Journal of Data Science 20 (1), 95–110
January 2022

DOI: 10.6339/22-JDS1034
Computing in Data Science

Computational Challenges of t and Related Copulas

Erik Hintz[1,*], Marius Hofert[1], and Christiane Lemieux[1]

[1]*Department of Statistics and Actuarial Science, University of Waterloo, 200 University Avenue West, Waterloo, ON, N2L 3G1, Canada*

Abstract

The present paper addresses computational and numerical challenges when working with t copulas and their more complicated extensions, the grouped t and skew t copulas. We demonstrate how the R package `nvmix` can be used to work with these copulas. In particular, we discuss (quasi-)random sampling and fitting. We highlight the difficulties arising from using more complicated models, such as the lack of availability of a joint density function or the lack of an analytical form of the marginal quantile functions, and give possible solutions along with future research ideas.

Keywords *copulas; density; distribution function; estimation; sampling*

1 Introduction

Copulas, such as the t copula, are widely used for modelling dependence in risk management and other disciplines. In contrast to the Gaussian copula, the t copula allows for modeling tail dependence. Denote by t_ν the distribution function of a univariate t distribution with ν degrees-of-freedom. The t copula is the implicit copula of a multivariate t distributed random vector, $\boldsymbol{X} \sim t_d(\nu, P)$ with stochastic representation

$$\boldsymbol{X} \stackrel{\mathrm{d}}{=} \sqrt{W} A \boldsymbol{Z},$$

where $W \sim \mathrm{IG}(\nu/2, \nu/2)$ is independent of $\boldsymbol{Z} \sim \mathrm{N}_d(\boldsymbol{0}, I_d)$ and $AA^\top = P$ for a correlation matrix P. The random vector $\boldsymbol{U} = (t_\nu(X_1), \dots, t_\nu(X_d))$ then follows the *t copula*, in short $\boldsymbol{U} \sim C^t_{\nu,P}$. If $P = I_d$, all lower-dimensional margins of $\boldsymbol{U}$ follow the same distribution. To overcome this limitation, Daul et al. (2003) proposed grouped t copulas, where different groups of margins are allowed to have different degrees of freedom. More precisely, a random vector $\boldsymbol{X}'$ follows a *grouped t distribution*, if it has stochastic representation

$$\boldsymbol{X}' = \mathrm{diag}(\sqrt{\boldsymbol{W}}) A \boldsymbol{Z},$$

where $(W_1, \dots, W_d)$ is comonotone with $W_j \sim t_{\nu_j}$; components with the same degrees-of-freedom parameter are considered to be in the same group, hence the name. The random vector $\boldsymbol{U}' = (t_{\nu_1}(X_1), \dots, t_{\nu_d}(X_d))$ then follows the *grouped t copula*, in short $\boldsymbol{U}' \sim C^{gt}_{\boldsymbol{\nu},P}$, where $\boldsymbol{\nu} = (\nu_1, \dots, \nu_d)$. Moving from a t to a grouped t copula provides more flexibility, but also poses significant computational challenges, since not even the density of a grouped t distribution is available analytically.

*Corresponding author. Email: erik.hintz@uwaterloo.ca.

Received January 5, 2022; Accepted January 6, 2022

A possible limitation of t and grouped t copulas is their radial symmetry, which means that both joint tails are equal; this falsely implies, for example, that joint large losses are as likely as joint large gains when these copulas are used for modeling financial (log-)returns $\boldsymbol{X}$. A remedy is the skew t copula; see, for instance, (McNeil et al., 2015, Chapter 7). A d-dimensional random vector $\boldsymbol{X}''$ has a d-dimensional *skew t distribution*, denoted by $\boldsymbol{X}'' \sim st_d(\nu, \boldsymbol{\mu}, P, \boldsymbol{\gamma})$, if it satisfies the stochastic representation

$$\boldsymbol{X}'' \stackrel{\mathrm{d}}{=} \boldsymbol{\mu} + W\boldsymbol{\gamma} + \sqrt{W}A\boldsymbol{Z}, \tag{1}$$

where $\boldsymbol{\gamma} \in \mathbb{R}^d$ denotes the *skewness parameter vector* and $\boldsymbol{Z} \sim \mathrm{N}_d(\boldsymbol{0}, I_d)$ is independent of $W \sim \mathrm{IG}(\nu/2, \nu/2)$; this is the multivariate version of the skew t distribution proposed by Barndorff-Nielsen (1977). We remark that different definitions of the skew t distribution can be found in the literature; see for instance Azzalini (2013) for an alternative model referred to as a skew t distribution. Unlike the grouped t, the skew t distribution does have a closed form expression for its density, see (McNeil et al., 2015, p. 191). If, for $\gamma \in \mathbb{R}$, $st_{\nu,\gamma}$ denotes the distribution function of a univariate skew t distribution, the random vector $\boldsymbol{U}'' = (st_{\nu,\gamma_1}(X_1''), \dots, st_{\nu,\gamma_d}(X_d''))$ follows the *skew t copula*, $\boldsymbol{U}'' \sim \boldsymbol{C}^{st}_{\nu,\boldsymbol{\gamma},P}$. The major computational challenge when moving from a t copula to a skew t copula is the fact that a fast approximation of the distribution function $st_{\nu,\gamma}$ is not available anymore and must be estimated. Even worse, the density of a skew t copula involves the quantile functions st^{-1}_{ν,γ_j}, which must be computed numerically.

The present paper highlights these numerical challenges and demonstrates how the R package *nvmix* can be used to work with t, grouped t and skew t copulas. Besides showing how *nvmix* can be used to generate pseudo- and quasi-random variates from these copulas and what numerical and computational challenges arise by working with these more complicated models, we demonstrate and compare various fitting procedures for these copulas; see Hofert et al. (2022) and Hintz et al. (2022).

2 Sampling the t, Grouped t and Skew t Copula

Our R package *nvmix* provides functions to sample from the t, the grouped t and the skew t copula based on their stochastic representations. We focus on the more interesting cases of the grouped t and skew t copula. In order to generate the copula samples, we first sample from the grouped t or skew t distribution and then apply the corresponding probability integral transformation. In the case of the t and grouped t distribution, the margins are univariate t, so we use the R function `pt()`. The skew t case is more complicated because it involves the approximation of the distribution function of the univariate skew t distribution, a point we address in the second subsection.

2.1 Sampling the Grouped t Copula with Pseudo- and Quasi-Random Numbers

Grouped t copulas can be easily sampled from using the function `rgStudentcopula()`:

```
library(nvmix) # load the package
str(rgStudentcopula)
```

```
## function (n, groupings = 1:d, df, scale = diag(2), factor = NULL,
##           method = c("PRNG", "sobol", "ghalton"), skip = 0)
```

The argument `groupings` is a vector of length d so that `groupings[j]` gives the group of the jth component, while the vector `df` has length equal to the number of groups, that is, `df[k]` contains the degrees-of-freedom for the kth group. For efficiency, `rgStudentcopula()` also allows to provide the argument `factor` which is a lower triangular matrix A so that $AA^\top = P$; if not supplied, it is computed via `chol()`. The argument `method` specifies the type of uniform numbers used for sampling from the copula; if `method = "PRNG"` we obtain classical pseudo-random samples, while `method = "sobol"` and `method = "ghalton"` use the Sobol' or generalized Halton sequence of the R package *qrng* of Hofert and Lemieux (2019).

Consider, for instance, a 4-dimensional grouped t copula, where (marginally) $(U_1, U_2) \sim C^t_{\nu_1,0.7}$ and $(U_3, U_4) \sim C^t_{\nu_2,0.7}$, with $\nu_1 = 0.5$ and $\nu_2 = 25$ chosen far apart to make the effect of different groups visible. This model can be easily sampled from using the function `rgStudentcopula()`:

```
library(copula) # for pairs2()
set.seed(1) # set a seed for reproducibility
d <- 4 # dimension
groupings <- c(1, 1, 2, 2) # (U1, U2) and (U3, U4) are each in the same group
df <- c(0.5, 25) # degrees-of-freedom for groups 1 and 2
rho <- 0.7 # off-diagonal entry of P
P <- matrix(rho, ncol = d, nrow = d)
diag(P) <- 1 # P is now a correlation matrix
n <- 1e3 # sample size
U <- rgStudentcopula(n, groupings = groupings, df = df, scale = P)
pairs2(U, pch = ".")
```

As an application, consider the problem of evaluating this copula at $\boldsymbol{u} = (u, \dots, u)$ for some $u \in (0, 1)$, i.e., we estimate $C^{gt}_{\boldsymbol{\nu},P}(u, \dots, u) = \mathbb{P}(U_1 \leqslant u, \dots, U_d \leqslant u)$.

We first use pseudo-random numbers and define a function returning the Monte Carlo estimate along with an estimate of the absolute error (with confidence level at least 99.99%).

```
MC_est <- function(n, u) {
   le.u <- apply(rgStudentcopula(n, groupings = groupings, df = df,
                                 scale = P) <= u, 1, all)
   c(est = mean(le.u), abs.err = 3.5 * sqrt(var(le.u)/n))
}
```

Next, we replace pseudo-random numbers by a randomized Sobol' sequence, resulting in a randomized quasi-Monte Carlo (RQMC) method. Because quasi-random numbers are correlated, we use $B = 15$ independent repetitions to compute the estimate and variance.

```
RQMC_est <- function(n, u, B = 15) {
   le.u <- sapply(1:B, function(i) mean(apply(rgStudentcopula(n, df = df,
         groupings = groupings, scale = P, method = "sobol") <= u, 1, all)))
   c(est = mean(le.u), abs.err = 3.5 * sqrt(var(le.u)/B))
}
```

For a fair comparison, we must match the total number of function evaluations, as is done in the following example.

```
u <- 0.5 # evaluation point
n <- 2^10 # sample size
B <- 15 # number of replications
set.seed(2) # for reproducibility
t.MC   <- system.time(MC.out <- MC_est(B * n, u = u))
```

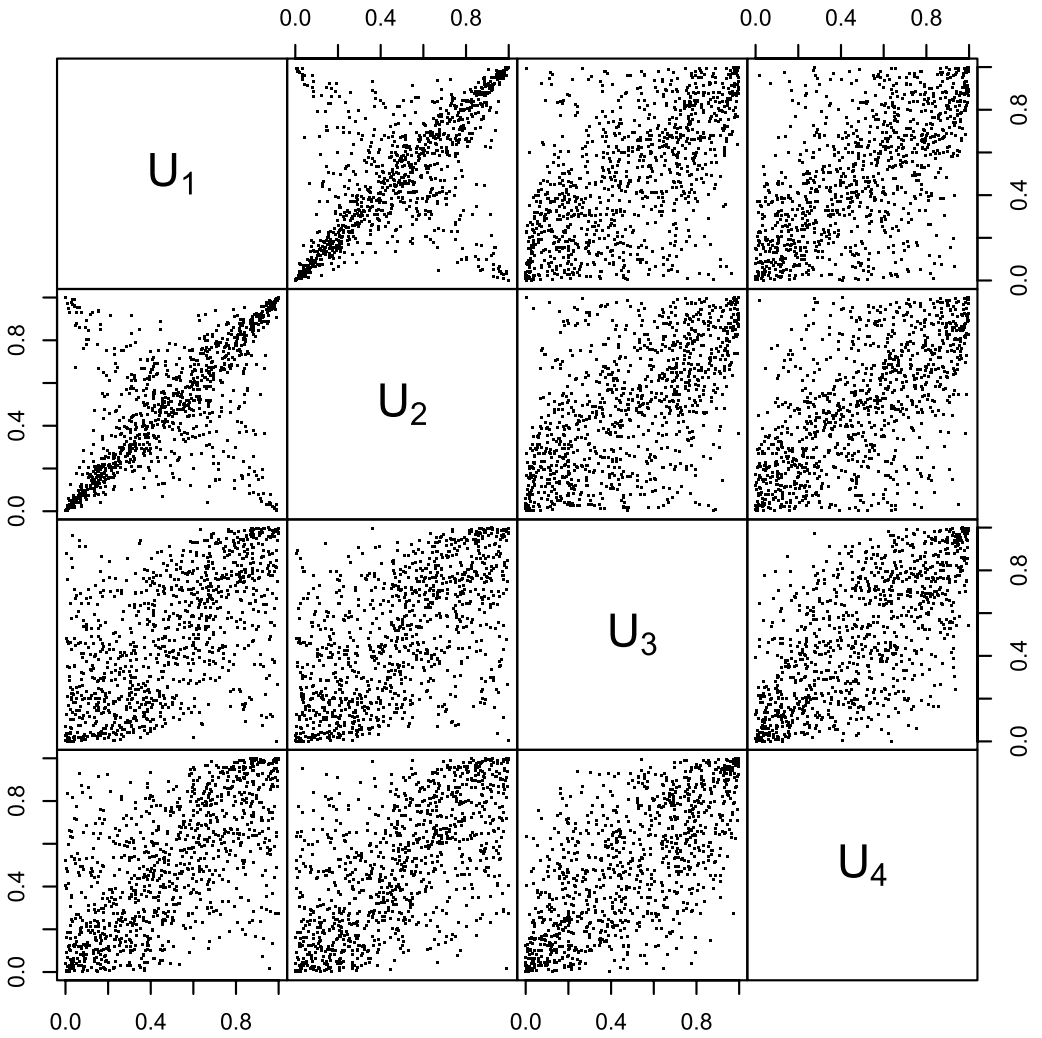

Figure 1: Pairs plot of a grouped t copula sample.

```
t.RQMC <- system.time(RQMC.out <- RQMC_est(n, u = u, B = B))
matrix(c(MC.out, RQMC.out), ncol = 2, dimnames = list(c("est", "abs.err"),
                                                      c("MC", "RQMC")))
```

```
##                MC        RQMC
## est     0.26861979 0.269075521
## abs.err 0.01251778 0.006283564
```

By taking the quotient of the squared absolute errors, we see that using a Sobol' sequence reduces the variance by a factor of 4; we chose to display absolute errors here to be consistent with the example below. We see that with the function `rgStudentcopula()`, replacing pseudo- by quasi-random numbers can be easily accomplished.

The same probability can be estimated efficiently using a reformulation of the problem (reducing the dimension of the problem by 1), variable reordering (reducing the variance) and RQMC; see Hintz et al. (2020) and Hintz et al. (2021) for details. This is implemented in the function `pgStudentcopula()`, which automatically samples until the default absolute tolerance

of 10^{-3} or a user-provided tolerance is reached; the latter can be specified in the `control` argument, see `?get_set_param()`. The full code is available in the source code of the package *nvmix*. This is particularly convenient, as the user does not need to decide in advance how large the sample size needs to be.

```
t.pgstcop <- system.time(p <- pgStudentcopula(rep(u, d), groupings = groupings,
                                              df = df, scale = P))
print(p)
```

```
## [1] 0.2706164
## attr(,"abs. error")
## [1] 6.331122e-05
## attr(,"rel. error")
## [1] 0.0002339519
## attr(,"numiter")
## [1] 1
```

Not only is `pgStudentcopula()` more accurate, it is also faster:

```
c("MC" = t.MC[3], "RQMC" = t.RQMC[3], "pgStudentcop" = t.pgstcop[3])
```

```
##           MC.elapsed         RQMC.elapsed pgStudentcop.elapsed
##                0.064                0.075                0.016
```

Next, rather than estimating the copula at a single value of $\boldsymbol{u}$, we construct a plot of the copula diagonal for several $\boldsymbol{u}$ (chosen small to highlight the challenges when estimating small probabilities). We omit code for the more involved plots from now on.

```
u <- seq(from = 0.003, to = 0.2, length.out = 27)
MC.out  <- sapply(u, function(u.) MC_est(B * n, u = u.)[1])
RQMC.out <- sapply(u, function(u.) RQMC_est(n,  u = u., B = B)[1])
pgstcop.out <- pgStudentcopula(matrix(u, ncol = d, nrow = 27),
                               groupings = groupings, df = df, scale = P)
```

It is evident from Figure 2 that `pgStudentcopula()` yields the curve, followed by `RQMC_est()` and then `MC_est()`.

2.2 Sampling from the Skew-*t* Copula and the Effect of Estimated Margins

Moving from a (grouped) t copula to a skew t copula means moving from a model where the margins are readily available (via `pt()` and `qt()`) to a model where all marginal distribution and quantile functions need to be estimated. Recall that if $\boldsymbol{X} = (X_1, \dots, X_d)$ follows the stochastic representation (1), then $\boldsymbol{U} = (st_{\nu,\gamma_1}(X_1), \dots, st_{\nu,\gamma_d}(X_d))$ is distributed according to the skew-t copula. Computation of $st_{\nu,\gamma_j}()$ requires numerical integration; Yoshiba (2018a) suggest using the R function `integrate()`, as is done in the R package *ghyp*, see Weibel et al. (2020). Note that sampling n d-dimensional copula realizations requires to approximate nd integrals.

A popular method to avoid using the correct marginal distribution functions is to replace them by their empirical distribution functions. That is, if $\boldsymbol{X}_1, \dots, \boldsymbol{X}_n \overset{\text{ind.}}{\sim} st_d(\nu, \gamma, P)$, we compute

$$\hat{F}_j(x) = \frac{1}{n+1} \sum_{i=1}^{n} \mathbb{1}_{\{X_{ij} \leqslant x\}}$$

for $j = 1, \dots, d$, and compute a *pseudo-copula sample* via $(\hat{F}_1(X_{i1}), \dots, \hat{F}_d(X_{id}))$ for $i = 1, \dots, n$.

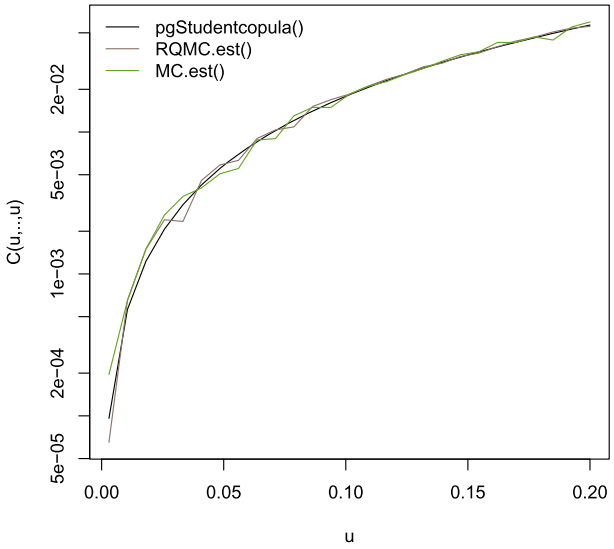

Figure 2: Estimating the copula diagonal of a grouped t copula via MC, RQMC and via `pgStudentcopula()`.

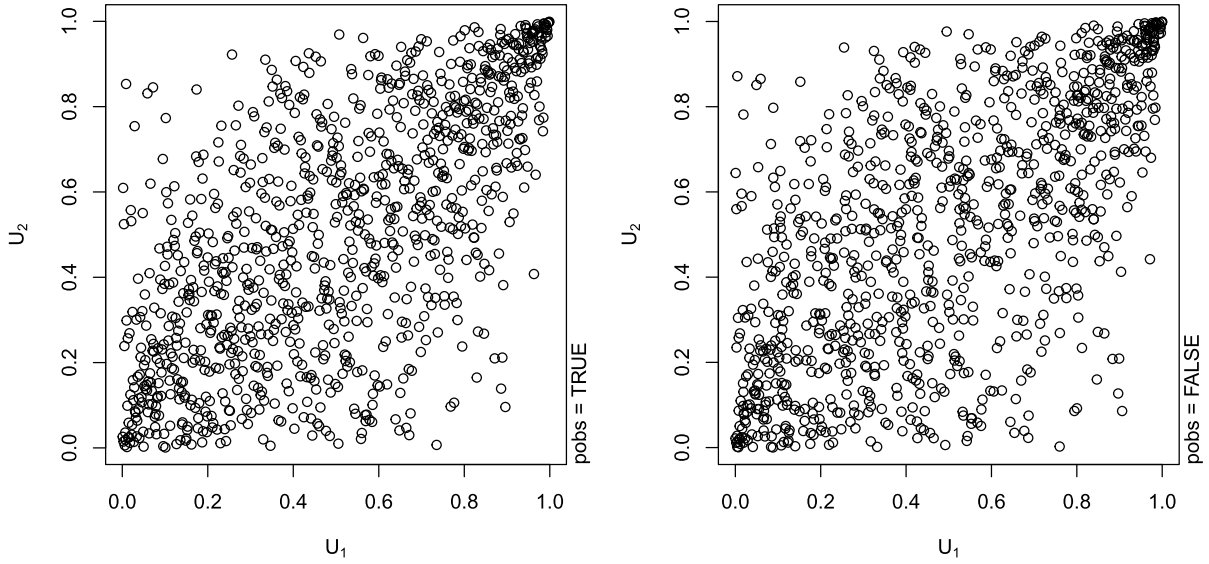

Figure 3: Estimating the copula diagonal of a grouped t copula via MC, RQMC and via `pgStudentcopula()`.

The function `rskewtcopula()` of the R package *nvmix* provides both of these methods, which can be chosen by setting the argument `pseudo = TRUE` or `pseudo = FALSE`, respectively. We use both methods in the following code and compare their run times. Figure 3 shows the first 2 margins of the aforementioned samples.

```
n <- 1000; d <- 10; rho <- 0.5; gamma <- rep(1, d); df <- 7
scale <- matrix(rho, ncol = d, nrow = d); diag(scale) <- 1
set.seed(1)
```

```
system.time(samplecop.pobs <- rskewtcopula(n, scale = scale, gamma = gamma,
                                           df = df, pseudo = TRUE))
```

```
##    user  system elapsed
##   0.003   0.000   0.004
```

```
set.seed(1)
system.time(samplecop.pskewt <- rskewtcopula(n, scale = scale, gamma = gamma,
                                             df = df, pseudo = FALSE))
```

```
##    user  system elapsed
##   2.608   0.042   2.871
```

Using `pseudo = TRUE` typically works well for large n, as $\hat{F}_j(x)$ converges to $F_j(x)$ almost surely for $n \to \infty$. For small sample sizes, however, inference based on pseudo-observations can lead to flawed results. We illustrate this by estimating the rather simple probability $\mathbb{P}(U_1 > 0.9, \dots, U_d > 0.9)$ for $d = 10$.

```
## Compute estimates and record run-time when estimating the probability
## based on pseudo = TRUE and pseudo = FALSE
ns <- c(50, 100, 250, 500, 750, 1000, 2000, 3000, 4000, 5000) # sample sizes
res <- array(NA, dim = c(length(ns), 2, 2),
             dimnames = list(n = ns, pobs = c(TRUE, FALSE), c("est", "CPU")))
for(i in seq_along(ns)) {
   set.seed(12) # same seed for all sample sizes
   res[i,1,2] <- system.time(res[i,1,1] <- mean(apply(
      rskewtcopula(ns[i], scale = scale, gamma = gamma, df = df, pseudo = TRUE),
      1, function(i) all(i > 0.9))))[3]
   set.seed(12) # same random numbers for 'pseudo = TRUE' and 'pseudo = FALSE'
   res[i,2,2] <- system.time(res[i,2,1] <- mean(apply(
      rskewtcopula(ns[i], scale = scale, gamma = gamma, df = df, pseudo = FALSE),
      1, function(i) all(i > 0.9))))[3]
}
```

Figure 4 shows the estimates obtained (left) and the run time needed (right), both with a logarithmic y-axis. Clearly, a relatively large sample size is needed so that both methods give approximately the same result. In simulation studies like the one below, this is not restrictive as we can very quickly sample a large number of pseudo-observations. Furthermore, even when the sample size n is small, one can sample $n' > n$ observations, compute an estimate of the empirical distribution function and return n of them. Nevertheless, this example highlights that care must be taken when working with a given dataset with relatively small size.

3 Estimating the t, Grouped t and Skew t Copula

In this section, we demonstrate how the R package *nvmix* can be used to fit the t, grouped t and skew t copula to data, hereby highlighting the computational challenges arising from moving from the classical t copula to a more complicated model.

3.1 Parameter Estimation for the t Copula

Fitting the t copula to data by means of maximum likelihood (ML) estimation requires optimization of the $(d(d-1)/2+1)$-dimensional copula log-likelihood function. This optimization

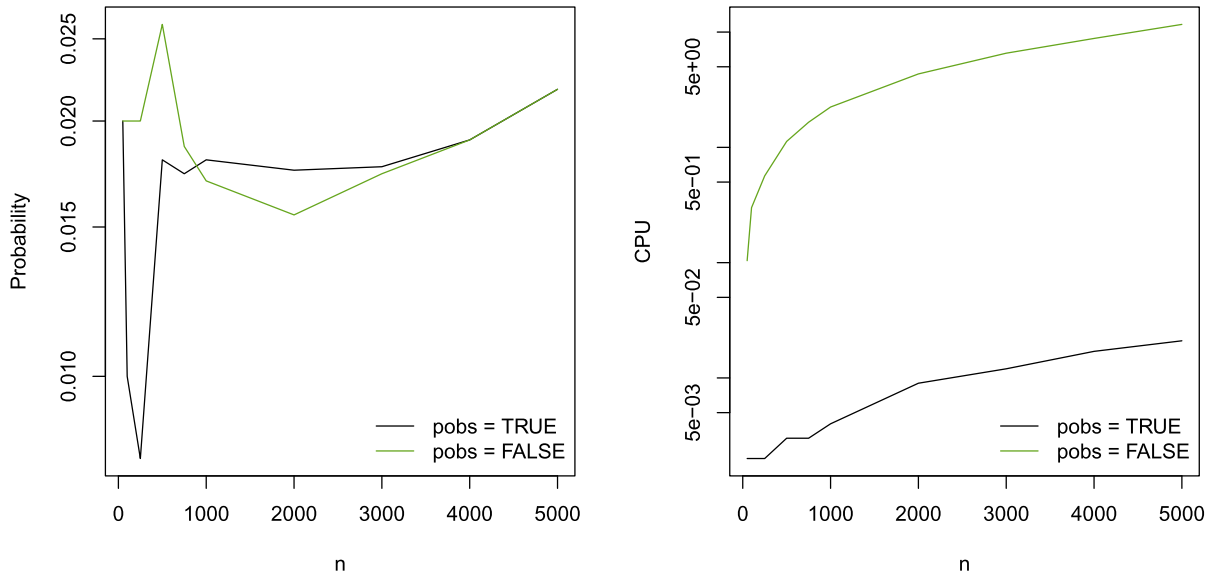

Figure 4: Estimated probabilities (left) and run times needed (right) when estimating the survival copula based on (pseudo-)observations.

problem has non-linear constraints (as positive-semidefiniteness of P must be ensured), making this problem particularly hard. The R package *copula* (see Yan (2007), Kojadinovic and Yan (2010), Hofert and Mächler (2011) and Hofert et al. (2020)) currently uses such ML procedure via the function `fitCopula()`, which searches for the optimum in the space of all square matrices and rejects those that are not positive definite.

Yoshiba (2018a) instead represent the Cholesky factor L of P (a lower triangular matrix L such that $LL^\top = P$) using angles $\theta_{ij} \in [0, \pi)$ for $j = 1, \dots, i-2$ and $\theta_{i,i-1} \in [0, 2\pi)$ for $i = 2, \dots, d$; see (Yoshiba, 2018a, Equation 12) for details. The benefit of this transformation is that one can proceed by maximizing the copula log-likelihood without imposing non-linear constraints (or rejecting non-positive definite matrices). We refer to this method as "Full-MLE". Note that the optimization problem still has $\mathcal{O}(d^2)$ parameters, making this procedure ill-suited for higher dimensions d. In order to reduce the dimensionality of the optimization problem, note that if the degrees-of-freedom ν is known, estimating P is equivalent to estimating the scale matrix of a multivariate t distribution, which can be done efficiently using the EM algorithm; see Dempster et al. (1977), Protassov (2004), (McNeil et al., 2015, Chapter 15.1). This motivates maximizing the profile log-likelihood, given by

$$\log L^*(\nu; \boldsymbol{U}_1, \dots, \boldsymbol{U}_n) = \log L(\nu, \hat{P}(\nu); \boldsymbol{U}_1, \dots, \boldsymbol{U}_n),$$

where $\hat{P}(\nu)$ is the ML estimator of P based on the samples $\boldsymbol{X}_i$, $i = 1, \dots, n$, with $\boldsymbol{X}_i = (t_\nu^{-1}(U_{i1}), \dots, t_\nu^{-1}(U_{id}))$. Note that $\log L^*$ is only a function of ν and thus univariate. We refer to this method as "EM-MLE".

Another popular estimation method for t copulas explained in (Mashal and Zeevi, 2002, Appendix C) (see also Demarta and McNeil (2005)) is to empirically estimate all pairwise Kendall's tau ρ_{ij}^τ, $1 \leqslant i < j \leqslant d$, and then map these estimates to a correlation matrix P using $P_{ij} = \sin(\pi \rho_{ij}^\tau / 2)$. With an estimate of P at hand, the degrees-of-freedom ν can be estimated by optimizing a univariate log-likelihood function. We refer to this method as "Moment-MLE".

The function `fitStudentcopula()` from the R package *nvmix* provides all previously mentioned estimation methods:

```
str(fitStudentcopula)
```

```
## function (u, fit.method = c("Moment-MLE", "EM-MLE", "Full-MLE"), df.init = NULL,
##           df.bounds = c(0.1, 30), control = list(), verbose = TRUE)
```

One can supply initial estimates for the degrees-of-freedom ν and bounds for it via the arguments `df.init` and `df.bounds`. This function returns an object of class `fitgStudentcopula`, for which the `print()` method can be used to output all necessary information, as demonstrated in the following example:

```
n <- 500; d <- 7; rho <- 0.5; df <- 3.5
scale <- matrix(rho, ncol = d, nrow = d); diag(scale) <- 1
set.seed(123)
sample <- rStudentcopula(n, df = df, scale = scale)
fitStudentcopula(sample)
```

```
## Call: fitStudentcopula(u = sample)
## Input data: 500 7-dimensional observations.
## Fitting a t copula with unknown scale matrix and method  Moment-MLE
## Log-likelihood at reported parameter estimates: 967.763800
## Estimated degrees-of-freedom
## [1] 3.512113
## Estimated 'scale' matrix:
##        [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]
## [1,] 1.0000 0.4988 0.4328 0.4663 0.5029 0.4534 0.4590
## [2,] 0.4988 1.0000 0.4875 0.5192 0.5397 0.5529 0.5021
## [3,] 0.4328 0.4875 1.0000 0.4720 0.5355 0.5499 0.5225
## [4,] 0.4663 0.5192 0.4720 1.0000 0.5560 0.4958 0.4979
## [5,] 0.5029 0.5397 0.5355 0.5560 1.0000 0.5763 0.5000
## [6,] 0.4534 0.5529 0.5499 0.4958 0.5763 1.0000 0.5027
## [7,] 0.4590 0.5021 0.5225 0.4979 0.5000 0.5027 1.0000
```

By default, the function `fitStudentcopula()` uses the "Moment-MLE" method, which is the fastest and typically most reliable. In the following, we perform a simulation study to investigate the performance of the three methods. For comparison, we also include the `fitCopula()` function of the R package *copula* as a fourth method.

```
set.seed(1)
methods <- c("Moment-MLE", "EM-MLE", "Full-MLE", "Full-MLE ('copula')")
reps <- 50 # number of replications for each method
res <- array(, dim = c(length(methods), reps, 4),
             dimnames = list(method = methods, rep = 1:reps,
                             c("loglik", "cpu", "df", "rho12")))
for(i in 1:reps) {
   U <- rStudentcopula(n, df = df, scale = scale) # sample
   t.mm <- system.time(fit.mm <- fitStudentcopula(U, fit.method = "Moment-MLE"))[1]
   t.em <- system.time(fit.em <- fitStudentcopula(U, fit.method = "EM-MLE"))[1]
   t.fl <- system.time(fit.fl <- fitStudentcopula(U, fit.method = "Full-MLE"))[1]
   t.fl.cop <- system.time(fit.fl.cop <- fitCopula(
      tCopula(dim = d, dispstr = "un"), U, method = "ml", estimate.variance = FALSE,
      start = c(fit.mm$scale[upper.tri(fit.mm$scale)], 5)))[1]
```

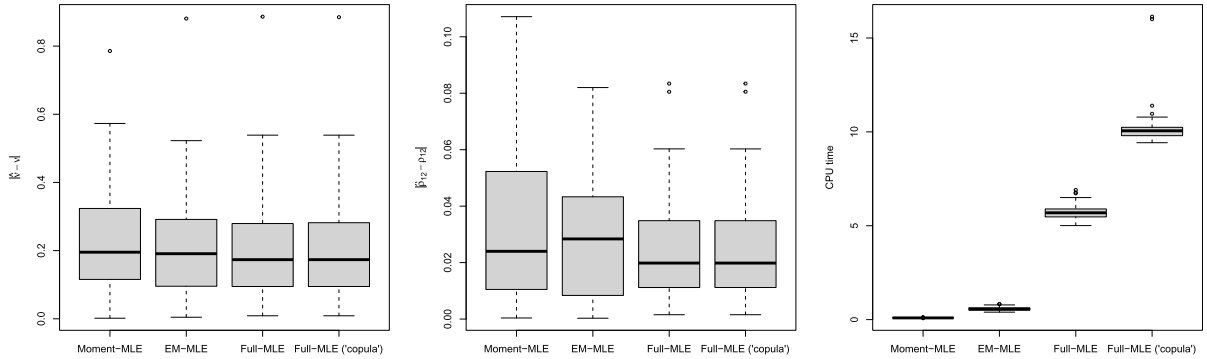

Figure 5: Boxplots of absolute errors for the degrees-of-freedom (left) and correlation parameter (middle) and of the run times (right) for a 7-dimensional t copula.

```
    res[,i,"loglik"] <- c(fit.mm$max.ll, fit.em$max.ll, fit.fl$max.ll,
                          fit.fl.cop@loglik)
    res[,i,"cpu"] <- c(t.mm, t.em, t.fl, t.fl.cop)
    res[,i,"df"] <- c(fit.mm$df, fit.em$df, fit.fl$df,
                      fit.fl.cop@estimate[d*(d-1)/2+1])
    res[,i,"rho12"] <- c(fit.mm$scale[1, 2], fit.em$scale[1, 2], fit.fl$scale[1, 2],
                         fit.fl.cop@estimate[1])
}
```

Figure 5 confirms that all methods give reasonable estimates and that all methods perform similarly but differ significantly in run-time. Furthermore, the function `fitCopula()` from *copula* is substantially slower than the "Full-MLE" method implemented in the function `fitStudentcopula()`, while giving almost identical results. The "EM-MLE" method is not much slower than "Moment-MLE".

3.2 Parameter Estimation for the Grouped t Copula

Moving from a t copula to a grouped t copula means moving from a model with known density to a model where the density needs to be estimated.

Daul et al. (2003) consider a grouped t copula where each group has size at least 2, so that all subgroups are t copulas. The authors suggest to estimate the degrees-of-freedom separately in each group. Luo and Shevchenko (2010) consider the grouped t copula with d groups (each group belongs to its own group of size 1) and suggest to jointly estimate the d degrees-of-freedom parameters by maximizing the copula log-likelihood. In both references, the matrix P is estimated by estimating pairwise Kendall's tau and using the approximate identity $\rho_{ij}^\tau \approx 2\arcsin(\rho_{ij})/\pi$ for $i \neq j$. Hintz et al. (2020) provide an efficient algorithm to compute the grouped t copula density, which involves numerical integration via RQMC. This is a key ingredient in (Hintz et al., 2020, Algorithm 2), implemented in the R function `fitgStudentcopula()`, for estimating copula parameters, which, given (pseudo-)observations $\boldsymbol{U}_1, \dots, \boldsymbol{U}_n$, works as follows:

1. Estimate all pairwise Kendall's tau and use the approximate identity $\rho_{ij}^\tau \approx 2\arcsin(\rho_{ij})/\pi$ to form a correlation matrix $\hat{P}$.
2. Find initial parameters $\hat{\nu}_k$ in all subgroups k with $d_k \geqslant 2$ by maximizing the marginal t copula log-likelihood. For groups with $d_k = 1$, choose the initial estimate from prior/expert experience.

3. With initial estimates $\hat{\nu}_k$, $k = 1, \ldots, S$, where S is the number of groups, maximize the (joint) grouped t copula log-likelihood over $(\nu_1, \ldots, \nu_S)$.

Daul et al. (2003) stop after the second step, which means that their procedure does not account for the dependence between the groups. We demonstrate this in the following simulation using the function `fitgStudentcopula()` where we simulate, for each sample size, 10 realizations of the estimators in Daul et al. (2003) (initial estimates) and Hintz et al. (2020) (MLEs). Note that, in order to control for the effect of an estimated scale matrix, we suppress its estimation by supplying it as argument `scale`. We set `verbose = FALSE` to suppress warnings, a point which we come back to below.

```
ns <- c(50, 250, 500, 750, 1000) # sample sizes
reps <- 10 # number of repetitions for each sample size
d <- 4 # dimension
df <- c(3, 8) # degrees-of-freedom for each group
grp <- rep(1:2, each = 2) # 2 components in each group
set.seed(1)
P <- cov2cor(rWishart(1, d, diag(d))[,,1]) # same known scale for all reps
fit.res <- array(, dim = c(length(ns), reps, length(df), 2),
                 dimnames = list(n = ns, rep = 1:reps, df = c("df1", "df2"),
                                 est = c("init", "MLE")))
for(j in 1:reps) {
   set.seed(j)
   sample <- rgStudentcopula(max(ns), groupings = groupings, scale = P, df = df)
   for(i in seq_along(ns)){
      fit <- fitgStudentcopula(u = sample[1:ns[i],], groupings = grp,
                               scale = P, verbose = FALSE)
      fit.res[i,j,,"init"] <- fit$df.init
      fit.res[i,j,,"MLE"]  <- fit$df
   }
}
```

As can be seen from Figure 6, maximization of the copula log-likelihood jointly over all degrees-of-freedom parameters improves the precision. For instance, even when $n = 1000$, the initial estimates for `df2` are much more fluctuating than the MLEs for `dff2`. The price to pay is a substantially longer run time, as the underlying procedure optimizes an estimated log-likelihood.

Optimizing an estimated log-likelihood poses another problem: In contrast to the true log-likelihood function, an estimate thereof may be bumpy, non-differentiable and multimodal, so that optimizers may not converge quickly. In order to limit the run time, the function `fitgStudentcopula()` uses at most 100 function calls. Via the argument `control` (see `?get_set_param()`), we can change this default by supplying a list `control.optim` which is passed to the underlying `optim()` call. We demonstrate this in a 6-dimensional example with a randomly sampled correlation matrix.

```
set.seed(42)
n <- 100; d <- 6; groupings <- rep(1:3, each = 2); df <- c(1, 4, 8)
rho <- runif(1, -1, 1)
P <- matrix(rho, ncol = d, nrow = d); diag(P) <- 1
U <- rgStudentcopula(n, groupings = groupings, df = df, scale = P)
(fit <- fitgStudentcopula(u = U, groupings = groupings))
```

```
## Warning in fitgStudentcopula(u = U, groupings = groupings): Maximum number
## of iterations exhausted in optim(); consider increasing 'optim.maxit' in the
```

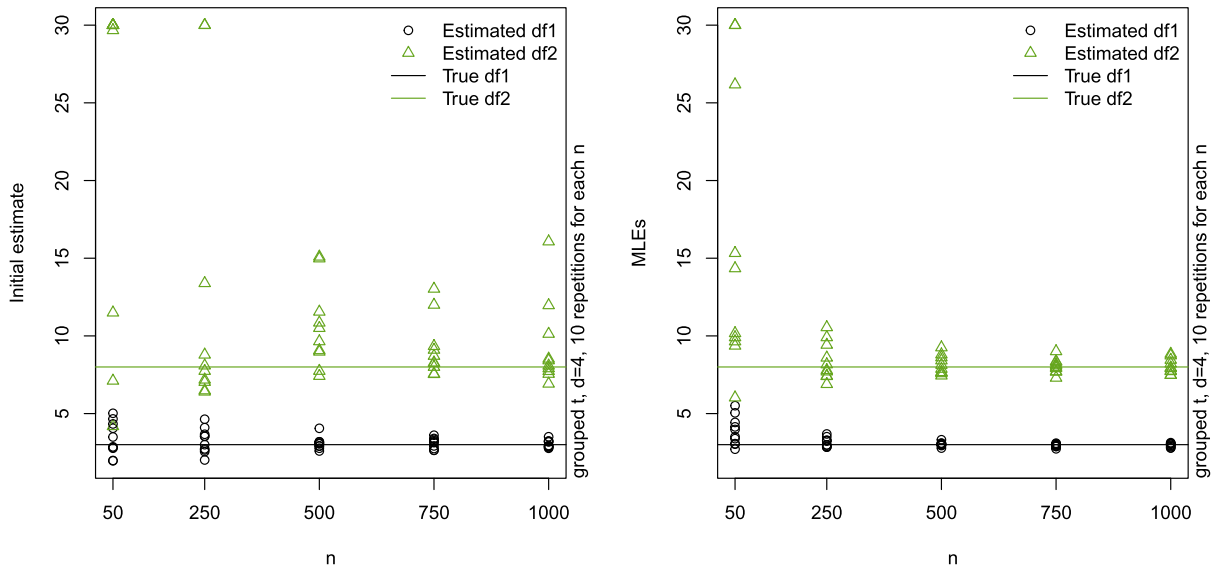

Figure 6: Initial estimates (left) and MLEs (right) for the degrees-of-freedom parameters of a grouped t copula with 2 groups.

```
## control argument.
## Call: fitgStudentcopula(u = U, groupings = groupings)
## Input data: 100 6-dimensional observations.
## Fitting a grouped t copula with unknown scale matrix and 3 group(s) and group ...
## Group
## 1 2 3
## 2 2 2
## Approximated log-likelihood at reported parameter estimates: 378.901900
## Fitting method used:  joint-MLE
## Estimated degrees-of-freedom for each group
## [1] 1.187898 3.632169 6.416981
## Estimated 'scale' matrix:
##        [,1]   [,2]   [,3]   [,4]   [,5]   [,6]
## [1,] 1.0000 0.8433 0.7149 0.8000 0.6892 0.7934
## [2,] 0.8433 1.0000 0.7961 0.8273 0.7648 0.8704
## [3,] 0.7149 0.7961 1.0000 0.7453 0.7145 0.8686
## [4,] 0.8000 0.8273 0.7453 1.0000 0.7950 0.7769
## [5,] 0.6892 0.7648 0.7145 0.7950 1.0000 0.7528
## [6,] 0.7934 0.8704 0.8686 0.7769 0.7528 1.0000
```

We can run `fitgStudentcopula()` again with a larger maximum number of function evaluations in the underlying `optim()` call, in which case the warning does not appear. The estimates do not differ significantly from the previous ones in this case.

```
(fit <- fitgStudentcopula(u = U, groupings = groupings,
                          control = list(control.optim = list(maxit = 200))))
```

```
## Call: fitgStudentcopula(u = U, groupings = groupings, control = list(...) ...
## Input data: 100 6-dimensional observations.
## Fitting a grouped t copula with unknown scale matrix and 3 group(s) and group ...
## Group
```

```
## 1 2 3
## 2 2 2
## Approximated log-likelihood at reported parameter estimates: 378.840400
## Fitting method used:  joint-MLE
## Estimated degrees-of-freedom for each group
## [1] 1.185490 3.627914 6.408446
## Estimated 'scale' matrix:
##        [,1]   [,2]   [,3]   [,4]   [,5]   [,6]
## [1,] 1.0000 0.8433 0.7149 0.8000 0.6892 0.7934
## [2,] 0.8433 1.0000 0.7961 0.8273 0.7648 0.8704
## [3,] 0.7149 0.7961 1.0000 0.7453 0.7145 0.8686
## [4,] 0.8000 0.8273 0.7453 1.0000 0.7950 0.7769
## [5,] 0.6892 0.7648 0.7145 0.7950 1.0000 0.7528
## [6,] 0.7934 0.8704 0.8686 0.7769 0.7528 1.0000
```

3.3 Parameter Estimation for the Skew t Copula

Parameter estimation of the skew t copula was addressed in Yoshiba (2018a). The authors suggest a "Full-MLE" method which optimizes the log-likelihood over the Cholesky factor, skewness parameters and degrees-of-freedom jointly. They present two methods for estimating the marginal quantile functions st^{-1}: One works by constructing a spline function for st and solving, for $p \in (0, 1)$, the equation $st(x) - p = 0$ for x numerically via `uniroot()`; this could be made faster by constructing a spline to approximate the quantile function by merely swapping the arguments, a point we will investigate in future research. The other method works by sampling from the univariate skew t, applying the distribution function st (estimated via `integrate()`) and then interpolating; see Yoshiba (2018a) for details. This method is implemented in the R function `fitskewtcopula()` based on code from the supplement provided by Yoshiba (2018a). In order to reduce the dimension of the optimization problem and to improve the accuracy of the estimated matrix P, we also implemented the "EM-MLE" method for this model, which follows the same logic as the "EM-MLE" method in the t case discussed earlier.

Parameter estimation of the skew t copula is the most challenging problem discussed in this paper: The log-likelihood is rather flat over a range of parameters (ν, γ), making numerical maximization even in small dimensions difficult. Furthermore, for each log-likelihood evaluation, nd-many quantiles need to be estimated, making this procedure not only slow, but also causing numerical problems as we optimize an estimated, possibly bumpy, log-likelihood function.

To illustrate the flatness of the profile log-likelihood, we plot, for fixed P, the profile log-likelihood of 3-dimensional, equi-skewed sample as a function of the degrees-of-freedom parameter ν and the skewness γ in Figure 7.

```
set.seed(12)
n <- 1000; d <- 3; rho <- 0.5; gamma <- rep(0.8, d); df <- 10
scale <- matrix(rho, ncol = d, nrow = d); diag(scale) <- 1
U <- rskewtcopula(n, scale = scale, gamma = gamma, df = df, pseudo = FALSE)
## Define log-likelihood function with known 'scale'
loglik <- function(par) sum(dskewtcopula(
   U, scale = scale, df = par[1], gamma = rep(par[2], d), log = TRUE))
## Evaluate 'loglik' on a grid and plot
n.grid <- 19
df_ <- seq(5, 11, length.out = n.grid)
gam_ <- seq(0, 2, length.out = n.grid)
```

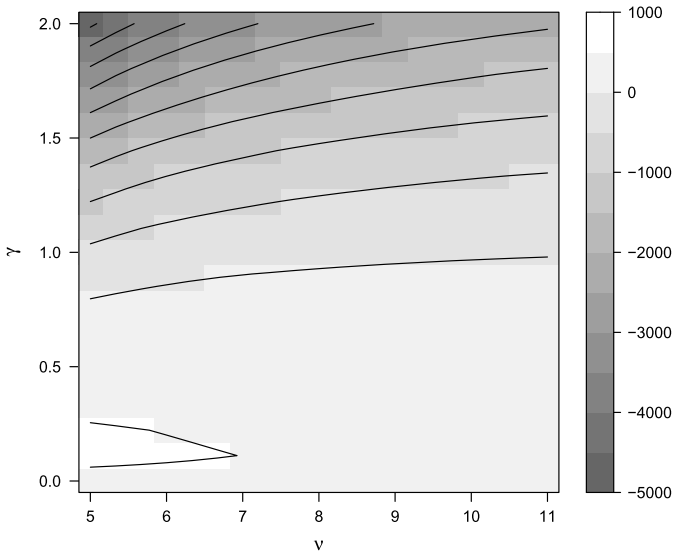

Figure 7: Profile log-likelihood for a trivariate skew t copula sample for fixed P.

```
grid <- expand.grid("nu" = df_, "gamma" = gam_)
lik <- apply(grid, 1, loglik)
val.lik <- cbind(grid, "loglik" = lik)
contourplot2(val.lik)
```

Next, we illustrate the difficulties arising when fitting skew t copulas by using "EM-MLE" and "Full-MLE" combined with two different optimization methods, `L-BFGS-B` and `Nelder-Mead`, and print the results below. All methods below use as a starting value for P the correlation matrix resulting from estimating all pairwise Kendall's tau and using the approximate identity $\rho_{ij}^{\tau} \approx 2\arcsin(\rho_{ij})/\pi$ as in the grouped case. Unless provided via the arguments `df.init` and `gamma.init`, the starting values for the degrees-of-freedom ν and $\boldsymbol{\gamma}$ are the midpoints of the intervals described by `df.bounds` and `gamma.bounds`, respectively.

```
t.em <- system.time(fit.em <- nvmix:::fitskewtcopula(
   U, fit.method = "EM-MLE", df.bounds = c(5, 15), gamma.bounds = c(0, 2)))[3]
t.fl <- system.time(fit.fl<- nvmix:::fitskewtcopula(
   U, fit.method = "Full-MLE", df.bounds = c(5, 15), gamma.bounds = c(0, 2)))[3]
t.em.neld <- system.time(fit.em.neld <- nvmix:::fitskewtcopula(
   U, fit.method = "EM-MLE", df.bounds = c(5, 15), gamma.bounds = c(0, 2),
   optim.method = "Nelder-Mead"))[3]
t.fl.neld <- system.time(fit.fl.neld<- nvmix:::fitskewtcopula(
   U, fit.method = "Full-MLE", df.bounds = c(5, 15), gamma.bounds = c(0, 2),
   optim.method = "Nelder-Mead"))[3]
```

```
##          EM (L-BFGS-B) Full (L-BFGS-B) EM (Nelder-Mead) Full (Nelder-Mead)
## max.ll     594.18325808    3.558158e+13     594.18399711       3.640864e+02
## df           5.00000000    1.161977e+01       5.00000003       5.000001e+00
## gamma        0.02091612    6.202521e-01       0.02161272       1.079474e-06
## rho[1,2]     0.54504929   -9.883032e-01       0.54510074       4.872877e-01
## rho[1,3]     0.58314636   -1.968595e-01       0.58320572       4.564044e-01
## rho[2,3]     0.54960991    4.503924e-02       0.54964702       5.544778e-01
## CPU         27.87400000    5.966400e+01      19.69400000       2.339300e+01
```

The results indicate that "EM-MLE" is faster than "Full-MLE", but none of the methods perform satisfactorily: The degrees-of-freedom in "EM-MLE" are exactly the left boundary point of the argument `df.bounds` and `gamma` is largely underestimated. "Full-MLE" combined with the optimizer "L-BFGS-B" gives poor results (the log-likelihood in the order of 10^{13} indicates numerical problems), while the optimizer "Nelder-Mead" produces, similarly to the EM methods, a degrees-of-freedom estimate at the boundary and $\gamma \approx 0$. In this case, the "EM-MLE" method gives a larger log-likelihood than "Full-MLE". In contrast to "Full-MLE", changing the underlying optimizer does not substantially change the results of "EM-MLE".

This particular example shall highlight the complications when fitting skew t copulas to data. The function `fitskewtcopula()` can be found, for experimental purposes, in the source code of the R package *nvmix*, but it is not exported. We plan to explore how such computational challenges can be addressed in future research.

4 Conclusion

While widely used (thanks to its tractability), the t copula imposes radial symmetry on the model that may not be justified in practice. More complicated, non-elliptical models such as the grouped t and skew t copulas have been proposed. The price to pay for moving to a more evolved model are substantial from a computational point of view, which we demonstrated in this paper. While in the case of the grouped t copula we were able to provide algorithms and software for parameter estimation, the skew t case is even more complicated due to the lack of an available marginal distribution and its quantile function. A possible avenue for future research would be to develop faster integration routines to compute the distribution function based on the stochastic representation and using properties of the model rather than relying on the R function `integrate()`. Furthermore, the notion of marginal consistency has only been defined for elliptical distributions; see Kano (1994) and Wang and Yan (2013). It would be interesting to study how this notion can be extended to non-elliptical distributions, for instance, using the notion of grouped elliptical distributions defined in Hintz et al. (2020). Finally, a non trivial task when using the grouped t copula is to decide which components belong to the same group, unless there is a natural grouping given, for instance, by industry sectors. It would be interesting to develop fast clustering algorithms to assist with this task that work well in high dimensions.

Supplementary Material

This paper can be reproduced with the R script `reproduce.R` and the R package *nvmix*, version 0.0-7.

Acknowledgment

We would like to thank Editor Jun Yan for valuable feedback on this work.

References

Azzalini A (2013). *The Skew-Normal and Related Families*, volume 3. Cambridge University Press.

Barndorff-Nielsen O (1977). Exponentially decreasing distributions for the logarithm of particle size. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 353(1674): 401–419.

Daul S, De Giorgi E, Lindskog F, McNeil A (2003). The grouped t-copula with an application to credit risk. Available at SSRN: http://dx.doi.org/10.2139/ssrn.1358956.

Demarta S, McNeil A (2005). The t copula and related copulas. *International Statistical Review*, 73(1): 111–129.

Dempster A, Laird D, Rubin N (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B, Methodological*, 39(1): 1–38.

Hintz E, Hofert M, Lemieux C (2020). Grouped normal variance mixtures. *Risks*, 8(4): 103.

Hintz E, Hofert M, Lemieux C (2021). Normal variance mixtures: distribution, density and parameter estimation. *Computational Statistics & Data Analysis*, 157C: 107175.

Hintz E, Hofert M, Lemieux C (2022). Multivariate normal variance mixtures in R: the R package nvmix. *Journal of Statistical Software*. To appear.

Hofert M, Hintz E, Lemieux C (2022). nvmix: multivariate normal variance mixtures. R package version 0.0-7, https://CRAN.R-project.org/package=nvmix.

Hofert M, Kojadinovic I, Maechler M, Yan J (2020). copula: multivariate dependence with copulas. R package version 1.0-0, https://CRAN.R-project.org/package=copula.

Hofert M, Lemieux C (2019). qrng: (randomized) quasi-random number generators. R package version 0.0-7, https://CRAN.R-project.org/package=qrng.

Hofert M, Mächler M (2011). Nested Archimedean copulas meet R: the nacopula package. *Journal of Statistical Software*, 39(9): 1–20.

Kano Y (1994). Consistency property of elliptic probability density functions. *Journal of Multivariate Analysis*, 51(1): 139–147.

Kojadinovic I, Yan J (2010). Modeling multivariate distributions with continuous margins using the copula R package. *Journal of Statistical Software*, 34(9): 1–20.

Luo X, Shevchenko P (2010). The t copula with multiple parameters of degrees of freedom: bivariate characteristics and application to risk management. *Quantitative Finance*, 10(9): 1039–1054.

Mashal R, Zeevi A (2002). Beyond correlation: extreme co-movements between financial assets. Available at SSRN: http://dx.doi.org/10.2139/ssrn.317122.

McNeil A, Frey R, Embrechts P (2015). *Quantitative Risk Management: Concepts, Techniques and Tools*. Princeton University Press.

Protassov R (2004). EM-based maximum likelihood parameter estimation for multivariate generalized hyperbolic distributions with fixed λ. *Statistics and Computing*, 14(1): 67–77.

Wang X, Yan J (2013). Practical notes on multivariate modeling based on elliptical copulas. *Journal de la Société Française de Statistique*, 154(1): 102–115.

Weibel M, Luethi D, Breymann W (2020). ghyp: generalized hyperbolic distributions and its special cases. R package version 1.6.1, https://CRAN.R-project.org/package=ghyp.

Yan J (2007). Enjoy the joy of copulas: with a package copula. *Journal of Statistical Software*, 21(4): 1–21.

Yoshiba T (2018a). Maximum likelihood estimation of skew-t copulas with its applications to stock returns. *Journal of Statistical Computation and Simulation*, 88(13): 2489–2506.