Package: sporeg (via r-universe)

August 31, 2024

Title A package to reconstruct sporadic passive tracking data and measure the bias and uncertainty incurred through the process

Version 0.1.0

Description This package is the corresponding code for the Methods in Ecology and Evolution article, ``A novel process to infer the reliability of ecological information derived from passive acoustic telemetry track reconstruction''. The intended workflow is to simulate tracks inside a specified polygon, calculate quadrat counts inside grid cells, derive passive detection data from the simulated tracks using geolocations of listening stations, reconstruct the tracks using continuous-time correlated random walk models, re-route any spurious track intersections with a specified boundary, calculate quadrat counts of reconstructed tracks within grid cells, and compare quadrat counts between simulated and reconstructed tracks. This process is designed to be iterated to achieve statistical power. The user can then proceed with modeling the relationship between environmental variables and a desirable result (i.e., a good fit) to determine what environmental variables affect the odds of a good fit in grid cells.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Suggests knitr, rmarkdown, car, geosphere, mapview, oceanmap, parallel, sp, data.table

VignetteBuilder knitr

Imports dplyr, glatos (>= 0.7.3), pathroutr (>= 0.2.1), sf (>= 1.0-12), lubridate, momentuHMM, purrr, tibble, tidyr, units, magrittr, stats

Remotes jmlondon/pathroutr, ocean-tracking-network/glatos

Depends R (>= 3.5.0)

LazyData true

Repository https://ocean-tracking-network.r-universe.dev

RemoteUrl https://github.com/mebowers5/sporeg

RemoteRef HEAD

RemoteSha e2cdfaca0bc82714fce48758da15423647c6473a

Contents

`comp_trks` *Compare grid cell counts*

Description

Compare grid cell counts

Usage

```
comp_trks(
  sim_trks,
  stations,
  land_barrier,
  vis_graph,
  multi.grid,
  HSgrid,
  snap_tolerance
)
```

Arguments

`sim_trks`	Tracks to which modeled tracks should be compared
`stations`	A sf object comprised of receiver station locations with a buffer around them that represents the range of the receiver (polygons)
`land_barrier`	A sf polygon of a barrier object around which tracks should be re-routed
`vis_graph`	A visibility graph created from the barrier object @seealso pathroutr::prt_visgraph
`multi.grid`	boolean
`HSgrid`	When multi.grid = FALSE, a sf polygon grid; When multi.grid = TRUE, a list of sf polygon grids
`snap_tolerance`	The tolerance (in meters) at which an intersection between a station and a track should snap to the station centroid. It is recommended that the snap_tolerance be equal to the station buffer size.

Value

A data frame with counts and differences by grid cell ID ("gid")

`cts` *Counts*

Description

The function allows you to calculate counts per grid cell in demonstrative modeled movement data.

Usage

```
cts(sg, df)
```

Arguments

`sg`	spatial grid created from `grid_res` function
`df`	sf object of reconstructed, re-routed, optionally buffered tracks created from `sub_rrt` function

Value

A simple feature object with counts associated with grid cell IDs `"gid"` den_rcvs *Summarize receiver density and counts*

Description

This function provides summary statistics on the densities and counts of receivers in a grid.

Usage

```
den_rcvs(df)
```

Arguments

df	A simple feature polygon object (a grid)

Value

A data frame object with minimum, mean, and maximum of receiver densities and counts in km^-2

Examples

```
# Apply den_rcvs to list of grid resolutions

library(sporeg)
library(dplyr)
library(sf)
library(data.table)
load(system.file("extdata", "res.Rda", package = "sporeg"))

rcv_dens <- lapply(res, den_rcvs)

rcv_dens <- data.table::rbindlist(rcv_dens, idcol = 'resolution') %>%
left_join(tibble(resolution = 1:4,
res_name = c("100km", "50km", "25km", "10km")),
by = "resolution") %>%
mutate(res_name = ordered(res_name, levels = c("100km", "50km", "25km", "10km")))
```

depth_data *Calculate mean depth*

Description

This function allows you to calculate mean depth per grid cell.

Usage

```
depth_data(HSgrid, depth)
```

Arguments

HSgrid	list object of multiple grids
depth	simple feature point object with "altitude" attribute data

Value

A simple feature multipolygon object with mean depth per grid cell

---

dif_co — *Different cut off*

Description

Different cut off

Usage

```
dif_co(df, depth_limit)
```

Arguments

df	data frame object consisting of results from iterative reconstruction process
depth_limit	a depth value in meters that represents the new depth cut off of interest

Value

a data frame object containing the percentage of grid cells that contained a good fit out of all grid cells that had an average depth (mean_depth) less than or equal to the depth_limit

Examples

```
# Apply dif_co to list of grid resolutions

library(sporeg)
library(dplyr)
library(sf)
library(data.table)
load(system.file("extdata", "res.Rda", package = "sporeg"))

depth_limit <- 300 #[m]

dif_depth <- lapply(res, dif_co) %>%
data.table::rbindlist(., idcol = 'resolution') %>%
left_join(tibble(resolution = 1:4,
res_name = c("100km", "50km", "25km", "10km")),
by = "resolution") %>%
dplyr::mutate(res_name = ordered(res_name, levels = c("100km", "50km", "25km", "10km")))
```

d_shore_rcvs — *Distance to shore and density of receivers*

Description

This function determines the distance to shore and the density of receivers for each grid cell.

Usage

```
d_shore_rcvs(km, study_site, land_barrier, epsg, sts_pts)
```

Arguments

`km`	grid cell resolution in km. one-sided length of grid cell, assumes desired grid cell is to be squared
`study_site`	simple feature polygon object that encompasses the entire study site
`land_barrier`	simple feature (multi)polygon object to route tracks around
`epsg`	epsg code for desired coordinated system transformation
`sts_pts`	a simple feature (multi)point object representing receiver locations

Value

A simple feature multipolygon object with information on distance to shore from grid cell center, receiver presence/absence, counts, and densities

gd_fts — *Good fits summary statistics*

Description

This function summarizes minimum, maximum, and mean statistics for depth (mean_depth), distance from shore (d_shore), receiver count (count), receiver presence/absence (p_a), and the percentage of grid cells missing receivers for all of the good fits (g_fit).

Usage

```
gd_fts(df)
```

Arguments

`df`	data frame object consisting of results from iterative reconstruction process

Value

a data frame object with summary statistics from grid cells that contained good fits

Examples

```
# Apply gd_fts to list of grid resolutions

library(sporeg)
library(dplyr)
library(sf)
library(data.table)

load(system.file("extdata", "res.Rda", package = "sporeg"))

gd_fit_char <- lapply(res, gd_fts) %>%
data.table::rbindlist(., idcol = 'resolution') %>%
 left_join(tibble(resolution = 1:4,
                  res_name = c("100km", "50km", "25km", "10km")),
          by = "resolution") %>%
 dplyr::mutate(res_name = ordered(res_name, levels = c("100km", "50km", "25km", "10km")))
```

`get_or` *Get odds ratios*

Description

This function calculates the odds ratios for each variable in your model.

Usage

```
get_or(model)
```

Arguments

`model`	A model object created from the `nlme` package

Value

A data frame object with a point estimate and lower and upper 95% confidence interval values associated with each covariate

Examples

```
# Get the odds ratio for each variable in the final model

load(system.file("extdata", "fit2.100km.Rda", package = "sporeg"))

odds_100km <- get_or(fit2.100km) %>%
dplyr::mutate(res_name = "100km")
```

gps_chr *Gaps characteristics*

Description

Gaps characteristics

Usage

```
gps_chr(df)
```

Arguments

df	data frame object consisting of results from iterative reconstruction process

Value

a data frame object with summary statistics that provide insight into the locations of the gaps in the network receiver array that were closed by the reconstruction process

Examples

```
# Apply gps_chr to list of grid resolutions

library(sporeg)
library(dplyr)
library(sf)
library(data.table)
load(system.file("extdata", "res.Rda", package = "sporeg"))

clsd_gp_chars <- lapply(res, gps_chr) %>%
data.table::rbindlist(., idcol = 'resolution') %>%
  left_join(tibble(resolution = 1:4,
                   res_name = c("100km", "50km", "25km", "10km")),
            by = "resolution") %>%
        dplyr::mutate(res_name = ordered(res_name, levels = c("100km", "50km", "25km", "10km")))
```

gps_fld *Gaps filled*

Description

This function calculates how well the reconstructions closed gaps in the network receiver array

Usage

```
gps_fld(df)
```

Arguments

df	data frame object consisting of results from iterative reconstruction process

Value

a data frame object with a percentage of grid cells that contained a good fit out of those that lacked receivers

Examples

```
# Apply gps_fld to list of grid resolutions

library(sporeg)
library(dplyr)
library(sf)
library(data.table)
load(system.file("extdata", "res.Rda", package = "sporeg"))

clsd_gps <- lapply(res, gps_fld) %>%
data.table::rbindlist(., idcol = 'resolution') %>%
 left_join(tibble(resolution = 1:4,
                  res_name = c("100km", "50km", "25km", "10km")),
           by = "resolution") %>%
 dplyr::mutate(res_name = ordered(res_name, levels = c("100km", "50km", "25km", "10km")))
```

---

grid_res   *Create grid*

Description

This function creates a grid inside a specified polygon.

Usage

```
grid_res(km, study_site, epsg, what)
```

Arguments

km	grid cell resolution in km. one-sided length of grid cell, assumes desired grid cell is to be squared
study_site	simple feature polygon object in which grid cells should be created
epsg	epsg numeric code for desired coordinate system transformation
what	"polygons" for grid cell polygons or "centers" for center points of grid cells

Value

grid comprised of polygons or points depending on `what` parameter

Examples

```
# NOTE: The study site must be in a projected coordinate system (e.g., WGS 84; EPSG: 3857)
# when it is initially fed into the function grid_res

library(sporeg)
library(dplyr)
library(sf)
load(system.file("extdata", "site_depth.Rda", package = "sporeg"))

site_depth <- site_depth %>% sf::st_transform(., 3857)
HS_100km_grid <- grid_res(100, site_depth, 4269, "polygons")
```

---

make_line *Make a line*

Description

This functions creates a line by connecting start and end points.

Usage

```
make_line(start_x, start_y, end_x, end_y)
```

Arguments

start_x	longitudinal sf coordinate object from the desired start point
start_y	latitudinal sf coordinate object from the desired start point
end_x	longitudinal sf coordinate object from the desired end point
end_y	latitudinal sf coordinate object from the desired end point

Value

A simple feature geometry object or simple feature linestring object

Examples

```
library(sporeg)
library(dplyr)
library(sf)
library(purrr)
library(tidyr)

load(system.file("extdata", "at_dly_locs.Rda", package = "sporeg"))

at_lines <- at_dly_locs %>%
 dplyr::group_by(ID, time) %>%
 sf::st_transform(3857) %>%
 dplyr::summarise(pt = sf::st_combine(geometry)) %>%
```

```
  sf::st_centroid() %>%
  dplyr::mutate(lat = sf::st_coordinates(pt)[,2],
         lon = sf::st_coordinates(pt)[,1]) %>%
  dplyr::arrange(ID, time) %>% #Order data for making lines
  dplyr::mutate(start_x = lon, start_y = lat,
  end_x = dplyr::lead(lon), end_y = dplyr::lead(lat)) %>%
  sf::st_as_sf(coords = c("lon", "lat"), crs = 3857) %>%
  dplyr::filter(!is.na(end_y)) %>%
  tidyr::nest() %>%
  dplyr::mutate(
    data = purrr::map(data,
                      ~ dplyr::mutate(.x,
                              x = purrr::pmap(.l = list(start_x, start_y, end_x, end_y),
                                              .f = make_line))))
```

---

powr *Run a power analysis*

Description

This is a wrapper function for running a power analysis on iterative simulation and reconstruction methods process.

Usage

```
powr(output, sig.level, power, delta, n)
```

Arguments

output	the resulting data frame from the iterative methods process
sig.level	numeric. the desired level of significance to achieve
power	numeric. the desired level of power to achieve
delta	numeric. the desired effect size to achieve
n	integer. the number of replicates/sample size. should be assigned NULL if desiring sample size

Value

A data frame object

Examples

```
# Use powr wrapper function on example results

library(sporeg)
library(dplyr)
library(data.table)
load(system.file("extdata", "results.Rda", package = "sporeg"))
```

```
results <- lapply(results, data.table::rbindlist, idcol = 'resolution')
results <- data.table::rbindlist(results, idcol = 'iteration') %>%
left_join(tibble(resolution = 1:4,
res_name = c("100km", "50km", "25km", "10km")),
by = "resolution")

df_var <- zero_var(results)
pow_stat <- df_var %>%
dplyr::group_by(res_name, gid) %>%
dplyr::summarise(sd = sd(dif)) %>%
dplyr::ungroup() %>%
dplyr::group_by(res_name)

res <- pow_stat %>% dplyr::filter(res_name == "100km")

anims <- 30
power <- 0.8
delta <- anims*0.01 # a delta within 1% of the total number of animals
sig.level <- 0.95
n <- NULL

pwr_100km <- powr(res, sig.level, power, delta)
print("Grid cell resolution: 100 km x 100 km")
pwr_100km
```

`rcv_chr` *Receiver characteristics*

Description

Receiver characteristics

Usage

```
rcv_chr(df)
```

Arguments

`df`	data frame object consisting of results from iterative reconstruction process

Value

a data frame object with summary statistics that provide insight into the locations of the receivers in the network receiver array

Examples

```
# Apply rcv_chr to list of grid resolutions

library(sporeg)
library(dplyr)
library(sf)
library(data.table)
load(system.file("extdata", "res.Rda", package = "sporeg"))

rcv_chars <- lapply(res, rcv_chr) %>%
data.table::rbindlist(., idcol = 'resolution') %>%
left_join(tibble(resolution = 1:4,
res_name = c("100km", "50km", "25km", "10km")),
            by = "resolution") %>%
              dplyr::mutate(res_name = ordered(res_name,
              levels = c("100km", "50km", "25km", "10km")))
```

---

simul_trks *Simulate tracks*

Description

This function simulates tracks inside a specified polygon.

Usage

```
simul_trks(
  anims,
  study_site,
  theta,
  vmin,
  vmax,
  rel_site,
  crs,
  n_days,
  initHeading
)
```

Arguments

anims	integer. quantity of desired animals to simulate
study_site	simple feature polygon object that encompasses area where tracks are allowed to be simulated.
theta	argument from `glatos::crw_in_polygon` function
vmin	numeric. minimum velocity from which to sample step length
vmax	numeric. maximum velocity from which to sample step length

rel_site	simple feature polygon object in which simulated animals are "released" or where simulated tracks begin
crs	EPSG code for study_site polygon. Must be projected coordinate system
n_days	integer. number of days that tracks should be simulated
initHeading	argument from glatos::crw_in_polygon function

Value

simple feature (multi)linestring object that represents individual simulated tracks

Examples

```
library(sporeg)
library(sf)
library(glatos)

load(system.file("extdata", "rel_site.Rda", package = "sporeg"))
load(system.file("extdata", "study_site.Rda", package = "sporeg"))

anims <- 30
yr <- 1
theta <- c(0, 1.74)
vmin <- 0.98
vmax <- 1.58
crs <- 3857
n_days <- 365*yr
initHeading <- 0

tracks <- simul_trks(anims, study_site, theta, vmin, vmax, rel_site, crs, n_days, initHeading)
```

---

sub_rrt *Re-route tracks around a barrier*

Description

This function allows you to re-route demonstrative movement data that has been reconstructed using a movement model around a polygon barrier.

Usage

```
sub_rrt(track_data, CRS, barrier, vis_graph, buffer)
```

Arguments

track_data	point location data with latitude and longitude information
CRS	epsg code for desired coordinate system transformation
barrier	polygon or multipolygon sf object

vis_graph — vis_graph object created from pathroutr::prt_vis_graph function

buffer — desired buffer size for re-routed tracks; generally should relate to range of receivers; distance should use the same units as the final coordinate system

Value

A simple feature (multi)polygon object

Examples

```
library(sporeg)
library(dplyr)
library(pathroutr)
library(sf)
load(system.file("extdata", "subset.Rda", package = "sporeg"))
load(system.file("extdata", "atlcoast.Rda", package = "sporeg"))

CRS <- 3857
barrier <- atlcoast
vis_graph <- pathroutr::prt_visgraph(barrier)
buffer <- 650

tbuff650 <- sub_rrt(subset, CRS, barrier, vis_graph, buffer)
```

---

zero_var — *Remove zero variance*

Description

This function removes grid cells that lacked any variance.

Usage

```
zero_var(df)
```

Arguments

df — data frame object of iterative methods results

Value

a data frame object with grid IDs that did not exhibit zero variance

Examples

```
# Remove grid cells with zero variance
library(sporeg)
library(dplyr)
load(system.file("extdata", "results.Rda", package = "sporeg"))

results <- lapply(results, data.table::rbindlist, idcol = 'resolution')
results <- data.table::rbindlist(results, idcol = 'iteration')
results <- results %>%
left_join(tibble(resolution = 1:4,
res_name = c("100km", "50km", "25km", "10km")),
by = "resolution")

df_var <- zero_var(results)
```

Index