Package 'paleopop'

January 7, 2025

Type Package

Title Pattern-Oriented Modeling Framework for Coupled Niche-Population Paleo-Climatic Models

Version 2.1.7

Maintainer July Pilowsky <pilowskyj@caryinstitute.org>

URL https://github.com/GlobalEcologyLab/paleopop/

BugReports https://github.com/GlobalEcologyLab/paleopop/issues

Description This extension of the poems pattern-oriented modeling (POM) framework provides a collection of modules and functions customized for paleontological time-scales, and optimized for single-generation transitions and large populations, across multiple generations.

Depends R (>= 3.6.0)

License GPL (>= 3)

Encoding UTF-8

Language en-AU

LazyData true

RoxygenNote 7.3.2

Imports poems (>= 1.0.0), R6 (>= 2.5.0), sf (>= 0.9), trend (>= 1.1.4)

Collate data.R PaleoPopModel.R PaleoPopResults.R PaleoRegion.R paleopop_simulator.R paleopop-package.R region_subset.R

Suggests testthat (>= 3.0.0), knitr, rmarkdown, markdown, dplyr, raster

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Sean Haythorne [aut], July Pilowsky [aut, cre] (<https://orcid.org/0000-0002-6376-2585>), Stuart Brown [aut] (<https://orcid.org/0000-0002-0669-1418>), Damien Fordham [aut] (<https://orcid.org/0000-0003-2137-5592>)

Repository CRAN

Date/Publication 2025-01-07 16:20:02 UTC

Contents

`bison_hs_raster` *Bison vignette habitat suitability raster*

Description

A *raster* dataset defining estimated habitat suitability values for each grid cells of the Siberian study region of the bison example vignette.

Usage

```
bison_hs_raster
```

Format

A *raster::RasterStack* object:

dimensions 21 rows by 180 columns by 1001 layers

resolution 2 by 2 degree grid cells

extent longitude -180 to 180 degrees; latitude 42 to 84 degrees

values Estimated habitat suitability values of 0 to 1

Source

TBA

paleopop *paleopop: Ensemble population modeling and simulation on paleo time scales*

Description

The paleopop package is an extension of the poems framework of R6 classes, which simulate populations on a dynamic landscape and validate the results via pattern-oriented modeling. paleopop adds functionality for modeling populations over paleo time scales.

Details

The new functions and R6 classes added by paleopop to the poems framework are:

paleopop_simulator function: Analogous to the population_simulator function in poems, this is the engine of simulation in paleopop, handling input parameters, simulating over long time scales, and outputting up to six different types of results.

- PaleoRegion class: Inherited from Region, this class defines a geographic region that changes over time, creating a temporal mask that defines which cells are occupiable at a time step.
- region_subset function: a utility function for subsetting regions defined by coordinates.
- PaleoPopModel class: Inherited from SimulationModel, this class encapsulates the input parameters utilized by the paleopop_simulator.
- PaleoPopResults class: Inherited from SimulationResults, this class encapsulates the results generated by the paleopop_simulator, as well as dynamically generating additional derived results.

PaleoPopModel *R6 class representing a population model for the paleopop simulator*

Description

R6 class representing a spatially-explicit demographic-based population model. It extends the poems class with parameters for the paleopop_simulator. It inherits functionality for creating a nested model, whereby a nested template model with fixed parameters is maintained when a model is cloned for various sampled parameters. Also provided are extensions to the methods for checking the consistency and completeness of model parameters.

Super classes

poems::GenericClass -> poems::GenericModel -> poems::SpatialModel -> poems::SimulationModel -> PaleoPopModel

Public fields

attached A list of dynamically attached attributes (name-value pairs).

Active bindings

`simulation_function` Name (character string) or source path of the default simulation function, which takes a model as an input and returns the simulation results.

`model_attributes` A vector of model attribute names.

`region` A `Region` (or inherited class) object specifying the study region.

`coordinates` Data frame (or matrix) of X-Y population coordinates (WGS84) in longitude (degrees West) and latitude (degrees North).

`random_seed` Number to seed the random number generation for stochasticity.

`time_steps` Number of simulation time steps.

`years_per_step` Number of years per time step.

`populations` Number of population cells.

`initial_abundance` Array (matrix) of initial abundance values at each population cell.

`transition_rate` Rate (numeric) of transition between generations at each time-step.

`standard_deviation` Standard deviation (numeric) for applying environmental stochasticity to transition rates.

`compact_decomposition` List containing a compact transposed (Cholesky) decomposition *matrix* (t_decomposition_compact_matrix) and a corresponding *map* of population indices (t_decomposition_compact_map), as per `SpatialCorrelation` class attributes.

`carrying_capacity` Array (or matrix) of carrying capacity values at each population cell (across time).

`density_dependence` Density dependence type ("competition", "logistic", or "ceiling").

`growth_rate_max` Maximum growth rate (utilized by density dependence processes).

`dispersal_data` List of data frames of non-zero dispersal rates and indices for constructing a compact dispersal matrix, and optional changing rates over time, as per class `DispersalGenerator` *dispersal_data* attribute.

`dispersal_target_k` Target population carrying capacity threshold for density dependent dispersal.

`harvest` Boolean for utilizing harvesting.

`harvest_max` Proportion harvested per year (annual time scale - not generational).

`harvest_g` The "G" parameter in the harvest function.

`harvest_z` The "Z" parameter in the harvest function.

`harvest_max_n` Maximum density per grid cell.

`human_density` Matrix of human density (fraction) ($populations rows by $time_steps columns).

`abundance_threshold` Abundance threshold (that needs to be exceeded) for each population to persist.

`occupancy_threshold` Threshold for the number of populations occupied (that needs to be exceeded) for all populations to persist.

`results_selection` List of results selection from ("abundance", "ema", "extirpation", "harvested", "occupancy", "human_density").

`attribute_aliases` A list of alternative alias names for model attributes (form: `alias = "attribute"`) to be used with the set and get attributes methods.

`template_model` Nested template model for fixed (non-sampled) attributes for shallow cloning.

`sample_attributes` Vector of sample attribute names (only).

`required_attributes` Vector of required attribute names (only), i.e. those needed to run a simulation.

`error_messages` A vector of error messages encountered when setting model attributes.

`warning_messages` A vector of warning messages encountered when setting model attributes.

Methods

Public methods:

- `PaleoPopModel$list_consistency()`
- `PaleoPopModel$list_completeness()`
- `PaleoPopModel$clone()`

Method `list_consistency()`**:** Returns a boolean to indicate if (optionally selected or all) model attributes (such as dimensions) are consistent.

Usage:

```
PaleoPopModel$list_consistency(params = NULL)
```

Arguments:

`params` Optional array of parameter/attribute names.

Returns: List of booleans (or NAs) to indicate consistency of selected/all attributes.

Method `list_completeness()`**:** Returns a list of booleans (or NAs) for each parameter to indicate attributes that are necessary to simulate the model have been set and are consistent/valid.

Usage:

```
PaleoPopModel$list_completeness()
```

Returns: List of booleans (or NAs) for each parameter to indicate to indicate completeness (and consistency).

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
PaleoPopModel$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
library(poems)
library(raster)
# Ring Island example region
coordinates <- data.frame(x = rep(seq(-178.02, -178.06, -0.01), 5),
                          y = rep(seq(19.02, 19.06, 0.01), each = 5))
```

```
template_raster <- Region$new(coordinates = coordinates)$region_raster # full extent
sealevel_raster <- template_raster
template_raster[][c(7:9, 12:14, 17:19)] <- NA # make Ring Island
sealevel_raster[][c(7:9, 12:14, 17:18)] <- NA
raster_stack <- raster::stack(x = append(replicate(9, template_raster), sealevel_raster))
region <- PaleoRegion$new(template_raster = raster_stack)

# Model template
template_model <- PaleoPopModel$new(simulation_function = "paleopop_simulator", # the default
                                    region = region, years_per_step = 25, # default: 1 year
                                      time_steps = 10)
template_model$required_attributes # more requirements than the SimulationModel object in poems
template_model$is_complete() # the required attributes have not been filled in
template_model#is_consistent() # however, the attributes that are filled in are consistent
```

`PaleoPopResults` *R6 class representing paleopop simulator results.*

Description

R6 class for encapsulating and dynamically generating spatially-explicit `paleopop_simulator` results, as well as optional re-generated `Generator` for niche carrying capacity and/or human density.

Super classes

`poems::GenericClass` -> `poems::GenericModel` -> `poems::SpatialModel` -> `poems::SimulationResults` -> `PaleoPopResults`

Public fields

`attached` A list of dynamically attached attributes (name-value pairs).

Active bindings

`model_attributes` A vector of model attribute names.

`region` A `Region` (or inherited class) object specifying the study region.

`coordinates` Data frame (or matrix) of X-Y population coordinates (WGS84) in longitude (degrees West) and latitude (degrees North).

`time_steps` Number of simulation time steps.

`burn_in_steps` Optional number of initial 'burn-in' time steps to be ignored.

`occupancy_mask` Optional binary mask array (matrix), data frame, or raster (stack) for each cell at each time-step of the simulation including burn-in.

`trend_interval` Optional time-step range (indices) for trend calculations (assumes indices begin after the burn-in when utilized).

`abundance` Matrix of population abundance across simulation time-steps (*populations* rows by *duration* columns).

`abundance_trend` Trend or average Sen's `slope` of total abundance (optionally across a time-step interval).

`ema` Matrix of population expected minimum abundance (EMA) across simulation time-steps (*populations* rows by *duration* columns).

`extirpation` Array of population extirpation times.

`extinction_location` The weighted centroid of cells occupied in the time-step prior to the extirpation of all populations (if occurred).

`harvested` Matrix of the number of animals harvested from each population at each time-step (*populations* rows by *duration* columns).

`occupancy` Array of the number of populations occupied at each time-step.

`carrying_capacity` Optional matrix of simulation input carrying capacity to be combined with results (*populations* rows by *duration* columns).

`human_density` Optional matrix of simulation input human density to be combined with results (*populations* rows by *duration* columns).

`all` Nested simulation results for all cells.

`parent` Parent simulation results for individual cells.

`default` Default value/attribute utilized when applying primitive metric functions (e.g. max) to the results.

`attribute_aliases` A list of alternative alias names for model attributes (form: `alias = "attribute"`) to be used with the set and get attributes methods.

`error_messages` A vector of error messages encountered when setting model attributes.

`warning_messages` A vector of warning messages encountered when setting model attributes.

Methods

Public methods:

- `PaleoPopResults$clone()`

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
PaleoPopResults$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
library(raster)
library(poems)
# Ring Island example region
coordinates <- data.frame(x = rep(seq(-178.02, -178.06, -0.01), 5),
                          y = rep(seq(19.02, 19.06, 0.01), each = 5))
template_raster <- Region$new(coordinates = coordinates)$region_raster # full extent
sealevel_raster <- template_raster
template_raster[][c(7:9, 12:14, 17:19)] <- NA # make Ring Island
```

```
sealevel_raster[][c(7:9, 12:14, 17:18)] <- NA
raster_stack <- raster::stack(x = append(replicate(9, template_raster), sealevel_raster))
region <- PaleoRegion$new(template_raster = raster_stack)

# Model template
model_template <- PaleoPopModel$new(
region = region,
time_steps = 10,
years_per_step = 12, # years per generational time-step
standard_deviation = 0.1,
growth_rate_max = 0.6,
harvest = FALSE,
populations = region$region_cells,
initial_abundance = seq(9000, 0, -1000),
transition_rate = 1.0,
carrying_capacity = rep(1000, 17),
dispersal = (!diag(nrow = 17, ncol = 17))*0.05,
density_dependence = "logistic",
dispersal_target_k = 10,
occupancy_threshold = 1,
abundance_threshold = 10,
results_selection = c("abundance")
)

# Simulations
results <- paleopop_simulator(model_template)

# Results
results_model <- PaleoPopResults$new(results = results, region = region, time_steps = 10)
results_model$extirpation # cells where the population goes to zero are marked 1
results_model$occupancy # indicates with 0 and 1 which cells are occupied at each time step
results_model$ema # expected minimum abundance
```

`paleopop_simulator` *Runs a customized population model simulation.*

Description

Simulates a population model customized for paleontological time-scales, optimized for single-generation transitions and large populations, across multiple generations and returns simulation results. Each generational time-step includes:

1. Density dependence calculations
2. Environmental stochasticity calculations
3. Generational transition calculations
4. Harvest calculations
5. Dispersal calculations
6. Results collection

Usage

```
paleopop_simulator(inputs)
```

Arguments

`inputs` Nested list/object with named elements:

`random_seed` Number to seed the random number generation for stochasticity.

`time_steps` Number of simulation time steps.

`years_per_step` Number of years per time step.

`populations` Number of populations.

`initial_abundance` Array of initial abundances for each population.

`transition_rate` Rate of transition (or fecundity) between generations.

`standard_deviation` Standard deviation applied to transition rates.

`compact_decomposition` List containing a compact transposed (Cholesky) decomposition *matrix* (t_decomposition_compact_matrix) and a corresponding *map* of population indices (t_decomposition_compact_map), as per `SpatialCorrelation` class attributes.

`carrying_capacity` Matrix of carrying capacities (*populations* rows by *time_steps* columns).

`density_dependence` Density dependence type ("competition", "logistic", or "ceiling").

`growth_rate_max` Maximum growth rate (for "competition" or "logistic" density dependence).

`harvest` Boolean for utilizing harvesting.

`harvest_max` Proportion harvested per year (note: annual time scale - not generational).

`harvest_g` The *G* parameter in the harvest function.

`harvest_z` The *Z* parameter in the harvest function.

`harvest_max_n` Maximum density per grid cell.

`human_density` Matrix of human density (fraction) (*populations* rows by *time_steps* columns).

`dispersal_data` List of data frames of non-zero dispersal rates and indices for constructing a compact dispersal matrix, and optional changing rates over time, as per class `DispersalGenerator` *dispersal_data* attribute.

`dispersal_target_k` Target population carrying capacity threshold for density dependent dispersal.

`abundance_threshold` Abundance threshold (that needs to be exceeded) for each population to persist.

`occupancy_threshold` Threshold for the number of populations occupied (that needs to be exceeded) for all populations to persist.

`results_selection` List of results selection from: "abundance", "ema", "extirpation", "harvested", "occupancy", "human_density".

Value

Simulation results as a nested list (as selected):

`abundance` Matrix of simulation abundances (*populations* rows by *time_steps* columns).

`ema` Matrix of expected minimum abundances (*populations* rows by *time_steps* columns).

`extirpation` Array of extirpation times for each population.

`harvested` Matrix of estimated individuals harvested (*populations* rows by *time_steps* columns).

`occupancy` Array of number of populations occupied at each time-step.

`human_density` Matrix of human densities, (*populations* rows by *time_steps* columns).

Examples

```
library(raster)
library(poems)
# Ring Island example region
coordinates <- data.frame(x = rep(seq(-178.02, -178.06, -0.01), 5),
                          y = rep(seq(19.02, 19.06, 0.01), each = 5))
template_raster <- Region$new(coordinates = coordinates)$region_raster # full extent
sealevel_raster <- template_raster
template_raster[][c(7:9, 12:14, 17:19)] <- NA # make Ring Island
sealevel_raster[][c(7:9, 12:14, 17:18)] <- NA
raster_stack <- raster::stack(x = append(replicate(9, template_raster), sealevel_raster))
region <- PaleoRegion$new(template_raster = raster_stack)

# Model template
model_template <- PaleoPopModel$new(
region = region,
time_steps = 10,
years_per_step = 12, # years per generational time-step
standard_deviation = 0.1,
growth_rate_max = 0.6,
harvest = FALSE,
populations = region$region_cells,
initial_abundance = seq(9000, 0, -1000),
transition_rate = 1.0,
carrying_capacity = rep(1000, 17),
dispersal = (!diag(nrow = 17, ncol = 17))*0.05,
density_dependence = "logistic",
dispersal_target_k = 10,
occupancy_threshold = 1,
abundance_threshold = 10,
results_selection = c("abundance")
)

# Simulations
results <- paleopop_simulator(model_template) # input as PaleoPopModel object
inputs <- model_template$get_attributes()
paleopop_simulator(inputs) # input as list of attributes
```

PaleoRegion *R6 class representing a paleontological region.*

Description

R6 class representing a study region of temporally changing spatial grid cells, defined via a *RasterLayer* object (see raster) and a temporal mask indicating which cells are included at each time step.

Super classes

poems::GenericClass -> poems::Region -> PaleoRegion

Public fields

attached A list of dynamically attached attributes (name-value pairs).

Active bindings

coordinates Data frame (or matrix) of X-Y population (WGS84) coordinates in longitude (degrees West) and latitude (degrees North) (get and set), or distance-based coordinates dynamically returned by region raster (get only).

region_raster A *RasterLayer* object (see raster) defining the region with finite values (NAs elsewhere).

use_raster Boolean to indicate that a raster is to be used to define the region (default TRUE).

strict_consistency Boolean to indicate that, as well as resolution, extent and CRS, consistency checks also ensure that a raster's finite/occupiable cells are the same or a subset of those defined by the region (default TRUE).

temporal_mask Matrix of booleans indicating which region cells are included at each time step.

region_cells Dynamically calculated number of region coordinates or raster cells with finite/non-NA values.

region_indices Dynamically calculated region indices for raster cells with finite/non-NA values (all if not a raster).

Methods

Public methods:

- PaleoRegion$new()
- PaleoRegion$raster_from_values()
- PaleoRegion$temporal_mask_raster()
- PaleoRegion$clone()

Method new(): Initialization method sets temporally changing raster layers for paleontological region.

Usage:

```
PaleoRegion$new(template_raster = NULL, remove_zeros = FALSE, ...)
```

Arguments:

`template_raster` A *RasterLayer*, *RasterBrick*, or *RasterStack* object (see `raster`) defining the paleontological region with example finite values (NAs elsewhere)

`remove_zeros` Boolean to indicate that cells that are zero across all layers (times) are to be removed, i.e. set to NA (default is FALSE).

`...` Additional parameters passed individually.

Method `raster_from_values()`**:** Converts an array (or matrix) of values into a raster (or stack) consistent with the region raster (matching extent, resolution, and finite/NA cells), and with the temporal mask (if any) applied.

Usage:

```
PaleoRegion$raster_from_values(values)
```

Arguments:

`values` An array (or matrix) of values to be placed in the raster (or stack) having dimensions consistent with the region cell number.

Returns: A *RasterLayer* (or *RasterStack/Brick*) object consistent with the region raster with temporal mask (if any) applied.

Method `temporal_mask_raster()`**:** Returns the temporal mask as a raster stack/brick object consistent with the region raster.

Usage:

```
PaleoRegion$temporal_mask_raster()
```

Returns: A *RasterStack/Brick*) object with temporal mask values of 1 (true) and NA elsewhere.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
PaleoRegion$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
library(poems)
library(raster)
# Ring Island example region
coordinates <- data.frame(x = rep(seq(-178.02, -178.06, -0.01), 5),
                          y = rep(seq(19.02, 19.06, 0.01), each = 5))
template_raster <- Region$new(coordinates = coordinates)$region_raster # full extent
sealevel_raster <- template_raster
template_raster[][c(7:9, 12:14, 17:19)] <- NA # make Ring Island
sealevel_raster[][c(7:9, 12:14, 17:18)] <- NA
raster_stack <- raster::stack(x = append(replicate(9, template_raster), sealevel_raster))
region <- PaleoRegion$new(template_raster = raster_stack)
raster::plot(region$temporal_mask_raster()[[1]], main = "Ring Island (first timestep)",
             xlab = "Longitude (degrees)", ylab = "Latitude (degrees)",
```

```
            colNA = "blue", legend = FALSE)
raster::plot(region$temporal_mask_raster()[[10]], main = "Ring Island (last timestep)",
            xlab = "Longitude (degrees)", ylab = "Latitude (degrees)",
            colNA = "blue", legend = FALSE)
```

`region_subset` *Function generates a region subset of matrix values based on a subset of coordinates within the original region (using nearest spatial neighbor if coordinates differ).*

Description

`region_subset` generates a region subset of matrix values based on a subset of coordinates within the original region (using nearest spatial neighbor if coordinates differ).

Usage

```
region_subset(orig_coords = NULL, orig_matrix = NULL, subset_coords = NULL)
```

Arguments

`orig_coords`	Data frame (or matrix) of original/full region of X-Y coordinates (WGS84) in longitude (degrees West) and latitude (degrees North).
`orig_matrix`	Matrix of original values with rows corresponding to the original/full region coordinates.
`subset_coords`	Data frame (or matrix) of X-Y subset region coordinates (WGS84) in longitude (degrees West) and latitude (degrees North).

Value

A matrix of values corresponding to the subset region coordinates (using nearest spatial neighbor if original and subset coordinates differ).

Examples

```
coordinates <- data.frame(x = rep(seq(-178.02, -178.06, -0.01), 5),
                          y = rep(seq(19.02, 19.06, 0.01), each = 5))
values <- matrix(seq(1, 25, 1))
subset <- data.frame(x = rep(seq(-178, -178.04, -0.005), 7),
                     y = rep(seq(19.03, 19.06, 0.005), each = 9))
region_subset(coordinates, values, subset) # nearest neighbor interpolation
```

`siberia_raster` *Bison vignette Siberia raster*

Description

A `raster` dataset defining the grid cells of the Siberia study region in a temporally dynamic manner for the bison example vignette.

Usage

```
siberia_raster
```

Format

A *raster::RasterStack* object:

dimensions 21 rows by 180 columns by 1001 layers

resolution 2 by 2 degree grid cells

crs WGS84 latitude longitude

extent longitude -180 to 180 degrees; latitude 42 to 84 degrees

values region defined by 913 cells with value of 1, surrounded by non-region `NA` values

Source

TBA

Index