Package: PPMiss (via r-universe)

September 8, 2024

Type Package

Date 2023-09-04

Title Copula-Based Estimator for Long-Range Dependent Processes under Missing Data

Version 0.1.1

Depends R (>= 4.0.0)

Imports copula, pracma, stats, zoo

Description Implements the copula-based estimator for univariate long-range dependent processes, introduced in Pumi et al. (2023) <doi:10.1007/s00362-023-01418-z>. Notably, this estimator is capable of handling missing data and has been shown to perform exceptionally well, even when up to 70% of data is missing (as reported in <arXiv:2303.04754>) and has been found to outperform several other commonly applied estimators.

License GPL (>= 3)

Encoding UTF-8

NeedsCompilation yes

RoxygenNote 7.2.3

Author Taiane Schaedler Prass [aut, cre, com] (<https://orcid.org/0000-0003-3136-909X>), Guilherme Pumi [aut, ctb] (<https://orcid.org/0000-0002-6256-3170>)

Maintainer Taiane Schaedler Prass <taianeprass@gmail.com>

Date/Publication 2023-09-04 22:20:02 UTC

Repository https://tsprass.r-universe.dev

RemoteUrl https://github.com/cran/PPMiss

RemoteRef HEAD

RemoteSha 78f3c663b2b275970f8c6cc2d4cd68b940d2177a

Contents

arfima.coefs 2
d.fit 3
k1fun 6
kdens 8
PPMiss.copulas 8

Index **11**

`arfima.coefs` *Coefficients of an ARFIMA(p,d,q) model*

Description

This function calculates the coefficients $c_k, k \geq 0$ corresponding to $\theta(z)\phi^{-1}(z)(1-z)^{-d} = \sum_{k=0}^{\infty} c_k z^k$, up to a truncation point

Usage

```
arfima.coefs(ar = NULL, ma = NULL, d = 0, trunc = 1)
```

Arguments

`ar`	the coefficients of the autoregressive polinomial. Default is NULL
`ma`	the coefficients of the moving average polinomial. Default is null
`d`	the long memory parameter. Default is 0.
`trunc`	the truncation point. Default is 1.

Value

The coefficients values up to order 'trunc'.

Examples

```
cks <- arfima.coefs(d = 0.3, trunc = 5)
cks

cks <- arfima.coefs(d = 0.1, trunc = 5, ar = 0.5, ma = 0.6)
cks
```

`d.fit` *Long memory parameter estimation*

Description

Let θ_h be the copula parameter associated to (X_t, X_{t+h}) and $\hat{\theta}_h$ be an estimate of θ_h based on pseudo observations. The long memory parameter d is estimated by

$$\hat{d} := \text{argmin}_{|d|<0.5}\left\{ \sum_{h=s}^{m} \left| \hat{K}_1(\hat{\theta}_h - a) - \frac{\Gamma(1-d)}{\Gamma(d)} h^{2d-1} \right|^r \right\}, \quad r > 0$$

.

Usage

```
d.fit(xt, copula = gauss, dCdtheta = dCtheta_gauss, theta.lower = -1,
  theta.upper = 1, optim.method = "Brent", method = "mpl", s = 1,
  m = 24, theta.start = 0.1, empirical = TRUE, r = 2, a = 0,
  d.interval = c(-0.5, 0.5))
```

Arguments

`xt`	a vector with the observed time series. Missing observations are allowed.
`copula`	an object of class 'copula'. Readily available options are `frank`, `amh`, `fgm` and `gauss`. Other copulas can be used but the user must provide the corresponding `dCdtheta`. Default is `gauss`.
`dCdtheta`	a two parameter function that returns the limit of the copula derivative, with respect to θ, as θ goes to a, where a is such that $C_a(u, v) = uv$. Readily available options are `dCtheta_frank`, `dCtheta_amh`, `dCtheta_fgm` and `dCtheta_gauss`. Default is `dCtheta_gauss`.
`theta.lower`	the lower bound for θ. Default is -1.
`theta.upper`	the upper bound for θ. Default is 1.
`optim.method`	a character string specifying the optimization method. For all available options see `optim`. Default is 'Brent'. See *fitCopula* for more details.
`method`	a character string specifying the copula parameter estimator used. This can be one of: 'mpl', 'itau', 'irho', 'itau.mpl' or 'ml'. See *fitCopula* for details. Default is 'mpl'.
`s`	integer. The smallest lag h considered in the estimation. Default is 1.
`m`	integer. The largest lag h considered in the estimation. Default is 24.
`theta.start`	starting value for the parameter optimization via `optim`.
`empirical`	logical. If `TRUE`, the sample estimators for the density and quantile functions are considered. Otherwise, the gaussian density and quantile functions are used. Default is `TRUE`

r	the exponent used in the Minkowski distance used to calculate $\hat{d}$. Default is 2, the Euclidean distance.
a	the value of θ such that $\lim_{\theta \to a} C_\theta(u, v) = uv$, is the product (or independence) copula. Default is 0, which is the common value for the available copulas, namely, `frank`, `amh`, `fgm` and `gauss`.
d.interval	a vector of size 2 giving the lower and upper bound for the long memory parameter d. Default is `c(-0.5,0.5)`.

Value

$\hat{d}$, the estimated value of d.

Examples

```
#-------------------------
# ARFIMA(0,d,0) process
#-------------------------
trunc <- 50000
n = 1000
cks <- arfima.coefs(d = 0.25, trunc = trunc)
eps <- rnorm(trunc+n)
x <- sapply(1:n, function(t) sum(cks*rev(eps[t:(t+trunc)])))

#----------------------
# Original time series
#----------------------
# For Frank copula, -Inf < theta < Inf. However, "Brent" requires
# finite lower and upper bounds so we use c(-100, 100) here
d_frank <- d.fit(xt = x, copula = frank, dCdtheta = dCtheta_frank,
                 theta.lower = -100, theta.upper = 100)
d_amh <- d.fit(xt = x, copula = amh, dCdtheta = dCtheta_amh,
                 theta.lower = -1, theta.upper = 1)
d_fgm <- d.fit(xt = x, copula = fgm, dCdtheta = dCtheta_fgm,
                 theta.lower = -1, theta.upper = 1)
d_gauss <- d.fit(xt = x, copula = gauss, dCdtheta = dCtheta_gauss,
                 theta.lower = -1, theta.upper = 1)

c(FRANK = d_frank, AMH = d_amh, FGM = d_fgm, GAUSS = d_gauss)

#---------------------------
# Adding some missing values
#---------------------------
index <- sample(1:n, size = round(0.2*n))
xt <- x
xt[index] <- NA

d_frank_m <- d.fit(xt = xt, copula = frank,
                   dCdtheta = dCtheta_frank,
                   theta.lower = -100, theta.upper = 100)
d_amh_m <- d.fit(xt = xt, copula = amh, dCdtheta = dCtheta_amh,
                 theta.lower = -1, theta.upper = 1)
d_fgm_m <- d.fit(xt = xt, copula = fgm, dCdtheta = dCtheta_fgm,
```

```
                  theta.lower = -1, theta.upper = 1)
d_gauss_m <- d.fit(xt = xt, copula = gauss,
                   dCdtheta = dCtheta_gauss,
                   theta.lower = -1, theta.upper = 1)

data.frame(
  series = c("Complete", "Missing"),
  FRANK = c(d_frank, d_frank_m),
  AMH = c(d_amh, d_amh_m),
  FGM = c(d_fgm, d_fgm_m),
  GAUSS = c(d_gauss, d_gauss_m))

#-------------------------
# ARFIMA(1,d,1) process
#-------------------------
# For a faster algorithm to generate ARFIMA processes,
# see the package "arfima"
trunc <- 50000
cks <- arfima.coefs(d = 0.35, trunc = trunc, ar = -0.2, ma = 0.4)
n = 1000
eps <- rnorm(trunc+n)
x <- sapply(1:n, function(t) sum(cks*rev(eps[t:(t+trunc)])))

#----------------------
# Original time series
#----------------------
# For Frank copula, -Inf < theta < Inf. However, "Brent" requires
# finite lower and upper bounds so we use c(-100, 100) here
d_frank <- d.fit(xt = x, copula = frank, dCdtheta = dCtheta_frank,
                 theta.lower = -100, theta.upper = 100)
d_amh <- d.fit(xt = x, copula = amh, dCdtheta = dCtheta_amh,
                 theta.lower = -1, theta.upper = 1)
d_fgm <- d.fit(xt = x, copula = fgm, dCdtheta = dCtheta_fgm,
                 theta.lower = -1, theta.upper = 1)
d_gauss <- d.fit(xt = x, copula = gauss, dCdtheta = dCtheta_gauss,
                 theta.lower = -1, theta.upper = 1)

c(FRANK = d_frank, AMH = d_amh, FGM = d_fgm, GAUSS = d_gauss)

#---------------------------
# Adding some missing values
#---------------------------
n = 1000
index <- sample(1:n, size = round(0.2*n))
xt <- x
xt[index] <- NA

d_frank_m <- d.fit(xt = xt, copula = frank,
                   dCdtheta = dCtheta_frank,
                   theta.lower = -100, theta.upper = 100)
d_amh_m <- d.fit(xt = xt, copula = amh, dCdtheta = dCtheta_amh,
                 theta.lower = -1, theta.upper = 1)
d_fgm_m <- d.fit(xt = xt, copula = fgm, dCdtheta = dCtheta_fgm,
```

```
                  theta.lower = -1, theta.upper = 1)
d_gauss_m <- d.fit(xt = xt, copula = gauss,
                   dCdtheta = dCtheta_gauss,
                   theta.lower = -1, theta.upper = 1)

data.frame(
  series = c("Complete", "Missing"),
  FRANK = c(d_frank, d_frank_m),
  AMH = c(d_amh, d_amh_m),
  FGM = c(d_fgm, d_fgm_m),
  GAUSS = c(d_gauss, d_gauss_m))
```

---

k1fun *Constant K1*

Description

Calculates an estimate for the constant K_1 given by

$$K_1 = \int_0^1 \int_0^1 \frac{1}{l_0(u) l_n(v)} \lim_{\theta \to a} \frac{\partial C_\theta(u,v)}{\partial \theta} \, dudv,$$

where $l_m(x) := F_m'\big(F_m^{(-1)}(x)\big)$, a is such that $C_a(u,v) = uv$ (the product copula), and $\{F_n\}_{n \geq 0}$ is a sequence of absolutely continuous distribution functions.

Usage

```
k1fun(dCdtheta, fun, data, empirical, mean = 0, sd = 1)
```

Arguments

dCdtheta	a function providing the limit as $\theta \to a$ of the copula derivative with respect to θ. For the readily available copulas, namely, frank, amh, fgm and gauss, $a = 0$.
fun	optionally, a function providing an estimator for the probability density function.
data	the observed time series. Only used to obtain the quantile function when empirical = TRUE.
empirical	logical. If TRUE, the sample estimators for the density and quantile functions are considered. Otherwise, the gaussian density and quantile functions are used instead.
mean	the mean of the gaussian distribution. Only used if empirical = FALSE
sd	the standard deviation of the gaussian distribution. Only used if empirical = FALSE

Details

Here F' and $F^{(-1)}$ are replaced by sample estimators for these functions or the gaussian density and quantile functions are used, depending on the context.

The function kdens is used as sample estimator of F' and quantile is the sample estimator of $F^{(-1)}$.

Value

The value of K_1.

Examples

```
trunc = 50000
cks <- arfima.coefs(d = 0.25, trunc = trunc)
eps <- rnorm(trunc+1000)
x <- sapply(1:1000, function(t) sum(cks*rev(eps[t:(t+trunc)])))

# kernel density function
dfun <- kdens(x)

# calculating K1 using four copulas and empirical estimates for F' and F^{(-1)}
K1_frank_e <- k1fun(dCdtheta = dCtheta_frank, fun = dfun,
                    data = x, empirical = TRUE)
K1_amh_e <- k1fun(dCdtheta = dCtheta_amh, fun = dfun,
                    data = x, empirical = TRUE)
K1_fgm_e <- k1fun(dCdtheta = dCtheta_fgm, fun = dfun,
                    data = x, empirical = TRUE)
K1_gauss_e <- k1fun(dCdtheta = dCtheta_gauss, fun = dfun,
                    data = x, empirical = TRUE)

# calculating K1 using four copulas and gaussian marginals
K1_frank_g <- k1fun(dCdtheta = dCtheta_frank, fun = NULL, data = NULL,
                     empirical = FALSE, mean = mean(x), sd = sd(x))
K1_amh_g <- k1fun(dCdtheta = dCtheta_amh, fun = NULL, data = NULL,
                     empirical = FALSE, mean = mean(x), sd = sd(x))
K1_fgm_g <- k1fun(dCdtheta = dCtheta_fgm, fun = NULL, data = NULL,
                     empirical = FALSE, mean = mean(x), sd = sd(x))
K1_gauss_g <- k1fun(dCdtheta = dCtheta_gauss, fun = NULL, data = NULL,
                     empirical = FALSE, mean = mean(x), sd = sd(x))

# comparing results
 data.frame(MARGINAL = c("Empirical", "Gaussian"),
            FRANK = c(K1_frank_e, K1_frank_g),
            AMH = c(K1_amh_e,  K1_amh_g),
            FGM = c(K1_fgm_e, K1_fgm_g),
            GAUSS = c(K1_gauss_e, K1_gauss_g))
```

kdens *Kernel density estimator*

Description

The probability density function F' is estimated using a kernel density approach. More specifically, first $y_i = \hat{f}(x_i^*)$ is estimated using $T = 512$ (default for the function `density`) equally spaced points x_i^*, $1 \leq i \leq T$, in the interval $[x_{(1)} - 3b, x_{(n)} + 3b]$, where b is the bandwidth for the Gaussian kernel density estimator, chosen by applying the Silverman's rule of thumb (the default procedure in `density`). A cubic spline interpolation (the default method for `spline`) is then applied to the pairs $\{(x_i^*, y_i)\}_{i=1}^T$ to obtain $\hat{F}_n'(x)$ for all $x \in [x_{(1)} - 3b, x_{(n)} + 3b]$.

Usage

```
kdens(x)
```

Arguments

x	the data from which the estimate is to be computed.

Value

a function that approximates the probability density function.

Examples

```
# creating a time series
trunc = 50000
cks <- arfima.coefs(d = 0.25, trunc = trunc)
eps <- rnorm(trunc+1000)
x <- sapply(1:1000, function(t) sum(cks*rev(eps[t:(t+trunc)])))

# kernel density function
dfun <- kdens(x)

# plot
curve(dfun, from = min(x), to = max(x))
```

PPMiss.copulas *Copula functions and the corresponding derivative limit.*

Description

Implemented copulas and the corresponding derivative limit, as $\theta \to a$, where a is such that $C_a(u, v) = uv$. An estimate for θ is obtained based on the copula function used. and the derivatives are used to obtain an estimate for K_1. The functions 'frank', 'amh', 'fgm' and 'gauss' are shortcuts for `copula::frankCopula()`, `copula::amhCopula()`, `copula::fgmCopula()` and `copula::normalCopula()` from package 'copula', respectively.

Usage

```
frank

dCtheta_frank(u, v)

amh

dCtheta_amh(u, v)

fgm

dCtheta_fgm(u, v)

gauss

dCtheta_gauss(u, v)
```

Arguments

u	a real number between 0 and 1.
v	a real number between 0 and 1.

Format

An object of class `frankCopula` of length 1.

An object of class `amhCopula` of length 1.

An object of class `fgmCopula` of length 1.

An object of class `normalCopula` of length 1.

Details

The constant K_1 is given by

$$K_1 = \int_0^1 \int_0^1 \frac{1}{l_0(u) l_n(v)} \lim_{\theta \to a} \frac{\partial C_\theta(u, v)}{\partial \theta} \, du dv,$$

where $I = [0, 1]$, $l_m(x) := F_m'\big(F_m^{(-1)}(x)\big)$ and $\{F_n\}_{n \geq 0}$ is a sequence of absolutely continuous distribution functions

Value

Archimedean copula objects of class 'frankCopula', 'amhCopula' or a Farlie-Gumbel-Morgenstern copula object of class 'fgmCopula' or an elliptical copula object of class 'normalCopula'. For details, see `archmCopula`, `fgmCopula` and `ellipCopula`.

The derivative functions return the limit, as $\theta \to 0$, of the derivative with respect to θ, corresponding to the copula functions.

Index

∗ **datasets**
PPMiss.copulas, 8

amh *(*PPMiss.copulas*)*, 8
archmCopula, *10*
arfima.coefs, 2

copula::amhCopula(), *9*
copula::fgmCopula(), *9*
copula::frankCopula(), *9*
copula::normalCopula(), *9*

d.fit, 3
dCtheta_amh *(*PPMiss.copulas*)*, 8
dCtheta_fgm *(*PPMiss.copulas*)*, 8
dCtheta_frank *(*PPMiss.copulas*)*, 8
dCtheta_gauss *(*PPMiss.copulas*)*, 8
density, *8*

ellipCopula, *10*

fgm *(*PPMiss.copulas*)*, 8
fgmCopula, *10*
fitCopula, *3*
frank *(*PPMiss.copulas*)*, 8

gauss *(*PPMiss.copulas*)*, 8

k1fun, 6
kdens, *7*, 8

optim, *3*

PPMiss.copulas, 8

quantile, *7*

spline, *8*