Statistical inference of vine copulas using the R-package `VineCopula`

Eike Christian Brechmann
brechmann@ma.tum.de

Technische Universität München

May 23, 2013

Available software for vines

In general...

- "Uncertainty analysis with Correlations" (UNICORN, TU Delft) includes some functionality for vines.

In R...

- Packages for bivariate and multivariate copulas (`copula`, `fCopulae`, `QRMlib`,...).
- Daniel Berg (U Oslo/NR): `copulaGOF`/`CopulaLib`.

The R-packages `VineCopula` and `CDVine` fill this gap.

Available software for vines

In general...

- "Uncertainty analysis with Correlations" (UNICORN, TU Delft) includes some functionality for vines.

In R...

- Packages for bivariate and multivariate copulas (`copula`, `fCopulae`, `QRMlib`,...).
- Daniel Berg (U Oslo/NR): `copulaGOF`/`CopulaLib`.

The R-packages `VineCopula` and `CDVine` fill this gap.

Scope of VineCopula

VineCopula provides functions for bivariate analysis...

- graphical tools
- analytical tools
- selection and estimation of bivariate copulas
- simulation of bivariate copulas

BiCop...

...and for multivariate analysis using (simplified) regular vine copulas.

- sequential and joint maximum likelihood estimation
- simulation of vine copulas
- model selection
- illustration of vine trees

RVine...

General assumption: Data lies in the unit hypercube $[0,1]^d$.

Scope of VineCopula

VineCopula provides functions for bivariate analysis...

- graphical tools
- analytical tools
- selection and estimation of bivariate copulas
- simulation of bivariate copulas

BiCop...

...and for multivariate analysis using (simplified) regular vine copulas.

- sequential and joint maximum likelihood estimation
- simulation of vine copulas
- model selection
- illustration of vine trees

RVine...

General assumption: Data lies in the unit hypercube $[0,1]^d$.

Scope of VineCopula

VineCopula provides functions for bivariate analysis...

- graphical tools
- analytical tools
- selection and estimation of bivariate copulas
- simulation of bivariate copulas

BiCop...

...and for multivariate analysis using (simplified) regular vine copulas.

- sequential and joint maximum likelihood estimation
- simulation of vine copulas
- model selection
- illustration of vine trees

RVine...

General assumption: Data lies in the unit hypercube $[0,1]^d$.

The package CDVine

- Functionality for the sub-classes of C- and D-vines.
- Links to the package VineCopula: C2RVine and D2RVine.
- Vignette:

  Brechmann & Schepsmeier (2013).
  Modeling dependence with C- and D-vine copulas:
  The R-package CDVine.
  Journal of Statistical Software 52(3), 1–27.

The building blocks: Bivariate copula families

Each family is denoted by a number to shorten notation (0 = indep.).

- Elliptical copulas:

  `family` = 1 Gaussian copula

  `family` = 2 Student's t copula

- One parameter Archimedean copulas:

- Two parameter Archimedean copulas:

Density, distribution & *h*-functions: `BiCopPDF`, `BiCopCDF` & `BiCopHfunc`.

The building blocks: Bivariate copula families

Each family is denoted by a number to shorten notation (0 = indep.).

- Elliptical copulas:

  `family` = 1 Gaussian copula
  `family` = 2 Student's t copula

- One parameter Archimedean copulas:

  `family` = 3 Clayton copula
  `family` = 4 Gumbel copula
  `family` = 5 Frank copula
  `family` = 6 Joe copula

- Two parameter Archimedean copulas:

Density, distribution & *h*-functions: `BiCopPDF`, `BiCopCDF` & `BiCopHfunc`.

The building blocks: Bivariate copula families

Each family is denoted by a number to shorten notation (0 = indep.).

- Elliptical copulas:

  `family` = 1 Gaussian copula
  `family` = 2 Student's t copula

- One parameter Archimedean copulas:

  `family` = 3 Clayton copula
  `family` = 4 Gumbel copula
  `family` = 5 Frank copula
  `family` = 6 Joe copula

- Two parameter Archimedean copulas:

  `family` = 7 Clayton-Gumbel (BB1) copula
  `family` = 8 Joe-Gumbel (BB6) copula
  `family` = 9 Joe-Clayton (BB7) copula
  `family` = 10 Joe-Frank (BB8) copula

Density, distribution & *h*-functions: `BiCopPDF`, `BiCopCDF` & `BiCopHfunc`.

The building blocks: Bivariate copula families

Each family is denoted by a number to shorten notation (0 = indep.).

- Elliptical copulas:
  - `family` = 1 Gaussian copula
  - `family` = 2 Student's t copula
- One parameter Archimedean copulas:
  - `family` = 13 survival Clayton copula
  - `family` = 14 survival Gumbel copula
  - `family` = 5 Frank copula
  - `family` = 16 survival Joe copula
- Two parameter Archimedean copulas:
  - `family` = 17 survival BB1 copula
  - `family` = 18 survival BB6 copula
  - `family` = 19 survival BB7 copula
  - `family` = 20 survival BB8 copula

Density, distribution & *h*-functions: `BiCopPDF`, `BiCopCDF` & `BiCopHfunc`.

The building blocks: Bivariate copula families

Each family is denoted by a number to shorten notation (0 = indep.).

- Elliptical copulas:

  `family` = 1 Gaussian copula
  `family` = 2 Student's t copula

- One parameter Archimedean copulas:

  `family` = 23 rotated Clayton copula (90 degrees)
  `family` = 24 rotated Gumbel copula (90 degrees)
  `family` = 5 Frank copula
  `family` = 26 rotated Joe copula (90 degrees)

- Two parameter Archimedean copulas:

  `family` = 27 rotated BB1 copula (90 degrees)
  `family` = 28 rotated BB6 copula (90 degrees)
  `family` = 29 rotated BB7 copula (90 degrees)
  `family` = 30 rotated BB8 copula (90 degrees)

Density, distribution & h-functions: `BiCopPDF`, `BiCopCDF` & `BiCopHfunc`.

The building blocks: Bivariate copula families

Each family is denoted by a number to shorten notation (0 = indep.).

- Elliptical copulas:

  `family` = 1 Gaussian copula
  `family` = 2 Student's t copula

- One parameter Archimedean copulas:

  `family` = 33 rotated Clayton copula (270 degrees)
  `family` = 34 rotated Gumbel copula (270 degrees)
  `family` = 5 Frank copula
  `family` = 36 rotated Joe copula (270 degrees)

- Two parameter Archimedean copulas:

  `family` = 37 rotated BB1 copula (270 degrees)
  `family` = 38 rotated BB6 copula (270 degrees)
  `family` = 39 rotated BB7 copula (270 degrees)
  `family` = 40 rotated BB8 copula (270 degrees)

Density, distribution & *h*-functions: `BiCopPDF`, `BiCopCDF` & `BiCopHfunc`.

The building blocks: Bivariate copula families

Each family is denoted by a number to shorten notation (0 = indep.).

- Elliptical copulas: (parameters: `par`, `par2` (degrees of freedom))

  `family` = 1 Gaussian copula
  `family` = 2 Student's t copula

- One parameter Archimedean copulas: (parameter: `par`)

  `family` = 33 rotated Clayton copula (270 degrees)
  `family` = 34 rotated Gumbel copula (270 degrees)
  `family` = 5 Frank copula
  `family` = 36 rotated Joe copula (270 degrees)

- Two parameter Archimedean copulas: (parameters: `par`, `par2`)

  `family` = 37 rotated BB1 copula (270 degrees)
  `family` = 38 rotated BB6 copula (270 degrees)
  `family` = 39 rotated BB7 copula (270 degrees)
  `family` = 40 rotated BB8 copula (270 degrees)

Density, distribution & *h*-functions: `BiCopPDF`, `BiCopCDF` & `BiCopHfunc`.

The building blocks: Bivariate copula families

Each family is denoted by a number to shorten notation (0 = indep.).

- Elliptical copulas: (parameters: `par`, `par2` (degrees of freedom))

  `family` = 1 Gaussian copula
  `family` = 2 Student's t copula

- One parameter Archimedean copulas: (parameter: `par`)

  `family` = 33 rotated Clayton copula (270 degrees)
  `family` = 34 rotated Gumbel copula (270 degrees)
  `family` = 5 Frank copula
  `family` = 36 rotated Joe copula (270 degrees)

- Two parameter Archimedean copulas: (parameters: `par`, `par2`)

  `family` = 37 rotated BB1 copula (270 degrees)
  `family` = 38 rotated BB6 copula (270 degrees)
  `family` = 39 rotated BB7 copula (270 degrees)
  `family` = 40 rotated BB8 copula (270 degrees)

Density, distribution & *h*-functions: `BiCopPDF`, `BiCopCDF` & `BiCopHfunc`.

Rotation of copulas

- Rotate Archimedean copulas to capture negative dependence: if $(U_1, U_2) \sim C_{90^\circ}$, then $(1 - U_1, U_2) \sim C_{0^\circ}$.
- Survival copulas correspond to rotation by 180 degrees.

Clayton copulas rotated by 0, 90, 180 and 270 degrees

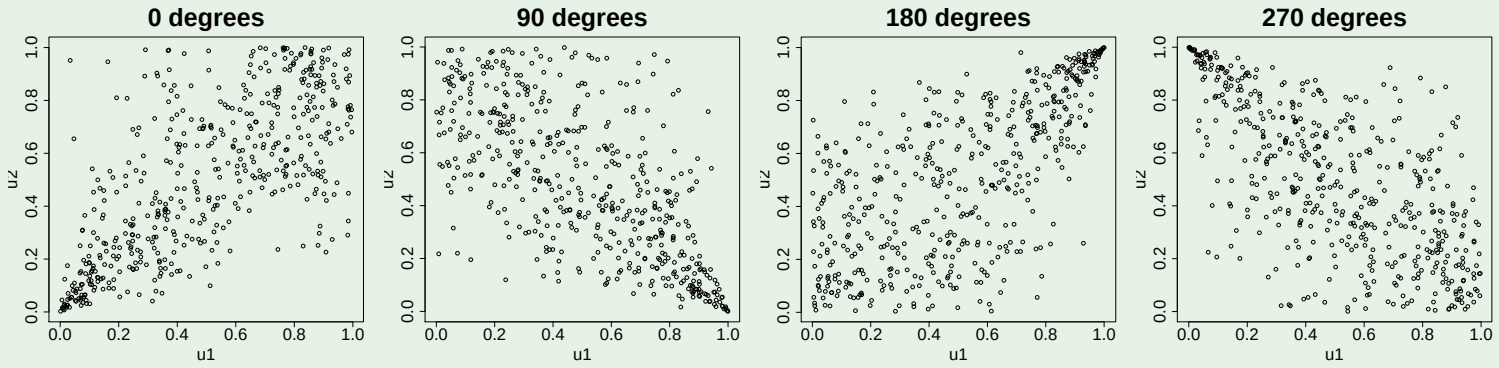
```
> dat0 = BiCopSim(N=500, family=3, par=2)
> dat90 = BiCopSim(N=500, family=23, par=-2)
> dat180 = BiCopSim(N=500, family=13, par=2)
> dat270 = BiCopSim(N=500, family=33, par=-2)
```

Rotation of copulas

- Rotate Archimedean copulas to capture negative dependence: if $(U_1, U_2) \sim C_{90^\circ}$, then $(1 - U_1, U_2) \sim C_{0^\circ}$.
- Survival copulas correspond to rotation by 180 degrees.

Clayton copulas rotated by 0, 90, 180 and 270 degrees

```
> dat0 = BiCopSim(N=500, family=3, par=2)
> dat90 = BiCopSim(N=500, family=23, par=-2)
> dat180 = BiCopSim(N=500, family=13, par=2)
> dat270 = BiCopSim(N=500, family=33, par=-2)
```



Two parameter Archimedean copulas

- The BB1 and the BB7 copula can model tail-asymmetric dependence with different non-zero lower and upper tail dependence.
- Density expressions and derivatives are however numerically involved.

Clayton-Gumbel (BB1) copula

```
> BiCopPar2Tau(family=7,
+     par=0.25, par2=2.5)

[1] 0.64

> BiCopPar2TailDep(family=7,
+     par=0.25, par2=2.5)

$lower
[1] 0.33

$upper
[1] 0.68
```

Two parameter Archimedean copulas

- The BB1 and the BB7 copula can model tail-asymmetric dependence with different non-zero lower and upper tail dependence.
- Density expressions and derivatives are however numerically involved.

Clayton-Gumbel (BB1) copula

```
> BiCopPar2Tau(family=7,
+     par=0.25, par2=2.5)

[1] 0.64

> BiCopPar2TailDep(family=7,
+     par=0.25, par2=2.5)

$lower
[1] 0.33

$upper
[1] 0.68
```

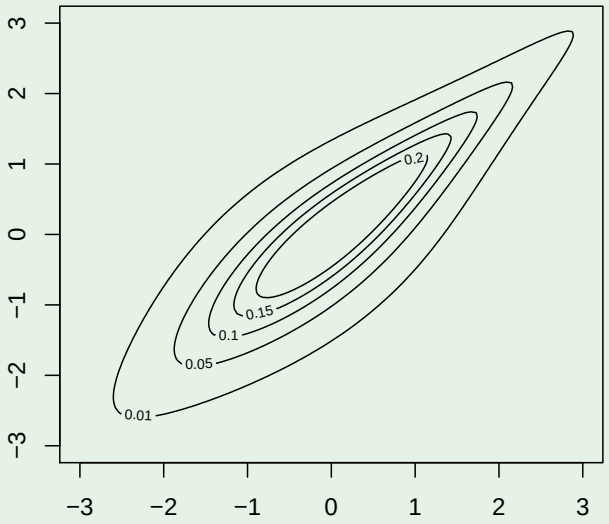
Two parameter Archimedean copulas

- The BB1 and the BB7 copula can model tail-asymmetric dependence with different non-zero lower and upper tail dependence.
- Density expressions and derivatives are however numerically involved.

Clayton-Gumbel (BB1) copula

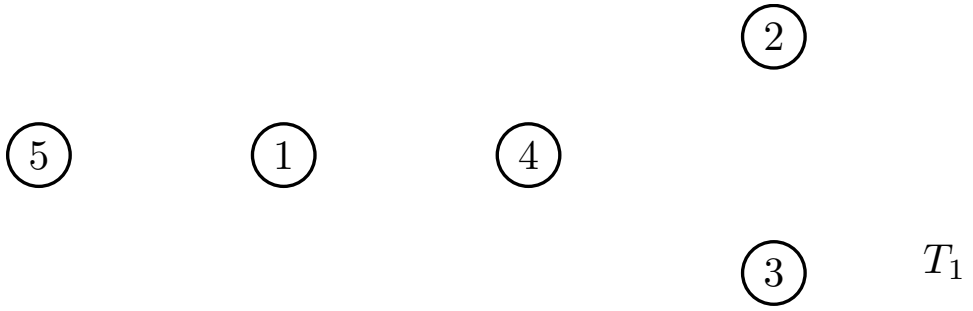
```
> BiCopPar2Tau(family=7,
+     par=0.25, par2=2.5)

[1] 0.64

> BiCopPar2TailDep(family=7,
+     par=0.25, par2=2.5)

$lower
[1] 0.33

$upper
[1] 0.68
```

```
> BiCopMetaContour(family=7,
+     par=0.25, par2=2.5)
```



Reminder: R-vine copulas

Three components: vine trees, pair copulas, copula parameters

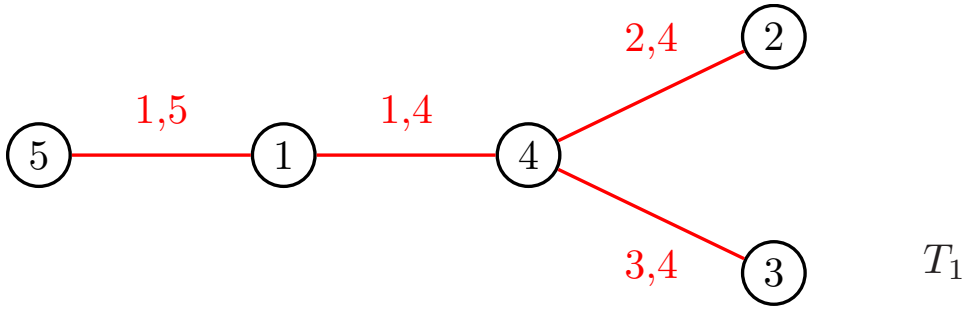


Density

$$c = c_{14} \cdot c_{15} \cdot c_{24} \cdot c_{34} \cdot c_{12;4} \cdot c_{13;4} \cdot c_{45;1} \cdot c_{23;14} \cdot c_{35;14} \cdot c_{25;134}$$

Reminder: R-vine copulas

Three components: vine trees, pair copulas, copula parameters

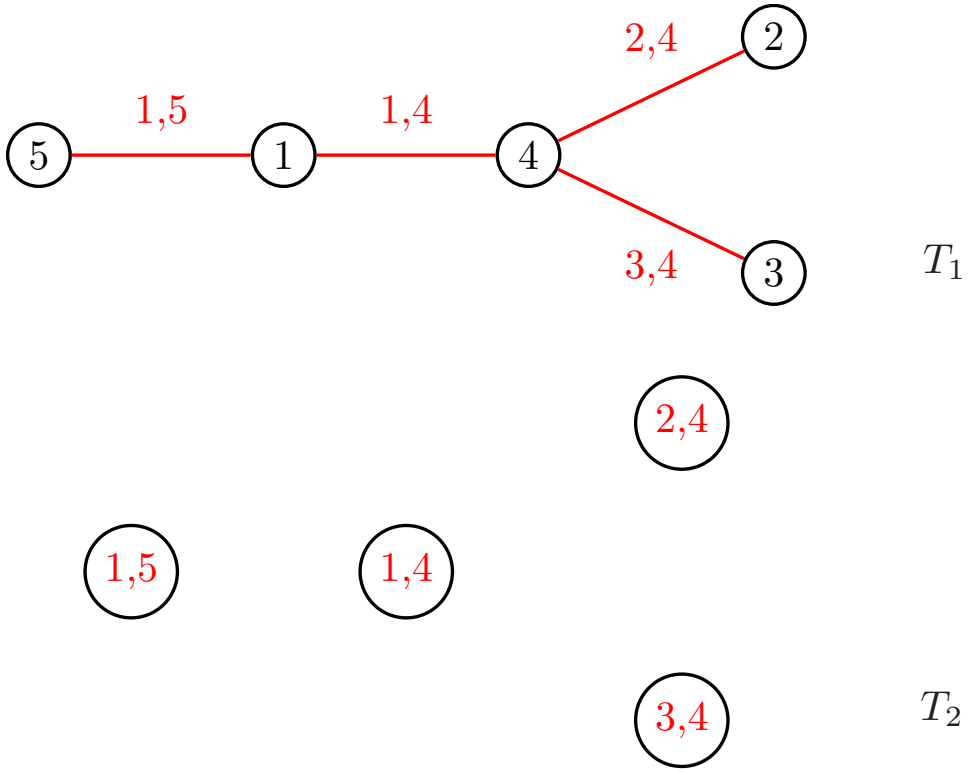


Density

$$c = c_{14} \cdot c_{15} \cdot c_{24} \cdot c_{34}$$
$$\cdot\, c_{12;4} \cdot c_{13;4} \cdot c_{45;1}$$
$$\cdot\, c_{23;14} \cdot c_{35;14}$$
$$\cdot\, c_{25;134}$$

Reminder: R-vine copulas

Three components: vine trees, pair copulas, copula parameters

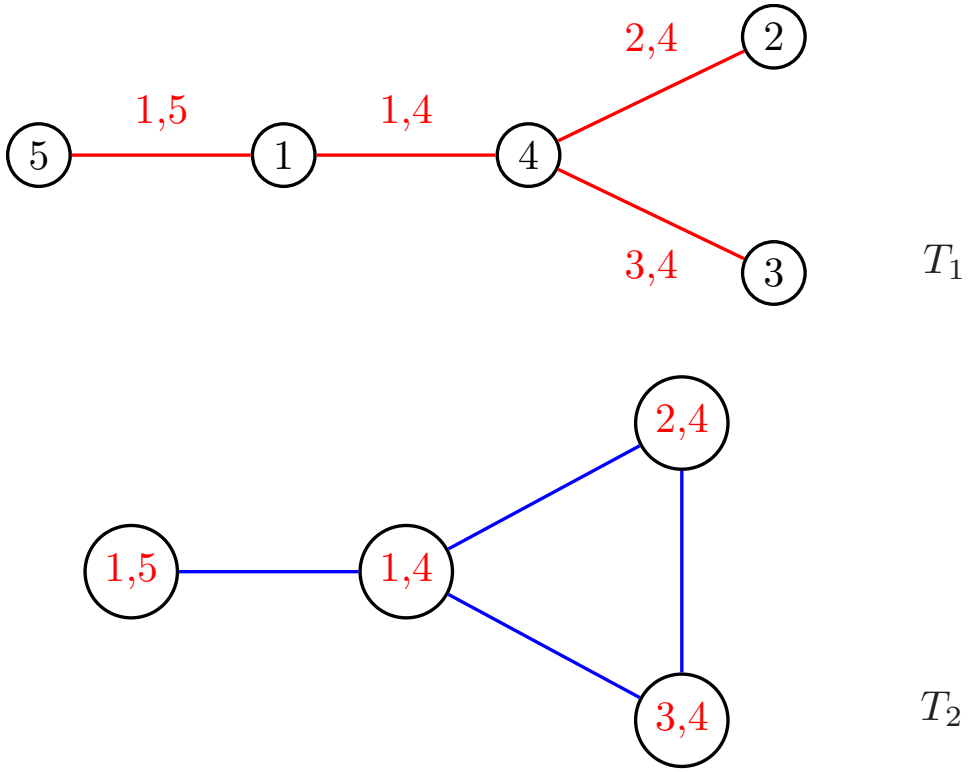


Density

$$c = c_{14} \cdot c_{15} \cdot c_{24} \cdot c_{34} \cdot c_{12;4} \cdot c_{13;4} \cdot c_{45;1} \cdot c_{23;14} \cdot c_{35;14} \cdot c_{25;134}$$

Reminder: R-vine copulas

Three components: vine trees, pair copulas, copula parameters

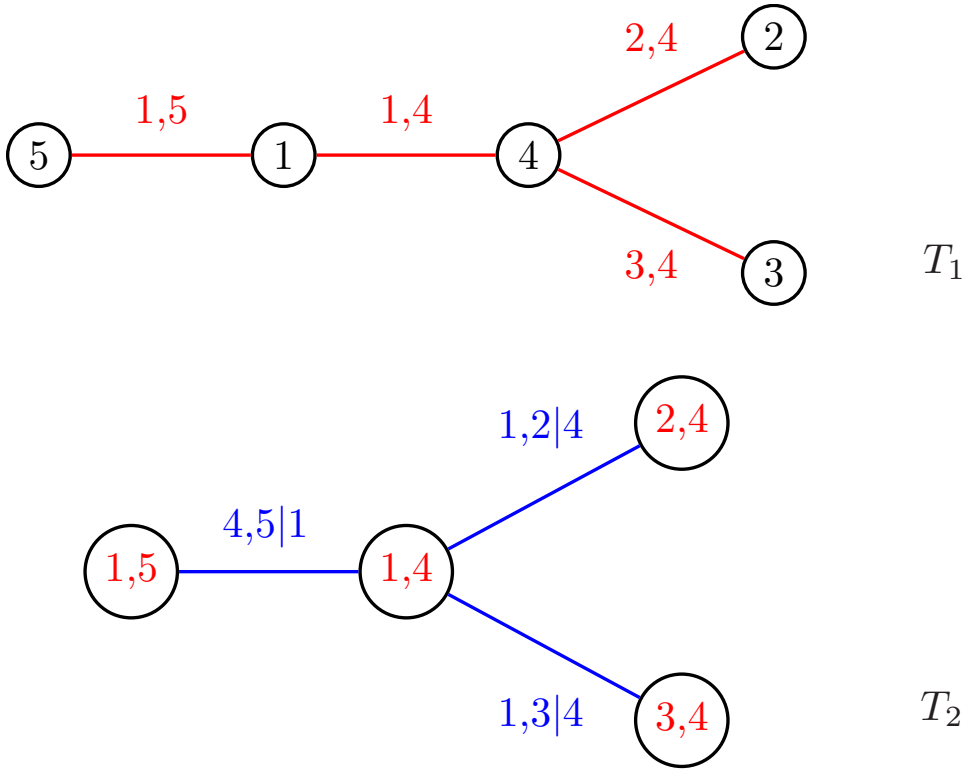


Density

$$c = c_{14} \cdot c_{15} \cdot c_{24} \cdot c_{34} \cdot c_{12;4} \cdot c_{13;4} \cdot c_{45;1} \cdot c_{23;14} \cdot c_{35;14} \cdot c_{25;134}$$

Reminder: R-vine copulas

Three components: vine trees, pair copulas, copula parameters

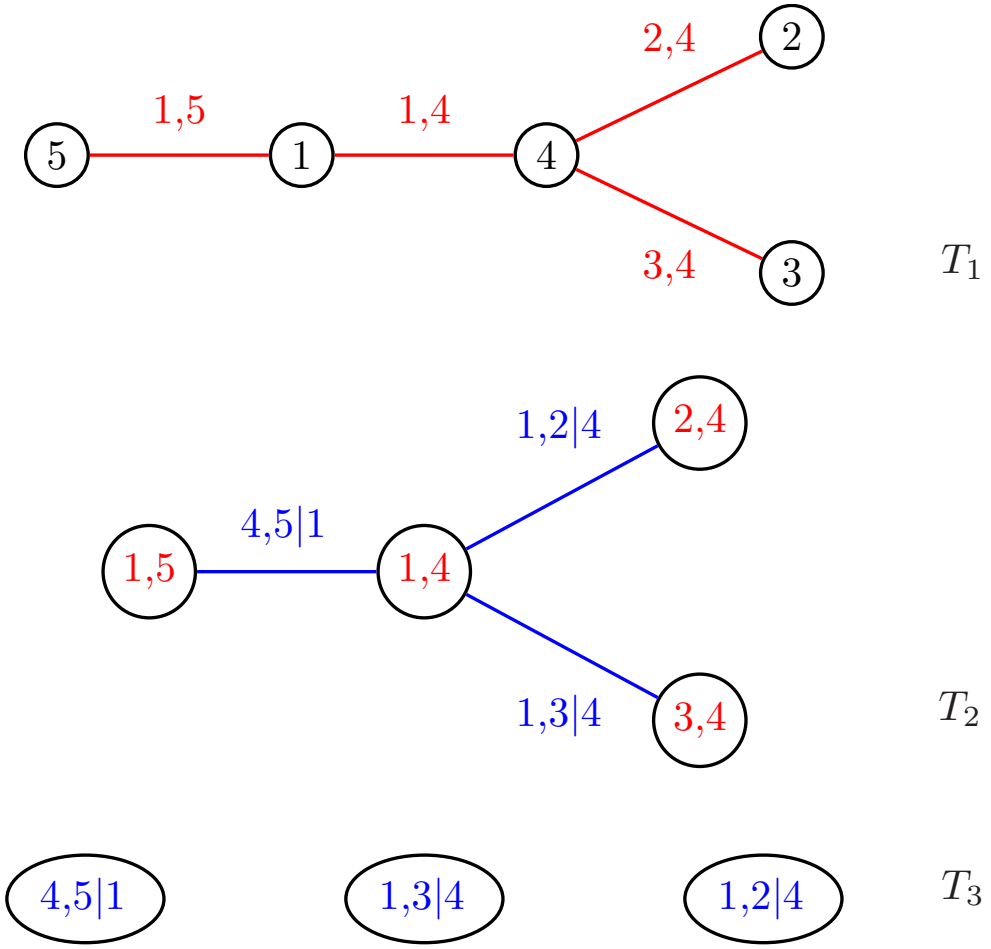


Density

$$c = c_{14} \cdot c_{15} \cdot c_{24} \cdot c_{34} \cdot c_{12;4} \cdot c_{13;4} \cdot c_{45;1} \cdot c_{23;14} \cdot c_{35;14} \cdot c_{25;134}$$

Reminder: R-vine copulas

Three components: vine trees, pair copulas, copula parameters

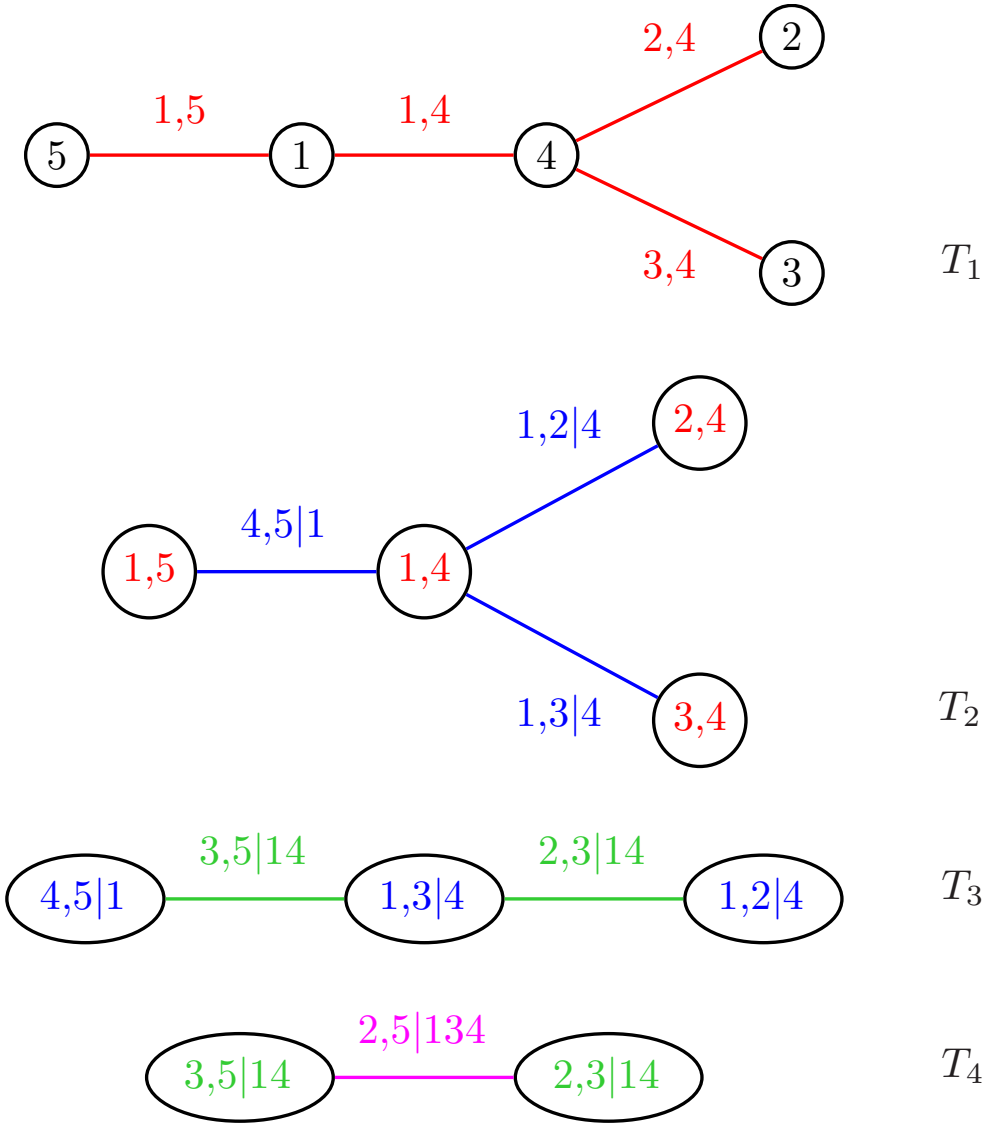


Density

$$c = c_{14} \cdot c_{15} \cdot c_{24} \cdot c_{34} \cdot c_{12;4} \cdot c_{13;4} \cdot c_{45;1} \cdot c_{23;14} \cdot c_{35;14} \cdot c_{25;134}$$

Reminder: R-vine copulas

Three components: vine trees, pair copulas, copula parameters



Density

$$c = c_{14} \cdot c_{15} \cdot c_{24} \cdot c_{34}$$
$$\cdot\, c_{12;4} \cdot c_{13;4} \cdot c_{45;1}$$
$$\cdot\, c_{23;14} \cdot c_{35;14}$$
$$\cdot\, c_{25;134}$$

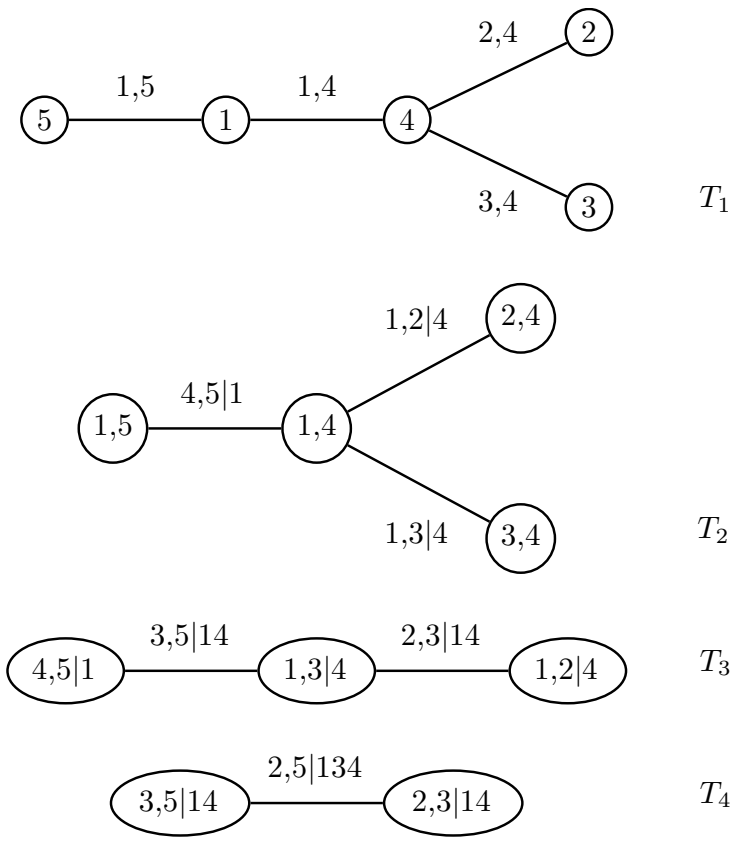
Storing R-vine copulas in matrix notation

Efficient enconding of R-vine models needed for statistical inference.

- Matrix notation by Morales-Nápoles et al. (2010) and Dißmann et al. (2013).

$$\begin{pmatrix} 2 & & & & \\ 5 & 3 & & & \\ 3 & 5 & 4 & & \\ 1 & 1 & 5 & 5 & \\ 4 & 4 & 1 & 1 & 1 \end{pmatrix}$$

1. $\{2,4\}, \{3,4\}, \{4,1\}, \{5,1\}$
2. $\{2,1|4\}, \{3,1|4\}, \{4,5|1\}$
3. $\{2,3|14\}, \{3,5|14\}$
4. $\{2,5|314\}$



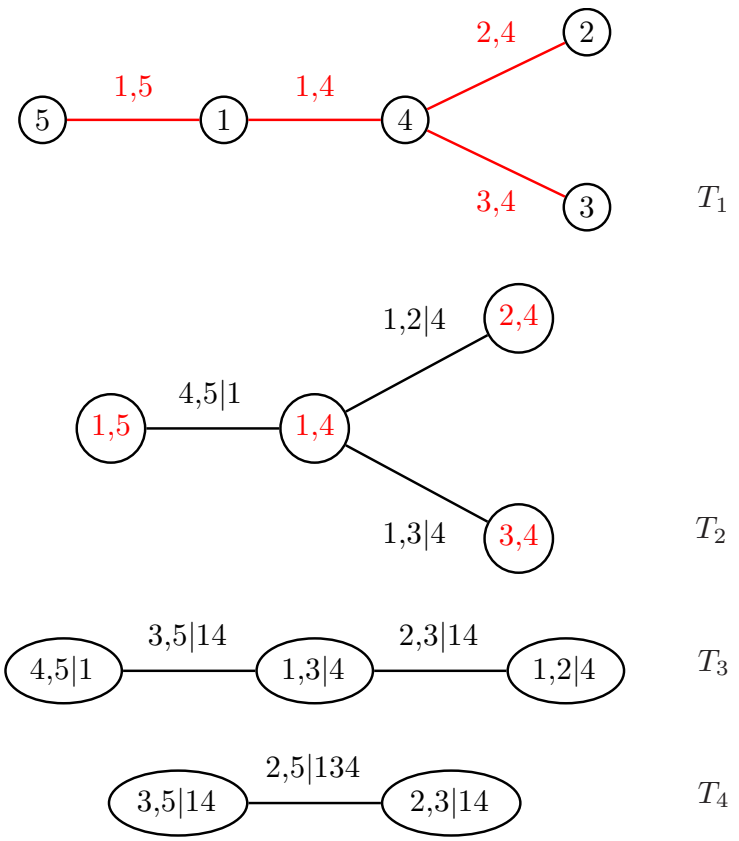
Storing R-vine copulas in matrix notation

Efficient enconding of R-vine models needed for statistical inference.

- Matrix notation by Morales-Nápoles et al. (2010) and Dißmann et al. (2013).

$$\begin{pmatrix} 2 & & & & \\ 5 & 3 & & & \\ 3 & 5 & 4 & & \\ 1 & 1 & 5 & 5 & \\ 4 & 4 & 1 & 1 & 1 \end{pmatrix}$$

1. $\{2,4\},\{3,4\},\{4,1\},\{5,1\}$
2. $\{2,1|4\},\{3,1|4\},\{4,5|1\}$
3. $\{2,3|14\},\{3,5|14\}$
4. $\{2,5|314\}$



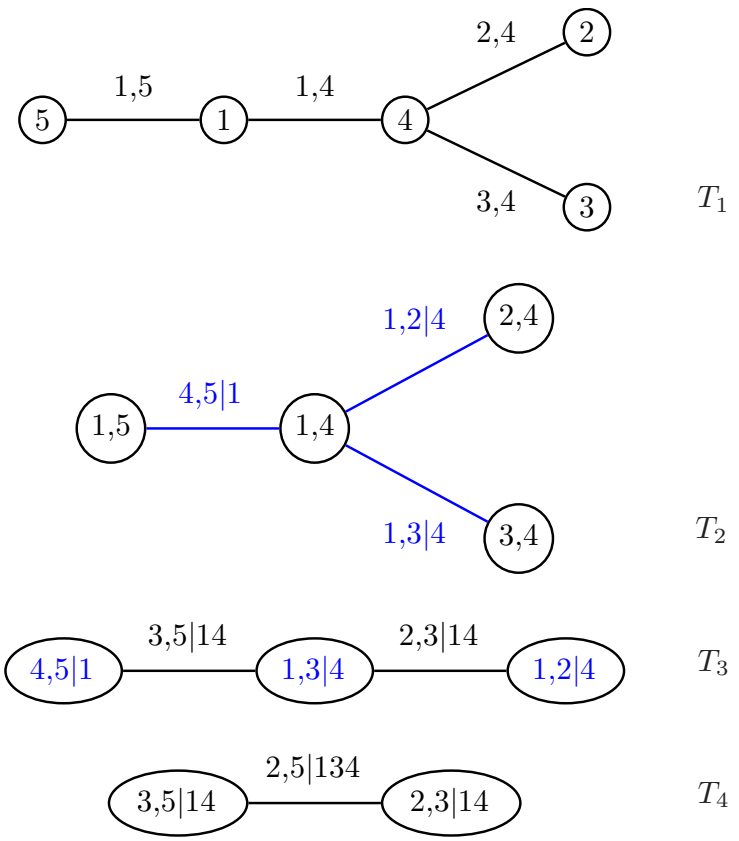
Storing R-vine copulas in matrix notation

Efficient enconding of R-vine models needed for statistical inference.

- Matrix notation by Morales-Nápoles et al. (2010) and Dißmann et al. (2013).

$$\left(\begin{array}{ccccc} 2 & & & & \\ 5 & 3 & & & \\ 3 & 5 & 4 & & \\ 1 & 1 & 5 & 5 & \\ \hline 4 & 4 & 1 & 1 & 1 \end{array}\right)$$

1. $\{2,4\},\{3,4\},\{4,1\},\{5,1\}$
2. $\{2,1|4\},\{3,1|4\},\{4,5|1\}$
3. $\{2,3|14\},\{3,5|14\}$
4. $\{2,5|314\}$



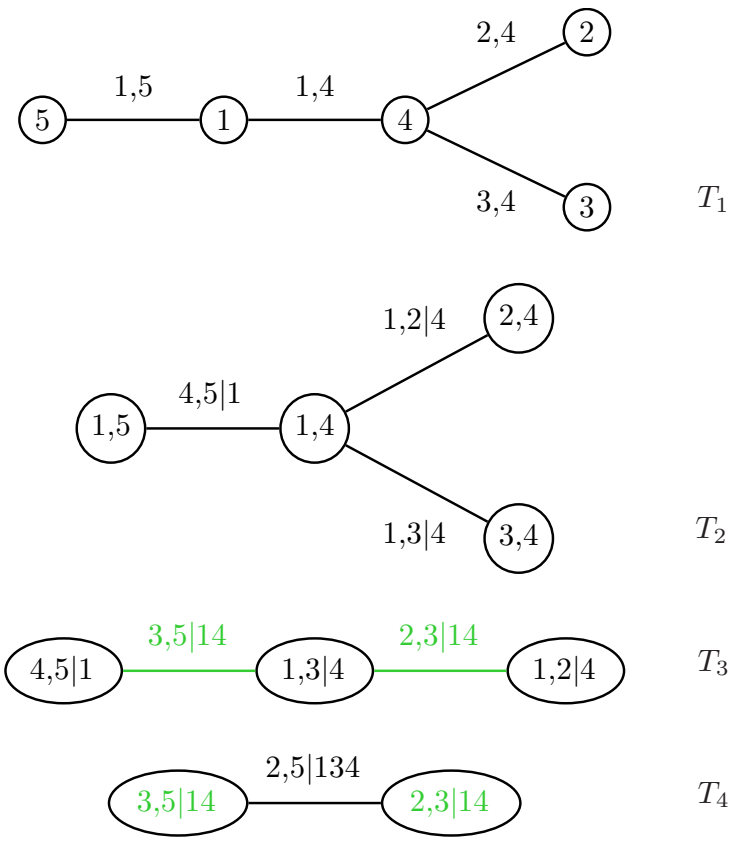
Storing R-vine copulas in matrix notation

Efficient enconding of R-vine models needed for statistical inference.

- Matrix notation by Morales-Nápoles et al. (2010) and Dißmann et al. (2013).

$$\left(\begin{array}{ccccc} 2 & & & & \\ 5 & 3 & & & \\ 3 & 5 & 4 & & \\ \hline 1 & 1 & 5 & 5 & \\ 4 & 4 & 1 & 1 & 1 \end{array}\right)$$

1. $\{2,4\},\{3,4\},\{4,1\},\{5,1\}$
2. $\{2,1|4\},\{3,1|4\},\{4,5|1\}$
3. $\{2,3|14\},\{3,5|14\}$
4. $\{2,5|314\}$



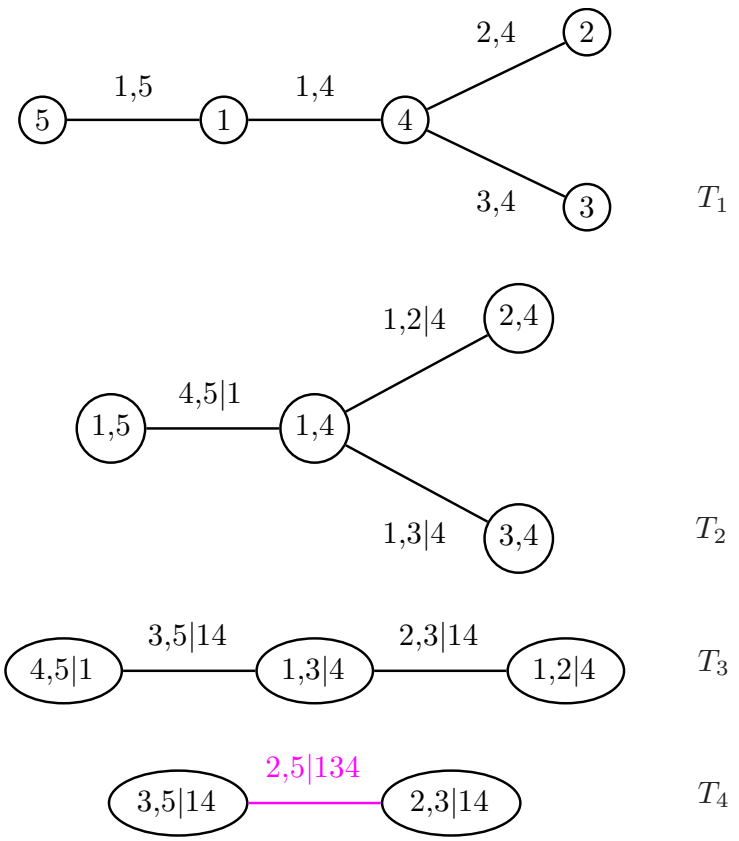
Storing R-vine copulas in matrix notation

Efficient enconding of R-vine models needed for statistical inference.

- Matrix notation by Morales-Nápoles et al. (2010) and Dißmann et al. (2013).

$$\begin{pmatrix} 2 & & & & \\ 5 & 3 & & & \\ \hline 3 & 5 & 4 & & \\ 1 & 1 & 5 & 5 & \\ 4 & 4 & 1 & 1 & 1 \end{pmatrix}$$

1. $\{2,4\},\{3,4\},\{4,1\},\{5,1\}$
2. $\{2,1|4\},\{3,1|4\},\{4,5|1\}$
3. $\{2,3|14\},\{3,5|14\}$
4. $\{2,5|314\}$



R-vine copula and parameter matrices

Copula families and parameters can be stored in associated matrices.

$$\begin{pmatrix} 2 & & & & \\ 5 & 3 & & & \\ 3 & 5 & 4 & & \\ 1 & 1 & 5 & 5 & \\ 4 & 4 & 1 & 1 & 1 \end{pmatrix} \longrightarrow \begin{pmatrix} C_{25;314} & & & \\ C_{23;14} & C_{35;14} & & \\ C_{21;4} & C_{31;4} & C_{45;1} & \\ C_{24} & C_{34} & C_{41} & C_{51} \end{pmatrix}$$

R-vine matrix objects

An RVineMatrix object contains all required matrices:

```
> Matrix = c(2,5,3,1,4,0,3,5,1,4,0,0,4,5,1,
+            0,0,0,5,1,0,0,0,0,1)
> Matrix = matrix(Matrix,5,5)
>
> family = c(0,1,3,4,4,0,0,3,4,1,0,0,0,4,1,
+            0,0,0,0,3,0,0,0,0,0)
> family = matrix(family,5,5)
>
> par = c(0,0.2,0.9,1.5,3.9,0,0,1.1,1.6,0.9,0,0,0,1.9,0.5,
+         0,0,0,0,4.8,0,0,0,0,0)
> par = matrix(par,5,5)
>
> par2 = matrix(0,5,5)
>
> RVM = RVineMatrix(Matrix=Matrix, family=family, par=par,
+                   par2=par2, names=c("V1","V2","V3","V4","V5"))
```

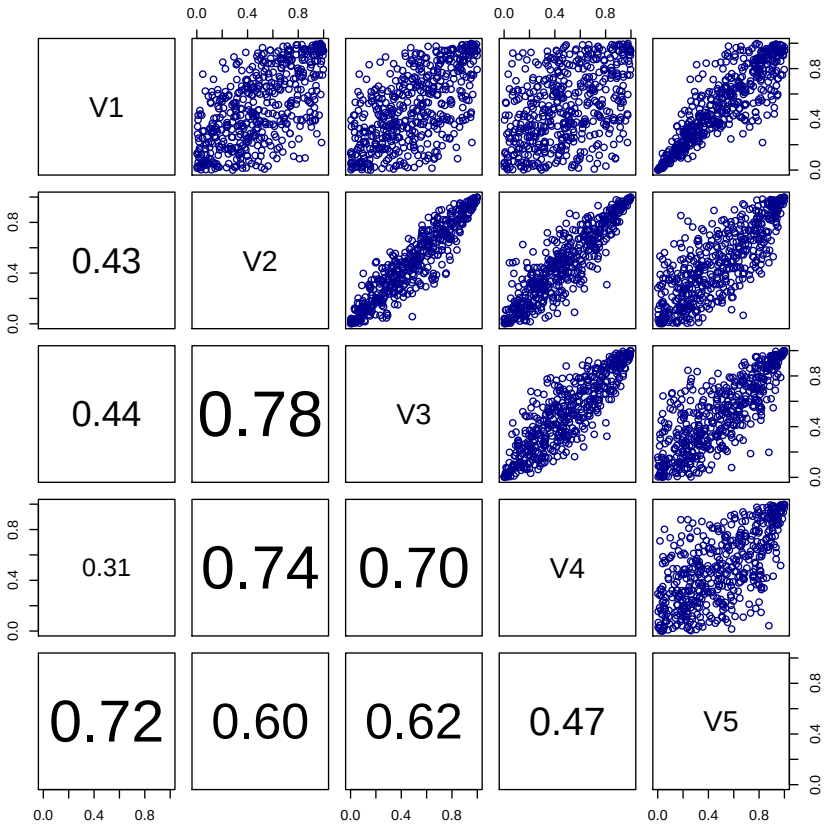
Simulation

```
> simdat = RVineSim(500, RVM)
```

```
> head(simdat)
       V1   V2   V3   V4   V5
[1,] 0.51 0.24 0.42 0.33 0.45
[2,] 0.23 0.14 0.16 0.12 0.20
[3,] 0.65 0.38 0.46 0.29 0.70
[4,] 0.43 0.18 0.08 0.08 0.26
[5,] 0.86 0.86 0.85 0.86 0.87
[6,] 0.71 0.71 0.80 0.68 0.88
```

Simulation

```
> simdat = RVineSim(500, RVM)

> head(simdat)
       V1   V2   V3   V4   V5
[1,] 0.51 0.24 0.42 0.33 0.45
[2,] 0.23 0.14 0.16 0.12 0.20
[3,] 0.65 0.38 0.46 0.29 0.70
[4,] 0.43 0.18 0.08 0.08 0.26
[5,] 0.86 0.86 0.85 0.86 0.87
[6,] 0.71 0.71 0.80 0.68 0.88
```



Example

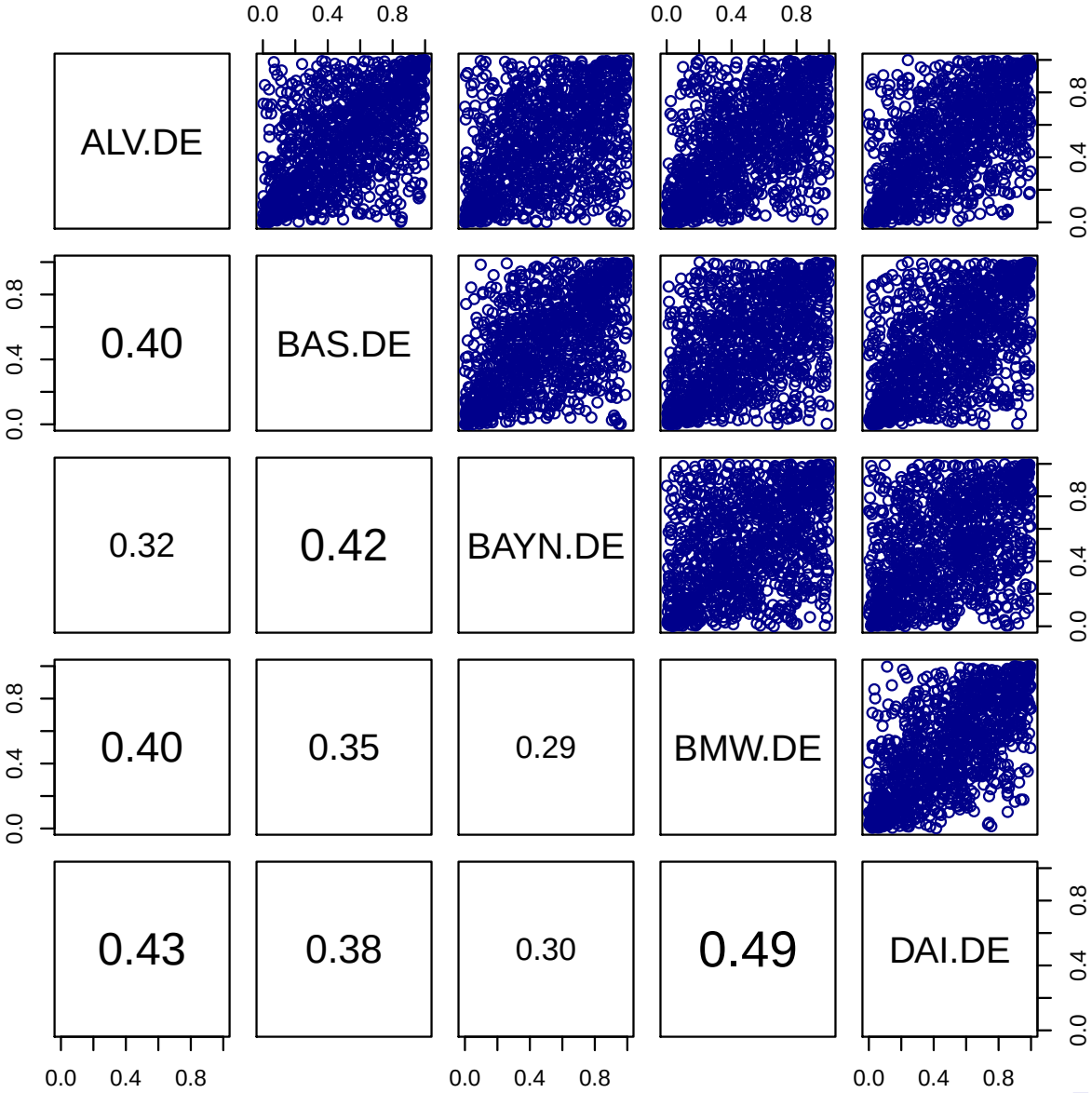
- Daily log returns of 15 major German stocks.
- Observed from January 2005 to August 2009 (1158 observations).
- Time series are filtered using GARCH(1,1) with Student's t innovations.
- Data set of standardized residuals transformed to [0,1].

Load into workspace:

```
> data(daxreturns)
```

Now: selection of trees, pair copulas and parameters in inverse order.

A first look at the data



Parameter estimation I

- Sequential estimation (based on `BiCopEst`)
  - either using bivariate inversion of Kendall's τ:

    ```
    > RVineSeqEst(data, RVM, method="itau")
    ```
  - or bivariate maximum likelihood estimation:

    ```
    > RVineSeqEst(data, RVM, method="mle")
    ```
- Very fast, since only bivariate estimation.
- Provides good starting values for joint maximum likelihood estimation.

Parameter estimation II

- Maximum likelihood estimation of all parameters jointly (log-likelihood computation: `RVineLogLik`).

```
> RVineMLE(data, RVM, start, start2, maxit,
+          grad, hessian, se)
```

- Starting values can be calculated using sequential estimation.
- Analytical gradient can be used for numerical optimization (see `RVineGrad`).
- Standard errors can be computed based on the analytical Hessian (see `RVineStdError` and `RVineHessian`).

Parameter estimation II

- Maximum likelihood estimation of all parameters jointly (log-likelihood computation: `RVineLogLik`).

```
> RVineMLE(data, RVM, start=0, start2=0, maxit,
+          grad, hessian, se)
```

- Starting values can be calculated using sequential estimation.
- Analytical gradient can be used for numerical optimization (see `RVineGrad`).
- Standard errors can be computed based on the analytical Hessian (see `RVineStdError` and `RVineHessian`).

Parameter estimation II

- Maximum likelihood estimation of all parameters jointly (log-likelihood computation: `RVineLogLik`).

```
> RVineMLE(data, RVM, start, start2, maxit,
+          grad=TRUE, hessian, se)
```

- Starting values can be calculated using sequential estimation.
- Analytical gradient can be used for numerical optimization (see `RVineGrad`).
- Standard errors can be computed based on the analytical Hessian (see `RVineStdError` and `RVineHessian`).

Parameter estimation II

- Maximum likelihood estimation of all parameters jointly (log-likelihood computation: `RVineLogLik`).

```
> RVineMLE(data, RVM, start, start2, maxit,
+          grad, hessian=TRUE, se=TRUE)
```

- Starting values can be calculated using sequential estimation.
- Analytical gradient can be used for numerical optimization (see `RVineGrad`).
- Standard errors can be computed based on the analytical Hessian (see `RVineStdError` and `RVineHessian`).

Parameter estimation III

```
> mle = RVineMLE(data=daxreturns[,1:5], RVM, start=0, start2=0,
+                grad=TRUE, hessian=TRUE, se=TRUE)
```

```
> mle$RVM$par
     [,1]  [,2] [,3] [,4] [,5]
[1,] 0.00 0.000 0.00 0.00    0
[2,] 0.14 0.000 0.00 0.00    0
[3,] 0.70 0.079 0.00 0.00    0
[4,] 1.30 1.206 1.42 0.00    0
[5,] 1.42 0.366 0.52 0.99    0
```

```
> mle$se
      [,1]  [,2]  [,3]  [,4] [,5]
[1,] 0.000 0.000 0.000 0.000    0
[2,] 0.029 0.000 0.000 0.000    0
[3,] 0.059 0.037 0.000 0.000    0
[4,] 0.031 0.028 0.036 0.000    0
[5,] 0.032 0.024 0.020 0.059    0
```

Parameter estimation III

```
> mle = RVineMLE(data=daxreturns[,1:5], RVM, start=0, start2=0,
+                grad=TRUE, hessian=TRUE, se=TRUE)

> mle$RVM$par
     [,1]  [,2] [,3] [,4] [,5]
[1,] 0.00 0.000 0.00 0.00    0
[2,] 0.14 0.000 0.00 0.00    0
[3,] 0.70 0.079 0.00 0.00    0
[4,] 1.30 1.206 1.42 0.00    0
[5,] 1.42 0.366 0.52 0.99    0

> mle$se
      [,1]  [,2]  [,3]  [,4] [,5]
[1,] 0.000 0.000 0.000 0.000    0
[2,] 0.029 0.000 0.000 0.000    0
[3,] 0.059 0.037 0.000 0.000    0
[4,] 0.031 0.028 0.036 0.000    0
[5,] 0.032 0.024 0.020 0.059    0
```

Parameter estimation III

```
> mle = RVineMLE(data=daxreturns[,1:5], RVM, start=0, start2=0,
+                grad=TRUE, hessian=TRUE, se=TRUE)

> mle$RVM$par
     [,1]  [,2] [,3] [,4] [,5]
[1,] 0.00 0.000 0.00 0.00    0
[2,] 0.14 0.000 0.00 0.00    0
[3,] 0.70 0.079 0.00 0.00    0
[4,] 1.30 1.206 1.42 0.00    0
[5,] 1.42 0.366 0.52 0.99    0

> mle$se
      [,1]  [,2]  [,3]  [,4] [,5]
[1,] 0.000 0.000 0.000 0.000    0
[2,] 0.029 0.000 0.000 0.000    0
[3,] 0.059 0.037 0.000 0.000    0
[4,] 0.031 0.028 0.036 0.000    0
[5,] 0.032 0.024 0.020 0.059    0
```

Pair copula selection

- Manually using tools for bivariate analysis (e.g., contour plots or goodness-of-fit tests: `BiCopMetaContour`, `BiCopGofTest`,...).
- Automatically using AIC or BIC from a set of copula families: `BiCopSelect` (bivariate) or `RVineCopSelect` (multivariate).

```
> cops = RVineCopSelect(data=daxreturns[,1:5], familyset=NA,
+                       Matrix=Matrix, selectioncrit="AIC",
+                       indeptest=FALSE, level=0.05)
```

```
> cops$family
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    0    0    0    0
[2,]    5    0    0    0    0
[3,]    2    5    0    0    0
[4,]    2   20    2    0    0
[5,]   17    2   14   20    0
```

Pair copula selection

- Manually using tools for bivariate analysis (e.g., contour plots or goodness-of-fit tests: `BiCopMetaContour`, `BiCopGofTest`,...).
- Automatically using AIC or BIC from a set of copula families: `BiCopSelect` (bivariate) or `RVineCopSelect` (multivariate).

```
> cops = RVineCopSelect(data=daxreturns[,1:5], familyset=NA,
+                       Matrix=Matrix, selectioncrit="AIC",
+                       indeptest=FALSE, level=0.05)

> cops$family
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    0    0    0    0
[2,]    5    0    0    0    0
[3,]    2    5    0    0    0
[4,]    2   20    2    0    0
[5,]   17    2   14   20    0
```

Pair copula selection

- Manually using tools for bivariate analysis (e.g., contour plots or goodness-of-fit tests: `BiCopMetaContour`, `BiCopGofTest`,...).
- Automatically using AIC or BIC from a set of copula families: `BiCopSelect` (bivariate) or `RVineCopSelect` (multivariate).

```
> cops = RVineCopSelect(data=daxreturns[,1:5], familyset=NA,
+                       Matrix=Matrix, selectioncrit="AIC",
+                       indeptest=FALSE, level=0.05)
```

```
> cops$family
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    0    0    0    0
[2,]    5    0    0    0    0
[3,]    2    5    0    0    0
[4,]    2   20    2    0    0
[5,]   17    2   14   20    0
```

Pair copula selection

- Manually using tools for bivariate analysis (e.g., contour plots or goodness-of-fit tests: `BiCopMetaContour`, `BiCopGofTest`,...).
- Automatically using AIC or BIC from a set of copula families: `BiCopSelect` (bivariate) or `RVineCopSelect` (multivariate).

```
> cops = RVineCopSelect(data=daxreturns[,1:5], familyset=NA,
+                       Matrix=Matrix, selectioncrit="AIC",
+                       indeptest=FALSE, level=0.05)

> cops$family
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    0    0    0    0
[2,]    5    0    0    0    0
[3,]    2    5    0    0    0
[4,]    2   20    2    0    0
[5,]   17    2   14   20    0
```

Vine tree selection I

- Sequential tree-by-tree selection according to Dißmann et al. (2013):
  1. Select maximum spanning tree (respecting the proximity condition) in terms of the absolute empirical pairwise Kendall's τ values.
  2. Select and estimate pair copulas of the tree.
  3. Compute transformed observations using *h*-functions and go back to Step 1.

- R- and C-vine copulas can be selected.
- The vine copula can be truncated to reduce the model complexity.

Vine tree selection I

- Sequential tree-by-tree selection according to Dißmann et al. (2013):
  1. Select maximum spanning tree (respecting the proximity condition) in terms of the absolute empirical pairwise Kendall's τ values.
  2. Select and estimate pair copulas of the tree.
  3. Compute transformed observations using *h*-functions and go back to Step 1.

```
> RVineStructureSelect(data, familyset, type,
+                      selectioncrit, indeptest,
+                      level, trunclevel)
```

- R- and C-vine copulas can be selected.
- The vine copula can be truncated to reduce the model complexity.

Vine tree selection I

- Sequential tree-by-tree selection according to Dißmann et al. (2013):
  1. Select maximum spanning tree (respecting the proximity condition) in terms of the absolute empirical pairwise Kendall's τ values.
  2. Select and estimate pair copulas of the tree.
  3. Compute transformed observations using *h*-functions and go back to Step 1.

```
> RVineStructureSelect(data, familyset, type="RVine",
+                      selectioncrit, indeptest,
+                      level, trunclevel)
```

- R- and C-vine copulas can be selected.
- The vine copula can be truncated to reduce the model complexity.

Vine tree selection I

- Sequential tree-by-tree selection according to Dißmann et al. (2013):
  1. Select maximum spanning tree (respecting the proximity condition) in terms of the absolute empirical pairwise Kendall's τ values.
  2. Select and estimate pair copulas of the tree.
  3. Compute transformed observations using *h*-functions and go back to Step 1.

```
> RVineStructureSelect(data, familyset, type,
+                      selectioncrit, indeptest,
+                      level, trunclevel=2)
```

- R- and C-vine copulas can be selected.
- The vine copula can be truncated to reduce the model complexity.

Vine tree selection II

```
> rvm = RVineStructureSelect(data=daxreturns)
```

```
> rvm$Matrix
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15]
 [1,]   15    0    0    0    0    0    0    0    0     0     0     0     0     0     0
 [2,]   11    4    0    0    0    0    0    0    0     0     0     0     0     0     0
 [3,]    9   11    3    0    0    0    0    0    0     0     0     0     0     0     0
 [4,]    7    9   11   11    0    0    0    0    0     0     0     0     0     0     0
 [5,]   13    7    9   12    9    0    0    0    0     0     0     0     0     0     0
 [6,]    3   13    7    8   12    7    0    0    0     0     0     0     0     0     0
 [7,]   12    3   13   10    8   12   13    0    0     0     0     0     0     0     0
 [8,]    8   12    5    5   10    8   12    5    0     0     0     0     0     0     0
 [9,]   10    8    1    2    5   10    8   12    1     0     0     0     0     0     0
[10,]    2   10   12   13    2    5   10    8   12    12     0     0     0     0     0
[11,]    1    2    8    9   13    2    5   10    8    10     8     0     0     0     0
[12,]   14    1    6    7    1   13    2    2   10     6    10    10     0     0     0
[13,]    6   14   14   14   14    1   14    1    2    14     6     6     2     0     0
[14,]    4    6   10    6    6   14    6   14   14     2    14    14     6    14     0
[15,]    5    5    2    1    7    6    1    6    6     8     2     2    14     6     6
```

Vine tree selection II

```
> rvm = RVineStructureSelect(data=daxreturns)
```

```
> rvm$Matrix
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15]
 [1,]   15    0    0    0    0    0    0    0    0     0     0     0     0     0     0
 [2,]   11    4    0    0    0    0    0    0    0     0     0     0     0     0     0
 [3,]    9   11    3    0    0    0    0    0    0     0     0     0     0     0     0
 [4,]    7    9   11   11    0    0    0    0    0     0     0     0     0     0     0
 [5,]   13    7    9   12    9    0    0    0    0     0     0     0     0     0     0
 [6,]    3   13    7    8   12    7    0    0    0     0     0     0     0     0     0
 [7,]   12    3   13   10    8   12   13    0    0     0     0     0     0     0     0
 [8,]    8   12    5    5   10    8   12    5    0     0     0     0     0     0     0
 [9,]   10    8    1    2    5   10    8   12    1     0     0     0     0     0     0
[10,]    2   10   12   13    2    5   10    8   12    12     0     0     0     0     0
[11,]    1    2    8    9   13    2    5   10    8    10     8     0     0     0     0
[12,]   14    1    6    7    1   13    2    2   10     6    10    10     0     0     0
[13,]    6   14   14   14   14    1   14    1    2    14     6     6     2     0     0
[14,]    4    6   10    6    6   14    6   14   14     2    14    14     6    14     0
[15,]    5    5    2    1    7    6    1    6    6     8     2     2    14     6     6
```

Illustrating R-vine copula models

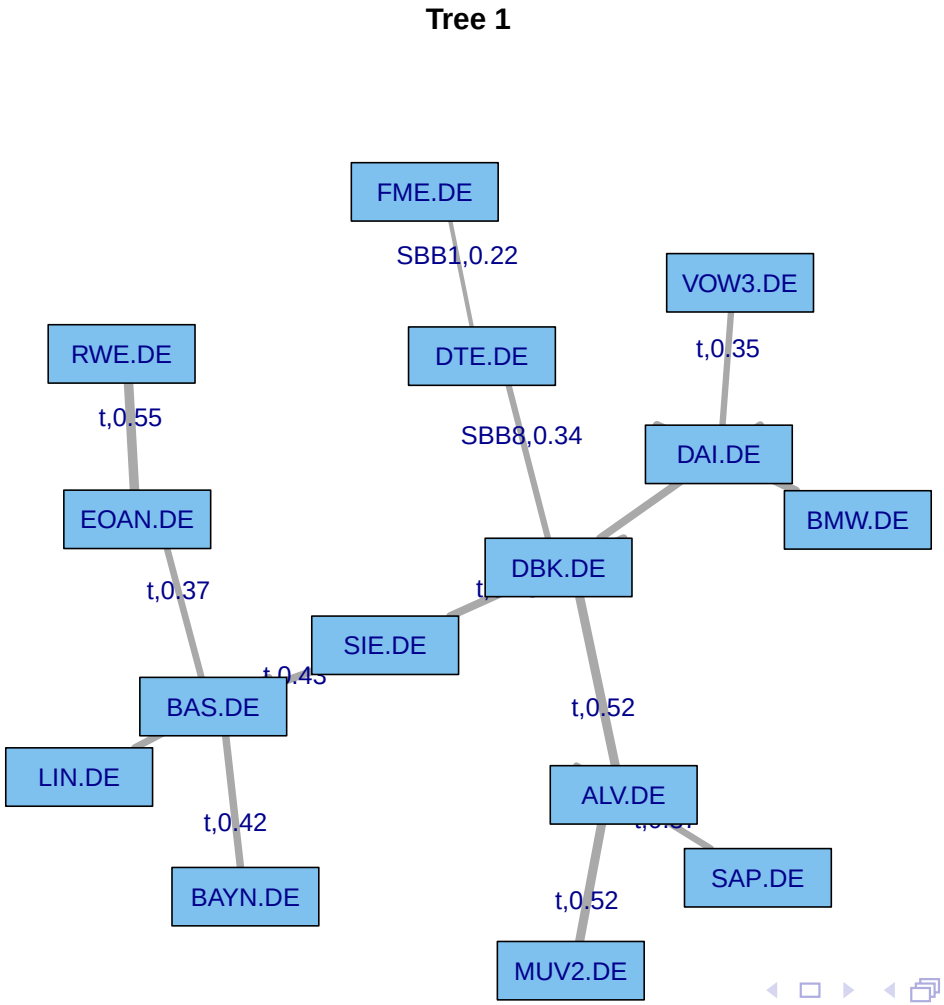
Selected R-vine trees:

```
> RVineTreePlot(data=NULL, RVM=rvm, tree=1,
+               edge.labels=c("family","theotau"))
```

Illustrating R-vine copula models

Selected R-vine trees:

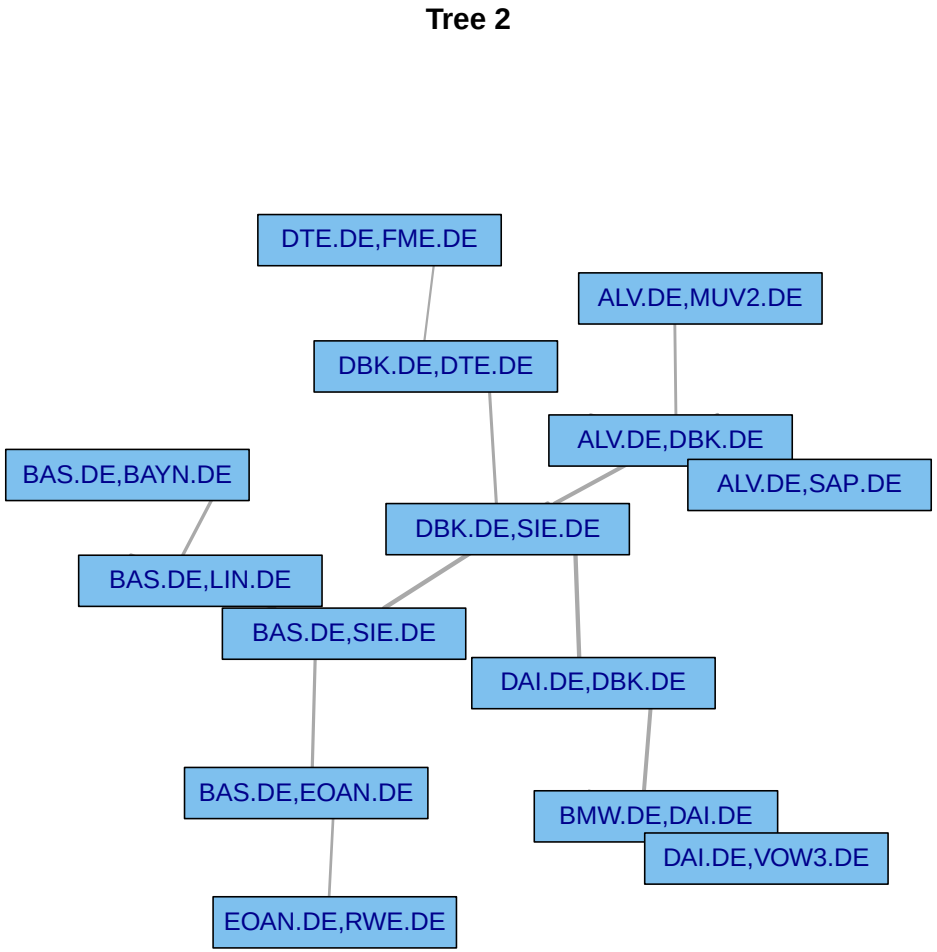
```
> RVineTreePlot(data=NULL, RVM=rvm, tree=1,
+                 edge.labels=c("family","theotau"))
```



Illustrating R-vine copula models

Selected R-vine trees:

```
> RVineTreePlot(data=NULL, RVM=rvm, tree=2,
+                edge.labels=FALSE)
```



Discrimination among vine copula models I

- AIC and BIC: `RVineAIC` and `RVineBIC`.
- Tests by Vuong (1989) and by Clarke (2007) for non-nested comparisons of two R-vine models RVM1 and RVM2:

```
> RVineVuongTest(data, RVM1, RVM2)
> RVineClarkeTest(data, RVM1, RVM2)
```

Discrimination among vine copula models II

- Select a C-vine copula for comparison:

```
> cvm = RVineStructureSelect(daxreturns, type="CVine")
```

- Compare the models in terms of AIC values and the Vuong test:

```
> c(RVineAIC(daxreturns,rvm)$AIC, RVineAIC(daxreturns,cvm)$AIC)
[1] -9808.44 -9804.42

> RVineVuongTest(daxreturns, rvm, cvm)
$statistic
[1] 0.27

$statistic.Akaike
[1] 0.068

$statistic.Schwarz
[1] -0.44
...
```

- The models are essentially indistinguishable.

Discrimination among vine copula models II

- Select a C-vine copula for comparison:

```
> cvm = RVineStructureSelect(daxreturns, type="CVine")
```

- Compare the models in terms of AIC values and the Vuong test:

```
> c(RVineAIC(daxreturns,rvm)$AIC, RVineAIC(daxreturns,cvm)$AIC)
[1] -9808.44 -9804.42

> RVineVuongTest(daxreturns, rvm, cvm)
$statistic
[1] 0.27

$statistic.Akaike
[1] 0.068

$statistic.Schwarz
[1] -0.44
...
```

- The models are essentially indistinguishable.

Outlook

Current projects and plans:

- Move code from `C` to `C++`.
- Parallelize numerical maximum likelihood estimation.
- Implement asymmetric copulas (Tawn).
- Bayesian estimation and model selection.

Thank you to my package coauthors: Ulf Schepsmeier & Jakob Stöber

Outlook

Current projects and plans:

- Move code from `C` to `C++`.
- Parallelize numerical maximum likelihood estimation.
- Implement asymmetric copulas (Tawn).
- Bayesian estimation and model selection.

Thank you to my package coauthors: Ulf Schepsmeier & Jakob Stöber

Bibliography

Brechmann, E. C. and C. Czado (2013).
Risk management with high-dimensional vine copulas: An analysis of the Euro Stoxx 50.
Statistics & Risk Modeling, forthcoming.

Brechmann, E. C. and U. Schepsmeier (2013).
Modeling dependence with C- and D-vine copulas: The R-package CDVine.
Journal of Statistical Software 52(3), 1–27.

Clarke, K. A. (2007).
A simple distribution-free test for nonnested model selection.
Political Analysis 15(3), 347–363.

Dißmann, J., E. C. Brechmann, C. Czado, and D. Kurowicka (2013).
Selecting and estimating regular vine copulae and application to financial returns.
Computational Statistics & Data Analysis 59(1), 52–69.

Morales-Nápoles, O., R. M. Cooke, and D. Kurowicka (2010).
About the number of vines and regular vines on n nodes.
Working paper.

Vuong, Q. H. (1989).
Ratio tests for model selection and non-nested hypotheses.
Econometrica 57(2), 307–333.

Thank you very much for your attention!

Visit: http://cran.r-project.org/web/packages/VineCopula/