Package 'Diel.Niche'

June 23, 2023

Type Package

Title Diel niche modeling via multinomial inequalities

Description Functions to simulate data, fit and compare models, and plot results relating to estimating probabilties of diel modality.

Version 0.0.1.0

Maintainer Brian Gerber <bgerber@uri.edu>

RoxygenNote 7.2.3

Encoding UTF-8

License GPL-3

Depends R (>= 3.5)

Imports stats (>= 4.2.0),
MASS (>= 7.3-56),
utils (>= 4.2.0),
coda (>= 0.19-4),
multinomineq (>= 0.2.3),
plotly (>= 4.10.0),
sfsmisc (>= 1.1)

Suggests knitr (>= 1.39),
rmarkdown (>= 2.14),
bayesplot (>= 1.9.0),
ggplot2 (>= 3.3.6),
testthat (>= 3.0.0),
overlap (>= 0.3.4),
lubridate(>= 1.9.2),
ggthemes(>= 4.2.4),
suncalc(>= 0.5.1),
devtools(>= 2.4.5)

VignetteBuilder knitr

Config/testthat/edition 3

LazyData true

R topics documented:

`diel.data` *Data for Diel.Niche*

Description

A multi-species dataset containing detection frequencies during the twilight, daytime, and nighttime. Data come from camera-trapping along an urban gradient in Chicago, Illinois, USA.

Usage

```
diel.data
```

Format

A data frame with 232 rows and 19 variables:

scientificName Genus and species name

twilight Frequency of detections during the twilight period; sunrise and sunset.

day Frequency of detections during the daytime period.

night Frequency of detections during the nighttime period.

trap_nights The number of days all cameras were active.

nsite The number of unique camera locations.

min_date The first date of sampling in the sampling period.

max_date The last date of sampling in the sampling period.

mean_lat Avergage latitude of all camera locations during the sampling period.

mean_lon Avergage longtitude of all camera locations during the sampling period.

season Seasonal classification of the sampling period by summer, autumn, winter, and spring.

country Country where thes sampling occured.

phylum Phylum of the species.

class Class of the species.

order Order of the species.

family Damily of the species.

Project Project code name, and the city, state, and country location.

unit_type The length of the sampling period.

Common_name Common name for the species.

Activity_Literature Activity classification for the species from the literature; see Cox, D. T. C., Gardner, A. S., & Gaston, K. J. (2021). Diel niche variation in mammals associated with expanded trait space. Nature communications, 12(1), 1753.

Source

https://doi.org/10.1111/gcb.15800

`diel.fit` *Diel Modeling*

Description

Diel model hypotheses evaluation and parameter estimation. This is essentially a wrapper function for functions provided by the package multinomineq.

Usage

```
diel.fit(
  y,
  hyp.set,
  bf.fit = TRUE,
  post.fit = FALSE,
  diel.setup = NULL,
  prior = NULL,
  n.chains = 1,
  n.mcmc = 3000,
  burnin = 500,
  prints = TRUE,
  alt.optim = FALSE,
  delta = NULL
)
```

Arguments

`y`	a matrix of frequencies of animal detections. Each row is a replicate dataset. Rows should be limited when using P-s hyps (1 or 2). The matrix should always be three columns in this order: twilight, day, night. If all frequencies are 0, posteriors will be sampled from the prior according to the hypotheses.
`hyp.set`	Vector of diel hypotheses names representing hypotheses set or individual hypotheses.
`bf.fit`	If TRUE, will calculate bayes factors for the model sit. Default is TRUE.
`post.fit`	If TRUE, will fit posterior samples to all models in hyp.set. Default is FALSE.
`diel.setup`	A list of multinomial inequalities (Matrix A and vector b), representing diel hypotheses setup using the function 'diel.ineq'. If not provided, it will use the defaults of the diel.ineq function.
`prior`	Prior probabilities for models used in bayes factors. Defaults to equal among models.
`n.chains`	the number of chains to use when fitting models
`n.mcmc`	Number of mcmc iterations.
`burnin`	Burn-in number of mcmc iterations.
`prints`	Whether to print messages about model fitting.

`alt.optim` Default is FALSE. If TRUE, uses an alternative approach to derive the bayes factors. It can be more stable, but takes a bit longer.

`delta` Error tolerance of equality constraint hypotheses (e.g., AV.EQ). Does not apply to inequality constraint hypotheses. Needs to be >0 and <1, but ideally near zero. Default error tolernances to try are 0.05^(1:4)

Value

A list of outputs, including bayes factors for a model set, model bayes factor inputs, posterior samples, warning indicator, and posterior predictive checks.

A list of outputs

`bf.table` Bayes factor for hyopthesis set

`bf` A list of ordered individual model bayes factor inputs

`post.samp` A list of ordered matrices for model posterior distributions

`ms.model` The name of the most supported model detrerminded by the maximum probability of support from the bayes factors

`ppc` A list of ordered model posterior predictive check output

`ms.ppc` Posterior predictive check output from the most supported model

`post.samp.ms.model`
Posterior distributions of the most supported model

`gelm.diag` The Gelman-Rubin (Rhat) point and credible interval estimates to test converfence for parameters for all models fitted. Only provided when post.fit=TRUE and n.chains > 1.

`ms.gelm.diag` The Gelman-Rubin (Rhat) point and credible interval estimates for parameters for of the most supported model. Only provided when post.fit=TRUE, bf.fit=TRUE, and n.chains > 1.

Required libraries: multinomineq, retry, MASS

Examples

```
out=diel.fit(y=t(matrix(c(10,100,10))),hyp.set=hyp.sets("Traditional"))
```

`diel.ineq` *Inequality Setup*

Description

Multinomial model inequalities for diel hypotheses. Given a numeric set of values between 0 and 1, `diel.ineq()` will generate the inequality constraints that can be used within `diel.fit`.

Usage

```
diel.ineq(
  xi = NULL,
  e = NULL,
  e.D = NULL,
  e.N = NULL,
  e.CR = NULL,
  e.EC = NULL,
  e.AV = NULL,
  xi.t.D = NULL,
  xi.t.N = NULL,
  xi.t.CR = NULL,
  xi.t.C = NULL,
  eta.D = NULL,
  eta.N = NULL,
  eta.CR = NULL,
  eta.C = NULL,
  p.avail = NULL,
  separation = NULL
)
```

Arguments

xi	Default c(0.8, 0.1). The first element is the minimum threshold probability of singular hypotheses (e.g., Diurnal; Traditional Hypothesis set). The second element is the minimum probability for the General Hypothesis set. See details for additional information.
e	Default is 0.10. A single value of variation for probabilities. If specified, it will be applied to all hypotheses, regardless of whether individual epsilon hypotheses values are specified.
e.D	Default is 0.10. A single value of variation for the Diurnal hypothesis (Variation Hypothesis Set).
e.N	Default is 0.10. A single value of variation for the Nocturnal hypothesis (Variation Hypothesis Set).
e.CR	Default is 0.10. A single value of variation for the Crepuscular hypothesis (Variation Hypothesis Set).
e.EC	Default is 0.10. A single value of variation for the Evan Cathemeral hypothesis (Variation Hypothesis Set).
e.AV	Default is 0.10. A single value of variation for the Available Cathemeral hypothesis.
xi.t.D	Default c(0.80). A single value of the lower threshold value for the Diurnal hypothesis (Threshold Hypothesis Set).
xi.t.N	Default c(0.80). A single value of the lower threshold value for the Nocturnal hypothesis (Threshold Hypothesis Set)
xi.t.CR	Default c(0.80). A single value of the lower threshold value for the Crepuscular hypothesis (Threshold Hypothesis Set)
xi.t.C	Default c(0.2). A single value of the lower threshold value for the Cathemeral hypothesis (Threshold Hypothesis Set)

eta.D	Default c(0.90). A single value of the most probable value for the Diurnal hypothesis (Variation Hypothesis Set)
eta.N	Default c(0.90). A single value of the most probable value for the Nocturnal hypothesis (Variation Hypothesis Set)
eta.CR	Default c(0.90). A single value of the most probable value for the Crepuscular hypothesis (Variation Hypothesis Set)
eta.C	Default c(0.33). A single value of the most probable value for the Cathemeral hypothesis (Variation Hypothesis Set)
p.avail	Default c(0.166666,0.4166667). A vector of the available time in the periods of crepuscular and diurnal. Nighttime availability is found by subtraction.
separation	Default is 0. However, you can separate the hypotheses to create empty space between hypotheses probability space

Details

In the event that xi is a scalar, the second value will be calculated as `(1-xi)/2`, where `xi` is the scalar provided by the user.

The values provided here generated the requisite matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$. For additional details on how these constraints are used in a multinomial model, see Heck and Davis-Stober (2019).

Value

diel.hyp A list of diel hypotheses as multinomial inequalities.

inputs	Includes all inputted values; epsilon, xi, and p.avail.
D.th, N.th, CR.th, EC.th, C.th, D.max, N.max, CR.max, D.var, N.var, CR.var, C.var, AC.var, AV.var, Unco	Each is a list of three elements: Hypotheis Descriptive Name, A matrix, and b vector.

References

Heck, D. W., & Davis-Stober, C. P. (2019). Multinomial models with linear inequality constraints: Overview and improvements of computational methods for Bayesian inference. Journal of mathematical psychology, 91, 70-87.

Examples

```
diel.ineq()
diel.ineq(e=0.01) #To replace all epsilon values with 0.01.
#To replace the default values with a new epsilon value and a new xi value.
```

---

find.prob.hyp *Finding probabilities for a given hypothesis*

Description

Function that inputs a given hypothesis and outputs as many possible probability sets that match the diel hypothesis (i.e., satisfies the inequality constrints). Allows for inequalities, equalities, and non-linear inputs.

Usage

```
find.prob.hyp(hyp, diel.setup = NULL, fast = TRUE)
```

Arguments

hyp	hypothesis name, for example, D.max
diel.setup	A list of created by diel.ineq(). Contains matrices A and vector b, or matrices A and C, and vectors b and d.
fast	Default is TRUE, which uses p.options2 instead of p.options3. Does not apply to equality hypotheses.

Value

A matrix of probabilities that match hypothesis specified by hyp

Examples

```
find.prob.hyp(hyp ="D.max")
```

---

hyp.sets *Hypothesis Sets*

Description

Call defined hypotheses sets within Diel.Niche

Usage

```
hyp.sets(hyp.in = NULL)
```

Arguments

hyp.in	Hypothesis set code names, see details for additional information.

Details

To see all available hypothesis sets, set hyp.in = NULL to observe just the names of the available hypothesis sets or hyp.in = "list" to see the hypothesis sets as well as codes for the competing hypotheses within each set. avaialable hypothesis sets include. Currently, the available hypothesis sets include Traditional, General, Threshold, Maximizing, Variation, and Selection. When inputted into hyp.in, they must be a scalar character object and the first letter must be capitalized.

Value

Names of hypotheses for the set. If NULL, the names of all hypotheses sets are returned. If "list", all hypotheses are printed.

Examples

```
hyp.sets()
hyp.sets("Traditional")
```

`posthoc.example` *Post-hoc niche classification example data*

Description

The necessary components to use `prob.overlap` with the output from `overlap::densityPlot`.

Usage

```
posthoc.example
```

Format

A list object with three components

tiger.kde the outputted matrix from `overlap::densityPlot()` when fit to the tiger data subset from their `kerinci` data object.

- x: a column vector of length 10000 that ranges from 0 to 24, which represents time in a day
- y: a column vector of length 10000 that is the kernel density estimate for each respective data point of `x`

dawn.range a numeric vector of length 2 that is the start and end of dawn in proportional hours for this example.

dusk.range a numeric vector of length 2 that is the start and end of dusk in proportional hours for this example.

Details

When we generated this kernel density estimate we set the `n.grid` argument within `overlap::densityPlot()` to 10000. We did this because to get the probability of each diel period we must integrate under the curve, which needs a lot of samples to do accurately. To get dawn and dusk range we used the `suncalc` package with the general area of the Kerinci Seblat National Park (where the tiger data came from), used `Asia/Jakarta` as the local time zone, and chose `2008-03-12` as the date to query `c("dusk","night","dawn","nightEnd")`. As this area is close to the equator there is little variation over the year in when dawn and dusk start, so the time of year matters less in this specific case. In our case, dawn occurred between nightEnd and dawn whereas dusk occurred between dusk and night.

The help file for `posthoc.niche` includes the code used to generate all the pieces of this specific file.

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.

posthoc.niche *Posthoc diel niche*

Description

Diel model hypotheses based on posthoc probabilitiy values For each input of twilight, daytime, and nightime probabilities, the function will return the defined diel hyppthesis

Usage

```
posthoc.niche(y, hyp, diel.setup = NULL)
```

Arguments

y	a matrix of probabilities of twilight, daytime, and nightime (three columns). Each row is a replicate.
hyp	Vector of diel hypotheses names representing hypotheses set or individual hypotheses.
diel.setup	A list of multinomial inequalities (Matrix A and vector b), representing diel hypotheses setup using the function 'diel.ineq'. If not provided, it will use the defaults of the diel.ineq function.

Details

This function only uses the diel inequalities to provide a post-hoc estimate of which diel phenotype is associated to the data for a given species. As such, the output from this function is only a point estimate, and there is no uncertainty associated to this estimate. If you are interested in the uncertainty estimate (which we suggest you should be), there are two things you could do. First, if you have the data in hand use `diel.fit` instead, and select the appropriate hypothesis set via `hyp.sets`. Second, if you want to use the output from `overlap`, then you could also bootstrap the kernel density estimate and use `prob.overlap` in conjunction with `posthoc.niche` to get a distribution of classificaitons. Following this, you can calculate the proportion of times different diel phenotypes were selected.

Furthermore, in order for `posthoc.niche` to work, we integrate under the kernel density estimate. Therefore, it is very important to increase the `n.grid` argument in `overlap::densityPlot` to a sufficient number (e.g., over 10,000).

Value

data.frame of the probabilities of activity during twilight, daytime, nighttime, and matched hypothesis from the hypothesis set provided.

Examples

```
#'  data("posthoc.example")

 diel_probs <- prob.overlap(
   posthoc.example$tiger.kde,
   dawn = posthoc.example$dawn.range,
   dusk = posthoc.example$dusk.range
```

```
 )

 tiger_niche <- Diel.Niche::posthoc.niche(
  y = diel_probs,
  hyp = hyp.sets("Traditional")
)

# look at output.
tiger_niche

#       p.twi     p.day   p.night             Hypothesis
#1 0.09626836 0.6007204 0.3027826 Cathemeral Traditional

## Not run:

# NOTE:  For this example we have to make some assumptions
#  because the data in overlap we used do not have 1) spatial coordinates,
#  2) the datetime information. Likewise, we also show here how to
#  calculate the start of dawn and dusk using suncalc and lubridate.

# Load libraries
library(overlap)
library(suncalc)
library(lubridate)
library(Diel.Niche)

### Step 1. Get Kernel Density Estimate

# load data
data("kerinci")

# subset to a single species
tiger <- kerinci[kerinci$Sps == "tiger",]

# convert to Radians
tiger$Rad <- tiger$Time * 2 * pi

# get kde, you need to increase the number
#  of grid points in order to use
#  Diel.Niche::posthoc.niche()
tiger_kde <- overlap::densityPlot(
  tiger$Rad,
  extend = NULL,
  n.grid = 25000
)

### Step 2. Get ranges for dawn and dusk

# Because we don't know the date the cameras were deployed
#  we are just going to choose a single date. Since this
#  location is close to the equator there should not
#  be much temporal variation anyways.

my_times <- suncalc::getSunlightTimes(
  date = as.Date("2008-03-12"),
  lat = -2.41,
  lon =  101.4836,
```

```
  keep = c("dusk", "night", "dawn","nightEnd"),
  tz = "Asia/Jakarta"
)

# strip the month / year from these times
my_times$dusk <- lubridate::hms(
  format(
    my_times$dusk,
    "%H:%M:%S"
  )
)

my_times$night <- lubridate::hms(
  format(
    my_times$night,
    "%H:%M:%S"
  )
)
my_times$dawn <- lubridate::hms(
  format(
    my_times$dawn,
    "%H:%M:%S"
  )
)
my_times$nightEnd <- lubridate::hms(
  format(
    my_times$nightEnd,
    "%H:%M:%S"
  )
)

# calculate the time when dawn & dusk starts and stops
my_dawn <- c(my_times$nightEnd, my_times$dawn)

my_dusk <- c(my_times$dusk, my_times$night)

# convert these to fractional hours, given that there
#  are 86400 seconds in a day.
my_dawn <- (as.numeric(my_dawn) / 86400) * 24
my_dusk <- (as.numeric(my_dusk) / 86400) * 24

### Step 3. Combining the kernel density estimate with overlap.

diel_probs <- Diel.Niche::prob.overlap(
  densityplot = tiger_kde,
  dawn = my_dawn,
  dusk = my_dusk
)

### Step 4. Classify diel niche posthoc

tiger_niche <- Diel.Niche::posthoc.niche(
  y = diel_probs,
  hyp = hyp.sets("Traditional")
)

# look at output.
```

```
tiger_niche

#       p.twi     p.day   p.night             Hypothesis
#1 0.09626836 0.6007204 0.3027826 Cathemeral Traditional

## End(Not run)
```

`prob.overlap` *Kernel Overlap density integration*

Description

Integrate kernel density to derive probability of twilight, daytime, nighttime

Usage

```
prob.overlap(densityplot, dawn = c(6, 7), dusk = c(17, 18))
```

Arguments

`densityplot`	a densityPlot object from package overlap. See details for additional information.
`dawn`	beginning and end numeric (0-24) times for dawn. This is in proportional hours such that 12.5 would be 12:30. See details for additional information.
`dusk`	beginning and end numeric (0-24) times for dusk. This is in proportional hours such that 12.5 would be 12:30. See details for additional information.

Details

When creating the density plot, it is important to increase the `n.grid` argument, because to this function integrates the area under the curve to compute the associated probabilities. We suggest to start setting `n.grid` in `overlap::densityPlot` to at least 10000, but you may be able to do less. Essentially, if the outputted sum is within about a thousandth from one (e.g., 0.999) then the output from this function can be used in `posthoc.niche`.

To compute proportional hours, you will need to start with a time object. Changing that time to a numeric should convert it to seconds from midnight. Given that there are 86,400 seconds in a day, divide by that number and multiply by 24 to create the proportional hours. See examples for some code on how to do this.

@examples data("posthoc.example")

diel_probs <- prob.overlap(posthoc.example$tiger.kde, dawn = posthoc.example$dawn.range, dusk = posthoc.example$dusk.range)

Value

A matrix of three probabilities.

`sim.diel` *Simulate*

Description

Simulate diel data

Usage

```
sim.diel(
  n.sim = 1,
  reps = 1,
  n.sample = 100,
  hyp,
  diel.setup = NULL,
  sd.error = 0,
  fast = TRUE,
  return.probs = FALSE
)
```

Arguments

`n.sim`	The number of simulated datasets (integer)
`reps`	The number of sets (integer) of probabilities to use when simulating crepuscular, daytime, and nocturnal frequencies
`n.sample`	The number of total samples (integer) for a given simulation
`hyp`	The hypothesis code to simulate data from
`diel.setup`	Multinomial inequalities for hypotheses setup using function 'diel.ineq'.
`sd.error`	Normal distribution standard deviation to simulate error to add to the probabilities on the logit-scale. Default is 0
`fast`	Default is TRUE, which uses a less precise probability interval sequence (0.005 vs 0.001). Does not apply to equality hyps.
`return.probs`	Default is FALSE. If TRUE, returns probabilities from the hypothesis.

Value

A list of outputs

`y`	Matrix of simulated datasets
`p`	Probabilities used to simulate the data

Examples

```
sim.diel(n.sim=1,reps=1,n.sample=100,hyp="D.th")
```

`triplot` *Plot Diel Hypothesis or Hypothesis Set Along with Posterior Samples*

Description

Plots the diel niche space and posterior disribution of a fitted model.

Usage

```
triplot(
  fit = NULL,
  hyp = NULL,
  diel.setup = NULL,
  posteriors = NULL,
  more.points = FALSE,
  x.scene = 2.5,
  y.scene = 1,
  z.scene = 0.3,
  axis.size = 16,
  axis.lab.size = 18,
  legend.lab.size = 15
)
```

Arguments

`fit`	a list object output from the function 'diel.fit'.
`hyp`	a vector of hypothesis code names (characters)
`diel.setup`	If NULL uses the default diel.ineq function. If a fit object is provided it will come from this object. Otherwise, a list of multinomial inequalities (Matrix A and vector b) representing diel hypotheses and the function needed for model fitting.
`posteriors`	Posterior samples output from the function 'diel.fit'.
`more.points`	To use more points for hyps in plotting. Default is FALSE.
`x.scene`	3d graphical parameter
`y.scene`	3d graphical parameter
`z.scene`	3d graphical parameter
`axis.size`	3d graphical parameter
`axis.lab.size`	3d graphical parameter
`legend.lab.size`	3d graphical parameter

Value

A plotly 3d plot

Examples

```
out=diel.fit(y=cbind(11,87,2),hyp="D",post.fit=TRUE)
triplot(out)
```

what.hyp *Hypothesis Codes*

Description

Call defined hypotheses sets

Usage

```
what.hyp(hyp.in = NULL)
```

Arguments

hyp.in	hypothesis code name or NULL.

Value

Full name of hypothesis if code name is provided. If NULL a general description of hypotheses is provided.

Examples

```
what.hyp()
what.hyp("D.th")
```

Index