Package 'anipaths'

August 2, 2025

Type Package

Title Animation of Multiple Trajectories with Uncertainty

Version 0.10.5

Date 2025-3-1

Maintainer Henry Scharf <hscharf@arizona.edu>

Description Animation of observed trajectories using spline-based interpolation (see for example, Buderman, F. E., Hooten, M. B., Ivan, J. S. and Shenk, T. M. (2016), <doi:10.1111/2041-210X.12465> ``A functional model for characterizing long-distance movement behaviour''. Methods Ecol Evol). Intended to be used exploratory data analysis, and perhaps for preparation of presentations.

License GPL-3

RoxygenNote 7.3.2

Depends R (>= 3.5.0)

Imports animation, RColorBrewer, sf, crawl, mgcv, grDevices, ggmap, dplyr, ellipse, ggplot2, igraph, lubridate, magrittr, mvtnorm, stringr, tidyr, tidyselect

Suggests knitr, rmarkdown, terra, testthat

VignetteBuilder knitr

LazyData true

Encoding UTF-8

NeedsCompilation no

Author Henry Scharf [aut, cre], Kristine Dinh [aut], Rosales Hugo [aut], Rivera Angelica [aut]

Repository CRAN

Date/Publication 2025-03-02 03:50:02 UTC

Contents

`animate_paths` *animate paths*

Description

Animates telemetry data for the purposed of EDA using smoothing splines to interpolate the observed locations. The animations are particularly useful when examining multiple simultaneous trajectories. The output of the call to `animate_paths()` should bring up a browser window that shows the animation. Additionally, the images generated in `images/` (or else the value set for `imgdir`) may be used with ffmpeg, latex, or other presentation software that can build animations directly from a sequence of images.

Usage

```
animate_paths(
  paths,
  coord = c("x", "y"),
  Time.name = "time",
  background = NULL,
  bg.axes = TRUE,
  bg.misc = NULL,
  bg.opts = NULL,
  blur.size = 8,
  covariate = NULL,
  covariate.colors = c("black", "white"),
  covariate.legend.loc = "bottomright",
  covariate.thresh = NULL,
  crawl.mu.color = "black",
  crawl.plot.type = "point.tail",
  date.col = "black",
```

```
  delta.t = NULL,
  dev.opts = list(),
  dimmed = NULL,
  ID.name = NULL,
  interpolation_type = "gam",
  interval = 1/12,
  legend.loc = "topright",
  main = NULL,
  max_refit_attempts = 10,
  method = "html",
  n.frames = NULL,
  network = NULL,
  network.colors = NULL,
  network.thresh = 0.5,
  network.times = NULL,
  network.ring.trans = 1,
  network.ring.wt = 3,
  network.segment.trans = 0.5,
  network.segment.wt = 3,
  override = FALSE,
  par.opts = list(),
  paths.proj = "+proj=longlat",
  paths.tranform.crs = "+proj=aea",
  plot.date = TRUE,
  pt.alpha = 0.4,
  pt.cex = 1,
  pt.colors = NULL,
  pt.wd = 1,
  res = 1.5,
  return.paths = FALSE,
  s_args = NULL,
  simulation = FALSE,
  simulation.iter = 12,
  tail.alpha = 0.6,
  tail.colors = "gray87",
  tail.length = 5,
  tail.wd = 1,
  theme_map = NULL,
  times = NULL,
  uncertainty.level = NA,
  uncertainty.type = 1,
  whole.path = FALSE,
  xlim = NULL,
  ylim = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

paths	Either a data.frame with longitudes/eastings, latitudes/northings, IDs, and times (see coord, ID.name, and Time.name), a SpatialPointsDataFrame with IDs and times, or a list of data.frames containing the longitudes, latitudes, and times for each individual (with names provided). If all paths are already synchronous, another option for passing the data is to define paths as a list of matrices, all with the same number of rows, and to specify the times separately via the next argument. This situation might arise when, for example, locations the user wishes to animated correspond to realizations/sampler from a discrete-time movement model. Covariates may be provided as named columns of the matrices in paths.
coord	A character vector of length 2 giving the names of the longitude/easting and latitude/northing columns in the paths data.frame (in that order). This is required if paths is not a SpatialPointsDataFrame.
Time.name	The name of the columns in paths gving the observation times. This column must be of class POSIXt, or numeric.
background	Three possibilities: (1) A single background image over which animation will be overlayed, or a SpatRaster objects with one layers corresponding to each frame. (2) A list with values center (long/lat), zoom, and maptype (see ggmap::get_googlemap()) which will be used to generate a background for the animation based on Google maps tiles. Additional arguments may be added which will be passed to ggmap::get_googlemap(). (3) A logical value of TRUE, which will cue the function to get the best Google Map tile combination it can come up with. Note: ggmap must be installed for (2) and (3). Note: if you are calling animate_paths() several times in a short period of time you may get an error from Google for trying to pull tiles too often (e.g., Error in download.file(url, destfile = tmp, quiet = !messaging, mode = "wb") : cannot open URL 'http://maps.googleapis...'). Waiting a minute or so usually solves this.
bg.axes	logical: should animation place axis labels when using a background image (default is TRUE). If RGoogleMaps is used to produce background, labels will be "northing" and "easting". Otherwise, the strings given to coord will be used.
bg.misc	Character string which will be executed as R code after generating the background, and before adding trajectories, etc.
bg.opts	Options passed to plot() function call that makes background in each frame. For example, this could be used to specify blue ocean and gray land cover if background is a MULTIPOLYGON simple features object and bg.opts = list(bg = "dodgerblue4", col = "gray", border = "gray").
blur.size	a integer of the size for blur points; default is 8
covariate	The name of the column in paths that identifies the covariate to be mapped to a ring of color around each point.
covariate.colors	vector of colors which will be used in their given order to make a color ramp (see colorRamp())
covariate.legend.loc	either the location of the covariate legend, or NA if no legend is desired

covariate.thresh
: if changed from its default value of NULL, the interpolated value of the covariate will be binarized based on this numeric value.

crawl.mu.color
: color for the main predictions for crawl interpolation; default is black

crawl.plot.type
: a character string of what type of the plot you wish to generate when interpolation_type = "crawl". Default is "point.tail" for points with tails; input "point" for point plot and input "blur" for blur point plot; ; input "blur.point" for blur point with tails.

date.col
: default is "black"

delta.t
: The gap in time between each frame in the animation. Specify one of delta.t or n.frames. If both are specified, delta.t is used.

dev.opts
: Options passed to png() before creating each frame.

dimmed
: Numeric vector of individuals to "dim" in the animation. Order corresponds to the order of the ID.name variable, or order of paths list.

ID.name
: The name of the column in paths that identifies each individual. If left as NULL (default), a single individual is assumed.

interpolation_type
: a character string of the type of interpolation. Default is "gam" for a generalized addictive model. Use "crawl" to interpolate using crawl package. Note: due to the ongoing shift in PROJ4/6 standards, warning about CRS comments may appear.

interval
: Seconds per frame in animation. Default is 1/12 (or 12 frames per second).

legend.loc
: passed to first argument of legend() function. Default is "topright". NA removes legend.

main
: Title for each frame.

max_refit_attempts
: an integer of number of resampling when the fit for crawl failed to run; default is 10

method
: either "html" (default) or "mp4". The latter requires the user has installed ffmpeg (see ?animation::saveVideo()).

n.frames
: The number of frames used to animate the complete time domain of the data.

network
: Array of dimensions (# individuals, # individuals, n.frames) that gives a dyanmic network structure among the individuals.

network.colors
: A symmetric matrix of dimension length(paths) $\times$ length(paths) giving the colors associated with each pairwise relationship.

network.thresh
: Network structure is summarized in the animation in a binary way, regardless of whether or not the network is continuously weighted or not. The value of network.thresh determines the level below which no connection is shown, and above which an active connection is shown via colored rings and connecting segments.

network.times
: Numeric vector. If network time grid doesn't match n.frames, supply the times at which the network has been evaluated so it can be interpolated using smoothing splines.

`network.ring.trans`
: transparency of network segments (default is 1)

`network.ring.wt`
: thickness of network rings (default is 3)

`network.segment.trans`
: transparency of network segments (default is 0.5)

`network.segment.wt`
: thickness of network segments (default is 3)

`override`
: Logical variable toggling where or not to override warnings about how long the animation procedure will take.

`par.opts`
: Options passed to `par()` before creating each frame.

`paths.proj`
: PROJ.4 string corresponding to the projection of the data. Default is "+proj=longlat".

`paths.tranform.crs`
: a PROJ.4 string of coordinate projection transformation based on the animals' location; default is "+proj=aea +lat_1=30 +lat_2=70".

`plot.date`
: Logical variable toggling date text at the time center of the animation.

`pt.alpha`
: alpha value for the points

`pt.cex`
: A numeric value giving the character expansion (size) of the points for each individual. Default is 1.

`pt.colors`
: A vector of colors to be used for each individual in the animation. Default values come from Color Brewer palettes. When a network is provided, this is ignored and individuals are all colored black. If `NA`, no plot colors are chosen to distinguish individuals. This can be useful when making animations involving a covariate. Consider also setting `legend.loc` to `NA` in this case.

`pt.wd`
: size of the points; default is 1

`res`
: Resolution of images in animation. Increase this for higher quality (and larger) images.

`return.paths`
: logical. Default is `FALSE`, but if `TRUE` then the interpolated paths are returned and no animation is produced.

`s_args`
: Default is `NULL`, in which case `anipaths` attempts to select a reasonable number of knots for the GAM interpolation. Alternatively, the user can provide a list of arguments to `mgcv::s()` the same length and order as number of unique individuals (i.e., `unique(paths[, ID.name])`). Each entry in the list should be a named list/vector (e.g., `s_args = list(list(k = 10), list(k = 12), ...)`).

`simulation`
: logical. Generate simulation predictions to have multiple projects for the animal paths; default is `FALSE`.

`simulation.iter`
: an integer of how many paths the crawl model will generate; default is 5.

`tail.alpha`
: alpha value for the tails

`tail.colors`
: default is `"gray87"`. Can be single color or vector of colors.

`tail.length`
: Length of the tail trailing each individual.

`tail.wd`
: Thickness of tail trailing behind each individual. Default is 1.

theme_map	plot theme for ggplot, default is NULL
times	If all paths are already synchronous, another option for passing the data is to define paths as a list of matrices, all with the same number of rows, and to specify the times separately via this argument.
uncertainty.level	value in (0, 1) corresponding to level at which to draw uncertainty ellipses. NA (default) results in no ellipses.
uncertainty.type	State what type of uncertainty plot 1 is default for tails more than 1 is amount of predicted trajectories for each unique individual and blurs for blur plot
whole.path	logical. If TRUE (default = FALSE), the complete interpolated trajectories will be plotted in the background of the animation. If whole.path = TRUE, consider also setting tail.length = 0.
xlim	Boundaries for plotting. If left undefined, the range of the data will be used.
ylim	Boundaries for plotting. If left undefined, the range of the data will be used.
verbose	logical; TRUE prints messages about fitting details
...	other arguments to be passed to ani.options to animation options such as the time interval between image frames.

Value

video file, possibly a directory containing the individual images, or interpolated paths.

Examples

```
##
vultures$POSIX <- as.POSIXct(vultures$timestamp, tz = "UTC")
vultures_paths <- vultures[vultures$POSIX > as.POSIXct("2009-03-01", origin = "1970-01-01") &
  vultures$POSIX < as.POSIXct("2009-05-01", origin = "1970-01-01"), ]
animate_paths(
  paths = vultures_paths,
  delta.t = "week",
  coord = c("location.long", "location.lat"),
  Time.name = "POSIX",
  ID.name = "individual.local.identifier"
)
## Not run:
background <- list(
  center = c(-90, 10),
  zoom = 3,
  maptype = "satellite"
)
library(ggmap)
library(RColorBrewer)
COVARIATE <- cos(as.numeric(vultures_paths$timestamp) /
  diff(range(as.numeric(vultures_paths$timestamp))) * 4 * pi)
animate_paths(
  paths = cbind(vultures_paths, COVARIATE),
  delta.t = "week",
```

```
  coord = c("location.long", "location.lat"),
  Time.name = "POSIX", covariate = "COVARIATE",
  covariate.colors = brewer.pal(n = 9, "RdYlGn"),
  ID.name = "individual.local.identifier",
  background = background
)

# animation using crawl interpolation
animate_paths(
  paths = vultures_paths,
  delta.t = "week",
  coord = c("location.long", "location.lat"),
  Time.name = "POSIX",
  ID.name = "individual.local.identifier",
  interpolation_type = "crawl"
)

## End(Not run)

# Run to remove files generated by this function
system("rm -r js; rm -r css; rm -r images; rm index.html")
```

`blur_point` *blur ellipses function*

Description

blur ellipses function

Usage

```
blur_point(
  x,
  levels = seq(0.001, 1 - 0.1, l = 15),
  alpha_mult,
  col = "black",
  center
)
```

Arguments

`x`	An object. In the default method the parameter x should be a correlation between -1 and 1 or a square positive definite matrix at least 2x2 in size. It will be treated as the correlation or covariance of a multivariate normal distribution.
`levels`	contour levels
`alpha_mult`	multiplier on transparency level
`col`	default is black
`center`	two-vector giving center of ellipse

check_overwrite *Check overwrite*

Description

Check overwrite

Usage

```
check_overwrite(method, return.paths, ...)
```

Arguments

method	passed from animate_paths()
return.paths	passed from animate_paths()
...	passed from animate_paths(); used to check for user-specified value for img.name

Value

NULL, unless there is risk of over-writting and the user interrupts animation (FALSE)

covariate_interp *Synchronous interpolation of covariate using either GAM (same as paths) or piece-wise constant if covariate is a factor*

Description

Synchronous interpolation of covariate using either GAM (same as paths) or piece-wise constant if covariate is a factor

Usage

```
covariate_interp(paths, covariate = NULL, Time.name, time.grid, s_args)
```

Arguments

paths	lists of data.frames containing positions, times, and covariate for each individual
covariate	character string giving name of covariate variable in data.frames
Time.name	character string giving name of time variable in data.frames
time.grid	grid of possible times to use for interpolation (individuals will only be interpolated to times within the range of observation times)
s_args	arguments to mgcv::s() for GAM interpolation method

Value

list of interpolated covariate by individual

gam_interp *GAM interpolation using* mgcv:gam().

Description

GAM interpolation using mgcv:gam().

Usage

```
gam_interp(
  formula = NULL,
  y,
  time,
  pred_times,
  se.fit = T,
  s_args = NULL,
  uncertainty.type,
  verbose = F
)
```

Arguments

formula	optionally specify formula for mgcv::gam() using y as response and time as predictor.
y	observations
time	times for observations
pred_times	prediction times
se.fit	logical default is TRUE; should standard pointwise errors be computed for interpolation
s_args	Arguments to mgcv::s() can be passed using a named list/vector.
uncertainty.type	State what type of uncertainty plot 1 is default for tails more than 1 is amount of predicted trajectories for each unique individual and blurs for blur plot
verbose	logical; TRUE prints messages about fitting details

Value

interpolated values

get_googlemap_min_scale

Figure out scale and centering of google map by transforming reported lat long bounding box back to web mercator

Description

Figure out scale and centering of google map by transforming reported lat long bounding box back to web mercator

Usage

```
get_googlemap_min_scale(map)
```

Arguments

map	ggmap object

Value

scale (factor by which web mercator has been shrunk) and min (leftmost, bottom most coordinate of rectangle)

googlemap_proj

adjust center + scale for google map plotting

Description

adjust center + scale for google map plotting

Usage

```
googlemap_proj(x, map)
```

Arguments

x	sf object
map	ggmap object

Value

two-column matrix of locations from x projected to match map

network_interp *Synchronous interpolation of network using piece-wise constant interpolation*

Description

Synchronous interpolation of network using piece-wise constant interpolation

Usage

```
network_interp(network = NULL, network.times, time.grid)
```

Arguments

network	array of network observations of dimension (n.indiv, n.indiv, length(network.times))
network.times	vector of times at which network observations are made
time.grid	times at which network will be interpolated

Value

array of dimension n.indiv, n.indiv, length(time.grid))

new_alpha *Get good alpha_mult*

Description

Get good alpha_mult

Usage

```
new_alpha(sd1, sd2)
```

Arguments

sd1	standard deviation of longitude
sd2	standard deviation of latitude

Value

scalar value to be used for alpha_mult in blur_point()

paths_gam_interp *Synchronous GAM interpolation of all paths*

Description

Synchronous GAM interpolation of all paths

Usage

```
paths_gam_interp(
  paths,
  coord,
  Time.name,
  time.grid,
  s_args = NULL,
  uncertainty.type,
  verbose = F
)
```

Arguments

paths	lists of data.frames containing positions, times, and covariate for each individual
coord	two-vector of character strings giving names of x and y coordinates in data.frames
Time.name	character string giving name of time variable in data.frames
time.grid	grid of possible times to use for interpolation (individuals will only be interpolated to times within the range of observation times)
s_args	List of arguments to `mgcv::s()` the same length as number of unique individuals. Each entry in the list should be a named list/vector.
uncertainty.type	State what type of uncertainty plot 1 is default for tails more than 1 is amount of predicted trajectories for each unique individual and blurs for blur plot
verbose	logical; TRUE prints messages about fitting details

Value

list of interpolated paths by individual

`plot.paths_animation` *Plot animation path interpolation*

Description

This is mainly intended as a way to check that the interpolations used in the animation are working as expected.

Usage

```
## S3 method for class 'paths_animation'
plot(x, ..., i = 1, level = 0.05, type = "path", ylim_x = NULL, ylim_y = NULL)
```

Arguments

`x`	`paths_animation` object as created through a call to `animate_paths()`.
`...`	additional arguments passed to `plot`.
`i`	index of individual to plot (corresponds to index in `unique(paths[, 'ID.name'])`).
`level`	confidence level for error bands. `NA` removes bands.
`type`	either `"path"` (default) for two marginal interpolation plots, or `"covariate"` for a single interpolation plot
`ylim_x`	y-axis limits for marginal plots (x, easting, etc.)
`ylim_y`	y-axis limits for marginal plots (y, northing, etc.)

Examples

```
vultures$POSIX <- as.POSIXct(vultures$timestamp, tz = "UTC")
vultures_paths <- vultures[vultures$POSIX > as.POSIXct("2009-03-22", origin = "1970-01-01") &
  vultures$POSIX < as.POSIXct("2009-04-05", origin = "1970-01-01"), ]
interpolated_paths <-
  animate_paths(
    paths = vultures_paths,
    delta.t = 3600 * 6,
    coord = c("location.long", "location.lat"),
    Time.name = "POSIX",
    ID.name = "individual.local.identifier",
    s_args = rep(list(list(k = 10)), 6),
    return.paths = TRUE
  )
plot(interpolated_paths, i = 2)
```

`vultures` *GPS locations of turkey vultures.*

Description

A dataset containing a subset of the locations of turkey vultures (2003–2006), with time stamps, from:

Usage

```
vultures
```

Format

A data frame with 215719 rows and 11 variables:

timestamp time of observation

location.long logitude

location.lat latitude

individual.local.identifier identifier for each individual ...

Details

Dodge S, Bohrer G, Bildstein K, Davidson SC, Weinzierl R, Mechard MJ, Barber D, Kays R, Brandes D, Han J (2014) Environmental drivers of variability in the movement ecology of turkey vultures (Cathartes aura) in North and South America. Philosophical Transactions of the Royal Society B 20130195. doi:10.1098/rstb.2013.0195

Bildstein K, Barber D, Bechard MJ (2014) Data from: Environmental drivers of variability in the movement ecology of turkey vultures (Cathartes aura) in North and South America. Movebank Data Repository. doi:10.5441/001/1.46ft1k05

Source

doi:10.5441/001/1.46ft1k05 Bildstein K, Barber D, Bechard MJ (2014) Data from: Environmental drivers of variability in the movement ecology of turkey vultures (Cathartes aura) in North and South America. Movebank Data Repository.

`whales` *GPS locations of three species of whales.*

Description

A dataset containing locations of whales, with time stamps, from:

Usage

```
whales
```

Format

A data frame with 4303 rows and 4 variables:

timestamp time of observation

location.long logitude

location.lat latitude

individual.local.identifier identifier for each individual ...

Details

Irvine LM, Winsor MH, Follett TM, Mate BR, Palacios DM (2020) An at-sea assessment of Argos location accuracy for three species of large whales, and the effect of deep-diving behavior on location error. Animal Biotelemetry 8:20.

Irvine LM, Follett TM, Winsor MH, Mate BR, Palacios DM (2020) Data from: Study "Blue and fin whales Southern California 2014-2015 - Argos data". Movebank Data Repository. doi:10.5441/001/1.98f5r6d0

Source

`doi:10.5441/001/1.98f5r6d0` Irvine LM, Follett TM, Winsor MH, Mate BR, Palacios DM (2020) Data from: Study "Blue and fin whales Southern California 2014-2015 - Argos data". Movebank Data Repository.

Index