Package ‘cylcop’

October 30, 2022

Title Circular-Linear Copulas with Angular Symmetry for Movement Data

Version 0.2.0

URL https://github.com/r-lib/devtools

BugReports https://github.com/r-lib/devtools/issues

Maintainer Florian Hodel <florian.hodel@yahoo.com>

Description Classes (S4) of circular-linear, symmetric copulas with corresponding methods, extending the 'copula' package. These copulas are especially useful for modeling correlation in discrete-time movement data. Methods for density, (conditional) distribution, random number generation, bivariate dependence measures and fitting parameters using maximum likelihood and other approaches. The package also contains methods for visualizing movement data and copulas.

License GPL (>= 2)

Encoding UTF-8

Depends R (>= 3.5),

Imports circular, stats, purrr, dplyr (>= 0.7.0), copula, stringr, rlang, methods, GoFKernel, MASS, data.table, infotheo, ggplot2, utils, rgl, viridis, plotly, cowplot, movMF, Rdpack, mixR, transport

RdMacros Rdpack

RoxygenNote 7.2.1

Collate 'cyl_cop_class.R' 'Ccond.R' 'aaaglobal.R' 'correlation.R' 'cyl_cubsec.R' 'cyl_quadsec.R' 'cyl_rect_combine.R' 'cyl_rot_combine.R' 'cyl_vonmises.R' 'cylcop-package.R' 'density.R' 'deprecated.R' 'fit_cop_corr.R' 'fit_cop_mle.R' 'fit_margin.R' 'gof.R' 'joint_distr.R' 'mixed_linear.R' 'mixedvonmises.R' 'opt_auto.R' 'plotting_functions.R' 'simulate_trajectory.R' 'utils.R' 'wrappedcauchy.R' 'zzz.R'

NeedsCompilation no

Author Florian Hodel [aut, cre] (<https://orcid.org/0000-0002-0099-1006>)

Repository CRAN

Date/Publication 2022-10-29 22:00:21 UTC

R topics documented:

`angstep2xy` *Calculate the Next Position in a Trajectory from a Turn Angle and a Step Length*

Description

The x-y-coordinates of a position in 2-D space is calculated from the angle between that position and the 2 previous ones in the trajectory and the distance between that position and the previous one.

Usage

```
angstep2xy(angle, steplength, prevp1, prevp2)
```

Arguments

`angle`	numeric value of the turn angle or a `circular` object, either in $[0, 2\pi)$ or in $[-\pi, \pi)$
`steplength`	numeric value giving the distance between the position and the previous one.
`prevp1`	numeric vector holding the x and y coordinates of the previous position.
`prevp2`	numeric vector holding the x and y coordinates of the position before the previous one.

Value

The function returns a numeric vector holding the x and y coordinates of the position

Examples

```
angstep2xy(1.5*pi, 2, prevp1 = c(1, 4), prevp2 = c(2, 7.5))
angstep2xy(-0.5*pi, 2, c(1, 4), c(2, 7.5))
```

bearing *Compass Bearing of a Line Between 2 Points*

Description

The angle between a line between 2 points in Euclidean 2-D space and the line from (0,0) to (0,1) is calculated. In other words, the compass bearing of a line between 2 points where north is 0. Angles increase in clockwise direction.

Usage

```
bearing(point1, point2, fullcirc = FALSE)
```

Arguments

point1	numeric vector holding the x and y coordinates of the first point.
point2	numeric vector holding the x and y coordinates of the second point.
fullcirc	logical value indicating whether the output should be an angle on $[0, 2\pi)$ or $[-\pi, \pi)$.

Value

If `fullcirc = FALSE`, the function returns a numeric value (angle) from the interval $[-\pi, \pi)$.
If `fullcirc = TRUE`, the function returns a numeric value numeric from the interval $[0, 2\pi)$.

Examples

```
bearing(c(3,5), c(1,4))
bearing(c(3,5), c(1,4), fullcirc = TRUE)
```

ccylcop *Conditional Distributions of Circular-Linear Copulas*

Description

Calculates the conditional distributions and their inverses of circular-linear copulas and 2-dimensional linear-linear copulas.

Usage

```
ccylcop(u, copula, cond_on = 2, inverse = FALSE, ...)

## S4 method for signature 'Copula'
ccylcop(u, copula, cond_on, inverse)

## S4 method for signature 'cyl_cubsec'
ccylcop(u, copula, cond_on = 2, inverse = FALSE)

## S4 method for signature 'cyl_quadsec'
ccylcop(u, copula, cond_on = 2, inverse = FALSE)

## S4 method for signature 'cyl_rect_combine'
ccylcop(u, copula, cond_on = 2, inverse = FALSE)

## S4 method for signature 'cyl_rot_combine'
ccylcop(u, copula, cond_on = 2, inverse = FALSE)

## S4 method for signature 'cyl_vonmises'
ccylcop(u, copula, cond_on = 2, inverse = FALSE)
```

Arguments

`u`	matrix (or vector) of numeric values in $[0, 1]^2$, containing as first column the circular (periodic) and as second the linear dimension.
`copula`	R object of class '`cyl_copula`'. or '`Copula`' (package '**copula**', only 2-dimensional).
`cond_on`	column number of `u` on which the copula is conditioned. E.g if `cond_on = 2`, the function calculates for each element in the first column of `u` the copula conditional on the corresponding element in the second column.
`inverse`	logical indicating whether the inverse of the conditional copula is calculated.
`...`	additional arguments.

Details

This is a generic that calls the function `copula::cCopula()` for 2-dimensional '`Copula`' objects from the '**copula**' package for which `copula::cCopula()` is available. If `copula::cCopula()` is not available, the conditional copula is calculated numerically. For '`cyl_copula`' objects, the conditional copula is calculated analytically or numerically (depending on the copula and the values of `u`). Note that the input arguments and the output of `cylcop::ccylcop()` differ from those of `copula::cCopula()`.

Value

A vector containing the values of the distribution of the copula at `u[,-cond_on]` conditional on the values of `u[,cond_on]`.

References

Nelsen RB (2006). *An Introduction to Copulas*, volume 139 of *Lecture Notes in Statistics*. Springer New York, New York, NY. ISBN 978-0-387-98623-4, doi:10.1007/9781475730760, https://link.springer.com/book/10.1007/978-1-4757-3076-0.

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/.

See Also

copula::cCopula()

Examples

```
cop <- cyl_quadsec(0.1)
#calculate C_u(v) with u = 0.1 and v = 0.5
cylcop::ccylcop(u = c(0.1, 0.5), copula = cop, cond_on = 1, inverse = FALSE)
#calculate C^-1_v(u) with u = 0.1 and v = 0.5 and with u = 0.4 and v = 0.2
cylcop::ccylcop(u = rbind(c(0.1, 0.5), c(0.4, 0.2)), copula = cop, cond_on = 2, inverse = TRUE)
```

---

cor_cyl *Estimate a Rank-Based Circular-Linear Correlation Coefficient*

Description

The code is based on Mardia (1976), Solow et al. (1988) and Tu (2015). The function returns a numeric value between 0 and 1, not -1 and 1, positive and negative correlation cannot be discerned. Note also that the correlation coefficient is independent of the marginal distributions.

Usage

```
cor_cyl(theta, x)
```

Arguments

theta	numeric vector of angles (measurements of a circular variable).
x	numeric vector of step lengths (measurements of a linear variable).

Value

A numeric value between 0 and 1, the circular-linear correlation coefficient.

References

Mardia KV (1976). "Linear-Circular Correlation Coefficients and Rhythmometry." *Biometrika*, **63**(2), 403–405. ISSN 00063444, doi:10.2307/2335637.

Solow AR, Bullister JL, Nevison C (1988). "An application of circular-linear correlation analysis to the relationship between Freon concentration and wind direction in Woods Hole, Massachusetts." *Environmental Monitoring and Assessment*, **10**(3), 219–228. ISSN 1573-2959, doi:10.1007/BF00395081, https://doi.org/10.1007/BF00395081.

Tu R (2015). "A Study of the Parametric and Nonparametric Linear-Circular Correlation Coefficient." *California Polytechnic State University, San Luis Obispo*, 1–24. https://digitalcommons.calpoly.edu/statsp/51/.

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/.

See Also

`mi_cyl()`, `fit_cylcop_cor()`.

Examples

```
set.seed(123)

cop <- cyl_quadsec(0.1)

#draw samples and calculate the correlation coefficient
sample <- rcylcop(100, cop)
cor_cyl(theta = sample[,1], x = sample[,2])

#the correlation coefficient is independent of the marginal distribution.
sample <- traj_sim(100,
  cop,
  marginal_circ = list(name = "vonmises", coef  = list(0, 1)),
  marginal_lin = list(name = "weibull", coef = list(shape = 2))
)
cor_cyl(theta = sample$angle, x = sample$steplength)
cor_cyl(theta = sample$cop_u, x = sample$cop_v)

# Estimate correlation of samples drawn from circular-linear copulas with
# perfect correlation
cop <- cyl_rect_combine(copula::normalCopula(1))
sample <- rcylcop(100, cop)
cor_cyl(theta = sample[,1], x = sample[,2])
```

cramer_vonmises *Cramér-von-Mises criterion*

Description

Calculate the Cramér-von-Mises criterion with a p-value (via parametric bootstrapping) to assess the goodness of fit of a parametric copula compared to the empirical copula of the data.

Usage

```
cramer_vonmises(
  copula,
  theta,
  x,
  n_bootstrap = 1000,
  parameters = NULL,
  optim.method = "L-BFGS-B",
  optim.control = list(maxit = 100)
)
```

Arguments

copula	R object of class 'cyl_copula' or 'Copula' (package '**copula**'.
theta	numeric vector of angles (measurements of a circular variable) or "circular" component of pseudo-observations.
x	numeric vector of step lengths (measurements of a linear variable) or "linear" component of pseudo-observations.
n_bootstrap	integer number of bootstrap replicates. If n_bootstrap is smaller than 1, no p-value is calculated.
parameters	vector of character strings holding the names of the parameters to be optimized when using the bootstrap procedure. These can be any parameters in copula@parameters. Default is to optimize the first 2 parameters. parameters has no effect if copula is of class 'Copula' (package '**copula**'
optim.method	character string, optimizer used in optim(), can be "Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN", or "Brent".
optim.control	list of additional controls passed to optim().

Details

The Cramér-von Misses criterion is calculated as the sum of the squared differences between the empirical copula and the parametric copula, copula, evaluated at the pseudo-observations obtained from theta and x. If the bootstrap procedure is used, a random sample is drawn from copula and converted to pseudo-observations. A new (set of) copula parameter(s) is then fit to those pseudo-observations using maximum likelihood (function cylcop::fit_cylcop_ml()).

Value

A list of length 2 containing the Cramér-von Mises criterion and the p-value.

References

Genest C, Rémillard B (2008). "Validity of the parametric bootstrap for goodness-of-fit testing in semiparametric models." *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, **44**(6), 1096 – 1127. doi:10.1214/07AIHP148.

Examples

```
set.seed(1234)
sample <- rcylcop(100,cyl_cubsec(0.1, 0.1))

opt_cop <- fit_cylcop_ml(copula = cyl_quadsec(),
  theta = sample[,1],
  x = sample[,2],
  parameters = "a",
  start = 0
)$copula
cramer_vonmises(opt_cop,
  theta = sample[,1],
  x = sample[,2],
  n_bootstrap=5)
```

Cylcop — *Distribution, Density, and Random Number Generation for Circular-Linear Copulas'*

Description

Calculate the distribution (`pcylcop()`), the density (`dcylcop()`), and generate random samples (`rcylcop()`) of a '`cyl_copula`' object or a '`Copula`' object (package '**copula**', only 2-dimensional).

Usage

```
pcylcop(u, copula)

rcylcop(n, copula)

dcylcop(u, copula, log = FALSE)

## S4 method for signature 'matrix,Copula'
dcylcop(u, copula)

## S4 method for signature 'numeric,Copula'
rcylcop(n, copula)
```

```
## S4 method for signature 'matrix,Copula'
pcylcop(u, copula)

## S4 method for signature 'numeric,cyl_cubsec'
rcylcop(n, copula)

## S4 method for signature 'matrix,cyl_cubsec'
dcylcop(u, copula)

## S4 method for signature 'matrix,cyl_cubsec'
pcylcop(u, copula)

## S4 method for signature 'numeric,cyl_quadsec'
rcylcop(n, copula)

## S4 method for signature 'matrix,cyl_quadsec'
dcylcop(u, copula)

## S4 method for signature 'matrix,cyl_quadsec'
pcylcop(u, copula)

## S4 method for signature 'numeric,cyl_rect_combine'
rcylcop(n, copula)

## S4 method for signature 'matrix,cyl_rect_combine'
dcylcop(u, copula)

## S4 method for signature 'matrix,cyl_rect_combine'
pcylcop(u, copula)

## S4 method for signature 'numeric,cyl_rot_combine'
rcylcop(n, copula)

## S4 method for signature 'matrix,cyl_rot_combine'
dcylcop(u, copula)

## S4 method for signature 'matrix,cyl_rot_combine'
pcylcop(u, copula)

## S4 method for signature 'numeric,cyl_vonmises'
rcylcop(n, copula)

## S4 method for signature 'matrix,cyl_vonmises'
dcylcop(u, copula)

## S4 method for signature 'matrix,cyl_vonmises'
pcylcop(u, copula)
```

Arguments

u	matrix (or vector) of numeric values in $[0,1]^2$, containing as first column the circular (periodic) and as second the linear dimension
copula	R object of class 'cyl_copula'. or 'Copula' (package '**copula**', only 2-dimensional).
n	number of random samples to be generated with `rcylcop()`.
log	logical indicating if the logarithm of the density should be returned (`dcylcop()`).

Details

For 'Copula' objects, `pcylcop()` and `rcylcop()` just call the functions of the '**copula**' package `pCopula()` and `rCopula()`, respectively. The density is, however, calculated differently in `dcylcop()` and `dCopula()`. The difference is that `copula::dCopula()` will return a density of 0 for points on the boundary of the unit square, whereas `dcylcop()` will return the correct density on the boundaries for both 'cyl_copula' and 'Copula' objects.

Value

The functions `pcylcop()` and `dcylcop()` give a vector of length `nrow(u)` containing the distribution and the density, respectively, at the corresponding values of `u`. The function `rcylcop()` generates a matrix with 2 columns and `n` rows containing the random samples.

References

Nelsen RB (2006). *An Introduction to Copulas*, volume 139 of *Lecture Notes in Statistics*. Springer New York, New York, NY. ISBN 978-0-387-98623-4, doi:10.1007/9781475730760, https://link.springer.com/book/10.1007/978-1-4757-3076-0.

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/.

See Also

`copula::dCopula()`, `copula::pCopula()`, `copula::rCopula()`.

Examples

```
set.seed(123)

cop <- cyl_quadsec(0.1)
rcylcop(5, cop)
pcylcop(c(0.3, 0.1), cop)
pcylcop(rbind(c(0.3, 0.1), c(0.2, 1)), cop)

cop <- cyl_rot_combine(copula::frankCopula(2), shift = TRUE)
dcylcop(u = rbind(c(0.1, 0.4), c(1.0, 0.2)), copula = cop)
dcylcop(c(0.1, 0.3), cyl_quadsec(0.1), log = TRUE)

cop <- copula::normalCopula(0.3)
copula::dCopula(c(.Machine$double.eps,0.2),cop)
```

```
copula::dCopula(c(0,0.2),cop)
dcylcop(c(.Machine$double.eps,0.2),cop)
dcylcop(c(0,0.2),cop)
```

`cylcop-deprecated` *Deprecated functions*

Description

These functions are provided for compatibility with older version of the cylcop package. They may eventually be completely

Usage

```
scat_plot(...)
```

Arguments

`...`	Parameters to be passed to the new versions of the functions

Details

`scat_plot()`	is replaced by `plot_joint_scat()`
`traj_plot()`	is replaced by `plot_track()`
`circ_plot()`	is replaced by `plot_joint_circ()`
`cop_scat_plot()`	is replaced by `plot_cop_scat()`
`cop_plot()`	is replaced by `plot_cop_surf()`
`make_traj()`	is replaced by `traj_sim()`
`qmixedvonmises()`	is replaced by `qvonmisesmix()`
`mle.mixedvonmises()`	is replaced by `mle.vonmisesmix()`

`cylcop_get_option` *Get Package Options*

Description

Currently the only option (`"silent"`) is to toggle verbosity on or off.

Usage

```
cylcop_get_option(option = NULL)
```

Arguments

`option`	character string, the name of the option.

Value

The numeric value of option. If no argument is provided, a list of all options is printed.

See Also

`cylcop_set_option()`

Examples

```
cylcop_get_option("silent")
cylcop_get_option()
```

`cylcop_set_option` *Set Package Options*

Description

Currently the only option is to toggle verbosity on or off.

Usage

```
cylcop_set_option(silent = FALSE)
```

Arguments

`silent`	logical, suppress all sounds and messages.

Value

No output, only side effects.

See Also

`cylcop_get_option()`

Examples

```
cylcop_set_option(silent = FALSE)
```

cyl_copula-class *An S4 Class of Bivariate Copulas on the Cylinder*

Description

The class 'cyl_copula' follows somewhat the structure of the class 'Copula' of the package '**copula**'. It contains circular-linear copulas.

Slots

name character string holding the name of the copula.

parameters numeric vector holding the parameter values.

param.names character vector holding the parameter names.

param.lowbnd numeric vector holding the lower bounds of the parameters.

param.upbnd numeric vector holding the upper bounds of the parameters.

Extended by

'cyl_copula' is extended by the following classes:

- 'cyl_vonmises': von Mises copulas.
- 'cyl_quadsec': Copulas with quadratic sections.
- 'cyl_cubsec': Copulas with cubic sections.
- 'cyl_rot_combine': Linear combinations of copulas and their 180 degree rotations.
- 'cyl_rect_combine': Rectangular patchwork copulas.

Objects from the Class

Objects are created by the functions cyl_vonmises(), cyl_quadsec(), cyl_cubsec(), cyl_rot_combine(), and cyl_rect_combine().

References

Hodel FH, Fieberg JR (2022). "Circular-Linear Copulae for Animal Movement Data." *Methods in Ecology and Evolution*. doi:10.1111/2041210X.13821.

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/.

Examples

```
cop <- cyl_quadsec(0.1)
is(cop)
```

`cyl_cubsec` *Construction of '`cyl_cubsec`' Objects*

Description

Constructs a circular-linear copula with cubic sections of class '`cyl_cubsec`'.

Usage

```
cyl_cubsec(a = 1/(2 * pi), b = 1/(2 * pi))
```

Arguments

`a`	numeric value of the first parameter of the copula. It must be in $[-1/(2\pi)), 1/(2\pi))]$.
`b`	numeric value of the second parameter of the copula. It must be in $[-1/(2\pi)), 1/(2\pi))]$.

Value

An R object of class '`cyl_cubsec`'.

References

Nelsen RB, Quesada-Molina JJ, Rodr\'iguez-Lallena JA (1997). "Bivariate copulas with cubic sections." *Journal of Nonparametric Statistics*, **7**(3), 205–220. ISSN 10485252, doi:10.1080/10485259708832700.

Hodel FH, Fieberg JR (2022). "Circular-Linear Copulae for Animal Movement Data." *Methods in Ecology and Evolution*. doi:10.1111/2041210X.13821.

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, `https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/`.

Examples

```
cop <- cyl_cubsec(a = 0.1, b = -0.1)
if(interactive()){
 plot_cop_surf(copula = cop, type = "pdf", plot_type = "ggplot")
}
```

cyl_cubsec-class *An S4 Class of Bivariate Copulas with Cubic Sections*

Description

This class contains bivariate circular-linear copulas with cubic sections in the linear dimension. They are periodic in the circular dimension, u, and symmetric with respect to $u = 0.5$. Therefore, they can capture correlation in data where there is symmetry between positive and negative angles. These copulas are described by two parameters, `a` and `b`.

Slots

`name` character string holding the name of the copula.

`parameters` numeric vector holding the parameter values.

`param.names` character vector holding the parameter names.

`param.lowbnd` numeric vector holding the lower bounds of the parameters.

`param.upbnd` numeric vector holding the upper bounds of the parameters.

Objects from the Class

Objects are created by `cyl_cubsec()`.

Extends

Class '`cyl_cubsec`' extends class '`cyl_copula`'.

References

Nelsen RB, Quesada-Molina JJ, Rodr\'iguez-Lallena JA (1997). “Bivariate copulas with cubic sections.” *Journal of Nonparametric Statistics*, **7**(3), 205–220. ISSN 10485252, doi:10.1080/10485259708832700.

Hodel FH, Fieberg JR (2022). “Circular-Linear Copulae for Animal Movement Data.” *Methods in Ecology and Evolution*. doi:10.1111/2041210X.13821.

Hodel FH, Fieberg JR (2021). “Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry.” *bioRxiv*. doi:10.1101/2021.07.14.452253, `https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/`.

`cyl_quadsec` *Construction of '`cyl_quadsec`' Objects*

Description

Constructs a circular-linear copula with cubic sections of class '`cyl_quadsec`'.

Usage

```
cyl_quadsec(a = 1/(2 * pi))
```

Arguments

`a`	numeric value of the parameter of the copula. It must be in $[-1/(2\pi)), 1/(2\pi))]$.

Value

An R object of class '`cyl_quadsec`'.

References

Quesada-Molina JJ, Rodr\'iguez-Lallena JA (1995). "Bivariate copulas with quadratic sections." *Journal of Nonparametric Statistics*, **5**(4), 323–337. ISSN 10290311, doi:10.1080/10485259508832652.

Hodel FH, Fieberg JR (2022). "Circular-Linear Copulae for Animal Movement Data." *Methods in Ecology and Evolution*. doi:10.1111/2041210X.13821.

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, `https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/`.

Examples

```
cop <- cyl_quadsec(a = 0.1)
if(interactive()){
 plot_cop_surf(copula = cop, type = "pdf", plot_type = "ggplot")
}
```

`cyl_quadsec-class` *An S4 Class of Bivariate Copulas with Quadratic Sections*

Description

This class contains bivariate circular-linear copulas with quadratic sections in the linear dimension. They are periodic in the circular dimension, u, and symmetric with respect to $u = 0.5$. Therefore, they can capture correlation in data where there is symmetry between positive and negative angles. These copulas are described by one parameter, `a`.

Slots

`name` character string holding the name of the copula.

`parameters` numeric vector holding the parameter value.

`param.names` character vector holding the parameter name.

`param.lowbnd` numeric vector holding the lower bound of the parameter.

`param.upbnd` numeric vector holding the upper bound of the parameter.

Objects from the Class

Objects are created by `cyl_quadsec()`.

Extends

Class '`cyl_quadsec`' extends class '`cyl_copula`'.

References

Quesada-Molina JJ, Rodr\'iguez-Lallena JA (1995). "Bivariate copulas with quadratic sections." *Journal of Nonparametric Statistics*, **5**(4), 323–337. ISSN 10290311, doi:10.1080/10485259508832652.

Hodel FH, Fieberg JR (2022). "Circular-Linear Copulae for Animal Movement Data." *Methods in Ecology and Evolution*. doi:10.1111/2041210X.13821.

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, `https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/`.

cyl_rect_combine *Construction of* 'cyl_rect_combine' *Objects*

Description

Constructs a circular-linear copula of class 'cyl_rect_combine' from a rectangular patchwork of copulas.

Usage

```
cyl_rect_combine(
  copula,
  background = indepCopula(),
  low_rect = c(0, 0.5),
  up_rect = "symmetric",
  flip_up = TRUE
)
```

Arguments

copula	'Copula' object of the package '**copula**' or 'cyl_vonmises' object, the copula in the rectangles.
background	'cyl_copula' or 'Copula' object of the package '**copula**', the copula where no rectangles overlay the unit square. If this copula is not symmetric, the overall cyl_rect_combine-copula will also not be symmetric.
low_rect	numeric vector of length 2 containing the lower and upper edge (u-value) of the lower rectangle.
up_rect	numeric vector of length 2 containing the lower and upper edge (u-value) of the upper rectangle, or the character string "symmetric" if it should be the mirror image (with respect to u=0.5) of the lower rectangle.
flip_up	logical value indicating whether the copula (copula) is rotated 90 degrees in the upper (flip_up = TRUE) or lower rectangle.

Value

An R object of class 'cyl_rect_combine'.

References

Durante F, Saminger-Platz S, Sarkoci P (2009). "Rectangular patchwork for bivariate copulas and tail dependence." *Communications in Statistics - Theory and Methods*, **38**(15), 2515–2527. ISSN 03610926, doi:10.1080/03610920802571203.

Hodel FH, Fieberg JR (2022). "Circular-Linear Copulae for Animal Movement Data." *Methods in Ecology and Evolution*. doi:10.1111/2041210X.13821.

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/.

Examples

```
#symmetric rectangles spanning entire unit square
cop <- cyl_rect_combine(copula::frankCopula(2))
if(interactive()){
 plot_cop_surf(copula = cop, type = "pdf", plot_type = "ggplot", resolution = 20)
}

#symmetric rectangles, independence copula as background
cop <- cyl_rect_combine(copula::frankCopula(2),
  low_rect = c(0, 0.3),
  up_rect = "symmetric",
  flip_up = FALSE
)
if(interactive()){
 plot_cop_surf(copula = cop, type = "pdf", plot_type = "ggplot", resolution = 20)
}

#symmetric rectangles, cy_quadsec-copula as background
cop <- cyl_rect_combine(copula::normalCopula(0.3),
  low_rect = c(0.1, 0.4),
  up_rect = "symmetric",
  background = cyl_quadsec(-0.1)
)
if(interactive()){
 plot_cop_surf(copula = cop, type = "pdf", plot_type = "ggplot", resolution = 20)
}

#asymmetric rectangles, von Mises copula as background.
#!!Not a symmetric circular linear copula!!
cop <- cyl_rect_combine(copula::normalCopula(0.3),
  low_rect = c(0.1, 0.4),
  up_rect = c(0.5, 0.7),
  background = cyl_vonmises(mu = pi, kappa = 0.3)
)
if(interactive()){
 plot_cop_surf(copula = cop, type = "pdf", plot_type = "ggplot", resolution = 20)
}
```

`cyl_rect_combine-class` *An S4 Class of Circular-Linear Copulas Generated from a Rectangular Patchwork*

Description

This class contains bivariate circular-linear copulas generated from linear-linear bivariate '`Copula`' objects of the package '**copula**' or circular-linear copulas of class '`cyl_copula`'. 2 non-overlapping rectangles are laid over the unit square, both have width 1 in v-direction. In the area covered by the first rectangle, the copula is derived from a linear-linear bivariate 'Copula' object. Rectangle 2 contains the same copula as rectangle 1, but 90 degrees rotated. In the area not covered by the rectangles, the "background", the copula is derived from a circular-linear 'cyl_copula' object. The copula regions are combined in a way that the overall result on the entire unit square is also a copula.

Details

With appropriate choices of the rectangles this results in copulas that are periodic in u-direction (and not in v-direction) and therefore are circular-linear. When the 2 rectangles are mirror images with respect to $u = 0.5$, the resulting overall copula is symmetric with respect to $u = 0.5$, i.e. there is symmetry between positive and negative angles.

Note that as "background copula", we can also chose a linear-linear copula, the overall result will then, however, not be a symmetric circular linear copula.

Slots

`name` character string holding the name of the copula.

`parameters` numeric vector holding the parameter values.

`param.names` character vector the parameter names.

`param.lowbnd` numeric vector holding the lower bounds of the parameters.

`param.upbnd` numeric vector holding the upper bounds of the parameters.

`sym.cop` 'Copula' object of the package '**copula**' or 'cyl_vonmises' object. The copula in the rectangles.

`background.cop` 'cyl_vonmises' or 'Copula' object of the package '**copula**', the copula where no rectangles overlay the unit square. If this copula is not symmetric, the overall `cyl_rect_combine`-copula will also not be symmetric.

`flip_up` logical value indicating whether the copula (`sym.cop`) is rotated 90 degrees in the upper or lower rectangle.

`sym_rect` logical value indicating whether the upper rectangle was forced to be a mirror image of the lower one with respect to u=0.5 at the construction of the object.

Objects from the Class

Objects are created by `cyl_rect_combine()`.

Extends

Class '`cyl_rect_combine`' extends class 'Copula'.

References

Durante F, Saminger-Platz S, Sarkoci P (2009). "Rectangular patchwork for bivariate copulas and tail dependence." *Communications in Statistics - Theory and Methods*, **38**(15), 2515–2527. ISSN 03610926, doi:10.1080/03610920802571203.

Hodel FH, Fieberg JR (2022). "Circular-Linear Copulae for Animal Movement Data." *Methods in Ecology and Evolution*. doi:10.1111/2041210X.13821.

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/.

`cyl_rot_combine` *Construction of* 'cyl_rot_combine' *Objects*

Description

Constructs a circular-linear copula of class 'cyl_rot_combine' from linear combinations of copulas.

Usage

```
cyl_rot_combine(copula, shift = FALSE)
```

Arguments

copula	linear-linear 2-dimensional 'Copula' object of the package '**copula**'.
shift	logical value indicating whether the (u-periodic) copula should be shifted by 0.5 in u direction.

Value

An R object of class 'cyl_rot_combine'.

References

Nelsen RB (2006). *An Introduction to Copulas*, volume 139 of *Lecture Notes in Statistics*. Springer New York, New York, NY. ISBN 978-0-387-98623-4, doi:10.1007/9781475730760, https://link.springer.com/book/10.1007/978-1-4757-3076-0.

Hodel FH, Fieberg JR (2022). "Circular-Linear Copulae for Animal Movement Data." *Methods in Ecology and Evolution*. doi:10.1111/2041210X.13821.

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/.

Examples

```
cop <- cyl_rot_combine(copula = copula::frankCopula(param = 3), shift = TRUE)
if(interactive()){
 plot_cop_surf(copula = cop, type = "pdf", plot_type = "ggplot", resolution = 20)
}

cop <- cyl_rot_combine(copula = copula::claytonCopula(param = 10), shift = FALSE)
if(interactive()){
 plot_cop_surf(copula = cop, type = "pdf", plot_type = "ggplot", resolution = 20)
}
```

`cyl_rot_combine-class` *An S4 Class of Circular-Linear Copulas generated from Linear Combinations of Copulas*

Description

This class contains bivariate circular-linear copulas, generated from linear-linear bivariate 'Copula' objects of the package '**copula**', by taking the arithmetic mean of the original copula and the 90 deg rotated copula. This results in copulas that are periodic in the circular dimension, u, and symmetric with respect to $u = 0.5$, i.e. positive and negative angles.

Slots

`name` character string holding the name of the copula.

`parameters` numeric vector holding the parameter values.

`param.names` character vector the parameter names.

`param.lowbnd` numeric vector holding the lower bounds of the parameters.

`param.upbnd` numeric vector holding the upper bounds of the parameters.

`orig.cop` linear-linear 2-dimensional 'Copula' object of the package '**copula**'.

`shift` logical value indicating whether the (u-periodic) copula should be shifted by 0.5 in u direction.

Objects from the Class

Objects are created by `cyl_rot_combine()`.

Extends

Class '`cyl_rot_combine`' extends class '`Copula`'.

References

Nelsen RB (2006). *An Introduction to Copulas*, volume 139 of *Lecture Notes in Statistics*. Springer New York, New York, NY. ISBN 978-0-387-98623-4, doi:10.1007/9781475730760, https://link.springer.com/book/10.1007/978-1-4757-3076-0.

Hodel FH, Fieberg JR (2022). "Circular-Linear Copulae for Animal Movement Data." *Methods in Ecology and Evolution*. doi:10.1111/2041210X.13821.

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/.

cyl_vonmises *Construction of 'cyl_vonmises' Objects*

Description

Constructs a circular-linear von Mises copula according to Johnson and Wehrly (1978) of class 'cyl_vonmises'.

Usage

```
cyl_vonmises(mu = 0, kappa = 1, flip = FALSE)
```

Arguments

mu	numeric value giving the mean of the von Mises function used to construct the copula.
kappa	numeric value giving the concentration of the von Mises function used to construct the copula.
flip	logical value indicating whether the copula should be rotated 90 degrees to capture negative correlation.

Value

An R object of class 'cyl_vonmises'.

References

Johnson RA, Wehrly TE (1978). "Some Angular-Linear Distributions and Related Regression Models." *Journal of the American Statistical Association ISSN:*, **73**(363), 602–606. ISSN 00401706, doi:10.2307/1270921.

Hodel FH, Fieberg JR (2022). "Circular-Linear Copulae for Animal Movement Data." *Methods in Ecology and Evolution*. doi:10.1111/2041210X.13821.

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/.

Examples

```
cop <- cyl_vonmises(mu=pi, kappa=10, flip = TRUE)
if(interactive()){
 plot_cop_surf(copula = cop, type = "pdf", plot_type = "ggplot", resolution = 20)
}

cop <- cyl_vonmises(mu=0, kappa=8, flip = FALSE)
if(interactive()){
 plot_cop_surf(copula = cop, type = "pdf", plot_type = "ggplot", resolution = 20)
}
```

cyl_vonmises-class *An S4 Class of Bivariate von Mises Copulas*

Description

This class contains circular-linear copulas that are based on the approach by Johnson and Wehrly (1978) with a von Mises periodic function. They are periodic in the circular dimension, u, but not symmetric with respect to $u = 0.5$ i.e. there is no symmetry between positive and negative angles.

Slots

`name` character string holding the name of the copula.

`parameters` numeric vector holding the parameter values.

`param.names` character vector holding the parameter names.

`param.lowbnd` numeric vector holding the lower bounds of the parameters.

`param.upbnd` numeric vector holding the upper bounds of the parameters.

`flip` logical value indicating whether the copula should be rotated 90 degrees to capture negative correlation.

Objects from the Class

Objects are created by `cyl_vonmises()`.

Extends

Class '`cyl_vonmises`' extends class '`cyl_copula`'.

References

Johnson RA, Wehrly TE (1978). "Some Angular-Linear Distributions and Related Regression Models." *Journal of the American Statistical Association ISSN:*, **73**(363), 602–606. ISSN 00401706, doi:10.2307/1270921.

Hodel FH, Fieberg JR (2022). "Circular-Linear Copulae for Animal Movement Data." *Methods in Ecology and Evolution*. doi:10.1111/2041210X.13821.

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, `https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/`.

dens — *Density, Distribution, Random Number Generation and Quantiles of Kernel Density Estimates*

Description

Calculate the density (`ddens()`), the distribution (`pdens()`), the quantiles (`qdens()`) and generate random samples (`rdens()`) of a kernel density estimate as returned by `fit_angle()` or `fit_steplength()`.

Usage

```
rdens(n, density)

ddens(x, density)

pdens(x, density)

qdens(p, density)
```

Arguments

`n`	integer value, the number of random samples to be generated with `rdens()`.
`density`	a '`density`' object (for linear kernel density estimates) or a '`density.circular`' object (for circular kernel density estimates) containing information about the kernel density estimate. These objects can be obtained using `fit_angle(..., parametric = FALSE)` or `fit_steplength(..., parametric = FALSE)`.
`x`	numeric vector giving the points where the density or distribution function is evaluated.
`p`	numeric vector giving the probabilities where the quantile function is evaluated.

Value

`ddens()` and `pdens()` give a vector of length `length(x)` containing the density or distribution function at the corresponding values of `x`. `qdens()` gives a vector of length `length(p)` containing the quantiles at the corresponding values of `p`. The function `rdens()` generates a vector of length `n` containing the random samples.

See Also

`fit_angle()`, `fit_steplength()`, `fit_steplength()`.

Examples

```
set.seed(123)

steps <- rweibull(10, shape=3)
dens <- fit_steplength(x = steps, parametric = FALSE)
ddens(c(0.1,0.3), dens)
pdens(c(0.1,0.3), dens)
qdens(c(0.1,0.3), dens)
rdens(4, dens)

angles <- full2half_circ(
  circular::rvonmises(10, mu = circular::circular(0), kappa = 2)
)
dens <- fit_angle(theta = angles, parametric = FALSE)
ddens(c(0.1,0.3), dens)
pdens(c(0.1,0.3), dens)
qdens(c(0.1,0.3), dens)
rdens(4, dens)
```

---

fit_angle *Fit a Circular Univariate Distribution*

Description

This function finds parameter estimates of the marginal circular distribution (with potentially fixed mean), or gives a kernel density estimate using a von Mises smoothing kernel.

Usage

```
fit_angle(
  theta,
  parametric = c("vonmises", "wrappedcauchy", "vonmisesmix", FALSE),
  bandwidth = NULL,
  mu = NULL,
  ncomp = 2
)
```

Arguments

theta	numeric vector of angles in $[-\pi, \pi)$.
parametric	either a character string describing what distribution should be fitted ("vonmises", "wrappedcauchy", or "vonmisesmix"), or the logical FALSE if a non-parametric estimation (kernel density) should be made.
bandwidth	If parametric = FALSE, the numeric value of the kernel density bandwidth. Default is cylcop::opt_circ_bw(theta, "nrd").
mu	(optional) numeric vector, fixed mean direction(s) of the parametric distribution.
ncomp	integer, number of components of the mixed von Mises distribution. Only has an effect if parametric="vonmisesmix".

Value

If a parametric estimate is made, a list is returned containing the estimated parameters, their standard errors (if available), the log-likelihood, the AIC and the name of the distribution. If a non-parametric estimate is made, the output is a 'density.circular' object obtained with the function circular::density.circular() of the '**circular**' package.

See Also

circular::density.circular(), fit_angle(), opt_circ_bw().

Examples

```
set.seed(123)

silent_curr <- cylcop_get_option("silent")
cylcop_set_option(silent = TRUE)

n <- 10 #n (number of samples) is set small for performance.

angles <- rvonmisesmix(n,
  mu = c(0, pi),
  kappa = c(2,1),
  prop = c(0.5, 0.5)
)

bw <- opt_circ_bw(theta = angles,
  method="nrd",
  kappa.est = "trigmoments"
)
dens_non_param <- fit_angle(theta = angles,
  parametric = FALSE,
  bandwidth = bw
)

param_estimate <- fit_angle(theta = angles,
  parametric = "vonmisesmix"
)
param_estimate_fixed_mean <- fit_angle(theta = angles,
  parametric = "vonmisesmix",
  mu = c(0, pi),
  ncomp =2
)

cylcop_set_option(silent = silent_curr)
```

---

fit_cylcop_cor *Estimate Copula Parameters from Correlation Measures*

Description

This function implements a simple search of the parameter space of a 'cyl_copula' object to find the parameter values that lead to a correlation that is closest to the correlation in the data (theta and x). In some special cases of 'cyl_rect_combine' copulas, the parameter can be obtained analytically from Kendall's tau of the data.

Usage

```
fit_cylcop_cor(copula, theta, x, acc = NULL, n = 10000, method, ...)

## S4 method for signature 'cyl_vonmises'
fit_cylcop_cor(copula, theta, x, acc, n, method = "cor_cyl")

## S4 method for signature 'cyl_quadsec'
fit_cylcop_cor(copula, theta, x, acc, n, method = "cor_cyl")

## S4 method for signature 'cyl_cubsec'
fit_cylcop_cor(
  copula,
  theta,
  x,
  acc,
  n,
  method = "cor_cyl",
  parameter = "both"
)

## S4 method for signature 'cyl_rot_combine'
fit_cylcop_cor(copula, theta, x, acc, n, method = "mi_cyl")

## S4 method for signature 'cyl_rect_combine'
fit_cylcop_cor(copula, theta, x, acc, n, method = "tau", background = FALSE)

optCor(copula, theta, x, acc = NULL, n = 10000, method, ...)
```

Arguments

copula	R object of class 'cyl_copula'.
theta	numeric vector of angles (measurements of a circular variable).
x	numeric vector of step lengths (measurements of a linear variable).
acc	numeric value, the interval of the copula parameter at which to evaluate the correlation.
n	numeric value, the number of sample points at each optimization step.
method	character string describing what correlation metric to use. Either a rank-based circular-linear correlation coefficient ("cor_cyl"), mutual information ("mi_cyl"), or Kendall's tau ("tau").
...	Additional parameters (see individual methods).

parameter — For 'cyl_cubsec' copulas: A character string specifying which parameter of the copula to optimize, "a", "b", or "both"

background — For 'cyl_rect_combine' copulas : A logical value describing whether to optimize the parameter of the background copula, (background = TRUE) or the one of the copula in the rectangles (background = FALSE).

Details

The code assumes that the correlation captured by the copula increases monotonously with the copula parameter values. It starts with a parameter value close to the minimum for that copula and calculates the correlation for a sample of size n from that copula. Next, the parameter is doubled and again the correlation for a sample of size n calculated. After this exponential search pattern, a binary search is implemented similarly between the bounds found with the exponential search. For this binary search, the interval between those bounds is split into small intervals of length acc. Thus, smaller values of acc lead to higher accuracy.

If a 'cyl_rect_combine' copula has rectangles spanning the entire unit square and as background the independence copula, Kendall's tau can be used to analytically calculate the parameter value leading to the correlation of the data. No search is necessary in this case. This makes it the recommended method to use for those 'cyl_rect_combine' copulas. optCor() is an alias for fit_cylcop_cor.

See also individual methods (below) for more detailed explanations.

Value

numeric vector containing the estimated parameter value(s).

Functions

- fit_cylcop_cor(cyl_vonmises): only parameter "kappa" can be optimized, since parameter "mu" does not influence the correlation.
- fit_cylcop_cor(cyl_quadsec): the absolute value of the parameter is optimized, positive and negative values give the same correlation.
- fit_cylcop_cor(cyl_cubsec): optimization of parameters, "a" and "b", can be done separately or simultaneously.
- fit_cylcop_cor(cyl_rot_combine): the circular-linear correlation coefficient will give a value close to 0 for any parameter value. It therefore only makes sense to use method = "mi_cyl" for the optimization.
- fit_cylcop_cor(cyl_rect_combine): if the rectangles span the entire unit square and the background is the independence copula, it is recommended to use method = "tau", since this calculates the copula parameter analytically. If there is a background copula, other than the independence copula, its parameter can be optimized by setting background=TRUE.

References

Hodel FH, Fieberg JR (2022). "Circular-Linear Copulae for Animal Movement Data." *Methods in Ecology and Evolution*. doi:10.1111/2041210X.13821.

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/.

See Also

mi_cyl(), cor_cyl(), fit_cylcop_ml(), opt_auto(), copula::fitCopula().

Examples

```
set.seed(123)

sample <- rcylcop(100, cyl_rect_combine(copula::frankCopula(2)))
fit_cylcop_cor(cyl_rect_combine(copula::frankCopula()),
  theta = sample[,1],
  x = sample[,2],
  method = "tau"
)

fit_cylcop_cor(cyl_rect_combine(copula::frankCopula()),
  theta = sample[,1],
  x = sample[,2],
  method = "mi_cyl",
  n = 100
)

fit_cylcop_cor(cyl_rect_combine(copula::claytonCopula()),
  theta = sample[,1],
  x = sample[,2],
  method = "tau"
)

fit_cylcop_cor(cyl_quadsec(), theta = sample[,1], x = sample[,2], method = "mi_cyl")
fit_cylcop_cor(cyl_quadsec(), theta = sample[,1], x = sample[,2], method = "cor_cyl")
fit_cylcop_cor(cyl_quadsec(),
  theta = sample[,1],
  x = sample[,2],
  method = "cor_cyl",
  n = 100,
  acc = 0.001
)

optCor(cyl_quadsec(),
 theta = sample[,1],
 x = sample[,2],
 method = "mi_cyl")
```

`fit_cylcop_ml` *Estimate Parameters of a Circular-Linear Copula According to Maximum Likelihood*

Description

The code of this function is based on `copula::fitCopula()`. A circular-linear copula is fit to a set of bivariate observations.

Usage

```
fit_cylcop_ml(
  copula,
  theta,
  x,
  parameters = NULL,
  start = NULL,
  lower = NULL,
  upper = NULL,
  optim.method = "L-BFGS-B",
  optim.control = list(maxit = 100),
  estimate.variance = FALSE,
  traceOpt = FALSE
)

optML(
  copula,
  theta,
  x,
  parameters = NULL,
  start = NULL,
  lower = NULL,
  upper = NULL,
  optim.method = "L-BFGS-B",
  optim.control = list(maxit = 100),
  estimate.variance = FALSE,
  traceOpt = FALSE
)
```

Arguments

`copula`	R object of class '`cyl_copula`'.
`theta`	numeric vector of angles (measurements of a circular variable) or "circular" component of pseudo-observations.
`x`	numeric vector of step lengths (measurements of a linear variable) or "linear" component of pseudo-observations.

parameters	vector of character strings holding the names of the parameters to be optimized. These can be any parameters in copula@parameters. Default is to optimize the first 2 parameters or the single parameter if copula only has 1.
start	vector of starting values of the parameters. Default is to take the starting values from copula.
lower	(optional) vector of lower bounds of the parameters.
upper	(optional) vector of upper bounds of the parameters.
optim.method	character string, optimizer used in optim(), can be "Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN", or "Brent". Default is "L-BFGS-B".
optim.control	list of additional controls passed to optim().
estimate.variance	logical value, denoting whether to include an estimate of the variance (NOT YET IMPLEMENTED).
traceOpt	logical value, whether to print information regarding convergence, current values, etc. during the optimization process.

Details

The data is first converted to pseudo observations to which the copula is then fit. Therefore, the result of the optimization will be exactly the same whether measurements (theta=theta and x=x) or pseudo observations (theta=copula::pobs(theta,x)[,1] and x=copula::pobs(theta,x)[,2]) are provided. If you wish to fit parameters of a 'Copula' object (package '**copula**'), use the function copula::fitCopula(). optML() is an alias for fit_cylcop_ml.

Value

A list of length 3 containing the same type of 'cyl_copula' object as copula, but with optimized parameters, the log-likelihood and the AIC.

References

Hodel FH, Fieberg JR (2022). "Circular-Linear Copulae for Animal Movement Data." *Methods in Ecology and Evolution*. doi:10.1111/2041210X.13821.

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/.

See Also

copula::fitCopula(), fit_cylcop_cor(), opt_auto().

Examples

```
set.seed(123)

sample <- rcylcop(100,cyl_quadsec(0.1))
fit_cylcop_ml(copula = cyl_quadsec(),
  theta = sample[,1],
```

```
  x = sample[,2],
  parameters = "a",
  start = 0
)
fit_cylcop_ml(copula = cyl_rect_combine(copula::frankCopula()),
  theta = sample[,1],
  x = sample[,2],
  parameters = "alpha",
  start = 1
)


sample <- rjoint(
  n = 100,
  copula = cyl_cubsec(0.1, -0.08),
  marginal_1 = list(name = "vonmisesmix", coef = list(
     mu = c(pi, 0),
     kappa = c(2, 5),
     prop = c(0.3, 0.7)
    )),
  marginal_2 = list(name = "exp", coef = list(0.3))
  )
  fit_cylcop_ml(copula = cyl_cubsec(),
  theta = sample[,1],
  x = sample[,2],
  parameters = c("a","b"),
  start = c(0,0),
  upper= c(0.1, 1/(2*pi))
)

optML(copula = cyl_quadsec(),
  theta = sample[,1],
  x = sample[,2],
  parameters = "a",
  start = 0
)
```

`fit_steplength` *Fit a Linear Univariate Distribution*

Description

This function finds parameter estimates of the marginal linear distribution, or gives a kernel density estimate using a Gaussian smoothing kernel.

Usage

```
fit_steplength(
  x,
  parametric = c("beta", "cauchy", "chi-squared", "chisq", "exponential", "exp", "gamma",
```

```
  "lognormal", "lnorm", "lognorm", "logistic", "normal", "t", "weibull", "normalmix",
   "weibullmix", "gammamix", "lnormmix", FALSE),
  start = NULL,
  bandwidth = NULL,
  ncomp = 2
)
```

Arguments

x	numeric vector of measurements of a linear random variable in $[0, \infty)$.
parametric	either a character string describing what distribution should be fitted ("beta", "cauchy", "chi-squared", "exponential", "gamma", "lognormal", "logistic", "normal", "t", "weibull","normalmix", "weibullmix", "gammamix", or "lnormmix"), or the logical FALSE if a non-parametric estimation (kernel density) should be made.
start	(optional, except when parametric = "chi-squared") named list containing the parameters to be optimized with initial values.
bandwidth	numeric value for the kernel density bandwidth. Default is cylcop::opt_lin_bw(x, "nrd").
ncomp	integer, number of components of the mixed distribution. Only has an effect if parametric %in% c("normalmix", "weibullmix", "gammamix","lnormmix").

Value

If a parametric estimate is made, a list is returned containing the estimated parameters, their standard errors, the log-likelihood, the AIC and the name of the distribution. If a non-parametric estimate is made, the output is a a 'density' object, which is obtained with the function GoFKernel::density.reflected() of the '**GoFKernel**' package.

See Also

GoFKernel::density.reflected(), fit_angle(), opt_lin_bw().

Examples

```
require(graphics)
set.seed(123)

silent_curr <- cylcop_get_option("silent")
cylcop_set_option(silent = TRUE)

n <- 100 #n (number of samples) is set small for performance.

x <- rweibull(n, shape = 10)

dens_non_param <- fit_steplength(x = x, parametric = FALSE)
weibull <- fit_steplength(x = x, parametric = "weibull")
gamma <- fit_steplength(x = x, parametric = "gamma")
chisq <- fit_steplength(x = x, parametric = "chi-squared", start = list(df = 1))
```

```
true_dens <- dweibull(seq(0, max(x), length.out = 200),
  shape = 10
)
dens_weibull <- dweibull(seq(0, max(x),length.out = 200),
  shape = weibull$coef$shape,
  scale = weibull$coef$scale
)
dens_gamma <- dgamma(seq(0, max(x),length.out = 200),
  shape = gamma$coef$shape,
  rate = gamma$coef$rate
)
dens_chisq <- dchisq(seq(0, max(x),length.out = 200),
  df = chisq$coef$df
)

plot(seq(0,max(x),length.out = 200), true_dens, type = "l")
lines(dens_non_param$x, dens_non_param$y, col = "red")
lines(seq(0,max(x),length.out = 200), dens_weibull, col = "green")
lines(seq(0,max(x),length.out = 200), dens_gamma, col = "blue")
lines(seq(0,max(x),length.out = 200), dens_chisq, col = "cyan")

cylcop_set_option(silent = silent_curr)
```

`full2half_circ` *Convert Angle from Full Circle to Half Circle*

Description

Converts an angle from the full circle (i.e. in the interval $[0, 2\pi)$) to an angle on the half circle (i.e. in the interval $[-\pi, \pi)$).

Usage

```
full2half_circ(angle)
```

Arguments

`angle`	numeric value of an angle or a `circular`-object in $[0, 2\pi)$.

Value

The numeric value of the angle in $[-\pi, \pi)$.

Examples

```
full2half_circ(0 * pi) / pi
full2half_circ(0.5 * pi) / pi
full2half_circ(1 * pi) / pi
full2half_circ(1.5 * pi) / pi
full2half_circ(2 * pi) / pi
```

---

gammamix — *Density, Distribution, Quantiles and Random Number Generation for the mixed gamma distribution*

Description

The number of components in the mixed gamma distribution is specified by the length of the parameter vectors. The quantiles are numerically obtained from the distribution function using monotone cubic splines.

Usage

```
rgammamix(n, shape, rate = 1, scale = 1/rate, prop)

dgammamix(x, shape, rate = 1, scale = 1/rate, prop)

pgammamix(q, shape, rate = 1, scale = 1/rate, prop)

qgammamix(p, shape, rate = 1, scale = 1/rate, prop)
```

Arguments

`n`	integer value, the number of random samples to be generated with `rgammamix()`.
`shape`	numeric vector holding the shape parameter of the components.
`rate`	numeric vector an alternative way to specify the scale (`scale = 1 / rate`).
`scale`	numeric vector holding the scale parameter of the components.
`prop`	numeric vector, holding the mixing proportions of the components.
`x`	numeric vector giving the points where the density function is evaluated.
`q`	numeric vector giving the quantiles where the distribution function is evaluated.
`p`	numeric vector giving the probabilities where the quantile function is evaluated.

Value

- `dgammamix()` gives a vector of length `length(x)` containing the density at `x`.
- `pgammamix()` gives a vector of length `length(q)` containing the distribution function at the corresponding values of `q`.
- `qgammamix()` gives a vector of length `length(p)` containing the quantiles at the corresponding values of `p`.
- `rgammamix()` generates a vector of length `n` containing the random samples.

Examples

```
rgammamix(10, shape = c(1, 3, 7), scale = c(2, 2, 4), prop = c(0.6, 0.3, 0.1))

dgammamix(c(0, 2, 1), shape = c(1, 3), rate = c(2, 2), prop = c(0.6, 0.4))

prob <- pgammamix(c(0.1, 7), shape = c(1, 3, 7), scale = c(2, 2, 4), prop = c(0.6, 0.3, 0.1))
prob
qgammamix(prob, shape = c(1, 3, 7), scale = c(2, 2, 4), prop = c(0.6, 0.3, 0.1))
```

`half2full_circ` *Convert Angle from Half Circle to Full Circle*

Description

Converts an angle from the half circle (i.e. in the interval $[-\pi, \pi)$) to an angle on the full circle (i.e. in the interval $[0, 2\pi)$).

Usage

```
half2full_circ(angle)
```

Arguments

`angle`	numeric value of an angle or a `circular`-object in $[-\pi, \pi)$.

Value

The numeric value of the angle in $[0, 2\pi)$.

Examples

```
half2full_circ(-1 * pi) / pi
half2full_circ(-0.5 * pi) / pi
half2full_circ(-0 * pi) / pi
half2full_circ(0.5 * pi) / pi
```

joint — *Density, Distribution, Quantiles and Random Number Generation for joint distributions*

Description

The bivariate joint distributions are described in terms of two marginal distributions and a copula

Usage

```
rjoint(n, copula, marginal_1, marginal_2)

djoint(x, copula, marginal_1, marginal_2)

pjoint(q, copula, marginal_1, marginal_2)
```

Arguments

`n`	integer value, the number of random samples to be generated with `rjoint()`.
`copula`	R object of class '`cyl_copula`'. or '`Copula`' (package '**copula**', only 2-dimensional).
`marginal_1`	named list (for parametric estimates) or a '`density`' object (for linear kernel density estimates) or a '`density.circular`' object (for circular kernel density estimates). The output of functions `fit_angle()` and `fit_steplength()` can be used here directly.
`marginal_2`	This input is similar to `marginal_1`.
`x`	matrix (or vector) of numeric values giving the points (in 2 dimensions) where the density function is evaluated.
`q`	matrix (or vector) of numeric values giving the points (in 2 dimensions) where the distribution function is evaluated.

Details

If entered "by hand", the named lists describing the parametric distributions (`marginal_1` and `marginal_2`) must contain 2 entries:

1. `name`: a character string denoting the name of the distribution. For a circular distribution, it can be `"vonmises"`, `"vonmisesmix"`, or `"wrappedcauchy"`. For a linear distribution, it must be a string denoting the name of a linear distribution in the environment, i.e. the name of its distribution function without the "p", e.g. "norm" for normal distribution
2. `coef`: For a circular distribution `coef` is a (named) list of parameters of the circular marginal distribution as taken by the functions `qvonmises()`, `qvonmisesmix()`, or `qwrappedcauchy()`. For a linear distribution, `coef` is a named list containing the parameters of the distribution given in `"name"`.

Value

- `djoint()` gives a vector of length `length(x)` containing the density at `x`.
- `pjoint()` gives a vector of length `length(q)` containing the distribution function at the corresponding values of `q`.
- `rjoint()` generates a vector of length `n` containing the random samples.

Examples

```
cop <- copula::normalCopula(0.6)
marginal_1 <- list(name="exp",coef=list(rate=2))
marginal_2 <- list(name="lnorm", coef=list(0,0.1))

sample <- rjoint(10,cop,marginal_1,marginal_2)
pjoint(sample,cop,marginal_1,marginal_2)
djoint(sample,cop,marginal_1,marginal_2)

cop <- cyl_quadsec()
marginal_1 <- list(name="wrappedcauchy", coef=list(location=0,scale=0.3))
marginal_2 <- list(name="weibull",coef=list(shape=3))

sample <- rjoint(10,cop,marginal_1,marginal_2)
marginal_1 <- fit_angle(theta=sample[,1], parametric=FALSE)
marginal_2 <- fit_steplength(x=sample[,2],parametric="lnorm")
pjoint(c(0.3*pi,4),cop,marginal_1,marginal_2)
djoint(c(0,2),cop,marginal_1,marginal_2)
```

---

lnormmix — *Density, Distribution, Quantiles and Random Number Generation for the mixed log-normal distribution*

Description

The number of components in the mixed log-normal distribution is specified by the length of the parameter vectors. The quantiles are numerically obtained from the distribution function using monotone cubic splines.

Usage

```
rlnormmix(n, meanlog, sdlog, prop)

dlnormmix(x, meanlog, sdlog, prop)

plnormmix(q, meanlog, sdlog, prop)

qlnormmix(p, meanlog, sdlog, prop)
```

Arguments

n	integer value, the number of random samples to be generated with `rlnormmix()`.
meanlog	numeric vector holding the means of the components on the log scale.
sdlog	numeric vector holding the standard deviations of the components on the log scale.
prop	numeric vector, holding the mixing proportions of the components.
x	numeric vector giving the points where the density function is evaluated.
q	numeric vector giving the quantiles where the distribution function is evaluated.
p	numeric vector giving the probabilities where the quantile function is evaluated.

Value

- `dlnormmix()` gives a vector of length `length(x)` containing the density at `x`.
- `plnormmix()` gives a vector of length `length(q)` containing the distribution function at the corresponding values of `q`.
- `qlnormmix()` gives a vector of length `length(p)` containing the quantiles at the corresponding values of `p`.
- `rlnormmix()` generates a vector of length `n` containing the random samples.

Examples

```
rlnormmix(10, meanlog = c(1, 3, 7), sdlog = c(2, 2, 4), prop = c(0.6, 0.3, 0.1))

dlnormmix(c(0, 2, 1), meanlog = c(1, 3), sdlog = c(2, 2), prop = c(0.6, 0.4))

prob <- plnormmix(c(0.1, 7), meanlog = c(1, 3, 7), sdlog = c(2, 2, 4), prop = c(0.6, 0.3, 0.1))
prob
qlnormmix(prob, meanlog = c(1, 3, 7), sdlog = c(2, 2, 4), prop = c(0.6, 0.3, 0.1))
```

---

mi_cyl   *Estimate the Mutual Information Between a Circular and a Linear Random Variable*

Description

The empirical copula is obtained from the data (`theta` and `x`), and the mutual information of the 2 components is calculated. This gives a non-negative number that can be normalized to lie between 0 and 1.

Usage

```
mi_cyl(theta, x, normalize = TRUE, symmetrize = FALSE)
```

Arguments

theta	numeric vector of angles (measurements of a circular variable).
x	numeric vector of step lengths (measurements of a linear variable).
normalize	logical value whether the mutual information should be normalized to lie within $[0, 1]$.
symmetrize	logical value whether it should be assumed that right and left turns are equivalent. If theta can take values in $[-\pi, \pi)$, this means that positive and negative angles are equivalent.

Details

First, the two components of the empirical copula, u and v are obtained. Then the mutual information is calculated via discretizing u and v into `length(theta)^(1/3)` bins. The mutual information can be normalized to lie between 0 and 1 by dividing by the product of the entropies of u and v. This is done using functions from the '**infotheo**' package.

Even if u and v are perfectly correlated (i.e. `cor_cyl` goes to 1 with large sample sizes), the normalized mutual information will not be 1 if the underlying copula is periodic and symmetric. E.g. while `normalCopula(1)` has a correlation of 1 and a density that looks like a line going from $(0, 0)$ to $(1, 1)$, `cyl_rect_combine(normalCopula(1))` has a density that looks like "<". The mutual information will be 1 in the first case, but not in the second. Therefore, we can set `symmetrize = TRUE` to first convert (if necessary) theta to lie in $[-\pi, \pi)$ and then multiply all angles larger than 0 with -1. The empirical copula is then calculated and the mutual information is obtained from those values. It is exactly 1 in the case of perfect correlation as captured by e.g. `cyl_rect_combine(normalCopula(1))`.

Note also that the mutual information is independent of the marginal distributions. However, `symmetrize=TRUE` only works with angles, not with pseudo-observations. When x and theta are pseudo-observations, information is lost due to the ranking, and symmetrization will fail.

Value

A numeric value, the mutual information between theta and x in nats.

References

Ma J, Sun Z (2011). "Mutual Information Is Copula Entropy." *Tsinghua Science and Technology*, **16**(1), 51-54. ISSN 1007-0214, doi:10.1016/S10070214(11)700086, https://www.sciencedirect.com/science/article/pii/S1007021411700086/.

Calsaverini RS, Vicente R, Systems C, Artes ED (2009). "An information-theoretic approach to statistical dependence: Copula information." *Europhysics Letters*, **88**(6), 1–6. doi:10.1209/0295-5075/88/68003, https://iopscience.iop.org/article/10.1209/0295-5075/88/68003/.

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/.

See Also

`cor_cyl()`, `fit_cylcop_cor()`.

Examples

```
set.seed(123)

cop <- cyl_quadsec(0.1)
marg1 <- list(name="vonmises",coef=list(0,4))
marg2 <- list(name="lnorm",coef=list(2,3))

#draw samples and calculate the mutual information.
sample <- rjoint(100,cop,marg1,marg2)
mi_cyl(theta = sample[,1],
  x = sample[,2],
  normalize = TRUE,
  symmetrize = FALSE
)

#the correlation coefficient is independent of the marginal distribution.
 sample <- traj_sim(100,
  cop,
  marginal_circ = list(name = "vonmises", coef  = list(0, 1)),
  marginal_lin = list(name = "weibull", coef = list(shape = 2))
)

mi_cyl(theta = sample$angle,
  x = sample$steplength,
  normalize = TRUE,
  symmetrize = FALSE)
mi_cyl(theta = sample$cop_u,
  x = sample$cop_v,
  normalize = TRUE,
  symmetrize = FALSE)

# Estimate correlation of samples drawn from circular-linear copulas
# with perfect correlation.
cop <- cyl_rect_combine(copula::normalCopula(1))
sample <- rjoint(100,cop,marg1,marg2)
# without normalization
mi_cyl(theta = sample[,1],
  x = sample[,2],
  normalize = FALSE,
  symmetrize = FALSE
)
#with normalization
mi_cyl(theta = sample[,1],
  x = sample[,2],
  normalize = TRUE,
  symmetrize = FALSE
)
#only with normalization and symmetrization do we get a value of 1
mi_cyl(theta = sample[,1],
  x = sample[,2],
  normalize = TRUE,
  symmetrize = TRUE
```

```
)
```

`mle.vonmisesmix` *Mixed von Mises Maximum Likelihood Estimates*

Description

Computes the maximum likelihood estimates for the parameters of a mixed von Mises distribution: the mean directions, the concentration parameters, and the proportions of the distributions. The code is a simplified version of `movMF::movMF()` with the added feature of optionally fixed mean directions (Hornik and Grün 2014).

Usage

```
mle.vonmisesmix(theta, mu = NULL, ncomp = 2)
```

Arguments

`theta`	numeric vector of angles.
`mu`	(optional) numeric vector of length `ncomp` holding the mean directions (angles). If not specified the mean directions are estimated.
`ncomp`	positive integer specifying the number of components of the mixture model.

Details

The function complements the '**circular**' package, which provides functions to make maximum likelihood estimates of e.g. von Mises (`circular::mle.vonmises()`), or wrapped Cauchy distributions (`circular::mle.wrappedcauchy()`)

Value

A list containing the optimized parameters `mu`, `kappa`, and `prop`.

References

Hornik K, Grün B (2014). "movMF : An R Package for Fitting Mixtures of von Mises-Fisher Distributions." *Journal of Statistical Software*, **58**. doi:10.18637/jss.v058.i10..

See Also

`movMF::movMF()`, `circular::mle.vonmises()`, `dvonmisesmix()`, `qvonmisesmix()`.

Examples

```
set.seed(123)

n <- 10
angles <- rvonmisesmix(n,
  mu = c(0, pi),
  kappa = c(2, 1),
  prop = c(0.4,0.6)
)
mle.vonmisesmix(theta = angles)
mle.vonmisesmix(theta = angles, mu = c(0, pi))
```

---

normmix — *Density, Distribution, Quantiles and Random Number Generation for the mixed normal distribution*

Description

The number of components in the mixed normal distribution is specified by the length of the parameter vectors. The quantiles are numerically obtained from the distribution function using monotone cubic splines.

Usage

```
rnormmix(n, mu, sigma, prop)

dnormmix(x, mu, sigma, prop)

pnormmix(q, mu, sigma, prop)

qnormmix(p, mu, sigma, prop)
```

Arguments

n	integer value, the number of random samples to be generated with `rnormmix()`.
mu	numeric vector holding the means of the components.
sigma	numeric vector holding the standard deviations of the components.
prop	numeric vector, holding the mixing proportions of the components.
x	numeric vector giving the points where the density function is evaluated.
q	numeric vector giving the quantiles where the distribution function is evaluated.
p	numeric vector giving the probabilities where the quantile function is evaluated.

Value

- dnormmix() gives a vector of length length(x) containing the density at x.
- pnormmix() gives a vector of length length(q) containing the distribution function at the corresponding values of q.
- qnormmix() gives a vector of length length(p) containing the quantiles at the corresponding values of p.
- rnormmix() generates a vector of length n containing the random samples.

Examples

```
rnormmix(10, mu = c(0, 3, 7), sigma = c(2, 2, 4), prop = c(0.6, 0.3, 0.1))

dnormmix(c(0, 2, 1), mu = c(0, 3), sigma = c(2, 2), prop = c(0.6, 0.4))

prob <- pnormmix(c(0.1, 7), mu = c(0, 3, 7), sigma = c(2, 2, 4), prop = c(0.6, 0.3, 0.1))
prob
qnormmix(prob, mu = c(0, 3, 7), sigma = c(2, 2, 4), prop = c(0.6, 0.3, 0.1))
```

---

numerical_conditional_cop

Numerically Calculate the Conditional Copula

Description

Numerically Calculate the Conditional Copula

Usage

```
numerical_conditional_cop(u, copula, cond_on)
```

Arguments

u	matrix or vector of numeric values in I^2, containing as first column the circular (periodic) and as second the linear dimension.
copula	R object of class 'cyl_copula' or 'Copula' (package '**copula**', only 2-dimensional).
cond_on	column number of u on which the copula is conditioned. E.g. if cond_on = 2, the function calculates for each element in the first column of u the copula conditional on the element in the second column.

Value

A vector containing the values of the distribution of the copula at u[,-cond_on] conditional on the values of u[,cond_on].

References

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, `https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/`.

Hodel FH, Fieberg JR (2022). "Circular-Linear Copulae for Animal Movement Data." *Methods in Ecology and Evolution*. doi:10.1111/2041210X.13821.

See Also

`ccylcop()`, `numerical_inv_conditional_cop()`.

Examples

```
cop <- cyl_quadsec(0.1)
u <- cbind(c(0.3, 0.1), c(0.7, 0.3))
numerical_conditional_cop(u = u, cop = cop, cond_on = 1)
```

`numerical_inv_conditional_cop`
Numerically calculate the inverse of the conditional copula

Description

Numerically calculate the inverse of the conditional copula

Usage

```
numerical_inv_conditional_cop(u, copula, cond_on)
```

Arguments

`u`	matrix or vector of numeric values in I^2, containing as first column the circular (periodic) and as second the linear dimension.
`copula`	R object of class '`cyl_copula`' or '`Copula`' (package '**copula**', only 2-dimensional).
`cond_on`	column number of `u` on which the copula is conditioned. E.g if `cond_on = 2`, the function calculates for each element in the first column of `u` the inverse of the Copula conditional on the element in the second column.

Value

A vector containing the values of the inverse distribution of the copula at `[u,-cond_on]` conditional on the values of `[u,cond_on]`.

References

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/.

Hodel FH, Fieberg JR (2022). "Circular-Linear Copulae for Animal Movement Data." *Methods in Ecology and Evolution*. doi:10.1111/2041210X.13821.

See Also

ccylcop(), numerical_conditional_cop().

Examples

```
cop <- cyl_quadsec(0.1)
u <- cbind(c(0.3, 0.1), c(0.7, 0.3))
numerical_inv_conditional_cop(u = u, cop = cop, cond_on = 1)
```

---

opt_auto *Automatically Find the Best Fitting Copula*

Description

The parameters of 15 different circular-linear copulas are fitted to data and sorted according to AIC. For each copula, first, a starting value for the maximum likelihood estimation (MLE) is found using fit_cylcop_cor(). Then, MLE is carried out with a "reasonable" setup using fit_cylcop_ml(). If MLE fails, parameters obtained with fit_cylcop_cor() are reported.

Usage

```
opt_auto(theta, x)
```

Arguments

theta	numeric vector of angles (measurements of a circular variable).
x	numeric vector of step lengths (measurements of a linear variable).

Value

A list containing 3 lists: Descriptions of the copulas, the 'cyl_copula' objects with fitted parameters, and the AIC. The lists are sorted by ascending AIC. If fit_cylcop_ml() has failed, the reported parameters are the ones obtained with fit_cylcop_cor() and the AIC is set to NA.

References

Hodel FH, Fieberg JR (2022). "Circular-Linear Copulae for Animal Movement Data." *Methods in Ecology and Evolution*. doi:10.1111/2041210X.13821.

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, `https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/`.

See Also

`fit_cylcop_cor()`, `fit_cylcop_ml()`

Examples

```
set.seed(123)

#Optimal copula is independent of marginals.
data <- rcylcop(100,cyl_quadsec(0.1))

#This takes a few seconds to run.
copula_lst <- opt_auto(theta = data[,1], x = data[,2])
```

`opt_circ_bw` *Find the Optimal Bandwidth for a Circular Kernel Density Estimate*

Description

This function basically wraps `circular::bw.cv.ml.circular()` and `circular::bw.nrd.circular()` of the '**circular**' package, simplifying their inputs. For more control, these '**circular**' functions could be used directly. The normal reference distribution (`"nrd"`) method of finding the bandwidth parameter might give very bad results, especially for multi-modal population distributions. In these cases it can help to set `kappa.est = "trigmoments"`.

Usage

```
opt_circ_bw(theta, method = c("cv", "nrd"), kappa.est = "trigmoments")
```

Arguments

`theta`	numeric vector of angles in $[-\pi, \pi)$.
`method`	character string describing the method, either `"cv"` (cross-validation), or `"nrd"` leading to a rule-of-thumb estimate.
`kappa.est`	character string describing how the spread is estimated. Either maximum likelihood `"ML"`, or trigonometric moment `"trigmoments"`.

Details

method="nrd" is somewhat similar to the linear case (see fit_steplength()). Instead of matching a normal distribution to the data and then calculating its optimal bandwidth, a von Mises distribution is used. To match that von Mises distribution to the data we can either find its concentration parameter kappa using maximum likelihood (kappa.est="ML") or by trigonometric moment matching (kappa.est="trigmoments"). When the data is multimodal, fitting a (unimodal) von Mises distribution using maximum likelihood will probably give bad results. Using kappa.est="trigmoments" potentially works better in those cases.

As an alternative, the bandwidth can be found by maximizing the cross-validation likelihood (method="cv"). However, with this leave-one-out cross-validation scheme, at every likelihood optimization step, $n(n-1)$ von Mises densities need to be calculated, where n =length(theta). Therefore, this method can become quite slow with large sample sizes.

Value

A numeric value, the optimized bandwidth.

See Also

circular::bw.cv.ml.circular(), circular::bw.nrd.circular(), opt_circ_bw().

Examples

```
require(circular)
require(graphics)
set.seed(123)
n <- 10  #n (number of samples) is set small for performance. Increase n to
         # a value larger than 1000 to see the effects of multimodality

angles <- rvonmisesmix(n,
  mu = c(0,pi),
  kappa = c(2,1),
  prop = c(0.5,0.5)
)
bw1 <- opt_circ_bw(theta = angles, method="nrd", kappa.est = "ML")
bw2 <- opt_circ_bw(theta = angles, method="nrd", kappa.est = "trigmoments")
bw3 <- opt_circ_bw(theta = angles, method="cv")

dens1 <- fit_angle(theta = angles, parametric = FALSE, bandwidth = bw1)
dens2 <- fit_angle(theta = angles, parametric = FALSE, bandwidth = bw2)
dens3 <- fit_angle(theta = angles, parametric = FALSE, bandwidth = bw3)
true_dens <- dvonmisesmix(
  seq(-pi,pi,0.001),
  mu = c(0,pi),
  kappa = c(2,1),
  prop = c(0.5,0.5)
)
if(interactive()){
 plot(seq(-pi, pi, 0.001), true_dens, type = "l")
 lines(as.double(dens1$x), as.double(dens1$y), col = "red")
 lines(as.double(dens2$x), as.double(dens2$y), col = "green")
```

```
 lines(as.double(dens3$x), as.double(dens3$y), col = "blue")
}
```

---

opt_lin_bw *Find the Optimal Bandwidth for a Linear Kernel Density Estimate*

Description

This function wraps `stats::bw.ucv()` and `stats::bw.nrd()` of the '**stats**' package, simplifying their inputs. For more control, these '**stats**' functions could be used directly.

Usage

```
opt_lin_bw(x, method = c("cv", "nrd"))
```

Arguments

`x`	numeric vector of linear measurements.
`method`	character string describing the method used to find the optimal bandwidth. Either `"cv"` (cross-validation), or `"nrd"` (rule-of-thumb estimate).

Details

The normal reference distribution (`nrd`) method involves matching a normal distribution to the data using an empirical measure of spread. The optimal bandwidth for that normal distribution can then be exactly calculated by minimizing the mean integrated square error. `method="cv"` finds the optimal bandwidth using unbiased cross-validation.

Value

A numeric value, the optimized bandwidth.

See Also

`stats::bw.ucv()`, `stats::bw.nrd()` `opt_lin_bw()`.

Examples

```
require(graphics)
set.seed(123)
n <- 1000

x <- rweibull(n, shape = 10)
bw1 <- opt_lin_bw(x = x, method="nrd")
bw2 <- opt_lin_bw(x = x, method="cv")

dens1 <- fit_steplength(x = x, parametric = FALSE, bandwidth = bw1)
dens2 <- fit_steplength(x = x, parametric = FALSE, bandwidth = bw2)
```

```
true_dens <- dweibull(seq(0,max(x),length.out = 200), shape = 10)

plot(seq(0,max(x),length.out = 200), true_dens, type = "l")
lines(dens1$x, dens1$y, col = "red")
lines(dens2$x, dens2$y, col = "green")
```

---

plot,cyl_copula,missing-method
Plot 'cyl_copula' *Objects*

Description

Method for plot() to draw a scatter plot of a random sample from a circular-linear copula.

Usage

```
## S4 method for signature 'cyl_copula,missing'
plot(x, n = 1000, ...)
```

Arguments

x	R object of class 'cyl_copula'.
n	sample size of the random sample drawn from x.
...	additional arguments passed to plot().

Value

An invisible NULL. As side effect, a plot is produced.

Examples

```
set.seed(123)

plot(cyl_quadsec(0.1))
plot(cyl_vonmises(0,2), n = 100)
plot(cyl_quadsec(0.1),
   xlab = "something",
   ylab = "something else",
   main = "clever title",
   col = "red",
   fg = "blue",
   asp= 1)
```

plot_circ_hist *Circular Histogram of Turn Angles*

Description

This function produces a circular histogram of turn angles, i.e. angles on the the half-circle between -pi and pi.

Usage

```
plot_circ_hist(theta, nbars = 20)
```

Arguments

theta	numeric vector of angles (measurements of a circular variable) or "circular" component of pseudo-observations. They must be on the half-circle, i.e. theta must be in $[-\pi, \pi)$.
nbars	numeric integer, the number of bins (bars) in the histogram.

Value

A 'ggplot' object.

References

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/.

See Also

plot_joint_scat().

Examples

```
set.seed(123)

theta <- cylcop::rvonmisesmix(n = 100,
  mu = c(0, pi),
  kappa = c(5, 2),
  prop = c(4, 2)
)
plot1 <- plot_circ_hist(theta)
```

plot_cop_scat — *Scatterplot of Copula Values*

Description

This function produces a scatterplot ('ggplot' object) of a sample from a copula. Either a sample is provided as input, or a sample is drawn from a copula to quickly visualize it.

Usage

```
plot_cop_scat(traj = NULL, u = NULL, v = NULL)
```

Arguments

traj	a data.frame containing the trajectory produced by e.g. `traj_sim()`, which must contain the columns `traj$cop_u` and `traj$cop_v`.
u	(alternatively) numeric vector of first components of pseudo-observations or draws from a copula.
v	(alternatively) numeric vector of second components of pseudo-observations or draws from a copula.

Details

Alternatively, instead of plotting a sample from a copula `cop` using `scatterplot(copula=cop)`, you can also use `plot(cop)`. If a trajectory is provided and `n` is smaller than `nrow(traj)`, `n` steps are randomly selected from the trajectory and plotted.

Value

A 'ggplot' object, the scatterplot.

References

Hodel FH, Fieberg JR (2022). "Circular-Linear Copulae for Animal Movement Data." *Methods in Ecology and Evolution*. doi:10.1111/2041210X.13821.

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/.

See Also

`plot_track()`, `plot_joint_circ()`, `plot_cop_surf()`, `plot_joint_scat()`.

Examples

```
set.seed(123)
traj <- traj_sim(100,
  copula = cyl_quadsec(0.1),
  marginal_circ = list(name = "vonmises", coef  = list(0, 1)),
  marginal_lin = list(name = "weibull", coef = list(shape = 3))
)
plot_cop_scat(traj = traj)

sample <- rcylcop(100,cyl_quadsec(0.1))
plot_cop_scat(u = sample[,1], v = sample[,2])
```

---

plot_cop_surf *Surface Plot or Heat Map of the Distribution or the Density of a Copula*

Description

This function plots the distribution or the density of a copula. It can produce a surface plot using either functions from the '**rgl**' or from the '**plotly**' package, or it can produce a heat map using functions from '**ggplot2**'.

Usage

```
plot_cop_surf(
  copula,
  type = "pdf",
  plot_type = "rgl",
  resolution = 50,
  n_gridlines = 11
)
```

Arguments

`copula`	'`cyl_copula`' or a '`Copula`' object from the package '**copula**'.
`type`	character string describing what is plotted, either `"pdf"` or `"cdf"`.
`plot_type`	character string describing what type of plot is produced. Available plot types are: `"rgl"`: surface plot, `"plotly"`: interactive surface plot, or `"ggplot"`: heatmap
`resolution`	numeric value. The density or distribution will be calculated at `resolution^2` points.
`n_gridlines`	numeric value giving the number of grid lines drawn in u and v direction.

Value

Depending on `plot_type`, a '`ggplot`' object is returned, or a '**plotly**' visualization or '**rgl**' plot is produced.

References

Hodel FH, Fieberg JR (2022). “Circular-Linear Copulae for Animal Movement Data.” *Methods in Ecology and Evolution*. doi:10.1111/2041210X.13821.

Hodel FH, Fieberg JR (2021). “Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry.” *bioRxiv*. doi:10.1101/2021.07.14.452253, https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/.

See Also

plot_cop_scat(), plot_track(), plot_joint_circ(), plot_joint_scat().

Examples

```
if(interactive()){
 plot_cop_surf(copula::frankCopula(2),
   type="pdf",
   plot_type="ggplot",
   resolution = 5
 )
 plot_cop_surf(copula::frankCopula(2),
   type="cdf",
   plot_type="ggplot",
   resolution = 5
 )

#opens a new window
  plot_cop_surf(cyl_quadsec(0.1),
    type="pdf",
    plot_type="rgl"
  )
  plot_cop_surf(cyl_quadsec(0.1),
    type="pdf",
    plot_type="rgl",
    n_gridlines = 60
  )

  plot_cop_surf(cyl_quadsec(0.1),
    type="pdf",
    plot_type="plotly",
    n_gridlines = 10,
    resolution = 10
  )
}
```

---

plot_joint_box *Circular Boxplot of Turn Angles and Step Lengths*

Description

This function produces circular boxplots (a 'ggplot' object) of the turn angles corresponding to specific quantiles of the step lengths.

Usage

```
plot_joint_box(
  traj = NULL,
  theta = NULL,
  x = NULL,
  levels = 5,
  marginal_lin = NULL,
  spacing = 0.3,
  legend_pos = "right"
)
```

Arguments

traj	data.frame containing the trajectory produced by e.g. `traj_sim()`. It must contain the columns `traj$angle` and `traj$steplength`.
theta	(alternatively) numeric vector of angles (measurements of a circular variable) or "circular" component of pseudo-observations.
x	(alternatively) numeric vector of step lengths (measurements of a linear variable) or "linear" component of pseudo-observations.
levels	integer value between 1 and 15, the number of quantiles into which the step lengths are split.
marginal_lin	named list (for parametric estimates) or a 'density' object (for kernel density estimates). The output of function `fit_steplength()` can be used here directly for both cases. If `marginal_lin` is specified, the limits of the quantiles of the step lengths are determined from that distribution instead of from the data specified with `traj$steplength` or `x`.
spacing	numeric value between 0 and 10 determining the spacing between the boxplots.
legend_pos	character string denoting the position of the legend (limits of the step length quantiles). Either `"left"`, `"right"`, `"top"`, or `"bottom"`

Details

The step lengths are split into quantiles. For each quantile a boxplot of the corresponding turn angles is produced and wrapped around the circle. The turn angle values are plotted as scatter plot overlaying the boxplot. Outliers are plotted in red. The median of the turn angles is defined as the center of the shortest arc that connects all points. The length of the whiskers is 1.5 times the interquartile range.

You can either specify `traj` or the angels (`theta`) and step lengths (codex). If entered "by hand", the named list describing the marginal linear distribution (for `marginal_lin`) must contain 2 entries:

1. `name`: a character string denoting the name of the linear distribution, i.e. the name of its distribution function without the "p", e.g. "norm" for normal distribution.
2. `coef`: a named list containing the parameters of the distribution given in `"name"`.

Value

A 'ggplot' object, the circular boxplot.

References

Hodel FH, Fieberg JR (2022). “Circular-Linear Copulae for Animal Movement Data.” *Methods in Ecology and Evolution*. doi:10.1111/2041210X.13821.

Hodel FH, Fieberg JR (2021). “Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry.” *bioRxiv*. doi:10.1101/2021.07.14.452253, https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/.

See Also

plot_cop_scat(), plot_track(), plot_joint_circ(), plot_cop_surf().

Examples

```
set.seed(1234)

traj <- traj_sim(100,
  copula = cyl_rect_combine(copula::frankCopula(6)),
  marginal_circ = list(name= "vonmises", coef=list(0, 2)),
  marginal_lin = list(name = "weibull", coef=list(shape=3))
)

plot1 <- plot_joint_box(traj)
plot2 <- plot_joint_box(traj,
  marginal_lin=list(name = "weibull", coef=list(shape=3))
)
```

---

plot_joint_circ *Circular Scatterplot of Turn Angles and Step Lengths*

Description

This function produces a circular scatterplot with the step lengths plotted as distance from the center of a circle and the turn angles as angles (polar coordinates).

Usage

```
plot_joint_circ(traj = NULL, theta = NULL, x = NULL)
```

Arguments

`traj`	data.frame containing the trajectory produced by e.g. `traj_sim()`. It must contain the columns `traj$angle` and `traj$steplength`.
`theta`	(alternatively) numeric vector of angles (measurements of a circular variable) or "circular" component of pseudo-observations.
`x`	(alternatively) numeric vector of step lengths (measurements of a linear variable) or "linear" component of pseudo-observations.

Details

You can either specify `traj` or the angels and step lengths `theta` and `x`.

Value

A '`ggplot`' object.

References

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, `https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/`.

See Also

`plot_cop_scat()`, `plot_track()`, `plot_cop_surf()`, `plot_joint_scat()`.

Examples

```
set.seed(123)

traj <- traj_sim(100,
  copula = cyl_quadsec(0.1),
  marginal_circ = list(name="vonmises",coef=list(0, 1)),
  marginal_lin = list(name="weibull", coef=list(shape=3))
)
plot1 <- plot_joint_circ(traj)
```

`plot_joint_scat` *Scatterplot of Turn Angles and Step Lengths*

Description

This function produces a scatterplot ('`ggplot`' object) of the turn angles and step lengths.

Usage

```
plot_joint_scat(
  traj = NULL,
  theta = NULL,
  x = NULL,
  periodic = FALSE,
  plot_margins = FALSE
)
```

Arguments

`traj`	data.frame containing the trajectory produced by e.g. `traj_sim()`. It must contain the columns `traj$angle` and `traj$steplength`.
`theta`	(alternatively) numeric vector of angles (measurements of a circular variable).
`x`	(alternatively) numeric vector of step lengths (measurements of a linear variable).
`periodic`	logical value denoting whether the plot should be periodically extended past -pi and pi.
`plot_margins`	logical determining whether the marginal kernel density estimates are computed and plotted. Alternatively, `plot_margins` can be a list of length 2 containing first a kernel density estimate for `theta` and second a kernel density estimate for `x`. The first entry must be of type `'density.circular'` (as returned e.g. by `fit_angle(theta, parametric=FALSE)`), and the second entry must be of type `"density"` (as returned e.g. by `fit_steplength(x, parametric=FALSE)`).

Details

You can either specify `traj` or the angels and step lengths (`theta` and `x`). If `plot_margins=T`, the code will attempt to find appropriate bandwidths for the kernel density estimate autonomously, also taking into account computational time. For more control over the actual method and parameters used to obtain the kernel density estimates, you can calculate them "by hand" using e.g. `fit_angle(theta, parametric=FALSE)` and `fit_steplength(x, parametric=FALSE)`.

Value

A '`ggplot`' object, the scatterplot.

References

Hodel FH, Fieberg JR (2022). "Circular-Linear Copulae for Animal Movement Data." *Methods in Ecology and Evolution*. doi:10.1111/2041210X.13821.

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, `https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/`.

See Also

`plot_cop_scat()`, `plot_track()`, `plot_joint_circ()`, `plot_cop_surf()`.

Examples

```
set.seed(123)
traj <- traj_sim(100,
  copula = cyl_quadsec(0.1),
  marginal_circ = list(name = "vonmises", coef  = list(0, 1)),
  marginal_lin = list(name = "weibull", coef = list(shape = 3))
)

plot1 <- plot_joint_scat(traj)
plot2 <- plot_joint_scat(traj, periodic = TRUE)
plot3 <- plot_joint_scat(theta=traj$angle, x=traj$steplength, periodic = TRUE, plot_margins=TRUE)

bw <- opt_circ_bw(theta = traj$angle, method = "nrd",kappa.est = "trigmoments")
ang_dens <- fit_angle(theta=traj$angle, parametric=FALSE, bandwidth=bw)
step_dens <- fit_steplength(x=traj$steplength, parametric=FALSE)
plot4 <- plot_joint_scat(traj, periodic = TRUE, plot_margins=list(ang_dens, step_dens))
```

`plot_track` *Plot a Trajectory in Euclidean Space*

Description

This function plots the locations of a trajectory or multiple trajectories.

Usage

```
plot_track(traj = NULL, x_coord = NULL, y_coord = NULL)
```

Arguments

`traj`	data.frame containing the trajectory produced by e.g. `traj_sim()`. It must contain the columns `traj$pos_x` and `traj$pos_y`. It is also possible to specify a list of such data.frames containing multiple trajectories.
`x_coord`	(alternatively) numeric vector of x-coordinates or a list of x-coordinate vectors of multiple trajectories.
`y_coord`	(alternatively) numeric vector of y-coordinates or a list of y-coordinate vectors of multiple trajectories.

Value

A '`ggplot`' object.

References

Hodel FH, Fieberg JR (2022). "Circular-Linear Copulae for Animal Movement Data." *Methods in Ecology and Evolution*. doi:10.1111/2041210X.13821.

Hodel FH, Fieberg JR (2021). "Cylcop: An R Package for Circular-Linear Copulae with Angular Symmetry." *bioRxiv*. doi:10.1101/2021.07.14.452253, https://www.biorxiv.org/content/10.1101/2021.07.14.452253v3/.

See Also

`plot_cop_scat()`, `plot_joint_circ()`, `plot_cop_surf()`, `plot_joint_scat()`.

Examples

```
set.seed(123)
traj <- traj_sim(50,
  copula = cyl_quadsec(0.1),
  marginal_circ = list(name = "vonmises", coef  = list(0, 1)),
  marginal_lin = list(name = "weibull", coef = list(shape = 3))
)
plot1 <- plot_track(traj=traj)

x_coord <- list(runif(10),runif(20),runif(3))
y_coord <- list(runif(10),runif(20),runif(3))

plot2 <- plot_track(x_coord=x_coord, y_coord=y_coord)
```

---

prob,cyl_copula-method

Calculate the C-Volume of a '`cyl_copula`' Copula

Description

This is a method corresponding to the generic `prob()` in the '**copula**' package.

Usage

```
## S4 method for signature 'cyl_copula'
prob(x, l, u)
```

Arguments

x	R object of class '`cyl_copula`'.
l	numeric vector of length 2 holding the coordinates of the lower left corner in $[0, 1]^2$.
u	numeric vector of length 2 holding the coordinates of the upper right corner in $[0, 1]^2$.

Value

A numeric in $[0, 1]$, the probability that a draw from the 2-dimensional copula x falls in the rectangle defined by l and u.

See Also

copula::prob

Examples

```
cop <- cyl_quadsec(0.1)
prob(cop, l = c(0.1, 0.3), u = c(0.3, 0.9))
```

---

set_cop_param   *Change Attributes of 'cyl_copula' Objects*

Description

These methods can be used, e.g. in other functions, to give users limited access to the parameters of a copula.

Usage

```
set_cop_param(copula, param_val, param_name, ...)

## S4 method for signature 'cyl_cubsec'
set_cop_param(copula, param_val, param_name)

## S4 method for signature 'cyl_quadsec'
set_cop_param(copula, param_val, param_name)

## S4 method for signature 'cyl_rect_combine'
set_cop_param(copula, param_val, param_name)

## S4 method for signature 'cyl_rot_combine'
set_cop_param(copula, param_val, param_name)

## S4 method for signature 'cyl_vonmises'
set_cop_param(copula, param_val, param_name)
```

Arguments

copula	R object of class 'cyl_copula'.
param_val	numeric vector holding the values to which the parameters given in copula@parameters should be changed.
param_name	vector of character strings holding the names of the parameters to be changed.
...	additional arguments.

Details

Note that for a rectangular patchwork copula ('cyl_rect_combine') the attribute rectangles_symmetric cannot be changed by set_cop_param(), since rectangular patchwork copulas with symmetric rectangles are treated as distinct from rectangular patchwork copulas with potentially asymmetric rectangles. Therefore, when changing one of the bounds of the lower rectangle of such a copula, the corresponding bound of the upper rectangle is automatically changed as well (see examples).

Value

A 'cyl_copula' object with the changed parameters.

Examples

```
cop <- cyl_rect_combine(copula::normalCopula(0.2),low_rect = c(0.1,0.4), up_rect="symmetric")
cop
cop <- set_cop_param(cop, param_val = c(0.1, 0.3), param_name = c("rho.1", "low_rect2"))
cop <- cyl_rect_combine(copula::normalCopula(0.2),low_rect = c(0.1,0.4), up_rect=c(0.6,0.9))
cop
cop <- set_cop_param(cop, param_val = 0.3, param_name = "low_rect2")
cop
```

---

show,cyl_copula-method

Print Information of 'cyl_copula' Objects

Description

Methods for function show() in package **cylcop**

Usage

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

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

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

Arguments

object	R object of class 'cyl_copula'.

Value

An invisible NULL. As side effect, information on object is printed.

traj_get — Get a Trajectory from Coordinates

Description

The function calculates step lengths and turn angles from x- and y-coordinates and calculates pseudo-observations from those step lengths and turn angles.

Usage

```
traj_get(x_coords, y_coords)
```

Arguments

x_coords	vector of numeric values containing the x-coordinates of a trajectory.
y_coords	vector of numeric values containing the y-coordinates of a trajectory.

Value

A data.frame containing the trajectory. It has 6 columns containing the x and y coordinates, the turn angles, the step lengths, and the pseudo-observations.

See Also

traj_sim().

Examples

```
set.seed(123)

traj <- traj_sim(n = 5,
copula = cyl_quadsec(0.1),
marginal_circ = list(name="vonmises",coef=list(0, 1)),
marginal_lin = list(name="weibull",coef=list(shape=3))
)

traj_from_coords <- traj_get(traj[,1], traj[,2])
```

traj_sim *Generate a Trajectory with Correlated Step Lengths and Turn Angles*

Description

The function draws values from a circular-linear bivariate distribution of turn angles and step lengths specified by the marginal distributions and a circular-linear copula. From the start point (0,0) and the second (potentially user specified) point, a trajectory is then built with these turn angles and step lengths.

Usage

```
traj_sim(
  n,
  copula,
  marginal_circ,
  marginal_lin,
  ignore_first = TRUE,
  pos_2 = NULL
)
```

Arguments

`n`	integer, number of trajectory steps to generate.
`copula`	'`cyl_copula`' object.
`marginal_circ`	named list (for parametric estimates) or a '`density.circular`' object (for kernel density estimates). The output of function `fit_angle()` can be used here directly for both cases.
`marginal_lin`	named list (for parametric estimates) or a '`density`' object (for kernel density estimates). The output of function `fit_steplength()` can be used here directly for both cases.
`ignore_first`	logical value. If `ignore_first = TRUE` (default), a trajectory of length `n+2` is generated and the first two steps of that trajectory are removed.
`pos_2`	(optional) numeric vector of length 2 containing the coordinates of the second point in the trajectory. The first point is always at (0,0). If no value is specified, the second point is obtained by going in a random direction from the first point for a distance drawn from the marginal step length distribution.

Details

Samples are drawn from the circular-linear copula and then transformed using the quantile functions of the marginal circular and the marginal linear distribution. To generate draws from any bivariate joint distribution (not necessarily a circular-linear one) without also producing a trajectory, the function `rjoint()` can be used.

If entered "by hand", the named lists describing the parametric distributions (`marginal_circ` and `marginal_lin`) must contain 2 entries:

1. `name`: a character string denoting the name of the distribution. For the circular distribution, it can be `"vonmises"`, `"vonmisesmix"`, or `"wrappedcauchy"`. For the linear distribution, it must be a string denoting the name of a linear distribution in the environment, i.e. the name of its distribution function without the "p", e.g. "norm" for normal distribution
2. `coef`: For the circular distribution `coef` is a (named) list of parameters of the circular marginal distribution as taken by the functions `qvonmises()`, `qvonmisesmix()`, or `qwrappedcauchy()`. For the linear distribution, `coef` is a named list containing the parameters of the distribution given in `"name"`.

Value

A data.frame containing the trajectory. It has 6 columns containing the x and y coordinates, the turn angles, the step lengths, and the values sampled from the copula.

See Also

`traj_get()`, `fit_steplength()`, `fit_angle()`, `plot_track()`, `plot_cop_scat()`, `plot_joint_scat()`, `plot_joint_circ()`.

Examples

```
require(circular)
set.seed(123)

traj <- traj_sim(n = 5,
copula = cyl_quadsec(0.1),
marginal_circ = list(name="vonmises",coef=list(0, 1)),
marginal_lin = list(name="weibull",coef=list(shape=3))
)

traj

angles <- rvonmisesmix(100,
  mu = c(0, pi),
  kappa = c(2, 3),
  prop = c(0.4, 0.6)
)
angles <- full2half_circ(angles)
bw <- opt_circ_bw(theta = angles, method = "nrd", kappa.est = "trigmoments")
marg_ang <- fit_angle(theta = angles, parametric = FALSE, bandwidth = bw)

steplengths <- rlnorm(100, 0, 0.3)
marg_stepl <- fit_steplength(x = steplengths, parametric = "lnorm")

traj_sim(n = 5,
copula = cyl_quadsec(0.1),
marginal_circ = marg_ang,
marginal_lin = marg_stepl,
ignore_first = FALSE,
pos_2 = c(5,5)
)
```

`vonmisesmix`	*Density, Distribution, Quantiles and Random Number Generation for the mixed von Mises Distribution*

Description

The number of components in the mixed von Mises distribution is specified by the length of the parameter vectors. The quantiles are numerically obtained from the distribution function using monotone cubic splines.

Usage

```
rvonmisesmix(n, mu, kappa, prop)

dvonmisesmix(theta, mu, kappa, prop)

pvonmisesmix(theta, mu, kappa, prop)

qvonmisesmix(p, mu, kappa, prop)
```

Arguments

`n`	integer value, the number of random samples to be generated with `rvonmisesmix()`.
`mu`	numeric vector holding the mean directions.
`kappa`	numeric vector holding the concentration parameters.
`prop`	numeric vector, holding the mixing proportions of the components.
`theta`	numeric vector giving the angles where the density or distribution function is evaluated.
`p`	numeric vector giving the probabilities where the quantile function is evaluated.

Value

- `dvonmisesmix()` gives a vector of length `length(theta)` containing the density at `theta`.
- `pvonmisesmix()` gives a vector of length `length(theta)` containing the distribution function at the corresponding values of `theta`.
- `qvonmisesmix()` gives a vector of length `length(p)` containing the quantiles at the corresponding values of `p`.
- `rvonmisesmix()` generates a vector of length `n` containing the random samples, i.e. angles in $[-\pi, \pi)$.

Examples

```
rvonmisesmix(10, mu = c(0, pi, pi/2), kappa = c(2, 2, 4), prop = c(0.6, 0.3, 0.1))

dvonmisesmix(c(0, 2, pi, 1), mu = c(0, pi), kappa = c(2, 2), prop = c(0.6, 0.4))

prob <- pvonmisesmix(c(0.1, pi), mu = c(0, pi, pi/2), kappa = c(2, 2, 4), prop = c(0.6, 0.3, 0.1))
prob
qvonmisesmix(prob, mu = c(0, pi, pi/2), kappa = c(2, 2, 4), prop = c(0.6, 0.3, 0.1))
```

---

wasserstein *Calculate the Wasserstein Distance*

Description

The Wasserstein distance is calculated based on the Euclidean distance between two copula PDFs on a grid, or between a copula PDF and pseudo-observations.

Usage

```
wasserstein(
  copula,
  copula2 = NULL,
  theta = NULL,
  x = NULL,
  n_grid = 2500,
  p = 2
)
```

Arguments

copula	R object of class 'cyl_copula'. or 'Copula' (package '**copula**', only 2-dimensional).
copula2	R object of class 'cyl_copula'. or 'Copula' (package '**copula**', only 2-dimensional).
theta	(alternatively) numeric vector of angles (measurements of a circular variable) or "circular" component of pseudo-observations.
x	(alternatively) numeric vector of step lengths (measurements of a linear variable) or "linear" component of pseudo-observations.
n_grid	integer number of grid cells at which the PDF of the copula(s) is calculated Default is 2500
p	integer power (1 or 2) to which the Euclidean distance between points is taken in order to compute transportation costs.

Details

Note that when comparing 2 copula PDFs (i.e. `theta = NULL` and `x = NULL`), the calculated Wasserstein distance will depend on the number of grid cells (`n_grid`) used to approximate the PDFs. The distance will converge to a certain value with a higher number of grid cells, but the computational time will also increase. The default of 2500 seems to be a good (empirically determined) compromise. The same is true when calculating the Wasserstein distance between a copula PDF and pseudo-observations. There, it is also important to only compare distances that use the same number of observations.

The code is based on the functions `transport::wasserstein()` and `transport::semidiscrete()`.

Value

numeric, the pth Wasserstein distance

Examples

```
set.seed(1234)
copula1 <- cyl_quadsec(0.1)
copula2 <- cyl_rect_combine(copula::frankCopula(2))
wasserstein(copula=copula1,copula2 = copula2,p=2,n_grid=20)
wasserstein(copula=copula1,copula2 = copula1,p=2,n_grid=20)
wasserstein(copula=copula1,copula2 = copula::frankCopula(2),p=2,n_grid=20)

 sample <- rjoint(10,
  copula1,
  marginal_1 = list(name = "vonmises", coef  = list(0, 1)),
  marginal_2 = list(name = "weibull", coef = list(3,4))
)

wasserstein(copula=copula1, theta=sample[,1], x=sample[,2], n_grid=20)
```

---

weibullmix — *Density, Distribution, Quantiles and Random Number Generation for the mixed Weibull distribution*

Description

The number of components in the mixed Weibull distribution is specified by the length of the parameter vectors. The quantiles are numerically obtained from the distribution function using monotone cubic splines.

Usage

```
rweibullmix(n, shape, scale, prop)

dweibullmix(x, shape, scale, prop)

pweibullmix(q, shape, scale, prop)
```

```
qweibullmix(p, shape, scale, prop)
```

Arguments

`n`	integer value, the number of random samples to be generated with `rweibullmix()`.
`shape`	numeric vector holding the shape parameter of the components.
`scale`	numeric vector holding the scale parameter of the components.
`prop`	numeric vector, holding the mixing proportions of the components.
`x`	numeric vector giving the points where the density function is evaluated.
`q`	numeric vector giving the quantiles where the distribution function is evaluated.
`p`	numeric vector giving the probabilities where the quantile function is evaluated.

Value

- `dweibullmix()` gives a vector of length `length(x)` containing the density at `x`.
- `pweibullmix()` gives a vector of length `length(q)` containing the distribution function at the corresponding values of `q`.
- `qweibullmix()` gives a vector of length `length(p)` containing the quantiles at the corresponding values of `p`.
- `rweibullmix()` generates a vector of length `n` containing the random samples.

Examples

```
rweibullmix(10, shape = c(1, 3, 7), scale = c(2, 2, 4), prop = c(0.6, 0.3, 0.1))

dweibullmix(c(0, 2, 1), shape = c(1, 3), scale = c(2, 2), prop = c(0.6, 0.4))

prob <- pweibullmix(c(0.1, 7), shape = c(1, 3, 7), scale = c(2, 2, 4), prop = c(0.6, 0.3, 0.1))
prob
qweibullmix(prob, shape = c(1, 3, 7), scale = c(2, 2, 4), prop = c(0.6, 0.3, 0.1))
```

`wrappedcauchy` *Density, Distribution, Quantiles and Random Number Generation for the Wrapped Cauchy Distribution*

Description

The distribution function (`pwrappedcauchy()`) and quantiles (`qwrappedcauchy()`) of the wrapped Cauchy distribution cannot be obtained analytically. They are therefore missing in the '**circular**' package and are obtained here numerically. Random number generation (`rwrappedcauchy()`) and density (`dwrappedcauchy()`) don't need a numerical approximation and are provided here for consistency in parametrization with the other wrapped Cauchy functions.

Usage

```
rwrappedcauchy(n, location = 0, scale = 1)

dwrappedcauchy(theta, location = 0, scale = 1)

pwrappedcauchy(theta, location = 0, scale = 1, K = 100, check_prec = FALSE)

qwrappedcauchy(p, location = 0, scale = 1, K = 100, check_prec = FALSE)
```

Arguments

n	integer value, the number of random samples to be generated with `rwrappedcauchy()`.
location	numeric value, the mean of the distribution.
scale	numeric value, the parameter tuning the spread of the density. It must be non-negative.
theta	numeric vector giving the angles where the density or distribution function is evaluated.
K	integer value, the number of "wraps" used in each direction to approximate the distribution.
check_prec	logical, whether to check if the precision of the numerical approximation with the current parameters is higher than 99%.
p	numeric vector giving the probabilities where the quantile function is evaluated.

Details

One could alternatively convert `scale` to `rho` via `rho = exp(-scale)` and use `circular::rwrappedcauchy(theta, mu=location rho=rho)` or `circular::dwrappedcauchy(theta, mu=location rho=rho)`.

The wrapped Cauchy cdf, for which there is no analytical expression, is calculated by wrapping the Cauchy distribution K times around the circle in each direction and summing the Cauchy cdfs at each point of the circle. Let Ω follow a Cauchy distribution and Θ a wrapped Cauchy distribution, where Θ can take values $\theta \in [-\pi, \pi)$. $Pr(\Theta \leq \theta)$ is approximated as

$$\sum_{k=-K}^{K} Pr(\Omega \leq \theta + 2\pi k) - Pr(\Omega \leq -\pi + 2\pi k).$$

The quantiles are calculated by numerical inversion.

Value

- `dwrappedcauchy()` gives a vector of length `length(theta)` containing the density at `theta`.
- `pwrappedcauchy()` gives a vector of length `length(theta)` containing the distribution function at the corresponding values of `theta`.
- `qwrappedcauchy()` gives a vector of length `length(p)` containing the quantiles at the corresponding values of `p`.
- `rwrappedcauchy()` generates a vector of length `n` containing the random samples, i.e. angles in $[-\pi, \pi)$.

See Also

circular::dwrappedcauchy(), circular::rwrappedcauchy().

Examples

```
set.seed(123)

rwrappedcauchy(10, location = 0, scale =3)

dwrappedcauchy(c(0.1, pi), location = pi, scale =2)
circular::dwrappedcauchy(circular::circular(c(0.1,pi)), mu = circular::circular(pi), rho =exp(-2))

prob <- pwrappedcauchy(c(0.1, pi), location = pi, scale =2)
prob
qwrappedcauchy(prob, location = pi, scale =2)
```

Index