Package: FLightR (via r-universe)

October 20, 2026

Type Package

Title Reconstruct Animal Paths from Solar Geolocation Loggers Data

Version 0.5.6.900

Date xxx

Description Spatio-temporal locations of an animal are computed from annotated data with a hidden Markov model via particle filter algorithm. The package is relatively robust to varying degrees of shading. The hidden Markov model is described in Movement Ecology - Rakhimberdiev et al. (2015) <doi:10.1186/s40462-015-0062-5>, general package description is in the Methods in Ecology and Evolution - Rakhimberdiev et al. (2017) <doi:10.1111/2041-210X.12765> and package accuracy assessed in the Journal of Avian Biology - Rakhimberdiev et al. (2016) <doi:10.1111/jav.00891>.

URL https://CRAN.R-project.org/package=FLightR

BugReports https://github.com/eldarrak/FLightR/issues

Depends R (>= 4.1.0)

Imports bit, geosphere, ggmap, ggplot2, CircStats, circular, fields, maps, mgcv, nlme, parallel, RcppArmadillo, sf, suntools, truncnorm

License GPL-3

ByteCompile true

Encoding UTF-8

Suggests testthat, knitr, rmarkdown, yaml

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libicu-dev libjpeg-dev libpng-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

Repository https://eldarrak.r-universe.dev

Date/Publication 2026-06-22 12:11:43 UTC

RemoteUrl https://github.com/eldarrak/flightr

RemoteRef HEAD

RemoteSha 83c77ec9a52e92a789a32ee8ea04ce0cdb2d468c

Contents

BAStag2TAGS . . . 2
find.stationary.location . . . 3
find.times.distribution . . . 5
FLightR2Movebank . . . 5
GeoLight2TAGS . . . 6
get.tags.data . . . 7
get_ZI_distances . . . 9
make.calibration . . . 10
make.grid . . . 11
make.prerun.object . . . 12
map.FLightR.ggmap . . . 14
match_gps_to_twilights . . . 16
plot_likelihood . . . 17
plot_lon_lat . . . 18
plot_slopes_by_location . . . 19
plot_util_distr . . . 20
posterior_point_distribution . . . 22
read_gps_track . . . 23
read_validation_config . . . 23
run.particle.filter . . . 24
stationary.migration.summary . . . 27
twGeos2TAGS . . . 28
validation_summary . . . 29

Index **30**

`BAStag2TAGS` *Function to write down twilights annotated in BAStag package data in so-called TAGS format*

Description

this function converts combines twilights detected in BAStag with raw data and writes them down in TAGS format that can be easily read by `get.tags.data`

Usage

```
BAStag2TAGS(raw, twl, threshold, filename = NULL)
```

Arguments

raw	original data - dataframe with two columns first column must contain time and second measured light levels
twl	twilights object from preprocess.light function
threshold	threshold value used for twilight definition in preprocess.light
filename	if NULL data.frame in TAGS format will be returned otherwise .csv file in TAGS format will be written

Details

TAGS format returned or written as .csv by this function is a dataframe with columns

datetime date and time in ISO 8601 format e.g. 2013-06-16T00:00:11.000Z

light light value measured by tag

twilight assigned by the software numeric indication of whether the record belongs to sunrise (1), sunset (2) or none of those (0)

excluded indication of whether a twilight was excluded during manual inspection (logical, TRUE | FALSE)

interp indication of whether the light value at twilight was interpolated (logical, TRUE | FALSE)

The fields excluded and interp may have values of TRUE only for twilight > 0.

Value

NULL if filename is provided or TAGS formatted dataframe.

Author(s)

Eldar Rakhimberdiev & Simeon Lisovski

See Also

twGeos2TAGS and GeoLight2TAGS

---

find.stationary.location

find unknown calibration location

Description

Functions attempts to find a location where The function attempts to find a location for a time period assuming animal was not moving. Does not work well will shaded data!

Usage

```
find.stationary.location(
  Proc.data,
  calibration.start,
  calibration.stop,
  plot = TRUE,
  initial.coords = NULL,
  print.optimization = TRUE,
  reltol = 1e-04
)
```

Arguments

`Proc.data`	processed data object generated by `get.tags.data`
`calibration.start`	POSIXct time when stationary period started
`calibration.stop`	POSIXct time when stationary period ended
`plot`	plots every iteration
`initial.coords`	location vector with initial values for location (longitude and latitude). Should be close (+-2000 km from the real location)
`print.optimization`	do you want every optimization iteration to be printed? If TRUE - Lon, Lat, calibration mean and calibration sd are being printed. Optimization tries to minimize the latter.
`reltol`	tolerance for optimization, see `stats::optim()` for more details

Details

The idea behind the function is that it tries to minimize variance between slopes for the whole period by optimizing location. It can be seen as an extension of Hill-Ekstrom calibration idea.

Value

vector with coordinates - longitude and latitude.

Author(s)

Eldar Rakhimberdiev

Examples

```
#this example takes about 15 minutes to run

File<-system.file("extdata", "Godwit_TAGS_format.csv", package = "FLightR")
Proc.data<-get.tags.data(File)
plot_slopes_by_location(Proc.data=Proc.data, location=c(5.43, 52.93))
abline(v=as.POSIXct("2013-08-20", tz='GMT')) # end of first calibration period
```

```
abline(v=as.POSIXct("2014-05-05", tz='GMT')) # start of the second calibration period
Location<-find.stationary.location(Proc.data, '2013-07-20', '2013-08-20', initial.coords=c(10, 50))
```

`find.times.distribution` *extracts times of arrival and departure to/from spatial extent*

Description

Idea of this functions is to extract schedules for known location

Usage

```
find.times.distribution(Result, Spatial.Index)
```

Arguments

`Result`	FLightR result object obtained from `run.particle.filter`
`Spatial.Index`	Row numbers for spatial grid (`Result$Spatial$Grid`) to estimate schedules for.

Value

dataframe with columns for 0.025, 0.25, 0.5, 0.75, 0.975 probability of line crossing and rows for every crossing.

Author(s)

Eldar Rakhimberdiev

`FLightR2Movebank` *Summary of estimated locations for Movebank*

Description

Summarize result object in the format required for upload to Movebank

Usage

```
FLightR2Movebank(Result, alpha = 0.5, filename = NULL)
```

Arguments

Result	FLightR result object obtained from run.particle.filter
alpha	coverage of the credible intervals for now only two options: 0.95 or 0.5.
filename	if NULL data.frame in TAGS format will be returned otherwise .csv file in TAGS format will be written

Details

This function accepts FLightR results object.

Value

NULL if filename is provided or Movebank formatted dataframe.

Author(s)

Eldar Rakhimberdiev

GeoLight2TAGS *Function to write down twilights annotated in GeoLight package data in so-called TAGS format*

Description

this function converts combines twilights detected in BAStag to twGeos with raw data and writes them down in TAGS format that can be easily read by get.tags.data

Usage

```
GeoLight2TAGS(raw, gl_twl, threshold, filename = NULL)
```

Arguments

raw	original data - dataframe with two columns first column must contain time and second measured light levels
gl_twl	twilights object from GeoLight
threshold	threshold value used for twilight definition in GeoLight
filename	if NULL data.frame in TAGS format will be returned otherwise .csv file in TAGS format will be written

Details

TAGS format returned or written as .csv by this function is a dataframe with columns

`datetime` date and time in ISO 8601 format e.g. 2013-06-16T00:00:11.000Z

`light` light value measured by tag

`twilight` assigned by the software numeric indication of whether the record belongs to sunrise (1), sunset (2) or none of those (0)

`excluded` indication of whether a twilight was excluded during manual inspection (logical, `TRUE | FALSE`)

`interp` indication of whether the light value at twilight was interpolated (logical, `TRUE | FALSE`)

The fields `excluded` and `interp` may have values of `TRUE` only for `twilight > 0`.

Value

`NULL` if `filename` is provided or TAGS formatted dataframe.

Author(s)

Eldar Rakhimberdiev & Simeon Lisovski

See Also

`twGeos2TAGS` and `BAStag2TAGS`

`get.tags.data` *read TAGS formatted data*

Description

Reads the data frame with detected twilight events into the FLightR

Usage

```
get.tags.data(
  filename = NULL,
  start.date = NULL,
  end.date = NULL,
  log.light.borders = "auto",
  log.irrad.borders = "auto",
  saves = c("auto", "max", "mean"),
  measurement.period = NULL,
  impute.on.boundaries = FALSE,
  tag.type = "auto",
  light.scale = "auto",
  logger.mode = NULL,
  raw.light.bounds = NULL,
  tag.settings = NULL
)
```

Arguments

filename	the name of the file which the data are to be read from. File is supposed to be comma separated file of TAGS format. If it does not contain an absolute path, the file name is relative to the current working directory, getwd(). Tilde-expansion is performed where supported. This can be a compressed file. Alternatively, file can be a readable text-mode connection (which will be opened for reading if necessary, and if so closed (and hence destroyed) at the end of the function call). File can also be a complete URL.
start.date	date of beginning of relevant data collection in POSIXct format.
end.date	date of end of relevant data collection in POSIXct format.
log.light.borders	Numeric vector with length of 2 for minimum and maximum log(light) levels to use. Alternatively character value 'auto', that will allow FLightR to assign these values according to detected tag type.
log.irrad.borders	Numeric vector with length of 2 for minimum and maximum log(irradiance) values to use. Alternatively character value 'auto', that will allow FLightR to assign these values according to detected tag type.
saves	character values informing FLightR if min or max values were used by logger.
measurement.period	Value in seconds defining how often tag was measuring light levels. If NULL value will be taken from known values for detected tag type.
impute.on.boundaries	logical, if FLightR should approximate values at boundaries. Set it to TRUE only if you have vary few active points at each twilight, e.g if tag was saving every 10 minutes or so.
tag.type	Logger type. The default, "auto", keeps the historical automatic detection from light minimum and maximum values. Explicit values such as "mk", "intigeo", "intigeo_mode_1", "gdl1", "gdl2v1", "gdl2v2_gdlpam", and "lat2000" bypass automatic tag-type detection.
light.scale	Scale of the light values in the TAGS file. Use "auto" for historical automatic detection, "raw" for untransformed positive light values, or "log" when the file already contains log-light values. If "log" is supplied, FLightR converts values back to raw light internally so downstream processing does not log-transform them twice.
logger.mode	Optional convenience alias combining tag.type and light.scale, e.g. "mk-real", "mk-raw", "mk-log", "intigeo-real", or "intigeo-log". If supplied, it overrides tag.type and light.scale.
raw.light.bounds	Optional raw light-value bounds for a custom logger. These are stored in Proc.data$tag.settings and Proc.data$Metadata$Import for diagnostics.
tag.settings	Optional list with any of tag.type, light.scale, logger.mode, or raw.light.bounds. Values in this list override the corresponding arguments.

Details

The returned object has many parts, the important are: (1) the recorded light data, (2) the detected twilight events, (3) light level data at the moment of each determined sunrise and sunset and around them (24 fixes before and 24 after), and (4) technical parameters of the tag, i. e. its type, saving and measuring period (the periodicity, in seconds, at which a tag measures and saves data). Import settings are stored both in the historical top-level fields, such as `tagtype`, `log_transformed`, `log.light.borders`, and `log.irrad.borders`, and in `Proc.data$Metadata$Import` for structured provenance.

Value

list, which is to be further processed with the FLightR.

Examples

```
File<-system.file("extdata", "Godwit_TAGS_format.csv", package = "FLightR")
Proc.data<-get.tags.data(File)
Proc.data<-get.tags.data(File, tag.type = "intigeo", light.scale = "log")
```

`get_ZI_distances` *Estimate distances moved between twilights*

Description

This function estimate distances with all zeros from stationary periods. This means many of the resulting movements will have 0 as the distance

Usage

```
get_ZI_distances(Result)
```

Arguments

`Result`	An object created by `run.particle.filter`.

Value

a data frame containing median and quartiles for the distances and also departure and arrival time

Author(s)

Eldar Rakhimberdiev

`make.calibration` *Creates a calibration object, further used for calculation of coordinates in the* `run.particle.filter`.

Description

Function estimates all necessary parameters from the calibration data logged in a known location or locations.

Usage

```
make.calibration(
  Proc.data,
  Calibration.periods,
  model.ageing = FALSE,
  plot.each = FALSE,
  plot.final = FALSE,
  likelihood.correction = "auto",
  fixed.logSlope = c(NA, NA),
  suggest.irrad.borders = FALSE,
  return.slopes = FALSE
)
```

Arguments

`Proc.data`	processed data object generated by `get.tags.data`
`Calibration.periods`	a data frame containing start and end dates of all the calibration periods (POSIXct) and geographic coordinates of the corresponding calibration locations.
`model.ageing`	if set to TRUE, accounts for the tag ageing (with opacification of its transparent shell of a light sensor), resulting into decreasing sensitivity of the device. This option is useful only if there were several calibration periods or if calibration period was very long (~ longer than a month).
`plot.each`	Do you want every twilight to be plotted while processing
`plot.final`	Do you want final calibration graph to be plotted. On the graph you can see all the observed versus expected light levels. All slopes should be similar.
`likelihood.correction`	will estimate correction of likelihood for the current calibration parameters. Highly recommended not to be change from 'auto'. In this case FLightR will switch it to FALSE in case tag saved data on 10 minutes or longer period.
`fixed.logSlope`	these are mean (1) and SD (2) for distribution of slopes. Should normally be estimated from the data (and thus default is c(NA, NA)). Change any of these two finite values if you want them to be predetermined and not estimated from the calibration data.

`suggest.irrad.borders`
: experimental parameter! If set to TRUE function will try to find the best values for the log.irrad.borders

`return.slopes`
: if true function will return estimated individual twilight slopes.

Value

calibration object to be uses in the `make.prerun.object`

Author(s)

Eldar Rakhimberdiev

Examples

```
File<-system.file("extdata", "Godwit_TAGS_format.csv", package = "FLightR")
Proc.data<-get.tags.data(File, end.date=as.POSIXct('2013-08-20', tz='GMT'))
Calibration.periods<-data.frame(
       calibration.start=NA,
       calibration.stop=as.POSIXct("2013-08-20", tz='GMT'),
       lon=5.43, lat=52.93)
     #use c() also for the geographic coordinates, if you have more than one calibration location
       # (e. g.,  lon=c(5.43, 6.00), lat=c(52.93,52.94))
print(Calibration.periods)

# NB Below likelihood.correction is set to FALSE for fast run!
# Leave it as default 'auto' for real examples
Calibration<-make.calibration(Proc.data, Calibration.periods, likelihood.correction=FALSE)
```

---

make.grid *makes spatial grid*

Description

This function makes a rectangular grid with use defined boundaries and probabilities of being stationary.

Usage

```
make.grid(
  left = -180,
  bottom = -90,
  right = 180,
  top = 90,
  distance.from.land.allowed.to.use = c(-Inf, Inf),
  distance.from.land.allowed.to.stay = c(-Inf, Inf),
  plot = TRUE,
  return.distances = FALSE,
```

```
  probability.of.staying = 0.5
)
```

Arguments

`left`	- left boundary in degrees (-180 <= left <= 180)
`bottom`	- lower boundary in degrees (-90 <= bottom <= 90)
`right`	- right boundary in degrees (-180 <= right <= 180)
`top`	- top boundary in degrees (-90 <= right <= 90)
`distance.from.land.allowed.to.use`	- define how far from the shore animal could occur. Unit - km, negative values are for inland and positive for offshore directions. `Inf` stays for infinity
`distance.from.land.allowed.to.stay`	- define how far from the shore animal could stay stationary between twilights. Unit - km, negative values are for inland and positive for offshore directions. `Inf` stays for infinity
`plot`	show a plot of final grid.
`return.distances`	- return distances to the shoreline
`probability.of.staying`	- assigned probability value for grid cells that do not satisfy `distance.from.water.allowed.to.stay`

Value

dataframe with coordinates(lon and lat) and `probability.of.staying`

Author(s)

Eldar Rakhimberdiev

Examples

```
Grid<-make.grid(left=-14, bottom=30, right=13, top=57,
  distance.from.land.allowed.to.use=c(-Inf, Inf),
  distance.from.land.allowed.to.stay=c(-Inf, Inf))
```

`make.prerun.object` *combines data, calibration and sets up priors*

Description

This function is one step before `run.particle.filter`. It combines data, calibration, spatial extent and movement priors and estimates spatial likelihoods that used later in the particle filter.

Usage

```
make.prerun.object(
  Proc.data,
  Grid,
  start,
  end = start,
  Calibration,
  threads = -1,
  Decision = 0.05,
  Direction = 0,
  Kappa = 0,
  M.mean = 300,
  M.sd = 500,
  likelihood.correction = TRUE
)
```

Arguments

`Proc.data`	Processed data object created by `get.tags.data`.
`Grid`	Spatial grid created by `make.grid`.
`start`	release location (lat, lon).
`end`	end of the track location. Will use `start` by default. Use NA in case of unknown end point.
`Calibration`	Calibration object created by `make.calibration`.
`threads`	number of parallel threads to use. default is -1, which means FLightR will use all available threads except 1. Value 1 will force sequential evaluation
`Decision`	prior for migration probability values from 0 to 1 are allowed
`Direction`	Direction prior for direction of migration (in degrees) with 0 pointing to the North
`Kappa`	concentration parameter for vonMises distribution, 0 means uniform or even distribution. Will set some prior for direction for all the track, so is not recommended to be changed
`M.mean`	Prior for mean distance travelled between consecutive twilights, km
`M.sd`	Prior for sd of distance travelled between consecutive twilights, the higher the value is the wider is the the distribution
`likelihood.correction`	Should likelihood correction estimated during `make.calibration` run be used?

Value

Object to be uses in the `run.particle.filter`

Author(s)

Eldar Rakhimberdiev

Examples

```
File<-system.file("extdata", "Godwit_TAGS_format.csv", package = "FLightR")
# to run example fast we will cut the real data file by 2013 Aug 20
Proc.data<-get.tags.data(File, end.date=as.POSIXct('2013-07-02', tz='GMT'))
Calibration.periods<-data.frame(
       calibration.start=NA,
       calibration.stop=as.POSIXct("2013-08-20", tz='GMT'),
       lon=5.43, lat=52.93)
     #use c() also for the geographic coordinates, if you have more than one calibration location
       # (e. g.,  lon=c(5.43, 6.00), lat=c(52.93,52.94))
print(Calibration.periods)

# NB Below likelihood.correction is set to FALSE for fast run!
# Leave it as default TRUE for real examples
Calibration<-make.calibration(Proc.data, Calibration.periods, likelihood.correction=FALSE)

Grid<-make.grid(left=0, bottom=50, right=10, top=56,
  distance.from.land.allowed.to.use=c(-Inf, Inf),
  distance.from.land.allowed.to.stay=c(-Inf, Inf))

all.in<-make.prerun.object(Proc.data, Grid, start=c(5.43, 52.93),
                           Calibration=Calibration, threads=2)
```

`map.FLightR.ggmap` *plots result over map*

Description

plots track over map with probability cloud. Can plot only part of the track if dates are specified. Note that you can use it only after obtaining and registering in you current session Google Api Key. For details on the API key, see https://ornithologyexchange.org/forums/topic/38315-mapflightrggmap-error/.

Usage

```
map.FLightR.ggmap(
  Result,
  dates = NULL,
  plot.cloud = TRUE,
  map.options = NULL,
  plot.options = NULL,
  save.options = NULL,
  zoom = "auto",
  return.ggobj = FALSE,
  seasonal.colors = TRUE,
  seasonal.donut.location = "topleft",
  seasonal.donut.proportion = 0.5,
```

```
  save = TRUE
)
```

Arguments

Result	FLightR result object obtained from run.particle.filter
dates	either NULL if all twilights should be included or data.frame with first column - start of the period and second end of the period. Each line represents a new period
plot.cloud	Should probability cloud be plotted? If TRUE cloud is estimated by ggplot2::stat_density2d().
map.options	options passed to ggmap::get_map(), note that zoom option is defined separately
plot.options	plotting options. Not defined yet!
save.options	options passed to ggplot2::ggsave(). Filename should be defined here.
zoom	Zoom for map. If 'auto' FLightR will try to find optimal zoom level by downloading different size maps and checking whether all the points fit the map.
return.ggobj	Should ggobj be returned for subsequent checks and/or replotting
seasonal.colors	if true points of the track will have seasonal colors
seasonal.donut.location	if NULL - no color wheel placed, otherwise select one of 'bottomleft', 'bottomright', 'topleft'
seasonal.donut.proportion	how much of X axis should color wheel occupy. return either NULL or ggplot2 class object
save	should function save results with ggplot2::ggsave()?

Value

if 'return.ggobj=TRUE' return ggplot object otherwise returns 'NULL'.

Author(s)

Eldar Rakhimberdiev

Examples

```
File<-system.file("extdata", "Godwit_TAGS_format.csv", package = "FLightR")
# to run example fast we will cut the real data file by 2013 Aug 20
Proc.data<-get.tags.data(File, end.date=as.POSIXct('2013-06-25', tz='GMT'))
Calibration.periods<-data.frame(
       calibration.start=as.POSIXct(c(NA, "2014-05-05"), tz='GMT'),
       calibration.stop=as.POSIXct(c("2013-08-20", NA), tz='GMT'),
       lon=5.43, lat=52.93)
    #use c() also for the geographic coordinates, if you have more than one calibration location
       # (e. g.,  lon=c(5.43, 6.00), lat=c(52.93,52.94))
```

```
# NB Below likelihood.correction is set to FALSE for fast run!
# Leave it as default TRUE for real examples
Calibration<-make.calibration(Proc.data, Calibration.periods, likelihood.correction=FALSE)

Grid<-make.grid(left=0, bottom=50, right=10, top=56,
  distance.from.land.allowed.to.use=c(-Inf, Inf),
  distance.from.land.allowed.to.stay=c(-Inf, Inf))

all.in<-make.prerun.object(Proc.data, Grid, start=c(5.43, 52.93),
                           Calibration=Calibration, threads=2)
# here we will run only 1e4 partilces for a very short track.
# One should use 1e6 particles for the full run
Result<-run.particle.filter(all.in, threads=1,
           nParticles=1e3, known.last=TRUE,
           precision.sd=25, check.outliers=FALSE)

## Not run:
map.FLightR.ggmap(Result, seasonal.donut.location=NULL, zoom=6, save=FALSE)

## End(Not run)
# for this short track without variance seasonal donut does not work,
# but for normall track it will.
```

`match_gps_to_twilights` *Match GPS fixes to FLightR twilight times*

Description

Match GPS fixes to FLightR twilight times

Usage

```
match_gps_to_twilights(Result, gps, max_time_diff_hours = 6)
```

Arguments

`Result`	A FLightR Result-like object containing twilight times.
`gps`	A data.frame with time, lon, and lat columns.
`max_time_diff_hours`	Maximum allowed absolute time difference.

Value

A data.frame of matched twilight and GPS records.

plot_likelihood *plot likelihood surface over map*

Description

plots specific likelihood surface over map

Usage

```
plot_likelihood(object, date = NULL, twilight.index = NULL)
```

Arguments

object	either output from make.prerun.object or run.particle.filter
date	either NULL or a date (possibly with time) closest to the twilight you wan to be plotted
twilight.index	number of likelihood surface to be plotted

Details

function plots likelihoods before particle filter run, so these are pure results of calibrations without any movement model

Value

'NULL'

Author(s)

Eldar Rakhimberdiev

Examples

```
File<-system.file("extdata", "Godwit_TAGS_format.csv", package = "FLightR")
# to run example fast we will cut the real data file by 2013 Aug 20
Proc.data<-get.tags.data(File, end.date=as.POSIXct('2013-07-02', tz='GMT'))
Calibration.periods<-data.frame(
       calibration.start=as.POSIXct(c(NA, "2014-05-05"), tz='GMT'),
       calibration.stop=as.POSIXct(c("2013-08-20", NA), tz='GMT'),
       lon=5.43, lat=52.93)
     #use c() also for the geographic coordinates, if you have more than one calibration location
       # (e. g.,  lon=c(5.43, 6.00), lat=c(52.93,52.94))

# NB Below likelihood.correction is set to FALSE for fast run!
# Leave it as default TRUE for real examples
Calibration<-make.calibration(Proc.data, Calibration.periods, likelihood.correction=FALSE)

Grid<-make.grid(left=0, bottom=50, right=10, top=56,
```

```
  distance.from.land.allowed.to.use=c(-Inf, Inf),
  distance.from.land.allowed.to.stay=c(-Inf, Inf))

all.in<-make.prerun.object(Proc.data, Grid, start=c(5.43, 52.93),
                           Calibration=Calibration, threads=2)
plot_likelihood(all.in, twilight.index=10)
```

`plot_lon_lat` *plots result by longitude and latitude*

Description

This function plots result by latitude and longitude in either vertical or horizontal layout.

Usage

```
plot_lon_lat(Result, scheme = c("vertical", "horizontal"))
```

Arguments

`Result`	FLightR result object obtained from `run.particle.filter`
`scheme`	either 'vertical' or 'horizontal' layouts

Value

'NULL'

Author(s)

Eldar Rakhimberdiev

Examples

```
File<-system.file("extdata", "Godwit_TAGS_format.csv", package = "FLightR")
# to run example fast we will cut the real data file by 2013 Aug 20
Proc.data<-get.tags.data(File, end.date=as.POSIXct('2013-07-02', tz='GMT'))
Calibration.periods<-data.frame(
       calibration.start=as.POSIXct(c(NA, "2014-05-05"), tz='GMT'),
       calibration.stop=as.POSIXct(c("2013-08-20", NA), tz='GMT'),
       lon=5.43, lat=52.93)
     #use c() also for the geographic coordinates, if you have more than one calibration location
       # (e. g.,  lon=c(5.43, 6.00), lat=c(52.93,52.94))

# NB Below likelihood.correction is set to FALSE for fast run!
# Leave it as default TRUE for real examples
Calibration<-make.calibration(Proc.data, Calibration.periods, likelihood.correction=FALSE)

Grid<-make.grid(left=0, bottom=50, right=10, top=56,
```

```
  distance.from.land.allowed.to.use=c(-Inf, Inf),
  distance.from.land.allowed.to.stay=c(-Inf, Inf))

all.in<-make.prerun.object(Proc.data, Grid, start=c(5.43, 52.93),
                           Calibration=Calibration, threads=2)
# here we will run only 1e4 partilces for a very short track.
# One should use 1e6 particles for the full run
Result<-run.particle.filter(all.in, threads=1,
          nParticles=1e3, known.last=TRUE,
          precision.sd=25, check.outliers=FALSE)

plot_lon_lat(Result)
```

`plot_slopes_by_location` *plots log of observed versus expected slope by time for a known location*

Description

The function calculates and plots calibration slopes for sunsets and sunrises for every day of the tracking period, based on the assumption that the tag remained in the same (calibration) location all the time.

Usage

```
plot_slopes_by_location(
  Proc.data,
  location,
  log.light.borders = "auto",
  log.irrad.borders = "auto",
  ylim = NULL,
  xlim = NULL
)
```

Arguments

`Proc.data`	processed data object generated by `get.tags.data`
`location`	vector with longitude and latitude of calibration location (degrees).
`log.light.borders`	numeric vector with length of 2 for minimum and maximum log(light) levels to use. Default value 'auto', will take these values from the Proc.data object.
`log.irrad.borders`	numeric vector with length of 2 for minimum and maximum log(irradiance) values to use. Default value 'auto', will take these values from the Proc.data object.

ylim	the y limits of the plot. The default value, NULL, indicates that the range of the finite values to be plotted should be used.
xlim	the x limits of the plot. The default value, NULL, otherwise can be POSIXct or character in a form readable by as.POSIXct().

Details

The plot of calibration slopes is used for finding start and end dates of a calibration period (the time period, during which the tag remained in the calibration location with coordinates (x,y)). During the calibration period, the calibration slopes vary little both, between the twilight events (sunrises and sunsets) and in time. When the tag changes location, the slopes for sunrises and sunsets start to deviate. There may potentially be several calibration periods for the same location (if the bird returned to the same location several times). The boundaries (start and end dates) of each of these periods are captured visually. If there were more than one calibration location, the procedure is repeated, once for each location. All the obtained calibration periods can be entered in a data frame 'Calibration.periods', for further analysis. Each line of the data frame contains start and end dates (if applicable) of the calibration period and geographic coordinates of the location.

Value

'NULL'

Author(s)

Eldar Rakhimberdiev

Examples

```
File<-system.file("extdata", "Godwit_TAGS_format.csv", package = "FLightR")
Proc.data<-get.tags.data(File)
plot_slopes_by_location(Proc.data=Proc.data, location=c(5.43, 52.93))
abline(v=as.POSIXct("2013-08-20", tz='GMT')) # end of first calibration period
abline(v=as.POSIXct("2014-05-05", tz='GMT')) # start of the second calibration period
```

---

plot_util_distr *plots resulting track over map with uncertainty shown by space utilisation distribution*

Description

May be use not only for the whole track but for a set of specific dates, e.g. to show spatial uncertainty during migration. Note that you can use it only after obtaining and registering in you current session Google Api Key. For details on the API key, see https://ornithologyexchange.org/forums/topic/38315-mapflightrggmap-error/.

Usage

```
plot_util_distr(
  Result,
  dates = NULL,
  map.options = NULL,
  percentiles = c(0.4, 0.6, 0.8),
  zoom = "auto",
  geom_polygon.options = NULL,
  save.options = NULL,
  color.palette = NULL,
  use.palette = TRUE,
  background = NULL,
  plot = TRUE,
  save = TRUE
)
```

Arguments

Result	FLightR result object obtained from run.particle.filter
dates	Use NULL if all twilights will be used for plotting, one integer if specific twilight should be plotted (line number in Result$Results$Quantiles). Use data.frame with first column - start of the period and second - end of the period and each line represents a new period to plot specific periods, e.g. wintering or migration.
map.options	options passed to ggmap::get_map(), note that zoom option is defined separately
percentiles	Probability breaks for utilisation distribution
zoom	Zoom for map. If 'auto' FLightR will try to find optimal zoom level by downloading different size maps and checking whether all the points fit the map.
geom_polygon.options	options passed to ggplot2::geom_polygon()
save.options	options passed to ggplot2::ggsave(). Filename should be defined here.
color.palette	colors for probability contours. Either NULL or grDevices::colorRampPalette() object
use.palette	should the same colors be used for polygon boundaries as for polygon filling?
background	if provided will be used as a background. Must be created by ggmap::get_map()
plot	should function produce a plot?
save	should function save results with ggplot2::ggsave()?

Value

list with two parts

res_buffers	spatial buffers for defined probability values
p	ggplot2::ggplot() object

Author(s)

Eldar Rakhimberdiev

Examples

```
File<-system.file("extdata", "Godwit_TAGS_format.csv", package = "FLightR")
# to run example fast we will cut the real data file by 2013 Aug 20
Proc.data<-get.tags.data(File, end.date=as.POSIXct('2013-06-25', tz='GMT'))
Calibration.periods<-data.frame(
       calibration.start=as.POSIXct(c(NA, "2014-05-05"), tz='GMT'),
       calibration.stop=as.POSIXct(c("2013-08-20", NA), tz='GMT'),
       lon=5.43, lat=52.93)
     #use c() also for the geographic coordinates, if you have more than one calibration location
       # (e. g.,  lon=c(5.43, 6.00), lat=c(52.93,52.94))

# NB Below likelihood.correction is set to FALSE for fast run!
# Leave it as default TRUE for real examples
Calibration<-make.calibration(Proc.data, Calibration.periods, likelihood.correction=FALSE)

Grid<-make.grid(left=0, bottom=50, right=10, top=56,
  distance.from.land.allowed.to.use=c(-Inf, Inf),
  distance.from.land.allowed.to.stay=c(-Inf, Inf))

all.in<-make.prerun.object(Proc.data, Grid, start=c(5.43, 52.93),
                           Calibration=Calibration, threads=1)
# here we will run only 1e4 partilces for a very short track.
# One should use 1e6 particles for the full run
Result<-run.particle.filter(all.in, threads=1,
           nParticles=1e3, known.last=TRUE,
           precision.sd=25, check.outliers=FALSE)

## Not run:
plot_util_distr(Result, zoom=6, save=FALSE)

## End(Not run)
```

---

posterior_point_distribution

Extract posterior point distribution for one twilight

Description

Extract posterior point distribution for one twilight

Usage

```
posterior_point_distribution(Result, twilight_index)
```

Arguments

`Result`	A FLightR Result-like object.
`twilight_index`	Integer twilight index.

Value

A data.frame with lon, lat, and weight columns.

`read_gps_track` *Read a GPS track for validation*

Description

Reads a CSV file containing GPS fixes. Required columns are time, lon, and lat. The time column is converted to POSIXct in UTC when needed.

Usage

```
read_gps_track(path)
```

Arguments

`path`	Path to a private local GPS CSV file.

Value

A data.frame with POSIXct time and numeric lon/lat columns.

`read_validation_config` *Read local validation configuration*

Description

Reads a YAML configuration file for local, private double-tag validation. The parser uses the yaml package when available and otherwise supports the simple key/value structure used by the example config.

Usage

```
read_validation_config(path)
```

Arguments

`path`	Path to a local validation config file.

Value

A named list with validation configuration.

`run.particle.filter` *Run Particle Filter*

Description

Main function of FLightR, it takes fully prepared object created by `make.prerun.object` and produces a result object that can be used for plotting etc.

Usage

```
run.particle.filter(
  all.out,
  cpus = NULL,
  threads = -1,
  nParticles = 1e+06,
  known.last = TRUE,
  precision.sd = 25,
  behav.mask.low.value = 0,
  k = NA,
  plot = TRUE,
  cluster.type = "PSOCK",
  a = 45,
  b = 1500,
  L = 90,
  adaptive.resampling = 0.99,
  check.outliers = FALSE,
  sink2file = FALSE,
  add.jitter = FALSE,
  profile.phases = FALSE,
  profile.top.level = FALSE,
  propagation.backend = c("auto", "cached", "legacy", "partial_cached"),
  verbose = FALSE
)
```

Arguments

`all.out`	An object created by `make.prerun.object`.
`cpus`	another way to specify `threads`
`threads`	An amount of threads to use for the particle-filter step. Default is -1. Value 1 forces sequential evaluation. Note that parallelism in `make.prerun.object` can be useful, but recent validation benchmarks found the particle-filter PSOCK path slower than `threads = 1` for the tested data/backend; benchmark your own data before using `threads > 1` here.

`nParticles` total amount of particles to be used with the run. 10 000 (1e4) is recommended for the preliminary run and 1 000 000 (1e6) for the final

`known.last` Set to FALSE if your bird was not at a known place during last twilight in the data

`precision.sd` if `known.last` then what is the precision of this information. Will be used to resample particles proportionally to their distance from the known last point with probability `P = dnorm(0, precision.sd)`

`behav.mask.low.value`
Probability value that will be used instead of 0 in the behavioural mask. If set to 1 behavioural mask will not be active anymore

`k` Kappa parameter from vonMises distribution. Default is NA, otherwise will generate particles in a direction of a previous transitions with kappa = k

`plot` Should function plot preliminary map in the end of the run?

`cluster.type` see help to package parallel for details

`a` minimum distance that is used in the movement model - left boundary for truncated normal distribution of distances moved between twilights. Default is 45 for as default grid has a minimum distance of 50 km.

`b` Maximum distance allowed to fly between two consecutive twilights

`L` how many consecutive particles to resample

`adaptive.resampling`
Above what level of ESS resampling should be skipped

`check.outliers` switches ON the online outlier routine

`sink2file` will write run details in a file instead of showing on the screen

`add.jitter` will add spatial jitter inside a grid cell for the median estimates

`profile.phases` logical; if `TRUE`, stores detailed particle-filter phase timings in the returned object. Intended for diagnostics.

`profile.top.level`
logical; if `TRUE`, stores top-level timing for particle-filter post-processing phases in the returned object.

`propagation.backend`
Movement proposal backend. `"auto"` currently uses the fast cached backend when possible and falls back to legacy on construction failure. `"partial_cached"` lazily caches reachable destinations only for visited grid cells and is recommended for larger spatial extents; local validation benchmarks observed about 20-25x faster particle-filter runtime for a full-length 1e6-particle run on one tested dataset and machine. Actual speedups depend on grid size, backend, particle count, hardware, and thread settings. `"cached"` eagerly builds candidate lists for all grid cells and can be fastest on moderate grids. `"legacy"` keeps the original per-proposal distance calculation for fallback/reproducibility.

`verbose` Controls particle-filter console output. The default `FALSE` is quiet and suppresses normal progress messages while keeping warnings and errors visible. Use `TRUE` for normal progress messages, or `"debug"` for detailed loop diagnostics.

Value

FLightR object, containing output and extracted results. It is a list with the following elements

`Indices` List with prior information and indices

`Spatial` Spatial data - Grid, Mask, spatial likelihood

`Calibration` all calibration parameters

`Data` original data

`Results` The main results object. Main components of it are

Quantiles dataframe containing results on locations. Each line corresponds to a twilight

Movement.results dataframe containing all the movement results, Note - time at line n means time of the end of transition between n and n-1

outliers id of twilights excluded by online outlier detection tool

LL -Log likelihood

Points.rle run length encoding object with posterior distribution for every twilight. Note that numbers of points correspond to line numbers in `$Spatial$Grid`

Transitions.rle run length encoding object with all the transitions

Author(s)

Eldar Rakhimberdiev

Examples

```
File<-system.file("extdata", "Godwit_TAGS_format.csv", package = "FLightR")
# to run example fast we will cut the real data file by 2013 Aug 20
Proc.data<-get.tags.data(File, end.date=as.POSIXct('2013-07-02', tz='GMT'))
Calibration.periods<-data.frame(
       calibration.start=NA,
       calibration.stop=as.POSIXct("2013-08-20", tz='GMT'),
       lon=5.43, lat=52.93)
     #use c() also for the geographic coordinates, if you have more than one calibration location
       # (e. g.,  lon=c(5.43, 6.00), lat=c(52.93,52.94))
print(Calibration.periods)

# NB Below likelihood.correction is set to FALSE for fast run!
# Leave it as default TRUE for real examples
Calibration<-make.calibration(Proc.data, Calibration.periods, likelihood.correction=FALSE)

Grid<-make.grid(left=0, bottom=50, right=10, top=56,
  distance.from.land.allowed.to.use=c(-Inf, Inf),
  distance.from.land.allowed.to.stay=c(-Inf, Inf))

all.in<-make.prerun.object(Proc.data, Grid, start=c(5.43, 52.93),
                           Calibration=Calibration, threads=2)
# here we will run only 1e4 partilces for a very short track.
# One should use 1e6 particles for the full run.
Result<-run.particle.filter(all.in, threads=1,
```

```
            nParticles=1e3, known.last=TRUE,
            precision.sd=25, check.outliers=FALSE)
```

`stationary.migration.summary` *find potential stationary periods and estimates their location and movement schedule*

Description

This function will find any sites where birds stayed longer than `min.stay`. Potential movement is detected by the minimum probability of movement `prob.cutoff`.

Usage

```
stationary.migration.summary(Result, prob.cutoff = 0.1, min.stay = 3)
```

Arguments

`Result`	FLightR result object obtained from `run.particle.filter`
`prob.cutoff`	Minimum probability that defines movement
`min.stay`	Minimum duration of stationary period (in twilights)

Value

list with stationary and movement statistics

Author(s)

Eldar Rakhimberdiev

Examples

```
File<-system.file("extdata", "Godwit_TAGS_format.csv", package = "FLightR")
# to run example fast we will cut the real data file by 2013 Aug 20
Proc.data<-get.tags.data(File, end.date=as.POSIXct('2013-06-25', tz='GMT'))
Calibration.periods<-data.frame(
       calibration.start=as.POSIXct(c(NA, "2014-05-05"), tz='GMT'),
       calibration.stop=as.POSIXct(c("2013-08-20", NA), tz='GMT'),
       lon=5.43, lat=52.93)
     #use c() also for the geographic coordinates, if you have more than one calibration location
       # (e. g.,  lon=c(5.43, 6.00), lat=c(52.93,52.94))

# NB Below likelihood.correction is set to FALSE for fast run!
# Leave it as default TRUE for real examples
Calibration<-make.calibration(Proc.data, Calibration.periods, likelihood.correction=FALSE)

Grid<-make.grid(left=0, bottom=50, right=10, top=56,
```

```
  distance.from.land.allowed.to.use=c(-Inf, Inf),
  distance.from.land.allowed.to.stay=c(-Inf, Inf))

all.in<-make.prerun.object(Proc.data, Grid, start=c(5.43, 52.93),
                           Calibration=Calibration, threads=1)
# here we will run only 1e4 partilces for a very short track.
# One should use 1e6 particles for the full run.
Result<-run.particle.filter(all.in, threads=1,
          nParticles=1e3, known.last=TRUE,
          precision.sd=25, check.outliers=FALSE)

Summary<-stationary.migration.summary(Result, prob.cutoff=1)
# Use lower cut offs for real runs!
```

`twGeos2TAGS` *Function to write down twilights annotated in twGeos package data in so-called TAGS format*

Description

this function converts combines twilights detected in twGeos with raw data and writes them down in TAGS format that can be easily read by `get.tags.data`

Usage

```
twGeos2TAGS(raw, twl, threshold, filename = NULL)
```

Arguments

`raw`	original data - dataframe with two columns first column must contain time and second measured light levels
`twl`	twilights object from `preprocess.light` function
`threshold`	threshold value used for twilight definition in `preprocess.light`
`filename`	if NULL data.frame in TAGS format will be returned otherwise .csv file in TAGS format will be written

Details

TAGS format returned or written as .csv by this function is a dataframe with columns

`datetime` date and time in ISO 8601 format e.g. 2013-06-16T00:00:11.000Z

`light` light value measured by tag

`twilight` assigned by the software numeric indication of whether the record belongs to sunrise (1), sunset (2) or none of those (0)

`excluded` indication of whether a twilight was excluded during manual inspection (logical, `TRUE | FALSE`)

`interp` indication of whether the light value at twilight was interpolated (logical, `TRUE | FALSE`)

The fields `excluded` and `interp` may have values of `TRUE` only for `twilight > 0`.

Value

NULL if filename is provided or TAGS formatted dataframe.

Author(s)

Eldar Rakhimberdiev & Simeon Lisovski

See Also

BAStag2TAGS and GeoLight2TAGS

---

validation_summary *Summarize validation against matched GPS fixes*

Description

Summarize validation against matched GPS fixes

Usage

```
validation_summary(Result, gps_matched)
```

Arguments

Result	A FLightR Result-like object with posterior point distributions.
gps_matched	Output of match_gps_to_twilights().

Value

A one-row data.frame with validation metrics.

Index

BAStag2TAGS, 2, *7*, *29*

find.stationary.location, 3
find.times.distribution, 5
FLightR2Movebank, 5

GeoLight2TAGS, *3*, 6, *29*
get.tags.data, *2*, *4*, *6*, 7, *10*, *13*, *19*, *28*
get_ZI_distances, 9

make.calibration, 10, *13*
make.grid, 11, *13*
make.prerun.object, *11*, 12, *17*, *24*
map.FLightR.ggmap, 14
match_gps_to_twilights, 16

plot_likelihood, 17
plot_lon_lat, 18
plot_slopes_by_location, 19
plot_util_distr, 20
posterior_point_distribution, 22

read_gps_track, 23
read_validation_config, 23
run.particle.filter, *5*, *6*, *9*, *10*, *12*, *13*, *15*, *17*, *18*, *21*, 24, *27*

stationary.migration.summary, 27

twGeos2TAGS, *3*, *7*, 28

validation_summary, 29