# Package 'ridgetorus'

May 9, 2026

**Type** Package

**Title** PCA on the Torus via Density Ridges

**Version** 1.0.3

**Date** 2025-07-27

**Description** Implementation of a Principal Component Analysis (PCA) in the torus via density ridge estimation. The main function, ridge_pca(), obtains the relevant density ridge for bivariate sine von Mises and bivariate wrapped Cauchy distribution models and provides the associated scores and variance decomposition. Auxiliary functions for evaluating, fitting, and sampling these models are also provided. The package provides replicability to García-Portugués and Prieto-Tirado (2023) <doi:10.1007/s11222-023-10273-9>.

**License** GPL-3

**LazyData** true

**Depends** R (>= 3.5.0), Rcpp

**Imports** rootSolve, sdetorus, sphunif, circular

**Suggests** BAMBI, covr, DirStats, GGally, ggplot2, knitr, markdown, mvtnorm, numDeriv, rmarkdown, testthat, viridisLite

**LinkingTo** Rcpp, RcppArmadillo

**URL** https://github.com/egarpor/ridgetorus

**BugReports** https://github.com/egarpor/ridgetorus/issues

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**NeedsCompilation** yes

**Author** Eduardo García-Portugués [aut, cre] (ORCID: <https://orcid.org/0000-0002-9224-4111>), Arturo Prieto-Tirado [aut] (ORCID: <https://orcid.org/0000-0003-0193-2988>)

**Maintainer** Eduardo García-Portugués <edgarcia@est-econ.uc3m.es>

**Repository** CRAN

**Date/Publication** 2025-07-27 17:00:02 UTC

# Contents



---

`ridgetorus-package` `ridgetorus`*: PCA on the Torus via Density Ridges*

---

## Description

Implementation of a Principal Component Analysis (PCA) in the torus via density ridge estimation. The main function, ridge_pca(), obtains the relevant density ridge for bivariate sine von Mises and bivariate wrapped Cauchy distribution models and provides the associated scores and variance decomposition. Auxiliary functions for evaluating, fitting, and sampling these models are also provided. The package provides replicability to García-Portugués and Prieto-Tirado (2023) <doi:10.1007/s11222-023-10273-9>.

## Author(s)

Eduardo García-Portugués and Arturo Prieto-Tirado.

## References

García-Portugués, E. and Prieto-Tirado, A. (2023). Toroidal PCA via density ridges. *Statistics and Computing*, 33(5):107. doi:10.1007/s11222023102739

## See Also

Useful links:

- https://github.com/egarpor/ridgetorus
- Report bugs at https://github.com/egarpor/ridgetorus/issues

---

`biv_lrt` *Tests of homogeneity and independence in bivariate sine von Mises and wrapped Cauchy distributions*

---

## Description

Performs the following likelihood ratio tests for the concentrations in bivariate sine von Mises and wrapped Cauchy distributions: (1) *homogeneity*: $H_0 : \kappa_1 = \kappa_2$ vs. $H_1 : \kappa_1 \neq \kappa_2$, and $H_0 : \xi_1 = \xi_2$ vs. $H_1 : \xi_1 \neq \xi_2$, respectively; (2) *independence*: $H_0 : \lambda = 0$ vs. $H_1 : \lambda \neq 0$, and $H_0 : \rho = 0$ vs. $H_1 : \rho \neq 0$. The tests (1) and (2) can be performed simultaneously.

## Usage

```
biv_lrt(x, hom = FALSE, indep = FALSE, fit_mle = NULL, type, ...)
```

## Arguments

| | |
|---|---|
| `x` | matrix of dimension `c(n, 2)` containing the `n` observations of the pair of angles. |
| `hom` | test the homogeneity hypothesis? Defaults to `FALSE`. |
| `indep` | test the independence hypothesis? Defaults to `FALSE`. |
| `fit_mle` | output of `fit_bvm_mle` or `fit_bwc_mle` with `hom = FALSE`. Computed internally if not provided. |
| `type` | either `"bvm"` (bivariate sine von Mises) or `"bwc"` (bivariate wrapped Cauchy). |
| `...` | optional parameters passed to `fit_bvm_mle` and `fit_bwc_mle`, such as `start`, `lower`, or `upper`. |

## Value

A list with class `htest`:

| | |
|---|---|
| `statistic` | the value of the likelihood ratio test statistic. |
| `p.value` | the $p$-value of the test (computed using the asymptotic distribution). |
| `alternative` | a character string describing the alternative hypothesis. |
| `method` | description of the type of test performed. |
| `df` | degrees of freedom. |
| `data.name` | a character string giving the name of `theta`. |
| `fit_mle` | maximum likelihood fit. |
| `fit_null` | maximum likelihood fit under the null hypothesis. |

## References

Kato, S. and Pewsey, A. (2015). A Möbius transformation-induced distribution on the torus. *Biometrika*, 102(2):359–370. doi:10.1093/biomet/asv003

Singh, H., Hnizdo, V., and Demchuk, E. (2002). Probabilistic model for two dependent circular variables. *Biometrika*, 89(3):719–723. doi:10.1093/biomet/89.3.719

## Examples

```
## Bivariate sine von Mises

# Homogeneity
n <- 200
mu <- c(0, 0)
kappa_0 <- c(1, 1, 0.5)
kappa_1 <- c(0.7, 0.1, 0.25)
samp_0 <- r_bvm(n = n, mu = mu, kappa = kappa_0)
samp_1 <- r_bvm(n = n, mu = mu, kappa = kappa_1)
biv_lrt(x = samp_0, hom = TRUE, type = "bvm")
biv_lrt(x = samp_1, hom = TRUE, type = "bvm")

# Independence
kappa_0 <- c(0, 1, 0)
kappa_1 <- c(1, 0, 1)
samp_0 <- r_bvm(n = n, mu = mu, kappa = kappa_0)
samp_1 <- r_bvm(n = n, mu = mu, kappa = kappa_1)
biv_lrt(x = samp_0, indep = TRUE, type = "bvm")
biv_lrt(x = samp_1, indep = TRUE, type = "bvm")

# Independence and homogeneity
kappa_0 <- c(3, 3, 0)
kappa_1 <- c(3, 1, 0)
samp_0 <- r_bvm(n = n, mu = mu, kappa = kappa_0)
samp_1 <- r_bvm(n = n, mu = mu, kappa = kappa_1)
biv_lrt(x = samp_0, indep = TRUE, hom = TRUE, type = "bvm")
biv_lrt(x = samp_1, indep = TRUE, hom = TRUE, type = "bvm")

## Bivariate wrapped Cauchy

# Homogeneity
xi_0 <- c(0.5, 0.5, 0.25)
xi_1 <- c(0.7, 0.1, 0.5)
samp_0 <- r_bwc(n = n, mu = mu, xi = xi_0)
samp_1 <- r_bwc(n = n, mu = mu, xi = xi_1)
biv_lrt(x = samp_0, hom = TRUE, type = "bwc")
biv_lrt(x = samp_1, hom = TRUE, type = "bwc")

# Independence
xi_0 <- c(0.1, 0.5, 0)
xi_1 <- c(0.3, 0.5, 0.2)
samp_0 <- r_bwc(n = n, mu = mu, xi = xi_0)
samp_1 <- r_bwc(n = n, mu = mu, xi = xi_1)
```

```
biv_lrt(x = samp_0, indep = TRUE, type = "bwc")
biv_lrt(x = samp_1, indep = TRUE, type = "bwc")

# Independence and homogeneity
xi_0 <- c(0.2, 0.2, 0)
xi_1 <- c(0.1, 0.2, 0.1)
samp_0 <- r_bwc(n = n, mu = mu, xi = xi_0)
samp_1 <- r_bwc(n = n, mu = mu, xi = xi_1)
biv_lrt(x = samp_0, indep = TRUE, hom = TRUE, type = "bwc")
biv_lrt(x = samp_1, indep = TRUE, hom = TRUE, type = "bwc")
```

---

bvm *Density evaluation, sampling, and parameter estimation of the bivariate sine von Mises distribution*

---

## Description

Computation of the density and normalizing constant $T(\kappa_1, \kappa_2, \lambda)$ of the bivariate sine von Mises

$$f(\theta_1, \theta_2) = T(\kappa_1, \kappa_2, \lambda) \exp\{\kappa_1 \cos(\theta_1 - \mu_1) + \kappa_2 \cos(\theta_2 - \mu_2) + \lambda \sin(\theta_1 - \mu_1) \sin(\theta_2 - \mu_2)\}.$$

Simulation of samples from a bivariate sine von Mises.

Maximum likelihood and method of moments estimation of the parameters $(\mu_1, \mu_2, \kappa_1, \kappa_2, \lambda)$.

## Usage

```
d_bvm(x, mu, kappa, log_const = NULL)

const_bvm(kappa, M = 25, MC = 10000)

r_bvm(n, mu, kappa)

fit_bvm_mm(
  x,
  lower = c(0, 0, -30),
  upper = c(30, 30, 30),
  start = NULL,
  M = 25,
  hom = FALSE,
  indep = FALSE,
  ...
)

fit_bvm_mle(
  x,
  start = NULL,
  M = 25,
  lower = c(-pi, -pi, 0, 0, -30),
```

```
  upper = c(pi, pi, 30, 30, 30),
  hom = FALSE,
  indep = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| x | matrix of size c(nx, 2) with the angles on which the density is evaluated. |
| mu | circular means of the density, a vector of length 2. |
| kappa | vector of length 3 with the concentrations $(\kappa_1, \kappa_2)$ and the dependence parameter $\lambda$ of the density. |
| log_const | logarithm of the normalizing constant. Computed internally if NULL (default). |
| M | truncation of the series expansion for computing the normalizing constant. Defaults to 25. |
| MC | Monte Carlo replicates for computing the normalizing constant when there is no series expansion. Defaults to 1e4. |
| n | sample size. |
| lower, upper | vectors of length 5 with the bounds for the likelihood optimizer. Default to c(-pi, -pi, 0, 0, -30) and c(pi, pi, 30, 30, 30). |
| start | a vector of length 5 with the initial values for the maximum likelihood optimizer. The first two entries are disregarded in fit_bvm_mm. If NULL (default), the starting values are taken from the estimation of marginal von Mises in fit_bvm_mm. In fit_bvm_mle, the method of moments estimates are employed. |
| hom | assume a homogeneous distribution with equal marginal concentrations? Defaults to FALSE. |
| indep | set the dependence parameter to zero? Defaults to FALSE. |
| ... | further parameters passed to mleOptimWrapper. |

## Value

- d_bvm: a vector of length nx with the density evaluated at x.
- const_bvm: the value of the normalizing constant $T(\kappa_1, \kappa_2, \lambda)$.
- r_bvm: a matrix of size c(n, 2) with the random sample.
- fit_mme_bvm, fit_mle_bvm: a list with the estimated parameters $(\mu_1, \mu_2, \kappa_1, \kappa_2, \lambda)$ and the object opt containing the optimization summary.

## References

Mardia, K. V., Hughes, G., Taylor, C. C., and Singh, H. (2008). A multivariate von Mises with applications to bioinformatics. *Canadian Journal of Statistics*, 36(1):99–109. doi:10.1002/cjs.5550360110

Singh, H., Hnizdo, V., and Demchuk, E. (2002). Probabilistic model for two dependent circular variables. *Biometrika*, 89(3):719–723. doi:10.1093/biomet/89.3.719

## Examples

```
## Density evaluation

mu <- c(0, 0)
kappa <- 3:1
nth <- 50
th <- seq(-pi, pi, l = nth)
x <- as.matrix(expand.grid(th, th))
const <- const_bvm(kappa = kappa)
d <- d_bvm(x = x, mu = mu, kappa = kappa, log_const = log(const))
filled.contour(th, th, matrix(d, nth, nth), col = viridisLite::viridis(31),
               levels = seq(0, max(d), l = 30))

## Sampling and estimation

n <- 100
samp <- r_bvm(n = n, mu = mu, kappa = kappa)
(param_mm <- fit_bvm_mm(samp)$par)
(param_mle <- fit_bvm_mle(samp)$par)
```

---

bwc *Density evaluation, sampling, and parameter estimation of the bivariate wrapped Cauchy distribution*

---

## Description

Computation of the density of a bivariate wrapped Cauchy:

$f(\theta_1, \theta_2) = c(\xi_1, \xi_2, \rho)\{c_0(\xi_1, \xi_2, \rho) - c_1(\xi_1, \xi_2, \rho)\cos(\theta_1 - \mu_1) - c_2(\xi_1, \xi_2, \rho)\cos(\theta_2 - \mu_2) - c_3(\xi_1, \xi_2, \rho)\cos(\theta_1 - \mu_1)\cos(\theta_2$

Simulation of samples from a bivariate wrapped Cauchy.

Maximum likelihood and method of moments estimation of the parameters $(\mu_1, \mu_2, \xi_1, \xi_2, \rho)$.

## Usage

```
d_bwc(x, mu, xi)

r_bwc(n, mu, xi)

fit_bwc_mm(x, hom = FALSE, indep = FALSE)

fit_bwc_mle(
  x,
  start = NULL,
  lower = c(-pi, -pi, 0, 0, -1 + 0.001),
  upper = c(pi, pi, 1 - 0.001, 1 - 0.001, 1 - 0.001),
  hom = FALSE,
  indep = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| x | matrix of size c(nx, 2) with the angles on which the density is evaluated. |
| mu | circular means of the density, a vector of length 2. |
| xi | a vector of length 3 with the marginal concentrations $(\xi_1, \xi_2)$, and the dependence parameter $\rho$. |
| n | sample size. |
| hom | assume a homogeneous distribution with equal marginal concentrations? Defaults to FALSE. |
| indep | set the dependence parameter to zero? Defaults to FALSE. |
| start | a vector of length 5 with the initial values for the maximum likelihood optimizer. If NULL (default), the method of moments estimates are employed. |
| lower, upper | vectors of length 5 with the bounds for the likelihood optimizer. Default to c(-pi, -pi, 0, 0, -1 + 1e-3) and c(pi, pi, 1 - 1e-3, 1 - 1e-3, 1 - 1e-3). |
| ... | further parameters passed to mleOptimWrapper. |

## Value

- d_bwc: a vector of length nx with the density evaluated at x.
- r_bwc: a matrix of size c(n, 2) with the random sample.
- fit_mme_bwc, fit_mle_bwc: a list with the parameters $(\mu_1, \mu_2, \xi_1, \xi_2, \rho)$ and the object opt containing the optimization summary.

## Author(s)

The original code for r_bwc was supplied by Arthur Pewsey.

## References

Kato, S. and Pewsey, A. (2015). A Möbius transformation-induced distribution on the torus. *Biometrika*, 102(2):359–370. doi:10.1093/biomet/asv003

## Examples

```
## Density evaluation

mu <- c(0, 0)
xi <- c(0.3, 0.5, 0.4)
nth <- 50
th <- seq(-pi, pi, l = nth)
x <- as.matrix(expand.grid(th, th))
d <- d_bwc(x = x, mu = mu, xi = xi)
filled.contour(th, th, matrix(d, nth, nth), col = viridisLite::viridis(20),
               levels = seq(0, max(d), l = 20))

## Sampling and estimation

n <- 100
```

```
samp <- r_bwc(n = n, mu = mu, xi = xi)
(param_mm <- fit_bwc_mm(samp)$par)
(param_mle <- fit_bwc_mle(samp)$par)
```

---

bwn *Density evaluation, sampling, and parameter estimation of the bivariate wrapped normal distribution*

---

## Description

Computation of the density of the bivariate wrapped normal.

Simulation of pairs of samples from a bivariate wrapped normal.

Maximum likelihood estimation of the parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

## Usage

```
d_bwn(x, mu, Sigma, kmax = 2)

r_bwn(n, mu, Sigma)

fit_bwn_mle(
  x,
  kmax = 2,
  lower = c(-pi, -pi, 0.001, 0.001, -1),
  upper = c(pi, pi, 20, 20, 1),
  ...
)
```

## Arguments

| | |
|---|---|
| `x` | matrix of size `c(nx, 2)` with the angles on which the density is evaluated. |
| `mu` | circular means of the density, a vector of length 2. |
| `Sigma` | covariance matrix of size `c(2, 2)`. |
| `kmax` | integer number up to truncate the wrapped normal series in `-kmax:kmax`. Defaults to 2. |
| `n` | sample size. |
| `lower, upper` | vector of length 5 with the bounds of the parameters for the maximum likelihood optimizer. |
| `...` | further parameters passed to `mleOptimWrapper`. |

## Value

- `d_bwn`: a vector of length `nx` with the density evaluated at `x`.
- `r_bwn`: a matrix of size `c(n, 2)` with the random sample.
- `fit_bwn_mle`: a list with the parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and the object `opt` containing the optimization summary.

## Examples

```
## Density evaluation

mu <- c(0, 0)
Sigma <- 3 * matrix(c(1.5, 0.75, 0.75, 1), nrow = 2, ncol = 2)
nth <- 50
th <- seq(-pi, pi, l = nth)
x <- as.matrix(expand.grid(th, th))
d <- d_bwn(x = x, mu = mu, Sigma = Sigma)
filled.contour(th, th, matrix(d, nth, nth), col = viridisLite::viridis(31),
               levels = seq(0, max(d), l = 30))

## Sampling and estimation

n <- 100
samp <- r_bwn(n = 100, mu = mu, Sigma = Sigma)
(param_mle <- fit_bwn_mle(samp)$par)
```

---

# earthquakes *Japanese earthquakes dataset*

---

## Description

Pre-earthquake direction of steepest descent and the direction of lateral ground movement before and after, respectively, an earthquake in Noshiro (Japan) in 1983.

## Usage

```
earthquakes
```

## Format

A data frame with 678 rows and 2 variables:

**theta1** Direction of steepest descent.

**theta2** Direction of lateral ground movement.

## Details

The direction is measured in radians in $[0, 2\pi)$ with $0$ / $\frac{\pi}{2}$ / $\pi$ / $\frac{3\pi}{2}$ / $2\pi$ representing the East / North / West / South / East directions.

## References

Hamada, M. and O'Rourke, T. (1992). Case Studies of Liquefaction & Lifeline Performance During Past Earthquake. Volume 1: Japanese Case Studies. Technical Report NCEER-92-0001. National Center for Earthquake Engineering Research, University at Buffalo. doi:10.1016/0886-7798(93)90146m

Jones, M. C., Pewsey, A., and Kato, S. (2015). On a class of circulas: copulas for circular distributions. *Annals of the Institute of Statistical Mathematics*, 67(5):843–862. doi:10.1007/s10463014-04936

Rivest, L.-P. (1997). A decentred predictor for circular-circular regression. *Biometrika*, 84(3):717–726. doi:10.1093/biomet/84.3.717

## Examples

```
# Load data
data("earthquakes")

# Transform the data into [-pi, pi)
earthquakes <- sdetorus::toPiInt(earthquakes)
plot(earthquakes, xlab = expression(theta[1]), ylab = expression(theta[2]),
     xlim = c(-pi, pi), ylim = c(-pi, pi), axes = FALSE)
sdetorus::torusAxis()


# Perform TR-PCA
fit <- ridge_pca(x = earthquakes)
show_ridge_pca(fit)
```

---

`frechet` *Fréchet statistics on the torus*

---

## Description

Computes the Fréchet mean, variance, and standard deviation of a sample on the $d$-torus $[-l, l)^d$, $d \geq 1$, with $-l \equiv l$ identified.

## Usage

```
frechet_mean(x, l = pi, N = 500, draw_plot = FALSE)

frechet_ss(x, l = pi, N = 500, draw_plot = FALSE)
```

## Arguments

| | |
|---|---|
| `x` | sample of angles on $[-l, l)$, a vector or a matrix. |
| `l` | half-period of the circular data. Can be a vector of length `ncol(x)` if `x` is a matrix. Defaults to `pi`. |
| `N` | size of the grid in $[-l, l)$ for the exhaustive search of the mean. Defaults to `5e2`. |
| `draw_plot` | draw a diagnostic plot showing the Fréchet loss function? Defaults to `FALSE`. |

## Value

- `frechet_mean`: a list with the marginal Fréchet means (`mu`), variances (`var`), and standard deviations (`sd`).
- `frechet_ss`: a list with the Fréchet variances (`var`) and the cumulative proportion of total variance explained (`var_exp`).

## Examples

```
## Circular data

# Sample from a wrapped normal
x <- sdetorus::toPiInt(rnorm(n = 5e2, mean = 2, sd = 1))
frechet_mean(x = x)

# Sample from a bimodal distribution
x <- sdetorus::toPiInt(rnorm(n = 5e2, mean = c(1, -2), sd = c(0.5, 0.75)))
frechet_mean(x = x)

# Periodic data in [-2, 2)
x <- sdetorus::toInt(rnorm(n = 5e2, mean = c(-2, 1), sd = 2:1),
                     a = -2, b = 2)
frechet_mean(x = x, l = 2)

## Toroidal data

# Sample from a multivariate wrapped normal
n <- 50
S <- rbind(c(2.5, -0.2, 0.5),
           c(-0.2, 1.5, -0.5),
           c(0.5, -0.5, 0.75))
x <- sdetorus::toPiInt(mvtnorm::rmvnorm(n, mean = c(0, 1.5, -2), sigma = S))
(f <- frechet_mean(x = x))

# Total Fréchet variance is sum of marginal variances
sum(torus_dist(x, y = f$mu, squared = TRUE)) / n
sum(f$var)

# Cumulative proportion of variances
frechet_ss(x)
```

---

`ridge_curve` *Fourier-fitted ridge curve and related utilities*

---

## Description

Given the angles `theta` in $[-\pi, \pi)$, `ridge_curve` computes the Fourier-fitted ridge curve $(\theta, r_1(\theta))$ or $(r_2(\theta), \theta)$, where

$$r_j(\theta) := \operatorname{atan2}(S_m(\theta), C_m(\theta))$$

with $C_m(x) := a_0/2 + \sum_{k=1}^{m} a_k \cos(kx)$ and $S_m(x) := \sum_{k=1}^{m} b_k \sin(kx)$ for $j = 1, 2$. `der_ridge_curve` and `dist_ridge_curve` compute the derivatives of and the distances along these curves, respectively. `alpha_ridge_curve` provides a uniform grid of the ridge curve using the arc-length parametrization. `proj_ridge_curve` gives the ridge's $\theta$ for which the curve is closer to any point on $[-\pi, \pi)^2$.

## Usage

```
ridge_curve(
  theta,
  mu = c(0, 0),
  coefs = list(cos_a = c(0, 0), sin_b = 0),
  ind_var = 1,
  at2 = TRUE
)

der_ridge_curve(
  theta,
  mu = c(0, 0),
  coefs = list(cos_a = c(0, 0), sin_b = 0),
  ind_var = 1,
  norm = NULL,
  at2 = TRUE
)

dist_ridge_curve(
  alpha,
  mu = c(0, 0),
  coefs = list(cos_a = c(0, 0), sin_b = 0),
  ind_var = 1,
  N = 500,
  der = TRUE,
  shortest = TRUE,
  at2 = TRUE
)

arclength_ridge_curve(
  mu = c(0, 0),
  coefs = list(cos_a = c(0, 0), sin_b = 0),
  ind_var = 1,
  N = 500,
  L = 500,
  at2 = TRUE
)

proj_ridge_curve(
  x,
  mu = c(0, 0),
  coefs = list(cos_a = c(0, 0), sin_b = 0),
  ind_var = 1,
```

```
  N = 500,
  ridge_curve_grid = NULL,
  arclength = FALSE,
  at2 = TRUE
)
```

## Arguments

| | |
|---|---|
| theta | vector $\theta$ of size nth. |
| mu | a vector of size 2 giving $(\mu_1, \mu_2)$. Defaults to c(0, 0). |
| coefs | list of coefficients cos_a ($a_k$) and sin_b ($b_k$ giving the Fourier fit of the ridge curve. Defaults to list(cos_a = c(0, 0), sin_b = 0). See examples. |
| ind_var | index $j$ of the variable that parametrizes the ridge. Defaults to 1. |
| at2 | do the atan2 fit instead of the sine fit (only using $S_m$)? Defaults to TRUE. at2 = FALSE is not recommended to use. |
| norm | normalize tangent vectors? If different from NULL (the default), the vectors are normalized to have the given norm. |
| alpha | a vector of size 2. |
| N | number of discretization points for approximating curve lengths. Defaults to 5e2. |
| der | use derivatives to approximate curve lengths? Defaults to TRUE. |
| shortest | return the shortest possible distance? Defaults to TRUE. |
| L | number of discretization points for computing the arc-length parametrization curve lengths. Defaults to 5e2. |
| x | a matrix of size c(nx, 2) with angular coordinates. |
| ridge_curve_grid | if provided, the ridge_curve evaluated at a grid of size N. If not provided, it is computed internally. Useful for saving computations. |
| arclength | use the arc-length parametrization to compute the projections? This yields a more uniform grid for searching the projections. Defaults to TRUE. |

## Value

- ridge_curve: a matrix of size c(nth, 2) with the ridge curve evaluated at theta.
- der_ridge_curve: a matrix of size c(nth, 2) with the derivatives of the ridge curve evaluated at theta.
- dist_ridge_curve: the distance between two points along the ridge curve, a non-negative scalar.
- proj_ridge_curve: a list with (1) the theta's that give the points in the ridge curve that are the closest (in the flat torus distance) to x (a matrix of size c(nx, 2)); (2) the indexes of ridge_curve_grid in which those theta's were obtained.
- arclength_ridge_curve: a vector of size N giving the theta angles that yield a uniform-length grid of the ridge curve.

## Examples

```
mu <- c(-0.5, 1.65)
th <- seq(-pi, pi, l = 200)
K <- 5
coefs <- list(cos_a = 1 / (1:(K + 1))^3, sin_b = 1 / (1:K)^3)
rid1 <- ridge_curve(theta = th, mu = mu, coefs = coefs, ind_var = 1)
rid2 <- ridge_curve(theta = th, mu = mu, coefs = coefs, ind_var = 2)
plot(mu[1], mu[2], xlim = c(-pi, pi), ylim = c(-pi, pi), axes = FALSE,
     xlab = expression(theta[1]), ylab = expression(theta[2]),
     pch = "*", col = 5, cex = 3)
sdetorus::linesTorus(rid1[, 1], rid1[, 2], col = 1)
sdetorus::linesTorus(rid2[, 1], rid2[, 2], col = 2)
abline(v = mu[1], lty = 3, col = 5)
abline(h = mu[2], lty = 3, col = 5)
points(ridge_curve(theta = mu[1], mu = mu, coefs = coefs, ind_var = 1),
       col = 1)
points(ridge_curve(theta = mu[2], mu = mu, coefs = coefs, ind_var = 2),
       col = 2)
sdetorus::torusAxis()

## der_ridge_curve

th <- seq(-pi, pi, l = 10)
mu <- c(0.5, 1.5)
K <- 5
coefs <- list(cos_a = 1 / (1:(K + 1))^3, sin_b = 1 / (1:K)^3)
rid1 <- ridge_curve(theta = th, mu = mu, coefs = coefs, ind_var = 1)
rid2 <- ridge_curve(theta = th, mu = mu, coefs = coefs, ind_var = 2)
v1 <- der_ridge_curve(theta = th, mu = mu, coefs = coefs, ind_var = 1,
                      norm = 0.5)
v2 <- der_ridge_curve(theta = th, mu = mu, coefs = coefs, ind_var = 2,
                      norm = 0.5)
points(rid1, pch = 16, col = 1)
points(rid2, pch = 16, col = 2)
arrows(x0 = rid1[, 1], y0 = rid1[, 2],
       x1 = (rid1 + v1)[, 1], y1 = (rid1 + v1)[, 2],
       col = 3, angle = 5, length = 0.1)
arrows(x0 = rid2[, 1], y0 = rid2[, 2],
       x1 = (rid2 + v2)[, 1], y1 = (rid2 + v2)[, 2],
       col = 4, angle = 5, length = 0.1)

## dist_ridge_curve

# Distances accuracy
a <- c(-pi / 2, pi)
mu <- c(-pi / 2, pi / 2)
dist_ridge_curve(alpha = a, mu = mu, coefs = coefs, der = TRUE, N = 1e6)
dist_ridge_curve(alpha = a, mu = mu, coefs = coefs, der = FALSE, N = 1e6)
dist_ridge_curve(alpha = a, mu = mu, coefs = coefs, der = TRUE, N = 1e2)
dist_ridge_curve(alpha = a, mu = mu, coefs = coefs, der = FALSE, N = 1e2)

## arclength_ridge_curve
```

```
mu <- c(-pi / 2, pi / 2)
alpha <- arclength_ridge_curve(mu = mu, coefs = coefs, ind_var = 1, N = 25)
alpha <- sdetorus::toPiInt(c(alpha, alpha[1]))
rid <- ridge_curve(theta = alpha, mu = mu, coefs = coefs, ind_var = 1)
plot(mu[1], mu[2], pch = "*", col = 5, cex = 3, xlim = c(-pi, pi),
     ylim = c(-pi, pi), axes = FALSE, xlab = expression(theta[1]),
     ylab = expression(theta[2]))
sdetorus::linesTorus(rid[, 1], rid[, 2], col = 1, pch = 16)
points(rid[, 1], rid[, 2], pch = 16, col = 1)
abline(v = mu[1], lty = 3, col = 5)
abline(h = mu[2], lty = 3, col = 5)
sdetorus::torusAxis()

## proj_ridge_curve

mu <- c(0, 0)
n <- 25
x <- matrix(runif(2 * n, -pi, pi), nrow = n, ncol = 2)
col <- rainbow(n)
th <- seq(-pi, pi, l = 100)
old_par <- par(no.readonly = TRUE)
par(mfrow = c(1, 2))
for (j in 1:2) {

  plot(x, xlim = c(-pi, pi), ylim = c(-pi, pi), axes = FALSE,
       xlab = expression(theta[1]), ylab = expression(theta[2]), col = col)
  rid <- ridge_curve(theta = th, mu = mu, coefs = coefs, ind_var = j)
  sdetorus::linesTorus(x = rid[, 1], y = rid[, 2], lwd = 2)
  abline(v = mu[1], lty = 3)
  abline(h = mu[2], lty = 3)
  points(mu[1], mu[2], pch = "*", cex = 3)
  sdetorus::torusAxis()
  theta_projs <- proj_ridge_curve(x = x, mu = mu, coefs = coefs, ind_var = j,
                                  ridge_curve_grid = rid)$theta_proj
  projs <- ridge_curve(theta = theta_projs, mu = mu, coefs = coefs,
                       ind_var = j)
  points(projs, col = col, pch = 3)
  for (i in 1:n) {

    sdetorus::linesTorus(x = c(x[i, 1], projs[i, 1]),
                         y = c(x[i, 2], projs[i, 2]), col = col[i], lty = 3)

  }

}
par(old_par)
```

---

`ridge_distr` *Connected component of the toroidal density ridge of a bivariate sine von Mises, bivariate wrapped Cauchy, and bivariate wrapped normal*

---

## Description

Computation of the connected component of the density ridge of in a given set of points or, if not specified, in a regular grid on $[-\pi, \pi)$.

## Usage

```
ridge_bvm(mu, kappa, eval_points, subint_1, subint_2)

ridge_bwc(mu, xi, eval_points, subint_1, subint_2)

ridge_bwn(mu, Sigma, kmax = 2, eval_points, subint_1, subint_2)
```

## Arguments

| | |
|---|---|
| `mu` | circular means of the density, a vector of length 2. |
| `kappa` | vector of length 3 with the concentrations $(\kappa_1, \kappa_2)$ and the dependence parameter $\lambda$ of the density. |
| `eval_points` | evaluation points for the ridge. |
| `subint_1` | number of points for $\theta_1$. |
| `subint_2` | number of points for $\theta_2$ at each $\theta_1$. |
| `xi` | a vector of length 3 with the marginal concentrations $(\xi_1, \xi_2)$, and the dependence parameter $\rho$. |
| `Sigma` | covariance matrix of size `c(2, 2)`. |
| `kmax` | integer number up to truncate the wrapped normal series in `-kmax:kmax`. Defaults to 2. |

## Value

A matrix of size `c(subint_1, 2)` containing the points of the connected component of the ridge.

## References

Ozertem, U. and Erdogmus, D. (2011). Locally defined principal curves and surfaces. *Journal of Machine Learning Research*, 12(34):1249–1286. doi:10.6083/M4ZG6Q60

## Examples

```
# Bivariate von Mises
mu <- c(0, 0)
kappa <- c(0.3, 0.5, 0.4)
nth <- 100
th <- seq(-pi, pi, l = nth)
x <- as.matrix(expand.grid(th, th))
d <- d_bvm(x = x, mu = mu, kappa = kappa)
image(th, th, matrix(d, nth, nth), col = viridisLite::viridis(20))
ridge <- ridge_bvm(mu = mu, kappa = kappa, subint_1 = 5e2,
                   subint_2 = 5e2)
points(ridge)
```

```
# Bivariate wrapped Cauchy
mu <- c(0, 0)
xi <- c(0.3, 0.6, 0.25)
nth <- 100
th <- seq(-pi, pi, l = nth)
x <- as.matrix(expand.grid(th, th))
d <- d_bwc(x = x, mu = mu, xi = xi)
image(th, th, matrix(d, nth, nth), col = viridisLite::viridis(20))
ridge <- ridge_bwc(mu = mu, xi = xi, subint_1 = 5e2, subint_2 = 5e2)
points(ridge)

# Bivariate wrapped normal
mu <- c(0, 0)
Sigma <- matrix(c(10, 3, 3, 5), nrow = 2)
nth <- 100
th <- seq(-pi, pi, l = nth)
x <- as.matrix(expand.grid(th, th))
d <- d_bwn(x = x, mu = mu, Sigma = Sigma)
image(th, th, matrix(d, nth, nth), col = viridisLite::viridis(20))
ridge <- ridge_bwn(mu = mu, Sigma = Sigma, subint_1 = 5e2,
                   subint_2 = 5e2)
points(ridge)
```

---

`ridge_fourier_fit` *Fourier expansion of a given curve*

---

## Description

Computation of the Fourier expansion coefficients of a given curve.

## Usage

```
ridge_fourier_fit(curve, K = 15, norm_prop = 1, N = 1280, at2 = TRUE)
```

## Arguments

| | |
|---|---|
| `curve` | points of the curve. |
| `K` | number of terms in the Fourier expansion. Defaults to `15`. |
| `norm_prop` | percentage of explained norm. Defaults to `1`. |
| `N` | number of Gaussian quadrature points, passed to Gauss_Legen_nodes. Defaults to `1280`. |
| `at2` | do the `atan2` fit instead of the sine fit (only using $S_m$)? Defaults to `TRUE`. `at2 = FALSE` is not recommended to use. |

## Value

The coefficients of the fit (see ridge_curve). A list with entries:

| | |
|---|---|
| cos_a | contains $a_0, a_1, \ldots, a_m$. |
| sin_b | contains $b_1, \ldots, b_m$. |

## Examples

```
# Zero mean
ridge0 <- ridge_bvm(mu = c(0, 0), kappa = c(1, 2, -5), subint_1 = 5e2,
                    subint_2 = 5e2)
coefs <- ridge_fourier_fit(ridge0)
th <- seq(-pi, pi, l = 500)
plot(ridge0, xlim = c(-pi, pi), ylim = c(-pi, pi))
points(ridge_curve(th, mu = c(0, 0), coefs = coefs, at2 = TRUE), col = 3,
       cex = 0.5)

# Non-zero mean from a zero-mean ridge
mu <- c(1.4, 2)
ridge1 <- ridge_bvm(mu = mu, kappa = c(1, 2, -5), subint_1 = 5e2,
                    subint_2 = 5e2) # Just for plot
plot(ridge1, xlim = c(-pi, pi), ylim = c(-pi, pi))
points(mu[1], mu[2], col = 4, pch = "*", cex = 5)
points(ridge_curve(th, mu = mu, coefs = coefs), col = 3, cex = 0.5)

# Other zero-mean example
mu <- c(0, 0)
ridge <- ridge_bwc(mu = mu, xi = c(0.3, 0.5, 0.7), subint_1 = 5e2,
                   subint_2 = 5e2)
plot(ridge, xlim = c(-pi, pi), ylim = c(-pi, pi))
coefs <- ridge_fourier_fit(ridge)
points(ridge_curve(th, mu = mu, coefs = coefs), col = 4, cex = 0.5)

# Another zero-mean example
mu <- c(0, 0)
ridge <- ridge_bwc(mu = mu, xi = c(0.8, 0.1, 0.75), subint_1 = 5e2,
                   subint_2 = 5e2)
plot(ridge, xlim = c(-pi, pi), ylim = c(-pi, pi))
coefs <- ridge_fourier_fit(ridge)
points(ridge_curve(th, mu = mu, coefs = coefs), col = 4, cex = 0.5)
```

---

ridge_pca *Toroidal PCA via density ridges*

---

## Description

This function computes the whole process of toroidal PCA via density ridges on a given sample: parameter estimation of the underlying distribution, estimation of the connected component of the ridge, and determination of its Fourier expansion from which to obtain the first and second scores.

## Usage

```
ridge_pca(
  x,
  type = c("auto", "bvm", "bwc")[1],
  N = 500,
  K = 15,
  scale = TRUE,
  lrts = TRUE,
  alpha = 0.05,
  at2 = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| x | matrix of dimension c(n, 2) containing the n observations of the pair of angles. |
| type | either "bvm" (bivariate sine von Mises), "bwc" (bivariate wrapped Cauchy), or "auto" (default). "auto" performs both fits and uses the one with lowest BIC. |
| N | number of discretization points for approximating curve lengths. Defaults to 5e2. |
| K | number of terms in the Fourier expansion. Defaults to 15. |
| scale | scale the resulting scores to $[-\pi, \pi)^2$? Defaults to TRUE. |
| lrts | run biv_lrt to check the null hypothesis of homogeneous concentration parameters using likelihood ratio tests? If TRUE (default), enforces the special horizontal/vertical/diagonal cases to become "sticky" fits. |
| alpha | significance level for the homogeneity test. |
| at2 | do the atan2 fit instead of the sine fit (only using $S_m$)? Defaults to TRUE. at2 = FALSE is not recommended to use. |
| ... | optional parameters passed to fit_bvm_mle and fit_bwc_mle, such as start, lower, or upper. |

## Value

A list with:

| | |
|---|---|
| mu_hat | estimated circular means of the sample. |
| coefs_hat | estimated Fourier coefficients. |
| ind_var | indexing variable. |
| scores | scores for each of the sample points. |
| var_exp | percentage of explained variance. |
| fit_mle | maximum likelihood fit. |
| bic_fit | BIC of the fit. |
| data | original sample. |
| scales | vector of length 2 with the scale limits for the axes. |

| | |
|---|---|
| type | type of fit performed. |
| p_hom | $p$-value of the homogeneity test. |
| p_indep | $p$-value of the independence test. |

## Examples

```
## Bivariate von Mises

n <- 100
x <- r_bvm(n = n, mu = c(1, 2), kappa = c(0.4, 0.4, 0.5))
fit <- ridge_pca(x = x, type = "bvm")
show_ridge_pca(fit = fit, col_data = "red")

x <- r_bvm(n = n, mu = c(2, 1), kappa = c(1, 2, 0))
fit <- ridge_pca(x = x, type = "bvm")
show_ridge_pca(fit = fit, col_data = "red")

x <- r_bvm(n = n, mu = c(2, 1), kappa = c(3, 2, 0))
fit <- ridge_pca(x = x, type = "bvm")
show_ridge_pca(fit = fit, col_data = "red")

## Bivariate wrapped Cauchy

x <- r_bwc(n = n, mu = c(1, 2), xi = c(0.2, 0.2, 0.5))
fit <- ridge_pca(x = x, type = "bwc")
show_ridge_pca(fit = fit, col_data = "red")

x <- r_bwc(n = n, mu = c(1, 2), xi = c(0.2, 0.8, 0))
fit <- ridge_pca(x = x, type = "bwc")
show_ridge_pca(fit = fit, col_data = "red")

x <- r_bwc(n = n, mu = c(1, 2), xi = c(0.5, 0.2, 0))
fit <- ridge_pca(x = x, type = "bwc")
show_ridge_pca(fit = fit, col_data = "red")
```

---

# ridge_scores *Scores and scales for Fourier-fitted ridge curves*

---

## Description

Computation of PCA scores for Fourier-fitted ridge curves. The scores are defined as follows:

- First scores: signed distances along the ridge curve of the data projections to $\mu$.
- Second scores: signed toroidal distances from the data points to their ridge projections.

The scores can be scaled to $(-\pi, \pi)$ or remain as $(l/2, m_2)$, where $l$ is the length of the curve and $m_2$ is the maximal absolute second score.

## Usage

```
ridge_scores(
  x,
  mu = c(0, 0),
  coefs = list(cos_a = c(0, 0), sin_b = 0),
  ind_var = 1,
  N = 500,
  scale = TRUE,
  at2 = TRUE
)

max_score_2(
  mu = c(0, 0),
  coefs = list(cos_a = c(0, 0), sin_b = 0),
  ind_var = 1,
  L = 25,
  f = 2,
  at2 = TRUE
)
```

## Arguments

| | |
|---|---|
| x | a matrix of size c(nx, 2) with angular coordinates. |
| mu | a vector of size 2 giving $(\mu_1, \mu_2)$. Defaults to c(0, 0). |
| coefs | list of coefficients cos_a ($a_k$) and sin_b ($b_k$ giving the Fourier fit of the ridge curve. Defaults to list(cos_a = c(0, 0), sin_b = 0). See examples. |
| ind_var | index $j$ of the variable that parametrizes the ridge. Defaults to 1. |
| N | number of discretization points for approximating curve lengths. Defaults to 5e2. |
| scale | scale the resulting scores to $[-\pi, \pi)^2$? Defaults to TRUE. |
| at2 | do the atan2 fit instead of the sine fit (only using $S_m$)? Defaults to TRUE. at2 = FALSE is not recommended to use. |
| L | grid along he variable ind_var used for searching the maximum allowed second score. Defaults to 25. |
| f | factor for shrinking the grid on the variable that is different to ind_var. Defaults to 2. |

## Details

The mean $\mu$ corresponds to the first score being null.

## Value

ridge_scores returns a list with:

| | |
|---|---|
| scores | a matrix of size c(nx, 2) with the ridge scores. |

scales a vector of length 2 with the scale limits for the axes.

max_score_2 computes the maximum allowed second score to rescale if scale = TRUE.

## Examples

```
mu <- c(-0.5, 1.65)
th <- seq(-pi, pi, l = 200)
K <- 5
coefs <- list(cos_a = 1 / (1:(K + 1))^3, sin_b = 1 / (1:K)^3)
n <- 10
col <- rainbow(n)

set.seed(13213)
old_par <- par(no.readonly = TRUE)
par(mfrow = c(2, 2))
for (j in 1:2) {

  # Simulate synthetic data close to the ridge curve
  rid <- ridge_curve(theta = th, mu = mu, coefs = coefs, ind_var = j)
  ind <- sort(sample(length(th), size = n))
  eps <- 0.25 * matrix(runif(2 * n, -1, 1), nrow = n, ncol = 2)
  x <- sdetorus::toPiInt(rid[ind, ] + eps)

  # Plot ridge and synthetic data, with signs from the second scores
  s <- ridge_scores(x, mu = mu, coefs = coefs, ind_var = j)$scores
  plot(x, xlim = c(-pi, pi), ylim = c(-pi, pi), axes = FALSE,
       xlab = expression(theta[1]), ylab = expression(theta[2]), col = col,
       pch = ifelse(sign(s[, 2]) == 1, "+", "-"), cex = 1.25)
  sdetorus::linesTorus(rid[, 1], rid[, 2], lwd = 2)
  abline(v = mu[1], lty = 3)
  abline(h = mu[2], lty = 3)
  points(mu[1], mu[2], pch = "*", cex = 3)
  sdetorus::torusAxis()

  # Projections
  theta_projs <- proj_ridge_curve(x = x, mu = mu, coefs = coefs,
                                  ind_var = j, ridge_curve_grid = rid,
                                  )$theta_proj
  projs <- ridge_curve(theta = theta_projs, mu = mu, coefs = coefs,
                       ind_var = j)
  for (i in 1:n) {

    sdetorus::linesTorus(x = c(x[i, 1], projs[i, 1]),
                         y = c(x[i, 2], projs[i, 2]),
                         col = col[i], lty = 3)

  }

  # Scores plot
  plot(s, xlim = c(-pi, pi), ylim = c(-pi, pi), axes = FALSE,
       xlab = "Score 1", ylab = "Score 2", col = col,
       pch = ifelse(sign(s[, 2]) == 1, "+", "-"))
```

```
  sdetorus::torusAxis()
  abline(v = 0, lty = 3)
  abline(h = 0, lty = 3)
  points(0, 0, pch = "*", cex = 3)

}
par(old_par)
```

---

`santabarbara` *Santa Barbara currents*

---

## Description

The Santa Barbara Channel is a coastal area in California. This dataset contains the sea currents in the four areas present in the data application in García-Portugués and Prieto-Tirado (2022). Precisely, it contains the 24-hour speed-weighted mean of the currents' direction in each of the four areas downloaded from the NOAA High Frequency Radar National Server.

## Usage

```
santabarbara
```

## Format

A data frame with 1092 rows and 4 variables:

**A** Sea current direction at zone A.

**B** Sea current direction at zone B.

**C** Sea current direction at zone C.

**D** Sea current direction at zone D.

## Details

The selection of these four areas is motivated by previous studies on the Santa Barbara currents, like Auad et al. (1998). The direction is measured in radians in $[-\pi, \pi)$ with $-\pi$ / $-\frac{\pi}{2}$ / 0 / $\frac{\pi}{2}$ / $\pi$ representing the East / South / West / North / East directions. The script performing the data preprocessing is available at `data-acquisition.R`. The data was retrieved on 2022-10-21.

## References

Auad, G., Hendershott, M. C., and Winant, C. D. (1998). Wind-induced currents and bottom-trapped waves in the Santa Barbara Channel. *Journal of Physical Oceanography*, 28(1):85–102. doi:10.1175/15200485(1998)028<0085:WICABT>2.0.CO;2

García-Portugués, E. and Prieto-Tirado, A. (2023). Toroidal PCA via density ridges. *Statistics and Computing*, 33(5):107. doi:10.1007/s11222023102739

## Examples

```
# Load data
data("santabarbara")
AB_zone <- santabarbara[c("A","B")]


# Perform TR-PCA
fit <- ridge_pca(x = AB_zone)
show_ridge_pca(fit)
```

---

show_ridge_pca *Illustration of toroidal PCA via density ridges*

---

## Description

Shows the scores computation for PCA via density ridges on $[-\pi, \pi)^2$.

## Usage

```
show_ridge_pca(
  fit,
  n_max = 500,
  projs = TRUE,
  projs_lines = TRUE,
  signs = TRUE,
  col_data = 1,
  col_projs = c(3, 4),
  main = "",
  N = 500,
  at2 = TRUE
)
```

## Arguments

| | |
|---|---|
| fit | the output of ridge_pca. |
| n_max | maximum number of data points to draw. These are sampled from the data provided. Defaults to 500. |
| projs | draw projections? Defaults to TRUE. |
| projs_lines | draw projection lines? Defaults to TRUE. |
| signs | plot the original data points with + and - facets depending on the signs of the second scores? Defaults to TRUE. |
| col_data | color(s) for the data points. Defaults to 1. |
| col_projs | a vector of size 2 giving the colors for the curve-projected data and the ridge curve, respectively. Defaults to c(3, 4). |
| main | caption of the plot. Empty by default. |

| | |
|---|---|
| N | number of discretization points for approximating curve lengths. Defaults to 5e2. |
| at2 | do the atan2 fit instead of the sine fit (only using $S_m$)? Defaults to TRUE. at2 = FALSE is not recommended to use. |

## Value

Nothing, the functions are called to produce plots.

## Examples

```
# Generate data
set.seed(987654321)
n <- 50
S1 <- rbind(c(1, -0.7), c(-0.7, 1))
S2 <- rbind(c(1, 0.5), c(0.5, 1))
x <- rbind(mvtnorm::rmvnorm(n, mean = c(0, pi / 2), sigma = S1),
           mvtnorm::rmvnorm(n, mean = c(pi, -pi / 2), sigma = S2))
x <- sdetorus::toPiInt(x)
col <- rainbow(2)[rep(1:2, each = n)]

# ridge_pca and its visualization
fit <- ridge_pca(x = x, at2 = FALSE)
show_ridge_pca(fit = fit, col_data = col, at2 = FALSE)
fit2 <- ridge_pca(x = x, at2 = TRUE)
show_ridge_pca(fit = fit2, col_data = col, at2 = TRUE)
```

---

# torus_dist *Toroidal distances*

---

## Description

Computation of distances on $[-\pi, \pi)^d$, $d \geq 1$, between two sets of observations.

## Usage

```
torus_dist(x, y, squared = FALSE)
```

## Arguments

| | |
|---|---|
| x | a matrix of size c(nx, d) with angles on $[-\pi, \pi)$. |
| y | either a matrix with the same size as x or a vector of size nx. |
| squared | return the squared distance? Defaults to FALSE. |

## Details

The maximal distance on $[-\pi, \pi)^d$ is $\sqrt{d}\pi$.

## Value

A vector of size nx with the distances between the observations of x and y.

## Examples

```
# Illustration of torus distances
n <- 10
x <- matrix(runif(2 * n, -pi, pi), nrow = n, ncol = 2)
y <- c(pi / 2, pi / 3)
col <- rainbow(n)
plot(x, xlim = c(-pi, pi), ylim = c(-pi, pi), axes = FALSE, col = col,
     xlab = expression(theta[1]), ylab = expression(theta[2]), pch = 16)
sdetorus::torusAxis()
points(y[1], y[2], col = 1, pch = 17)
for (i in 1:n) {

  sdetorus::linesTorus(x = c(x[i, 1], y[1]),
                       y = c(x[i, 2], y[2]), lty = 2, col = col[i])

}
text(x = x, labels = sprintf("%.2f", torus_dist(x, y)), col = col, pos = 1)
```

---

# torus_pairs *Toroidal pairs plot*

---

## Description

Pairs plots for data on $[-\pi, \pi)^d$, $d \geq 2$. The diagonal panels contain kernel density estimates tailored to circular data.

## Usage

```
torus_pairs(
  x,
  max_dim = 10,
  columns = NULL,
  col_data = 1,
  ylim_dens = c(0, 1.5),
  bwd = "ROT",
  scales = rep(pi, ncol(x))
)
```

## Arguments

| | |
|---|---|
| x | a matrix of size c(nx, d) with angles on $[-\pi, \pi)$. |
| max_dim | the maximum number of scores to produce the scores plot. Defaults to 10. |
| columns | if specified, the variables to be plotted. If NULL (the default), the first max_dim variables are plotted. |

| | |
|---|---|
| col_data | color(s) for the data points. Defaults to 1. |
| ylim_dens | common ylim for the diagonal plots. Defaults to c(0, 1). |
| bwd | type of bandwidth selector used in the kernel density plots. Either "ROT", "EMI", "AMI", "LSCV", or "LCV". See bw_dir_pi and bw_dir_cv. Defaults to "ROT". |
| scales | scales of the torus. Defaults to rep(pi, ncol(x)). |

## Details

The default bandwidth selector is the Rule-Of-Thumb (ROT) selector in García-Portugués (2013). It is fast, yet it may oversmooth non-unimodal densities. The EMI selector gives more flexible fits.

## Value

A ggplot. The density plots show the Fréchet means (red bars) and the Fréchet standard deviations (gray text).

## References

García-Portugués, E. (2013). Exact risk improvement of bandwidth selectors for kernel density estimation with directional data. *Electronic Journal of Statistics*, 7:1655–1685. doi:10.1214/13-ejs821

## Examples

```
# Generate data
n <- 50
set.seed(123456)
x <- sdetorus::toPiInt(rbind(
  mvtnorm::rmvnorm(n = n, mean = c(-pi, -pi) / 2,
                   sigma = diag(0.1, nrow = 2)),
  mvtnorm::rmvnorm(n = n, mean = c(-3 * pi / 2, 0) / 2,
          sigma = diag(0.1, nrow = 2)),
  mvtnorm::rmvnorm(n = n, mean = c(0, pi / 2),
                   sigma = diag(0.1, nrow = 2))
))
col <- rainbow(3)[rep(1:3, each = n)]

# Torus pairs
torus_pairs(x, col_data = col)

fit <- ridge_pca(x = x)
torus_pairs(fit$scores, col_data = col)
```

wind *Texas wind dataset*

## Description

Wind direction at 6:00 and 7:00 from June 1, 2003 to June 30, 2003, in radians, measured at a weather station in Texas coded as C28-1.

## Usage

```
wind
```

## Format

A data frame with 30 rows and 2 variables:

**theta1** Direction at 6:00 am.

**theta2** Direction at 12:00 noon.

## Details

The direction is measured in radians in $[-\pi, \pi)$ with $-\pi/-\frac{-\pi}{2}/0/-\frac{\pi}{2}/\pi$ representing the East/South/West/North/East directions.

## References

Johnson, R. A. and Wehrly, T. (1977). Measures and models for angular correlation and angular-linear correlation. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(2):222–229. https://www.jstor.org/stable/2984799

## Examples

```
# Load data
data("wind")
plot(wind, xlab = expression(theta[1]), ylab = expression(theta[2]),
     xlim = c(-pi, pi), ylim = c(-pi, pi), axes = FALSE)
sdetorus::torusAxis()


# Perform TR-PCA
fit <- ridge_pca(x = wind)
show_ridge_pca(fit)
```

# Index