Package ‘TrackReconstruction’

October 12, 2022

Type Package

Title Reconstruct Animal Tracks from Magnetometer, Accelerometer,
Depth and Optional Speed Data

Version 1.3

Date 2021-12-10

Author Brian Battaile

Maintainer Brian Battaile <brian.battaile@gmail.com>

Depends R(>= 2.10.0), fields, RColorBrewer

Suggests lattice, onion, plotrix, rgl, scatterplot3d

Description Reconstructs animal tracks from magnetometer, accelerometer, depth and op-
tional speed data. Designed primarily using data from Wildlife Computers Daily Diary tags de-
ployed on northern fur seals.

License GPL (>= 2)

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2021-12-10 22:00:09 UTC

R topics documented:

TrackReconstruction-package ........................................... 2
bathymetry .................................................................. 4
CalcBearing .................................................................. 5
CalcDistance ................................................................ 6
CalcLatitude ................................................................ 7
CalcLongitude ............................................................... 8
DeadReckoning ................................................................ 9
GapFinder .................................................................... 11
GeoRef ....................................................................... 12
georef1min01 ............................................................... 14
georef1min02 ............................................................... 15
Description

Given accelerometer, magnetometer, depth and optional speed data, this package will reconstruct animal paths. Given GPS data, pseudotracks can be geolocated. The package was designed using northern fur seals (Callorhinus ursinus) but is probably suitable as a launching pad for other animals.

Details

Package: TrackReconstruction
Type: Package
Version: 1.3
Date: 2021-12-10
License: GPL (>=2)

Includes functions to Standardize logger data, format GPS data, find gaps in the raw data, navigation functions to calculate distance, bearing, latitude and longitude, functions to calculate pseudotracks, to georeference the tracks and finally to plot the tracks with color bathymetry. A vignette includes some additional Perl code to deal with large data files as well as additional plotting code to make 3D plots and 2D animations.

DeadReckoning
TrackReconstruction-package

GapFinder
GeoReference
GeoRef (a wrapper for GeoReference)
GPStable
GraphLimits
Mapper
Navigation Functions (CalcBearing CalcLatitude CalcLongitude CalcDistance)
Splitter
Standardize

Author(s)
Brian Battaile
Maintainer: Brian Battaile <brian.battaile@gmail.com>

References

See Also
R packages diveMove, adehabitat, animalTrack, argosfilter and crawl

Examples
betas<-Standardize(1,1,-1,1,1,1,-57.8,68.76,-61.8,64.2,-70.16,58.08,
-10.1,9.55,-9.75,9.72, -9.91,9.43)
# get declination and inclination data for study area
decinc<-c(10.228,65.918)
# data set with 6 associated GPS fixes in the "gpsdata" data set
data(rawdata)
DRoutput<-DeadReckoning(rawdata,betas,decinc,Hz=16,RmL=2,DepthHz=1,SpdCalc=3,MaxSpd=3.5)
# prepare GPS data
data(gpsdata02)
gpsformat<-GPStable(gpsdata02)
Georeferenced<-GeoRef(DRoutput,gpsformat)
plot(Georeferenced$Longitude,Georeferenced$Latitude,pch=".") #Start
points(gpsformat$Longitude[2],gpsformat$Latitude[2],pch="S",col="Red") #Start
points(gpsformat$Longitude[7],gpsformat$Latitude[7],pch="F",col="Blue") # Finish
bathymetry

# Intermediate GPS points
points(gpsformat$Longitude[3:6],gpsformat$Latitude[3:6],pch="*",col="Red")

## Not run:
# plot the data with a bathymetric background, note how the axis dimensions have changed to
give a more realistic path relative to that produced in the simple plot call. See the vignette
# or ?bathymetry for information on how to get bathymetric data for your study area.
data(bathymetry)
# This may take a minute or two
image.xyz=tapply(bathymetry$Depth, list(bathymetry$Long, bathymetry$Lat), unique)
Mapper(Georeferenced, gpsformat[2:7,], image.xyz, ExpFact = 500, minlat = 51, maxlat = 60,
minlong = -177, maxlong = -163, Title = "Fun Graph!!!")
# That entire animals trip plotted
data(georef1min02)
data(gpsdata02)
Mapper(georef1min02, gpsdata02, image.xyz, ExpFact = 500, minlat = 51, maxlat = 60,
minlong = -177, maxlong = -163, Title = "Fun Graph!!!")

## End(Not run)

bathymetry

Bathymetry data for the Eastern Bering Sea

Description

Bathymetry data for the Eastern Bering Sea at a resolution of 30 arc seconds downloaded from
http://www.gebco.net

Usage

data(bathymetry)

Format

A data frame with 1814400 observations on the following 3 variables.

Long  a numeric vector
Lat   a numeric vector
Depth a numeric vector

Details

If you want to do color graphing of the tracks within R I have provided a Mapper function to do
this but it requires gridded (raster) bathymetric data to create the background map. The source
for this example data set comes from the General Bathymetric Chart of the Oceans or GEBCO at
https://www.gebco.net.
Examples

data(bathymetry)
str(bathymetry)
head(bathymetry);tail(bathymetry)
bathymetryBogs<-subset(bathymetry,Long<(-166) & Long >=(-169)
& Lat<= 54.5 & Lat >=53,select=Long:Depth)

image.xyz=tapply(bathymetryBogs$Depth, list(bathymetryBogs$Lon, bathymetryBogs$Lat), unique)
#create palette for depth colors
Bathymetry.palette<-colorRampPalette(brewer.pal(9, "Blues"),bias=3)

#Plot the background map image
image.plot(image.xyz,
col=c(rev(Bathymetry.palette(200)),terrain.colors(100)),#gray(0:20/20),
breaks=round(c(seq(from=min(image.xyz),to=0,length.out=201),seq(from=max(image.xyz)/101
,to=max(image.xyz),length.out=100))))
,#smallplot=2 #plots legend off x axis
)
## Not run:
#If you want to map the entire bathymetry file, it takes a while
image.xyz=tapply(bathymetry$Depth, list(bathymetry$Lon, bathymetry$Lat), unique)
#create palette for depth colors
Bathymetry.palette<-colorRampPalette(brewer.pal(9, "Blues"),bias=3)

#Plot the background map image
image.plot(image.xyz,
col=c(rev(Bathymetry.palette(200)),terrain.colors(100)),#gray(0:20/20),
breaks=round(c(seq(from=min(image.xyz),to=0,length.out=201),seq(from=max(image.xyz)/101
,to=max(image.xyz),length.out=100))))
,#smallplot=2 #plots legend off x axis
)
## End(Not run)

CalcBearing

Description

Calculates Bearing given initial latitude and longitude and ending latitude and longitude in decimal
degrees and returns bearing in radians.

Usage

CalcBearing(initialLat, initialLong, finalLat, finalLong)

Arguments

initialLat  initial Latitude
initialLong  initial Longitude
finalLat  ending Latitude
finallong  ending Longitude
Details

Data must be provided in decimal degrees (e.g. 162.546). Longitude values west of the prime meridian are 0 to -180. Latitude values south of the equator are 0 to -90. As one travels from point A to point B, the bearing to point B changes. This function calculates the bearing from point A to point B at point A. This function is primarily used internally in the GPStable and GeoReference functions.

Value

Bearing in Radians

Author(s)

Brian Battaile

References

https://www.movable-type.co.uk/scripts/latlong.html

Examples

#Bearing to Los Angeles from New York
Bearing<-CalcBearing(40.7697,-73.9735,34.0522,-118.2428)
Bearing
Bearing*360/(2*pi) #transform radians to degrees

CalcDistance

Calculate Distance Between Two Points

Description

Calculates the distance between two points on the globe given latitude and longitude in decimal degrees and returning distance in kilometers.

Usage

CalcDistance(initialLat, initialLong, finalLat, finalLong)

Arguments

initialLat  initial Latitude
initialLong  initial Longitude
finalLat  ending Latitude
finalLong  ending Longitude
Details
Data must be provided in decimal degrees (e.g. 162.546). Longitude values west of the prime meridian are 0 to -180. Latitude values south of the equator are 0 to -90. This function is primarily used internally in the GPStable and GeoReference functions.

Value
Distance in Kilometers

Author(s)
Brian Battaile

References
https://www.movable-type.co.uk/scripts/latlong.html

Examples
```r
# Distance from New York to Los Angeles given Latitude and Longitude of New York and Los Angeles.
Howfar<-CalcDistance(40.7697,-73.9735,34.0522,-118.2428)
Howfar
```

CalcLatitude

Description
Calculates an unknown latitude given an initial latitude, distance traveled in meters and bearing.

Usage
CalcLatitude(initialLat, distance, bearing)

Arguments
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>initialLat</td>
<td>initial Latitude</td>
</tr>
<tr>
<td>distance</td>
<td>distance in meters</td>
</tr>
<tr>
<td>bearing</td>
<td>bearing in radians</td>
</tr>
</tbody>
</table>

Details
Data must be provided in decimal degrees (e.g. 162.546). Longitude values west of the prime meridian are 0 to -180. Latitude values south of the equator are 0 to -90. Calculates an unknown latitude given an initial latitude, distance traveled in meters and bearing in radians. This function is primarily used internally in the gpstable and GeoReference functions.
CalcLongitude

Value
Latitude in radians

Author(s)
Brian Battaile

References
https://www.movable-type.co.uk/scripts/latlong.html

Examples
#Calculate the latitude of Los Angeles given a starting point in Manhattan and the distance
#and bearing to Los Angeles.
Lat<-CalcLatitude (40.7697,3938000,-1.507516)
#transform to degrees
Lat*360/(2*pi)

CalculatedLongitude Calculate Longitude

Description
Calculate the longitude of a destination given initial latitude and longitude, distance traveled, bearing and final latitude.

Usage
CalcLongitude(initialLat, destinationLat, initialLong, distance, bearing)

Arguments
initialLat initial Latitude
destinationLat final Latitude
initialLong initial Longitude
distance distance from initial to final in meters
bearing bearing from point of departure to final destination

Details
Calculates a destination’s longitude given initial latitude and longitude in degrees (e.g.162.546, Longitude values west of the prime meridian are 0 to -180. Latitude values south of the equator are 0 to -90), destination latitude in RADIANS (result of CalcLatitude function), distance traveled in meters and bearing in radians. This function is primarily used internally in the GPStable and GeoReference functions.
DeadReckoning

Value

Longitude of final destination in radians.

Author(s)

Brian Battaile

References

https://www.movable-type.co.uk/scripts/latlong.html

Examples

#Longitude of Los Angeles starting in Manhattan
Long <- CalcLongitude(40.7697, 0.5943228, -73.9735, 3938000, -1.507516)
#transform to degrees
Long*360/(2*pi)

#Use results of CalcLatitude
Lat<CalcLatitude (40.7697, 3938000, -1.507516)
Long <- CalcLongitude(40.7697, Lat, -73.9735, 3938000, -1.507516)
#transform to degrees
Long*360/(2*pi)

DeadReckoning  Calculates Dead Reckoning

Description

This function takes triaxial magnetometer, accelerometer and optional speed data to estimate the
path traveled (pseudotrack) by a tagged animal based on Wilson et al. (2007).

Usage

DeadReckoning(rawdata, betas, decinc, Hz = 16, RmL = 2, DepthHz = 1, SpdCalc=1, MaxSpd=NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rawdata</td>
<td>matrix or data frame of magnetometer and accelerometer with column headings specified in Details.</td>
</tr>
<tr>
<td>betas</td>
<td>matrix or data frame of coefficients (slope and intercept) that standardizes magnetometer and accelerometer data between -1 and +1. This can come from the Standardize function.</td>
</tr>
<tr>
<td>decinc</td>
<td>vector of c(Declination, Inclination) values in decimal degrees format in that order.</td>
</tr>
</tbody>
</table>
**Hz**

Frequency of data collection in Hz of magnetometers. The assumption here is that the accelerometer Hz is at least as high as the magnetometers and that all the magnetometer channels have the same Hz. See Details.

**RmL**

Running mean length in seconds, required to separate "dynamic" and "static" acceleration.

**DepthHz**

Frequency of data collection in Hz of Depth channel.

**SpdCalc**

Integer of value 1, 2, 3 or 4 (5 and 6 are unsupported) depending on type of speed data supplied, see details.

**MaxSpd**

If SpdCalc=3, the max speed of your animal in m/s. If SpdCalc=4, the constant speed of your animal in m/s.

### Details

Raw data must have columns DateTime or separated as Date and Time, plus Depth, MagSurge, MagHeave, MagSway, AccSurge, AccHeave, AccSway and optional Speed in a data frame with those exact column names (Mag=Magnetometer and Acc=Accelerometer). It must have an extra number of rows =Hz*RmL*0.5 at beginning and end that will be trimmed on final output. Because some users may program tags to collect acceleration data at a higher Hz than magnetometer data (for example when looking for a Jerk signal), the DeadReckoning function automatically subsets the data set to those samples (rows) that do NOT have "NA" in the MagSurge column.

Betas is a [2,6] data frame with intercept in the first row and slope in the second row and can be directly taken from Standardize function output or user input. If you do not use the output of the Standardize function, the row names must be c("B0 Intercept","B1 Slope") and the column names must be c("MagSurge","MagHeave","MagSway","AccSurge","AccHeave","AccSway") though the order is not important. See the help on the Standardize function for more information.

Declination and inclination data can be found at various internet sites such as the World Magnetic Model 2010 Calculator from the British Geological Survey (see references).

RmL—please read Shepard et al. (2008) and perhaps Wilson et al. (2007) for information on an appropriate Running Mean Length for your data and study animal.

**SpdCalc**

=1 if Speed is supplied in rawdata at the same Hz as Acc and Mag Hz in m/s.

=2 if Speed is supplied in m/s but at Hz lower than Acc and Mag.

=3 if no speed is supplied and should be estimated from integrating (a running sum) the dynamic portion of the surge channel of acceleration and normalized using a linear model to meters/second from 0 to your max speed (MaxSpd) input.

=4 a constant speed is assumed which is entered as the MaxSpd.

=5 if no speed is supplied and should be estimated from accent and decent rates. ***Not yet implemented.***

=6 if speed is to be calculated from sound data. ***Not yet implemented***

### Value

A data.frame with columns DateTime, Xdim, Ydim, Depth and Speed. DateTime is the data and time columns pasted together using as.character. Xdim and Ydim are cartesian coordinates from the origin in meters. Depth has been linearly interpolated from initial Depth Hz to Magnetometer Hz and Speed is either that supplied by the user or calculated via one of the methods outlined above.
GapFinder

Author(s)
Brian Battaile

References

One possible source for declination and inclination data http://www.geomag.bgs.ac.uk/data_service/models_compass/wmm_calc.html


Examples

# A simple square example
betas<-Standardize(1,1,1,1,1,-1,-1,-1,1,1,1,-1,1,-1,1,-1,1)
data(square)
decinc<-c(0,65)
DRoutput<-DeadReckoning(square,betas,decinc,Hz=1,RmL=2,DepthHz=1,SpdCalc=1)
plot(DRoutput$Ydim,DRoutput$Xdim)

# Standardize tag output
betas<-Standardize(1,1,-1,1,1,-57.8,68.76,-61.8,64.2,-70,16,58.08,
# Declination and inclination data for study area
decinc<-c(0.125,65.918)
data set with 7 associated GPS fixes in the "gpsdata02" data set
data(rawdata)
DRoutput<-DeadReckoning(rawdata,betas,decinc,Hz=16,RmL=2,DepthHz=1,SpdCalc=3,MaxSpd=3.5)
plot(DRoutput$Ydim,DRoutput$Xdim)

GapFinder

Find gaps in your data file

Description

The function looks at the DateTime or Date and Time stamps of your data file and sees if any consecutive entries have a difference of greater than the timediff parameter.

Usage

GapFinder(rawdata, timediff = 1, timeformat = "%d-%b-%Y %H:%M:%S")
Arguments

- **rawdata**
  Data with a combined DateTime or separate Date and Time stamp columns labeled as such.

- **timediff**
  The amount of time in seconds that is expected between each entry (row). Typically this would be your data collection Hz (for Hz=16, timediff=1/16, for Hz=0.5, timediff=2) but results will depend on whether or not you have information on decimal seconds. One second is probably small enough for most people and will avoid issues with decimal seconds, but this depends on the scale that is important to you.

- **timeformat**
  Format of the DateTime or separate Date and Time stamp after it is pasted together with a space between. See the `strptime` function for codes. The default value is the value of the example files.

Value

Returns a data frame with 4 columns indicating the row that the gap occurs, the amount of time between the gap, and the DateTime stamp on either end of the gap. The data frame holds a maximum of 1000 entries, so if you have more gaps, then you can alter the program to create a larger dataframe. Hopefully you don’t have anywhere near that many gaps.

Author(s)

Brian Battaile

Examples

```r
data(rawdatagap)
gaps<-GapFinder(rawdatagap, timediff = 1, timeformat = "%d-%b-%Y %H:%M:%S")

#how timediff and a lack of decimal seconds information interact given a Hz of 16.
gaps<-GapFinder(rawdatagap, timediff = 1/16, timeformat = "%d-%b-%Y %H:%M:%S")
```

---

**GeoRef**

*Georeference the Dead Reckoning data*

Description

A wrapper function for the GeoReference function that georeferences a deadreckoning track with multiple GPS relocations.

Usage

```r
GeoRef(drdata, gpsfdata)
```
Arguments

- **drdata**: Data frame from, or similar in format as that produced by the DeadReckoning function.
- **gpsfdata**: Data frame from, or similar in format as that produced by the GPStable function.

Details

Data that is not bookended by the DateTime stamp of the GPS relocations will be discarded. The DateTime stamp of the drdata and gpsfdata must be in the same format, see the `strptime` function help file for details.

Value

Returns a data frame with DateTime, Distance, LatRad, LongRad, Latitude, Longitude, Depth, Speed, NewX, NewY, and Bering of the Georeferenced Deadreckoned tracks. NewX and NewY are cartesian coordinates in meters. Distance is the distance from the origin to the georeferenced point in 2D.

Author(s)

Brian Battaile

References


Examples

```r
#Standardize tag output
betas<-Standardize(1,1,-1,1,1,1,-57.8,68.76,-61.8,64.2,-70.16,58.08,
                   -10.1,9.55,-9.75,9.72, -9.91,9.43)
#get declination and inclination data for study area
decinc<-c(10.228,65.918)
#data set with 6 associated GPS fixes in the "gpsdata" data set
data(rawdata)
DRoutput<-DeadReckoning(rawdata,betas,decinc,Hz=16,RmL=2,DepthHz=1,SpdCalc=3,MaxSpd=3.5)
#prepare GPS data
data(gpsdata02)
gpsformat<-GPStable(gpsdata02)
Georeferenced<-GeoRef(DRoutput,gpsformat)
plot(Georeferenced$Longitude,Georeferenced$Latitude,pch=".")
points(gpsformat$Longitude[2],gpsformat$Latitude[2],pch="S",col="Red") #Start
points(gpsformat$Longitude[7],gpsformat$Latitude[7],pch="F",col="Blue") #Finish
#Intermediate GPS points
points(gpsformat$Longitude[3:6],gpsformat$Latitude[3:6],pch="x",col="Red")
```
**Description**

Georeferenced data from a fur seal track in the Bering Sea. Data has been thinned to relocations every 1 minute. This data set is used as an example in the vignette. `gpsdata01` is the complementary gps data for this pseudotrack.

**Usage**

data(georef1min01)

**Format**

A data frame with 6681 observations on the following 6 variables.

- **DateTime** a character vector
- **LatRad** a numeric vector, latitude given in radians
- **LongRad** a numeric vector, longitude given in radians
- **Latitude** a numeric vector, latitude given in decimal degrees
- **Longitude** a numeric vector, longitude given in decimal degrees
- **Depth** a numeric vector, in meters

**Details**

Date files `georef1min02`, `georef1min03`, `georef1min26` and `georef1min95` have the same structure

**Examples**

data(georef1min01)
head(georef1min01); tail(georef1min01)
str(georef1min01)
plot(georef1min01$Longitude, georef1min01$Latitude, pch=".")
data(gpsdata01)
points(gpsdata01$Longitude, gpsdata01$Latitude, col="red", pch="*")
Description

Georeferenced data from a fur seal track in the Bering Sea. Data has been thinned to relocations every 1 minute. This data set is used as an example in the vignette and the Mapper function. gpsdata02 is the complementary gps data for this pseudotrack.

Usage

data(georef1min02)

Format

A data frame with 9147 observations on the following 6 variables.

DateTime a character vector
LatRad a numeric vector, latitude given in radians
LongRad a numeric vector, longitude given in radians
Latitude a numeric vector, latitude given in decimal degrees
Longitude a numeric vector, longitude given in decimal degrees
Depth a numeric vector, in meters

Details

Date files georef1min01, georef1min03, georef1min26 and georef1min95 have the same structure

Examples

data(georef1min02)
head(georef1min02); tail(georef1min02)
str(georef1min02)
plot(georef1min02$Longitude, georef1min02$Latitude, pch=".")
data(gpsdata02)
points(gpsdata02$Longitude, gpsdata02$Latitude, col="red", pch="*")
Description

Georeferenced data from a fur seal track in the Bering Sea. Data has been thinned to relocations every 1 minute. This data set is used as an example in the vignette. gpsdata03 is the complementary gps data for this pseudotrack.

Usage

data(georef1min03)

Format

A data frame with 6724 observations on the following 6 variables.

DateTime a character vector
LatRad a numeric vector, latitude given in radians
LongRad a numeric vector, longitude given in radians
Latitude a numeric vector, latitude given in decimal degrees
Longitude a numeric vector, longitude given in decimal degrees
Depth a numeric vector, in meters

Details

Date files georef1min01, georef1min02, georef1min26 and georef1min95 have the same structure

Examples

data(georef1min03)
head(georef1min03); tail(georef1min03)
str(georef1min03)
plot(georef1min03$Longitude, georef1min03$Latitude, pch=".")
data(gpsdata03)
points(gpsdata03$Longitude, gpsdata03$Latitude, col="red", pch="*")
Description

Georeferenced data from a fur seal track in the Bering Sea. Data has been thinned to relocations every 1 minute. This data set is used as an example in the Mapper function. gpsdata26 is the complementary gps data for this pseudotrack.

Usage

data(georef1min26)

Format

A data frame with 430 observations on the following 6 variables.

DateTime  a character vector
LatRad    a numeric vector, latitude given in radians
LongRad   a numeric vector, longitude given in radians
Latitude  a numeric vector, latitude given in decimal degrees
Longitude a numeric vector, longitude given in decimal degrees
Depth     a numeric vector, in meters

Details

Date files georef1min01, georef1min02, georef1min03 and georef1min95 have the same structure

Examples

data(georef1min26)
head(georef1min26); tail(georef1min26)
str(georef1min26)
plot(georef1min26$Longitude, georef1min26$Latitude, pch=".")
data(gpsdata26)
points(gpsdata26$Longitude, gpsdata26$Latitude, col="red", pch="*")
Description

Georeferenced data from a fur seal track in the Bering Sea. Data has been thinned to relocations every 1 minute. This data set is used as an example in the Mapper function. gpsdata95 is the complementary gps data for this pseudotrack.

Usage

data(georef1min95)

Format

A data frame with 6566 observations on the following 6 variables.

DateTime  a character vector
LatRad    a numeric vector, latitude given in radians
LongRad   a numeric vector, longitude given in radians
Latitude  a numeric vector, latitude given in decimal degrees
Longitude a numeric vector, longitude given in decimal degrees
Depth     a numeric vector, in meters

Details

Date files georef1min01, georef1min02, georef1min03 and georef1min26 have the same structure

Examples

data(georef1min95)
head(georef1min95); tail(georef1min95)
str(georef1min95)
plot(georef1min95$Longitude, georef1min95$Latitude, pch=".")
data(gpsdata95)
points(gpsdata95$Longitude, gpsdata95$Latitude, col="red", pch="*")
Description

Takes relocation data and forces it to go through two known points (such as GPS fixes) at the beginning and end by rotating the track and either expanding or contracting each section by a constant percentage. The primary function used in the wrapper function GeoRef, you should use GeoRef if you have more than two GPS locations for your animal track.

Usage

GeoReference(drdata, gpsdata)

Arguments

drdata Data frame from, or in a similar format as that produced by the DeadReckoning function

gpsdata Data frame from, or in a similar format as that produced by the GPStable function.

Details

TimeDate stamp of first entry of drdata and gpsdata must be the same. The GeoRef function does this for you my matching the DateTime stamp in the drdata and gpsdata. The DateTime stamp of the drdata and gpsdata must be in the same format.

Value

Returns a data frame with DateTime, Distance, LatRad, LongRad, Latitude, Longitude, Depth, Speed, NewX, NewY, and Bearing of the Georeferenced Deadreckoning tracks. NewX and NewY are cartesian coordinates in meters. Distance is the distance from the origin to the new point in 2D.

Author(s)

Brian Battaile

References

# Standardize tag output

```r
betas <- Standardize(1, 1, -1, 1, 1, -57.8, 68.76, -61.8, 64.2, -70.16, 58.08,
                     -10.1, 9.55, -9.75, 9.72, -9.91, 9.43)
```

# get declination and inclination data for study area

deinc <- c(10.228, 65.918)

# data set with 7 associated GPS fixes in the "gpsdata02" data set

data(rawdata)

```r
DRoutput <- DeadReckoning(rawdata, betas, decinc, Hz=16, RmL=2, DepthHz=1, SpdCalc=3, MaxSpd=3.5)
```

# prepare GPS data

data(gpsdata02)

gpsformat <- GPStable(gpsdata02)

```r
Georeferenced <- GeoReference(DRoutput, gpsformat[, c(2, 3)], Hz=1)
```

```r
plot(Georeferenced$Longitude, Georeferenced$Latitude, pch=".")
```

```r
points(gpsformat$Longitude[2], gpsformat$Latitude[2], pch="S", col="Red")
```

```r
points(gpsformat$Longitude[3], gpsformat$Latitude[3], pch="F", col="Blue")
```

---

**gpsdata01**

*GPS raw data*

---

**Description**

Complete GPS file for northern fur seal track in the Bering Sea. Data taken by a wildlife computers mk10-F tag. georef1min01 is the complementary pseudotrack data for these GPS points.

**Usage**

data(gpsdata01)

**Format**

A data frame with 233 observations on the following 3 variables.

- **DateTime** a character vector
- **Latitude** a numeric vector
- **Longitude** a numeric vector

**Examples**

data(gpsdata01)

```r
head(gpsdata01); tail(gpsdata01)
```

```r
str(gpsdata01)
```

```r
plot(gpsdata01$Longitude, gpsdata01$Latitude)
```
gpsdata02

Description

Complete gps file for northern fur seal track in the Bering Sea. Data taken by a wildlife computers mk10-F tag. This data set is used as an example in the Mapper, GraphLimits, GeoRef, Splitter and GeoReference functions and in the vignette. georef1min02 is the complementary pseudotrack data for these gps points.

Usage

data(gpsdata02)

Format

A data frame with 276 observations on the following 3 variables.

DateTime  a character vector
Latitude  a numeric vector
Longitude  a numeric vector

Examples

data(gpsdata02)
head(gpsdata02); tail(gpsdata02)
str(gpsdata02)
plot(gpsdata02$Longitude, gpsdata02$Latitude)

---

gpsdata03

Description

Complete gps file for northern fur seal track in the Bering Sea. Data taken by a wildlife computers mk10-F tag. georef1min03 is the complementary pseudotrack data for these gps points.

Usage

data(gpsdata03)

Format

A data frame with 57 observations on the following 3 variables.

DateTime  a character vector
Latitude  a numeric vector
Longitude  a numeric vector
Examples

```r
data(gpsdata03)
head(gpsdata03); tail(gpsdata03)
str(gpsdata03)
pplot(gpsdata03$Longitude, gpsdata03$Latitude)
```
GPStable

Format

A data frame with 93 observations on the following 3 variables.

DateTime a character vector
Latitude a numeric vector
Longitude a numeric vector

Examples

data(gpsdata95)
head(gpsdata95);tail(gpsdata95)
str(gpsdata95)
plot(gpsdata95$Longitude,gpsdata95$Latitude)

GPStable Format GPS data

Description

Takes DateTime, Latitude and Longitude in decimal degrees and converts Latitude and Longitude into radians and calculates bearing and distance between consecutive locations

Usage

GPStable(rawdata)

Arguments

rawdata data frame with DateTime, or Date and Time separately, in same format as dead-reckoning data, Latitude and Longitude in decimal degrees. Must use those column labels.

Details

Data must be provided in decimal degrees (e.g. 162.546). Longitude values west of the prime meridian are 0 to -180. Latitude values south of the equator are 0 to -90. DateTime must be in the same format as output of DeadReckoning function or separated as Date and Time that can be pasted together to create the same format as output of the Deadreckoning function. Distance is calculated using the Spherical Law of Cosines (see references).

Value

Returns a data frame with columns DateTime, Latitude(Decimal Degrees), Longitude(Decimal Degrees), LatRad(Radians), LongRad(Radians), BearingRad, BearingDeg, DistanceKm. BearingRad and BearingDeg are the bearing calculated from point x to point x+1. DistanceKm is the distance between point x and point x-1.
GraphLimits

Determine Latitude and Longitude limits of a graph

Description

Finds the minimum and maximum of the longitude and latitude and then adds on 2.5% of the latitude and longitude to create boarders in the graph and accounts in a very simple way for projection distortion so long as maps are not to large.

Usage

GraphLimits(infile)

Arguments

infile A file with columns of latitude and longitude labeled as Latitude and Longitude

Details

Internal function used in the Mapper function. It attempts to account for the change in distance covered between longitude as latitude changes by using the cos(latitude) of the center of the latitude of the graph, and adjusting the longitude to cover the same distance. Hence, the boarders on the graph are set first by making the distances in latitude and longitude approximately equal in meters, then 2.5% is added on each end.

Value

Returns a list of 4 scalars defining the boarders of the data

Author(s)

Brian Battaile
Examples

#Standardize tag output
betas<-Standardize(1,1,-1,1,1,1,-57.8,68.76,-61.8,64.2,-70.16,58.08,-10.1,9.55,-9.75,9.72,
-9.91,9.43)
#get declination and inclination data for study area
decinc<-c(10.228,65.918)
#data set with 11 associated GPS fixes in the "gpsdata" data set
data(rawdata)
DRoutput<-DeadReckoning(rawdata ,betas, decinc, Hz = 16, RmL = 2, DepthHz = 1, SpdCalc=3,
MaxSpd=3.5)
#prepare GPS data
data(gpsdata02)
gpsformat<-GPStable(gpsdata02)
Georeferenced<-GeoReference(DRoutput,gpsformat[1:2,])
Limits<-GraphLimits(Georeferenced)

-------------------

Mapper

Animal track plotting with a color map

Description

Makes a color map of your animal track with terrain colors for land and shades of blue for depth

Usage

Mapper(inFile, gpsFile, bathyFile, ExpFact = 500, minlat = 51, maxlat = 60,
minlong = -177, maxlong = -163, Title = "Main")

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inFile</td>
<td>Track file of latitude and longitude columns with those names (output of GeoReference function)</td>
</tr>
<tr>
<td>gpsFile</td>
<td>GPS file of latitude and longitude columns with those names</td>
</tr>
<tr>
<td>bathyFile</td>
<td>Basemap file made from latitude, longitude and depth data, see ?bathymetry for format details</td>
</tr>
<tr>
<td>ExpFact</td>
<td>Expansion Factor- a multiplier of how smooth you want your new background graph to be. Larger numbers make it look smoother. Experiment with it.</td>
</tr>
<tr>
<td>minlat</td>
<td>minimum latitude of the data from the image.xyz file, MUST BE A WHOLE NUMBER</td>
</tr>
<tr>
<td>maxlat</td>
<td>maximum latitude of the data from the image.xyz file, MUST BE A WHOLE NUMBER</td>
</tr>
<tr>
<td>minlong</td>
<td>minimum longitude of the data from the image.xyz file, MUST BE A WHOLE NUMBER</td>
</tr>
<tr>
<td>maxlong</td>
<td>maximum longitude of the data from the image.xyz file, MUST BE A WHOLE NUMBER</td>
</tr>
<tr>
<td>Title</td>
<td>Title of your graph</td>
</tr>
</tbody>
</table>
Mapper Details

See https://www.gebco.net do download the raw bathymetric data to create the bathyFile.

Value

Given a bathymetric data file that is much larger than a given track, this function zooms the graph into where the data are and smooths out the background data if they are pixilated. The idea is to have a single bathymetry file to make many graphs for many animals no matter where they go so long as the tracks are all within the area of the bathymetry file limits. It creates a square graph that should adjust for the differences in actual distances between latitude and longitude graduands as latitude changes.

Author(s)

Brian Battaile

Examples

#A file from GeoRef function that has been thinned to data every 1 minute
data(georef1min26)
data(gpsdata26)
#See the vignette or ?bathymetry for information on how to get
#bathymetric data for your study area
data(bathymetry)
#This subset is just to save time creating the image.xyz
bathymetryBogs<-subset(bathymetry,Long<=(-168)& Long >=(-169)
& Lat<= 55 & Lat >=53,select=Long:Depth)
image.xyzBogs=tapply(bathymetryBogs$Depth, list(bathymetryBogs$Long, bathymetryBogs$Lat), unique)
Mapper(georef1min26, gpsdata26, image.xyzBogs, ExpFact = 500, minlat = 53,
maxlat = 55, minlong = -169, maxlong = -168, Title = "Fun Graph!!!")

## Not run:
#Plot the entire bathymetry file (this will take a minute or two)
image.xyz=tapply(bathymetry$Depth, list(bathymetry$Long, bathymetry$Lat), unique)
Bathymetry.palatte<-colorRampPalette(brewer.pal(9, "Blues"),bias=3)
#Plot the background map image
image.plot(image.xyz,
col=c(rev(Bathymetry.palatte(200)),terrain.colors(100),gray(0:20/20),
breaks=round(c(seq(from=min(image.xyz),to=0,length.out=201),seq(from=max(image.xyz)/101
, to=max(image.xyz),length.out=100))))
,smallplot=2 #plots legend off x axis
)
#The next examples plot only the part of the background map that contains the relocation data
#Example 1
data(georef1min02)
data(gpsdata02)
Mapper(georef1min02, gpsdata02, image.xyz, ExpFact = 500, minlat = 51, maxlat = 60,
minlong = -177, maxlong = -163, Title = "Fun Graph!!!")

#Example 2
bathymetryBogs2<-subset(bathymetry,Long<=(-168)& Long >=(-169.2)
& Lat<= 54.3 & Lat >=53.1, select=Long:Depth)
image.xyzBogs2=tapply(bathymetryBogs2$Depth, list(bathymetryBogs2$Long, bathymetryBogs2$Lat),
unique)
image.plot(image.xyzBogs2,
col=c(rev(Bathymetry.palatte(200)), terrain.colors(100), #gray(0:20/20),
breaks=round(c(seq(from=min(image.xyz),to=0,length.out=201),seq(from=max(image.xyz)/101,
to=max(image.xyz),length.out=100))))
#,smallplot=2 #plots legend off x axis
)
win.graph()
#Note the reduction in pixelation
data(georef1min95)
data(gpsdata95)
Mapper(georef1min95, gpsdata95, image.xyz, ExpFact = 500, minlat = 51, maxlat = 60,
minlong = -177, maxlong = -163, Title = "Fun Graph!!!")

## End(Not run)

---

**rawdata**  
*Raw triaxial magnetometer and accelerometer data*

**Description**

Raw data file from a northern fur seal with date, time, internal temp, depth, triaxial magnetometer, accelerometer, and wet/dry fields. This data set is bookended by the first 7 gps points in the gpsdata02 data set and is used in the examples for the DeadReckoning, GeoReference, GraphLimits, Splitter and GeoRef functions.

**Usage**

data(rawdata)

**Format**

A data frame with 133100 observations on the following 9 variables.

- **Date** a character vector
- **Time** a character vector
- **Depth** a numeric vector
- **MagSurge** a numeric vector
- **MagSway** a numeric vector
- **MagHeave** a numeric vector
- **AccSurge** a numeric vector
- **AccSway** a numeric vector
- **AccHeave** a numeric vector
Examples

data(rawdata)
str(rawdata)
head(rawdata);tail(rawdata)
plot(rawdata$AccHeave)

Description

Raw data file from a fur seal with a gap in the data file. Used as an example in the GapFinder function.

Usage

data(rawdatagap)

Format

A data frame with 13738 observations on the following 9 variables.

Date  a character vector
Time  a character vector
Depth  a numeric vector
MagSurge  a numeric vector
MagSway  a numeric vector
MagHeave  a numeric vector
AccSurge  a numeric vector
AccSway  a numeric vector
AccHeave  a numeric vector

Examples

data(rawdatagap)
head(rawdatagap);tail(rawdatagap)
str(rawdatagap)
plot(rawdatagap$AccHeave)
**Splitter**

_Splits large data files by date and time._

**Description**

This function takes the large data files inherently produced by accelerometer and magnetometer biologgers and splits them into smaller files so that the TrackReconstruction functions can handle them or it splits the data between trips or GPS locations or however is needed. The splitting is done via matching Time and Date as character strings.

**Usage**

\[\text{Splitter}(\text{TagFile}, \text{Begin}, \text{End}, \text{RmL}, \text{Hz})\]

**Arguments**

- **TagFile**: Matrix or data frame of magnetometer and accelerometer and other data collected by biologgers with column headings specified in Details.
- **Begin**: A vector of DateTime such as 2009-08-12 05:45:35.0625 indicating the time that the new files should begin. Format must be the same as the Date and Time data in the TagFile, though the TagFile Date and Time need not be in the same column.
- **End**: A vector of DateTime such as 2009-08-12 05:45:35.0625 indicating the time that the new files should end. Format must be the same as the Date and Time data in the TagFile, though the TagFile Date and Time need not be in the same column.
- **RmL**: Running mean length in seconds, required to calculate the amount of time beyond the Begin and End times that is required for the DeadReckoning function and is trimmed off by the DeadReckoning function.
- **Hz**: Frequency of Accelerometer data collection in Hz, required to calculate the amount of time beyond the Begin and End times that is required for the DeadReckoning function and is trimmed off by the DeadReckoning function.

**Details**

TagFile must have columns named DateTime or separated as Date and Time. Begin and End must be in the same format as the DateTime column in TagFile or have the same format as the Date and Time columns when they are pasted together by Splitter. For example, if DateTime is 2009-08-21 14:08:06.0625, then Begin and End cannot be Jul/21/2009 14:08:06.0625. See `strptime` for formatting date and time data. If TagFile has a Date column of 2009-08-21 and a Time column of 14:08:06.0625, Splitter will paste them together for you to look like 2009-08-21 14:08:06.0625. You must have enough time on the beginning and end of the TagFile equivalent to RmL * Hz/2 before the first DateTime in the Begin vector and after the last DateTime in the End vector. If your Hz is greater than 1 and you do not have data on fractions of a second, then each time stamp will have copies equal to your sampling Hz. In such a case, the program matches with the first instance of the DateTime and warnings will be given, this may be important when calculating RmL * Hz/2 tails.
Value

Creates a list of data frames in the format of the TagFile, but with a single DateTime column if the TagFiles had separate Date and Time columns.

Author(s)

Brian Battaile

Examples

# data set with 6 associated GPS fixes in the "gpsdata" data set
data(rawdata)
data(gpsdata02)
Begin=gpsdata02$DateTime[2:6]
End=gpsdata02$DateTime[3:7]
splits<-Splitter(rawdata,Begin,End,RmL=2,Hz=16)
## Not run:
# The following is code to write your many new files
setwd() # first fill in the path to the folder where you want the data to be written to
for(i in 1:length(splits))
{
  num<-i
  num=ifelse(num<10 & length(Begin)>10,paste("0",num,sep=""),num)
  num=ifelse(num<100 & length(Begin)>100,paste("0",num,sep=""),num)
  num=ifelse(num<1000 & length(Begin)>1000,paste("0",num,sep=""),num)
  num=ifelse(num<10000 & length(Begin)>10000,paste("0",num,sep=""),num)
  # Create a name for the file
  Nombre<-paste("Animal01Trip03GPS_Section",num,".txt",sep="")
  write.table(splits[[i]],Nombre,sep="\t",row.names=FALSE,quote=FALSE)
}
## End(Not run)

square

Raw triaxial magnetometer and accelerometer data

Description

Fabricated data file 100 observations long to make a square. Useful for exercises to understand how the DeadReckoning algorithm works.

Usage

data(rawdata)
Standardize

Format
A data frame with 100 observations on the following 10 variables.

Date  a character vector
Time  a character vector
Depth a numeric vector
MagSurge a numeric vector
MagSway a numeric vector
MagHeave a numeric vector
AccSurge a numeric vector
AccSway a numeric vector
AccHeave a numeric vector
Speed a numeric vector

Examples

data(square)
str(square)
head(square); tail(square)

<table>
<thead>
<tr>
<th>Standardize</th>
<th>Standardize accelerometer and magnetometer data</th>
</tr>
</thead>
</table>

Description
Calculates intercept and slope values with a linear model to standardize accelerometer and magnetometer data to values between -1 and +1.

Usage

Standardize(MagOrSR, MagOrHV, MagOrSW, AccOrSR, AccOrHV, AccOrSW, magSRmin, magSRmax, magHVmin, magHVmax, magSWmin, magSWmax, accSRmin, accSRmax, accHVmin, accHVmax, accSWmin, accSWmax)

Arguments

MagOrSR         Surge Magnetometer Orientation -1 or 1
MagOrHV         Heave Magnetometer Orientation -1 or 1
MagOrSW         Sway Magnetometer Orientation -1 or 1
AccOrSR         Surge Accelerometer Orientation -1 or 1
AccOrHV         Heave Accelerometer Orientation -1 or 1
AccOrSW         Sway Accelerometer Orientation -1 or 1
magSRmin        Surge Magnetometer minimum
magSRmax  Surge Magnetometer maximum
magHVmin  Heave Magnetometer minimum
magHVmax  Heave Magnetometer maximum
magSWmin  Sway Magnetometer minimum
magSWmax  Sway Magnetometer maximum
accSRmin  Surge Accelerometer minimum
accSRmax  Surge Accelerometer maximum
accHVmin  Heave Accelerometer minimum
accHVmax  Heave Accelerometer maximum
accSWmin  Sway Accelerometer minimum
accSWmax  Sway Accelerometer maximum

Details

Standardization/Calibration/normalization/relativizing of accelerometer and magnetometer data. The right-hand-rule indicates the orientation (polarity) of the magnetometers and accelerometers required for the pseudotrack reconstruction algorithm, so that when the front, top or left side of the tag is facing the earth, the accelerometers are at the maximal reading (+1) when the tag is not moving. A similar rule applies for the magnetometers except the maximal reading for each axis will be when the front, top or left side is facing north and at the angle of inclination of the magnetic field (the angle at which the magnetic field enters the earth) at that location on the planet. If the tag sensors conform to this rule all the orientation parameters should be 1, any sensors that are opposite to this should have -1 as an orientation value. Instead of labeling the tag in X,Y and Z dimensions, the directions are labeled as Surge, Heave and Sway where Surge indicates the front (anterior) to back (posterior) axis, the Heave is the top (dorsal) to bottom (ventral) axis and the Sway is the right to left (lateral) axis.

Value

Returns a [2,6] matrix with intercept in the first row and slope in the second row and columns in the order of the orientation parameter input. Row names are c("B0 Intercept","B1 Slope") and column names are c("MagSurge","MagHeave","MagSway","AccSurge","AccHeave","AccSway").

Author(s)

Brian Buttaile

Examples

betas<Standardize(1,1,-1,1,1,-57.8,68.76,-61.8,64.2,-70.16,58.08,-10.1,9.55,-9.75,9.72,-9.91,9.43)
betas
Index

* datasets
  bathymetry, 4
  georef1min01, 14
  georef1min02, 15
  georef1min03, 16
  georef1min26, 17
  georef1min95, 18
  gpsdata01, 20
  gpsdata02, 21
  gpsdata03, 21
  gpsdata26, 22
  gpsdata95, 22
  rawdata, 27
  rawdatagap, 28
  square, 30

* package
  TrackReconstruction-package, 2

bathymetry, 4

CalcBearing, 5
CalcDistance, 6
CalcLatitude, 7
CalcLongitude, 8

DeadReckoning, 9

GapFinder, 11
GeoRef, 12
georef1min01, 14
georef1min02, 15
georef1min03, 16
georef1min26, 17
georef1min95, 18
GeoReference, 19
gpsdata01, 20
gpsdata02, 21
gpsdata03, 21
gpsdata26, 22
gpsdata95, 22

GPStable, 23
GraphLimits, 24

Mapper, 25

rawdata, 27
rawdatagap, 28

Splitter, 29
square, 30
Standardize, 31

TrackReconstruction
  (TrackReconstruction-package), 2
TrackReconstruction-package, 2

33