Package ‘overlap’

July 22, 2025

Type Package

Title Estimates of Coefficient of Overlapping for Animal Activity Patterns

Version 0.3.9

Date 2024-01-16

Depends suntools

Suggests sp

Description Provides functions to fit kernel density functions to data on temporal activity patterns of animals; estimate coefficients of overlapping of densities for two species; and calculate bootstrap estimates of confidence intervals. As in Ridout and Linkie (2009) <doi:10.1198/jabes.2009.08038>.

License GPL (>= 3)

NeedsCompilation yes

RoxygenNote 7.2.3

Author Mike Meredith [aut],
Martin Ridout [aut],
Liz A.D. Campbell [aut, cre] (ORCID: <`https://orcid.org/0000-0002-8302-7430`>)

Maintainer Liz A.D. Campbell <`lizadcampbell@gmail.com`>

Repository CRAN

Date/Publication 2024-01-19 11:10:02 UTC

Contents

`overlap-package` *Functions to estimate overlap of temporal activity patterns of animals*

Description

The times recorded on camera trap photos provide information on the period during the day that a species is most active. Species active at the same periods may interact as predator and prey, or as competitors. The functions in this package allow the overlap to be quantified, and provide means of estimating confidence intervals with bootstraps.

Details

The functions in this package were originally optimised for a simulation study. Hence, speed is important and checking of input is minimal. It is the user's responsibility to make sure that input is valid.

In particular, note that all times are measured in **radians**. If your original data use 0-24 hours or 0-1 days, convert to radians: see the example in `kerinci`. If you need fitted densities in other units, use the output from `densityPlot` or `overlapPlot`.

Author(s)

Mike Meredith, based on work by Martin Ridout.

References

Ridout & Linkie (2009) Estimating overlap of daily activity patterns from camera trap data. *Journal of Agricultural, Biological, and Environmental Statistics* 14:322-337

See Also

See `overlapTrue` for the definition of the coefficient of overlapping, and `overlapEst` for equations for the estimators. See `kerinci` for an example of calculation of overlap and confidence interval from real data.

The R package **activity** has more functions for analysis of animal activity patterns.

Examples

```
# Get example data:
data(simulatedData)

# Use defaults:
overlapEst(tigerObs, pigObs)
#     Dhat1     Dhat4     Dhat5
# 0.2908618 0.2692011 0.2275000

overlapEst(tigerObs, pigObs, type="Dhat4")
# Dhat4
# 0.2692011
```

`Bootstrap confidence intervals` *Confidence interval calculation from bootstrap samples.*

Description

`bootCI` calculates five different confidence intervals from bootstrap samples: see details: `bootCIlogit` calculates corrections on the logit scale and back-transforms.

Usage

```
bootCI(t0, bt, conf = 0.95)
bootCIlogit(t0, bt, conf = 0.95)
```

Arguments

`t0`	the statistic estimated from the original sample, usually the output from `overlapEst`.
`bt`	a vector of bootstrap statistics, usually the output from `bootEst`
`conf`	a (single!) confidence interval to estimate.

Details

Let t = true value of the statistic,
t0 = estimate of t based on the original sample,
bt = bootstrap estimates.

If bootstrap sampling introduces no bias, E[mean(bt)] = t0, otherwise BS bias = mean(bt) - t0.

Assuming that the original sampling causes the same bias as the bootstrap sampling, we write: mean(bt) - t0 = t0 - t, and hence calculate a bias-corrected estimate, t1 = 2 x t0 - mean(bt).

The percentiles CI, "perc", gives quantiles of the bootstrap values, interpolated if necessary. However, in general, the bootstrap estimates are biased, so "perc" should be corrected.

"basic" is a bias-corrected version of "perc", analogous to t1: 2 x t0 - `perc`.

"norm" gives tail cutoffs for a normal distribution with mean = t1 and sd = sd(bt).

These three are equivalent to the confidence intervals returned by `boot::boot.ci`. “basic” and “norm” are appropriate if you are using the bias-corrected estimator, t1. If you use the uncorrected estimator, t0, you should use “basic0” or “norm0”:

“basic0” is `perc` - mean(bt) + t0.

“norm0” gives tail cutoffs as before, but with mean = t0 instead of t1.

The "logit" versions perform the corrections on the logit scale and then back transform. This would be appropriate for probabilities or proportions.

Value

A named matrix with 2 columns for lower and upper limits and a row for each type of estimate. Values will be NA if the bootstrap sample is too small (after removing NAs) for estimation: 40 is the minimum for a 95% confidence interval, 200 for 99% (though for stable estimates you need at least 999 bootstrap estimates, preferably 10,000).

Author(s)

Mike Meredith

See Also

`boot.ci` in package `boot`. See `kerinci` for an example.

Examples

```
# See ?kerinci
```

`bootstrap functions` *Functions to generate bootstrap estimates of overlap*

Description

`bootstrap` takes two sets of times of observations and calculates bootstrap estimates of the chosen estimator of overlap. Alternatively, bootstrap estimates can be calculated in a 2-stage process: (1) create a matrix of bootstrap samples for each data set, using `resample`; (2) pass these matrices to `bootEst` to obtain the bootstrap estimates.

A vector of bootstrap estimates can then be used to produce confidence intervals with `bootCI`.

Usage

```
bootstrap(A, B, nb, smooth=TRUE, kmax=3, adjust=NA, n.grid=128,
    type=c("Dhat1", "Dhat4", "Dhat5"), cores=1)

resample(x, nb, smooth = TRUE, kmax = 3, adjust = 1, n.grid = 512)

bootEst(Amat, Bmat, kmax = 3, adjust=c(0.8, 1, 4), n.grid = 128,
      type=c("all", "Dhat1", "Dhat4", "Dhat5"), cores=1)
```

Arguments

`A, B`	vectors of times of observations of two different species in radians, ie. scaled to $[0, 2\pi]$.
`nb`	the number of bootstrap samples required
`smooth`	if TRUE, smoothed bootstrap samples are produced.
`kmax`	maximum value of k for optimal bandwidth estimation.
`adjust`	bandwidth adjustment. If `adjust=NA` in `bootstrap`, adjust will be set to 0.8 for `type="Dhat1"`, 1 for `type="Dhat4"` and 4 for `type="Dhat5"`.
`n.grid`	number of points at which to estimate density for comparison between species; smaller values give lower precision but run faster in bootstraps.
`type`	the name of the estimator to use, or "all" to produce all three estimates. See `overlapEst` for recommendations on which to use.
`cores`	the number of cores to use for parallel processing. If NA, all but one of the available cores will used. Parallel processing may take longer than serial processing if the bootstrap runs quickly.
`x`	a numeric vector of time-of-capture data in *radians*, ie. on $[0, 2\pi]$ scale
`Amat, Bmat`	matrices of resampled data for each species produced by `resample`; see Value below.

Value

The function `bootstrap` returns a vector of bootstrap estimates. If estimation fails for a bootstrap sample, the corresponding value will be NA.

The function `resample` returns a numeric matrix with each column corresponding to a bootstrap sample. Times are in *radians*. It may return a matrix of NAs if `smooth = TRUE` and bandwidth estimation fails.

Function `bootEst` with `type = "all"` returns a numeric matrix with three columns, one for each estimator of overlap, otherwise a vector of bootstrap estimates.

Author(s)

Mike Meredith, including code by Martin Ridout.

References

Ridout & Linkie (2009) Estimating overlap of daily activity patterns from camera trap data. *Journal of Agricultural, Biological, and Environmental Statistics* 14:322-337

See Also

`bootCI`.

Examples

```
data(simulatedData)
est <- overlapEst(tigerObs, pigObs, type="Dhat4")

boots <- bootstrap(tigerObs, pigObs, 99, type="Dhat4", cores=1)
mean(boots)
hist(boots)
bootCI(est, boots)

# alternatively:
tigSim <- resample(tigerObs, 99)
dim(tigSim)
pigSim <- resample(pigObs, 99)
boots <- bootEst(tigSim, pigSim, type="Dhat4", cores=1)
mean(boots)
```

`densityFit` *Fits von Mises kernel density to time-of-day data.*

Description

Fits von Mises kernel density to time-of-day data. Intended primarily for internal use: input checking is minimal.

Usage

```
densityFit(x, grid, bw)
```

Arguments

`x`	a vector of times of observations in *radians*, ie. scaled to $[0, 2\pi]$.
`grid`	a vector of times in *radians* for which the density is required. This could be a vector of equidistant values in $[0, 2\pi]$, eg. `seq(0, 2*pi, length=128)`, or it could be any set of times for which a density is needed.
`bw`	bandwidth, the concentration parameter for the von Mises kernel: smaller values result in smoother curves.

Value

Returns a vector of densities corresponding to the times in grid.

Author(s)

C code written by Mike Meredith.

See Also

`getBandWidth` for appropriate bandwidth.

Examples

```
# Get example data:
data(simulatedData)

densityFit(tigerObs, c(0, pi/2, pi, 3*pi/2, 2*pi), 50)
# Densities at 6am and 6pm are fairly high, at midnight and midday, tiny.
# A crepuscular species!
```

`densityPlot` *Plot fitted kernel densities*

Description

Fits a kernel density function to a data set and plots it.

Usage

```
densityPlot(A, xscale = 24, xcenter = c("noon", "midnight"),
   add = FALSE, rug = FALSE, extend = 'lightgrey',
   n.grid = 128, kmax = 3, adjust = 1, ...)
```

Arguments

`A`	a vector of times of observations in *radians*, ie. scaled to $[0, 2\pi]$. It must include at least 2 unique observations to fit a kernel density.
`xscale`	The scale for the x axis: 24 (the default) produces a curve with 0 to 24 hours. NA gives a scale in radians, labelled with pi.
`xcenter`	the center of the plot on the x axis: 'noon' (default) or 'midnight'.
`add`	If TRUE, the curve will be added to the existing plot. Use the same settings for xscale and xcenter as for the original plot.
`rug`	If TRUE, the original observations will be displayed as a rug at the bottom of the plot.
`extend`	If not NULL, the plot extends 3 hours before and after the main 24-hr period, and `extend` specifies the background colour; the plot is not extended if `extend = NULL`.
`n.grid`	Number of points at which to estimate the density for plotting; 100 is usually adequate to give a smooth-looking curve.
`kmax`	maximum value of k for optimal bandwidth estimation.
`adjust`	bandwidth adjustment (scalar).
`...`	Further arguments passed to the plotting functions, such as `col`, `lty`, `lwd` or `xlab`, `ylab`, `main`.

Value

Returns invisibly a data frame with x and y coordinates which can be used for further plotting or calculations; see examples.

Author(s)

Mike Meredith

Examples

```
# Get example data:
data(simulatedData)

# Do basic plot with defaults:
densityPlot(pigObs)

# Prettier plots:
densityPlot(pigObs, extend=NULL, lwd=2)
densityPlot(pigObs, rug=TRUE, main="Simulated data", extend='gold')
densityPlot(tigerObs, add=TRUE, rug=TRUE, col='red')
legend('topleft', c("Tiger", "Pig"), lty=1, col=c('black', 'red'), bg='white')
# Add vertical dotted lines to mark sunrise (say 05:30) and sunset (18:47):
# (times must be in hours if the x-axis is labelled in hours)
abline(v=c(5.5, 18+47/60), lty=3)

# A plot centered on midnight:
densityPlot(pigObs, xcenter = "m")
# Mark sunrise/sunset; values to the left of "00:00" are negative
# so subtract 24:
abline(v=c(5.5, (18+47/60) - 24), lty=3)

# Using object returned:
densityPlot(pigObs, rug=TRUE, lwd=3)
# Don't like the rug with lwd = 3?
pigDens <- densityPlot(pigObs, rug=TRUE)
lines(pigDens, lwd=3)

# Add shading below the curve:
pigDens <- densityPlot(pigObs, extend=NULL)
polygon(pigDens, col='skyblue') # works if density at midnight = 0
tigDens <- densityPlot(tigerObs, extend=NULL)
# Add vertices at (0,0) and (24, 0)
poly <- rbind(c(0,0), tigDens, c(24,0))
polygon(poly, col='pink', border=NA)
lines(tigDens, lwd=2)

# What proportion of the density lies between 9:00 and 15:00 hrs?
wanted <- pigDens$x > 9 & pigDens$x < 15
mean(pigDens$y[wanted]) * 6  # probability mass for the 6 hr period.

# Plotting time in radians:
densityPlot(pigObs, xscale=NA, rug=TRUE)
densityPlot(tigerObs, xscale=NA, add=TRUE, rug=TRUE, col='red')
```

`Example data` *Times of 'capture' of large mammals*

Description

Times of capture of large mammals in camera traps in Kerinci Seblat National Park, Indonesia.

Usage

```
data(kerinci)
```

Format

A data frame with 1098 rows and three columns:

Zone A number indicating which of four zones the record comes from.

Sps A factor indicating which species was observed: boar (wild pig), clouded leopard, golden cat, macaque, muntjac, sambar deer, tapir, or tiger.

Time The time of the observation on a scale of 0 to 1, where 0 and 1 both correspond to midnight

Source

Ridout, M.S. and Linkie, M. (2009) Estimating overlap of daily activity patterns from camera trap data. *Journal of Agricultural, Biological and Environmental Statistics*, 14, 322-337.

https://www.kent.ac.uk/smsas/personal/msr/overlap.html

Examples

```
data(kerinci)
str(kerinci)
# Time is in days, ie. 0 to 1:
range(kerinci$Time)
# Convert to radians:
timeRad <- kerinci$Time * 2*pi

# Extract data for tiger and tapir for Zone3:
spsA <- timeRad[kerinci$Zone == 3 & kerinci$Sps == 'tiger']
spsB <- timeRad[kerinci$Zone == 3 & kerinci$Sps == 'tapir']

# Plot the data:
overlapPlot(spsA, spsB)  # Tapir are mainly nocturnal
overlapPlot(spsA, spsB, xcenter="midnight")
legend('topleft', c("Tiger", "Tapir"), lty=c(1, 2), col=c("black", "blue"), bty='n')

# Check sample sizes:
length(spsA)
length(spsB)
# If the smaller sample is less than 50, Dhat1 gives the best estimates, together with
# confidence intervals from a smoothed bootstrap with norm0 or basic0 confidence interval.

# Calculate estimates of overlap:
( Dhats <- overlapEst(spsA, spsB) )  # or just get Dhat1
( Dhat1 <- overlapEst(spsA, spsB, type="Dhat1") )

# Do 999 smoothed bootstrap values:
bs <- bootstrap(spsA, spsB, 999, type="Dhat1", cores=1)
mean(bs)
hist(bs)
abline(v=Dhat1, col='red', lwd=2)
abline(v=mean(bs), col='blue', lwd=2, lty=3)

# Get confidence intervals:
bootCI(Dhat1, bs)['norm0', ]
bootCI(Dhat1, bs)['basic0', ]
```

`getBandWidth` *Optimal bandwidth calculation*

Description

Calculates the optimal bandwidth for von Mises kernel density estimation for a given sample. Used internally by other functions in the package.

Usage

```
getBandWidth(A, kmax = 3)
```

Arguments

`A`	a vector of times of observations in *radians*, ie. scaled to $[0, 2\pi]$.
`kmax`	maximum moment to use for estimation; see Ridout & Linkie 2009.

Value

Optimal bandwidth for the sample data, or NA if estimation fails.

Author(s)

Code by Martin Ridout, error handling modified by Mike Meredith.

References

Ridout & Linkie (2009) Estimating overlap of daily activity patterns from camera trap data. *Journal of Agricultural, Biological, and Environmental Statistics* 14:322-337

Taylor (2008) Automatic bandwidth selection for circular density estimation, *Computational Statistics and Data Analysis*, 52:3493-3500.

Examples

```
data(simulatedData)
getBandWidth(tigerObs, kmax = 3)
```

`overlapEst` *Estimates of coefficient of overlapping*

Description

Calculates up to three estimates of activity pattern overlap based on times of observations for two species.

Usage

```
overlapEst(A, B, kmax = 3, adjust=c(0.8, 1, 4), n.grid = 128,
    type=c("all", "Dhat1", "Dhat4", "Dhat5"))
```

Arguments

`A`	a vector of times of observations of species A in radians, ie. scaled to $[0, 2\pi]$.
`B`	a vector of times of observations of species B in radians.
`kmax`	maximum value of k for optimal bandwidth estimation.
`adjust`	bandwidth adjustment; either a single value used for all 3 overlap estimates, or a vector of 3 different values. This corresponds to *1/c* in Ridout & Linkie 2009.
`n.grid`	number of points at which to estimate density for comparison between species; smaller values give lower precision but run faster in simulations and bootstraps.
`type`	the name of the estimator to use: `Dhat4` is recommended if both samples are larger then 50, otherwise use `Dhat1`. See Details. The default is "all" for compatibility with older versions.

Details

See `overlapTrue` for the meaning of coefficient of overlapping, Δ.

These estimators of Δ use kernel density estimates fitted to the data to approximate the true density functions *f(t)* and *g(t)*. Schmid & Schmidt (2006) propose five estimators of overlap:

Dhat1 is calculated from vectors of densities estimated at T equally-spaced times, t, between 0 and 2π:

$$\hat{\Delta}_1 = \frac{2\pi}{T}\sum_{i=1}^{T}\min\{\hat{f}(t_i), \hat{g}(t_i)\}$$

For circular distributions, Dhat2 is equivalent to Dhat1, and Dhat3 is inapplicable.

Dhat4 and Dhat5 use vectors of densities estimated at the times of the observations of the species, *x* and *y*:

$$\hat{\Delta}_4 = \frac{1}{2}\left(\frac{1}{n}\sum_{i=1}^{n}\min\left\{1, \frac{\hat{g}(x_i)}{\hat{f}(x_i)}\right\} + \frac{1}{m}\sum_{j=1}^{m}\min\left\{1, \frac{\hat{f}(y_j)}{\hat{g}(y_j)}\right\}\right)$$

$$\hat{\Delta}_5 = \frac{1}{n}\sum_{i=1}^{n} I\left\{\hat{f}(x_i) < \hat{g}(x_i)\right\} + \frac{1}{m}\sum_{j=1}^{m} I\left\{\hat{g}(y_j) \le \hat{f}(y_j)\right\}$$

where *n, m* are the sample sizes and *I* is the indicator function (1 if the condition is true, 0 otherwise).

Dhat5 simply checks which curve is higher at each point; even tiny changes in the data can result in large, discontinuous changes in Dhat5, and it can take values > 1. Don't use Dhat5.

Comparing curves at times of actual observations works well if there are enough observations of each species. Simulations show that Dhat4 is best when the smallest sample has at least 50 observations. Dhat1 compares curves at `n.grid` equally spaced points, and is best for small samples.

Value

If `type = all`, a named vector of three estimates of overlap, otherwise a single estimate. Will be NA if optimal bandwidth estimation failed.

Author(s)

Mike Meredith, based on work by Martin Ridout.

References

Ridout & Linkie (2009) Estimating overlap of daily activity patterns from camera trap data. *Journal of Agricultural, Biological, and Environmental Statistics* 14:322-337

Schmid & Schmidt (2006) Nonparametric estimation of the coefficient of overlapping - theory and empirical application, *Computational Statistics and Data Analysis*, 50:1583-1596.

See Also

`overlapTrue`.

Examples

```
# Get example data:
data(simulatedData)

# Use defaults:
overlapEst(tigerObs, pigObs)
#     Dhat1     Dhat4     Dhat5
# 0.2908618 0.2692011 0.2275000
```

```
overlapEst(tigerObs, pigObs, type="Dhat4")
#    Dhat4
#    0.2692011
```

`overlapPlot` *Plot overlapping kernel densities*

Description

Fits kernel density functions to two data sets and plots them, shading the area corresponding to the coefficient of overlap.

Usage

```
overlapPlot(A, B, xscale = 24, xcenter = c("noon", "midnight"),
   linetype = c(1, 2), linecol = c("black", "blue"), linewidth = c(1, 1),
   olapcol = "lightgrey", rug=FALSE, extend=NULL,
   n.grid = 128, kmax = 3, adjust = 1, ...)
```

Arguments

`A, B`	vectors of times of observations for species A and species B in *radians*, ie. scaled to $[0, 2\pi]$. Each must include at least 2 unique observations to fit a kernel density.
`xscale`	the scale for the x axis: 24 (the default) produces a curve with 0 to 24 hours. NA gives a scale in radians, labelled with pi.
`xcenter`	the center of the plot on the x axis: 'noon' (default) or 'midnight'.
`linetype`	a vector of length 2 giving the line type for each species. Look for `lty` in `par` for ways to specify this.
`linecol`	a vector of length 2 giving the line colour for each species. See the Color Specification section in `par` for details.
`linewidth`	a vector of length 2 giving the line width for each species.
`olapcol`	the colour to use for the shaded area. See the Color Specification section in `par` for details.
`rug`	if TRUE, the original observations will be displayed as a rug at the bottom of the plot, A below B.
`extend`	If not NULL, the plot extends 3 hours before and after the main 24-hr period, and `extend` specifies the background colour; the plot is not extended if `extend = NULL`.
`n.grid`	number of points at which to estimate the density for plotting; 100 is usually adequate to give a smooth-looking curve.
`kmax`	maximum value of k for optimal bandwidth estimation.
`adjust`	bandwidth adjustment (scalar).
`...`	Further arguments passed to the plotting functions such as `main`, `xlab`, `ylab`, `ylim`. Values for `col`, `lwd`, `lty` should be passed with `linecol`, `linewidth`, `linetype`.

Value

Returns invisibly a data frame with columns:

`x`	a vector of equally-spaced times from midnight to midnight inclusive on the scale specified by `xscale`.
`densA`	a vector of length `x` with the fitted kernel density for species A.
`densB`	a similar vector for species B.

Author(s)

Mike Meredith

See Also

`densityPlot` for plotting a single density curve.

Examples

```
# Get example data:
data(simulatedData)

# Do basic plot with defaults:
overlapPlot(pigObs, tigerObs)
# Make it prettier:
overlapPlot(tigerObs, pigObs, linet = c(1,1), linec = c("red", "blue"),
  rug=TRUE, extend="lightgreen", main="Simulated data")
legend("topleft", c("Tiger", "Pig"), lty=1, col=c("red", "blue"), bg="white")
# Add vertical dotted lines to mark sunrise (05:30) and sunset (18:47):
# (times must be in hours if the x-axis is labelled in hours)
abline(v=c(5.5, 18+47/60), lty=3)

# A plot centered on midnight:
overlapPlot(pigObs, tigerObs, xcenter = "m", rug=TRUE)
# Mark sunrise/sunset; values to the left of "00:00" are negative
# so subtract 24:
abline(v=c(5.5, (18+47/60) - 24), lty=3)
```

`overlapTrue` *Calculates the true coefficient of overlapping between two distributions.*

Description

Calculates the true coefficient of overlapping between two distributions.

Usage

```
overlapTrue(d1, d2 = NULL)
```

Arguments

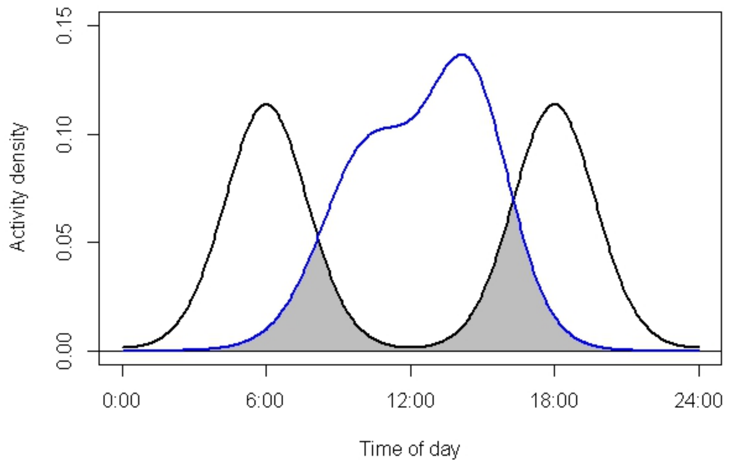
`d1`	either a vector or a 2-column matrix of densities for equidistant points from 0 to 2π; if densities for both 0 and 2π are included (and are equal), one will be ignored.
`d2`	a vector of densities as for d1; ignored if d1 is a matrix

Details

The coefficient of overlapping Δ for two probability density functions *f(x)* and *g(x)* is given by:

$$\Delta(f, g) = \int \min\{f(x), g(x)\}\,dx$$

If the two curves in the plot below represent activity patterns of two species, the coefficient of overlapping is the area under the lower of the two curves, shaded grey in the figure:



Value

The coefficient of overlap of the two distributions. The function is intended to calculate true overlap for simulated data. If the densities provided are fitted kernel densities, an estimate of overlap results.

Author(s)

Mike Meredith, based on code by Martin Ridout.

References

Ridout & Linkie (2009) Estimating overlap of daily activity patterns from camera trap data. *Journal of Agricultural, Biological, and Environmental Statistics* 14:322-337

See Also

overlapEst for various estimators of overlap.

Examples

```
data(simulatedData)

overlapTrue(tigerTrue, pigTrue)

overlapTrue(cbind(tigerTrue, pigTrue))
```

`Simulated call data` *Simulated data for bird calls influenced by sunrise*

Description

A simulated data set of bird calling activity. 80% occur around sunrise with a strong peak just before sunrise, the remainder occur around sunset. Changes in the times of sunrise and sunset through the year mean that both peaks appear to be broader than they should. The hypothetical location is near St Andrews, UK, longitude 3 degrees West, latitude 56 degrees North (CRS WGS84) and times are GMT throughout (not British Summer Time).

Usage

```
data(simCalls)
```

Format

The data set consists of a data frame with two columns:

time is a vector of 100 observations of bird calls in radians. Here $\pi/2$ corresponds to 6am and $3\pi/2$ to 6pm. The time zone is UTC (GMT).

dates is a character vector of dates in ISO format.

Source

Simulated data.

Examples

```
## See examples for the function 'sunTime'.
```

`Simulated data` *Simulated data for diel activity patterns*

Description

tigerObs and pigObs are simulated data sets with times of observation. tigerTrue and pigTrue are densities from which the simulated observations were drawn.

Usage

```
data(simulatedData)
```

Format

The data set consists of four vectors:

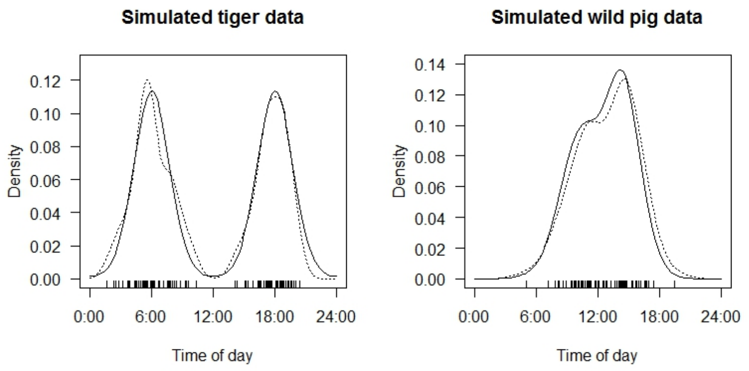
tigerObs is a vector of 100 observations of a crepuscular species in radians.

pigObs is a vector of 80 observations of a diurnal species in radians.

tigerTrue and pigTrue are vectors of densities at 128 times equidistant between 0 and 2π inclusive.

Details

The figures below show the true densities (solid line), the simulated data (rug at the foot of the plot) and a kernel density fitted to the simulated data (dotted line).



Examples

```
data(simulatedData)

xx <- seq(0, 2*pi, length=128)
plot(xx, tigerTrue, type='l') # True density from which sample was drawn
rug(tigerObs)
```

sunTime *Convert clock times to sun times*

Description

Converts a vector of clock times to "sun times", by mapping sunrise to $\pi/2$ and sunset to $3\pi/2$. Sunrise and sunset times are determined based on the dates and locations provided. See Nouvellet et al (2012) for a discussion. Requires the **suntools** package.

Usage

```
sunTime(clockTime, Dates, Coords)
```

Arguments

`clockTime`	a vector of times of observations in *radians*, ie. scaled to $[0, 2\pi]$.
`Dates`	a POSIXct object with the dates of the observations; the time zone must be set to the time zone used for 'clockTime'.
`Coords`	a SpatialPoints object with the locations of the observations, or with a single point giving a approximate location for the study area; the coordinates must be geographical coordinates, eg, WGS84, with long before lat.

Value

Returns a vector of "sun times" in *radians*, where $\pi/2$ corresponds to sunrise and $3\pi/2$ to sunset.

Author(s)

Mike Meredith.

References

Nouvellet et al (2012) Noisy clocks and silent sunrises: measurement methods of daily activity pattern. *Journal of Zoology* 286:179-184.

Examples

```
# Check that sp and suntools packages are installed
if(requireNamespace("sp") && requireNamespace("suntools")) {
  # Get example data:
  data(simCalls)
  str(simCalls)

  # Convert dates to a POSIXct object with the right time zone (GMT):
  Dates <- as.POSIXct(simCalls$dates, tz="GMT")

  # Create a SpatialPoints object with the location
  coords <- matrix(c(-3, 56), nrow=1)
  Coords <- sp::SpatialPoints(coords, proj4string=sp::CRS("+proj=longlat +datum=WGS84"))

  st <- sunTime(simCalls$time, Dates, Coords)

  exPar<-par(mfrow=c(2,1))
densityPlot(st, col='red', lwd=2, xaxt='n', main="Sun time")
axis(1, at=c(0, 6, 12, 18, 24),
labels=c("midnight", "sunrise", "noon", "sunset", "midnight"))
densityPlot(simCalls$time, lwd=2, main="Clock time")
  par(exPar)
}
```

Index