The Journal of Open Source Software

LocalCop: An R package for local likelihood inference for conditional copulas

Elif Fidan Acar[1, 2], **Martin Lysy**[3], **and Alan Kuchinsky**[1]

1 University of Manitoba **2** Hospital for Sick Children **3** University of Waterloo

DOI:

Software

- Review
- Repository
- Archive

Submitted:
Published:

Summary

Conditional copulas models allow the dependence structure between multiple response variables to be modelled as a function of covariates. **LocalCop** (Acar & Lysy, 2024) is an R/C++ package for computationally efficient semiparametric conditional copula modelling using a local likelihood inference framework developed in Acar, Craiu, & Yao (2011), Acar, Craiu, & Yao (2013) and Acar, Czado, & Lysy (2019).

Statement of Need

There are well-developed R packages such as **copula** (Hofert, Kojadinovic, Mächler, & Yan, 2023; Hofert & Mächler, 2011; Kojadinovic & Yan, 2010; Yan, 2007) and **VineCopula** (Nagler et al., 2023) for fitting copulas in various multivariate data settings. However, these software focus exclusively on unconditional dependence modelling and do not accommodate covariate information.

Aside from **LocalCop**, R packages for fitting conditional copulas are **gamCopula** (Nagler & Vatter, 2020) and **CondCopulas** (Derumigny, 2023). **gamCopula** estimates the covariate-dependent copula parameter using spline smoothing. While this typically has lower variance than the local likelihood estimate provided by **LocalCop**, it also tends to have lower accuracy (Acar et al., 2019). **CondCopulas** estimates the copula parameter using a semi-parametric maximum-likelihood method based on a kernel-weighted conditional concordance metric. **LocalCop** also uses kernel weighting, but it uses the full likelihood information of a given copula family rather than just that contained in the concordance metric, and is therefore more statistically efficient.

Local likelihood methods typically involve solving a large number of low-dimensional optimization problems and thus can be computationally intensive. To address this issue, **LocalCop** implements the local likelihood function in C++, using the R/C++ package **TMB** (Kristensen, Nielsen, Berg, Skaug, & Bell, 2016) to efficiently obtain the associated score function using automatic differentiation. Thus, **LocalCop** is able to solve each optimization problem very quickly using gradient-based algorithms. It also provides a means of easily parallelizing the optimization across multiple cores, rendering **LocalCop** competitive in terms of speed with other available software for conditional copula estimation.

Background

For any bivariate response vector (Y_1, Y_2), the conditional joint distribution given a covariate X is given by

$$F_X(y_1, y_2 \mid x) = C_X(F_{1|X}(y_1 \mid x), F_{2|X}(y_2 \mid x) \mid x), \tag{1}$$

The Journal of Open Source Software

Table 1: Copula families implemented in **LocalCop**.

Family	$\mathcal{C}(u,v \mid \theta,\nu)$	$\theta \in \Theta$	$\nu \in \Upsilon$	$g^{-1}(\eta)$	$\tau(\theta)$
Gaussian	$\Phi_\theta(\Phi^{-1}(u),\Phi^{-1}(v))$	$(-1,1)$	-	$\dfrac{e^\eta - e^{-\eta}}{e^\eta + e^{-\eta}}$	$\frac{2}{\pi}\arcsin(\theta)$
Student-t	$t_{\theta,\nu}(t_\nu^{-1}(u),t_\nu^{-1}(v))$	$(-1,1)$	$(0,\infty)$	$\dfrac{e^\eta - e^{-\eta}}{e^\eta + e^{-\eta}}$	$\frac{2}{\pi}\arcsin(\theta)$
Clayton	$(u^{-\theta}+v^{-\theta}-1)^{-\frac{1}{\theta}}$	$(0,\infty)$	-	e^η	$\frac{\theta}{\theta+2}$
Gumbel	$\exp\left[-\{(-\log u)^\theta + (-\log v)^\theta\}^{\frac{1}{\theta}}\right]$	$[1,\infty)$	-	$e^\eta + 1$	$1-\frac{1}{\theta}$
Frank	$-\frac{1}{\theta}\log\left\{1+\frac{(e^{-\theta u}-1)(e^{-\theta v}-1)}{e^{-\theta}-1}\right\}$	$(-\infty,\infty)\setminus\{0\}$	-	η	no closed form

where $F_{1|X}(y_1 \mid x)$ and $F_{2|X}(y_2 \mid x)$ are the conditional marginal distributions of Y_1 and Y_2 given X, and $C_X(u,v \mid x)$ is a conditional copula function. That is, for given $X=x$, the function $C_X(u,v \mid x)$ is a bivariate CDF with uniform margins.

The focus of **LocalCop** is on estimating the conditional copula function, which is modelled semi-parametrically as

$$C_X(u,v \mid x) = \mathcal{C}(u,v \mid \theta(x),\nu), \tag{2}$$

where $\mathcal{C}(u,v \mid \theta,\nu)$ is one of the parametric copula families listed in Table 1, the copula dependence parameter $\theta \in \Theta$ is an arbitrary function of X, and $\nu \in \Upsilon$ is an additional copula parameter present in some models. Since most parametric copula families have a restricted range $\Theta \subsetneq \mathbb{R}$, we describe the data generating model (DGM) in terms of the calibration function $\eta(x)$, such that

$$\theta(x) = g^{-1}(\eta(x)), \tag{3}$$

where $g^{-1}: \mathbb{R} \to \Theta$ an inverse-link function which ensures that the copula parameter has the correct range. The choice of $g^{-1}(\eta)$ is not unique and depends on the copula family. Table 1 displays the copula function $\mathcal{C}(u,v \mid \theta,\nu)$ for each of the copula families provided by **LocalCop**, along with other relevant information including the canonical choice of the inverse link function $g^{-1}(\eta)$. In Table 1, $\Phi^{-1}(p)$ denotes the inverse CDF of the standard normal; $t_\nu^{-1}(p)$ denotes the inverse CDF of the Student-t with ν degrees of freedom; $\Phi_\theta(z_1,z_2)$ denotes the CDF of a bivariate normal with mean $(0,0)$ and variance $\left[\begin{smallmatrix}1 & \theta \\ \theta & 1\end{smallmatrix}\right]$; and $t_{\theta,\nu}(z_1,z_2)$ denotes the CDF of a bivariate Student-t with location $(0,0)$, scale $\left[\begin{smallmatrix}1 & \theta \\ \theta & 1\end{smallmatrix}\right]$, and degrees of freedom ν.

Local likelihood estimation of the conditional copula parameter $\theta(x)$ uses Taylor expansions to approximate the calibration function $\eta(x)$ at an observed covariate value $X=x$ near a fixed point $X=x_0$, i.e.,

$$\eta(x) \approx \eta(x_0) + \eta^{(1)}(x_0)(x-x_0) + \ldots + \frac{\eta^{(p)}(x_0)}{p!}(x-x_0)^p.$$

One then estimates $\beta_k = \eta^{(k)}(x_0)/k!$ for $k=0,\ldots,p$ using a kernel-weighted local likelihood function

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \log\left\{c\left(u_i,v_i \mid g^{-1}(\boldsymbol{x}_i^T\boldsymbol{\beta}),\nu\right)\right\} K_h\left(\frac{x_i-x_0}{h}\right), \tag{4}$$

where (u_i,v_i,x_i) is the data for observation i, $\boldsymbol{x}_i = (1, x_i-x_0, (x_i-x_0)^2, \ldots, (x_i-x_0)^p)$, $\boldsymbol{\beta} = (\beta_0,\beta_1,\ldots,\beta_p)$, and $K_h(z)$ is a kernel function with bandwidth parameter $h>0$. Having maximized $\ell(\boldsymbol{\beta})$ in Equation 4, one estimates $\eta(x_0)$ by $\hat\eta(x_0) = \hat\beta_0$. Usually, a linear fit with $p=1$ suffices to obtain a good estimate in practice.

Usage

LocalCop is available on CRAN and GitHub. The two main package functions are:

- `CondiCopLocFit()`: For estimating the calibration function at a sequence of values $\boldsymbol{x}_0 = (x_{01}, \ldots, x_{0m})$.
- `CondiCopSelect()`: For selecting a copula family and bandwidth parameter using leave-one-out cross-validation (LOO-CV) with subsampling as described in Acar et al. (2019).

In the following example, we illustrate the model selection/tuning and fitting steps for data generated from a Clayton copula with conditional Kendall τ displayed in Figure 2. The CV metric for each combination of family and bandwidth are displayed in Figure 1.

```
library(LocalCop)   # local likelihood estimation
library(VineCopula) # simulate copula data
```

```
set.seed(2024)

# simulation setting
family <- 3                      # Clayton Copula
n_obs <- 300                     # number of observations
eta_fun <- function(x) {         # calibration function
  sin(5*pi*x) + cos(8*pi*x^2)
}
```

```
# simulate covariate values
x <- sort(runif(n_obs))

# simulate response data
eta_true <- eta_fun(x)                         # calibration parameter eta(x)
par_true <- BiCopEta2Par(family = family,      # copula parameter theta(x)
                         eta = eta_true)
udata <- VineCopula::BiCopSim(n_obs, family = family, par = par_true)
```

```
# model selection and tuning
bandset <- c(.02, .05, .1, .2) # set of bandwidth parameters
famset <- c(1, 2, 3, 4, 5)     # set of copula families
kernel <- KernGaus             # kernel function
degree <- 1                    # degree of local polynomial
n_loo <- 100                   # number of LOO-CV observations
                               # (can be much smaller than n_obs)
```

```
# calculate cv for each combination of family and bandwidth
cvselect <- CondiCopSelect(u1= udata[,1], u2 = udata[,2],
                           x = x, xind = n_loo,
                           kernel = kernel, degree = degree,
                           family = famset, band = bandset)
```

```
# extract the selected family and bandwidth from cvselect
cv_res <- cvselect$cv
i_opt <- which.max(cv_res$cv)
fam_opt <- cv_res[i_opt,]$family
band_opt <- cv_res[i_opt,]$band

# calculate eta(x) on a grid of values
x0 <- seq(0, 1, by = 0.01)
```

The Journal of Open Source Software

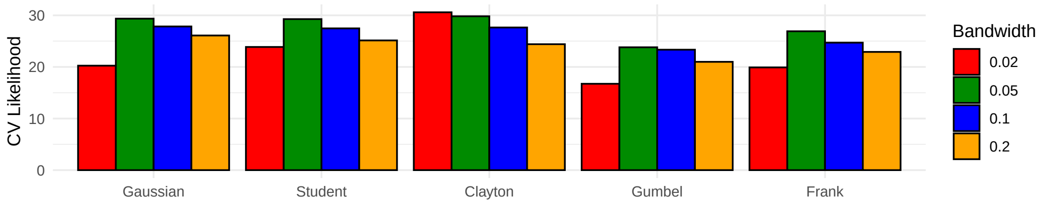

Figure 1: Cross-validation metric for each combination of family and bandwidth.

```
copfit <- CondiCopLocFit(u1 = udata[,1], u2 = udata[,2],
                         x = x, x0 = x0,
                         kernel = kernel, degree = degree,
                         family = fam_opt, band = band_opt)
# convert eta to Kendall tau
tau_loc <- BiCopEta2Tau(copfit$eta, family= fam_opt)
```

```
# simulate covariate values
x <- sort(runif(n_obs))

# simulate response data
eta_true <- eta_fun(x)                       # calibration parameter eta(x)
par_true <- BiCopEta2Par(family = family,   # copula parameter theta(x)
                         eta = eta_true)
udata <- VineCopula::BiCopSim(n_obs, family = family, par = par_true)
```

```
# model selection and tuning
bandset <- c(.02, .05, .1, .2) # set of bandwidth parameters
famset <- c(1, 2, 3, 4, 5)      # set of copula families
kernel <- KernGaus              # kernel function
degree <- 1                     # degree of local polynomial
n_loo <- 100                    # number of LOO-CV observations
                                # (can be much smaller than n_obs)
```

```
# calculate cv for each combination of family and bandwidth
cvselect <- CondiCopSelect(u1= udata[,1], u2 = udata[,2],
                           x = x, xind = n_loo,
                           kernel = kernel, degree = degree,
                           family = famset, band = bandset)
```

```
# extract the selected family and bandwidth from cvselect
cv_res <- cvselect$cv
i_opt <- which.max(cv_res$cv)
fam_opt <- cv_res[i_opt,]$family
band_opt <- cv_res[i_opt,]$band

# calculate eta(x) on a grid of values
x0 <- seq(0, 1, by = 0.01)
copfit <- CondiCopLocFit(u1 = udata[,1], u2 = udata[,2],
                         x = x, x0 = x0,
                         kernel = kernel, degree = degree,
                         family = fam_opt, band = band_opt)
# convert eta to Kendall tau
tau_loc <- BiCopEta2Tau(copfit$eta, family= fam_opt)
```

In Figure 2, we compare the true conditional Kendall τ to estimates using each of the

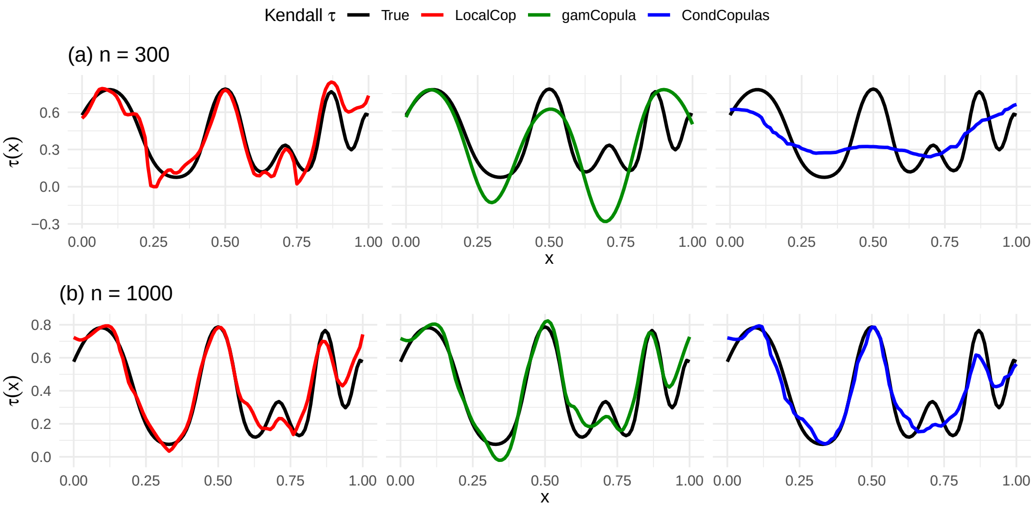

Figure 2: True vs estimated conditional Kendall τ using various methods.

three conditional copula fitting packages **LocalCop**, **gamCopula**, and **CondCopulas**, for sample sizes $n = 300$ and $n = 1000$. In **gamCopula**, selection of the copula family smoothing splines is done using the generalized CV framework provided by the R package **mgcv** (Wood, 2017). In **CondCopulas**, selection of the bandwidth parameter is done using LOO-CV. In this particular example, the sample size of $n = 300$ is not large enough for **gamCopula** to pick a sufficiently flexible spline basis, and **CondCopulas** picks a large bandwidth which oversmooths the data. For the larger sample size $n = 1000$, the three methods exhibit similar accuracy.

Acknowledgements

We acknowledge funding support from the Natural Sciences and Engineering Research Council of Canada Discovery Grants RGPIN-2020-06753 (Acar) and RGPIN-2020-04364 (Lysy).

References

Acar, E. F., Craiu, R. V., & Yao, F. (2011). Dependence calibration in conditional copulas: A nonparametric approach. *Biometrics*, *67*(2), 445–453.

Acar, E. F., Craiu, R. V., & Yao, F. (2013). Statistical testing of covariate effects in conditional copula models. *Electronic Journal of Statistics*, *7*, 2822–2850.

Acar, E. F., Czado, C., & Lysy, M. (2019). Dynamic vine copula models for multivariate time series data. *Econometrics and Statistics*, *12*, 181–197.

Acar, E. F., & Lysy, M. (2024). *LocalCop: LocalCop: Local likelihood inference for conditional copula models*. Retrieved from https://CRAN.R-project.org/package=LocalCop

Derumigny, A. (2023). *CondCopulas: Estimation and inference for conditional copula models*. Retrieved from https://CRAN.R-project.org/package=CondCopulas

Hofert, M., Kojadinovic, I., Mächler, M., & Yan, J. (2023). *Copula: Multivariate dependence with copulas*. Retrieved from https://CRAN.R-project.org/package=copula

Hofert, M., & Mächler, M. (2011). Nested archimedean copulas meet R: The nacopula package. *Journal of Statistical Software*, *39*(9), 1–20. Retrieved from https://www.jstatsoft.org/v39/i09/

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

Retrieved from https://www.jstatsoft.org/v34/i09/

Kristensen, K., Nielsen, A., Berg, C. W., Skaug, H., & Bell, B. M. (2016). TMB: Automatic differentiation and Laplace approximation. *Journal of Statistical Software*, *70*(5), 1–21. doi:10.18637/jss.v070.i05

Nagler, T., Schepsmeier, U., Stoeber, J., Brechmann, E. C., Graeler, B., & Erhardt, T. (2023). *VineCopula: Statistical inference of vine copulas*. Retrieved from https://CRAN.R-project.org/package=VineCopula

Nagler, T., & Vatter, T. (2020). *gamCopula: Generalized additive models for bivariate conditional dependence structures and vine copulas*. Retrieved from https://CRAN.R-project.org/package=gamCopula

Wood, S. N. (2017). *Generalized additive models: An introduction with R* (2nd ed.). Chapman; Hall/CRC.

Yan, J. (2007). Enjoy the joy of copulas: With a package copula. *Journal of Statistical Software*, *21*(4), 1–21. Retrieved from https://www.jstatsoft.org/v21/i04/