Package 'moveWindSpeed'

March 20, 2019

Title Estimate Wind Speeds from Bird Trajectories

Version 0.2.3

Description Estimating wind speed from trajectories of individually tracked birds using a maximum likelihood approach.

Depends R (>= 3.0.0), methods, move

Suggests testthat, knitr, rmarkdown

License GPL

URL https://gitlab.com/bartk/moveWindSpeed

BugReports https://gitlab.com/bartk/moveWindSpeed/issues

LazyData true

RoxygenNote 6.1.0

Imports Rcpp

LinkingTo Rcpp

VignetteBuilder knitr

NeedsCompilation yes

Author Bart Kranstauber [aut, cre],
Rolf Weinzierl [aut]

Maintainer Bart Kranstauber <bart.kranstauber@ieu.uzh.ch>

Repository CRAN

Date/Publication 2019-03-20 14:00:03 UTC

R topics documented:

`estimatePhi` *estimatePhi*

Description

An function to estimate phi (the autocorrelation of speed) from data. This is done using iterative calls to the wind speed optimization on a selection of segments.

Usage

```
estimatePhi(data,
  isThermallingFunction = getDefaultIsThermallingFunction(360, 4),
  maxPointsToUseInEstimate = 20, phiInitialEstimate = 0,
  isGoodPoint = NULL, returnPointsUsedInEstimate = F,
  windowSize = 29, ...)
```

Arguments

`data`	An move object or stack.
`isThermallingFunction`	The thermalling function to use.
`maxPointsToUseInEstimate`	Maximal number of desired windows for phi estimation
`phiInitialEstimate`	Initial phi estimate
`isGoodPoint`	The points to use for phi estimation as logical or numeric, if NULL then findGoodPoints is used.
`returnPointsUsedInEstimate`	an logical value, if the segments used for phi estimation should also be returned.
`windowSize`	An window size, odd number or the start and end of the window relative to the focal point
`...`	extra arguments for getWindSpeedEstimates

Value

a list with phi and the log likelihood and the number of locations used

Examples

```
data(storks)
estimatePhi(
  storks[[2]],
  windowSize = 19,
  isSamplingRegular = 1,
  isThermallingFunction = getDefaultIsThermallingFunction(360, 4),
  maxPointsToUseInEstimate = 10
)
```

`findGoodPoints` *Function to find good points for estimation of phi*

Description

The function tries to find non overlapping windows for phi optimization.

Usage

```
findGoodPoints(data, maxPointsToUseInEstimate, phiInitialEstimate,
  windowSize, ...)
```

Arguments

`data`	An move object.
`maxPointsToUseInEstimate`	The number of desired windows.
`phiInitialEstimate`	The initial value used for the autocorrelation when calculating the wind speed for finding suitable windows.
`windowSize`	An odd number providing the window size
`...`	passed on to getWindEstimates

Value

a logical vector with the focal locations

Examples

```
data(storks)
which(findGoodPoints( storks[[2]],
windowSize = 29,  isSamplingRegular = 1,
isThermallingFunction = getDefaultIsThermallingFunction(360, 4),  maxPointsToUseInEstimate = 10,
phiInitialEstimate = 0  ))
```

`getDefaultIsThermallingFunction` *A function to generate an isThermallingFunction*

Description

A function to generate an isThermallingFunction

Usage

```
getDefaultIsThermallingFunction(totalAngle = 360, minMeanSpeed = NULL)
```

Arguments

`totalAngle`	the cumulative angle that is required to consider an trajectory thermalling
`minMeanSpeed`	the minimal air speed that is required to decide of a track is thermalling

Value

a function is returned that based on a series of headings returns a logical value to indicate is a track is thermalling or not

Examples

```
fun<-getDefaultIsThermallingFunction(170)
fun(1:160)
fun(1:190, rep(2,190))
fun<-getDefaultIsThermallingFunction(170, 3)
fun(1:190, rep(2,190))
fun(1:190, rep(3.4,190))
```

`getIsFocalPointFunction` *A function to generate isFocalPoint functions*

Description

A function to generate isFocalPoint functions

Usage

```
getIsFocalPointFunction(isFocalPoint)
```

Arguments

`isFocalPoint`	a function, a boolean array from which such a function can be built, or a list of indices

Value

a function which decides if wind estimation is performed for a point in the input data

---

`getIsSamplingRegularFunction` *A function to generate functions used to check if a segment is regular*

Description

A function to generate functions used to check if a segment is regular

Usage

```
getIsSamplingRegularFunction(isSamplingRegular)
```

Arguments

`isSamplingRegular`
: a function which decides if a sequence of timestamps is regular or the interval which is considered regular

Value

a function which decides if a sequence of timestamps is regular

Examples

```
fun<-getIsSamplingRegularFunction(10)
fun(Sys.time()+1:5)
fun(Sys.time()+c(0,10,20,30))
fun(Sys.time()+c(0,10,20,31))
```

`getTrackSegments` *An helper function to extract trajectory segments for wind estimation from a track*

Description

An helper function to extract trajectory segments for wind estimation from a track

Usage

```
getTrackSegments(data, timestamps, windowSize = 29,
  isFocalPoint = function(i, ts) {     TRUE }, isSamplingRegular = 1,
  focalSampleBefore = 0)
```

Arguments

data	A two column dataframe.
timestamps	A series of POSIXct timestamps as long as the data.
windowSize	The window size (odd number) or two numbers giving the start and end of a window around a focal point.
isFocalPoint	an function taking location numbers and timestamps that is used to see if a location should be considered as an focal point. It can for example be used to speed up calculations by only considering every second location. An numeric value can also be provided then only these locations are considered
isSamplingRegular	Either an numeric or a function that is used to decide if a series of timestamps is regular. If numeric than it should correspond to the interval in seconds.
focalSampleBefore	An argument to be used if data is not the start of the location count.

Value

A list of ground speeds

Examples

```
length(getTrackSegments(data.frame(1:40,1:40), Sys.time()+1:40))
length(getTrackSegments(data.frame(1:40,1:40), Sys.time()+c(1:25,36:50), windowSize=11))
str(getTrackSegments(data.frame(1:40,1:40), Sys.time()+1:40, windowSize=39))
```

---

getWindEstimate *Estimate wind speed from a sample of ground speeds*

Description

Estimate wind speed from a sample of ground speeds

Usage

```
getWindEstimate(groundSpeeds, phi, windStart = c(0, 0))

## S4 method for signature 'matrix,numeric'
getWindEstimate(groundSpeeds, phi,
  windStart = c(0, 0))
```

Arguments

groundSpeeds	matrix with two columns representing the ground speeds.
phi	numeric of length one giving the auto correlation.
windStart	numeric of length 2 giving the wind speed where to optimize from.

Value

an list with parameter estimates

Examples

```
s<-seq(0,2*pi, .1)
set.seed(34)
getWindEstimate(cbind(4*cos(s)+3+rnorm(length(s)), 4*sin(s)+2+rnorm(length(s))),0)
getWindEstimate(cbind(4*cos(s)+3+rnorm(length(s),sd=.2), 4*sin(s)+2+rnorm(length(s),sd=.2)),0)
```

---

getWindEstimates *Generate wind estimates for a trajectories or data frame with wind speeds*

Description

Generate wind estimates for a trajectories or data frame with wind speeds

Usage

```
getWindEstimates(data, timestamps, ...)

## S4 method for signature 'MoveStack,missing'
getWindEstimates(data, timestamps, ...)

## S4 method for signature 'Move,missing'
getWindEstimates(data, timestamps,
  groundSpeedXY = NULL, ...)

## S4 method for signature 'data.frame,POSIXct'
getWindEstimates(data, timestamps,
  windowSize = 29, isFocalPoint = function(i, ts) {     TRUE },
  isSamplingRegular = 1, focalSampleBefore = 0,
  returnSegmentList = F, referenceGroundSpeed = NULL, ...)

## S4 method for signature 'list,ANY'
getWindEstimates(data, timestamps, phi = 0,
  isThermallingFunction = getDefaultIsThermallingFunction(360, 4),
  columnNamesWind = c("estimationSuccessful", "residualVarAirspeed",
  "windX", "windY", "windVarX", "windVarY", "windCovarXY", "windVarMax",
  "airX", "airY"), referenceGroundSpeed = NULL, ...)
```

Arguments

data	Move object, MoveStack or data.frame containing wind speeds
timestamps	timestamps of the speed observations
...	other possible arguments currently nothing else is implemented

`groundSpeedXY` an character of length 2 containing column names from the move object that need to be used as the x and y component of the ground speed vector

`windowSize` a numeric vector of length 1 or 2, if length 1 it is the size of the focal window data will be assigned to the central location. If length 2 the window size is `sum(windowSize)+1)` and the first element is the number of location before the focal locations, the second is the number of locations after the focal location.

`isFocalPoint` an function that based on location number and timestamps returns a logical vector if location should be included. Or a numeric/logical vector indicating the location numbers.

`isSamplingRegular`
: either a function that determines based on a vector of timestamps if the sampling interval is regular or a numeric value that corresponds to the time interval between observations in the dataset that is regular

`focalSampleBefore`
: The number of locations that occurred before the move object fed in the getWindEstimates function, used in case stacks are provided for example. This is most cases not useful for users.

`returnSegmentList`
: a logical value indicating if the list of segments to estimate wind over should be returned instead of the estimates

`referenceGroundSpeed`
: a number indicating which of the grounds speed vectors to take as a reference for air speed, by default the 0th/middle location of the window if that is specified by one number.

`phi` the auto correlation of air speed.

`isThermallingFunction`
: An function that based on a series of headings and speeds (wind corrected) decides if an segment should be considered thermalling.

`columnNamesWind`
: The column names used for storing the data in the returned objected after it has been calculated.

Value

a Move object, dataframe or a MoveStack depending on input

Examples

```
data("storks")
# run example for reduced dataset
windEst<-getWindEstimates(storks[format(timestamps(storks),"%H")=="12",][[2:3]])
windEst<-spTransform(windEst, center=TRUE)
plot(windEst)
# only plot few arrows of estimates
s<-windEst$estimationSuccessful & format(timestamps(windEst), "%S")=='00'
# enlarge arrows 30 times
arrows(coordinates(windEst)[s,1],coordinates(windEst)[s,2],
   coordinates(windEst)[s,1]+ windEst$windX[s]*30,
   coordinates(windEst)[s,2]+windEst$windY[s]*30)
```

getWindowSizeLR *Generate arguments for window size around focal point*

Description

A function to translate an window size argument to a standardized argument.

Usage

```
getWindowSizeLR(windowSize)
```

Arguments

windowSize	a pair of positive integers determining the window size left and right of a focal point or an odd number determining the size of a symmetrical window

Value

windowSize a pair of positive integers determining the window size left and right of a focal point

storks *Example stork data.*

Description

A dataset containing location data of 6 juvenile storks (Ciconia ciconia) on the 18th of august when migration just started. On several occasion the birds use thermals.

Usage

```
storks
```

Format

A MoveStack consisting of 22333 locations

Source

http://www.movebank.org/

Examples

```
data("storks")
```

`windEstimLogLik` *Estimate the log likelihood*

Description

Estimate the log likelihood

Usage

```
windEstimLogLik(sigma, phi)
```

Arguments

`sigma`	the residual variance in airspeed
`phi`	the autocorrelation used in the calculations

Value

the log likelihood

Examples

```
windEstimLogLik(c(1.3,.6,1.5,1.8),.3)
windEstimLogLik(c(1.3,.6,1.5,1.8),.5)
```

Index