

Package ‘trip’

October 12, 2009

Type Package

Title Spatial analysis of animal track data

Version 1.1-3

Depends methods, sp

Author Michael D. Sumner

Maintainer Michael D. Sumner <mdsumner@utas.edu.au>

Description Functions for accessing and manipulating spatial data for animal tracking. Filter for speed and create time spent plots from animal track data.

License GPL-2

Repository CRAN

Date/Publication 2009-10-12 10:36:51

R topics documented:

trip-package	2
adjust.duplicateTimes	2
argos.sigma	3
forceCompliance	4
makeGridTopology	5
oc.theme	5
readArgos	6
readDiag	8
sepIdGaps	8
speedfilter	9
spTransform-methods	11
TimeOrderedRecords	11
TimeOrderedRecords-class	12
trackDistance	13
trip	14
trip-class	15
tripGrid	16

Index**18**

trip-package	<i>Spatial analysis of animal track data</i>
--------------	--

Description

Functions for accessing and manipulating spatial data for animal tracking. Filter for speed and create time spent plots from animal track data. Classes `TimeOrderedRecords` and `trip` provide an extension to the `Spatial` classes.

Details

Package:	trip
Type:	Package
Version:	0.0-4
Date:	2006-08-10
License:	GPL2

Author(s)

Michael D. Sumner

Maintainer: Michael D. Sumner <mdsumner@utas.edu.au>

adjust.duplicateTimes	<i>Adjust duplicate DateTime values</i>
-----------------------	---

Description

Duplicated `DateTime` values within `ID` are adjusted forward (recursively) by one second until no duplicates are present. This is considered reasonable way of avoiding the nonsensical problem of duplicate times.

Usage

```
adjust.duplicateTimes(time, id)
```

Arguments

<code>time</code>	vector of <code>DateTime</code> values
<code>id</code>	vector of <code>ID</code> values, matching <code>DateTimes</code> that are assumed sorted within <code>ID</code>

Details

This function is used to remove duplicate time records in animal track data, rather than removing the record completely.

Value

The adjusted DateTime vector is returned.

Warning

I have no idea what goes on at CLS when they output data that are either not ordered by time or have duplicates. If this problem exists in your data it's probably worth finding out why.

Author(s)

Michael D. Sumner

References

<http://staff.acecrc.org.au/~mdsumner/>

See Also

[readArgos](#)

Examples

```
## DateTimes with a duplicate within ID
tms <- Sys.time() + c(1:6, 6, 7:10) *10
id <- rep("a", length(tms))
range(diff(tms))

## duplicate record is now moved one second forward
tms.adj <- adjust.duplicateTimes(tms, id)
range(diff(tms.adj))
```

argos.sigma

Assign numeric values for Argos "class"

Description

Assign numeric values for Argos "class" by matching the levels available to given numbers. An adjustment is made to allow sigma to be specified in kilometers, and the values returned are the approximate values for longlat degrees. It is assumed that the levels are part of an "ordered" factor from least precise to most precise.

Usage

```
argos.sigma(x, sigma = c(100, 80, 50, 20, 10, 4, 2), adjust = 111.12)
```

Arguments

<code>x</code>	factor of Argos location quality "classes"
<code>sigma</code>	numeric values (by default in kilometres)
<code>adjust</code>	a numeric adjustment to convert from kms to degrees

Details

The available levels in Argos are `levels = c("Z", "B", "A", "0", "1", "2", "3")`.

The actual sigma values given by default are (as far as can be determined) a reasonable stab at what Argos believes.

Value

Numeric values for given levels.

Examples

```
cls <- ordered(sample(c("Z", "B", "A", "0", "1", "2", "3"), 30, replace = TRUE),
               levels = c("Z", "B", "A", "0", "1", "2", "3"))
argos.sigma(cls)
```

`forceCompliance` *Function to ensure dates and times are in order with trip ID*

Description

A convenience function, that removes duplicate rows, sorts by the date-times within ID, and removes duplicates from a data frame or `SpatialPointsDataFrame`.

Usage

```
forceCompliance(x, tor)
```

Arguments

<code>x</code>	data.frame or <code>SpatialPointsDataFrame</code>
<code>tor</code>	character vector of names of date-times and trip ID columns

Value

Dataframe or `SpatialPointsDataFrame`.

Note

It's really important that data used are of a given quality, but this function makes the most common trip problems easy to apply.

Author(s)

Michael D. Sumner

See AlsoSee Also [trip](#)

 makeGridTopology *Generate a GridTopology from a Spatial object*

Description

Sensible defaults are assumed, to match the extents of data to a manageable grid. Approximations for kilometres in longlat can be made using `cellsize`.

Usage

```
makeGridTopology(obj, cells.dim = c(100, 100), xlim = NULL, ylim = NULL, buffer = 0)
```

Arguments

<code>obj</code>	~~Describe obj here~~
<code>cells.dim</code>	~~Describe cells.dim here~~
<code>xlim</code>	~~Describe xlim here~~
<code>ylim</code>	~~Describe ylim here~~
<code>buffer</code>	~~Describe buffer here~~
<code>cellsize</code>	~~Describe cellsize here~~

 oc.theme *SeaWiFS ocean colour colours*

Description

Generate ocean colour colours, using the SeaWiFS scheme

Usage

```
oc.theme(x = 50)
oc.colors(n)
```

Arguments

<code>x</code>	Number of colours to generate as part of a theme
<code>n</code>	Number of colours to generate

Details

This is a high-contrast palette, log-scaled originally for ocean chlorophyll.

Value

A set of colours or a theme object.

Author(s)

Michael D. Sumner

See Also

Similar functions in sp [sp.theme](#), [bpy.colors](#)

Examples

```
oc.colors(10)
library(lattice)
trellis.par.set(oc.theme)
```

readArgos

Read Argos "DAT" files

Description

Return a (Spatial) data frame of location records from raw Argos files. Multiple files may be read, and each set of records is appended to the data frame in turn. Basic validation of the data is enforced by default.

Usage

```
readArgos(x, correct.all = TRUE, dtFormat = "%Y-%m-%d %H:%M:%S", tz = "GMT", duplicateTimes.eps
```

Arguments

<code>x</code>	vector of file names of Argos data
<code>correct.all</code>	logical - enforce validity of data as much as possible? (see Details)
<code>dtFormat</code>	the DateTime format used by the Argos data "date" and "time" pasted together
<code>tz</code>	timezone - GMT/UTC is assumed
<code>duplicateTimes.eps</code>	what is the tolerance for times being duplicate?
<code>p4</code>	PROJ.4 projection string, "+proj=longlat" is assumed

Details

Basic validation checks for class `trip` are made, and enforced based on `correct.all`:

No duplicate records in the data, these are simply removed. Records are ordered by `DateTime` ("date", "time", "gmt") within ID ("ptt"). No duplicate `DateTime` values within ID are allowed: to enforce this the time values are moved forward by one second - this is done recursively and is not robust.

If validation fails the function will return a `SpatialPointsDataFrame`. Files that are not obviously of the required format are skipped.

Argos location quality data "class" are ordered, assuming that the available levels is `levels = c("Z", "B", "A", "0", "1", "2", "3")`.

A projection string is added to the data, assuming the PROJ.4 longlat - if any longitudes are greater than 360 the PROJ.4 argument "+over" is added.

Value

A `trip` object, if all goes well, or simply a `SpatialPointsDataFrame`.

Warning

This works on some Argos files I have seen, it is not a guaranteed method and is in no way linked officially to Argos.

Author(s)

Michael D. Sumner

References

The Argos data documentation is at http://www.cls.fr/welcome_en.html. Specific details on the PRV ("provide data") format can be found here http://www.cls.fr/manuel/html/chap4/chap4_4_8.htm.

See Also

`trip`, `SpatialPointsDataFrame`, `adjust.duplicateTimes`, for manipulating these data, and `argos.sigma` for relating a numeric value to Argos quality "classes". `sepIdGaps` for splitting the IDs in these data on some minimum gap.

`order`, `duplicated`, `,`, `ordered` for general R manipulation of this type.

readDiag *Read Argos DIAG format*

Description

Create a dataframe from Argos diag files.

Usage

```
readDiag(x)
```

Arguments

x one or more names of DIAG files

Details

~~ If necessary, more details than the description above ~~

Value

A data frame with 8 columns.

lon1, lat1	first pair of coordinates
lon2, lat2	second pair of coordinates
gmt	DateTimes as POSIXct
id	Platform Transmitting Terminal (PTT) ID
lq	Argos location quality class
iq	some other thing

sepIdGaps *Separate a set of IDs based on gaps*

Description

A new set of ID levels can be created by separating those given based on a minimum gap in another set of data. This is useful for separating instruments identified only by their ID into separate events in time.

Usage

```
sepIdGaps(id, gapdata, minGap = 3600 * 24 * 7)
```

Arguments

<code>id</code>	existing ID levels
<code>gapdata</code>	data matching <code>id</code> with gaps to use as separators
<code>minGap</code>	the minimum "gap" to use in <code>gapdata</code> to create a new ID level

Details

The assumption is that a week is a long time for a tag not to record anything.

Value

A new set of ID levels, named following the pattern that "ID" split into 3 would provided "ID", "ID_2" and "ID_3".

Warning

It is assumed that each vector provides is sorted by `gapdata` within `id`. No checking is done, and so it is suggested that this only be used on ID columns within existing, validated `trip` objects

Author(s)

Michael D. Sumner

See Also

[trip](#)

Examples

```
id <- gl(2, 8)
gd <- Sys.time() + 1:16
gd[c(4:6, 12:16)] <- gd[c(4:6, 12:16)] + 10000

sepIdGaps(id, gd, 1000)
```

speedfilter

Filter track data for speed

Description

Create a filter of a track for "bad" points implying a speed of motion that is unrealistic.

Usage

```
speedfilter(x, max.speed = NULL, test = FALSE, ...)
```

Arguments

<code>x</code>	trip object
<code>max.speed</code>	speed in kilometres per hour
<code>test</code>	cut the algorithm short and just return first pass
<code>...</code>	

Details

Using an algorithm (McConnell et al, 1992), points are tested for speed between previous / next and 2nd previous / next points. Contiguous sections with an root mean square speed above a given maximum have their highest rms point removed, then rms is recalculated, until all points are below the maximum. By default an (internal) root mean square function is used, this can be specified by the user.

If the coordinates of the `trip` data are not projected, or NA the distance calculation assumed longlat and kilometres (great circle). For projected coordinates the speed must match the units of the coordinate system. (The PROJ.4 argument "units=km" is suggested).

Value

Logical vector matching positions in the coordinate records that pass the filter.

Warning

This algorithm is not considered to be particularly relevant to the problems involved with location uncertainty in animal tracking. It is provided merely as an illustrative benchmark for further work.

It is possible for the filter to become stuck in an infinite loop, depending on the function passed to the filter. Several minutes is probably too long for hundreds of points, test on smaller sections if unsure.

Note

This algorithm was originally taken from IDL code by David Watts at the Australian Antarctic Division, and used in various other environments before the development of this version.

Author(s)

Michael D. Sumner

References

The algorithm comes from McConnell, B. J. and Chambers, C. and Fedak, M. A. (1992) Foraging ecology of southern elephant seals in relation to the bathymetry and productivity of the southern ocean. *Antarctic Science* 4 393-398

See Also

[trip](#)

`spTransform-methods`*Methods for Function spTransform in package "trip"*

Description

Transformation in package "trip" uses the facilities of the rgdal and sp packages.

Methods

"trip", CRSobj = CRS returns transformed coordinates of a "trip" object using the projection arguments in "CRSobj", of class CRS

See Also

Packages rgdal and sp for full documentation.

`TimeOrderedRecords` *Functions to specify and obtain DateTime and ID data from within (Spatial) data frames.*

Description

Specifies the DateTime and ID data within a data frame for objects of class `trip`. Functions also for obtaining the names of the columns used, and the data itself.

Usage

```
TimeOrderedRecords(x)
getTORnames(obj)
getTimeID(obj)
```

Arguments

<code>x</code>	Character vector of 2-elements, specifying the data columns of DateTimes and IDs.
<code>obj</code>	<code>trip</code> object.

Details

These simple functions are for creating classes with TimeOrdered data. The main use is for SpatialPointsDataFrames which are used with TimeOrderedRecords for the class `trip`.

Value

`TimeOrderedRecords` returns an object with a 2-element character vector, specifying the column names. `getTORnames` obtains the column names from an object extending the class, and `getTimeID` returns the data as a data frame from an object extending the class.

See Also

[trip](#), for the use of this class with [SpatialPointsDataFrame](#)

Examples

```
tor <- TimeOrderedRecords(c("time", "id"))
getTORnames(tor)
```

```
TimeOrderedRecords-class
      Class "TimeOrderedRecords"
```

Description

A simple class to act as a place-holder for DateTime and ID records in spatial data

Objects from the Class

Objects can be created by calls of the form `new("TimeOrderedRecords", TOR.columns)`. `TOR.columns` are a 2-element character vector specifying the DateTime and ID columns in an object of class `trip`.

Slots

TOR.columns: 2-element vector of class "character"

Methods

trip signature(obj = "ANY", TORnames = "TimeOrderedRecords"): create a trip object from a data frame

trip signature(obj = "trip", TORnames = "TimeOrderedRecords"): create a trip object from an existing trip object

Note

Future versions may change significantly, this class is very basic and could probably be implemented in a better way. Specifying TOR columns by formula would be a useful addition.

Author(s)

Michael D. Sumner

References

~put references to the literature/web site here ~

See Also

See Also [trip](#) for creating trip objects, and [trip-class](#) for the class.

Examples

```
tor <- new("TimeOrderedRecords", TOR.columns = c("datetime", "ID"))
tor <- TimeOrderedRecords(c("datetime", "ID"))
```

trackDistance	<i>Determine distance along a track</i>
---------------	---

Description

Calculate the distance between subsequent 2-D coordinates.

Usage

```
trackDistance(track, longlat = FALSE, push = 1)
```

Arguments

track	matrix of 2-columns, with x/y coordinates
longlat	logical - are the data longlat?
push	an offset for measuring distance between this and the next (push) location

Details

This uses the (fast) `spDistsN1`.

Value

Vector of distances between coordinates.

`trip`*Function to handle animal track data, organized as "trip"s*

Description

Extend the basic functionality of a `Spatial` data frame by specifying the data columns that define the "TimeOrdered" quality of the records.

Usage

```
trip(obj, TORnames)
```

Arguments

<code>obj</code>	A <code>SpatialPointsDataFrame</code> , containing at least two columns with the <code>DateTime</code> and <code>ID</code> data as per <code>TORnames</code>
<code>TORnames</code>	Either an object of <code>TimeOrderedRecords</code> , or a 2-element character vector specifying the <code>DateTime</code> and <code>ID</code> column of <code>obj</code>

Value

A `trip` object, with the usual slots of a `SpatialPointsDataFrame` and the added `TimeOrderedRecords`. For the most part this can be treated as a `data.frame` with `Spatial` coordinates.

Author(s)

Michael D. Sumner

See Also

[speedfilter](#), and [tripGrid](#) for simple(istic) speed filtering and spatial time spent gridding.

Examples

```
d <- data.frame(x = 1:10, y = rnorm(10), tms = Sys.time() + 1:10, id = gl(2, 5))
coordinates(d) <- ~x+y
tr <- trip(d, c("tms", "id"))
```

trip-class	<i>Class "trip" ~~~</i>
------------	-------------------------

Description

An extension of "SpatialPointsDataFrame" by including "TimeOrderedRecords". The records within the data frame are explicitly ordered by DateTime data within IDs.

Objects from the Class

Objects can be created by calls of the form `trip(obj = "SpatialPointsDataFrame", TORnames = "TimeOrderedRecords")`. The object contains all the slots present within a `SpatialPointsDataFrame`, particularly `data` which contains columns of at least those specified by `TOR.columns`

Slots

TOR.columns: Object of class "character" specifying the DateTime and ID columns (in that order) in `data`

data: Object of class "data.frame" the native data object for a Spatial data frame

Also, other slots usual to a `SpatialPointsDataFrame`

Extends

Class "TimeOrderedRecords", directly. Class "SpatialPointsDataFrame", directly. Class "SpatialPoints", by class "SpatialPointsDataFrame". Class "Spatial", by class "SpatialPointsDataFrame".

Methods

Most of the methods available are by virtue of the `sp` package. Some, such as `split.data.frame` have been added to SPDF so that `trip` has the same functionality.

trip signature ("ANY"): try to return a trip object

trip signature (obj = "SpatialPointsDataFrame", TORnames = "ANY") The usual construction

[signature (x = "trip"): subset rows or columns as per `SpatialPointsDataFrame`. In the case that `TimeOrderedRecords` columns are dropped, the object reverts to the straight `Spatial` version.

lines signature (x = "trip"): add lines to a plot with separate colours for each trip

plot signature (x = "trip", y = "missing"): plot as `SpatialPoints`

points signature (x = "trip"): add points to a plot using the Spatial coordinates

recenter signature (obj = "trip"): perform coordinate recentering, from the [-180,180] convention to [0, 360]

show signature (object = "trip"): print a short summary of the trip data

summary signature(object = "trip"): print a summary as per SpatialPointsDataFrame including a summary of the trip data

text signature(x = "trip"): add text to a plot using Spatial coordinates

trip signature(obj = "trip", TORnames = "TimeOrderedRecords"): (re)-create a trip object using TimeOrderedRecords

trip signature(obj = "trip", TORnames = "ANY"): (re)-create a trip object by some other means

subset signature(x = "trip", ...): subset a trip in the expected manner.

Warning

There are some kludges to allow `trip` to do things, such as replace `POSIXt` column data using `"$<-.trip"` and `"[<-.trip"` which should not be necessary once `sp` implements the new `data.frame` class of R >=2.4.0.

Note

~~further notes~~

Author(s)

~~who you are~~

References

~put references to the literature/web site here ~

See Also

[trip](#) for examples of directly using the class.

Examples

```
##----- Should be DIRECTLY executable !! -----
```

```
tripGrid
```

```
Generate a grid of time spent
```

Description

Create a grid of time spent from an object of class `trip` by approximating the time between locations for separate trip events.

Usage

```
tripGrid(x, grid = NULL, method = "count", dur = NULL, ...)
interpequal(x, dur = NULL, quiet = FALSE)
countPoints(x, dur = 1, grid = NULL)
kdePoints(x, h = NULL, grid = NULL, resetTime = TRUE, ...)
```

Arguments

x	object of class trip
grid	GridTopology - will be generated automatically if NULL
method	name of method for quantifying time spent, see Details
...	other arguments passed to <code>interpequal</code> or <code>kdePoints</code>
dur	The duration of time used to interpolate between available locations (see Details)
quiet	logical - report on difference between time summed and time in trip?
h	numeric vector of two elements specifying bandwidth for kernel density
resetTime	logical - reset the values of the kde grid to match the sum of the total time?

Details

The intention is for `tripGrid` to be used, rather than the underlying functions for interpolation or gridding.

Trip locations are first interpolated, based on an equal-time spacing between records. These interpolated points are then "binned" to a grid of cells. The time spacing is specified by the "dur"ation argument to `interpequal` in seconds (i.e. `dur = 3600` is used for 1 hour). Shorter time periods will require longer computation with a closer approximation to the total time spent in the gridded result.

Currently there are methods "count" and "kde" for quantifying time spent, corresponding to the functions "countPoints" and "kdePoints". "kde" uses kernel density to smooth the locations, "count" simply counts the points falling in a grid cell.

Value

`tripGrid` returns an object of class `SpatialGridDataFrame`, with one column "z" containing the time spent in each cell (usually assumed to be in hours).

Index

*Topic **IO**

readArgos, 6
readDiag, 7

*Topic **classes**

TimeOrderedRecords-class, 11
trip-class, 14

*Topic **color**

oc.theme, 5

*Topic **manip**

adjust.duplicateTimes, 2
argos.sigma, 3
forceCompliance, 4
makeGridTopology, 4
readArgos, 6
readDiag, 7
sepIdGaps, 8
speedfilter, 9
TimeOrderedRecords, 10
trackDistance, 12
trip, 13
tripGrid, 15

*Topic **methods**

spTransform-methods, 10

*Topic **package**

trip-package, 1

*Topic **spatial**

spTransform-methods, 10
[, trip-method (*trip-class*), 14
[[<- , trip, ANY, missing-method
(*trip-class*), 14
\$<- , trip, character (*trip-class*),
14
\$<- , trip, character-method
(*trip-class*), 14

adjust.duplicateTimes, 2, 7
argos.sigma, 3, 7
as.data.frame.trip (*trip-class*),
14

bpy.colors, 5

countPoints (*tripGrid*), 15

duplicated, 7

forceCompliance, 4

getTimeID (*TimeOrderedRecords*), 10
getTORnames (*TimeOrderedRecords*),
10

interpequal (*tripGrid*), 15

kdePoints (*tripGrid*), 15

lines, trip-method (*trip-class*), 14

makeGridTopology, 4

names.trip (*trip-class*), 14
names<- .trip (*trip-class*), 14

oc.colors (*oc.theme*), 5
oc.theme, 5
order, 7
ordered, 7

plot, trip, missing-method
(*trip-class*), 14

points, trip-method (*trip-class*),
14

readArgos, 3, 6
readDiag, 7
recenter, trip-method
(*trip-class*), 14

sepIdGaps, 7, 8
show, trip-method (*trip-class*), 14
sp.theme, 5
Spatial, 1

SpatialPointsDataFrame, 7, 11
speedfilter, 9, 13
spTransform
 (*spTransform-methods*), 10
spTransform, ANY-method
 (*spTransform-methods*), 10
spTransform, trip, CRS-method
 (*spTransform-methods*), 10
spTransform-methods, 10
subset, trip-method (*trip-class*),
 14
summary, trip-method (*trip-class*),
 14

text, trip-method (*trip-class*), 14
TimeOrderedRecords, 10
TimeOrderedRecords-class, 11
trackDistance, 12
trip, 4, 7, 8, 10–12, 13, 15
trip, ANY, ANY (*trip-class*), 14
trip, ANY, TimeOrderedRecords-method
 (*TimeOrderedRecords-class*),
 11
trip, SpatialPointsDataFrame, ANY-method
 (*trip-class*), 14
trip, trip, ANY-method
 (*trip-class*), 14
trip, trip, TimeOrderedRecords-method
 (*TimeOrderedRecords-class*),
 11
trip, trip-method (*trip-class*), 14
trip-class, 12
trip-class, 14
trip-package, 1
tripGrid, 13, 15