Package: tscopula (via r-universe)

October 8, 2024

Type Package

Title Time Series Copula Models

Version 0.4.2

Date 2024-10-02

Maintainer Alexander McNeil <alexanderjmcneil@gmail.com>

Description Functions for the analysis of time series using copula models. The package is based on methodology described in the following references. McNeil, A.J. (2021) <doi:10.3390/risks9010014>, Bladt, M., & McNeil, A.J. (2021) <doi:10.1016/j.ecosta.2021.07.004>, Bladt, M., & McNeil, A.J. (2022) <doi:10.1515/demo-2022-0105>.

Depends R (>= 3.5.0)

License GPL-3

LazyData true

RoxygenNote 7.3.1

Encoding UTF-8

Imports methods, stats, graphics, utils, stats4, zoo, xts, FKF, ltsa, rvinecopulib, arfima, Matrix, polynom, kdensity

Collate 'basic_objects.R' 'armacopula.R' 'sarmacopula.R' 'dvinecopula.R' 'dvinecopula2.R' 'dvinecopula3.R' 'dvinecopulavt.R' 'fitting_basic.R' 'margins.R' 'full_models.R' 'vtransforms.R' 'fitting_vtscopula.R' 'helper_vtarma.R' 'data.R'

Suggests knitr, rmarkdown

VignetteBuilder knitr

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

RemoteUrl https://github.com/ajmcneil/tscopula

RemoteRef HEAD

RemoteSha 1008de9d1b146c97033022e4e9948f8044c1aaca

Contents

acf2pacf *Compute partial autocorrelations from autocorrelations*

Description

Compute partial autocorrelations from autocorrelations

Usage

```
acf2pacf(rho)
```

Arguments

rho	vector of autocorrelation values (excluding 1).

Value

A vector of partial autocorrelation values with same length as `rho`.

Examples

```
rho <- ARMAacf(ar = -0.9, ma = 0.8, lag.max = 50)[-1]
alpha <- acf2pacf(rho)
```

AICc *Akaike Corrected Information Criterion*

Description

Akaike Corrected Information Criterion

Usage

```
AICc(object, ...)
```

Arguments

object	a fitted model object for which there exists a logLik method to extract the corresponding log-likelihood.
...	optionally more fitted model objects.

Value

If just one object is provided, a numeric value with the corresponding AICC value.

If multiple objects are provided, a data.frame with rows corresponding to the objects and columns representing the number of parameters in the model (df) and the AICC.

---

arma2dvine *Transform an armacopula into a dvinecopula or dvinecopula2 object*

Description

Transform an armacopula into a dvinecopula or dvinecopula2 object

Usage

```
arma2dvine(object)
```

Arguments

`object`	an object of class armacopula.

Value

An object of class dvinecopula (for AR copulas) or class dvinecopula2 (for MA or ARMA copulas).

Examples

```
arma2dvine(armacopula(list(ar = 0.5, ma = 0.4)))
```

---

armacopula *Constructor function for ARMA copula process*

Description

Constructor function for ARMA copula process

Usage

```
armacopula(pars = list(ar = 0, ma = 0))
```

Arguments

`pars`	list consisting of vector of AR parameters named ‘ar‘ and vector of MA parameters named ‘ma‘.

Value

An object of class armacopula.

Examples

```
armacopula(list(ar = 0.5, ma = 0.4))
```

armacopula-class *ARMA copula processes*

Description

Class of objects for ARMA copula processes.

Usage

```
## S4 method for signature 'armacopula'
coef(object)

## S4 method for signature 'armacopula'
show(object)

## S4 method for signature 'armacopula'
sim(object, n = 1000)

## S4 method for signature 'armacopula'
kendall(object, lagmax = 20)

## S4 method for signature 'armacopula'
predict(object, data, x, type = ”df”)
```

Arguments

`object` an object of the class.

`n` length of realization.

`lagmax` maximum value of lag.

`data` vector of past data values.

`x` vector of arguments of prediction function.

`type` type of prediction function ("df" for density, "qf" for quantile function or "dens" for density).

Methods (by generic)

- `coef(armacopula)`: Coef method for ARMA copula class
- `show(armacopula)`: Show method for ARMA copula process
- `sim(armacopula)`: Simulation method for armacopula class
- `kendall(armacopula)`: Calculate Kendall's tau values for armacopula model
- `predict(armacopula)`: Prediction method for armacopula class

Slots

`name` name of ARMA copula process.

`modelspec` vector containing number of AR and MA parameters.

`pars` list consisting of vector of AR parameters named ‘ar‘ and vector of MA parameters named ‘ma‘.

Examples

```
sim(armacopula(list(ar = c(0.5, 0.4), ma = -0.8)), n = 1000)
mod <- armacopula(list(ar = 0.95, ma = -0.85))
kendall(mod)
```

`armafit2dvine` *Transform a fitted armacopula into a fitted dvinecopula or dvinecopula2 object*

Description

Transform a fitted armacopula into a fitted dvinecopula or dvinecopula2 object

Usage

```
armafit2dvine(object)
```

Arguments

`object`	an object of class tscopulafit in which the copula is of class armacopula.

Value

An object of class tscopulafit in which the copula is a dvinecopula (for fitted AR copulas) or class dvinecopula2 (for fitted MA or ARMA copulas).

bitcoin *Bitcoin price data 2016-19*

Description

Time series of Bitcoin closing prices from 31 December 2015 to 31 December 2019 (1044 values). This permits the calculation of 4 calendar years of returns.

Usage

```
data(bitcoin)
```

Format

An object of class "xts".

Examples

```
data(bitcoin)
plot(bitcoin)
X <- (diff(log(bitcoin))[-1]) * 100
plot(X)
```

coerce,tscopula,tscm-method *Convert tscopula object to tscm object*

Description

Convert tscopula object to tscm object

Usage

```
## S4 method for signature 'tscopula,tscm'
coerce(from, to = "tsc", strict = TRUE)
```

Arguments

from	a tscopula object.
to	a tscm object.
strict	logical variable stating whether strict coercion should be enforced.

Value

A tscm object.

coerce,tscopulafit,tscmfit-method *Convert tscopulafit object to be tscmfit object*

Description

Convert tscopulafit object to be tscmfit object

Usage

```
## S4 method for signature 'tscopulafit,tscmfit'
coerce(from, to = "tscmfit", strict = TRUE)
```

Arguments

`from`	a tscopulafit object.
`to`	a tscmfit object.
`strict`	logical variable stating whether strict coercion should be enforced.

Value

A tscmfit object.

cpi *CPI inflation data 1959-2020*

Description

Time series of US quarterly CPI (consumer price index) data Q4 1959 to Q4 2020 (245 values) for studying inflation. These data were sourced from the OECD webpage and represent the total 'perspective' on inflation, including food and energy. They have been based to have a value of 100 in 2015.

Usage

```
data(cpi)
```

Format

An object of class `"xts"`.

Examples

```
data(cpi)
plot(cpi)
X <- (diff(log(cpi))[-1]) * 100
plot(X)
```

dmarg *Compute density of marginal model*

Description

Compute the density function of the marginal model.

Usage

```
dmarg(x, y, log = FALSE)
```

Arguments

`x` an object of class margin.

`y` vector of values for which density should be computed.

`log` logical variable specifying whether log density should be returned.

Value

A vector of values for the density.

Examples

```
margmod <- margin(”gauss”, pars = c(mu = 0, sigma = 1))
dmarg(margmod, c(-2, 0, 2), log = TRUE)
```

doubleweibull *Double Weibull distribution*

Description

Double Weibull distribution

Usage

```
ddoubleweibull(x, mu = 0.05, shape = 1, scale = 1, log = FALSE)

pdoubleweibull(q, mu = 0.05, shape = 1, scale = 1)

qdoubleweibull(p, mu = 0.05, shape = 1, scale = 1)

rdoubleweibull(n, mu = 0.05, shape = 1, scale = 1)
```

Arguments

`x`	vector of values.
`mu`	location parameter.
`shape`	shape parameter.
`scale`	scale parameter.
`log`	flag for log density.
`q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations.

Value

A vector of density, distribution function, quantile or random values.

`dvinecopula` *Constructor function for dvinecopula process*

Description

This function sets up a stationary d-vine process of finite order where the elements of the (finite-length) copula sequence may be any copulas that can be implemented using `bicop_dist` in the `rvinecopulib` package.

Usage

```
dvinecopula(family = "indep", pars = list(NULL), rotation = 0)
```

Arguments

`family`	a vector of family names
`pars`	a list containing the parameters of the copula at each lag
`rotation`	a vector of rotations

Details

Copulas may also be rotated through 90, 180 and 270 degrees. If the same `family` or same `rotation` is to be used at every lag, these arguments may be scalars. The `pars` argument must be a list with the same length as the copula sequence.

If a t copula is included, the correlation parameter precedes the degrees of freedom in the parameter vector. This copula should be referred to as "t" rather than "Student".

Value

An object of class dvinecopula.

Examples

```
dvinecopula(family = c("joe", "gauss", "t"), pars = list(3, .5, c(0.4, 4)), rotation = c(180, 0, 0))
```

---

dvinecopula-class *D-vine copula processes*

Description

Class of objects for d-vine copula processes.

Usage

```
## S4 method for signature 'dvinecopula'
coef(object)

## S4 method for signature 'dvinecopula'
show(object)

## S4 method for signature 'dvinecopula'
sim(object, n = 1000, innov = NA, start = NA)

## S4 method for signature 'dvinecopula'
predict(object, data, x, type = "df")

## S4 method for signature 'dvinecopula'
kendall(object, lagmax = 20)
```

Arguments

`object`	an object of the class.
`n`	length of realization.
`innov`	vector of innovations of length n.
`start`	vector of start values with length equal to order of process.
`data`	vector of past data values.
`x`	vector of arguments of prediction function.
`type`	type of prediction function ("df" for density, "qf" for quantile function or "dens" for density).
`lagmax`	maximum value of lag.

Methods (by generic)

- `coef(dvinecopula)`: Coef method for dvinecopula class
- `show(dvinecopula)`: Show method for dvinecopula class
- `sim(dvinecopula)`: Simulation method for dvinecopula class
- `predict(dvinecopula)`: Prediction method for dvinecopula class
- `kendall(dvinecopula)`: Calculate Kendall's tau values for pair copulas in d-vine copula

Slots

`name` name of the d-vine copula process.

`modelspec` list containing the family, number of parameters and rotations

`pars` list comprising of the parameters.

Examples

```
sim(dvinecopula("gauss", 0.5))
mixmod <- dvinecopula(family = c("gumbel", "gauss"), pars = list(1.5, -0.6))
kendall(mixmod)
```

`dvinecopula2` *Constructor function for dvinecopula2 process*

Description

This function sets up a stationary d-vine process of finite or infinite order based on a single copula family from a subset of those that can be implemented using `bicop_dist` in the `rvinecopulib` package.

Usage

```
dvinecopula2(
  family = "gauss",
  rotation = 0,
  kpacf = "kpacf_arma",
  pars = list(ar = 0.1, ma = 0.1),
  tautol = 1e-04,
  maxlag = Inf,
  negtau = "none"
)
```

Arguments

`family`	family name
`rotation`	a scalar specifying the rotation (default is 0)
`kpacf`	a character string giving the name of the Kendall pacf
`pars`	a list containing the parameters of the model
`tautol`	scalar value at which kpacf is truncated
`maxlag`	a scalar which can be used to force a given value for maximum lag
`negtau`	a character string specifying the treatment of negative Kendall's tau values

Details

The copula family may be any one-parameter family or the t copula family. The basic copula from which the sequence is built may be rotated through 180 degrees using the `rotation` argument; the default is no rotation (0 degrees).

The copulas are parameterized using the Kendall partial autocorrelation function (kpacf) specified by the `kpacf` argument. The default choice is the kpacf of a standard ARMA process which is implemented in the function `kpacf_arma`. The parameters of the kpacf should be set as a list using the `pars` argument; the required parameters should usually be clear from the documentation of the chosen kpacf function and must be correctly named.

If the kpacf takes a negative value at any lag and the standard copula is unable to model a negative dependency (e.g. Clayton, Gumbel, Joe and their 180 degree rotations) then one of four different treatments may be specified using the `negtau` parameter: "gauss" substitutes a Gaussian copula at that lag; "frank" substitutes a Frank copula; "right" and "left" rotate the copula through 90 degrees in a clockwise or anto-clockwise direction respectively.

In practice, the sequence of copulas will be truncated at the last copula for which the kpacf exceeds `tautol`. The `maxlag` parameter is typically used to force the truncation to take place at a lower lag (to increase speed). This can also be achieved by increasing the value of `tautol`.

If the t copula is chosen by setting `family` equal to "t", the list of parameters needs to be augmented with a component named "df" which is the degrees of freedom. In this case it makes sense to set `maxlag` to be a finite number to avoid models with tail dependencies at arbitrary lags which are not ergodic. The class dvinecopula3 is more suitable for working with t copulas with different degrees of freedom at different lags.

Value

An object of class dvinecopula2.

Examples

```
dvinecopula2(family = "joe", kpacf = "kpacf_arma",
pars = list(ar = 0.95, ma = -0.85), maxlag = 30)
```

---

dvinecopula2-class *D-vine copula processes of type 2*

Description

Class of objects for d-vine copula processes. See dvinecopula2 for more details.

Usage

```
## S4 method for signature 'dvinecopula2'
coef(object)

## S4 method for signature 'dvinecopula2'
show(object)
```

```
## S4 method for signature 'dvinecopula2'
sim(object, n = 1000)

## S4 method for signature 'dvinecopula2'
predict(object, data, x, type = "df")

## S4 method for signature 'dvinecopula2'
kendall(object, lagmax = 20)
```

Arguments

`object`	an object of the class.
`n`	length of realization.
`data`	vector of past data values.
`x`	vector of arguments of prediction function.
`type`	type of prediction function ("df" for density, "qf" for quantile function or "dens" for density).
`lagmax`	maximum value of lag.

Methods (by generic)

- `coef(dvinecopula2)`: Coef Method for dvinecopula2 class
- `show(dvinecopula2)`: Show method for dvinecopula2 class
- `sim(dvinecopula2)`: Simulation method for dvinecopula2 class
- `predict(dvinecopula2)`: Prediction method for dvinecopula2 class
- `kendall(dvinecopula2)`: Calculate Kendall's tau values for pair copulas in type 2 d-vine copula

Slots

`name` name of the d-vine copula process.

`modelspec` list containing the family, rotation, and name of KPACF

`pars` list comprising of the parameters.

Examples

```
copmod <- dvinecopula2(family = "joe", kpacf = "kpacf_arma",
pars = list(ar = 0.95, ma = -0.85), maxlag = 30)
kendall(copmod)
```

dvinecopula3 *Constructor function for dvinecopula3 process*

Description

This function sets up a stationary d-vine process of finite or infinite order based on a sequence of Gaussian copulas with a finite number of non-Gaussian substitutions at specified lags. The substituted families can be Gumbel, Clayton, Joe, Frank, t and BB1 copulas as implemented by the `bicop_dist` in the `rvinecopulib` package. The Gauss copula can be named in the list of substitutions but does not need to be.

Usage

```
dvinecopula3(
  location = 1,
  family = "gumbel",
  posrot = 0,
  negrot = 90,
  kpacf = "kpacf_arma",
  pars = list(ar = 0.1, ma = 0.1),
  auxpar = NA,
  tautol = 1e-04,
  maxlag = Inf
)
```

Arguments

`location`	vector of locations of copula substitutions
`family`	vector of family names for copula substitutions
`posrot`	vector of rotations for substituted families under positive dependence (default is 0)
`negrot`	vector of rotations for substituted families under negative dependence (default is 90)
`kpacf`	a character string giving the name of the Kendall pacf
`pars`	a list containing the parameters of the model
`auxpar`	vector of additional parameters for two-parameter copulas
`tautol`	scalar value at which kpacf is truncated
`maxlag`	a scalar which can be used to force a given value for maximum lag

Details

For the substituted copulas (other than t and Frank) the user must specify the rotation that should be used for positive dependencies (0 or 180) and the rotation that should be used for negative dependencies (90 or 270).

The copulas are parameterized using the Kendall partial autocorrelation function (kpacf) specified by the `kpacf` argument. The default choice is the kpacf of a standard ARMA process which is implemented in the function `kpacf_arma`. The parameters of the kpacf should be set as a list using the `pars` argument; the required parameters should usually be clear from the documentation of the chosen kpacf function and must be correctly named.

In practice, the sequence of copulas will be truncated at the last copula for which the kpacf exceeds `tautol`. The `maxlag` parameter is typically used to force the truncation to take place at a lower lag (to increase speed). This can also be achieved by increasing the value of `tautol`.

If one or more of the substituted copulas are t or BB1 copulas the argument `auxpar` should be used to specify the additional parameters. These are the degree-of-freedom parameter for t and the delta parameter for BB1; the former must be greater or equal 2 and the latter greater or equal 1.

Value

An object of class dvinecopula3.

Examples

```
dvinecopula3(location = c(1,4), family = c("Gumbel", "clayton"),
posrot = c(0, 180), negrot = c(90, 270), kpacf = "kpacf_arma",
pars = list(ar = 0.95, ma = 0.85), maxlag = 20)
```

---

dvinecopula3-class *D-vine copula processes of type 3*

Description

Class of objects for d-vine copula processes. See dvinecopula3 for more details.

Usage

```
## S4 method for signature 'dvinecopula3'
coef(object)

## S4 method for signature 'dvinecopula3'
kendall(object, lagmax = 20)

## S4 method for signature 'dvinecopula3'
show(object)

## S4 method for signature 'dvinecopula3'
sim(object, n = 1000)

## S4 method for signature 'dvinecopula3'
predict(object, data, x, type = "df")
```

Arguments

`object` an object of the class.

`lagmax` maximum value of lag to be considered.

`n` length of realization.

`data` vector of past data values.

`x` vector of arguments of prediction function.

`type` type of prediction function ("df" for density, "qf" for quantile function or "dens" for density).

Methods (by generic)

- `coef(dvinecopula3)`: Coef Method for dvinecopula3 class
- `kendall(dvinecopula3)`: Calculate Kendall's tau values for pair copulas in type 3 d-vine copula
- `show(dvinecopula3)`: Show method for dvinecopula3 class
- `sim(dvinecopula3)`: Simulation method for dvinecopula3 class
- `predict(dvinecopula3)`: Prediction method for dvinecopula2 class

Slots

`name` name of the d-vine copula process.

`modelspec` list containing the family, rotation, and name of KPACF

`pars` list comprising of the parameters.

`dvinecopulavt` *Constructor function for dvinecopulavt process*

Description

This function sets up a stationary d-vine process of finite or infinite order based on a single inverse-v-transformed copula family from a subset of those that can be implemented using `bicop_dist` in the `rvinecopulib` package.

Usage

```
dvinecopulavt(
  family = "joe",
  rotation = 0,
  kpacf = "kpacf_arma",
  pars = list(ar = 0.1, ma = 0),
  vt1 = Vlinear(0.5),
  vt2 = Vlinear(0.5),
  tautol = 1e-04,
  maxlag = Inf,
  V2override = FALSE
)
```

Arguments

`family`	family name
`rotation`	a scalar specifying the rotation (default is 0)
`kpacf`	a character string giving the name of the Kendall pacf
`pars`	a list containing the parameters of the model
`vt1`	first v-transform
`vt2`	second v-transform
`tautol`	scalar value at which kpacf is truncated
`maxlag`	a scalar which can be used to force a given value for maximum lag
`V2override`	logical variable stating whether 2-parameter v-transform should be permitted

Details

The permitted choices of base copula family are currently Joe, Gumbel, Frank or Clayton survival. If Clayton is chosen, the `rotation` argument must be set to 180, while if Joe or Gumbel are chosen, the `rotation` argument must be zero (which is the default); any other options will return an error

The copulas are parameterized using the Kendall partial autocorrelation function (kpacf) of the base copula sequence specified by the `kpacf` argument. The default choice is the kpacf of a standard ARMA process which is implemented in the function `kpacf_arma`. The parameters of the kpacf should be set as a list using the `pars` argument; the required parameters should usually be clear from the documentation of the chosen kpacf function and must be correctly named.

The arguments `vt1` and `vt2` are used to enter two parametric v-transforms which may be created, for example, by `Vlinear` or `V2p`. However, the latter is very slow and the variable `V2override` has to be set to `TRUE` if you want to include 2-parameter v-transforms. While fitting is possible, residual analysis and simulation are almost always prohibitively slow.

For data showing stochastic volatility, we expect positive serial dependencies in the base copula sequence. For this reason, we do not consider models where the kpacf takes negative values.

In practice, the sequence of base copulas will be truncated at the last copula for which the kpacf exceeds `tautol`. The `maxlag` parameter is typically used to force the truncation to take place at a lower lag (to increase speed). This can also be achieved by increasing the value of `tautol`.

Value

An object of class dvinecopulavt.

Examples

```
dvinecopulavt(family = "joe", kpacf = "kpacf_arma",
pars = list(ar = 0.95, ma = -0.85), maxlag = 30)
```

dvinecopulavt-class *D-vine copula processes with v-transforms*

Description

Class of objects for d-vine copula processes. See dvinecopulavt for more details.

Usage

```
## S4 method for signature 'dvinecopulavt'
coef(object)

## S4 method for signature 'dvinecopulavt'
show(object)

## S4 method for signature 'dvinecopulavt'
kendall(object, lagmax = 20)

## S4 method for signature 'dvinecopulavt'
sim(object, n = 1000, forcetrunc = TRUE)

## S4 method for signature 'dvinecopulavt'
predict(object, data, x, type = "df")
```

Arguments

`object`	an object of the class.
`lagmax`	maximum value of lag.
`n`	length of realization.
`forcetrunc`	logical parameter: TRUE truncates the copula sequence at lag 10 to accelerate simulation if copula sequence is longer; FALSE turns this feature off.
`data`	vector of past data values.
`x`	vector of arguments of prediction function.
`type`	type of prediction function ("df" for density, "qf" for quantile function or "dens" for density).

Methods (by generic)

- `coef(dvinecopulavt)`: Coef Method for dvinecopulavt class
- `show(dvinecopulavt)`: Show method for dvinecopulavt class
- `kendall(dvinecopulavt)`: Calculate Kendall's tau values for core pair copulas in d-vine copula model with v-transforms
- `sim(dvinecopulavt)`: Simulation method for dvinecopulavt class
- `predict(dvinecopulavt)`: Prediction method for dvinecopulavt class

Slots

`name` name of the d-vine copula process.

`modelspec` list containing the family, rotation, and name of KPACF

`pars` list comprising of the parameters.

Examples

```
copmod <- dvinecopulavt(family = "joe", kpacf = "kpacf_arma",
pars = list(ar = 0.95, ma = -0.85), maxlag = 30)
kendall(copmod)
```

`edf` *Construct empirical margin*

Description

Construct empirical margin

Usage

```
edf()
```

Value

An object of class margin signifying an empirical distribution function.

`fit` *Generic for estimating time series models*

Description

Methods are available for objects of class tscopulaU, vtscopula, tscopulafit, margin and tscm.

Usage

```
fit(x, y, ...)
```

Arguments

`x`	an object of the model class.
`y`	a vector or time series of data.
`...`	further arguments to be passed on.

Value

An object of the fitted model class.

fit,margin-method *Fit method for margin class*

Description

Fit method for margin class

Usage

```
## S4 method for signature 'margin'
fit(x, y, tsoptions = list(), control = list())
```

Arguments

x	an object of class margin.
y	a vector or time series of data.
tsoptions	list of optional arguments: hessian is logical variable specifying whether Hessian matrix should be returned; start is vector od named starting values
control	list of control parameters to be passed to the optim function.

Value

An object of class marginfit.

Examples

```
margmod <- margin("gauss", pars = c(mu = 0, sigma = 1))
data <- sim(margmod, n = 500)
fit(margmod, data)
```

fit,tscm-method *Fit method for tscm class*

Description

Fit method for tscm class

Usage

```
## S4 method for signature 'tscm'
fit(x, y, tsoptions = list(), control = list(), method = "IFM")
```

Arguments

`x`	an object of class tscm.
`y`	a vector or time series of data.
`tsoptions`	a list of parameters passed to fitting.
`control`	list of control parameters to be passed to the `optim` function.
`method`	character string specifying method.

Value

An object of class tscmfit.

Examples

```
mod <- tscm(dvinecopula(family = "gauss", pars = 0.5), margin("doubleweibull"))
y <- sim(mod)
fit(mod, y)
```

fit,tscopulafit-method

Fit method for tscopulafit class

Description

Fit method for tscopulafit class

Usage

```
## S4 method for signature 'tscopulafit'
fit(x, y, tsoptions = list(), control = list(warn.1d.NelderMead = FALSE))
```

Arguments

`x`	an object of class tscopulafit.
`y`	vector or time series of data to which the copula process is to be fitted.
`tsoptions`	list of options
`control`	list of control parameters to be passed to the `optim` function.

Value

An object of class tscopulafit.

Examples

```
ar1 <- armacopula(list(ar = 0.7))
data <- sim(ar1, 1000)
ar1fit <- fit(fit(ar1, data), sim(ar1, 1000))
```

`fit,tscopulaU-method` *Fit method for tscopulaU class*

Description

Fit method for tscopulaU class

Usage

```
## S4 method for signature 'tscopulaU'
fit(x, y, tsoptions = list(), control = list())
```

Arguments

`x`	an object of class tscopulaU.
`y`	vector or time series of data to which the copula process is to be fitted.
`tsoptions`	list of options
`control`	list of control parameters to be passed to the `optim` function.

Value

An object of class tscopulafit.

Examples

```
data <- sim(armacopula(list(ar = 0.5, ma = 0.4)), n = 1000)
fit(armacopula(list(ar = 0.5, ma = 0.4)), data)
```

`fit,vtscopula-method` *Fit method for vtscopula class*

Description

Fit object of class vtscopula to data using maximum likelihood.

Usage

```
## S4 method for signature 'vtscopula'
fit(
  x,
  y,
  tsoptions = list(),
  control = list(maxit = 2000, warn.1d.NelderMead = FALSE)
)
```

Arguments

x	an object of class vtscopula.
y	a vector or time series of data.
tsoptions	list of optional arguments: hessian is logical variable specifying whether Hessian matrix should be returned; method is choice of optimization method.
control	list of control parameters to be passed to the optim function.

Value

An object of class tscopulafit.

Examples

```
copobject <- armacopula(pars = list(ar = 0.6, ma = 0.2))
vtcop <- vtscopula(copobject, Vtransform = V2p())
y <- sim(vtcop)
fit(vtcop, y)
```

gauss *Gaussian distribution*

Description

Gaussian distribution

Usage

```
dgauss(x, mu = 0, sigma = 1, log = FALSE)

pgauss(q, mu = 0, sigma = 1)

qgauss(p, mu = 0, sigma = 1)

rgauss(n, mu = 0, sigma = 1)
```

Arguments

x	vector of values.
mu	location parameter.
sigma	scale parameter.
log	flag for log density.
q	vector of quantiles.
p	vector of probabilities.
n	number of observations.

Value

A vector of density, distribution function, quantile or random values.

gauss0 *Centred Gaussian distribution*

Description

Centred Gaussian distribution

Usage

```
dgauss0(x, sigma = 1, log = FALSE)

pgauss0(q, sigma = 1)

qgauss0(p, sigma = 1)

rgauss0(n, sigma = 1)
```

Arguments

x	vector of values.
sigma	scale parameter.
log	flag for log density.
q	vector of quantiles.
p	vector of probabilities.
n	number of observations.

Value

A vector of density, distribution function, quantile or random values.

glag *Generalized lagging function*

Description

Generalized lagging function

Usage

```
glag(x, lagmax = 20, glagplot = FALSE)
```

Arguments

`x`	an object of class tscopulafit.
`lagmax`	maximum value for lag.
`glagplot`	logical value indicating generalized lag plot.

Value

If `glagplot` is `TRUE` a list of generalized lagged datasets of maximum length 9 is returned to facilitate a generalized lagplot. If `glagplot` is `FALSE` a vector of length `lagmax` containing the Kendall rank correlations for the generalized lagged datasets is returned.

`kendall` *Generic for Kendall correlations*

Description

Methods are available for objects of class armacopula, dvinecopula, dvinecopula2 and vtscopula.

Usage

```
kendall(object, ...)
```

Arguments

`object`	an object of the model class.
`...`	further arguments to be passed to Kendall calculation.

Value

A vector of Kendall correlations.

`kfilter` *Kalman filter for ARMA copula model*

Description

Kalman filter for ARMA copula model

Usage

```
kfilter(x, y)
```

Arguments

`x`	an object of class armacopula.
`y`	a vector of data.

Value

A matrix or multivariate time series with columns consisting of conditional mean, standard deviation and residuals.

Examples

```
data <- sim(armacopula(list(ar = c(0.5, 0.4), ma = -0.8)), n = 1000)
kfilter(armacopula(list(ar = c(0.5, 0.4), ma = -0.8)), data)
```

kpacf_arfima *KPACF of ARFIMA process*

Description

KPACF of ARFIMA process

Usage

```
kpacf_arfima(k, theta)
```

Arguments

k	number of lags.
theta	list with components ar, ma and d specifying the ARFIMA parameters

Value

A vector of Kendall partial autocorrelations of length k.

kpacf_arma *KPACF of ARMA process*

Description

KPACF of ARMA process

Usage

```
kpacf_arma(k, theta)
```

Arguments

k	number of lags.
theta	list with components ar and ma specifying the ARMA parameters.

Value

A vector of Kendall partial autocorrelations of length k.

`kpacf_fbn` *KPACF of fractional Brownian noise*

Description

KPACF of fractional Brownian noise

Usage

```
kpacf_fbn(k, theta)
```

Arguments

`k` number of lags

`theta` parameter of process

Value

A vector of Kendall partial autocorrelations of length `k`.

`kpacf_sarma12` *KPACF of monthly seasonal ARMA process*

Description

KPACF of monthly seasonal ARMA process

Usage

```
kpacf_sarma12(k, theta)
```

Arguments

`k` number of lags.

`theta` list with components ar, ma, sar and sma specifying the ARMA and seasonal ARMA parameters.

Value

A vector of Kendall partial autocorrelations of length `k`.

`kpacf_sarma4` *KPACF of quarterly seasonal ARMA process*

Description

KPACF of quarterly seasonal ARMA process

Usage

```
kpacf_sarma4(k, theta)
```

Arguments

`k`	number of lags.
`theta`	list with components ar, ma, sar and sma specifying the ARMA and seasonal ARMA parameters.

Value

A vector of Kendall partial autocorrelations of length `k`.

`laplace` *Laplace distribution*

Description

Laplace distribution

Usage

```
dlaplace(x, mu = 0, scale = 1, log = FALSE)

plaplace(q, mu = 0, scale = 1)

qlaplace(p, mu = 0, scale = 1)

rlaplace(n, mu = 0, scale = 1)
```

Arguments

`x`	vector of values.
`mu`	location parameter.
`scale`	scale parameter.
`log`	flag for log density.
`q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations.

Value

A vector of density, distribution function, quantile or random values.

laplace0 *Centred Laplace distribution*

Description

Centred Laplace distribution

Usage

```
dlaplace0(x, scale = 1, log = FALSE)

plaplace0(q, scale = 1)

qlaplace0(p, scale = 1)

rlaplace0(n, scale = 1)
```

Arguments

x	vector of values.
scale	scale parameter.
log	flag for log density.
q	vector of quantiles.
p	vector of probabilities.
n	number of observations.

Value

A vector of density, distribution function, quantile or random values.

margin *Constructor function for margin*

Description

Constructor function for margin

Usage

```
margin(name, pars = NULL)
```

Arguments

`name`	character string giving name of distribution
`pars`	parameters of the distribution

Value

An object of class margin.

Examples

```
margin("sst")
```

margin-class *Marginal model for time series*

Description

Class of objects for marginal models for stationary time series. The object is given a name and there must exist functions pname, qname, dname and rname. As well as the parameters of the distribution, dname must have the logical argument log specifying whether log density should be computed.

Usage

```
## S4 method for signature 'margin'
coef(object)

## S4 method for signature 'margin'
sim(object, n = 1000)

## S4 method for signature 'margin'
show(object)
```

Arguments

`object`	an object of the class.
`n`	length of realization.

Methods (by generic)

- `coef(margin)`: Coef method for margin class
- `sim(margin)`: Simulation method for margin class
- `show(margin)`: Show method for margin class

Slots

`name` name of the marginal model class.

`pars` a numeric vector containing the named parameters of the distribution which are passed as arguments to pname, qname, dname and rname.

Examples

```
new("margin", name = "gauss", pars = c(mu = 0, sigma = 1))
margmod <- margin("gauss", pars = c(mu = 0, sigma = 1))
sim(margmod, n = 500)
```

`marginfit-class` *Fitted marginal model for time series*

Description

Fitted marginal model for time series

Usage

```
## S4 method for signature 'marginfit'
logLik(object)
```

Arguments

`object`	an object of the class.

Methods (by generic)

- `logLik(marginfit)`: logLik method for marginfit class

Slots

`margin` an object of class margin.

`data` numeric vector or time series of data.

`fit` a list containing details of the maximum likelihood fit.

`non_invert` *Check for invertibility of ARMA process*

Description

Check for invertibility of ARMA process

Usage

```
non_invert(ma)
```

Arguments

`ma`	vector of moving average parameters.

Value

A logical variable stating whether ARMA process is invertible.

`non_stat` *Check for causality of ARMA process*

Description

Check for causality of ARMA process

Usage

```
non_stat(ar)
```

Arguments

`ar`	vector of autoregressive parameters

Value

A logical variable stating whether ARMA process is causal.

pacf2acf *Compute autocorrelations from partial autocorrelations*

Description

Compute autocorrelations from partial autocorrelations

Usage

```
pacf2acf(alpha)
```

Arguments

`alpha`	vector of partial autocorrelation values.

Value

A vector of autocorrelation values with same length as `alpha`.

Examples

```
alpha <- ARMAacf(ar = -0.9, ma = 0.8, lag.max = 50, pacf = TRUE)
rho <- pacf2acf(alpha)
```

pacf2ar *Compute autoregressive coefficients from partial autocorrelations*

Description

Compute autoregressive coefficients from partial autocorrelations

Usage

```
pacf2ar(alpha)
```

Arguments

`alpha`	vector of partial autocorrelation values.

Value

A vector of autoregressive coefficients with same length as `alpha`.

Examples

```
alpha <- ARMAacf(ar = -0.9, ma = 0.8, lag.max = 50, pacf = TRUE)
phi <- pacf2ar(alpha)
```

pcoincide *Compute coincidence probability for v-transform*

Description

Computes the probability that if we v-transform a uniform random variable and then stochastically invert the v-transform, we get back to the original value.

Usage

```
pcoincide(x)
```

Arguments

x	an object of class Vtransform.

Value

The probability of coincidence.

Examples

```
pcoincide(Vlinear(delta = 0.4))
pcoincide(V3p(delta = 0.45, kappa = 0.5, xi = 1.3))
```

pedf *Adjusted empirical distribution function*

Description

Adjusted empirical distribution function

Usage

```
pedf(x, data, proper = FALSE)
```

Arguments

x	argument of empirical distribution function.
data	vector of data for constructing empirical distribution function.
proper	logical variable which when set to TRUE will return the standard empirical distribution function.

Value

a vector of same length as x

plot,marginfit,missing-method

Plot method for marginfit class

Description

Plot method for marginfit class

Usage

```
## S4 method for signature 'marginfit,missing'
plot(x, bw = FALSE)
```

Arguments

x	an object of class marginfit.
bw	logical variable specifying whether black-white options should be chosen.

Value

No return value, generates plot.

plot,tscmfit,missing-method

Plot method for tscmfit class

Description

Plot method for tscmfit class

Usage

```
## S4 method for signature 'tscmfit,missing'
plot(x, plottype = "residual", bw = FALSE, lagmax = 30)
```

Arguments

x	an object of class tscmfit.
plottype	type of plot required.
bw	logical variable specifying whether black-white options should be chosen.
lagmax	maximum lag value for dvinecopula2 plots

Value

No return value, generates plot.

plot,tscopulafit,missing-method

Plot method for tscopulafit class

Description

Plot method for tscopulafit class

Usage

```
## S4 method for signature 'tscopulafit,missing'
plot(x, plottype = "residual", bw = FALSE, lagmax = 30)
```

Arguments

x	an object of class tscopulafit.
plottype	type of plot required.
bw	logical variable specifying whether black-white options should be chosen.
lagmax	maximum lag value for Kendall plots

Value

No return value, generates plot.

Examples

```
data <- sim(armacopula(list(ar = 0.5, ma = 0.4)), n = 1000)
fit <- fit(armacopula(list(ar = 0.5, ma = 0.4)), data)
plot(fit)
```

plot,Vtransform,missing-method

Plot method for Vtransform class

Description

Plots the v-transform as well as its gradient or inverse. Can also plot the conditional probability that a series PIT falls below the fulcrum for a given volatility PIT value v.

Usage

```
## S4 method for signature 'Vtransform,missing'
plot(
  x,
  type = "transform",
  shading = TRUE,
  npoints = 200,
  lower = 0,
  upper = 1
)
```

Arguments

`x` an object of class Vtransform.

`type` type of plot: ’transform’ for plot of transform, ’inverse’ for plot of inverse, ’gradient’ for plot of gradient or ’pdown’ for plot of conditional probability.

`shading` logical variable specifying whether inadmissible zone for v-transform should be shaded

`npoints` number of plotting points along x-axis.

`lower` the lower x-axis value for plotting.

`upper` the upper x-axis value for plotting

Value

No return value, generates plot.

Examples

```
plot(Vsymmetric())
plot(V2p(delta = 0.45, kappa = 0.8), type = "inverse")
plot(V2p(delta = 0.45, kappa = 0.8), type = "gradient")
```

---

pmarg *Compute CDF of marginal model*

Description

Compute the cumulative distribution function of the marginal model.

Usage

```
pmarg(x, q)
```

Arguments

`x` an object of class margin.

`q` vector of values at which CDF should be computed.

Value

A vector of values for the CDF.

Examples

```
margmod <- margin("gauss", pars = c(mu = 0, sigma = 1))
pmarg(margmod, c(-2, 0, 2))
```

---

profilefulcrum *Profile likelihood for fulcrum parameter*

Description

Profile likelihood for fulcrum parameter

Usage

```
profilefulcrum(
  data,
  tscopula = dvinecopula(family = 1, pars = list(0.1)),
  locations = seq(0, 1, by = 0.1),
  plot = TRUE
)
```

Arguments

`data`	a vector or time series of data on (0,1).
`tscopula`	an object of class tscopulaU or vtscopula.
`locations`	vector containing locations of different values for fulcrum.
`plot`	logical values specifying whether plot should be created.

Value

A matrix containing fulcrum values and log likelihood values.

Examples

```
copobject <- armacopula(pars = list(ar = 0.6, ma = 0.2))
vtcop <- vtscopula(copobject, Vtransform = V2p())
y <- sim(vtcop)
profilefulcrum(y, vtcop)
```

qmarg *Compute quantiles of marginal model*

Description

Compute the quantile function of the marginal model.

Usage

```
qmarg(x, p)
```

Arguments

x	an object of class margin.
p	vector of probabilities for which quantiles should be computed.

Value

A vector of values for the quantile function.

Examples

```
margmod <- margin("gauss", pars = c(mu = 0, sigma = 1))
qmarg(margmod, c(0.05, 0.5, 0.95))
```

quantile,tscmfit-method *Quantile calculation method for VT-ARMA models*

Description

Quantile calculation method for VT-ARMA models

Usage

```
## S4 method for signature 'tscmfit'
quantile(x, alpha, last = FALSE)
```

Arguments

x	an object of class tscmfit based on underlying copula of class armacopula.
alpha	a scalar probability value
last	logical value asserting that only the last volatility prediction should be returned

Value

a vector of the same length as the data embedded in the tscmfit object.

Rbackward *Rosenblatt backward function with v-transforms*

Description

Rosenblatt backward function with v-transforms

Usage

```
Rbackward(x, u, pcs, vt1, vt2)
```

Arguments

`x`	vector argument of Rosenblatt functiom
`u`	matrix of conditioning values. Number of rows must be either 1 or same length as x. Number of columns should not be much more than 15 (due to repeated recursive calling)
`pcs`	list of pair copulas
`vt1`	first v-transform
`vt2`	second v-transform

Value

vector of same length as x

RforwardI *Inverse Rosenblatt forward function with v-transforms*

Description

Inverse Rosenblatt forward function with v-transforms

Usage

```
RforwardI(x, u, pcs, vt1, vt2)
```

Arguments

`x`	vector argument of Rosenblatt functiom
`u`	matrix of conditioning values. Number of rows must be either 1 or same length as x. Number of columns should not be much more than 15 (due to repeated recursive calling)
`pcs`	list of pair copulas
`vt1`	first v-transform
`vt2`	second v-transform

Value

vector of same length as x

`safe_ses` *Calculate standard errors safely*

Description

Calculate standard errors safely

Usage

```
safe_ses(hess)
```

Arguments

`hess`	a Hessian matrix from a model fit.

Value

a vector of standard errors.

`sarma2arma` *Transform a sarmacopula object into an armacopula object*

Description

Transform a sarmacopula object into an armacopula object

Usage

```
sarma2arma(object)
```

Arguments

`object`	an object of class sarmacopula.

Value

An object of class armacopula.

Examples

```
sarma2arma(sarmacopula(list(ar = 0.5, ma = 0.4, sar = 0.2, sma = 0.6), period = 4))
```

`sarma2dvine` *Transform a sarmacopula into a dvinecopula2 object*

Description

Transform a sarmacopula into a dvinecopula2 object

Usage

```
sarma2dvine(object)
```

Arguments

`object` an object of class sarmacopula.

Value

An object of class dvinecopula2.

Examples

```
sarma2dvine(sarmacopula(list(ar = 0.5, ma = 0.4, sar = 0.2, sma = 0.6), period = 4))
```

`sarmacopula` *Constructor function for SARMA copula process*

Description

Constructor function for SARMA copula process

Usage

```
sarmacopula(pars = list(ar = 0, ma = 0, sar = 0, sma = 0), period = 4)
```

Arguments

`pars` list consisting of vector of AR parameters named ‘ar‘ and vector of MA parameters named ‘ma‘, SAR parameters named ‘sar‘ and vector of SMA parameters named ‘sma‘.

`period` period of seasonal model.

Value

An object of class sarmacopula.

Examples

```
sarmacopula(list(ar = 0.5, ma = 0.4, sar = 0.2, sma = 0.6), period = 4)
```

`sarmacopula-class` *SARMA copula processes*

Description

Class of objects for seasonal ARMA copula processes.

Usage

```
## S4 method for signature 'sarmacopula'
coef(object)

## S4 method for signature 'sarmacopula'
show(object)

## S4 method for signature 'sarmacopula'
sim(object, n = 1000)

## S4 method for signature 'sarmacopula'
kendall(object, lagmax = 20)

## S4 method for signature 'sarmacopula'
predict(object, data, x, type = ”df”)
```

Arguments

`object`	an object of the class.
`n`	length of realization.
`lagmax`	maximum value of lag.
`data`	vector of past data values.
`x`	vector of arguments of prediction function.
`type`	type of prediction function ("df" for density, "qf" for quantile function or "dens" for density).

Methods (by generic)

- `coef(sarmacopula)`: Coef method for SARMA copula class
- `show(sarmacopula)`: Show method for SARMA copula process
- `sim(sarmacopula)`: Simulation method for sarmacopula class
- `kendall(sarmacopula)`: Calculate Kendall’s tau values for sarmacopula model
- `predict(sarmacopula)`: Prediction method for sarmacopula class

Slots

- `name` name of seasonal ARMA copula process.
- `modelspec` vector containing number of AR, MA, SAR and SMA parameters as well as the order D of seasonal differencing.
- `pars` list consisting of vector of AR parameters named ‘ar‘ and vector of MA parameters named ‘ma‘, SAR parameters named ‘sar‘ and vector of SMA parameters named ‘sma‘.

Examples

```
sim(sarma2arma(sarmacopula(list(ar = 0.5, ma = 0.4, sar = 0.2, sma = 0.6), period = 4)))
mod <- sarmacopula(list(ar = 0.5, ma = 0.4, sar = 0.2, sma = 0.6), period = 4)
kendall(mod)
```

sdoubleweibull *Skew double Weibull distribution*

Description

Skew double Weibull distribution

Usage

```
dsdoubleweibull(x, mu = 0.05, shape = 1, scale = 1, gamma = 1, log = FALSE)

psdoubleweibull(q, mu = 0.05, shape = 1, scale = 1, gamma = 1)

qsdoubleweibull(p, mu = 0.05, shape = 1, scale = 1, gamma = 1)

rsdoubleweibull(n, mu = 0.05, shape = 1, scale = 1, gamma = 1)
```

Arguments

`x`	vector of values.
`mu`	location parameter.
`shape`	shape parameter.
`scale`	scale parameter.
`gamma`	skewness parameter.
`log`	flag for log density.
`q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations.

Value

A vector of density, distribution function, quantile or random values.

sigmastarma *Standard deviation of innovations for armacopula*

Description

Uses the function `tacvfARMA` in the ltsa library.

Usage

```
sigmastarma(x)
```

Arguments

`x`	an object of class armacopula.

Value

The standard deviation of the standardized ARMA innovation distribution.

Examples

```
sigmastarma(armacopula(list(ar = c(0.5, 0.4), ma = -0.8)))
```

sim *Generic for simulating time series copula models*

Description

Methods are available for objects of class swncopula, armacopula, dvinecopula, dvinecopula2, margin and tscm.

Usage

```
sim(object, ...)
```

Arguments

`object`	an object of the model class.
`...`	further arguments to be passed to the simulation.

Value

A simulated realization from the time series model.

slaplace *Skew Laplace distribution*

Description

Skew Laplace distribution

Usage

```
dslaplace(x, mu = 0.05, scale = 1, gamma = 1, log = FALSE)

pslaplace(q, mu = 0.05, scale = 1, gamma = 1)

qslaplace(p, mu = 0.05, scale = 1, gamma = 1)

rslaplace(n, mu = 0.05, scale = 1, gamma = 1)
```

Arguments

x	vector of values.
mu	location parameter.
scale	scale parameter.
gamma	skewness parameter.
log	flag for log density.
q	vector of quantiles.
p	vector of probabilities.
n	number of observations.

Value

A vector of density, distribution function, quantile or random values.

sst *Skew Student t distribution*

Description

Skew Student t distribution

Usage

```
psst(q, df = 10, gamma = 1, mu = 0, sigma = 1)

qsst(p, df, gamma, mu, sigma)

dsst(x, df, gamma, mu, sigma, log = FALSE)

rsst(n, df, gamma, mu, sigma)
```

Arguments

q	vector of quantiles.
df	degrees of freedom.
gamma	skewness parameter.
mu	location parameter.
sigma	scale parameter.
p	vector of probabilities.
x	vector of values.
log	flag for log density.
n	number of observations.

Value

A vector of density, distribution function, quantile or random values.

st *Student t distribution*

Description

Student t distribution

Usage

```
pst(q, df = 10, mu = 0, sigma = 1)

qst(p, df, mu, sigma)

dst(x, df, mu, sigma, log = FALSE)

rst(n, df, mu, sigma)
```

Arguments

q	vector of quantiles.
df	degrees of freedom.
mu	location parameter.
sigma	scale parameter.
p	vector of probabilities.
x	vector of values.
log	flag for log density.
n	number of observations.

Value

A vector of density, distribution function, quantile or random values.

st0 *Centred Student t distribution*

Description

Centred Student t distribution

Usage

```
pst0(q, df = 10, sigma = 1)

qst0(p, df, sigma)

dst0(x, df, sigma, log = FALSE)

rst0(n, df, sigma)
```

Arguments

q	vector of quantiles.
df	degrees of freedom.
sigma	scale parameter.
p	vector of probabilities.
x	vector of values.
log	flag for log density.
n	number of observations.

Value

A vector of density, distribution function, quantile or random values.

`stochinverse` *Stochastic inverse of a v-transform*

Description

Stochastic inverse of a v-transform

Usage

```
stochinverse(x, v, tscopula = NULL, tol = .Machine$double.eps^0.75)
```

Arguments

`x`	an object of class Vtransform.
`v`	a vector, matrix or time series with values in [0, 1].
`tscopula`	a time series copula object.
`tol`	the desired accuracy (convergence tolerance) that is passed to `uniroot` if numerical inversion is used.

Value

A vector, matrix or time series with values in [0, 1].

Examples

```
stochinverse(Vsymmetric(), c(0, 0.25, 0.5, 0.75, 1))
```

`strank` *Calculate standardized ranks of data*

Description

Calculate standardized ranks of data

Usage

```
strank(x)
```

Arguments

`x`	a vector or time series of data.

Value

A vector or time series of standardized ranks in the interval (0,1)

Examples

```
strank(rnorm(100))
```

`swncopula` *Constructor function for strict white noise copula process*

Description

Constructor function for strict white noise copula process

Usage

```
swncopula()
```

Value

Object of class swncopula.

Examples

```
swncopula()
```

`swncopula-class` *Strict white noise copula process*

Description

Strict white noise copula process

Usage

```
## S4 method for signature 'swncopula'
sim(object, n = 1000)

## S4 method for signature 'swncopula'
coef(object)

## S4 method for signature 'swncopula'
show(object)
```

Arguments

`object`	an object of class swncopula.
`n`	numeric value for length of simulated realisation.

Methods (by generic)

- `sim(swncopula)`: Simulation method for strict white noise copula
- `coef(swncopula)`: Coef method for strict white noise copula
- `show(swncopula)`: Show method for strict white noise copula

Examples

```
sim(swncopula())
```

---

tscm *Constructor function for time series*

Description

Constructor function for time series

Usage

```
tscm(tscopula, margin = new("margin", name = "unif"))
```

Arguments

`tscopula`	an object of class tscopula.
`margin`	an object of class margin.

Value

An object of class tscm.

Examples

```
tscm(dvinecopula(family = "gauss", pars = 0.5), margin("doubleweibull"))
```

tscm-class *Full models*

Description

Class of objects for composite time series models consisting of stationary copula processes and marginal distributions.

Usage

```
## S4 method for signature 'tscm'
show(object)

## S4 method for signature 'tscm'
coef(object)

## S4 method for signature 'tscm'
sim(object, n = 1000)

## S4 method for signature 'tscm'
predict(object, data, x, type = "df", qtype = 7, proper = FALSE)

## S4 method for signature 'tscm'
kendall(object, lagmax = 20)
```

Arguments

`object`	an object of the class.
`n`	length of realization.
`data`	vector of past data values.
`x`	vector of arguments of prediction function.
`type`	type of prediction function ("df" for density, "qf" for quantile function or "dens" for density).
`qtype`	type of empirical quantile estimate.
`proper`	logical variable stating whether the standard empirical distribution function should be used when the margin is empirical; otherwise an improper distribution that is bounded away from 0 and 1 is used.
`lagmax`	maximum value of lag.

Methods (by generic)

- `show(tscm)`: Show method for tscm class
- `coef(tscm)`: Coefficient method for tscm class
- `sim(tscm)`: Simulation method for tscm class
- `predict(tscm)`: Prediction method for tscm class
- `kendall(tscm)`: Calculate Kendall’s tau values for pair copulas for tscm class

Slots

`tscopula` an object of class tscopula.

`margin` an object of class margin.

Examples

```
mod <- tscm(dvinecopula(family = "gauss", pars = 0.5), margin("doubleweibull"))
sim(mod)
```

---

tscmfit-class *Fitted tscm model*

Description

Class of objects for fitted tscm models.

Usage

```
## S4 method for signature 'tscmfit'
logLik(object)

## S4 method for signature 'tscmfit'
resid(object, trace = FALSE)

## S4 method for signature 'tscmfit'
predict(object, x, type = "df", qtype = 7, proper = FALSE)
```

Arguments

`object` an object of the class.

`trace` extract trace instead of residuals.

`x` vector of arguments of prediction function.

`type` type of prediction function ("df" for density, "qf" for quantile function or "dens" for density).

`qtype` type of empirical quantile estimate.

`proper` logical variable stating whether the standard empirical distribution function should be used when the margin is empirical; otherwise an improper distribution that is bounded away from 0 and 1 is used.

Methods (by generic)

- `logLik(tscmfit)`: method for tscmfit class
- `resid(tscmfit)`: Residual method for tscmfit class
- `predict(tscmfit)`: Prediction method for tscmfit class

Slots

`tscopula` an object of class tscopula.

`margin` an object of class margin.

`data` a vector or time series of data to which process has been fitted.

`fit` a list containing details of the fit.

`tscopula-class` *Time series copula processes*

Description

Class of objects for time series copula processes.

`tscopulafit-class` *Fitted time series copula processes*

Description

Class of objects for fitted time series copula processes.

Usage

```
## S4 method for signature 'tscopulafit'
sim(object, n = 1000)

## S4 method for signature 'tscopulafit'
kendall(object, lagmax = 20)

## S4 method for signature 'tscopulafit'
coef(object)

## S4 method for signature 'tscopulafit'
show(object)

## S4 method for signature 'tscopulafit'
logLik(object)

## S4 method for signature 'tscopulafit'
resid(object, trace = FALSE)

## S4 method for signature 'tscopulafit'
predict(object, x, type = "df")
```

Arguments

`object`	an object of class tscopulafit.
`n`	length of realization.
`lagmax`	maximum value of lag.
`trace`	extract trace instead of residuals.
`x`	vector of arguments of prediction function.
`type`	type of prediction function ("df" for density, "qf" for quantile function or "dens" for density).

Methods (by generic)

- `sim(tscopulafit)`: Simulation method for tscopulafit class
- `kendall(tscopulafit)`: Calculate Kendall's tau values for pair copulas for tscopulafit class
- `coef(tscopulafit)`: Coef method for tscopulafit class
- `show(tscopulafit)`: Show method for tscopulafit objects
- `logLik(tscopulafit)`: logLik method for tscopulafit class
- `resid(tscopulafit)`: Residual method for tscopulafit class
- `predict(tscopulafit)`: Prediction method for tscopulafit class

Slots

`tscopula` an object of class tscopula.

`data` a vector or time series of data.

`fit` a list containing details of the fit.

Examples

```
ar1 <- armacopula(list(ar = 0.7))
data <- sim(ar1, 1000)
ar1fit <- fit(ar1, data)
sim(ar1fit)
```

`tscopulaU-class` *Time series copulas of class tscopulaU*

Description

S4 Class union for basic time series copula types. These are armacopula, dvinecopula and dvinecopula2,

V2b *Constructor function for 2-parameter beta v-transform*

Description

Constructor function for 2-parameter beta v-transform

Usage

```
V2b(delta = 0.5, kappa = 1)
```

Arguments

delta	a value in (0, 1) specifying the fulcrum of the v-transform.
kappa	additional positive parameter of v-transform.

Value

An object of class Vtransform.

Examples

```
V2b(delta = 0.45, kappa = 1.2)
```

V2p *Constructor function for 2-parameter v-transform*

Description

Constructor function for 2-parameter v-transform

Usage

```
V2p(delta = 0.5, kappa = 1)
```

Arguments

delta	a value in (0, 1) specifying the fulcrum of the v-transform.
kappa	additional positive parameter of v-transform.

Value

An object of class Vtransform.

Examples

```
V2p(delta = 0.45, kappa = 1.2)
```

V3b *Constructor function for 3-parameter beta v-transform*

Description

Constructor function for 3-parameter beta v-transform

Usage

```
V3b(delta = 0.5, kappa = 1, xi = 1)
```

Arguments

`delta`	a value in (0, 1) specifying the fulcrum of the v-transform.
`kappa`	additional positive parameter of v-transform.
`xi`	additional positive parameter of v-transform.

Value

An object of class Vtransform.

Examples

```
V3b(delta = 0.45, kappa = 1.2, xi = 1.2)
```

V3p *Constructor function for 3-parameter v-transform*

Description

Constructor function for 3-parameter v-transform

Usage

```
V3p(delta = 0.5, kappa = 1, xi = 1)
```

Arguments

`delta`	a value in (0, 1) specifying the fulcrum of the v-transform.
`kappa`	additional positive parameter of v-transform.
`xi`	additional positive parameter of v-transform.

Value

An object of class Vtransform.

Examples

```
V3p(delta = 0.45, kappa = 0.8, xi = 1.1)
```

Vdegenerate *Constructor function for degenerate v-transform*

Description

Constructor function for degenerate v-transform

Usage

```
Vdegenerate()
```

Value

An object of class VtransformI.

Examples

```
Vdegenerate()
```

vdownprob *Calculate conditional down probability of v-transform*

Description

Calculate conditional down probability of v-transform

Usage

```
vdownprob(x, v)
```

Arguments

x	an object of class Vtransform.
v	a vector or time series with values in [0, 1].

Value

A vector or time series of values of gradient.

Examples

```
vdownprob(V2p(delta = 0.55, kapp = 1.2), c(0, 0.25, 0.5, 0.75, 1))
```

`vgradient` *Calculate gradient of v-transform*

Description

Calculate gradient of v-transform

Usage

```
vgradient(x, u)
```

Arguments

`x`	an object of class Vtransform.
`u`	a vector or time series with values in [0, 1].

Value

A vector or time series of values of gradient.

Examples

```
vgradient(Vsymmetric(), c(0, 0.25, 0.5, 0.75, 1))
```

`vinverse` *Calculate inverse of v-transform*

Description

If the Vtransform object is also a VtransformI object (an invertible v-transform) then the analytical inverse is used. Otherwise an inverse is found by numerical root finding with `uniroot`.

Usage

```
vinverse(x, v, tol = .Machine$double.eps^0.75)
```

Arguments

`x`	an object ofc lass Vtransform.
`v`	a vector or time series with values in [0, 1].
`tol`	the desired accuracy (convergence tolerance) that is passed to `uniroot` if numerical inversion is used.

Value

A vector or time series with values in [0, 1].

Examples

```
vinverse(Vsymmetric(), c(0, 0.25, 0.5, 0.75, 1))
```

Vlinear *Constructor function for linear v-transform*

Description

Constructor function for linear v-transform

Usage

```
Vlinear(delta = 0.5)
```

Arguments

delta	a value in (0, 1) specifying the fulcrum of the v-transform.

Value

An object of class VtransformI.

Examples

```
Vlinear(delta = 0.45)
```

Vsymmetric *Constructor function for symmetric v-transform*

Description

Constructor function for symmetric v-transform

Usage

```
Vsymmetric()
```

Value

An object of class VtransformI.

Examples

```
Vsymmetric()
```

vtrans *Evaluate a v-transform*

Description

Evaluate a v-transform

Usage

```
vtrans(x, u)
```

Arguments

x	an object of class Vtransform.
u	a vector or time series with values in [0, 1].

Value

A vector or time series with values in [0, 1].

Examples

```
vtrans(Vsymmetric(), c(0, 0.25, 0.5, 0.75, 1))
```

Vtransform-class *Class of v-transforms*

Description

This is the class of v-transforms. It contains the VtransformI subclass consisting of v-transforms with an analytical expression for the inverse.

Usage

```
## S4 method for signature 'Vtransform'
show(object)

## S4 method for signature 'Vtransform'
coef(object)
```

Arguments

object	an object of the class.

Methods (by generic)

- `show(Vtransform)`: Show method for Vtransform class
- `coef(Vtransform)`: Coef method for Vtransform class

Slots

`name` a name for the v-transform of class character.

`Vtrans` function to evaluate the v-transform.

`pars` vector containing the named parameters of the v-transform.

`gradient` function to evaluate the gradient of the v-transform.

Examples

```
V2p(delta = 0.5, kappa = 1.2)
```

`VtransformI-class` *Class of invertible v-transforms*

Description

This class inherits from the Vtransform class and contains v-transforms with an analytical expression for the inverse.

Slots

`name` a name for the v-transform of class character.

`Vtrans` function to evaluate the v-transform.

`pars` vector containing the named parameters of the v-transform.

`gradient` function to evaluate the gradient of the v-transform.

`inverse` function to evaluate the inverse of the v-transform.

Examples

```
Vlinear(delta = 0.55)
```

vtscopula *Constructor function for vtscopula object*

Description

Constructor function for vtscopula object

Usage

```
vtscopula(tscopulaU, Vtransform = Vlinear(), Wcopula = swncopula())
```

Arguments

`tscopulaU`	an object of class armacopula, dvinecopula or dvinecopula2.
`Vtransform`	an object of class Vtransform.
`Wcopula`	an object of class tscopula.

Value

An object of class vtscopula.

Examples

```
copobject <- armacopula(pars = list(ar = 0.6, ma = 0.2))
vtscopula(copobject, Vtransform = V2p())
```

vtscopula-class *Time series copula processes with v-transforms*

Description

Class of objects for v-transformed time series copula processes.

Usage

```
## S4 method for signature 'vtscopula'
show(object)

## S4 method for signature 'vtscopula'
coef(object)

## S4 method for signature 'vtscopula'
predict(object, data, x, type = "df")

## S4 method for signature 'vtscopula'
```

```
sim(object, n = 1000)

## S4 method for signature 'vtscopula'
kendall(object, lagmax = 20)
```

Arguments

`object`	an object of the class.
`data`	vector of past data values.
`x`	vector of arguments of prediction function.
`type`	type of prediction function ("df" for density, "qf" for quantile function or "dens" for density).
`n`	length of realization.
`lagmax`	maximum value of lag.

Methods (by generic)

- `show(vtscopula)`: Show method for vtscopula objects
- `coef(vtscopula)`: Coef method for vtscopula class
- `predict(vtscopula)`: Prediction method for vtscopula class
- `sim(vtscopula)`: Simulation method for vtscopula class
- `kendall(vtscopula)`: Calculate Kendall's tau values for vtscopula model

Slots

`Vcopula` object of class tscopulaU.

`Vtransform` object of class Vtransform.

`Wcopula` object of class tscopula.

Examples

```
copobject <- armacopula(pars = list(ar = 0.6, ma = 0.2))
sim(vtscopula(copobject, Vtransform = V2p()))
mod <- vtscopula(armacopula(list(ar = 0.95, ma = -0.85)))
kendall(mod)
```

Index