Package: PPMiss (via r-universe)

November 22, 2026

Type Package

Date 2026-11-21

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

Version 0.2.0

Depends R (>= 4.0.0)

Imports copula, pracma, stats, zoo

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Description Implements the copula-based estimator for univariate long-range dependent processes introduced in Pumi et al. (2024) <doi:10.1007/s00362-023-01418-z>. The estimator handles missing data and gaps directly without requiring prior imputation, and has been shown to perform well even when up to 70% of observations are missing (as reported in Pumi et al., 2023 <doi:10.48550/arXiv.2303.04754>), outperforming several commonly applied estimators.

URL https://github.com/tsprass/PPMiss

BugReports https://github.com/tsprass/PPMiss/issues

License GPL (>= 3)

Encoding UTF-8

NeedsCompilation yes

RoxygenNote 7.3.3

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

Maintainer Taiane Schaedler Prass <taianeprass@gmail.com>

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

Date/Publication 2026-11-22 22:34:46 UTC

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

RemoteRef HEAD

RemoteSha 830c9506e0d99df9e0a5a0c7c7773773d1b88593

Contents

arfima.coefs 2
d.fit 3
k1fun 6
kdens 8
PPMiss.copulas 9
theta.fit 10

Index **13**

`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 polynomial. Default is `NULL`.
`ma`	the coefficients of the moving average polynomial. 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 with (X_t, X_{t+h}) and $\hat{\theta}_h$ be an estimate of θ_h based on pseudo-observations. Given a set of positive integer lags $\mathcal{H}$ (by default $\mathcal{H} = \{s, s+1, \ldots, m\}$), the long memory parameter d is estimated by

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

where $\hat{\sigma}^2$ denotes the sample variance of the observed series and $\hat{K}_1$ is the estimated constant returned by `k1fun`.

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),
  lags = NULL,
  na.action = c("omit", "approx"),
  thetas = NULL,
  full.output = FALSE
)
```

Arguments

`xt`	a numeric vector with the observed time series. Missing observations (`NA`) are allowed.
`copula`	an object of class 'copula'. Readily available options are `frank`, `amh`, `fgm`, and `gauss`. Other copulas can be used, provided the user supplies 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 when lags is NULL. Default is 1.
m	integer. The largest lag h considered when lags is NULL. Default is 24.
theta.start	starting value for the parameter optimization via optim. Default is 0.1.
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 numeric vector of length 2 giving the lower and upper bounds for the long memory parameter d. Default is c(-0.5, 0.5).
lags	optional integer vector specifying arbitrary lags h to be used in the estimation (e.g., c(1, 2, 5, 10)). When provided, lags takes precedence over s and m. Default is NULL, which uses s:m (or infers lags from names(thetas) when thetas is provided and neither s nor m is specified).
na.action	character string controlling how lags with no complete pairs or failed copula fits are handled. If 'omit' (default), failed lags are dropped from the objective sum. If 'approx', missing copula estimates are linearly interpolated over lags via na.approx (with boundary values carried from the nearest valid lag) and clipped to [theta.lower, theta.upper].
thetas	optional numeric vector of precomputed copula parameter estimates $\hat{\theta}_h$ (such as returned by theta.fit). When supplied, the copula estimation step is skipped. Default is NULL.
full.output	logical. If FALSE (default), returns only $\hat{d}$. If TRUE, returns a list containing d, thetas, lags, and k1.

Value

If full.output = FALSE, a numeric scalar giving $\hat{d}$. If full.output = TRUE, a list with components:

d the estimated long memory parameter $\hat{d}$.

thetas named numeric vector of estimated copula parameters $\hat{\theta}_h$.

lags integer vector of lags considered.

k1 the estimated constant $\hat{K}_1$ returned by k1fun.

References

Pumi, G., Prass, T. S., & Lopes, S. R. C. (2024). A novel copula-based approach for parametric estimation of univariate time series through its covariance decay. *Statistical Papers*, 65(2), 1041–1063. doi:10.1007/s0036202301418z

Pumi, G., Choque Ulloa, G., & Prass, T. S. (2023). Estimation of long-range dependent models with missing data: to impute or not to impute? *arXiv preprint*. doi:10.48550/arXiv.2303.04754

See Also

theta.fit, k1fun, PPMiss.copulas

Examples

```
# Helper to fit all four built-in copulas to a series
fit_all <- function(series) {
  c(
    FRANK = d.fit(
      xt = series, copula = frank, dCdtheta = dCtheta_frank,
      theta.lower = -100, theta.upper = 100
    ),
    AMH = d.fit(xt = series, copula = amh, dCdtheta = dCtheta_amh),
    FGM = d.fit(xt = series, copula = fgm, dCdtheta = dCtheta_fgm),
    GAUSS = d.fit(xt = series, copula = gauss, dCdtheta = dCtheta_gauss)
  )
}

#-------------------------
# ARFIMA(0, d, 0) process
#-------------------------
set.seed(123)
trunc <- 5000
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)])))

# Complete series vs. 20% missing observations
xt <- x
xt[sample(1:n, size = round(0.2 * n))] <- NA

rbind(Complete = fit_all(x), Missing = fit_all(xt))

# Estimation using arbitrary lags and returning full output
res <- d.fit(xt = xt, lags = c(1, 2, 5, 10, 20), full.output = TRUE)
res$d
res$thetas
```

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

xt <- x
xt[sample(1:n, size = round(0.2 * n))] <- NA

rbind(Complete = fit_all(x), Missing = fit_all(xt))
```

---

k1fun *Constant K1*

Description

Calculates an estimate for the constant K_1 given by

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

where $l(x) := F'\big(F^{(-1)}(x)\big)$, F is the marginal distribution function (with density F' and quantile function $F^{(-1)}$), and a is such that $C_a(u,v) = uv$ (the product copula).

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

```
set.seed(123)
trunc <- 500
n <- 200
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)])))

# kernel density function
dfun <- kdens(x)

# calculating K1 using four copulas and empirical estimates for F' & 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 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
set.seed(123)
trunc <- 500
n <- 200
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)])))

# kernel density function
dfun <- kdens(x)

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

`PPMiss.copulas` *Copula objects and the corresponding derivative limits*

Description

Implemented copulas and the corresponding derivative limits as $\theta \to a$, where a is such that $C_a(u, v) = uv$. An estimate for θ is obtained based on the copula object used, and the derivatives are used to obtain an estimate for K_1. The objects '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

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

where $l(x) := F'\big(F^{(-1)}(x)\big)$ and F is the marginal absolutely continuous distribution function (with density F' and quantile function $F^{(-1)}$).

Value

Archimedean copula objects of class 'frankCopula' or 'amhCopula', 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 each copula.

`theta.fit` *Copula parameter estimation across lagged pairs*

Description

Estimates the bivariate copula parameter θ_h associated with lagged pairs (X_t, X_{t+h}) for a user-specified set of lags h, based on empirical or Gaussian pseudo-observations.

Usage

```
theta.fit(
  xt,
  copula = gauss,
  theta.lower = -1,
  theta.upper = 1,
  optim.method = "Brent",
  method = "mpl",
  s = 1,
  m = 24,
  theta.start = 0.1,
  empirical = TRUE,
```

```
  lags = NULL,
  na.action = c("omit", "approx")
)
```

Arguments

xt	a numeric vector with the observed time series. Missing observations (NA) are allowed.
copula	an object of class 'copula'. Readily available options are frank, amh, fgm, and gauss. Default is 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 when lags is NULL. Default is 1.
m	integer. The largest lag h considered when lags is NULL. Default is 24.
theta.start	starting value for the parameter optimization via optim. Default is 0.1.
empirical	logical. If TRUE, empirical pseudo-observations are computed via pobs. Otherwise, Gaussian marginals are used via pnorm. Default is TRUE.
lags	optional integer vector specifying arbitrary lags h to be estimated (e.g., c(1, 2, 5, 10)). When provided, lags takes precedence over s and m. Default is NULL, which uses s:m.
na.action	character string controlling how lags with no complete pairs or failed copula fits are handled. If 'omit' (default), failed lags are returned as NA (and omitted from the objective function in d.fit). If 'approx', missing estimates are linearly interpolated over lags via na.approx (with boundary values carried from the nearest valid lag) and clipped to [theta.lower, theta.upper].

Value

A named numeric vector with the estimated copula parameters $\hat{\theta}_h$, named by the corresponding lags.

References

Pumi, G., Prass, T. S., & Lopes, S. R. C. (2024). A novel copula-based approach for parametric estimation of univariate time series through its covariance decay. *Statistical Papers*, 65(2), 1041–1063. doi:10.1007/s0036202301418z

See Also

d.fit, PPMiss.copulas

Examples

```
set.seed(123)
trunc <- 500
n <- 200
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)])))

# Estimate theta on arbitrary lags
th <- theta.fit(xt = x, lags = c(1, 2, 5, 10))
th

# Reuse the estimated thetas in d.fit()
d.fit(xt = x, thetas = th)
```

Index

∗ **datasets**
PPMiss.copulas, 9

amh *(*PPMiss.copulas*)*, 9
archmCopula, *10*
arfima.coefs, 2

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

d.fit, 3, *11*
dCtheta_amh *(*PPMiss.copulas*)*, 9
dCtheta_fgm *(*PPMiss.copulas*)*, 9
dCtheta_frank *(*PPMiss.copulas*)*, 9
dCtheta_gauss *(*PPMiss.copulas*)*, 9
density, *8*

ellipCopula, *10*

fgm *(*PPMiss.copulas*)*, 9
fgmCopula, *10*
fitCopula, *4*, *11*
frank *(*PPMiss.copulas*)*, 9

gauss *(*PPMiss.copulas*)*, 9

k1fun, *3*, *5*, 6
kdens, *7*, 8

na.approx, *4*, *11*

optim, *4*, *11*

pnorm, *11*
pobs, *11*
PPMiss.copulas, *5*, 9, *11*

quantile, *7*

spline, *8*

theta.fit, *4, 5*, 10