,1} & t_{4,2} & t_{4,3} & 0 & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & \\ t_{n,1} & t_{n,2} & t_{n,3} & \dots & t_{n,n} & 0 \end{pmatrix}$$

$$\mathbf{P} = \begin{pmatrix} 0 & & & & & \\ p_{2,1} & 0 & & & & \\ p_{3,1} & p_{3,2} & 0 & & & \\ p_{4,1} & p_{4,2} & p_{4,3} & 0 & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & \\ p_{n,1} & p_{n,2} & p_{n,3} & \dots & p_{n,n} & 0 \end{pmatrix}$$

In der Matrix $\mathbf{T}$ werden die Typen der $n(n-1)/2$ Copulas gespeichert. Die dazugehörigen Parameter können in der Matrix $\mathbf{P}$ abgelesen werden. Natürlich gibt es Copulas mit mehr als einen Parameter, dann können zum Beispiel zusätzliche Matrizen für die restlichen Parameter verwendet werden. Die Theorie in dieser

Arbeit wird aber nur mit einer Parametermatrix $\mathbf{P}$ nähergebracht, denn mit mehreren Matrizen wäre nur der Schreibaufwand größer.

In [6] ist ein Algorithmus angegeben, welcher die Dichte einer R-Vine Copula für $(x_1, ..., x_n)$ berechnet. Der Algorithmus benötigt als Eingabe die Matrizen $\mathbf{M}$, $\mathbf{T}$ und $\mathbf{P}$. Die Diagonalelemente der Matrix $\mathbf{M}$ müssen dabei von n bis 1 angeordnet werden, sodass $m_{i,j} = n - i + 1$. Man kann jede Matrix auf diese Form bringen, wenn man die Knoten der R-Vine Bäume dementsprechend umbenennt. Dies hat aber keinen Einfluss auf die Matrizen $\mathbf{T}$ und $\mathbf{P}$. Um in jedem Schritt die richtigen Argumente für die bivariaten Copulas zu finden, wird eine weitere Matrix $\mathbf{M}^{max}$ eingeführt. Ein Element $m_{i,j}^{max}$ dieser Matrix ist das Maximum aller Elemente der j-ten Spalte von M, welche in den Zeilen i bis n vorkommen. Also $\mathbf{M}^{max} = (m_{i,j}^{max})_{i,j=1,...,n}$ mit $m_{i,j}^{max} = \max\{m_{i,j}, ..., m_{n,j}\}$, für $j = 1, ..., n$ und $i = j, ..., n$. Man beachte, dass die letzte Zeile der Matrix $\mathbf{M}$ mit der letzten Zeile der Matrix $\mathbf{M}^{max}$ übereinstimmt. Außerdem ist das Diagonalelement der Matrix $\mathbf{M}$ immer das größte Element der Spalte, wegen der zuvor gewählten Anordnung und der Definition von R-Vine Matrizen.

Der folgende Algorithmus aus [6] berechnet die Dichte einer R-Vine Copula. Die h-Funktion für die Copula $t_{i,j}$ bei gegebenem Parameter $p_{i,j}$ wird hier mit $h(.,.|t_{i,j}, p_{i,j})$ bezeichnet. Außerdem werden 2 Matrizen $\mathbf{V}^{direkt}$ und $\mathbf{V}^{indirekt}$ verwendet, um die Argumente der Copulas zu speichern.

Algorithmus 1 : Dichte einer R-Vine Spezifikation

Input : Matrizen $\mathbf{M}$, $\mathbf{T}$ und $\mathbf{P}$ für die R-Vine Spezifikation, wobei $m_{i,i} = n - i + 1$ für $i = 1, .., n$.

Output : Dichte einer R-Vine Verteilung an der Stelle $(x_1, ..., x_n)$ bei gegebener R-Vine Spezifikation.

Setze $F = 1$;
Es sei $\mathbf{V}^{direkt} = (v_{i,j}^{direkt} | i, j = 1, ..., n)$;
Des Weiteren sei $\mathbf{V}^{indirekt} = (v_{i,j}^{indirekt} | i, j = 1, ..., n)$;
Setze $(v_{n,1}^{direkt}, v_{n,2}^{direkt}, ..., v_{n,n}^{direkt}) = (F_n(X_n), F_{n-1}(X_{n-1}), ..., F_1(X_1))$;
Es sei $\mathbf{M}^{max} = (m_{i,j} | i, j = 1, ..., n)$ mit $m_{i,j}^{max} = \max\{m_{i,j}, ..., m_{n,j}\}$ für alle $j = 1, ..., n$ und $i = j, ..., n$;
for $j = n - 1, ..., 1$ **do**
    for $i = n, ..., j + 1$ **do**
        Setze $z_{i,j}^{(1)} = v_{i,j}^{direkt}$;
        if $m_{i,j}^{max} = m_{i,j}$ **then**
            Setze $z_{i,j}^{(2)} = v_{i,(n-m_{i,j}^{max}+1)}^{direkt}$
        else
            Setze $z_{i,j}^{(2)} = v_{i,(n-m_{i,j}^{max}+1)}^{indirekt}$
        end
        Setze $F = F \cdot c(z_{i,j}^{(1)}, z_{i,j}^{(2)} | t_{i,j}, p_{i,j})$;
        Setze $v_{i-1,j}^{direkt} = h(z_{i,j}^{(1)}, z_{i,j}^{(2)} | t_{i,j}, p_{i,j})$;
        Setze $v_{i-1,j}^{indirekt} = h(z_{i,j}^{(2)}, z_{i,j}^{(1)} | t_{i,j}, p_{i,j})$;
    end
end
return F

Das erste Argument einer Copula $t_{i,j}$ wird immer von der Matrix $\mathbf{V}^{direkt}$ entnommen. Das zweite Element wird von der Matrix $\mathbf{V}^{direkt}$ entnommen, falls $m_{i,j} = m_{i,j}^{max}$, ansonsten von der Matrix $\mathbf{V}^{indirekt}$. Diese zwei Matrizen können in der letzten Zeile die gleichen Elemente $F_n(X_n), ..., F_1(X_1)$ gespeichert haben. In diesem Algorithmus wird hierfür aber nur die Matrix $\mathbf{V}^{direkt}$ verwendet, weil $m_{n,j} = m_{n,j}^{max} \forall j \in \{1, ..., n\}$ und somit immer diese Matrix gewählt wird. Die erste for-Schleife durchläuft alle Spalten, bis auf die letzte Spalte, von links nach rechts. Die zweite for-Schleife iteriert über die Zeilen, wobei sie immer in der letzten Zeile startet und in der Zeile vor dem Diagonalelement endet. Nun wird gezeigt, dass die Argumente $z_{i,j}^{(1)}$ und $z_{i,j}^{(2)}$ für die bivariaten Copulas richtig gewählt werden. Es wird behauptet, dass

$$z_{i,j}^{(1)} = F_{m_{j,j}|\{m_{i+1,j},...,m_{n,j}\}}(x_{m_{j,j}}|x_{m_{i+1,j}},...,x_{m_{n,j}})$$

und

$$z_{i,j}^{(2)} = F_{m_{i,j}|\{m_{i+1,j},...,m_{n,j}\}}(x_{m_{i,j}}|x_{m_{i+1,j}},...,x_{m_{n,j}})$$

für $j = n-1,...,1$ und $i = n,...,j+1$. Diese Behauptung wird mittels Induktion gezeigt. Der Induktionsanfang wird für $i = n$ und k beliebig aus $1,...n$ gezeigt. Da die letzte Zeile von $\mathbf{V}^{direkt}$ mit den Werten $F_n(X_n), F_{n-1}(X_{n-1}),...,F_1(X_1)$ aufgefüllt ist, gilt

$$z_{n,j}^{(1)} = v_{n,j}^{direkt} = F_{n-j+1}(x_{n-j+1}) = F_{m_{j,j}}(x_{j,j}).$$

Des Weiteren gilt wegen $m_{n,j} = m_{n,j}^{max}$ auch

$$z_{n,j}^{(2)} = v_{n,n-m_{n,j}+1}^{direkt} = F_{m_{n,j}}(x_{n,j})$$

und somit gilt der Induktionsanfang. Es wird nun angenommen, dass für alle $n \geq i > I$ und für alle $k = i,...,n$

$$v_{i-1,j}^{direkt} = F_{m_{j,j}|m_{i,j},m_{i+1,j},...,m_{n,j}}(x_{m_{j,j}}|x_{m_{i,j}},x_{m_{i+1,j}},...,x_{m_{n,j}})$$

und

$$v_{i-1,j}^{indirekt} = F_{m_{i,j}|m_{j,j},m_{i+1,j},...,m_{n,j}}(x_{m_{i,j}}|x_{m_{j,j}},x_{m_{i+1,j}},...,x_{m_{n,j}})$$

gilt. Betrachtet man nun den I-ten Schritt, so wählt der Algorithmus für $z_{I,j}^{(1)} = v_{I,j}^{direkt}$. Durch die Induktionsannahme ergibt sich dann

$$v_{I,j}^{direkt} = F_{m_{j,j}|m_{I+1,j},...,m_{n,j}}(x_{m_{j,j}}|x_{m_{I+1,j}},...,x_{m_{n,j}}),$$

und somit wird für $z_{I,j}^{(1)}$ der richtige Wert gewählt.

Nun wird der schwierigere Teil, die Wahl des zweiten Arguments, bewiesen. Aus der Definition von R-Vine Matrizen ist bekannt, dass es ein $k \in \{i+1,...,n-1\}$ gibt, sodass

$$(m_{I,j},\{m_{I+1,j},...,m_{n,j}\}) \in B_M(k) \cup \widetilde{B}_M(k)$$

gilt. Es sei nun $(x,D) \in B_M(k)$, dann beinhalten x und D nur Elemente aus der k-ten Spalte und daraus folgt $\max\{x,\max D\} = m_{k,k}$. Dies gilt auch für $(x,D) \in \widetilde{B}_M(k)$ und deshalb gilt

$$m_{I,j}^{max} = \max(m_{I,j},\{m_{I+1,j},...,m_{n,j}\}) = \max\{B_M(k) \cup \widetilde{B}_M(k)\} = m_{k,k}.$$

Da die Diagonalelemente der Größe nach absteigend angeordnet sind ($m_{k,k} = n - k + 1$), ergibt sich $k = n - m_{I,j}^{max} + 1$. Dies erklärt die Indizierung von $v_{i,n-m_{i,j}^{max}+1}^{direkt}$ und $v_{i,n-m_{i,j}max+1}^{indirekt}$ im Algorithmus. Man betrachtet nun beide Fälle

$$(m_{I,j}, \{m_{I+1,j}, ..., m_{n,j}\}) \in B_M(k)$$

und

$$(m_{I,j}, \{m_{I+1,j}, ..., m_{n,j}\}) \in \widetilde{B}_M(k).$$

Für $(m_{I,j}, \{m_{I+1,j}, ..., m_{n,j}\}) \in B_M(k)$ gilt laut Definition

$$(m_{I,j}, \{m_{I+1,j}, ..., m_{n,j}\}) = (m_{k,k}, \{m_{I+1,k}, ..., m_{n,k}\}) \in B_M(k).$$

Daraus folgt $m_{I,j} = m_{k,k} = m_{I,j}^{max}$ und deshalb wird im Algorithmus die Bedingung $m_{I,j} = m_{I,j}^{max}$ geprüft. Wenn diese Bedingung erfüllt wird, ergibt sich für das zweite Argument der Copula $z_{I,j}^{(2)} = v_{I,n-m_{I,j}^{max}+1}^{direkt} = v_{i,k}^{direkt}$. Mit der Induktionsannahme folgt

$$z_{I,j}^{(2)} = F_{m_{k,k}|m_{I+1,k},m_{I+2,k},...,m_{n,k}}(x_{m_{k,k}}|x_{m_{I+1,k}}, x_{m_{I+2,k}}, ..., x_{m_{n,k}})$$

und weil $(m_{I,j}, \{m_{I+1,j}, ..., m_{n,j}\}) = (m_{k,k}, \{m_{I+1,k}, ..., m_{n,k}\})$, gilt schlussendlich die Behauptung

$$z_{I,j}^{(2)} = F_{m_{I,j}|m_{I+1,j},m_{I+2,j},...,m_{n,j}}(x_{m_{I,j}}|x_{m_{I+1,j}}, x_{m_{I+2,j}}, ..., x_{m_{n,j}}).$$

Für den zweiten Fall $(m_{I,j}, \{m_{I+1,j}, ..., m_{n,j}\}) \in \widetilde{B}_M(k)$ gilt wieder laut Definition

$$(m_{I,j}, \{m_{I+1,j}, ..., m_{n,j}\}) = (m_{I+1,k}, \{m_{k,k}, m_{I+2,k}, ..., m_{n,k}\}) \in \widetilde{B}_M(k).$$

Daraus lässt sich

$$m_{I,j} = m_{I+1,k} < m_{k,k} = m_{I,j}^{max}$$

herleiten. Die Bedingung $m_{I,j} = m_{I,j}^{max}$ wird in diesem Fall nicht erfüllt und es ergibt sich für das zweite Argument der Copula $z_{I,j}^{(2)} = v_{I,n-m_{I,j}^{max}+1}^{indirekt} = v_{i,k}^{direkt}$. Durch die Induktionsannahme ergibt sich dann

$$z_{I,j}^{(2)} = F_{m_{I+1,k}|m_{k,k},m_{I+2,k},...,m_{n,k}}(x_{m_{I+1,k}}|x_{m_{k,k}}, x_{m_{I+2,k}}, ..., x_{m_{n,k}})$$

und wegen $(m_{I,j}, \{m_{I+1,j}, ..., m_{n,j}\}) = (m_{I+1,k}, \{m_{k,k}, m_{I+2,k}, ..., m_{n,k}\})$ gilt auch die Behauptung für den zweiten Fall.

Der behandelte Algorithmus findet also die richtigen Argumente der Copulas und fügt alle bivariaten Copulas zusammen, um die Dichte von einer R-Vine Copula darzustellen. Für eine C-Vine Matrix stimmen $\mathbf{M}$ und $\mathbf{M}^{max}$ überein und deshalb wird die Bedingung $m_{i,j} = m_{i,j}^{max}$ im Algorithmus immer erfüllt. In diesem Fall wird die Matrix $\mathbf{V}^{indirekt}$ nicht verwendet. Für eine D-Vine Matrix dagegen, wird die Bedingung $m_{i,j} = m_{i,j}^{max}$ außer für die letzte Zeile nie erfüllt. Die zweiten Argumente der Copulas werden immer aus $\mathbf{V}^{indirekt}$ entnommen. Man kann also auch in diesem Algorithmus erkennen, dass C-Vines und D-Vines die Extremfälle von R-Vines sind. [5][6]

3.6 Log-Likelihood-Funktion einer R-Vine Copula mittels Matrizen

Algorithmus 1 kann leicht modifiziert werden, um den Wert der Log-Likelihood-Funktion einer R-Vine Copula an der Stelle $(x_1, ..., x_n)$ mit Hilfe der Matrizen $\mathbf{M}$, $\mathbf{T}$ und $\mathbf{P}$ zu berechnen. Dazu wird im Algorithmus F durch L ersetzt und in der ersten Zeile wird L auf 0 gesetzt. Die entscheidende Veränderung findet in der Zeile 14 statt. Diese Zeile wird auf "$L = L + \log c(z_{i,j}^{(1)}, z_{i,j}^{(2)} | t_{i,j}, p_{i,j})$" geändert. Die Log-Likelihood-Funktion kann dann beispielsweise verwendet werden, um die Parameter der bivariaten Copulas zu schätzen, wobei dies numerisch gelöst werden muss. Diese Methode wird auch als gemeinsame Likelihood-Schätzung bezeichnet.

Es gibt aber noch eine zweite Methode, um die Parameter einer R-Vine Copula zu schätzen. Diese ist in der Literatur unter dem Namen *sequential estimation* bekannt. Hierbei wird auch Likelihood-Schätzung verwendet, aber jeder Baum der R-Vine Struktur wird nacheinander abgearbeitet. Das bedeutet im ersten Schritt betrachtet man nur die Copulas im ersten Baum und schätzt separat deren Parameter. Mit Hilfe der *h-Funktion* werden die Daten transformiert, um mit diesen die Parameter der Copulas für den zweiten Baum separat zu schätzen. Man wiederholt diese Schritte bis alle Parameter der Copulas geschätzt worden sind. Bei dieser Prozedur müssen nur Parameter von bivariaten Copulas geschätzt werden und deshalb ist dieses Verfahren schneller als die Schätzung mit der gemeinsamen Likelihood-Funktion. Natürlich verspricht diese Methode kein globales Optimum, aber die Schätzungen sind gute Startwerte für die gemeinsame Likelihood-Schätzung.

Mit einer kleinen Ergänzung in Algorithmus 1 kann auch *sequential estimation* verwendet werden. Man muss nur vor Zeile 14 eine neue Zeile einfügen,

in welcher die Parameter $p_{i,j}$ der Copula geschätzt werden. Die Schätzung ist abhängig von den Werten $z_{i,j}^{(1)}$, $z_{i,j}^{(2)}$ und vom Copulatyp $t_{i,j}$. [5][6]

3.7 Zufallszahlen von R-Vine Copulas ziehen

Ziel ist es, Zufallszahlen von R-Vine Copulas zu ziehen. Dazu wird angenommen, dass alle eindimensionalen Randverteilungen uniform sind. Will man dann auch Zufallszahlen mit beliebigen eindimensionalen Randverteilungen generieren, so muss man die gezogenen Zufallszahlen der R-Vine Copula transformieren. Diese Transformation ist durch die Inversen der Randverteilungen gegeben.

$$x_1 := F_1^{-1}(u_1), x_1 := F_2^{-1}(u_2), ..., x_1 := F_n^{-1}(u_n)$$

Bedford und Cooke (2001) waren die ersten die sich mit dem generieren von Zufallszahlen für Vines beschäftigt hatten. Dißmann (2010) zeigte einen Algorithmus, welcher mit Hilfe von R-Vine Matrizen, Zufallszahlen ziehen konnte.

Ein allgemeiner Algorithmus, um Zufallszahlen von R-Vine Copulas zu ziehen, verwendet inverse Transformation von Wahrscheinlichkeitsintegralen (*inverse probability integral transform*). Zuerst werden unabhängige Stichproben $u_1, ..., u_n$ erzeugt, welche auf dem Intervall $[0, 1]$ uniform verteilt sind. Dann setze

$$\begin{aligned} x_1 &= u_1 \\ x_2 &= F_{2|1}^{-1}(u_2|x_1) \\ x_3 &= F_{3|1,2}^{-1}(u_3|x_1, x_2) \\ &\vdots \\ x_n &= F_{n|1,2,...,n-1}^{-1}(u_n|x_1, x_2, ..., x_{n-1}). \end{aligned}$$

Nun wird die zugehörige Inverse der *h-Funktion* verwendet, um $F_{i|1,2,...,i-1}^{-1}(u_i|x_1, x_2, ..., x_{i-1})$ für $i = 2, ..., n$ zu berechnen. Folgendes Beispiel soll die Idee der Methode etwas näher bringen.

Beispiel 3. *Es sei eine 3-dimensionale R-Vine Copula gegeben, wobei die Variablen der Reihe nach nummeriert werden. Es gibt dann nur eine mögliche R-Vine Struktur. Zuerst werden die uniform verteilten Zufallszahlen* u_1, u_2, u_3

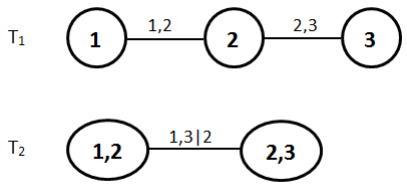

Abbildung 13: 3-dimensionale R-Vine Struktur

generiert und man kann $x_1 = u_1$ *setzen. Wie bereits bekannt ist, gilt*

$$F(x_2|x_1) = \frac{\partial C_{2,1}(x_2, x_1)}{\partial x_1} = h(x_2, x_1|t_{1,2}, p_{1,2}).$$

Daraus folgt

$$x_2 = h^{(-1)}(u_2, x_1|t_{1,2}, p_{1,2}).$$

Die Berechnung für x_3 *ist ähnlich.*

$$\begin{aligned} F(x_3|x_1, x_2) &= \frac{\partial C_{3,1|2}(F(x_3|x_2), F(x_1|x_2))}{\partial F(x_1|x_2)} \\ &= h\left(F(x_3|x_2), F(x_1|x_2)|t_{1,3}, p_{1,3}\right) \\ &= h\left(h(x_3, x_2|t_{2,3}, p_{2,3}), h(x_1, x_2|t_{1,2}, p_{1,2})|t_{1,3}, p_{1,3}\right) \end{aligned}$$

Daraus folgt dann

$$x_3 = h^{-1}\left(h^{-1}(u_3, h(x_1, x_2|p_{1,2})|p_{1,3}), x_2|p_{2,3}\right)$$

Wie man in diesem Beispiel erkennen kann, müssen immer die richtigen Argumente für die vorgegebenen Copulas verwendet werden. Dieses Problem kann wie im Algorithmus 1 mit Hilfe der Matrix $\mathbf{M}^{max}$ gelöst werden. Folgender Algorithmus 2 von Dißmann (2010) generiert Zufallszahlen für eine gegebene R-Vine Spezifikation, gegeben durch $\mathbf{M}$, $\mathbf{T}$ und $\mathbf{P}$. Die Diagonalelemente der Matrix $\mathbf{M}$ müssen dabei wieder absteigend angeordnet sein. Der Algorithmus verwendet die *h-Funktion* und deren Inverse für einen Copulatyp $t_{i,j}$ mit Parameter $p_{i,j}$. [10][5][6]

Algorithmus 2 : Zufallszahlen bei gegebener R-Vine Spezifikation generieren

Input : Matrizen $\mathbf{M}$, $\mathbf{T}$ und $\mathbf{P}$ für die R-Vine Spezifikation, wobei $m_{i,i} = n - i + 1$ für $i = 1, .., n$.

Output : Zufällige Beobachtungen $(x_1, ..., x_n)$ bei gegebener R-Vine Spezifikation.

Es sei $u_1, ..., u_n \overset{iid}{\sim} U[0, 1]$;
Es sei $\mathbf{V}^{direkt} = (v_{i,j}^{direkt} | i, j = 1, ..., n)$;
Es sei $\mathbf{V}^{indirekt} = (v_{i,j}^{indirekt} | i, j = 1, ..., n)$;
Setze $(v_{n,1}^{direkt}, v_{n,2}^{direkt}, ..., v_{n,n}^{direkt}) = (u_n, ..., u_1)$;
Es sei $\mathbf{M}^{max} = (m_{i,j} | i, j = 1, ..., n)$ mit $m_{i,j}^{max} = \max\{m_{i,j}, ..., m_{n,j}\}$ für alle $j = 1, ..., n$ und $i = j, ..., n$;
$x_1 = v_{n,n}^{direkt}$;
for $j = n - 1, ..., 1$ **do**
    for $i = j + 1, ..., n$ **do**
        if $m_{i,j}^{max} = m_{i,j}$ **then**
            Setze $z_{i,j}^{(2)} = v_{i,(n-m_{i,j}^{max}+1)}^{direkt}$
        else
            Setze $z_{i,j}^{(2)} = v_{i,(n-m_{i,j}^{max}+1)}^{indirekt}$
        end
        Setze $v_{n,j}^{direkt} = h^{(-1)}(v_{n,j}^{direkt}, z_{i,j}^{(2)} | t_{i,j}, p_{i,j})$;
    end
    $x_{n-j+1} = v_{n,j}^{direkt}$;
    for $i = n, ..., j + 1$ **do**
        Setze $z_{i,j}^{(2)} = v_{n,j}^{direkt}$;
        Setze $v_{i-1,j}^{direkt} = h(z_{i,j}^{(1)}, z_{i,j}^{(2)} | t_{i,j}, p_{i,j})$;
        Setze $v_{i-1,j}^{indirekt} = h(z_{i,j}^{(2)}, z_{i,j}^{(1)} | t_{i,j}, p_{i,j})$;
    end
end
return $(x_1, ..., x_n)$

3.8 R-Vine Copula anpassen

In diesem Abschnitt wird gezeigt, wie man alle Parameter einer R-Vine Copula schätzen kann. Es wird also wieder nur der Fall behandelt, dass alle eindimensionalen Randverteilungen uniform sind. Die Parameterschätzung für R-Vine Copulas unterteilt sich im Wesentlichen in 3 Aufgaben.

1. Auswahl einer R-Vine Struktur (Matrix $\mathbf{M}$).
2. Auswahl von bivariaten Copulas (Matrix $\mathbf{T}$).
3. Parameterschätzung der bivariaten Copulas (Matrix $\mathbf{P}$).

Es sollen nun alle drei Matrizen so berechnet werden, dass die zugehörige R-Vine Copula am besten an die vorgegebenen Daten angepasst wird. Eine Möglichkeit um solch "ein bestes" Modell zu finden, ist es, $\mathbf{P}$ für alle möglichen R-Vine Strukturen und eine Menge von verschiedenen Copulatypen zu schätzen. Das "beste" Modell sei beispielsweise jenes mit dem kleinsten AIC oder BIC. Aus verschiedenen Copulatypen einen Typ zu wählen erhöht zwar den Rechenaufwand, aber nicht so enorm wie das Testen von allen R-Vine Strukturen. Wie bereits bekannt ist, gibt es für n Variablen $\frac{n!}{2} \cdot 2^{\binom{n-2}{2}}$ R-Vines. In [6] wird ein Algorithmus vorgestellt, um auch in höheren Dimensionen eine gute R-Vine Struktur zu finden. Es wurde eine sequentielle heuristische Methode entwickelt, welche zuerst den Baum $T_1 = (N_1, E_1)$ konstruiert. Die Kanten von T_1 werden dann als Knoten für T_2 verwendet, welcher als nächster konstruiert wird. Dies wird solange fortgesetzt, bis die gesamte R-Vine Struktur aufgebaut ist. Die einzelnen Bäume werden so konstruiert, dass die verbundenen Variablen die höchste paarweise Abhängigkeit aufweisen. Deshalb muss man vorher festlegen, welches Abhängigkeitsmaß für die Berechnungen verwendet werden soll. Jeder Baum wird separat optimiert und deshalb kann man nicht garantieren, dass die so erhaltene R-Vine Struktur die "Beste" ist. Trotzdem berücksichtigt diese sequentielle Methode einige Eigenschaften von R-Vine-Copulas.

- Die Copulas im ersten Baum haben oft den größten Einfluss auf die Modellanpassung.
- Es ist sinnvoller Zufallsvariablen mit höheren Abhängigkeiten mit Hilfe von Copulas korrekt zu beschreiben als welche mit geringeren Abhängigkeiten. Es sind nämlich die Verteilungsfunktionen von verschiedenen Copulatypen, welche Unabhängigkeiten modellieren, ziemlich ähnlich.
- Mit dieser Methode werden in den späteren Bäumen geringere Abhängigkeiten modelliert. Dadurch wird auch der Einfluss von Rundungsfehlern reduziert.

Vor allem bei Anwendungen kann man davon ausgehen, dass die Abhängigkeiten einiger Variablen so gut bestimmt werden. Deshalb ist es sinnvoll in den ersten Bäumen die Copulas mit den höchsten Abhängigkeiten zu wählen, denn die transformierten Variablen für die späteren Bäume sind dann oft beinahe unabhängig. Anhand eines Beispiels soll diese Eigenschaft etwas näher gebracht werden. Dazu wird die mehrdimensionale Normalverteilung betrachtet. Für diese Verteilung können die bedingten Abhängigkeiten leicht berechnet werden.

Definition 10 (Partielle Korrelation)**.** *Gegeben seien die Zufallsvariablen* $X_1, X_2, X_3, ..., X_n$ *und* $X^*_{1|3,...,n}$, $X^*_{2|3,...,n}$ *seien lineare Funktionen von* $X_3, ..., X_n$, *welche* $\mathbb{E}[(X_1 - X^*_{1|3,...,n})^2]$ *bzw.* $\mathbb{E}[(X_2 - X^*_{2|3,...,n})^2]$ *minimieren. Dann kann der partielle Korrelationskoeffizient* $\rho_{1,2|3,...,n}$ *von* X_2 *und* X_3 *als die gewöhnliche Korrelation zwischen* $Y_1 := X_1 - X^*_{1|3,...,n}$ *und* $Y_2 := X_2 - X^*_{2|3,...,n}$ *definiert werden. Es gilt also*

$$\rho_{1,2|3,...,n} = \frac{\mathbb{E}[(Y_1 - \mathbb{E}[Y_1])(Y_2 - \mathbb{E}[Y_2])]}{\sqrt{Var(Y_1)Var(Y_2)}}.$$

Die partiellen Korrelationen können dann rekursiv berechnet werden.

$$\rho_{1,2|3,...,n} = \frac{\rho_{1,2|3,...,n-1} - \rho_{1,n|3,...,n-1} \cdot \rho_{2,n|3,...,n-1}}{\sqrt{1 - \rho^2_{1,n|3,...,n-1}}\sqrt{1 - \rho^2_{2,n|3,...,n-1}}}$$

Beispiel 4. *Als Beispiel wird nun eine mehrdimensionale Normalverteilung mit Erwartungswertvektor gleich dem Nullvektor und Korrelationen* $\rho_{1,2}$, $\rho_{1,3}$ *und* $\rho_{2,3}$ *betrachtet. Wählt man dann* $\rho_{1,2} = \rho_{2,3} > \rho_{1,3} > 0$ *dann ergibt sich*

$$\rho_{1,2|3} = \frac{\rho_{1,2} - \rho^2_{1,3}}{1 - \rho^2_{1,3}} < \rho_{1,2},$$

wobei $\rho_{1,2} \leq 1$ *und* $\rho_{1,2|3} > 0$, *weil die Korrelationsmatrix positiv definit ist. Wenn man also zuerste Die Variablen mit höheren Abhängigkeiten modelliert, so ist die Abhängigkeit von* X_1 *und* X_2 *bei gegebenen* X_3 *kleiner.*

Um eine R-Vine Copula Spezifizierung mit den oben genannten Eigenschaften zu generieren, wird in [6] Algorithmus 3 verwendet, welcher auf Kendall's Tau basiert. Es können aber auch andere Abhängigkeitsmaße verwendet werden. Kendall's Tau hat aber den Vorteil, dass man keine Verteilungen annehmen muss. Außerdem können so leicht verschiedene Copulatypen kombiniert werden.

Algorithmus 3 : Sequentielle Methode für die Auswahl eines R-Vine Modells basierend auf Kendall's Tau

Input : Realisierungen von unabhängig identisch verteilten Zufallsvektoren $(x_{l,1}, ..., x_{l,n}), l = 1, ..., N$

Output : R-Vine Copula Spezifikation.

1 Berechne die empirischen Kendall's Tau $\widehat{\tau}_{i,j}$ für alle möglichen Paarungen der Variablen $\{i, j\}, 1 \leq i < j \leq n$;

2 Wähle den Spannbaum, welcher die Summe der absoluten empirischen Kendall's Tau

$$\sum_{\substack{e=\{i,j\} \\ \text{aus dem spannenden Baum}}} |\widehat{\tau}_{i,j}|$$

maximiert;

3 Für jedes Paar {i,j} aus dem gewählten Spannbaum wähle eine Copula und berechne die zugehörigen Parameter. Dann transformiere $\widehat{F}_{i|j}(x_{l,i}, x_{l,j})$ und $\widehat{F}_{j|i}(x_{l,j}, x_{l,i})$ für $l = 1, ..., N$ mit Hilfe der angepassten Copula $\widehat{C}_{i,j}$;

4 **for** $j = 2, ..., n-2$ **do**

5 Berechne die empirischen Kendall's Tau $\widehat{\tau}_{i,j|D}$ für alle möglichen Paarungen der bedingten Variablen $\{i, j|D\}$ aus dem Baum T_i, welche die *proximity condition* erfüllen;

6 Für diese Paarungen $\{i, j|D\}$ wähle den Spannbaum, welcher die Summe der absoluten empirischen Kendall's Tau

$$\sum_{\substack{e=\{i,j|D\} \\ \text{aus dem spannenden Baum}}} |\widehat{\tau}_{i,j|D}|$$

maximiert;

7 Für jedes Paar {i,j—D} aus dem gewählten Spannbaum wähle eine Copula und berechne die zugehörigen Parameter. Dann transformiere $\widehat{F}_{i|j\cup D}(x_{l,i}, x_{l,j}, \underline{x}_{l,D})$ und $\widehat{F}_{j|i\cup D}(x_{l,j}, x_{l,i}, \underline{x}_{l,D})$ für $l = 1, ..., N$ mit Hilfe der angepassten Copula $\widehat{C}_{i,j|D}$;

8 **end**

In diesem Algorithmus müssen im Wesentlichen drei Probleme gelöst werden.

1. Suche für alle möglichen Kanten (*proximity condition*) aus $T_i, i = 1, ..., n-1$ einen Spannbaum, welcher die Summe der absoluten Kendall's Tau maximiert.

2. Wähle für jede der ausgewählten Kanten in $T_i, i = 1, ..., n-1$ eine passende Copula.

3. Schätze für jede Copula in $T_i, i = 1, ..., n-1$ die Parameter. Verwende die Copulas und deren Parameter um die transformierten Werte zu erhalten, welche im nächsten Baum verwendet werden, um Kendall's Tau zu berechnen.

Um das erste Problem zu lösen, wird der Algorithmus von Prim verwendet. Dieser Algorithmus sucht in einem zusammenhängenden, ungerichteten, kantengewichteten Graphen den minimalsten Spannbaum. Dieser Algorithmus kann aber auch umgeschrieben werden, um den maximaler Spannbaum zu finden.

Algorithmus 4 : Algorithmus von Prim für maximalen Spannbaum

Input : Zusammenhängender, ungerichteter, kantengewichteter Graph $G = (N, E)$

Output : Maximaler Spannbaum $T = (N, E')$

Setze $E' = \{\}$;
Setze $N' = \{x\}$, wobei x ein beliebiger Knoten aus N ist;
while E' *verbindet nicht alle Knoten von* N **do**
    Wähle eine Kante $e \in E$ mit maximalem Kantengewicht, welche einen noch nicht in T enthaltenen Knoten $y \in E$ mit T verbindet;
    Füge e zu E' hinzu;
    Füge y zu N' hinzu;
end
return $T = (N, E')$

Algorithmus 4 muss aber noch ein wenig modifiziert werden, damit das erste Problem gelöst werden kann. Es dürfen nämlich nur Kanten verwendet werden, welche die *proximity condition* erfüllen. Das bedeutet, dass nur Kanten aus der Menge

$$E_i^* = \{\{a, b\} \in N_i \times N_i : |a \cup b| = 1\}$$

gewählt werden können. In einem Beispiel soll dieser modifizierte Algorithmus etwas näher gebracht werden.

Beispiel 5. *Betrachtet man nun* $n = 5$ *Variablen, so können in* T_1 *aus* $5(5-1)/2 = 10$ *Kanten* $5 - 1 = 4$ *Kanten gewählt werden, welche einen maximalen Spannbaum bilden. Die Kanten, welche die Summe der absoluten empirischen Kendall's Tau maximiert, seien beispielsweise 15,12,24 und 23. Die gewählten*

Kanten in T_1 sind dann die Knoten in T_2. Vier Kanten in T_2 erfüllen die proximity condition und 3 davon bilden den maximalen Spannbaum. In T_3 erfüllen nur 2 von 3 Kanten die proximity condition und es gibt daher nur einen Spannbaum. Allgemein kann man erkennen, dass es keine Wahlmöglichkeiten mehr gibt, sobald eine D-Vine Struktur vorkommt.

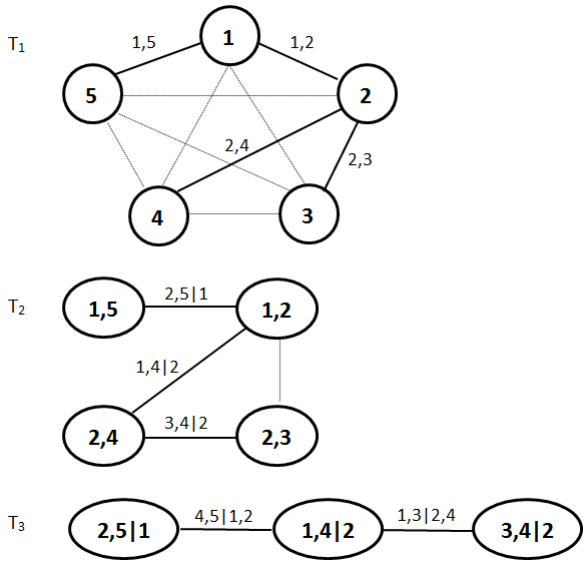

Abbildung 14: 5-dimensionale R-Vine Struktur

Der Algorithmus von Prim generiert immer einen Spannbaum, wenn der gegebene Graph zusammenhängend ist. In [6] wird gezeigt, dass diese Annahme im Algorithmus 3 immer erfüllt ist. Es ist also garantiert, dass in jedem Schritt, trotz *proximity condition*, immer ein Spannbaum gefunden wird.

Das zweite Problem soll so gelöst werden, dass die Wahl der Copula für jede Kante automatisch und objektiv entschieden wird. Ein Chi-Plot kann daher nicht verwendet werden. Für eine R-Vine Copula können alle bivariaten Copuluatypen von der unteren Fréchet Hoeffding Schranke bist zur oberen Fréchet Hoeffding Schranke verwendet werden. In der Praxis sind aber die wichtigsten Copulatypen, welche im ersten Kapitel vorgestellt wurden, oft ausreichend. Man kann sich daher auf eine Menge von Copulatypen, und deren rotierende Versionen beschränken. Umso größer diese Menge gewählt wird, umso mehr erhöht sich der Rechenaufwand. In dieser Menge können dann aber noch weitere Einschränkungen gemacht werden. Wenn man beispielsweise positive Abhängigkeit

modellieren will, kann man die Gumbel Copula oder auch die um 180° gedrehte Gumbel Copula wählen. Bei negativer Abhängigkeit können diese nicht verwendet werden, aber dafür kann man die um 90° oder 270° gedrehte Gumbel Copula wählen. Um nun eine gewisse Abhängigkeit zu modellieren, soll aus den möglichen Copulas die "beste" gewählt werden. Dazu werden alle möglichen Copulas mittels der Maximum-Likelihood-Methode an die Daten angepasst. Die Copula mit dem kleinsten AIC wird dann gewählt. Das Akaike Informationskriterium ist allgemein folgend definiert.

Definition 11. *Sei $x_1, ..., x_N$ eine Stichprobe, dann ist das **Akaike Informationskriterium** (AIC) gegeben durch*

$$AIC := -2 \sum_{i=1}^{N} \log f(x_i|\hat{\underline{\theta}}) + 2k,$$

wobei $\hat{\underline{\theta}}$ die Maximum-Likelihood-Schätzung von $\underline{\theta} = (\theta_1, ..., \theta_k)$ ist.

Somit ergibt sich für eine bivariate Copula C mit Parameter $\underline{\theta}$ bei gegebener Stichprobe $u_{i,j}; i = 1, ..., N; j = 1, 2$

$$AIC := -2 \sum_{i=1}^{N} \log c_{\hat{\underline{\theta}}}(u_{i,1}, u_{i,2}) + 2k.$$

Es kann natürlich auch das Bayesscheses Informationskriterium (BIC) verwendet werden, welches im Gegenteil zum AIC die Anzahl der Parameter mit dem Faktor $\log n$ bestraft. Brechmann [2] führte eine Simulationsstudie durch, in der einige Kriterien für die Wahl der Copulas verglichen wurden. Das Ergebnis der Studie zeigte, dass die Wahl mit Hilfe von AIC meist die beste Methode ist. Diese Methode ist vor allem zuverlässig und rechnerisch effizienter als eine Güteanpassung. Wenn die Abhängigkeiten und der Stichprobenumfang steigen, dann wird die Auswahl mittels AIC immer besser.

Durch die Lösung des zweiten Problems ist die Lösung für das dritte Problem trivial. Die Parameter der Copulas sind nämlich schon während der Anpassung geschätzt worden und können somit für die weiteren Berechnungen verwendet werden.

3.8.1 Bemerkungen

Algorithmus 3 liefert eine allgemeine R-Vine Struktur. Man könnte aber auch daran interessiert sein eine C-Vine oder D-Vine Struktur zu finden.

Bei einer **C-Vine** Struktur gibt es in jedem Baum T_i einen Knoten mit Grad $n - i$. Algorithmus 3 kann also geändert werden, so dass in jedem Schritt an Stelle eines Spannbaums ein Stern gesucht wird, welcher alle Knoten enthält und die Summe der Kantengewichte maximiert. Dazu kann Algorithmus 5 verwendet werden. Bei einer **D-Vine** Struktur ist es ausreichend T_1 zu konstruieren,

Algorithmus 5 : Algorithmus: Vollständiger Stern mit maximaler Summe der Kantengewichte

Input : Zusammenhängender, ungerichteter, kantengewichteter Graph $G = (N, E)$
Output : Stern $T = (N, E')$

$m = |N|$;
$w_{i,j}$ =Gewicht der Kante $(i, j) \in E$;
for $i = 1, ..., m$ **do**
  $\widehat{w}_i = \sum_{j=1}^{m} w_{i,j}$;
  $E_i = \bigcup_{j=1}^{m} \{(i, j)\}$;
end
$ind = argmax\{\widehat{w}_i; i = 1, ..., m\}$;
$E' = E_{ind}$;
return $T = (N, E')$

denn für die weiteren Bäume gibt es keine Wahlmöglichleiten der Kanten mehr. Dieses Problem scheint auf den ersten Blick ein Einfaches zu sein, aber man muss einen vollständigen Weg finden, welcher die Summe der Kantengewichte maximiert. Es ist also ein Hamilton-Weg gesucht, welcher die Summe der Kantengewichte maximiert. In der Literatur gibt es heuristische Lösungsverfahren für dieses Problem (siehe [2]), aber diese werden in dieser Arbeit nicht behandelt. [10][2][5][6]

Test auf Unabhängigkeit

Im Algorithmus 3 werden die Variablen so gewählt, dass in den ersten Bäumen hohe Abhängigkeiten modelliert werden und in späteren Bäumen kleinere Abhängigkeiten. Möglicherweise könnten daher in späteren Bäumen unabhängige Paare modelliert werden. Es wäre also nützlich vor der Copula Anpassung zu testen ob die Variablen unabhängig sind. Falls dies der Fall ist, so kann die Unabhängigkeitscopula Π verwendet werden und es müssen keine Parameter geschätzt werden. Als nächstes wird ein asymptotischer Unabhängigkeitstest vorgestellt, welcher auf Kendall's Tau basiert, weil im Algorithmus 3 auch die-

ses Abhängigkeitsmaß verwendet wird. Es soll getestet werden

$$H_0 : \tau = 0 \text{ vs. } H_1 : \tau \neq 0.$$

Als Teststatistik wird die empirische Berechnung von Kendall's Tau $\hat{\tau}$ verwendet. Unter der Nullhypothese ist die Statistik $\hat{\tau}$ nahezu normalverteilt mit Erwartungswert $\mu = 0$ und Varianz gleich

$$\sigma^2 = \frac{2(2N+5)}{9N(N-1)},$$

wobei N gleich dem Stichprobenumfang ist. Bei diesem Test zum approximativen Niveau α wird die Nullhypothese H_0 verworfen, wenn

$$T := \sqrt{\frac{9N(N-1)}{2(2N+5)}}|\hat{\tau}| > \Phi^{-1}\left(1 - \frac{\alpha}{2}\right).$$

Der p-value des Testes kann asymptotisch folgend berechnet werden.

$$P(|Z| > T) = 2P(Z > T) = 2(1 - \Phi(T)),$$

wobei mit Φ die Standardnormalverteilung bezeichnet wird. [2][5]

3.9 Modellvergleich

An die Daten können die verschiedensten R-Vine Spezifikationen angepasst werden. Es können verschiedene R-Vine Matrizen definiert werden und auch verschiedene Copulatypen gewählt werden. Auch die Parameter können mit verschiedenen Methoden geschätzt werden, wie zum Beispiel mit der Maximum-Likelihood-Methode oder auch mit Hilfe von Kendall's Tau. Wenn man nun mehrere Modelle angepasst hat, sollte das "beste" Modell unter diesen verwendet werden. Als Grundidee, um zwei Modelle zu vergleichen, wird die Kullback-Leibler-Divergenz verwendet. Mit dieser ist es möglich den Abstand zwischen einer unbekannten Verteilung und einem angepassten Modell mit geschätzten Parametern zu messen. Es sei h_0 die Dichte der wahren, aber unbekannten Verteilung einer Zufallsgröße X. Des Weiteren sei E_0 der Erwartungswert bezüglich der wahren Verteilung und $f(., \hat{\theta})$ das angepasste Modell. Die Kullback-Leibler-Divergenz ist dann folgend definiert:

$$KL(h_0, f, \hat{\theta}) = \int h_0(x) \log \frac{h_0(x)}{f(x, \hat{\theta})} dx = E_0(\log h_0(X)) - E_0(\log f(X, \hat{\theta}))$$

Es sollte nun das Modell gewählt werden, welches die Kullback-Leibler-Divergenz minimiert. Da aber die wahre Verteilung unbekannt ist, wird für gewöhnlich das Modell gewählt, welches $E_0(\log f(X, \hat{\theta}))$ maximiert.

Es soll nun der Vuong Test näher gebracht werden. [2][5] Mit diesem ist es möglich zwei Modelle $f_1(., \hat{\theta_1})$ und $f_2(., \hat{\theta_2})$ zu vergleichen. Der Vuong Test untersucht die Nullhypothese:

$$\begin{aligned} & KL(h_0, f_1, \hat{\theta_1}) = KL(h_0, f_2, \hat{\theta_2}) \\ \Leftrightarrow \quad & E_0\left(\log f_1(X, \hat{\theta_1})\right) = E_0\left(\log f_2(X, \hat{\theta_2})\right). \end{aligned}$$

Offensichtlich ist $f_1(., \hat{\theta_1})$ besser als $f_1(., \hat{\theta_1})$, wenn $E_0(\log f_1(X, \hat{\theta_1}))$ größer als $E_0(\log f_2(X, \hat{\theta_2}))$ ist und umgekehrt. Es soll aber getestet werden, ob der Unterschied signifikant ist. Dazu werden die asymptotischen Resultate betrachtet, um eine geeignete Teststatistik zu definieren. Für die Beobachtungen x_i definiere

$$m_i = \log\left(\frac{f_1(x_i, \hat{\theta_1})}{f_2(x_i, \hat{\theta_2})}\right),$$

wobei $i = 1, ..., N$. Vuong nutzte das Gesetz der Großen Zahlen und konnte so unter angemessenen Annahmen zeigen, dass

$$\frac{1}{N} LR_N(\hat{\theta_1}, \hat{\theta_2}) := \frac{1}{N} \sum_{i=1}^{N} m_i \xrightarrow[N\to\infty]{f.s.} \mu_0^m$$

wobei μ_0^m der Erwartungswert von $\underline{m} = (m_1, ..., m_N)$ ist. Mit Hilfe der Stichprobenvarianz von LR_N

$$\hat{\omega}^2 := \frac{1}{N} \sum_{i=1}^{N} (m_i - \overline{m})^2,$$

erhielt Vuong die asymptotische Verteilung von LR_N:

$$\nu = \frac{LR_N(\hat{\theta_1}, \hat{\theta_2})}{\sqrt{N\hat{\omega}^2}} \xrightarrow[N\to\infty]{d} N(0, 1).$$

Nun können die Hypothesen für den Test aufgestellt werden:

$$H_0 : \mu_0^m = 0 \text{ vs. } H_1 : \mu_0^m \neq 0.$$

Die Nullhypothese H_0 wird bei einem Signifikanzniveau α verworfen, wenn

$$|\nu| > \Phi^{-1}\left(1 - \frac{\alpha}{2}\right).$$

Außerdem gilt dann, wenn

- $\nu > \Phi^{-1}(1 - \frac{\alpha}{2})$ wird $f_1(., \hat{\theta_1})$ bevorzugt.
- $\nu < -\Phi^{-1}(1 - \frac{\alpha}{2})$ wird $f_2(., \hat{\theta_1})$ bevorzugt.

Wenn die Nullhypothese nicht abgelehnt werden kann, dann wird keines der beiden Modelle bevorzugt.
Bei diesem Test wird aber nicht beachtet, dass die zwei Modelle nicht gleich viel Parameter besitzen müssen. Deshalb fügt Vuong einen Korrekturfaktor basierend auf AIC bzw. BIC hinzu

$$\begin{aligned} LR_N^{Akaike}(\hat{\theta}_1, \hat{\theta}_2) &= LR_N(\hat{\theta}_1, \hat{\theta}_2) - k_1 - k_2, \\ LR_N^{Schwarz}(\hat{\theta}_1, \hat{\theta}_2) &= LR_N(\hat{\theta}_1, \hat{\theta}_2) - \frac{k_1}{2}\log(N) - \frac{k_2}{2}\log(N), \end{aligned}$$

wobei k_1 die Anzahl der Parameter des ersten Modells angibt und k_2 die des zweiten Modells.

Eine weitere Möglichkeit, um zwei R-Vine Spezifikationen zu vergleichen, ist durch die Verwendung von AIC (oder BIC) gegeben. Man berechnet den AIC (oder BIC) der beide Modelle und das Modell mit dem kleineren Wert wird bevorzugt. Für ein R-Vine Copula Modell kann bei gegebenen Stichproben $u_{i,j}; i = 1, ..., N; j = 1, ..., n$ das Akaike Informationskriterium durch

$$\begin{aligned} AIC &= -2\sum_{i=1}^{N}\log\prod_{i=1}^{n-1}\prod_{e\in E_i} c_{C_{e,a},C_{e,b}|D_e}(F_{C_{e,a}|D_e}, F_{C_{e,b}|D_e}) + 2k \\ &= -2\sum_{i=1}^{N}\sum_{i=1}^{n-1}\sum_{e\in E_i}\log\left(c_{C_{e,a},C_{e,b}|D_e}(F_{C_{e,a}|D_e}, F_{C_{e,b}|D_e})\right) + 2k \end{aligned}$$

berechnet werden. Die Anzahl der Parameter des Modells ist dabei mit k gegeben. [2][5]

3.10 Vereinfachung eines R-Vine Modells

Ein R-Vine Modell ist sehr felxibel und mit diesem ist es möglich einen n-dimensionalen Zufallsvektor zu beschreiben. Dabei werden $n(n-1)/2$ Copulas verwendet. Diese Copulas haben meist je ein bis zwei Parameter. Bei einem

wachsende n, steigt auch die Anzahl der Parameter. Es hat sich in Anwendungen gezeigt, dass die Modellierung der ersten Bäume wichtiger und einflussreicher ist, als die Modellierung der späteren Bäume. Die Idee ist es daher die R-Vine Struktur ab einem bestimmten Baum abzuschneiden bzw. zu trennen. Der erste Teil wird wieder mit verschiedenen Copulas angepasst. Der zweite Teil wird dann nur mit Unabhängigkeitscopulas angepasst. In der Literatur wird diese Vereinfachung *Truncation* genannt. Der Index K des letzten Baums T_K, welcher im ersten Teil vorkommt, wird als *Truncation-Level* bezeichnet. An Hand eines Beispiels soll diese Idee näher gebracht werden.

Beispiel 6. *Es seien $n = 5$ Variablen gegeben und es wird eine R-Vine Struktur festgelegt (siehe Abbildung 15). Als Truncation-Level wird $K = 2$ gewählt. Es werden daher an den ersten zwei Bäumen verschiedene Copulas angepasst und für die Bäume T_3 und T_4 werden nur UnabhängigkeistsCopulas verwendet.*

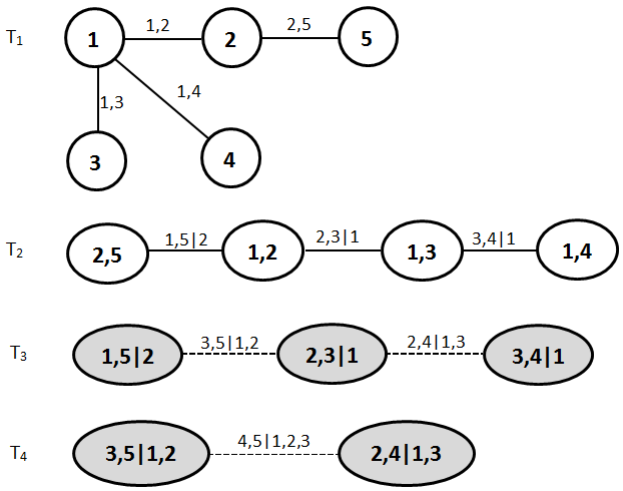

Abbildung 15: *Truncation* einer R-Vine Struktur

In Beispiel 6 müssen insgesamt nicht 10, sondern nur 7 Copulas angepasst werden. In höheren Dimensionen kann dieser Unterschied wesentlich größer sein und dadurch verringert sich der Rechenaufwand. Die Dichte einer R-Vine Verteilung vereinfacht sich bei einem vorgegebenen K auf

$$f_{1,\dots,n} = f_1 \cdot \dots \cdot f_n \cdot \prod_{i=1}^{K} \prod_{e \in E_i} c_{C_{e,a},C_{e,b}|D_e} \left(F_{C_{e,a}|D_e}, F_{C_{e,b}|D_e} \right).$$

Hätte man in Beispiel 6 K=1 gewählt, so hätten nur 4 Copulas angepasst werden müssen und die R-Vine Struktur wäre auf einen Markov-Modell reduziert

worden. Speziell für eine D-Vine Struktur mit zeitlich geordneten Knoten, kann die *Truncation* der Vine Struktur als ein Markov-Prozess K-ter Ordnung aufgefasst werden.
In Buch [9] wird eine Idee vorgestellt, um eine optimale *Truncation* einer R-Vine Struktur zu bekommen. Dazu wird die partielle Korrelation verwendet. Es werden zuerst die beiden Variablen $x, y \in I = \{1, ..., n\}$ gesucht, welche die absolute partielle Korrelation $\rho_{x,y|I\setminus\{x,y\}}$ minimieren. Die R-Vine Struktur wird dann nicht wie gewöhnlich vom ersten bis zum letzten Baum aufgebaut, sondern umgekehrt. Ziel ist es nämlich, dass in den höheren Bäumen die geringsten absoluten partiellen Korrelationen vorkommen. Der Baum T_{n-1} beinhaltet dann also die Kante $x, y|I\setminus\{x, y\}$. Im Buch [9] wird ein Algorithmus vorgestellt, welcher die R-Vine Struktur dann rückwärts aufbaut. Die Kanten mit den kleinsten absoluten partiellen Korrelationen, welche sich hauptsächlich in den späteren Bäumen befinden, werden dann mit UnabhängigkeitsCopulas modelliert.

Es gibt noch eine weitere Möglichkeit eine R-Vine Copula zu vereinfachen. Bei einer vorgegebenen R-Vine Struktur werden wieder bis zum Baum T_K verschiedene Copulas angepasst, wobei $K < (n-1)$. In den späteren Bäumen werden dann alle Zusammenhänge mit genau einem bestimmten Copulatyp beschrieben. Für diesen Copulatyp wird meist die Normal Copula gewählt. Mit dieser kann man nämlich negative als auch positive Abhängigkeiten modellieren und sie besitzt nur einen Parameter. Diese Methode wird *Simplification* genannt. Werden bei einer *Simplification* von R-Vines in den letzteren Bäumen nur Normal Copulas mit Parameter gleich 0 angepasst, so erhält man den Spezialfall *Truncation* von R-Vines. [9][2]

3.11 Eigenschaften von R-Vine Copulas

Mit R-Vine Copulas ist es möglich die verschiedensten Abhängigkeiten mehrdimensionaler Verteilungen zu modellieren. Es können auch bekannte mehrdimensionale Copulas aufgebaut werden.

Theorem 8. *Die mehrdimensionale Normal-Copula kann als R-Vine Copula dargestellt werden, in welcher alle bivariaten Copulas normal sind.*

Die Parameter der R-Vine Copula können dabei mit Hilfe der Korrelation und der partiellen Korrelation berechnet werden. Auch die Umkehrung gilt. Jede R-Vine Copula mit ausschließlich bivariaten Normal-Copulas ist eine mehrdimensionale Normal Copula. Auch die multivariate t-Copula kann als Spezialfall einer R-Vine Copula aufgefasst werden. Es dürfen dabei nur

bivariate t-Copulas verwendet werden.

Über die Abhängigkeiten einer R-Vine Copula können auch Aussagen getroffen werden. Wenn beispielsweise alle Copulas im ersten Baum untere (obere) *tail dependence* modellieren, dann haben alle bivariaten Randverteilungen von $F_{1,...,n}(x_1, ..., x_2)$ untere (obere) *tail dependence*. Auch über die Verwendung der Minimum und Maximum Copula können Eigenschaften gezeigt werden. Es sei beispielsweise eine R-Vine Copula mit 3 Variablen gegeben. Es werden dann die drei Abhängigkeiten $\tau_{1,2}$, $\tau_{1,3}$ und $\tau_{2,3|1}$ betrachtet, welche durch Kendall's Tau berechnet werden. Wird im ersten Baum die Maximum Copula für die Kanten 12 und 13 verwendet, so ergibt sich für $\tau_{2,3|1}$ der kleinste mögliche Wert. Der größte mögliche Wert den $\tau_{2,3|1}$ annehmen kann, wird erreicht, wenn im ersten Baum nur die Minimum Copula verwendet wird. [10][9]

4 Anwendung: VaR Vorhersage

Mit Hilfe von Vine Copulas ist es möglich die Abhängigkeiten von Zufallsvariablen zu modellieren. Dieser Abschnitt soll zeigen, wie die kennengelernte Theorie praktisch umgesetzt werden kann. Speziell in Banken können Vine Copulas verwendet werden, um ein gewisses Risiko vorherzusagen. Speziell ist man daran interessiert das Risiko für ein Portfolio, bestehend aus mehreren Positionen, zu berechnen. Die Rendite eines Portfolios zum Zeitpunkt t ergibt sich durch

$$R_{p,t} = \sum_{i=1}^{n} w_i R_{i,t},$$

wobei $R_{i,t}$ die Rendite der i-ten Position zum Zeitpunkt t ist und w_i das zugehörige Gewicht. Es kann dafür auch die Log-Rendite verwendet werden, welche in dieser Arbeit für die Berechnungen verwendet wird. Die Log-Rendite ist definiert als

$$R_{i,t} = \log\left(\frac{S_t}{S_{t-1}}\right) = \log S_t - \log S_{t-1},$$

wobei mit S_t der Preis der i-ten Postion zum Zeitpunkt t gegeben ist. [9]

4.1 Risikomessung-VaR

Im Bereich der Finanzmathematik gibt es einige Möglichkeiten ein Risikomaß zu definieren. Oft wünscht man sich, dass ein Risikomaß gewisse Eigenschaften erfüllt.

Definition 12. *Sei $\mathcal{L}$ ein linearer Raum mit messbaren Funktionen, dann wird das Funktional $\rho : \mathcal{L} \to \mathbb{R} \cup \{+\infty\}$* ***kohärentes Risikomaß*** *genannt, wenn folgende Eigenschaften erfüllt werden:*

- $X_1 \leq X_2 \Rightarrow \rho(X_1) \leq \rho(X_2)\ \forall X_1, X_2 \in \mathcal{L}$ ***(monoton)***
- $\rho(X_1 + X_2) \leq \rho(X_1) + \rho(X_2)\ \forall X_1, X_2 \in \mathcal{L}$ ***(subadditiv)***
- $\rho(kX) = k\rho(X)\ \forall X \in \mathcal{L}$ *und* $k \geq 0$ ***(positiv homogen)***
- $\rho(X + n) = \rho(X) + n\ \forall X \in \mathcal{L}$ *und* $n \in \mathbb{R}$ ***(translationsinvariant)***

Das bekannteste Risikomaß, welches in Anwendungen häufig verwendet wird, ist der VaR (*Value at Risk*). Dieses Risikomaß ist im Allgemeinen nicht subadditiv und deshalb auch nicht kohärent. Trotzdem wird dieses Risikomaß für die Berechnungen in dieser Arbeit verwendet, weil die Subadditivität in den meisten Fällen doch erfüllt ist. Außerdem kann der VaR leicht

interpretiert und implementiert werden. Es gibt natürlich andere Risikomaße, wie beispielsweise den *Expected Shortfall*, welche kohärent sind, aber diese werden in dieser Arbeit nicht näher betrachtet.

Definition 13. *Der **VaR** (Value at Risk) zum Konfidenzniveau* $1-\alpha$ *ist definiert als*

$$VaR_{1-\alpha} = F_X^{-1}(\alpha) = \inf\{x | F_X(x) \geq \alpha\},$$

wobei die Wertveränderung (Gewinne und Verluste), der betrachtenden Risikoposition (Asset) über einen bestimmten Zeitraum, mit der Zufallsvariable X beschrieben wird.

Der VaR gibt den Wert des Verlustes an, welcher mit einer Wahrscheinlichkeit von $1-\alpha$ innerhalb einer Zeitspanne nicht überschritten wird. Um den VaR für ein bestimmtes Portfolio zu schätzen, wird meist der *Varianz-Covarianz-Ansatz* verwendet. Dazu muss aber angenommen werden, dass das Portfolio normalverteilt ist. Es ist aber auch ein weiterer Ansatz mittels *Monte Carlo Simulation* möglich. Dazu wird ein Modell an die beobachteten Zeitreihen $R_{i,1}, ..., R_{i,T}$ angepasst, wobei mit $i = 1, ..., n$ der Index der verschiedenen Positionen gegeben ist. Es werden dann die Parameter des Modells geschätzt, um Vorhersagen $R_{i,T+1}^{(m)}, m = 1, ..., M$ zu generieren. Der VaR des Portfolios lässt sich dann durch

$$\hat{VaR}_{1-\alpha}^{p} = -\hat{q}_\alpha \left(\sum_{i=1}^{n} R_{i,T+1}^{(m)} \right)$$

schätzen, wobei mit $\hat{q}_\alpha$ das α-Quantil der Stichprobe bezeichnet wird.

Nachdem ein Modell an die Daten angepasst wird, ist es vernünftig zu prüfen, ob die Berechnungen des VaR in Ordnung sind. Die bekannteste Methode hierfür ist das *Backtesting*. Bei dieser Methode berechnet man für ein bestimmtes Zeitintervall den täglichen VaR und vergleicht diese Werte mit den Beobachtungen. Wenn viele Beobachtungen größer als der VaR sind, dann wird das Risiko unterschätzt. Wenn aber zu wenig Beobachtungen größer als der VaR sind, dann ist man zu konservativ. Bei einer Anzahl von T Beobachtungen erwartet man daher, dass $T \cdot \alpha$ dieser Beobachtungen einen größeren Wert als der berechnete VaR aufweisen. Kupiec führte den POF-Test (*proportion of failures*) ein. Dabei wird die Nullhypothese

$$H_0 : \alpha = \hat{\alpha} = \frac{x}{T}$$

geprüft, wobei x die Anzahl der Beobachtungen sind, welche größer als der berechnete VaR sind. Jeder Tag in der Testperiode wird als Bernoulli Experiment

aufgefasst. Die zugehörige Zufallsvariable $X_i, i = 1, ..., T$ nimmt mit Wahrscheinlichkeit α den Wert 1 und ansonsten den Wert 0 an. Somit ist die Anzahl der Beobachtungen, welche größer als der VaR sind, binomialverteilt

$$P(X = x) = \binom{T}{x} \alpha^x (1-\alpha)^{(T-x)}.$$

Nun kann der Likelihood-Quotienten-Test konstruiert werden.

$$LQ(x) = \frac{(1-\alpha)^{T-x}\alpha^x}{\left(1-\frac{x}{T}\right)^{T-x}\left(\frac{x}{T}\right)^x}$$

Unter der Annahme, dass die Nullhypothese korrekt ist, konvergiert die Teststatistik, gegeben durch $-2\ln(LQ(x))$, asymptotisch gegen χ_1^2. Bei einem Fehler erster Art von 5% kann die Nullhypothese verworfen werden, wenn die Teststatisitk größer als das 95%-Quatil der χ_1^2-Verteilung ist. [4][12]

4.2 Zeitreihen Analyse

Im praktischen Teil dieser Arbeit werden zuerst die Randverteilungen modelliert, um dann die Abhängigkeiten mehrerer Zufallsvariablen zu beschreiben. Da später ein Portfolio betrachtet wird, empfiehlt es sich die einzelnen Positionen mit Hilfe von Zeitreihen zu beschreiben.

Definition 14. *Ein Zeitreihen Modell für die Beobachtungen $\{x_t\}$ ist eine Spezifikation der gemeinsamen Verteilungen von Zufallsvariablen $\{X_t\}$, wobei $\{x_t\}$ die zugehörigen Realisierungen sind.*

Wenn es möglich ist, reicht es aus, anstelle der gemeinsamen Verteilungen nur die Erwartungswerte und Kovarianzen zu betrachten.

Definition 15. *Es sei $\{X_t\}$ eine Zeitreihe mit $\mathbb{E}(X_t^2) < \infty$. Dann ist **Mittelwertfunktion** von $\{X_t\}$ gleich*

$$\mu_X(t) = \mathbb{E}(X_t)$$

*und die **Kovarianzfunktion** von $\{X_t\}$ ist durch*

$$\gamma_X(r,s) = Cov(X_r, X_s) = \mathbb{E}\left((X_r - \mu_X(r))(X_s - \mu_X(s))\right)$$

gegeben, wobei r und s ganze Zahlen sind.

Eine wichtige Eigenschaft in der Zeitreihen Theorie ist die Stationarität. In den Anwendungen ist die schwache Stationarität oft ausreichend.

Definition 16. *Die Zeitreihe $\{X_t\}$ ist (**schwach**) **stationär**, wenn*

1. *$\mu_X(t)$ unabhängig von t ist und*
2. *$\gamma_X(t+h,t)$ ist unabhängig von t, für jedes $h \in \mathbb{Z}$.*

Die Kovarianzfunktion für stationäre Zeitreihen kann so definiert werden, dass diese nur noch von einem Parameter abhängt.

$$\gamma_X(h) := \gamma_X(h,0) = \gamma_X(t+h,t) = Cov(X_{t+h}, X_t).$$

Dabei wird $\gamma_X(h)$ als *Autokovarianzfunktion* (ACVF) bezeichnet. Dadurch lässt sich dann die *Autokorrelationsfunktion* (ACF)

$$\rho_h = \rho_X(h) = \frac{\gamma_X(h)}{\gamma_X(0)} = Cor(X_{t+h}, X_t)$$

herleiten. Da sich im Finanzbereich die Volatilität mit der Zeit meist ändert, ist es sinnvoll ein GARCH-Modell (*Generalized Autoregressive Conditional Heteroskedasticity*) an die Positionen anzupassen.

Definition 17. *Eine Zeitreihe $\{X_t\}$ heißt GARCH(p,q)-Zeitreihe, wenn sie rekursiv durch*

$$X_t = \sigma_t Z_t \;\; mit \;\; \sigma_t^2 = \omega + \sum_{i=1}^{q} \alpha_i X_{t-i}^2 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2$$

definiert ist, wobei $p \in \mathbb{N}_0$, $q \in \mathbb{N}$, $\omega > 0$, $\alpha_1, ..., \alpha_q, \beta_1, ..., \beta_p \geq 0$ und Z_t ist ein unabhängig, identisch verteilter Noise-Prozess mit $\mathbb{E}(Z_t) = 0$ und $Var(Z_t) = 1$ für alle $t \in \mathbb{Z}$.

In dieser Arbeit wird vor allem das GARCH(1,1)-Modell verwendet, welches für $\alpha_1 + \beta_1 < 1$ stationär ist. Die standardisierten Residuen eines GARCH(1,1)-Modells sind durch

$$\hat{Z}_t = X_t / \hat{\sigma}_t$$

gegeben, wobei $\hat{\sigma}_t$ die Schätzung für den Parameter σ_t ist. Die Parameter werden mit Hilfe der Likelihood-Funktion numerisch geschätzt. Nachdem ein Modell angepasst wird, kann man mit Hilfe des Ljung-Box Test prüfen, ob die Residuen unabhängig sind. Es wird getestet

$$H_0 : (\hat{Z}_t)_{t=1,...,n} \text{ sind unabhängig} \quad vs. \quad H_1 : \neg H_0.$$

Der Ljung-Box Test wird dann konstruiert, indem man die empirische Autokorrelationsfunktion berechnet, welche für die Lags $k = 1, ..., n-1$ durch

$$\hat{\rho}_h = \frac{\sum_{t=h+1}^{n} \hat{Z}_t \hat{Z}_{t-h}}{\sum_{t=1}^{n} \hat{Z}_t^2}$$

definiert ist. Die Teststatistik ist dann durch

$$Q = n(n+2) \sum_{h=1}^{m} \frac{\hat{\rho}_h^2}{n-h}$$

gegeben, wobei mit m die Anzahl der Lags angegeben ist, welche getestet werden sollen. Unter der Nullhypothese ist die Teststatistik χ^2-verteilt mit $m - p$ Freiheitsgraden, wobei p die Anzahl der Parameter des Modells sind. [3][5][6]

4.3 DAX-Daten

Es wird nun ein Portfolio betrachtet, welches einige Positionen aus dem DAX (Deutscher Aktienindex) umfasst. Es handelt sich dabei um folgende Unternehmen:

- Deutsche Bank AG (DBK)
- Commerzbank AG (CBK)
- Deutsche Post AG (DPW)
- Deutsche Telekom AG (DTE)
- Infineon Technologies AG (IFX)
- Deutsche Lufthansa AG (LHA)
- RWE AG (RWE)
- Sap AG (SAP)
- ThyssenKrupp AG (TKA)

Es werden je Daten von 4. Jänner 2010 bis 31. Dezember 2012, also 750 Datensätze untersucht. Dabei werden die ersten 650 Datensätze verwendet, um diese an GARCH(1,1) Modelle anzupassen. Die transformierten standardisierten Residuen werden dann mit Hilfe von Vine Copulas modelliert. Zum Schluss wird der tägliche VaR für den Zeitraum 10. Juli 2012 bis 31. Dezember 2012 berechnet und mit den aktuellen Daten verglichen. Die verwendeten Daten sind von `http://finance.yahoo.com` heruntergeladen worden. In Abbildung

16 wird die Entwicklung der Indizes von 4. Jänner 2010 bis 9. Juli 2012 dargestellt.

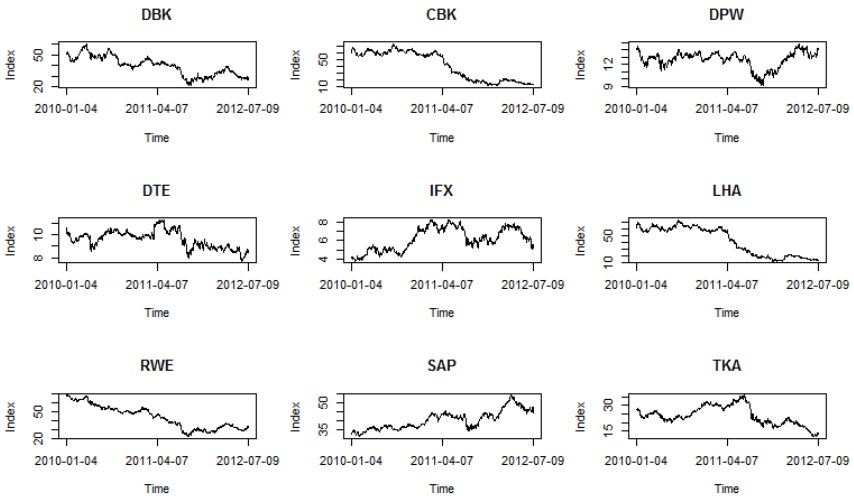

Abbildung 16: Entwicklung der Indizes

Für die Berechnungen werden die Log-Renditen verwendet. Diese werden in Abbildung 17 veranschaulicht.

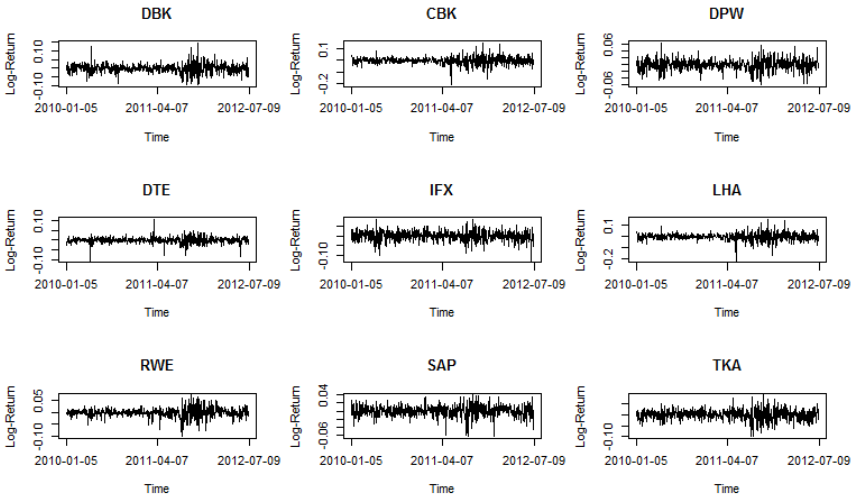

Abbildung 17: Entwicklung der Log-Renditen

Für die Zeitreihen Analyse in `R` kann nun das Paket `fGarch` verwendet werden. [16] Zuerst müssen aber die Log-Renditen berechnet werden, wobei man beachten muss, dass die Daten in der richtigen Reihenfolge vorliegen. Die aktuellen Daten sollen am Ende des Vektors gespeichert werden. Die Funktion `garchFit` aus dem Paket `fGarch` passt ein GARCH(1,1)-Modell an die ersten 649 Log-Returns an. Als Noise-Prozess wird eine Standardnormalverteilung verwendet.

```
> dbk1<-rev(dbk[,5])
> dbk2<-diff(log(dbk1))
> fit2<-garchFit(formula = ~ garch(1, 1), data = dbk2[1:649],
+ cond.dist ="norm")
```

Die Funktion `garchFit` berechnet auch den p-value für den Ljung-Box Test für verschieden Lags m und die Nullhypothese kann nicht verworfen werden. An den standardisierten Residuen wird dann eine empirische Verteilungsfunktion angepasst. Mit Hilfe dieser Verteilungsfunktion werden die Residuen auf $[0, 1]$ transformiert. Es könnte natürlich auch die Standardnormalverteilung für die Transformation verwendet werden. Die transformierten Daten besitzen einen Erwartungswert von 0.5007704 und eine Varianz von 0.08346174 $\approx 1/12$. Außerdem kann man beim Kolmogorov-Smirnov Test die Nullhypothese bei einem Signifikanzniveau von $q = 0.05$ nicht verwerfen und es kann angenommen werden, dass die transformierten Residuen uniform verteilt sind.

```
residuals1<-residuals(fit1,standardize=TRUE)
dist1<-ecdf(residuals1)
```

Für alle 9 Kurse werden GARCH(1,1)-Modelle angepasst, wobei für die Noise-Prozesse entweder die Normalverteilung oder die Student-t-Verteilung verwendet wird. Natürlich können aber auch andere Verteilungen für die Noise-Prozesse verwendet werden, doch mit den gewählten Verteilungen erhält man die "besten" Anpassungen. Die transformierten Residuen werden als `data.frame` im Objekt `A` gespeichert.

```
> plot(dist9(residuals9),dist9(residuals9))
> A<-matrix(nrow=length(residuals1),ncol=9,
+ dimnames=list(NULL,c("dbk","cbk","dpw","dte","ifx","lha","rwe","sap","tka")))
> A[,1]=dist1(residuals1)
> A[,2]=dist2(residuals2)
> A[,3]=dist3(residuals3)
> A[,4]=dist4(residuals4)
```

```
> A[,5]=dist5(residuals5)
> A[,6]=dist5(residuals6)
> A[,7]=dist5(residuals7)
> A[,8]=dist5(residuals8)
> A[,9]=dist5(residuals9)
> A<-data.frame(A)
```

Nun kann man sich den ersten Überblick über die Abhängigkeiten machen. In Abbildung 18 werden die paarweisen Streudiagramme der ersten 5 Unternehmen veranschaulicht. Außerdem wird Kendall's Tau für die verschiedenen Paarungen berechnet. Nun kann das Paket `VineCopula`, welches ausschließlich für

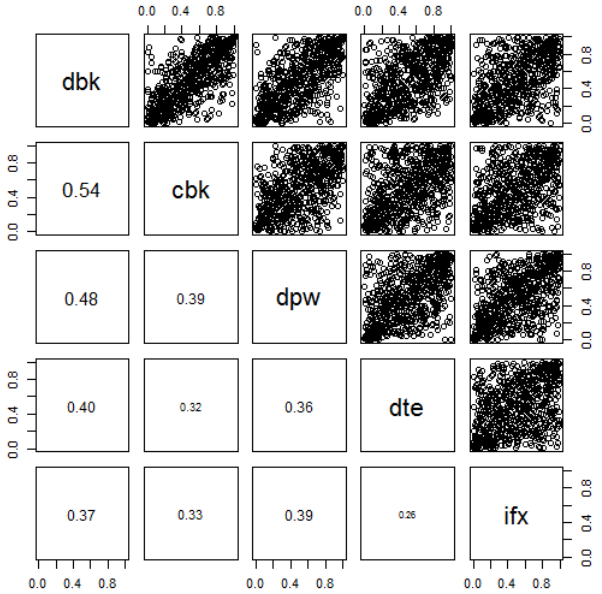

Abbildung 18: Kendall's Tau und Streudiagramme

uniform verteilte Daten auf $[0, 1]$ verwendet werden kann, genutzt werden, um eine R-Vine Copula anzupassen. [14] Der Algorithmus 3 aus dem vorigen Kapitel ist in der Funktion `RVineStructureSelect` implementiert. Diese Funktion kann nun für die Berechnungen verwendet werden, wobei als Auswahlkriterium für die bivariaten Copulatypen der AIC verwendet wird. Die Variablen werden dabei mit den Zahlen 1 bis 9 kodiert. Die geschätzten Parameter der Copula können mit `rvm$par` und `rvm$par2` aufgerufen werden. Die verwendeten bivariaten Copulatypen erhält man mit `rvm$family`, wobei diese auch kodiert sind. Die vollständige Liste der möglichen Copulas und deren Kodierung kann in der Anleitung des `VineCopula` Pakets gefunden werden.

```
> rvm<-RVineStructureSelect(data=A)
> rvm
R-vine matrix:
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
 [1,]    2    0    0    0    0    0    0    0    0
 [2,]    4    5    0    0    0    0    0    0    0
 [3,]    7    4    9    0    0    0    0    0    0
 [4,]    8    7    4    6    0    0    0    0    0
 [5,]    5    8    7    4    8    0    0    0    0
 [6,]    9    1    8    7    4    3    0    0    0
 [7,]    6    6    1    8    7    4    1    0    0
 [8,]    3    3    6    1    3    7    4    7    0
 [9,]    1    9    3    3    1    1    7    4    4

Where
1 <-> dbk
2 <-> cbk
3 <-> dpw
4 <-> dte
5 <-> ifx
6 <-> lha
7 <-> rwe
8 <-> sap
9 <-> tka
> rvm$family
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
 [1,]    0    0    0    0    0    0    0    0    0
 [2,]    3    0    0    0    0    0    0    0    0
 [3,]    2    3    0    0    0    0    0    0    0
 [4,]   23  124    3    0    0    0    0    0    0
 [5,]    2    1   10  124    0    0    0    0    0
 [6,]    5    2    5    5    5    0    0    0    0
 [7,]    5    5   14    5    5    5    0    0    0
 [8,]    1   10    1    2   20   20  214    0    0
 [9,]    2   14   14   14   14    5    5    5    0
```

Außerdem kann man mit der Funktion `copulaFromFamilyIndex` den Copulatyp entschlüsseln. Für die im ersten Kapitel kennengelernten Copulatypen gilt: 0=Unabhängigkeitscopula, 1=Normal Copula, 2=t-Copula, 3=Clayton Copula,

4=Gumbel Copula, 5=Frank Copula. Wenn beispielsweise vor dem Code noch eine 1 hinzugefügt wird, dann handelt es sich um die 180° gedrehte Copula.

Die Auswahl der bivariaten Copulas kann auch eingeschränkt werden. Mit dem Parameter `familyset` kann man einen Vektor mit den kodierten Copulatypen angeben, welche verwendet werden sollen. Beispielsweise kann mit `familyset=1` eine mehrdimensionale Normal Copula modelliert werden. Mit dem Parameter `type` kann man angeben ob eine R-Vine Struktur (0) oder speziell eine C-Vine Struktur (1) aufgebaut werden soll.

```
rvmG<-RVineStructureSelect(data=A,familyset=1)
rvmC<-RVineStructureSelect(data=A,type=1)
```

Wenn `indeptest=TRUE` gesetzt wird, dann wird der Unabhängigkeitstest aus dem vorigen Kapitel verwendet.

```
rvmind<-RVineStructureSelect(data=A,indeptest=TRUE)
> rvmind$family
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
 [1,]    0    0    0    0    0    0    0    0    0
 [2,]    0    0    0    0    0    0    0    0    0
 [3,]    0    0    0    0    0    0    0    0    0
 [4,]    0  124    0    0    0    0    0    0    0
 [5,]    0    1   10    0    0    0    0    0    0
 [6,]    5    2    5    5    5    0    0    0    0
 [7,]    5    5   14    5    5    5    0    0    0
 [8,]    1   10    1    2   20   20  214    0    0
 [9,]    2   14   14   14   14    5    5    5    0
```

Wie man hier erkennt, kann in den höheren Bäumen öfters die Unabhängigkeitscopula verwendet werden. Es empfiehlt sich hier auch das Modell etwas zu vereinfachen. Mit dem Parameter `trunclevel` kann das K festgelegt, so dass ab Baum T_K nur noch UnabhängigkeitsCopulas angepasst werden.

```
rvmK<-RVineStructureSelect(data=A,trunclevel=4)
```

Man kann sich aber auch zuerst eine Vine Struktur vorgeben und dann können sequentiell die bivariaten Copulatypen anpasst werden. Als Beispiel wird eine beliebige D-Vine Struktur aufgebaut.

```
> d=c(7,4,6,1,3,8,9,5,2)
> D=diag(d)
```

```
> for(i in 1:8){
+ D[(i+1):9,i]=rev(d[(i+1):9])
+ }
> D
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
 [1,]    7    0    0    0    0    0    0    0    0
 [2,]    2    4    0    0    0    0    0    0    0
 [3,]    5    2    6    0    0    0    0    0    0
 [4,]    9    5    2    1    0    0    0    0    0
 [5,]    8    9    5    2    3    0    0    0    0
 [6,]    3    8    9    5    2    8    0    0    0
 [7,]    1    3    8    9    5    2    9    0    0
 [8,]    6    1    3    8    9    5    2    5    0
 [9,]    4    6    1    3    8    9    5    2    2
```

Mit der Funktion `RVineCopSelect` werden dann sequentiell alle Copulatypen angepasst, wobei als Auswahlkriterium der AIC verwendet wird.

```
> DD<-RVineCopSelect(data=A,Matrix=D)
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
 [1,]    0    0    0    0    0    0    0    0    0
 [2,]  134    0    0    0    0    0    0    0    0
 [3,]  224    3    0    0    0    0    0    0    0
 [4,]    5   34  204    0    0    0    0    0    0
 [5,]    5   14    5    5    0    0    0    0    0
 [6,]    5    5    1    4   10    0    0    0    0
 [7,]    2    5    2    2   10    5    0    0    0
 [8,]    5    5   20    5    5    5   20    0    0
 [9,]    5   14   20    5   14   20   14    2    0
> rvmD<-RVineMatrix(Matrix=D,family=DD$family,par=DD$par,par2=DD$par2,
+ names=c("dbk","cbk","dpw","dte","ifx","lha","rwe","sap","tka"))
```

Die verschiedenen Modelle, welche konstruiert worden sind, haben alle, bis auf das C-Vine und D-Vine Modell, die gleiche R-Vine Struktur. Diese Modelle unterscheiden sich aber trotzdem noch durch ihre Copulatypen und die zugehörigen Parameter. Der AIC bzw. BIC der R-Vine Modelle kann mit der Funkion `RVineAIC` bzw. RVineBIC berechnet werden.

Typ	AIC	BIC
rvm	-3313.288	-3098.467
rvmG	-3073.466	-2912.351
rvmC	-3332.642	-3108.87
rvmD	-3320.24	-3100.944
rvmind	-3319.241	-3149.175
rvmK	-3298.666	-3150.977

Auch der Vuong Test ist in dem Paket `VineCopula` vorhanden. Mit diesem Test können die Modelle untereinander noch verglichen werden. Beispielsweise sollte das vereinfachte Modell `rvmK` dem Modell `rvmG` bevorzugt werden.

```
> RVineVuongTest(A,rvmK,rvmG)
$statistic
[1] 3.191722

$statistic.Akaike
[1] 3.279087

$statistic.Schwarz
[1] 3.474584

$p.value
[1] 0.001414273

$p.value.Akaike
[1] 0.001041436

$p.value.Schwarz
[1] 0.0005116463
```

Mit der Funktion `RVineSim` ist es nun möglich Zufallszahlen einer R-Vine Copula zu ziehen. Diese Zufallszahlen können dann rücktransformiert werden und die so erhaltenen geschätzten Residuen können für die Zeitreihen Analyse verwendet werden.

Im Buch [9] wird ein Algorithmus vorgestellt, welcher die berechneten VaR mit den aktuellen Daten vergleicht.

Algorithmus 6 : VaR *Backtesting*

for $t = 651, ..., 775$ **do**
  for $j = 1, ..., 9$ **do**
    Berechne, um einen Schritt, die Vorhersage für $\sigma_{j,t}$ bei gegebener Information bis zum Zeitpunkt t;
  end
  for $k = 1, ..., 1000$ **do**
    Generiere Zufallszahlen $u_1^{(k)}, ..., u_9^{(k)}$ der angepassten R-Vine Copula;
    Transformiere mit Hilfe der zugehörigen inversen Verteilungsfunktionen $u_1^{(k)}, ..., u_9^{(k)}$ zurück, um die geschätzten Residuen $z_1^{(k)}, ..., z_9^{(k)}$ zu erhalten;
    for $j = 1, ..., 9$ **do**
      Berechne die Log-Returns $r_{j,t}^{(k)} = c_{j,t} + \sigma_{j,t} z_j^{(k)}$, wobei $c_{j,t}$ der Mittelwert der letzten 100 beobachteten Log-Returns ist;
    end
    Berechne den Log-Return des Portfolios $r_{p,t}^{(k)} = \sum_{j=1}^{9} \frac{1}{9} r_{j,t}^{(k)}$;
  end
  for $q \in \{0.005, 0.01, 0.05\}$ **do**
    Berechne den täglichen VaR_t^q als das q-Quantil der Stichproben $r_{p,t}^{(k)}$;
  end
end

Für jedes $q \in \{0.005, 0.01, 0.05\}$ bestimme die Anzahl x_q der VaR_t^q welche größer als die beobachteten $r_{p,t}$ sind, wobei $t = 651, ..., 775$. Mit der Anzahl x_q und $T = 125$ kann der POF Test für jedes $q \in \{0.005, 0.01, 0.05\}$ angewendet werden, um die Vorhersage des VaR zu bewerten. Die Ergebnisse für das Modell `rvmK` werden in der folgenden Tabelle zusammengefasst. Es wurde das Modell `rvmK` gewählt, weil es das einfachste ist. Dieses besitzt den geringsten Rechenaufwand und liefert trotzdem brauchbare Ergebnisse.

q	0.005	0.01	0.05
beobachtet	0	0	5
erwartet	0.625	1.25	6.25
p-value	0.2629545	0.1129406	0.5956045

In Abbildun 19 werden die Log-Returns der aktuellen Daten von 10. Juni 2012 bis 31. Dezember 2012 mit den berechneten VaR verglichen.

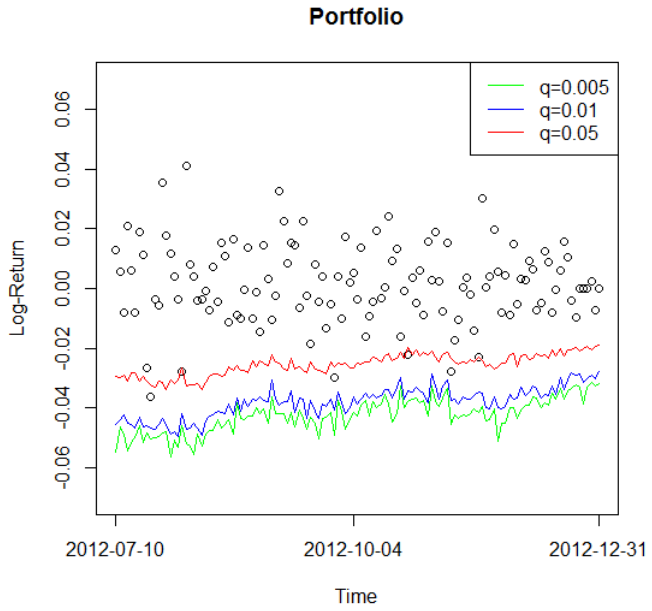

Abbildung 19: Punkte: die aktuellen Log-Returns; Linien: VaR für $q \in \{0.005, 0.01, 0.05\}$

Man kann Algorithmus 6 auch umändern, so dass keine Vine Copula verwendet werden muss. Man würde die Residuen direkt aus der jeweiligen Verteilung des Noise-Prozesses generieren (Standardnormalverteilung bzw. Student-t-Verteilung). Das Ergebnis wäre dann um einiges schlechter, siehe folgende Tabelle.

q	0.005	0.01	0.05
beobachtet	8	11	15
erwartet	0.625	1.25	6.25
p-value	2.652704e-07	6.755068e-08	0.002140185

Mit der Funktion `RVineTreePlot` ist es außerdem möglich alle Bäume einer R-Vine Struktur zu zeichnen. Die ersten 4 Bäume der R-Vine Struktur `rvm` sind in Abbildung 20 und Abbildung 21 aufgeführt. Bei den restlichen 4 Bäumen handelt es sich um D-Vine Strukturen. [9]

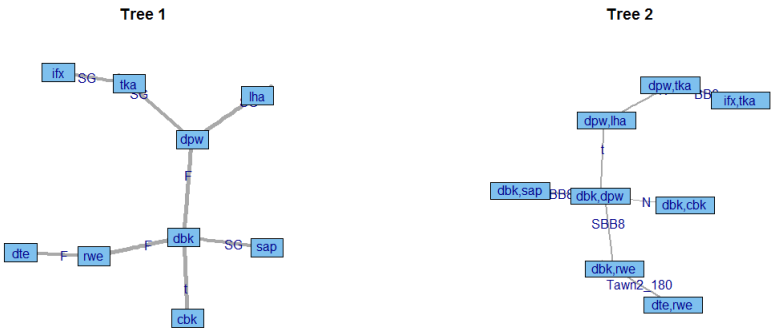

Abbildung 20: Baum 1 und Baum 2

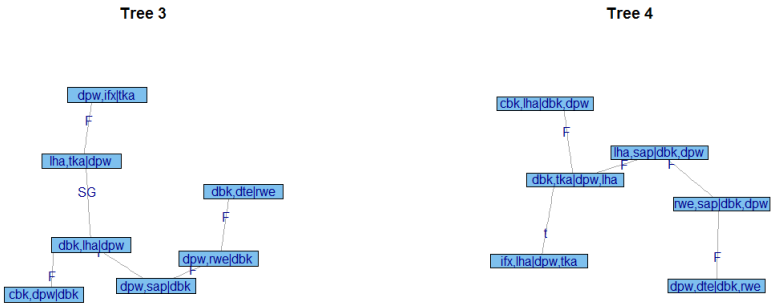

Abbildung 21: Baum 3 und Baum 4

5 Zusammenfassung

In dieser Arbeit wurde gezeigt, wie Copula Funktionen für die Modellierung von Abhängigkeiten verwendet werden können. Zu Beginn der Arbeit sind die wichtigsten Eigenschaften von Copula Funktionen näher gebracht worden. Dann wurde gezeigt wie man mit bivariaten Copulas eine mehrdimensionale Copula konstruieren kann. Es wurde dann gezeigt, dass mit Hilfe der Matrizen **M**, **T** und **P** alle Berechnungen für eine Vine Copula gemacht werden können. Um R-Vine Modelle an die Daten anzupassen, wurde eine sequentielle Methode verwendet, welche kein Optimum liefern muss. Für die abschließenden Berechnungen des VaR war diese Methode aber ausreichend. Vine Copulas sind sehr flexibel und können daher in den verschiedensten Anwendungsbereichen verwendet werden.

Natürlich können einige Methoden in der Theorie der Vine Copulas noch verbessert werden. Beispielsweise wäre man daran interessiert einen Algorithmus zu finden, welcher mit geringer Laufzeit die “beste“ R-Vine Copula an die Daten anpasst. Außerdem ist es sinnvoll im Bereich der bivariaten Copula Funktionen zu forschen, denn dann könnten noch “bessere“ R-Vine Copulas konstruiert werden.

Abbildungsverzeichnis

Literatur

[1] Aas K. (2004), Modelling the dependence structure of financial assets: A survey of four Copulas, Norwegian Computing Center

[2] Brechmann E.C. (2010), Truncated and simplified regular vines and their applications, Technische Universität München

[3] Brockwell P.J., Davis R.A. (2002), Introduction to Time Series and Forecasting, New York, Springer

[4] Buchacher G. (2013), LV Finanzstatistik, Universität Klagenfurt

[5] Dißmann J.F. (2010), Statistical Inference for Regular Vines and Application, Technische Universität München

[6] Dißmann J., Brechmann E.C., Czado C., Kurowicka D. (2012), Selecting and estimating regular vine copulas and application to financial returns, Technische Universität München, Delft University of Technology

[7] Dowd K. (2005), Measuring Market Risk; John Wiley & Sons Ltd

[8] Hlawatsch S., Reichling P. (2010), Konstruktion und Anwendung von Copulas in der Finanzwirtschaft, Working Paper No. 16/2010, Otto-von-Guericke-Universität Magdeburg Falkultät für Wirtschaftswissenschaft

[9] Kurowicka D., Joe H. (2011), Dependence Modeling, Vine Copula Handbook, Word Scientific

[10] Mai J., Scherer M. (2012), Simulating Copulas, Imperial College Press

[11] Nelsen R.B. (2006), An Introduction to Copulas, Springer Series in Statistics

[12] Neubauer D. (2013), Return and VaR prediction of equity portfolios using DAG and R-vine copula based models, Technische Universität München

[13] R Core Team (2014), R: A language and environment for statistical computing, R Foundation for Statistical Computing, Vienna, Austria, `http://www.R-project.org/`

[14] Schepsmeier U., Stoeber J., Brechmann E.C., Graeler B. (2014), VineCopula: Statistical inference of vine copulas, R package version 1.3, `http://CRAN.R-project.org/package=VineCopula`

[15] Schirmacher D., Schirmacher E. (2008), Multivariate Dependence Modeling using Pair-Copulas, Liberty Mutual Group, 175 Berkeley St., Boston MA 02116, USA

[16] Wuertz D., Chalabi Y. (2013), fGarch: Rmetrics - Autoregressive Conditional Heteroskedastic Modelling, R package version 3010.82, `http://CRAN.R-project.org/package=fGarch`