Package 'abmR'

June 15, 2021

Type Package

Title Agent-based Models in R

Version 0.1.13

Author Benjamin Gochanour, Javier Fernandez Lopez, Andrea Contina

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

Description A work-in-progress package for running agent-based models in R.

License GPL(>=3)

Suggests jpeg, knitr

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Depends R (>= 2.10)

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

R topics documented:

abmr 2
as.species 2
diseaseSIM 4

diseaseVIZ 6
energySIM 7
energyVIZ 9
get_ex_data 10
moveSIM 11
moveVIZ 13
tidy_results 14

Index **15**

`abmr` *Agent based modeling package*

Description

A work-in-progress package for running agent based models in R.

Author(s)

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

Javier Fernandez-Lopez <`jflopez.bio@gmail.com`>

Andrea Contina <`acontina@usgs.gov; acontina@gmail.com`>

`as.species` *Creates object of "species" class for input into moveSIM() and energySIM()*

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.

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)
```

`diseaseSIM` *Runs more advanced Brownian / Ornstein Uhlenbeck agent-based model for multiple replicates, with disease features.*

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

```
diseaseSIM(
  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,
  disease_loc,
  disease_radius = 5,
  disease_mortality = 0.5,
  disease_energy_interact = 40
)
```

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.
`disease_loc`	Dataframe of x,y coordinates specifying the location of disease clusters.
`disease_radius`	Numeric (0,Infty), what is the maximum distance from the source point is capable capable of causing disease?
`disease_mortality`	Numeric (0,1] What proportion of agents who pass directly over disease source (maximum disease load) will experience mortality? Assume mortality rate then linearly decreases to 0 at disease_radius.
`disease_energy_interact`	Numeric (0,100] Below what energy level should all agents exposed to disease die?

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=-100, y=55,
morphpar1=15, morphpar1mean=16, morphpar1sd=2,morphpar1sign="Pos",
morphpar2=19,morphpar2mean=18,morphpar2sd=1,morphpar2sign="Pos")

# Define disease locations
x=c(-99.475, -98.700, -102.725,-108.325,-98.700,-94.625,-95.175,-100.425, -103.675
,-102.475, -100.925, -100.025, -101.425,  -99.125,  -99.275)
y=c(32.225, 34.700, 34.575, 34.425, 34.700, 34.375, 32.525, 32.175, 31.675, 29.575, 28.225,
 26.275, 26.075, 25.025, 24.825)
 disease_points=data.frame(x=x,y=y)

# Run function
EX1 <- diseaseSIM(replicates=15,days=27,env_rast=ndvi_raster,
search_radius=400,
sigma=.1, dest_x=999, dest_y=999, mot_x=.9, mot_y=.9,
modeled_species=pabu.pop.new, optimum_lo=.6,optimum_hi=.8,init_energy=100,
direction="S",write_results=TRUE,single_rast=FALSE,mortality = TRUE,
energy_adj=c(30,25,20,5,0,-5,-5,-10,-20,-25,-30),disease_loc=disease_points,
disease_energy_interact = 60, disease_mortality=.5,disease_radius=300)

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

`diseaseVIZ` *Creates a plot/table of diseaseSIM() 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 diseaseSIM(), 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

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

Arguments

`data`	Data to be plotted, this object should be the output from diseaseSIM().
`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)

Examples

```
1. Run diseaseSIM()

EX1 <- diseaseSIM(replicates=15,days=27,env_rast=ndvi_raster,
search_radius=400,
sigma=.1, dest_x=999, dest_y=999, mot_x=.9, mot_y=.9,
modeled_species=pabu.pop.new, optimum_lo=.6,optimum_hi=.8,init_energy=100,
direction="S",write_results=TRUE,single_rast=FALSE,mortality = TRUE,
energy_adj=c(30,25,20,5,0,-5,-5,-10,-20,-25,-30),disease_loc=disease_points,
disease_energy_interact = 60, disease_mortality=.5,disease_radius=300)

2. Run diseaseVIZ() on your result

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

diseaseVIZ(EX1,type="summary_table")

diseaseVIZ(EX1,type="strat_table")

diseaseVIZ(EX1,type="gradient")
```

`energySIM` *Runs more advanced Brownian / Ornstein Uhlenbeck agent-based model for multiple replicates.*

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 = 5, days = 27, env_rast = ndvi_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 = FALSE, 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)

Examples

```
1. Run energySIM()

EX1=energySIM(replicates=5,days=27,env_rast=ndvi_raster, search_radius=400,
sigma=.1, dest_x=-108.6, dest_y=26.2, mot_x=.9, mot_y=.9,
modeled_species=pabu.pop,
optimum_lo=.6,optimum_hi=.8,init_energy=100,
direction="S",write_results=FALSE,single_rast=FALSE,mortality = TRUE)

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")
```

`get_ex_data` *Downloads data that is used in examples in vignette and documentation.*

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

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` *Runs basic Brownian / Ornstein Uhlenbeck agent-based model for multiple replicates.*

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

`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`	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 = 5, days = 27, env_rast = ndvi_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 = TRUE, single_rast = FALSE, mortality = T
)

# 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)

Examples

```
# 1. Run moveSIM()

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

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

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

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

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.

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?

Index

abmr, 2
as.species, 2

diseaseSIM, 4
diseaseVIZ, 6

energySIM, 7
energyVIZ, 9

get_ex_data, 10

moveSIM, 11
moveVIZ, 13

tidy_results, 14