Package 'hab'

December 26, 2014

Version 1.19.3

Date 2014-12-26

Title Habitat and movement functions

Description A set of functions related to habitat selection and movement analyses. Also includes a few patches for functions from the adehabitatXY package family.

Author Mathieu Basille

Maintainer Mathieu Basille <basille@ase-research.org>

Depends adehabitatMA, adehabitatLT, adehabitatHR

Suggests survival, tkrplot

License GPL (>= 3)

URL http://ase-research.org/basille/hab

R topics documented:

`acf.test` *Test of autocorrelation in SSFs*

Description

Test of autocorrelation and partial autocorrelation in SSF models, based on the estimation of autocorrelation functions (ACF).

Usage

```
acf.test(residuals, id, type = c("correlation", "covariance", "partial"),
  ci = 0.95)
```

Arguments

`residuals`	A vector of residuals on which to compute the ACF. The residuals must be sorted chronologically.
`id`	A vector of corresponding animal IDs.
`type`	A character string giving the type of ACF to be computed. Allowed values are `correlation` (default), `covariance` or `partial`.
`ci`	A numeric giving the confidence interval to be used to test the ACF.

Value

A list, with the following parameters:

- `acfk`: a list with the individual autocorrelation functions
- `threshold`: a vector with individual thresholds
- `lag`: a vector with the resulting individual lags

Author(s)

Mathieu Basille <basille@ase-research.org>

See Also

See `acf` for further details on the ACF.

as.ltraj *Working with Trajectories in 2D Space: the Class ltraj*

Description

Faster version of as.ltraj, which can be up to 5-10 times faster.

Usage

```
as.ltraj(xy, date = NULL, id, burst = id, typeII = TRUE,
  slsp = c("remove", "missing"), infolocs = data.frame(pkey = paste(id,
  date, sep = ".")))
```

Author(s)

Modified by Mathieu Basille <basille@ase-research.org>

See Also

See ld for further details on the function and all available arguments.

Examples

```
data(puechabonsp)
locs <- puechabonsp$relocs
xy <- coordinates(locs)
df <- as.data.frame(locs)
id <- df[,1]
da <- as.POSIXct(strptime(as.character(df$Date), "%y%m%d"))
ltr1 <- adehabitatLT:::as.ltraj(xy, da, id = id)
ltr2 <- as.ltraj(xy, da, id = id)
all.equal(ltr1, ltr2)
```

dl *Quick Conversion of Objects of Class ltraj from and to Dataframes*

Description

Faster versions of ld and dl.

Usage

```
dl(x, strict = TRUE)

ld(ltraj, strict = TRUE)
```

Arguments

strict	Logical. Whether to use the regular ld or dl functions, which enforce more verifications.

Details

In ld, strict = FALSE can be up to 10 times faster, but assumes that the ltraj is well structured (i.e. not modified by the user). In dl, strict = FALSE can be up to 20 times faster, but assumes that the trajectory parameters in the data frame (x/y increments, angles, etc.) are still valid (e.g. no locations have been removed).

Author(s)

Modified by Mathieu Basille <basille@ase-research.org>

See Also

See ld for further details on the function and all available arguments.

Examples

```
data(puechcirc)
puechcirc ## class ltraj

## ld
df1 <- adehabitatLT:::ld(puechcirc)
df2 <- ld(puechcirc, strict = FALSE)
all.equal(df1, df2)
## Note a difference in row names:
attr(df1, "row.names")
attr(df2, "row.names")

## dl
all.equal(dl(df2), adehabitatLT:::dl(df2))
dl(df2, strict = FALSE)
## Comparison regarding 'strict'
all.equal(dl(df2), dl(df2, strict = FALSE))
## Differences in row.names (numeric in regular 'dl', characters using
## 'strict = FALSE') + NAs in R2n (for a reason, 'puechcirc[[2]]'
## starts by a sequence of missing values, but has several 'R2n'
## values. As a result, 'strict = FALSE' keeps the 'R2n' values)
```

---

hab *Habitat selection and movement analyses*

Description

A set of functions related to habitat selection and movement analyses. Also includes a few patches for functions from the adehabitatXY package family. For a list of documented functions, use library(help = "hab")

Author(s)

Mathieu Basille <basille@ase-research.org>

`infolocs` *Infolocs of an Object of Class ltraj*

Description

Modified version of `infolocs` that returns `NULL` if `infolocs` exists, but `which` is not a colum of it.

Usage

```
infolocs(ltraj, which, perani = FALSE, simplify = FALSE)
```

Arguments

`perani`	Logical. Should the function return one element per burst (`FALSE`, default) or per animal (`TRUE`).
`simplify`	Logical. If a single variable is requested, should the function return 1-column data frames (`FALSE`, default) or vector (`TRUE`).

Author(s)

Modified by Mathieu Basille <basille@ase-research.org>

See Also

See `infolocs` for further details on the function and all available arguments.

Examples

```
## Load puechcirc data, and add some random infolocs:
data(puechcirc)
info <- list(data.frame(A = rnorm(80), B = runif(80), C = rpois(80,
    1)), data.frame(A = rnorm(69, 10), B = runif(69, 10, 11),
    C = rpois(69, 10)), data.frame(A = rnorm(66, -10), B = runif(66,
    -10, -9), C = rpois(66, 100)))
## Watch the row names:
info <- mapply(function(x, y) {
    row.names(x) <- row.names(y)
    return(x)
}, info, puechcirc, SIMPLIFY = FALSE)
infolocs(puechcirc) <- info

## Try to retrieve the column `toto`:
infolocs(puechcirc, "toto")
```

kerneloverlap *Spatial Interaction between Animals Monitored Using Radio-Tracking*

Description

Modified version of kerneloverlap, which now allows to select several methods at once, and works on SpatialPointsDataFrame or estUDm in the same function.

Usage

```
kerneloverlap(x, method = c("HR", "PHR", "VI", "BA", "UDOI", "HD"),
  percent = 95, conditional = FALSE, ...)

## S3 method for class 'kerneloverlap'
summary(object, ...)
```

Arguments

x	an object of class SpatialPointsDataFrame containing only one column (which is a factor indicating the identity associated to the relocations) or an object of class estUDm containing several home-ranges for which the overlap is to be calculated
method	the desired method(s)for the estimation of overlap (several methods can be chosen, defaults to all at once)
object	An array or matrix of class kerneloverlap

Value

An array of class kerneloverlap with as many matrices as chosen methods (if only one method is chosen, a matrix of class kerneloverlap); each matrix gives the value of indices of overlap for all pairs of animals.

Author(s)

Modified by Mathieu Basille <basille@ase-research.org>

See Also

See kerneloverlap for further details on the function and all available arguments.

Examples

```
## Prepare the data:
data(puechabonsp)
loc <- puechabonsp$relocs
elev <- puechabonsp$map

## Kernel overlap using various approaches:
```

```
kerneloverlap(puechabonsp$relocs[,1], grid = 200, meth = "HR", conditional = TRUE)
kerneloverlap(puechabonsp$relocs[,1], grid = elev, meth = c("HR", "VI"), conditional = TRUE)
kerneloverlap(puechabonsp$relocs[,1], grid = 200, conditional = TRUE)

## Summarizing the results:
summary(kerneloverlap(puechabonsp$relocs[,1], grid = 200))
summary(kerneloverlap(puechabonsp$relocs[,1], grid = 200, percent = 50))
```

`kernelUD` *Estimation of Kernel Home-Range*

Description

Modified version of `kernelUD`, which silences the use of `same4all` when working on a `SpatialPixels` as a `grid`.

Usage

```
kernelUD(xy, h = "href", grid = 60, same4all = FALSE, hlim = c(0.1,
  1.5), kern = c("bivnorm", "epa"), extent = 0.5, boundary = NULL)
```

Author(s)

Modified by Mathieu Basille <basille@ase-research.org>

See Also

See `kernelUD` for further details on the function and all available arguments.

Examples

```
## Prepare the data:
data(puechabonsp)
loc <- puechabonsp$relocs
elev <- puechabonsp$map

## Compute the kernel estimation
ker1 <- kernelUD(puechabonsp$relocs[,1], grid = elev, same4all = TRUE)

## Summarizing the overlaps
summary(kerneloverlap(ker1, conditional = TRUE))
```

kfold *kfold*

Description

Cross-validation for regression models.

Usage

```
kfold(mod, k = 5, nrepet = 100, jitter = FALSE, reproducible = TRUE,
  details = FALSE)

## S3 method for class 'coxph'
kfold(mod, k = 5, nrepet = 100, jitter = FALSE,
  reproducible = TRUE, details = FALSE)
```

Arguments

mod	A fitted model for which there exists a kfold method (currently coxph and clogit models).
k	The number of equal size subsamples of the partition.
nrepet	The number of repetitions.
jitter	Logical, whether to add some random noise to the predictions (useful when the model is fitted on categorical variables, which can produces error in the ranking process).
reproducible	Logical, whether to use a fixed seed for each repetition.
details	Logical, whether to return details of each repetition (useful for debugging).

Details

Note: needs complete names in the coxph/clogit call, and not 'cos(var)' or 'I(var*10)', except for 'strata()' and 'cluster()'.

Also needs complete case in the model (e.g. it fails if there is at least one NA in the observed steps for any variable of the model). Returns an error if some stratas have no case.

Value

A data frame with the correlations (cor) and the type of value (type).

Author(s)

Mathieu Basille <basille@ase-research.org>, with the help of Terry Therneau and Guillaume Bastille-Rousseau

`lincircor` *Linear-circular correlation*

Description

Compute Linear-circular correlation, and test it.

Usage

```
lincircor(xl, theta)

lincircor.test(xl, theta)
```

Arguments

`xl`	A numeric, providing the linear variable.
`theta`	A numeric, providing the circular variable, in radians (`rad = deg * pi / 180`).

Value

`lincircor` returns a numeric providing the linear-circular correlation. `lincircor.test` returns a list with class `"htest"` containing the following components:

- `statistic`: the value of the test statistic
- `parameter`: the degrees of freedom of the test statistic, which follows a F distribution if `x`, and `theta` are independent and `x` is normally distributed
- `p.value`: the p-value of the test
- `estimate`: the estimated measure of linear-circular correlation
- `null.value`: the value of the correlation measure under the null hypothesis, always `0`
- `alternative`: a character string describing the alternative hypothesis
- `method`: a character string indicating how the correlation was measured
- `data.name`: a character string giving the names of the data

Author(s)

Mathieu Basille <basille@ase-research.org>

References

Mardia, K. V. & Jupp, P. E. (2000) Directional statistics. Wiley, 429 pp.

Examples

```
## From Mardia & Jupp (2000), p.246
xl <- c(28, 85.2, 80.5, 4.7, 45.9, 12.7, 72.5, 56.6, 31.5, 112,
    20, 72.5, 16, 45.9, 32.6, 56.6, 52.6, 91.8, 55.2)
theta <- c(327, 91, 88, 305, 344, 270, 67, 21, 281, 8, 204, 86,
    333, 18, 57, 6, 11, 27, 84)
## 'xl' is not in radians (but the test is computed anyway)
lincircor(xl, theta)
## Conversion to radians
lincircor(xl, theta*pi/180)
## Test
lincircor.test(xl, theta*pi/180)
```

---

ltraj2sldf *Conversion of the class* ltraj *to the package* sp

---

Description

ltraj2spdf: Add a proj4string parameter, and keeps id and burst. ltraj2sldf: Add a proj4string parameter, and by is now a character, which accepts "id", "burst" (default) and "step". Warning: the conversion to SLDF using by = "step" is significantly longer, of a factor of ~50, than the other two (it takes about 10 sec for each 10,000 steps on a relatively recent computer).

Usage

```
ltraj2sldf(ltr, by = c("burst", "id", "step"), strict = TRUE,
  proj4string = CRS(as.character(NA)))

ltraj2spdf(ltr, strict = TRUE, proj4string = CRS(as.character(NA)))
```

Arguments

by	Sets the level of aggregation: if id, one object of class Lines corresponds to one animal; if burst (default), it corresponds to one burst; if step, it corresponds to one step.
strict	Logical (TRUE by default). See ld. For ltraj2sldf, only applies to by = "step".
proj4string	A valid proj4 string (see proj4string for more details).

Author(s)

Modified by Mathieu Basille <basille@ase-research.org>

See Also

See ltraj2spdf for further details on the function and all available arguments.

Examples

```
## Load puechcirc data, and add some random infolocs:
data(puechcirc)
info <- list(data.frame(A = rnorm(80), B = runif(80), C = rpois(80,
    1)), data.frame(A = rnorm(69, 10), B = runif(69, 10, 11),
    C = rpois(69, 10)), data.frame(A = rnorm(66, -10), B = runif(66,
    -10, -9), C = rpois(66, 100)))
## Watch the row names:
info <- mapply(function(x, y) {
    row.names(x) <- row.names(y)
    return(x)
}, info, puechcirc, SIMPLIFY = FALSE)
infolocs(puechcirc) <- info

## Conversion to SPDF:
summary(adehabitatLT:::ltraj2spdf(puechcirc))
summary(ltraj2spdf(puechcirc, strict = FALSE))
summary(ltraj2spdf(puechcirc, strict = FALSE, proj4string = CRS("+init=epsg:27572")))

## Conversion to SLDF:
summary(adehabitatLT:::ltraj2sldf(puechcirc, byid = TRUE))
summary(ltraj2sldf(puechcirc, by = "id"))
summary(ltraj2sldf(puechcirc, proj4string = CRS("+init=epsg:27572")))
summary(ltraj2sldf(puechcirc, strict = FALSE, by = "step",
    proj4string = CRS("+init=epsg:27572")))
```

---

makeCluster *Create independent clusters*

Description

Create independant clusters from a sequence of (observed + random) steps.

Usage

```
makeCluster(seq, strata, case, ndrop, nclust = NULL, nstep = NULL)
```

Arguments

seq	A numeric, indicating the continuous sequences of steps. See Details.
strata	A numeric, indicating the strata of each observed and random steps.
case	A numeric, with 1 for observed steps and 0 for random steps.
ndrop	The number of steps to drop between each cluster. They will be NAs in the resulting vector.
nclust	The ideal number of clusters to obtain. See Details.
nstep	The number of steps consisting in a single cluster. See Details.

Details

There must be one and only one case per strata. In addition, the sequences must be orderd in ascending order (there is no check on this), and the case and strata should correspond to that order.

It is only necessary to provide `nclust` to get at least that many clusters (if possible). The argument `nloc` can be set instead, if one wants exactly a number of successive steps in a cluster (much faster computation).

Value

A vector of the same length as `seq`, `strata` and `case`, with the cluster number for each step, and three attributes giving the number of clusters (`nclust`), the number of successive steps kept per cluster (`nstep`) and the number of successive steps dropped between each cluster (`ndrop`).

Author(s)

Mathieu Basille <basille@ase-research.org>

Examples

```
## case (1 observed + 9 random)
case <- rep(rep(1:0, c(1, 9)), 100)
## 100 stratas of 10 steps (1 observed + 9 random)
strata <- rep(1:100, each = 10)
## 5 sequences of 20*10 steps
seq <- rep(1:5, each = 200)
head(data.frame(case, strata, seq), 22)
## Make at least 15 clusters by dropping 2 steps between each of them
makeCluster(seq, strata, case, ndrop = 2, nclust = 15)
## Same result with 'nstep = 7' direcly
makeCluster(seq, strata, case, ndrop = 2, nstep = 7)
```

---

makeSeq *Define sequences of a trajectory*

Description

Define sequences of a trajectory based on the chronology of steps, allowing for gaps.

Usage

```
makeSeq(date, id = NULL, gap = 1, units = c("hours", "mins", "secs",
  "days", "weeks"))

summarySeq(seq, id = NULL)

## S3 method for class 'summarySeq'
print(x, ...)
```

Arguments

date	A POSIXt object, chronologically ordered (by individual if id is provided).
id	A vector giving the id of each step (must be the same length as date). If not provided, the function assumes all steps come from a single individual.
gap	The maximum time interval between two successives steps before starting a new sequence.
units	The unit of gap (default is hour).
seq	An vector giving a sequence in a trajectory, as given by makeSeq, or as a series of integers.
x	An object of class summarySeq.

Value

For makeSeq, a numeric vector, with the sequence number for each date, in the form of an integer series for each individual.

For summarySeq, an object of class summarySeq.

Author(s)

Mathieu Basille <basille@ase-research.org>

Examples

```
## Dummy data:
(steps <- data.frame(date = Sys.time() + c(1:10, 12, 14, 20:25) *
    3600, id = c(rep("toto", 4), rep("tata", 9), rep("toto",
    5))))

## 1-hour gap, all steps considered from a single individual:
makeSeq(steps$date)
## 2-hour gap, all steps considered from a single individual:
makeSeq(steps$date, gap = 2)
## 1-hour gap, individual id considered:
steps$seq <- makeSeq(steps$date, steps$id)
steps

## Note that `seq` uses a series starting from 1 for each individual. Uses
## something like this to have unique seq IDs among individuals:
steps$seqid <- paste(steps$id, steps$seq, sep = "-")
steps

## Summary of the sequence, individual id considered:
summarySeq(steps$seq, steps$id)
```

modWeights *Model weights using Information Theory*

Description

Compute model weights using Information Theory criteria.

Usage

```
modWeights(..., criterion = QIC, names = TRUE)
```

Arguments

...	All fitted model objects.
criterion	The function to be used as the Information Theory criterion. Only tested with QIC (default).
names	Logical, whether to assign names to the result.

Value

A numeric vector with the corresponding weights.

Author(s)

Mathieu Basille <basille@ase-research.org>

na.omit.ltraj *Handle Missing Values in Objects of Class 'ltraj'*

Description

na.omit removes missing locations from a ltraj object.

Usage

```
## S3 method for class 'ltraj'
na.omit(object, rec = TRUE, complete.steps = FALSE, ...)
```

Arguments

object	An object of class ltraj.
rec	Logical, whether to recompute descriptive parameters of the trajectory (in particular dx, dy, and angles). Use FALSE with care.
complete.steps	Logical, whether to keep only complete steps, i.e. steps characterized by start and end points, and turning angle (i.e. relative angle to the previous step).
...	Further arguments not used.

Value

A ltraj object, without missing locations.

Author(s)

Mathieu Basille <basille@ase-research.org>

Examples

```
data(puechcirc)
puechcirc
na.omit(puechcirc)
## Note the direct call to hab:::
hab:::na.omit.ltraj(puechcirc, rec = FALSE, complete.steps = TRUE)
```

---

plot.ltraj *Graphical Display of an Object of Class "ltraj"*

Description

New arguments to allow a better control by the user, plus a computation of xlim/ylim that ensures that the bounding box of the maps are the same for each burst (see parameter center). Also enables the plot of a SpatialPoints(DataFrame) in the background.

Usage

```
plot.ltraj(x, id = unique(adehabitatLT::id(x)),
  burst = adehabitatLT::burst(x), by = c("burst", "id", "none"),
  na.rm = TRUE, spixdf = NULL, spoldf = NULL, spotdf = NULL,
  xlim = NULL, ylim = NULL, center = FALSE, addpoints = TRUE,
  addlines = TRUE, box = FALSE, final = TRUE, mfrow, ppar = list(pch =
  21, col = "black", bg = "white"), lpar = list(), spixdfpar = list(col =
  gray((240:1)/256)), spoldfpar = list(col = "green"), spotdfpar = list(pch
  = 3, col = "darkgreen"), ...)
```

Arguments

by	Character, replaces perani. Either "burst" (identical to perani = FALSE), "id" (identical to perani = TRUE), or "none" to plot all bursts at once.
na.rm	Logical, whether to remove missing locations.
spotdf	An object of class SpatialPoints.
center	Logical, whether to center each plot around the current burst (default = FALSE).
box	Logical, whether to add a box after plotting each individual plot (useful when a map overlaps the plot borders, default = FALSE).

mfrow	A vector of the form c(nr, nc), which allows the user to define the numbers of rows (nr) and columns (nc) in the device (the default uses n2mfrow(length(id)) if length(id) <= 12, and mfrow = c(3, 4) otherwise).
ppar	A list of arguments that allows the user to modify point display, using any argument available to points. Default is list(pch = 21, col = "black", bg = "white"). See Details.
lpar	A list of arguments that allows the user to modify line display, using any argument available to lines. Default is list(), i.e. an empty list. See Details.
spixdfpar	A list of arguments that allows the user to modify SpatialPixelsDataFrame display, using any argument available to image. Default is list(col = gray((240:1)/256)).
spoldfpar	A list of arguments that allows the user to modify SpatialPolygons display, using any argument available to plot. Default is list(col = "green").
spotdfpar	A list of arguments that allows the user to modify SpatialPoints display, using any argument available to plot. Default is list(pch = 3, col = "darkgreen").

Details

It is possible to use point and line parameters globally for every trajectory displayed. In this case, ppar and lpar need just be a list of graphical parameters, such as list(pch = 21, col = "black", bg = "white"). It is also possible to use parameters for single bursts or individual, using atomic vectors with the same numbers of named elements than bursts/id, such as list(col = c(b1 = "blue", b2 = "red")). Finally, it is also possible to use parameters for single steps, using as graphical parameter a list of vectors of length equal to each trajectory. Such information can be based on infolocs, see Examples.

Author(s)

Modified by Mathieu Basille <basille@ase-research.org>

See Also

See plot.ltraj for further details on the function and all available arguments.

Examples

```
data(puechcirc)

## Point and line parameters
plot(puechcirc)
plot(puechcirc, ppar = list(pch = 2, cex = .5), lpar = list(lty = 2,
    col = grey(.5)))
## By burst
plot(puechcirc, ppar = list(col = c(CH930824 = "blue", CH930827 = "green",
    JE930827 = "red"), pch = 20), lpar = list(col = c(CH930824 = "blue",
    CH930827 = "green", JE930827 = "red")))

## Parameter 'by', 'mfrow' and 'id'
plot(puechcirc, by = "id")
plot(puechcirc, by = "id", mfrow = c(1, 2))
```

```
plot(puechcirc, id = "JE93")
plot(puechcirc, by = "none", ppar = list(col = c(CH930824 = "blue",
    CH930827 = "green", JE930827 = "red"), pch = 20), lpar = list(col =
    c(CH930824 = "blue", CH930827 = "green", JE930827 = "red")))

## Using parameters for single steps
info <- list(data.frame(col = sample(c("red",
         "grey"), 80, rep = TRUE), stringsAsFactors = FALSE),
         data.frame(col = sample(c("blue", "darkred"),
             69, rep = TRUE), stringsAsFactors = FALSE),
         data.frame(col = sample(c("darkgreen", "purple"),
             66, rep = TRUE), stringsAsFactors = FALSE))
info <- mapply(function(x, y) {
    row.names(x) <- row.names(y)
    return(x)
}, info, puechcirc, SIMPLIFY = FALSE)
infolocs(puechcirc) <- info

## By burst (default)
plot(puechcirc, ppar = list(pch = 19, col = infolocs(puechcirc,
    "col", simplify = TRUE)), lpar = list(col = infolocs(puechcirc,
    "col", simplify = TRUE)), na.rm = FALSE)

## By animal
plot(puechcirc, by = "id", ppar = list(pch = 19, col = infolocs(puechcirc,
    "col", simplify = TRUE, perani = TRUE)), lpar = list(col = infolocs(puechcirc,
    "col", simplify = TRUE, perani = TRUE)), na.rm = FALSE)

## Using a SpatialPixelsDataFrame
data(puechabonsp)

plot(puechcirc, spixdf = puechabonsp$map[,1])
plot(puechcirc, spixdf = puechabonsp$map[,1],
    ppar = list(pch = 2, cex = .5), lpar = list(lty = 2, col = "white"),
    spixdfpar = list(col = gray((1:240)/256)))

## Using a SpatialPolygonsDataFrame
cont <- getcontour(puechabonsp$map[,1])
plot(puechcirc, spoldf = cont)
plot(puechcirc, spoldf = cont, ppar = list(pch = 2, cex = .5),
    lpar = list(lty = 2, col = grey(.5)), spoldfpar = list(col = "cornsilk",
        border = grey(.5)))
```

---

plotltr *Changes in Traject Parameters Over Time*

Description

Four new arguments to allow a better control by the user (note that arguments pch and cex have been removed and are now used in ppar).

Usage

```
plotltr(x, which = "dist", perani = FALSE, addlines = TRUE, mfrow,
  ppar = list(pch = 16, cex = 0.7), lpar = list(), ...)
```

Arguments

perani	logical. If FALSE (default), one plot is drawn for each value of burst. If TRUE, one plot is drawn for each value of id, and the several bursts are superposed on the same plot for a given animal.
mfrow	A vector of the form c(nr, nc), which allows the user to define the numbers of rows (nr) and columns (nc) in the device (the default uses n2mfrow(length(id)) if length(id) <= 12, and mfrow = c(3, 4) otherwhise).
ppar	A list of arguments that allows the user to modify point display, using any argument available to points. Default is list(pch = 21, col = "black", bg = "white").
lpar	A list of arguments that allows the user to modify line display, using any argument available to lines. Default is list(), i.e. an empty list.

Details

Also corrects a bug if the burst contains infolocs, in which case other attributes (notably burst and id) were lost.

Author(s)

Modified by Mathieu Basille <basille@ase-research.org>

Modified by Mathieu Basille <basille@ase-research.org>

See Also

See plotltr for further details on the function and all available arguments.

Examples

```
data(puechcirc)

adehabitatLT:::plotltr(puechcirc, "cos(rel.angle)")
plotltr(puechcirc, "cos(rel.angle)")
## Not run:
plotltr(puechcirc, "cos(rel.angle)", ppar = list(pch = 2, cex = 2),
    lpar = list(lty = 2, lwd = 2), mfrow = c(2, 1))
## End(Not run)

adehabitatLT:::plotltr(puechcirc, "dist")
plotltr(puechcirc, "dist")
plotltr(puechcirc, "dist", ppar = list(pch = 3, col = "blue"),
    lpar = list(lty = 3, col = "red"), perani = TRUE)

adehabitatLT:::plotltr(puechcirc, "dx")
```

```
plotltr(puechcirc, "dx")
plotltr(puechcirc, "dx", ppar = list(col = rep(1:8, each = 6)),
    addlines = FALSE)
```

---

plotNAltraj *Highlighting the Patterns in Missing Values in Trajects*

Description

Five new arguments to allow a better control by the user.

Usage

```
plotNAltraj(x, perani = FALSE, addlines = TRUE, mfrow, ppar = list(pch =
  16, cex = 0.3), lpar = list(), ...)
```

Arguments

perani	logical. If FALSE (default), one plot is drawn for each value of burst. If TRUE, one plot is drawn for each value of id, and the several bursts are superposed on the same plot for a given animal.
addlines	Logical. Indicates whether lines should be added to the plot.
mfrow	A vector of the form c(nr, nc), which allows the user to define the numbers of rows (nr) and columns (nc) in the device (the default uses n2mfrow(length(id)) if length(id) <= 12, and mfrow = c(3, 4) otherwhise).
ppar	A list of arguments that allows the user to modify point display, using any argument available to points. Default is list(pch = 21, col = "black", bg = "white").
lpar	A list of arguments that allows the user to modify line display, using any argument available to lines. Default is list(), i.e. an empty list.

Author(s)

Modified by Mathieu Basille <basille@ase-research.org>

See Also

See plotNAltraj for further details on the function and all available arguments.

Examples

```
data(puechcirc)
adehabitatLT:::plotNAltraj(puechcirc)
plotNAltraj(puechcirc, perani = TRUE, addlines = FALSE, mfrow = c(1,
    2), ppar = list(pch = 15, cex = 0.5))
```

QIC *QIC: Quasi-likelihood under Independence Criterion*

Description

Generic function calculating the Quasi-likelihood under Independence Criterion for one or several fitted model objects.

Usage

```
QIC(mod, ...)

## S3 method for class 'coxph'
QIC(mod, ..., details = FALSE)
```

Arguments

mod	A coxph or clogit model.
...	Optionally more fitted model objects.
details	Logical, whether to provide detailed outputs (turned automatically to TRUE when several models are fitted).

Value

If details = FALSE, simply returns the QIC. If details = TRUE, returns a data frame presenting the QIC, the quasi-likelihood, the number of observations, the number of events, the number of paramaters and the trace. If several models are provided, two additional columns present the delta QIC and the model weights.

Author(s)

Mathieu Basille <basille@ase-research.org> and Thierry Duchesne

rdSteps *Draw random steps*

Description

Draw random steps from a ltraj object using empirical distributions of step lengths and turning angles.

Usage

```
rdSteps(x, nrs = 10, rand.dis = NULL, only.others = FALSE,
  simult = FALSE, distMax = Inf, strata = NULL, mask = NULL,
  reproducible = FALSE)
```

Arguments

x	A ltraj object.
nrs	The number of random steps to draw for each observed step (default = 10).
rand.dis	The random distributions for step lengths and turning angles to use. If NULL (default), it uses x as a basis; otherwise, another ltraj object must be provided, or a data.frame with columns "dist", "rel.angle" and "id".
only.others	Logical, draws step lengths and turning angles from all other individuals, excluding the current one.
simult	Logical, whether to draw step lengths and turning angles simultaneously, i.e. both measurements come from a single observed step, instead of being drawn independently.
distMax	Only draw step lengths and turning angles using steps shorter than this threshold. Default is Inf, i.e. all steps are kept.
strata	Character. How to name the stratas. In all cases, a new column strata is created. If NULL, a series of integer from 1 to the total number of steps (on every individuals) is used; if "row.names", the row names are used; otherwise the column provided is used.
mask	A SpatialPolygons used to redraw steps that end outside of the polygon. Use with caution: there is absolutely no guarantee that the function is not caught in an endless loop (warnings are issued if there are long or very long loops). Note that it is assumed that steps are in the same projection as the mask (if it is not the case, project the mask first using spTransform).
reproducible	Logical. If TRUE, results are made reproducible with the use of a seed, otherwise new random step lengths and turning angles are sampled at each call.

Details

Note that 1) only complete steps are kept (i.e. steps characterized by start and end points, and turning angle (i.e. relative angle to the previous step); and 2) the information stored in infolocs is transfered to all random steps within a strata (i.e. it assumes it is the same for all steps within a strata).

Value

A data frame, with new columns case (1 for observed steps and 0 for random steps) and strata (a unique value for each set of paired observed and random steps).

Author(s)

Mathieu Basille <basille@ase-research.org>

Examples

```
## Load the data
data(puechcirc)
##'
## Simple example to check the distributions of step lengths and turning
```

```
## angles
bla <- rdSteps(puechcirc)
boxplot(bla$rel.angle ~ bla$case)
boxplot(bla$dist ~ bla$case)

## Reproducibility and alternative random distributions

## 1) Default: using the same ltraj for the random distributions:
bla <- rdSteps(puechcirc, reproducible = TRUE)

## 2) Explicitly use the same ltraj for the random distributions:
bli <- rdSteps(puechcirc, rand.dis = puechcirc, reproducible = TRUE)

## Check that 2) is the same as 1)
all.equal(bla, bli)

## 3) Explicitly uses random distributions in a data.frame:
rand <- subset(ld(puechcirc), !(is.na(x) | is.na(dx) | is.na(rel.angle)) &
    dist <= Inf, select = c("dist", "rel.angle", "id"))
blo <- rdSteps(puechcirc, rand.dis = rand, reproducible = TRUE)

## Check that 3) is the same as 1)
all.equal(bla, blo)
```

`rec` *Recalculates the descriptive parameters of a ltraj*

Description

Modified version of `rec` that keeps the original `row.names`. Also throws an error if the `row.names` are different between the ltraj and its infolocs.

Usage

```
rec(x, slsp = c("remove", "missing"))
```

Author(s)

Modified by Mathieu Basille <basille@ase-research.org>

See Also

See `rec` for further details on the function and all available arguments.

Examples

```
data(puechcirc)
(bla <- rec(puechcirc))
head(puechcirc[[2]])
head(bla[[2]])
```

setNA *Place Missing Values in Objects of Class* ltraj

Description

The function does not allow a numeric for date.ref anymore. In addition, a warning is issued if the length of date.ref is not 1 or the length of the ltraj object. See setNA for further details on the function and all available arguments.

Usage

```
setNA(ltraj, date.ref, dt, tol = dt/10, units = c("sec", "min", "hour",
  "day"), ...)
```

Author(s)

Modified by Mathieu Basille <basille@ase-research.org>

See Also

See setNA for further details on the function and all available arguments.

subset.ltraj *Subsetting a ltraj*

Description

Return subsets of a ltraj which meet conditions (over the descriptive parameters of the ltraj or its infolocs).

Usage

```
## S3 method for class 'ltraj'
subset(x, subset, rec = FALSE, ...)
```

Arguments

x	A ltraj object.
subset	Logical expression indicating elements or rows to keep: missing values are taken as false.
rec	Logical, whether to recompute the ltraj parameters or not (default = FALSE).

Value

A ltraj object.

Author(s)

Mathieu Basille <basille@ase-research.org>

See Also

See `which.ltraj` to identify the relocations fullfilling a condition.

Examples

```
## Load puechcirc data and add some infolocs:
data(puechcirc)
info <- list(data.frame(A = rnorm(80), B = runif(80), C = rpois(80,
    1)), data.frame(A = rnorm(69, 10), B = runif(69, 10, 11),
    C = rpois(69, 10)), data.frame(A = rnorm(66, -10), B = runif(66,
    -10, -9), C = rpois(66, 100)))
## Watch the row names:
info <- mapply(function(x, y) {
    row.names(x) <- row.names(y)
    return(x)
}, info, puechcirc, SIMPLIFY = FALSE)
infolocs(puechcirc) <- info

## Different subsets:
(xsub1 <- subset(puechcirc, dist > 200, rec = FALSE))
(xsub2 <- subset(puechcirc, C == 3, rec = TRUE))
(xsub3 <- subset(puechcirc, C == 3, rec = FALSE))
```

`trajdyn` *Interactive Display of Objects of Class* `ltraj`

Description

Modified version of `trajdyn`, which allows to 1) increment by by points/steps, 2) only display `k` previous points/steps, 3) show the current step as a vector, 4) to modify point and line display, 5) show the current burst/loc/(infolocs) and 6) to update interactively a new or existing variable in `infolocs`.

Usage

```
trajdyn(x, burst = attr(x[[1]], "burst"), na.rm = TRUE, hscale = 1,
  vscale = 1, addvec = 1, by = 1, only = Inf, recycle = TRUE,
  ppar = list(pch = 16), lpar = list(lwd = 2), nvar = NULL,
  display = c("guess", "windows", "tk"), ...)
```

Arguments

na.rm	Logical, whether to remove missing locations.
addvec	Numeric, whether to hihglight the current location (1, default), the current step (2) or nothing (0).
by	The number of previous points/steps to increment at each step. Default is an increment of 1 point/step.
only	The number of previous points/steps to display. Default is Inf, i.e. all points/steps.
ppar	A list of arguments that allows the user to modify point display, using any argument available to points. Default is list(pch = 16). See Details.
lpar	A list of arguments that allows the user to modify line display, using any argument available to lines. Default is list(lwd = 2). See Details.
nvar	A character string giving the name of a variable.

Details

y selects the number of previous points/steps to increment at each step. It defaults to 1, i.e. an increment of 1 point/step. Choosing anything different than a positive number sets it back to 1.

k selects the number of previous points/steps to display. It defaults to Inf, i.e. all points/steps. Choosing anything different than a positive number sets it back to Inf.

v shows (or removes) the current step as a vector. Note that the vector connects the current location to the next available location (even if there are NAs in the data set).

s shows the current buffer and localisation numbers, together with the associated infolocs (if it exists). If a nvar is requested, only this variable is shown.

The argument nvar allows to work on a given variable: if a variable of that name already exists in infolocs(x), the values of the variable are retrieved from there; otherwise, a variable filled with NAs is used. If a nvar is requested, d "deletes" the value and resets it to NA; every other letter not in use in any option is saved in nvar. The letters available are: "c", "e", "f", "h", "j", "m", "t", "u", "w", "x".

On a QWERTY keyboard:

we t u f hj xcm

On a AZERTY keyboard:

e t u f hj m wxc

It is possible to use point and line parameters globally for every trajectory displayed. In this case, ppar and lpar need just be a list of graphical parameters, such as list(pch = 21,col = "black", bg = "white"). It is also possible to use parameters for single steps, using as graphical parameter a list of vectors of length equal to each trajectory. Such information can be based on infolocs, see Example.

Value

If a nvar is provided, return the original ltraj with updated values in infolocs(nvar).

Author(s)

Modified by Mathieu Basille <basille@ase-research.org>

Examples

```
## Not run:
data(puechcirc)
##'
## Use of `by` and `only` to select the previous k points/steps:
trajdyn(puechcirc, by = 10, only = 20)
##'
## Use of `ppar` and `lpar` globally:
trajdyn(puechcirc, ppar = list(col = "red"), lpar = list(col = "blue"))
##'
## Create some random `infolocs`:
info <- list(data.frame(col = sample(c("red", "grey"),
         80, rep = TRUE), stringsAsFactors = FALSE),
     data.frame(col = sample(c("blue", "darkred"),
         69, rep = TRUE), stringsAsFactors = FALSE),
     data.frame(col = sample(c("darkgreen", "purple"),
         66, rep = TRUE), stringsAsFactors = FALSE))
## Watch the row names:
info <- mapply(function(x, y) {
    row.names(x) <- row.names(y)
    return(x)
}, info, puechcirc, SIMPLIFY = FALSE)
infolocs(puechcirc) <- info
##'
## Use the infolocs to color points and steps:
trajdyn(puechcirc, by = 1, only = 20, ppar = list(pch = 19,
    col = infolocs(puechcirc, "col", simplify = TRUE)),
    lpar = list(col = infolocs(puechcirc, "col", simplify = TRUE)))
##'
## The same without removing the missing locations:
trajdyn(puechcirc, by = 1, only = 20, ppar = list(pch = 19,
    col = infolocs(puechcirc, "col", simplify = TRUE)),
    lpar = list(col = infolocs(puechcirc, "col", simplify = TRUE)),
    na.rm = FALSE)
##'
## Use of `nvar` to dynamically fill in new data:
(newtraj <- trajdyn(puechcirc, nvar = "Var"))

## End(Not run)
```

Index