

Package ‘abmR’

October 18, 2021

Type Package

Title Agent-Based Models in R

Version 1.0.4

Maintainer Benjamin Gochanour <ben.gochanour@gmail.com>

Description Supplies tools for running agent-based models (ABM) in R, as discussed in Gochanour et al. (2021) <[doi:10.1101/2021.09.15.460374](https://doi.org/10.1101/2021.09.15.460374)> The package contains two movement functions, each of which is based on the Ornstein-Uhlenbeck (OU) model (Ornstein & Uhlenbeck, 1930) <[doi:10.1103/PhysRev.36.823](https://doi.org/10.1103/PhysRev.36.823)>. It also contains several visualization and data summarization functions to facilitate the presentation of simulation results.

License GPL (>= 3)

Suggests jpeg, knitr

Encoding UTF-8

RoxygenNote 7.1.1

Depends R (>= 3.5)

Imports sp, rgdal, table1, googledrive, swfscMisc, geosphere, kableExtra, gtsummary, ggplot2, gstat, purrr, rnaturalearth, rnaturalearthdata, sf, tmap, raster, utils, stats, methods, rgeos

LazyData true

NeedsCompilation no

Author Benjamin Gochanour [aut, cre],
Javier Fernandez-Lopez [aut],
Andrea Contina [aut]

Repository CRAN

Date/Publication 2021-10-18 18:10:08 UTC

R topics documented:

as.species	2
energySIM	4

energyVIZ	6
EX1	7
ex_raster	8
get_ex_data	8
moveSIM	9
moveVIZ	11
tidy_results	12

Index	14
--------------	-----------

as.species	<i>Creates object of "species" class for input into moveSIM() and energySIM()</i>
------------	---

Description

Here we define the agents whose movement we will be modeling. The user must indicate the geographical origin of the agents but can also optionally specify values of morphological parameters, which will be standardized using (observed-mean/sd). The standardized values will affect the movement simulations by serving as a multiplier on "mot_x" and "mot_y" in the direction specified: "Pos" or "Neg".

Usage

```
as.species(
  x = NA,
  y = NA,
  morphpar1 = NA,
  morphpar1mean = NA,
  morphpar1sd = NA,
  morphpar1sign = NA,
  morphpar2 = NA,
  morphpar2mean = NA,
  morphpar2sd = NA,
  morphpar2sign = NA
)
```

Arguments

x	Species origin longitude value (degrees). Required.
y	Species origin latitude value (degrees). Required.
morphpar1	Observed value for morphological parameter 1 (numeric)
morphpar1mean	Population mean for morphological parameter 1 (numeric)
morphpar1sd	Population standard deviation for morphological parameter 1 (numeric)
morphpar1sign	Do higher values of morphpar1 lead to longer or shorter distances traveled each day? Specify "Pos" if longer and "Neg" if shorter.

morphpar2	Observed value for morphological parameter 2 (numeric)
morphpar2mean	Population mean for morphological parameter 2 (numeric)
morphpar2sd	Population standard deviation for morphological parameter 2 (numeric)
morphpar2sign	Do higher values of morphpar2 lead to longer or shorter distances traveled each day? Specify "Pos" if longer and "Neg" if shorter.

Details

In example 1 below, we have a bird with origin (-100,26) with observed wing chord length of 15 mm, while the population mean for this measure is 10 mm with a SD of 2 mm (these birds are bigger than average). We declare morphpar1sign = "Pos" because we assume longer wingchord length leads to longer flight. Here, "morphpar2" represents mass, and we want to model heavier than average birds. We assume that heavier birds will fly longer distance, so specify 'morphpar2sign' = 'Pos'. If we assumed that heavier birds will fly shorter distance, we would specify 'morphpar2sign'="Neg" to indicate the inverse relationship.

Value

Object of class "species" for input into moveSIM() or energySIM()

Examples

```
# Example 1 -- Birds
# -Origin (-100,26)
#-Morphpar1=Wing Chord: Observed=15 mm, Pop. Mean=10, Pop. SD=2; Pos effect
#on movement speed
#-Morphpar2=Mass; Observed=15 g, Pop. Mean=6, Pop. SD=2; Pos effect on
# movement speed

my_species=as.species(x=-100,y=26,morphpar1=15,morphpar1mean=10,
morphpar1sd=2, morphpar1sign="Pos", morphpar2=7,morphpar2mean=6,
morphpar2sd=1,morphpar2sign="Pos")

# Example 2 -- Terrestrial Mammals
#-Origin (-90,40)
#-Morphpar1=Leg length: Observed=1.1 m, Pop. Mean=1, Pop. SD=.2; Pos effect
#on movement speed
#-Morphpar2=Mass; Observed=50 kg, Pop. Mean=55 kg, Pop. SD=10 kg; Neg effect
#on movement speed

my_species2=as.species(x=-90,y=40,morphpar1=1.1,morphpar1mean=1,
morphpar1sd=0.2, morphpar1sign="Pos", morphpar2=50,morphpar2mean=55,
morphpar2sd=10 ,morphpar2sign="Neg")

# Example 3 -- Unspecified agents
# -Origin (-90,40)
# -Not interested in modeling effect of morphology

my_species3=as.species(x=-90,y=40)
```

energySIM	<i>Runs agent-based model (ABM) movement and energy budget simulations based on environmental data</i>
-----------	--

Description

Here, agent mortality occurs when agent reaches energy = 0. Agent energy stores are dynamic, and affect search area as a multiplier, so movement is directly affected by the quality of raster cells achieved. Results may be visualized with energyVIZ(). Relies on underlying function energySIM_helper(), which is not to be used alone.

Usage

```
energySIM(
  replicates = 100,
  days,
  modeled_species,
  env_rast,
  optimum_lo,
  optimum_hi,
  dest_x,
  dest_y,
  mot_x,
  mot_y,
  search_radius = 375,
  direction = "S",
  sigma = 0.1,
  mortality = TRUE,
  init_energy = 100,
  energy_adj = c(25, 20, 15, 10, 5, 0, -5, -10, -15, -20, -25),
  single_rast = FALSE,
  write_results = FALSE
)
```

Arguments

replicates	Integer, desired number of replicates per run, default 100.
days	Integer, How many days (timesteps) would you like to model? Range (1,nlayers(env_rast))
modeled_species	Object of class "species"
env_rast	Rasterstack or Rasterbrick with number of layers >= days
optimum_lo	Numeric, optimal environmental value (low)
optimum_hi	Numeric, optimal environmental value (high)
dest_x	Numeric, destination x coordinate (longitude)

dest_y	Numeric, destination y coordinate (latitude)
mot_x	Numeric, movement motivation in x direction, range (0,1], default 1.
mot_y	Numeric, movement motivation in y direction, range (0,1], default 1.
search_radius	Radius of semicircle search regions (in km). Default 375.
direction	Character, movement direction, one of "N", "S", "E", "W", or "R" (Random). Default "S".
sigma	Numeric, randomness parameter, range (-Inf, Inf). Default 0.1.
mortality	Logical, should low energy levels result in death? Default T.
init_energy	Numeric, initial energy in interval (0,100]
energy_adj	Numeric, Vector of length 11 representing desired energy gain/penalty corresponding to achieved env values in optimum range (1st element), and within 10, 20, ..., 80, 90, and 90+ percent (11th element) of the average of optimum hi and optimum lo. Recommend using default which is decreasing and symmetric about zero but can modify if desired.
single_rast	Logical, are you using a one-layer raster for all timesteps?. Default F.
write_results	Logical, save results to csv? Default F.

Details

For each timestep, agents can have status "Alive", "Stopped", or "Died". All agents start alive and may stop if, on a particular timestep, there are no non-NA raster values in the search region. This often occurs when agents are searching over an ocean or a large lake, for example. Once an agent stops, they remain stopped for the rest of the run. Similarly, once an agent dies, they retain this status for all subsequent timesteps. All timesteps with agent status "Stopped" or "Died" will have lat/lon=NA, so as to not affect subsequent analyses.

Value

Under "results", a (days+1 X replicates) rows X 9 column dataframe containing data on agent_id, day, longitude, latitude, current agent status (Alive, Stopped, or Died), energy, change in energy from last time_step, distance traveled from last timestep (in km), and final status. Using tidy_results() provides a cleaner display of results.

Under "run_params", a record of function parameters used as well as missing_pct and mortality_pct. missing_pct corresponds to the percent of rows in the results dataframe missing information on lon/lat, which occurs when the agent has "died" or "stopped". mortality_pct refers to the percentage of agents in the run that died.

Examples

```
# Define species object
pop1 <- as.species(
  x = -98.7, y = 34.7,
  morphpar1 = 15, morphpar1mean = 16, morphpar1sd = 2, morphpar1sign = "Pos",
  morphpar2 = 19, morphpar2mean = 18, morphpar2sd = 1, morphpar2sign = "Pos"
)
```

```
# Run function
EX1 <- energySIM(
  replicates = 3, days = 5, env_rast = ex_raster, search_radius = 400,
  sigma = .1, dest_x = -108.6, dest_y = 26.2, mot_x = .9, mot_y = .9,
  modeled_species = pop1,
  optimum_lo = .6, optimum_hi = .8, init_energy = 100,
  direction = "S", write_results = FALSE, single_rast = TRUE, mortality = TRUE)

# View Results in Clean Format
tidy_results(EX1, type = "results")
tidy_results(EX1, type = "run_params")
```

energyVIZ

Creates a plot/table of energySIM() results

Description

When type="plot", function plots the movement tracks versus the the straight line track between the origin and destination (unless the destination was unspecified in the call to energySIM(), then straight line track is omitted). When type="gradient", creates a gradient plot showing what regions cause agents to gain/lose energy. Two table options are also available using type="summary_table" or type="strat_table" (table with results stratified by energy gain or loss). Please see Vignette for examples of this output.

Usage

```
energyVIZ(
  data,
  type = "plot",
  title = "energySIM results",
  aspect_ratio = 1,
  label = FALSE,
  xlim = NULL,
  ylim = NULL
)
```

Arguments

data	Data to be plotted, this object should be the output from energySIM().
type	String from "plot", "gradient", "summary_table", or "strat_table"?
title	Title for the plot that is output.
aspect_ratio	Aspect ratio, defaults to 1.
label	Logical, label the origin and specified final destination?
xlim	Optionally specify desired x limits as a numeric vector: c(low,hi)
ylim	Optionally specify desired y limits as a numeric vector: c(low,hi)

Value

Plot or table displaying energySIM() results.

Examples

```
# 1. Define Population and Run energySIM()

#pop1 <- as.species(x=-98.7, y=34.7,
#morphpar1=15, morphpar1mean=16, morphpar1sd=2,morphpar1sign="Pos",
#morphpar2=19,morphpar2mean=18,morphpar2sd=1,morphpar2sign="Pos")

#EX1=energySIM(replicates=10,days=7,env_rast=ex_raster, search_radius=200,
#sigma=.1, dest_x=-108.6, dest_y=26.2, mot_x=.9, mot_y=.9,
#modeled_species=pop1,
#optimum_lo=.8,optimum_hi=.9,init_energy=100,
#direction="S",write_results=FALSE,single_rast=TRUE,mortality = TRUE)

data(EX1)

# 2. Run energyVIZ() on your result

energyVIZ(EX1,title="Visualizing EnergySIM results",type="plot", aspect_ratio=5/3,
label=TRUE)

#energyVIZ(EX1,type="summary_table")
#energyVIZ(EX1,type="strat_table")
#energyVIZ(EX1,type="gradient")
```

EX1

Example energySIM result

Description

This is an output for energySIM results.

Usage

```
data(EX1)
```

Format

An object of class `list` of length 2.

Examples

```
data(EX1)
```

`ex_raster`*Example Environmental Raster*

Description

Simulated environmental raster for usage in abmR examples.

Usage

```
data(ex_raster)
```

Format

An object of class RasterLayer of dimension 200 x 200 x 1.

Examples

```
data(ex_raster)
```

`get_ex_data`*Downloads example NDVI data*

Description

Warning: this function will download to your hard drive (to a location specified by your current working directory) the below files, totaling approximately 620 MB. Please do not attempt to use if you have insufficient hard drive space or Random Access Memory (RAM). Objects to be downloaded are listed under "details". The first time you use this function, you will be directed to your browser and required to sign in to your Google account to connect to the Tidyverse API. If you use the function a second time, you may simply follow the prompts and enter a number corresponding to the previous accounts listed.

Usage

```
get_ex_data()
```

Details

- `NDVI_2013_NA`: A raster stack containing daily NDVI data for North America, on a .05 x .05 degree grid. Data runs from 8/26/2013-9/21/2013.
- `NDVI_2013_NA_composite`: Single layer raster formed by taking mean of `NDVI_2013_NA`
- `NDVI_2013_Europe`: A raster stack containing daily NDVI data for Europe, on a .05 x .05 degree grid. Data runs from 8/26/2013-9/21/2013.
- `NDVI_2013_Europe_composite`: Single layer raster formed by taking mean of `NDVI_2013_Europe`

Value

No return value, called to download files to hard drive.

Source

Vermote, Eric; NOAA CDR Program. (2019): NOAA Climate Data Record (CDR) of AVHRR Normalized Difference Vegetation Index (NDVI), Version 5. NOAA National Centers for Environmental Information. <https://doi.org/10.7289/V5ZG6QH9>. Accessed 12/26/2020.

moveSIM	<i>Runs agent-based model (ABM) movement simulations based on environmental data</i>
---------	--

Description

Here, agent mortality occurs when agent fails to achieve suitable raster values at least `n_failures+1` timesteps in a row. Agent energy stores are not dynamic, so movement speed isn't directly affected by quality of raster cells achieved. Results may be analyzed with `moveVIZ()`. Relies on underlying function `moveSIM_helper()`, which is not to be used alone.

Usage

```
moveSIM(
  replicates = 100,
  days,
  modeled_species,
  env_rast,
  optimum,
  dest_x,
  dest_y,
  mot_x,
  mot_y,
  search_radius = 375,
  direction = "S",
  sigma = 0.1,
  mortality = TRUE,
  fail_thresh = 0.5,
  n_failures = 4,
  single_rast = FALSE,
  write_results = FALSE
)
```

Arguments

<code>replicates</code>	Integer, desired number of replicates per run. Default 100.
<code>days</code>	Integer, how many days (timesteps) would you like to model? Range (1,nlayers(env_rast))

modeled_species	Object of class "species"
env_rast	Rasterstack or Rasterbrick with number of layers \geq days
optimum	Numeric, optimal environmental value
dest_x	Numeric, destination x coordinate (longitude)
dest_y	Numeric, destination y coordinate (latitude)
mot_x	Numeric, movement motivation in x direction, range (0,1], default 1.
mot_y	Numeric, movement motivation in y direction, range (0,1], default 1.
search_radius	Radius of semicircle search regions (in km). Default 375.
direction	Character, movement direction, one of "N", "S", "E", "W", or "R" (Random). Default "S".
sigma	Numeric, randomness parameter, range (-Inf, Inf). Default 0.1.
mortality	Logical, should low energy levels result in death? Default T.
fail_thresh	What percentage deviation from optimum leads to death? E.g. default of .50 means 50 percent or greater deviation from optimum on a particular step constitutes failure.
n_failures	How many failures are allowable before agent experiences death (at n_failures+1). What constitutes a failure is determined by fail_thresh, range (1,days]. Default 4.
single_rast	Logical, are you using a one-layer raster for all timesteps? Default F.
write_results	Logical, save results to csv? Default F.

Details

For each timestep, agents can have status "Alive", "Stopped", or "Died". All agents start alive and may stop if, on a particular timestep, there are no non-NA raster values in the search region. This often occurs when agents are searching over an ocean or a large lake, for example. Once an agent stops, they remain stopped for the rest of the run. Similarly, once an agent dies, they retain this status for all subsequent timesteps. All timesteps with agent status "Stopped" or "Died" will have lat/lon=NA, so as to not affect subsequent analyses.

Arguments mortality, n_failures, and fail_thresh interact with each other. If mortality = F, values for n_failures and fail_thresh are ignored. If mortality=T, fail_thresh determines what constitutes a failure, and n_failures indicates how many failures are allowed before death. Note: If n_failures=days, this is equivalent to mortality=F.

Value

Under "results", a (days+1 * replicates) row X 7 column dataframe containing data on agent_id, day, longitude, latitude, current agent status (Alive, Stopped, or Died), distance traveled from last timestep (in km), and final status. Using tidy_results() provides a cleaner display of results.

Under "run_params", a record of function parameters used as well as missing_pct and mortality_pct. missing_pct corresponds to the percent of rows in the results dataframe missing information on lon/lat, which occurs when the agent has "died" or "stopped". mortality_pct refers to the percentage of agents in the run that died.

Examples

```
# Define species object
pop1 <- as.species(
  x = -98.7, y = 34.7, morphpar1 = 15, morphpar1mean = 16, morphpar1sd = 2,
  morphpar1sign = "Pos", morphpar2 = 19, morphpar2mean = 18, morphpar2sd = 1,
  morphpar2sign = "Pos")

# Run function
EX2 <- moveSIM(
  replicates = 3, days = 10, env_rast = ex_raster, search_radius = 300,
  sigma = .1, dest_x = -108.6, dest_y = 26.2, mot_x = .8, mot_y = .8,
  modeled_species = pop1, optimum = .6, n_failures = 5, fail_thresh = .40,
  direction = "S", write_results = FALSE, single_rast = TRUE, mortality = TRUE)

# View Results in Clean Format
tidy_results(EX2, type = "results")
tidy_results(EX2, type = "run_params")
```

moveVIZ

Creates a plot/table of moveSIM() results

Description

When type="plot", function plots the movement tracks versus the the straight line track between the origin and destination (unless the destination was unspecified in the call to moveSIM(), then straight line track is omitted). When type="summary_table", a summary table is output.

Usage

```
moveVIZ(
  data,
  type = "plot",
  title = "moveSIM results",
  aspect_ratio = 1,
  xlim = NULL,
  ylim = NULL
)
```

Arguments

data	Data to be plotted, this object should be the output from moveSIM().
type	"plot" or "summary_table", default "plot".
title	Title for the plot that is output.
aspect_ratio	Aspect ratio, defaults to 1.
xlim	Optionally specify desired x limits as a numeric vector: c(low,hi)
ylim	Optionally specify desired y limits as a numeric vector: c(low,hi)

Value

Plot or table displaying moveSIM() results.

Examples

```
# 1. Define Population and Run moveSIM()

pop1 <- as.species(x=-100, y=55,
  morphpar1=15, morphpar1mean=16, morphpar1sd=2, morphpar1sign="Pos",
  morphpar2=19, morphpar2mean=18, morphpar2sd=1, morphpar2sign="Pos")

EX2=moveSIM(replicates=2, days=27, env_rast=ex_raster, search_radius=550,
  sigma=.1, dest_x=-108.6, dest_y=26.2, mot_x=.8, mot_y=.8,
  modeled_species=pop1, optimum=.6, n_failures=5, fail_thresh=.40,
  direction="S", write_results=FALSE, single_rast=TRUE, mortality = TRUE)

# 2. Run moveVIZ() on your result
moveVIZ(EX2, title="Visualizing MoveSIM results", type="plot")

moveVIZ(EX2, type="summary_table")
```

tidy_results	<i>Prints results from moveSIM() or energySIM() in an easier-to-read table.</i>
--------------	---

Description

Prints results from moveSIM() or energySIM() in an easier-to-read table.

Usage

```
tidy_results(data, type = "results", nrows = NULL)
```

Arguments

data	The output from moveSIM() or energySIM() – a list of two dataframes.
type	"run_params" or "results", corresponding to which component of your moveSIM() or energySIM() output you'd like to print out. Default "results", which contains the movement data.
nrows	The number of rows to print.

Details

missing_pct and mortality_pct are not function parameters, but are nonetheless computed and returned here for your convenience

- missing_pct: What percent of rows exhibiting a missing location value (due to agent death or agent stopping)
- mortality_pct: What percent of simulated agents experienced death?

Value

Prints a cleaned table of moveSIM() or energySIM() results.

Index

* datasets

EX1, [7](#)

ex_raster, [8](#)

as.species, [2](#)

energySIM, [4](#)

energyVIZ, [6](#)

EX1, [7](#)

ex_raster, [8](#)

get_ex_data, [8](#)

moveSIM, [9](#)

moveVIZ, [11](#)

tidy_results, [12](#)