

# Package ‘LocalCop’

January 20, 2025

**Title** Local Likelihood Inference for Conditional Copula Models

**Version** 0.0.2

**Date** 2024-08-30

**Description** Implements a local likelihood estimator for the dependence parameter in bivariate conditional copula models. Copula family and local likelihood bandwidth parameters are selected by leave-one-out cross-validation. The models are implemented in 'TMB', meaning that the local score function is efficiently calculated via automated differentiation (AD), such that quasi-Newton algorithms may be used for parameter estimation.

**URL** <https://github.com/mlysy/LocalCop>,  
<https://mlysy.github.io/LocalCop/>

**BugReports** <https://github.com/mlysy/LocalCop/issues>

**License** GPL-3

**Encoding** UTF-8

**Depends** R (>= 3.5.0)

**LinkingTo** TMB, RcppEigen

**Imports** TMB (>= 1.7.20), VineCopula

**RoxygenNote** 7.3.1

**Suggests** testthat, parallel, knitr, rmarkdown, bookdown, kableExtra,  
dplyr, readr, tidyr, ggplot2

**VignetteBuilder** knitr

**NeedsCompilation** yes

**Author** Elif Fidan Acar [aut],  
Martin Lysy [aut, cre],  
Alan Kuchinsky [ctb]

**Maintainer** Martin Lysy <mlysy@uwaterloo.ca>

**Repository** CRAN

**Date/Publication** 2024-09-12 17:41:03 UTC

## Contents

LocalCop-package	2
CondiCopLikCV	3
CondiCopLocFit	5
CondiCopLocFun	7
CondiCopSelect	9
ConvertPar	11
KernFun	12
KernWeight	13
<b>Index</b>	<b>14</b>

---

LocalCop-package	<i>Local likelihood inference for conditional copula models.</i>
------------------	--

---

## Description

Fits a bivariate conditional copula  $C(u_1, u_2 | \theta_x)$ , where  $\theta_x$  is a variable dependence parameter, nonparametrically estimated from a single covariate  $x$  via local likelihood.

## Author(s)

**Maintainer:** Martin Lysy <mlysy@uwaterloo.ca>

Authors:

- Elif Acar <elif.acar@umanitoba.ca>

Other contributors:

- Alan Kuchinsky [contributor]

## See Also

Useful links:

- <https://github.com/mlysy/LocalCop>
- Report bugs at <https://github.com/mlysy/LocalCop/issues>

## Examples

```
# simulate data
set.seed(123)
family <- 5 # Frank copula
n <- 1000
x <- runif(n) # covariate values
eta_fun <- function(x) 2*cos(12*pi*x) # copula dependence parameter
eta_true <- eta_fun(x)
par_true <- BiCopEta2Par(family, eta = eta_true)
udata <- VineCopula::BiCopSim(n, family=family,
```

```

par = par_true$par)

# bandwidth and family selection
bandset <- c(.01, .04, .1) # bandwidth set
famset <- c(2, 5) # family set
n_loo <- 100 # number of leave-one-out observations in CV likelihood calculation
system.time({
  cvsel <- CondiCopSelect(u1= udata[,1], u2 = udata[,2],
                        x = x, family = famset, band = bandset,
                        xind = n_loo)
})

# compare estimates to true value
xseq <- cvsel$x
famsel <- cvsel$cv$family
bandsel <- cvsel$cv$band
etasel <- cvsel$eta
clrs <- c("red", "blue", "green4")
names(clrs) <- bandsel

plot_fun <- function(fam) {
  nband <- length(bandset)
  if(fam == 2) {
    famind <- 1:nband
    main <- "Student-t Copula"
  } else {
    famind <- nband+1:nband
    main <- "Frank Copula"
  }
  plot(xseq, BiCopEta2Tau(family, eta = eta_fun(xseq)),
       type = "l", lwd = 2, ylim = c(-.5, .5),
       xlab = expression(x), ylab = expression(tau(x)),
       main = main)
  for(ii in famind) {
    lines(xseq, BiCopEta2Tau(fam, eta = etasel[,ii]),
         col = clrs[as.character(bandsel[ii])], lwd = 1)
  }
  legend("bottomright", fill = clrs,
        legend = paste0("band_", bandsel[famind],
                        " = ", signif(cvsel$cv$cv[famind], 3)))
}

oldpar <- par(mfrow = c(1,2))
plot_fun(2)
plot_fun(5)
par(oldpar)

```

**Description**

Leave-one-out local likelihood copula parameter estimates are interpolated, then used to calculate the conditional copula likelihood function.

**Usage**

```
CondiCopLikCV(
  u1,
  u2,
  family,
  x,
  xind = 100,
  degree = 1,
  eta,
  nu,
  kernel = KernEpa,
  band,
  optim_fun,
  cveta_out = FALSE,
  cv_all = FALSE,
  cl = NA
)
```

**Arguments**

u1	Vector of first uniform response.
u2	Vector of second uniform response.
family	An integer defining the bivariate copula family to use. See <a href="#">ConvertPar()</a> .
x	Vector of observed covariate values.
xind	Vector of indices in <code>sort(x)</code> at which to calculate leave-one-out parameter estimates. Can also be supplied as a single integer, in which case <code>xind</code> equally spaced observations are taken from <code>x</code> .
degree	Integer specifying the polynomial order of the local likelihood function. Currently only 0 and 1 are supported.
eta, nu, kernel, band, optim_fun, cl	See <a href="#">CondiCopLocFit()</a> .
cveta_out	If TRUE, return the CV estimate of eta at each point in <code>x</code> in addition to the CV log-likelihood.
cv_all	If FALSE, evaluate the CV likelihood at only the leave-one-out observations specified by <code>xind</code> . Otherwise, interpolate the leave-one-out estimates of eta to all values in <code>x</code> , and evaluate the CV likelihood at all observations.

**Value**

If `cveta_out = FALSE`, scalar value of the cross-validated log-likelihood. Otherwise, a list with elements:

`x` The sorted values of `x`.  
`eta` The leave-one-out estimates interpolated from the values in `x.ind` to all of those in `x`.  
`nu` The scalar value of the estimated (or provided) second copula parameter.  
`loglik` The cross-validated log-likelihood.

### See Also

This function is typically used in conjunction with [CondiCopSelect\(\)](#); see example there.

---

CondiCopLocFit	<i>Local likelihood estimation.</i>
----------------	-------------------------------------

---

### Description

Estimate the bivariate copula dependence parameter `eta` at multiple covariate values.

### Usage

```

CondiCopLocFit(
  u1,
  u2,
  family,
  x,
  x0,
  nx = 100,
  degree = 1,
  eta,
  nu,
  kernel = KernEpa,
  band,
  optim_fun,
  cl = NA
)
  
```

### Arguments

<code>u1</code>	Vector of first uniform response.
<code>u2</code>	Vector of second uniform response.
<code>family</code>	An integer defining the bivariate copula family to use. See <a href="#">ConvertPar()</a> .
<code>x</code>	Vector of observed covariate values.
<code>x0</code>	Vector of covariate values within <code>range(x)</code> at which to fit the local likelihood. Does not have to be a subset of <code>x</code> .
<code>nx</code>	If <code>x0</code> is missing, defaults to <code>nx</code> equally spaced values in <code>range(x)</code> .
<code>degree</code>	Integer specifying the polynomial order of the local likelihood function. Currently only 0 and 1 are supported.

eta	Optional initial value of the copula dependence parameter (scalar). If missing will be estimated unconditionally by <code>VineCopula::BiCopEst()</code> .
nu	Optional initial value of second copula parameter, if it exists. If missing and required, will be estimated unconditionally by <code>VineCopula::BiCopEst()</code> . If provided and required, will not be estimated.
kernel	Kernel function to use. Should accept a numeric vector parameter and return a non-negative numeric vector of the same length. See <code>KernFun()</code> .
band	Kernel bandwidth parameter (positive scalar). See <code>KernWeight()</code> .
optim_fun	Optional specification of local likelihood optimization algorithm. See <b>Details</b> .
cl	Optional parallel cluster created with <code>parallel::makeCluster()</code> , in which case optimization for each element of $x_0$ will be done in parallel on separate cores. If <code>cl == NA</code> , computations are run serially.

### Details

By default, optimization is performed with the quasi-Newton algorithm provided by `stats::nlminb()`, which uses gradient information provided by automatic differentiation (AD) as implemented by **TMB**.

If the default method is to be overridden, `optim_fun` should be provided as a function taking a single argument corresponding to the output of `CondiCopLocFun()`, and return a scalar value corresponding to the estimate of  $\eta$  at a given covariate value in  $x_0$ . Note that **TMB** calculates the *negative* local (log)likelihood, such that the objective function is to be minimized. See **Examples**.

### Value

List with the following elements:

- `x` The vector of covariate values  $x_0$  at which the local likelihood is fit.
- `eta` The vector of estimated dependence parameters of the same length as  $x_0$ .
- `nu` The scalar value of the estimated (or provided) second copula parameter.

### Examples

```
# simulate data
family <- 5 # Frank copula
n <- 1000
x <- runif(n) # covariate values
eta_fun <- function(x) 2*cos(12*pi*x) # copula dependence parameter
eta_true <- eta_fun(x)
par_true <- BiCopEta2Par(family, eta = eta_true)
udata <- VineCopula::BiCopSim(n, family=family,
                             par = par_true$par)

# local likelihood estimation
x0 <- seq(min(x), max(x), len = 100)
band <- .02
system.time({
  eta_hat <- CondiCopLocFit(u1 = udata[,1], u2 = udata[,2],
```

```

        family = family, x = x, x0 = x0, band = band)
  })

# custom optimization routine using stats::optim (gradient-free)
my_optim <- function(obj) {
  opt <- stats::optim(par = obj$par, fn = obj$fn, method = "Nelder-Mead")
  return(opt$par[1]) # always return constant term, even if degree > 0
}
system.time({
  eta_hat2 <- CondiCopLocFit(u1 = udata[,1], u2 = udata[,2],
    family = family, x = x, x0 = x0, band = band,
    optim_fun = my_optim)
})

plot(x0, BiCopEta2Tau(family, eta = eta_fun(x0)), type = "l",
  xlab = expression(x), ylab = expression(tau(x)))
lines(x0, BiCopEta2Tau(family, eta = eta_hat$eta), col = "red")
lines(x0, BiCopEta2Tau(family, eta = eta_hat2$eta), col = "blue")
legend("bottomright", fill = c("black", "red", "blue"),
  legend = c("True", "optim_default", "Nelder-Mead"))

```

---

CondiCopLocFun

*Create a TMB local likelihood function.*


---

## Description

Wraps a call to [TMB::MakeADFun\(\)](#).

## Usage

```
CondiCopLocFun(u1, u2, family, x, x0, wgt, degree = 1, eta, nu)
```

## Arguments

u1	Vector of first uniform response.
u2	Vector of second uniform response.
family	An integer defining the bivariate copula family to use. See <a href="#">ConvertPar()</a> .
x	Vector of observed covariate values.
x0	Scalar covariate value at which to evaluate the local likelihood. Does not have to be a subset of x.
wgt	Vector of positive kernel weights.
degree	Integer specifying the polynomial order of the local likelihood function. Currently only 0 and 1 are supported.
eta	Value of the copula dependence parameter. Scalar or vector of length two, depending on whether degree is 0 or 1.
nu	Value of the other copula parameter. Scalar or vector of same length as u1. Ignored if family != 2.

**Value**

A list as returned by a call to `TMB::MakeADFun()`. In particular, this contains elements `fun` and `gr` for the *negative* local likelihood and its gradient with respect to `eta`.

**Examples**

```
# the following example shows how to create
# an unconditional copula likelihood function

# simulate data
n <- 1000 # sample size
family <- 2 # Student-t copula
rho <- runif(1, -1, 1) # unconditional dependence parameter
nu <- runif(1, 4, 20) # degrees of freedom parameter
udata <- VineCopula::BiCopSim(n, family = family, par = rho, par2 = nu)

# create likelihood function

# parameter conversion: equivalent to BiCopPar2Eta(family = 2, ...)
rho2eta <- function(rho) .5 * log((1+rho)/(1-rho))
nll_obj <- CondiCopLocFun(u1 = udata[,1], u2 = udata[,2], family = family,
                        x = rep(0, n), x0 = 0, # centered covariate x - x0 == 0
                        wgt = rep(1, n), # unweighted
                        degree = 0, # zero-order fit
                        eta = c(rho2eta(rho), 0),
                        nu = nu)

# likelihood function: recall that TMB requires a _negative_ ll
stucop_lik <- function(rho) {
  -nll_obj$fn(c(rho2eta(rho), 0))
}

# compare to VineCopula.
rhovec <- runif(50, -1, 1)
system.time({
  ll1 <- sapply(rhovec, stucop_lik) # LocalCop
})
system.time({
  ll2 <- sapply(rhovec, function(rho) {
    # VineCopula
    sum(log(VineCopula::BiCopPDF(u1 = udata[,1], u2 = udata[,2],
                                family = family,
                                par = rho, par2 = nu)))
  })
})

# difference between the two
range(ll1 - ll2)
```



---

CondiCopSelect      *Local likelihood bandwidth and/or family selection.*

---

### Description

Selects among a set of bandwidths and/or copula families the one which maximizes the cross-validated local likelihood. See [CondiCopLikCV\(\)](#) for details.

### Usage

```
CondiCopSelect(
  u1,
  u2,
  family,
  x,
  xind = 100,
  degree = 1,
  nu,
  kernel = KernEpa,
  band,
  nband = 6,
  optim_fun,
  cv_all = FALSE,
  full_out = TRUE,
  cl = NA
)
```

### Arguments

u1	Vector of first uniform response.
u2	Vector of second uniform response.
family	Vector of integers specifying the family set. See <a href="#">ConvertPar()</a> .
x	Vector of observed covariate values.
xind	Specification of xind for each bandwidth. Can be a scalar integer, a vector of nband integers, or a list of nband vectors of integers.
degree	Integer specifying the polynomial order of the local likelihood function. Currently only 0 and 1 are supported.
nu	Optional vector of fixed nu parameter for each family. If missing or NA get estimated from the data (if required)
kernel, optim_fun, cl	See <a href="#">CondiCopLocFit()</a> .
band	Vector of positive numbers specifying the bandwidth value set.
nband	If band is missing, automatically choose nband bandwidth values spanning the range of x.

cv_all	If FALSE, evaluate the CV likelihood at only the leave-one-out observations specified by xind. Otherwise, interpolate the leave-one-out estimates of eta to all values in x, and evaluate the CV likelihood at all observations.
full_out	Logical; whether or not to output all fitted models or just the selected family/bandwidth combination. See <b>Value</b> .

### Value

If full\_out = FALSE, a list with elements family and bandwidth containing the selected value of each. Otherwise, a list with the following elements:

cv A data frame with nBF = length(band) × length(family) rows and columns named family, band, and cv containing the cross-validated likelihood evaluated at each combination of bandwidth and family values.

x The sorted values of x.

eta A length(x) × nBF matrix of eta estimates, the columns of which are in the same order as the rows of cv.

nu A vector of length nBF second copula parameters, with zero if they don't exist.

### Examples

```
# simulate data
set.seed(123)
family <- 5 # Frank copula
n <- 1000
x <- runif(n) # covariate values
eta_fun <- function(x) 2*cos(12*pi*x) # copula dependence parameter
eta_true <- eta_fun(x)
par_true <- BiCopEta2Par(family, eta = eta_true)
udata <- VineCopula::BiCopSim(n, family=family,
                             par = par_true$par)

# bandwidth and family selection
bandset <- c(.01, .04, .1) # bandwidth set
famset <- c(2, 5) # family set
n_loo <- 100 # number of leave-one-out observations in CV likelihood calculation
system.time({
  cvsel <- CondiCopSelect(u1= udata[,1], u2 = udata[,2],
                        x = x, family = famset, band = bandset,
                        xind = n_loo)
})

# compare estimates to true value
xseq <- cvsel$x
famsel <- cvsel$cv$family
bandsel <- cvsel$cv$band
etasel <- cvsel$eta
clrs <- c("red", "blue", "green4")
names(clrs) <- bandset

plot_fun <- function(fam) {
```

```

nband <- length(bandset)
if(fam == 2) {
  famind <- 1:nband
  main <- "Student-t Copula"
} else {
  famind <- nband+1:nband
  main <- "Frank Copula"
}
plot(xseq, BiCopEta2Tau(family, eta = eta_fun(xseq)),
     type = "l", lwd = 2, ylim = c(-.5, .5),
     xlab = expression(x), ylab = expression(tau(x)),
     main = main)
for(ii in famind) {
  lines(xseq, BiCopEta2Tau(fam, eta = etasel[,ii]),
       col = clr[as.character(bandsel[ii])], lwd = 1)
}
legend("bottomright", fill = clr,
      legend = paste0("band_", bandsel[famind],
                     " = ", signif(cvsel$cv$cv[famind], 3)))
}

oldpar <- par(mfrow = c(1,2))
plot_fun(2)
plot_fun(5)
par(oldpar)

```

---

 ConvertPar

*Conversions between various bivariate copula parametrizations.*


---

## Description

Conversions between various bivariate copula parametrizations.

## Usage

```
BiCopEta2Par(family, eta, eta2 = 0)
```

```
BiCopPar2Eta(family, par, par2 = 0)
```

```
BiCopEta2Tau(family, eta, eta2 = 0)
```

```
BiCopTau2Eta(family, tau)
```

## Arguments

family	An integer defining the bivariate copula family to use. See <b>Details</b> .
eta, eta2	Vector of parameters on the eta scale. See <b>Details</b> .
par, par2	Vector of parameters on the par scale.
tau	Vector of parameters on the tau scale.

**Details**

The copula family integer codes are identical to those of the **VineCopula** package. Currently, the following families are implemented:

- 1 Gaussian copula.
- 2 Student-t copula.
- 3 Clayton copula.
- 4 Gumbel copula.
- 5 Frank copula.
- 13 Clayton copula – rotated 180 degrees.
- 14 Gumbel copula – rotated 180 degrees.
- 23 Clayton copula – rotated 90 degrees.
- 24 Gumbel copula – rotated 90 degrees.
- 33 Clayton copula – rotated 270 degrees.
- 34 Gumbel copula – rotated 270 degrees.

**Value**

Vector of converted parameters.

---

KernFun

*Local likelihood kernel functions.*

---

**Description**

Local likelihood kernel functions.

**Usage**

KernEpa(t)

KernGaus(t)

KernBeta(t, par = 0.5)

KernBiQuad(t)

KernTriAng(t)

**Arguments**

- t                      Vector of distances from mode of kernel.
- par                    Shape parameter for Beta kernel (positive scalar).

**Details**

Describe kernels here.

**Value**

Vector of kernel weights.

---

KernWeight	<i>Calculate local likelihood kernel weights.</i>
------------	---

---

**Description**

Calculate local likelihood kernel weights.

**Usage**

```
KernWeight(x, x0, band, kernel = KernEpa, band_type = "constant")
```

**Arguments**

x	Vector of observed covariate values.
x0	Scalar covariate value at which local likelihood estimation is performed.
band	Kernel bandwidth parameter (positive scalar). See <b>Details</b> .
kernel	Kernel function to use. Should accept a numeric vector parameter and return a non-negative numeric vector of the same length. See <a href="#">KernFun()</a> .
band_type	A character string specifying the type of bandwidth: either "constant" or "variable". See <b>Details</b> .

**Details**

For the constant bandwidth of size  $\text{band} = h$ , the weights are calculated as

$$\text{wgt} = \text{kernel}((x-x_0) / h) / h$$

where `kernel` is the kernel function. For bandwidth type "variable", a fixed fraction band of observations is used, i.e,

$$h = \text{sort}(\text{abs}(x-x_0))[\text{floor}(\text{band}*\text{length}(x))]$$
**Value**

A vector of nonnegative kernel weights of the same length as `x`.

**Examples**

```
x <- sort(runif(20))
x0 <- runif(1, min = min(x), max = max(x))
KernWeight(x, x0, band=0.3, kernel = KernEpa, band_type = "constant")
KernWeight(x, x0, band=0.3, kernel = KernEpa, band_type = "variable")
```

# Index

BiCopEta2Par (ConvertPar), 11  
BiCopEta2Tau (ConvertPar), 11  
BiCopPar2Eta (ConvertPar), 11  
BiCopTau2Eta (ConvertPar), 11

CondiCopLikCV, 3  
CondiCopLikCV(), 9  
CondiCopLocFit, 5  
CondiCopLocFit(), 4, 9  
CondiCopLocFun, 7  
CondiCopLocFun(), 6  
CondiCopSelect, 9  
CondiCopSelect(), 5  
ConvertPar, 11  
ConvertPar(), 4, 5, 7, 9

KernBeta (KernFun), 12  
KernBiQuad (KernFun), 12  
KernEpa (KernFun), 12  
KernFun, 12  
KernFun(), 6, 13  
KernGaus (KernFun), 12  
KernTriAng (KernFun), 12  
KernWeight, 13  
KernWeight(), 6

LocalCop (LocalCop-package), 2  
LocalCop-package, 2

parallel::makeCluster(), 6

stats::nlminb(), 6

TMB::MakeADFun(), 7, 8

VineCopula::BiCopEst(), 6