# Package 'epizootic'

July 22, 2025

**Title** Spatially Explicit Population Models of Disease Transmission in Wildlife

**Version** 1.0.0

**Description** This extension of the pattern-oriented modeling framework of the 'poems' package provides a collection of modules and functions customized for modeling disease transmission on a population scale in a spatiotemporally explicit manner. This includes seasonal time steps, dispersal functions that track disease state of dispersers, results objects that store disease states, and a population simulator that includes disease dynamics.

**License** GPL (>= 3)

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**URL** https://github.com/viralemergence/epizootic

**BugReports** https://github.com/viralemergence/epizootic/issues

**Depends** R (>= 4.3.0)

**Imports** purrr (>= 1.0.0), dplyr (>= 1.1.3), tibble (>= 3.2.1), R6 (>= 2.5.1), cli (>= 3.6.1), raster (>= 3.6), qs (>= 0.25.7), poems (>= 1.1.0), doParallel (>= 1.0.16), foreach (>= 1.5.1), Rcpp

**Suggests** testthat (>= 3.0.0), geosphere (>= 1.5), knitr (>= 1.17), rmarkdown (>= 1.6)

**Config/testthat/edition** 3

**Collate** 'DiseaseModel.R' 'RcppExports.R' 'SimulationHandler.R' 'aspatial_siri_seasons.R' 'check_aspatial_siri_inputs.R' 'check_simulator_inputs.R' 'data.R' 'disease_dispersal.r' 'disease_results.R' 'disease_simulator.R' 'disease_transformation.R' 'disease_transitions.R' 'epizootic-package.R'

**LinkingTo** Rcpp, RcppArmadillo

**VignetteBuilder** knitr

**LazyData** true

**NeedsCompilation** yes

**Author** July Pilowsky [aut, cre] (ORCID:
<https://orcid.org/0000-0002-6376-2585>),
National Science Foundation Biology Integration Institute 2213854 [fnd]

**Maintainer** July Pilowsky <pilowskyj@caryinstitute.org>

**Repository** CRAN

**Date/Publication** 2024-10-02 13:10:05 UTC

# Contents



---

`aspatial_siri` *Helper Function for Seasonal SIRI Simulation*

---

## Description

This function is an internal one that does the aspatial simulations within one population for one timestep, for any given season.

## Usage

```
aspatial_siri(
  initial_pop,
  season_length,
  mortality,
  transmission,
  recovery,
  fecundity,
  abundance_threshold,
```

```
  carrying_capacity,
  season
)
```

## Arguments

| | |
|---|---|
| initial_pop | A vector of length 8 showing the initial abundance for each combination of stage and compartment. |
| season_length | The length of the season in days. |
| mortality | A vector of length 8 with the mortality rates for each stage and compartment in the season in question. |
| transmission | A vector of length 8 with the transmission rates for each stage in the season in question. |
| recovery | A vector of length 8 with the recovery rates for each infected stage in the season in question. |
| fecundity | A vector of length 8 with the fecundity for each reproductive segment. |
| abundance_threshold | A quasi-extinction threshold below which a population becomes extinct. |
| carrying_capacity | A single numeric that indicates the carrying capacity of the population in this season. |
| season | Either "breeding" or "non-breeding." |

## Value

A vector of length 8 showing the abundance for each combination of stage and compartment at the end of the season.

## Examples

```
aspatial_siri(
 initial_pop = c(50000, 50000, 0, 1, 0, 0, 0, 0),
 season_length = 100,
 mortality = c(0.004, 0, 0.00505, 0.00105, 0.004, 0, 0.0045, 5e-04),
 fecundity = c(0, 15/182, 0, 15/182, 0, 15/182, 0, 15/182),
 transmission = c(0.00002, 0.00001, 0, 0, 7.84e-06, 3.92e-06, 0, 0),
 recovery = c(0, 0, 0.05714286, 0.05714286, 0, 0, 0.1, 0.1),
 carrying_capacity = 150000,
 abundance_threshold = 10,
 season = "breeding"
)
```

## bsl_raster *Raster of breeding season length for the house finch*

### Description

This is a RasterBrick object containing data on the breeding season length in days of the house finch in North America from 1994 to 2016. I created this dataset using a machine learning algorithm on season length data from the eastern bluebird, which is noted to have a very similar breeding season to the house finch. The raster has the same resolution as the `finch_region`.

### Usage

```
bsl_raster
```

### Format

A RasterBrick with 17066 cells and 23 layers.

## check_aspatial_siri_inputs *Helper function to check the validity of inputs to the* `siri_model_summer` *and* `siri_model_winter` *functions.*

### Description

This is an internal function that checks inputs to the `disease_simulator` function to make sure they are valid, and sets default values for needed inputs if their values are not supplied. The possible inputs for this function are the same as the possible inputs to the `disease_simulator` function.

### Usage

```
check_aspatial_siri_inputs(inputs)
```

### Arguments

| | |
|---|---|
| `inputs` | A nested list with named elements: |
| | `replicates` Number of replicate simulation runs. |
| | `time_steps` Number of simulation time steps. |
| | `populations` Number of populations. |
| | `stages` Number of life cycle stages. |
| | `compartments` Number of disease compartments (e.g., 3 for a SIR model). |
| | `abundance_threshold` A quasi-extinction threshold at which a population becomes extinct. |

`mortality` A vector of mortality rates, one for each combination of stages and compartments.

`mortality_unit` A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily.

`fecundity` A vector of fecundity rates, one for each combination of stages and compartments for which fecundity applies.

`fecundity_unit` A vector indicating whether fecundity rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily.

`fecundity_mask` A vector indicating which stages and compartments reproduce.

`transmission` A vector of transmission rates, one for each combination of stages and compartment for which transmission applies (see `transmission_mask` below.

`transmission_unit` A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily.

`transmission_mask` A vector indicating which stages and compartments are subject to transmission (i.e., classes susceptible to infection.)

`recovery` A vector of recovery rates, one for each combination of stages and compartment for which recovery applies (see `recovery_mask` below.)

`recovery_unit` A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily.

`recovery_mask` A vector indicating which compartments are subject to recovery (i.e., infected classes that can recover.)

`r` Simulation replicate.

`tm` Simulation time step.

`carrying_capacity` Array of carrying capacity values for each population at time step.

`breeding_season_length` Array of breeding season lengths in days for each population at time step.

`segment_abundance` Matrix of (current) abundance for each stage-compartment combo (rows) and population (columns) at time step.

`occupied_indices` Array of indices for populations occupied at (current) time step.

`simulator` `poems::SimulatorReference` object with dynamically accessible *attached* and *results* lists.

`additional attributes` Additional attributes when the transformation is optionally nested in a list.

## Value

A list identical to the inputs (if there are no errors.)

check_simulator_inputs *Helper function to check the validity of inputs to the* disease_simulator *function.*

## Description

This is an internal function that checks inputs to the disease_simulator function to make sure they are valid, and sets default values for needed inputs if their values are not supplied. The possible inputs for this function are the same as the possible inputs to the disease_simulator function.

## Usage

```
check_simulator_inputs(inputs)
```

## Arguments

inputs Nested list/object with named elements:

random_seed Number to seed the random number generation for stochasticity.

replicates Number of replicate simulation runs (default is 1.)

time_steps Number of simulation years. Required input.

seasons Number of seasons per year (default is 2.)

populations Number of populations. Required input.

coordinates Data frame (or matrix) of X-Y population coordinates.

stages Number of life cycle stages. Default: 1.

compartments Number of disease compartments (e.g., 3 for a SIR model). Default: 1.

region A poems::Region object defining the study region.

initial_abundance Array (or matrix) of initial abundances. There must be one column per population and one row per compartment/stage combination. By default, this should be in the order compartment by stage, e.g., 2 stage classes plus a SI model should be ordered as S1, S2, I1, I2. If a region object is attached, then initial abundance may be provided in the form of a raster with the same specs as the region raster and one layer per stage/compartment combination. If there is only one stage/compartment combination you may provide a vector with length populations. Required input.

carrying_capacity Array (matrix) of carrying capacity values at each population cell (populations rows by time_steps columns when across time). Required input.

breeding_season_length Array (matrix) of breeding season length values in days at each population cell (populations rows by time_steps columns when across time). Can also be a vector of length populations if the breeding season length does not change over time.

season_lengths Vector of season lengths in days. Length must equal seasons. If neither breeding_season_length nor season_lengths are provided, season lengths will default to 365/seasons.

correlation List containing either an environmental correlation matrix (correlation_matrix), a pre-calculated transposed (Cholesky) decomposition matrix (t_decomposition_matrix), or a compact transposed (Cholesky) decomposition matrix (t_decomposition_compact_matrix) and a corresponding map of population indices (t_decomposition_compact_map), as per poems::SpatialCorrelation class attributes.

mortality A vector of mortality rates, one for each combination of stages and compartments. Assumed by default to be daily mortality rates unless indicated otherwise (see below). If mortality varies by season, a list of mortality vectors with the same length as seasons may be provided instead. Required input.

mortality_unit A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily. Default: all 0. A list of vectors may be provided if this varies by season.

fecundity A vector of fecundity rates, one for each combination of stages and compartments for which fecundity applies (see fecundity_mask below). If fecundity varies among seasons, a list of fecundity vectors with the same length as seasons may be provided. Required input.

fecundity_unit A vector indicating whether fecundity rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily. Default: all 0. A list of vectors may be provided if this varies by season.

fecundity_mask A vector indicating which stages and compartments reproduce. Must be the same length as stages * compartments. A list of vectors may be provided if this varies by season. If no fecundity mask is provided, then it is assumed that all stages and compartments reproduce.

abundance_threshold A quasi-extinction threshold at which a population becomes extinct. Default: 0.

demographic_stochasticity Boolean for choosing demographic stochasticity for transition, dispersal, and/or other processes (default is TRUE).

transmission A vector of transmission rates, one for each combination of stages and compartment for which transmission applies (see transmission_mask below). If transmission varies by season, a list of transmission vectors with the same length as seasons may be provided instead. Required input.

transmission_unit A vector indicating whether transmission is daily or seasonal. 1 indicates seasonal, 0 indicates daily. Default: all 0. A list of vectors may be provided if this varies by season.

transmission_mask A vector indicating which stages and compartments are subject to transmission (i.e., classes susceptible to infection.) Must be the same length as compartments. A list of vectors may be provided if this varies by season. If no transmission mask is provided, then it is assumed that all stages in the first compartment are susceptible to infection.

recovery A vector of recovery rates, one for each combination of stages and compartment for which recovery applies (see recovery_mask below.) If

recovery varies by season, a list of recovery vectors the same length as `seasons` may be provided instead.

`recovery_unit` A vector indicating whether recovery rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily. Default: all 0. A list of vectors may be provided if this varies by season.

`recovery_mask` A vector indicating which compartments are subject to recovery (i.e., infected classes that can recover.) Must be the same length as `compartments`. A list of vectors may be provided if this varies by season. If no recovery mask is provided, then it is assumed that all stages in the second compartment can recover, if there is a second compartment.

`dispersal` A list that is either length 1 or the same length as `stages`. If it is length 1, the same dispersal will be applied across all stages. Within each element of the list, there should be either a function, a matrix of dispersal rates between populations (source columns to target rows) or a 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 `poems::DispersalGenerator` *dispersal_data* attribute).

`dispersal_source_n_k` Dispersal proportion (p) density dependence via source population abundance divided by carrying capacity (n/k), where p is reduced via a linear slope (defined by two list items) from n/k <= *cutoff* (p = 0) to n/k \>= *threshold* (aliases: *dispersal_n_k_cutoff* & *dispersal_n_k_threshold*).

`dispersal_target_k` Dispersal rate (r) density dependence via target population carrying capacity (k), where r is reduced via a linear slope (through the origin) when k <= *threshold* (alias: *dispersal_k_threshold*).

`dispersal_target_n` Dispersal rate (r) density dependence via target population abundance (n), where r is reduced via a linear slope (defined by two list items) from n \>= *threshold* to n <= *cutoff* (r = 0) or vice versa (aliases: *dispersal_n_threshold* & *dispersal_n_cutoff*).

`dispersal_target_n_k` Dispersal rate (r) density dependence via target population abundance divided by carrying capacity (n/k), where r is reduced via a linear slope (defined by two list items) from n/k \>= *threshold* to n/k <= *cutoff* (r = 0) or vice versa.

`season_functions` A list of population transformation functions (functions that change abundance across stages and compartments) the same length as `seasons`. The function must be in the form `function(params)`, where `params` is a list passed to the function containing:

`replicates` Number of replicate simulation runs (default is 1.)

`time_steps` Number of simulation years. Required input.

`seasons` Number of seasons per year (default is 2.)

`populations` Number of populations. Required input.

`stages` Number of life cycle stages. Default: 1.

`compartments` Number of disease compartments (e.g., 3 for a SIR model). Default: 1.

`breeding_season_length` Array (matrix) of breeding season length values in days at each population cell (`populations` rows by `time_steps` columns when across time).

season_lengths Vector of season lengths in days. Length must equal seasons.

mortality A vector of mortality rates, one for each combination of stages and compartments. Assumed by default to be daily mortality rates. Required input.

mortality_unit A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily. Default: all 0.

fecundity A vector of fecundity rates, one for each combination of stages and compartments for which fecundity applies (see fecundity_mask below). Required input.

fecundity_unit A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily. Default: all 0.

fecundity_mask A vector indicating which stages and compartments reproduce. Must be the same length as stages * compartments.

abundance_threshold A quasi-extinction threshold below which a population becomes extinct. Default: 0.

demographic_stochasticity Boolean for choosing demographic stochasticity for transition, dispersal, and/or other processes (default is TRUE).

transmission A vector of transmission rates, one for each combination of stages and compartments. Assumed by default to be daily transmission rates. Required input.

transmission_unit A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily. Default: all 0.

recovery A vector of recovery rates, one for each combination of stages and compartment for which recovery applies (see recovery_mask below.)

recovery_unit A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily. Default: all 0.

recovery_mask A vector indicating which compartments are subject to recovery (i.e., infected classes that can recover.) Must be the same length as compartments.

r Simulation replicate.

tm Simulation time step.

carrying_capacity Array of carrying capacity values for each population at time step.

segment_abundance Matrix of abundance for each combination of stage and compartment (rows) and population (columns) at time step.

occupied_indices Array of indices for populations occupied at time step.

simulator poems::SimulatorReference object with dynamically accessible *attached* and *results* lists.

additional attributes Additional attributes when the transformation is optionally nested in a list.

and returns a transformed stage abundance matrix.

simulation_order A list the same length as seasons. Each element in the list is a vector of named simulation processes in the desired order. Processes must be one of "transition", "dispersal", "season_functions", or "results."

`dispersal_type` A character vector that may contain "pooled" (if all individuals disperse the same), "stages", "compartments", or "segments", if different stages, compartments, or stage-compartment combinations disperse differently. If "pooled" is chosen, `dispersal` must be a list of length 1. If "stages" is chosen, it must be the same length as `stages`, if "compartments" is chosen, it must be the same length as `compartments`, and if "segments" is chosen, it must be the same length as stages*compartments. The default value is "pooled".

`results_selection` List of results selection from: "abundance" (default), "ema", "extirpation", "extinction_location", "occupancy"; "summarize" (default) or "replicate".

`results_breakdown` A string with one of these values: "segments" (default), "compartments", "stages" or "pooled." "segments" returns results for each segment (stage x compartment combination.) "compartments" returns results for each disease compartment. "stages" returns results for each life cycle stage. "pooled" returns results that are not broken down by stage or compartment.

`verbose` TRUE or FALSE, indicating if the user wants informative messages throughout the simulation process.

## Value

A list identical to the inputs, except with default values supplied to fill in any crucial missing values, as explained in the documentation above.

---

`DiseaseModel` *R6 class representing a disease model of the* Mycoplasma gallisepticum *outbreak in* Haemorhous mexicanus

---

## Description

A `R6::R6Class` class representing fixed settings for a spatially-explicit demographic-based SIRI model of disease dynamics. It extends the `poems::SimulationModel` class with parameters for the `disease_simulator` function. 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` -> `DiseaseModel`

## 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 `poems::Region` (or inherited class) object specifying the study region.

`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).

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

`replicates` Number of replicate simulation runs.

`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) or raster (stack) of initial abundance values at each population cell (for each age/stage).

`demographic_stochasticity` Boolean for choosing demographic stochasticity for transition, dispersal, harvest and/or other processes.

`standard_deviation` Standard deviation matrix (or single value) for applying environmental stochasticity to transition rates.

`correlation` Simulator-dependent attribute or list of attributes for describing/parameterizing the correlation strategy utilized when applying environmental stochasticity and/or other processes (see `poems::population_simulator`).

`stages` Number of life cycle stages (default 1).

`compartments` Number of disease compartments (default 1).

`results_breakdown` A string with one of these values: "segments" (default), "compartments", "stages" or "pooled." "segments" returns results for each segment (stage x compartment combination.) "compartments" returns results for each disease compartment. "stages" returns results for each life cycle stage. "pooled" returns results that are not broken down by stage or compartment.

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

`density_dependence` Simulator-dependent function, attribute or list of attributes for describing/parameterizing the density dependence strategy utilized (see `poems::population_simulator`).

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

`fecundity` A vector of fecundity rates, one for each combination of stages and compartments for which fecundity applies (see `fecundity_mask` below). If fecundity varies among seasons, a list of fecundity vectors with the same length as `seasons` may be provided. Required input.

`density_stages` Array of booleans or numeric (0-1) for each stage to indicate (the degree to) which stages are affected by density (default is 1 for all stages).

`translocation` Simulator-dependent function, attribute or list of attributes for describing/parameterizing translocation (management) strategies utilized (see `poems::population_simulator`).

`harvest` Simulator-dependent function, attribute or list of attributes for describing/parameterizing a harvest (organism removal/hunting) strategy (see `poems::population_simulator`).

`mortality` Simulator-dependent function, attribute or list of attributes to describe/parameterize a spatio-temporal mortality strategy (see `poems::population_simulator`).

`dispersal` Simulator-dependent function, attribute or list of attributes for describing/parameterizing the dispersal (migration) strategy utilized (see `disease_simulator`).

`dispersal_stages` Array of relative dispersal (0-1) for each stage to indicate the degree to which each stage participates in dispersal (default is 1 for all stages).

`dispersal_source_n_k` Simulator-dependent attribute for describing/parameterizing dispersal dependent on source population abundance divided by carrying capacity (see `disease_simulator`).

`dispersal_target_k` Simulator-dependent attribute for describing/parameterizing dispersal dependent on target population carrying capacity (see `disease_simulator`).

`dispersal_target_n` Simulator-dependent attribute (default is list with *threshold* and *cutoff*) of attributes for describing/parameterizing dispersal dependent on target population abundance (see `disease_simulator`).

`dispersal_target_n_k` Simulator-dependent attribute (default is list with *threshold* and *cutoff*) of attributes for describing/parameterizing dispersal dependent on target population abundance/capacity (see `poems::population_simulator`).

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

`seasons` Number of seasons in a year (default 1.) The first one is always treated as the breeding season.

`simulation_order` A vector of simulation process names in configured order of execution.

`results_selection` List of attributes to be included in the returned results of each simulation run, selected from: "abundance", "ema", "extirpation", "extinction_location", "harvested", "occupancy"; "summarize" or "replicate".

`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:

- `DiseaseModel$new()`
- `DiseaseModel$set_sample_attributes()`
- `DiseaseModel$list_consistency()`
- `DiseaseModel$clone()`

**Method** `new()`**:** Initialization method sets default aliases and given attributes individually and/or from a list.

*Usage:*

```
DiseaseModel$new(attribute_aliases = NULL, ...)
```

*Arguments:*

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

`...` Parameters passed via a *params* list or individually.

**Method** `set_sample_attributes()`**:** Sets the names (only - when *params* is a vector) and values (when *params* is a list and/or when name-value pairs are provided) of the sample attributes for the model.

*Usage:*

```
DiseaseModel$set_sample_attributes(params = list(), ...)
```

*Arguments:*

`params` List of parameters/attributes (names and values) or array of names only.

`...` Parameters/attributes passed individually.

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

*Usage:*

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

*Usage:*

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

*Arguments:*

`deep` Whether to make a deep clone.

---

`disease_dispersal` *Nested functions for population dispersal in a disease ecology simulation.*

---

## Description

Modular functions for the disease simulator for performing dispersal of segment (stage by compartment) abundance at a specified time step via dispersal rates provided. Dispersal can be handled identically for all stages and compartments, or can be handled differently by stage, compartment, or both.

## Usage

```
disease_dispersal(
  replicates,
  time_steps,
  populations,
  demographic_stochasticity,
  dispersal,
  dispersal_type,
  dispersal_source_n_k = NULL,
  dispersal_target_k = NULL,
  dispersal_target_n = NULL,
  dispersal_target_n_k = NULL,
  stages = NULL,
  compartments = NULL,
  simulator
)
```

## Arguments

`replicates` Number of replicate simulation runs.

`time_steps` Number of simulation time steps.

`populations` Number of populations.

`demographic_stochasticity`
Boolean for optionally choosing demographic stochasticity for the transformation.

`dispersal` Must be a list. Either a list of matrices of dispersal rates between populations (source columns to target rows) or a list of data frames of non-zero dispersal rates and indices for constructing a compact dispersal matrix, and optionally a list of lists of data frames showing changing rates over time (as per class `poems::DispersalGenerator` *dispersal_data* attribute). Alternatively a list of user-defined functions may be used: `function(params)`, where *params* is a list passed to the function containing:

- `replicates` Number of replicate simulation runs.
- `time_steps` Number of simulation time steps.
- `populations` Number of populations.
- `stages` Number of life cycle stages.
- `compartments` Number of disease compartments.
- `dispersal_type` Must be "pooled", "stages", "compartments", or "segments". This indicates whether dispersal should be handled differently by stage and/or compartment.
- `demographic_stochasticity` Boolean for optionally choosing demographic stochasticity for the transformation.
- `dispersal_source_n_k` Dispersal proportion (p) density dependence via source population abundance divided by carrying capacity (n/k), where p is reduced via a linear slope (defined by two list items) from n/k <= *cutoff* (p = 0) to n/k >= *threshold*.

`dispersal_target_k` Dispersal rate (r) density dependence via target population carrying capacity (k), where r is reduced via a linear slope (through the origin) when k <= *threshold*.

`dispersal_target_n` Dispersal rate (r) density dependence via target population abundance (n), where r is reduced via a linear slope (defined by two list items) from n >= *threshold* to n <= *cutoff* (r = 0) or vice versa.

`dispersal_target_n_k` Dispersal rate (r) density dependence via target population abundance divided by carrying capacity (n/k), where r is reduced via a linear slope (defined by two list items) from n/k >= *threshold* to n/k <= *cutoff* (r = 0) or vice versa.

`r` Simulation replicate.

`tm` Simulation time step.

`carrying_capacity` Array of carrying capacity values for each population at time step.

`segment_abundance` Matrix of abundance for each stage by compartment (rows) and population (columns) at time step.

`simulator` `poems::SimulatorReference` object with dynamically accessible *attached* and *results* lists.

returns the post-dispersal abundance matrix

`dispersal_type` Must be "pooled", "stages", "compartments", or "segments". This indicates whether dispersal should be handled differently by stage and/or compartment.

`dispersal_source_n_k`
Dispersal proportion (p) density dependence via source population abundance divided by carrying capacity (n/k), where p is reduced via a linear slope (defined by two list items) from n/k <= *cutoff* (p = 0) to n/k >= *threshold* or vice versa.

`dispersal_target_k`
Dispersal rate (r) density dependence via target population carrying capacity (k), where r is reduced via a linear slope (through the origin) when k <= *threshold*.

`dispersal_target_n`
Dispersal rate (r) density dependence via target population abundance (n), where r is reduced via a linear slope (defined by two list items) from n >= *threshold* to n <= *cutoff* (r = 0) or visa-versa.

`dispersal_target_n_k`
Dispersal rate (r) density dependence via target population abundance divided by carrying capacity (n/k), where r is reduced via a linear slope (defined by two list items) from n/k >= *threshold* to n/k <= *cutoff* (r = 0) or vice versa.

`stages` Number of life cycle stages.

`compartments` Number of disease compartments.

`simulator` `poems::SimulatorReference` object with dynamically accessible *attached* and *results* lists.

## Value

Dispersal function: `function(r, tm, carrying_capacity, segment_abundance)`, where:

`r` Simulation replicate.

`tm` Simulation time step.

`carrying_capacity` Array of carrying capacity values for each population at time step.

`segment_abundance` Matrix of abundance for each stage by compartment (rows) and population (columns) at time step.

`returns` New stage abundance matrix with dispersal applied.

---

`disease_results` *Nested functions for initializing, calculating and collecting disease simulator results.*

---

## Description

Modular functions for the disease simulator for initializing, calculating and collecting simulator results.

## Usage

```
disease_results(
  replicates,
  time_steps,
  seasons,
  stages,
  compartments,
  coordinates,
  initial_abundance,
  results_selection = NULL,
  results_breakdown = NULL
)
```

## Arguments

| | |
|---|---|
| `replicates` | Number of replicate simulation runs. |
| `time_steps` | Number of simulation time steps. |
| `seasons` | Number of seasons per time step. |
| `stages` | Number of life cycle stages. |
| `compartments` | Number of disease compartments. |
| `coordinates` | Data frame (or matrix) of X-Y population coordinates. |
| `initial_abundance` | Matrix of initial abundances at each combination of stage and compartment (in rows) for each population (in columns). |
| `results_selection` | List of results selection from: "abundance" (default), "ema", "extirpation", "extinction_location", "harvested", "occupancy"; "summarize" (default) or "replicate". "summarize" calculates mean, sd, min and max across replicates. "replicate" returns results separately for each replicate. |

`results_breakdown`
: A string with one of these values: "segments" (default), "compartments", "stages" or "pooled." "segments" returns results for each segment (stage x compartment combination.) "compartments" returns results for each disease compartment. "stages" returns results for each life cycle stage. "pooled" returns results that are not broken down by stage or compartment.

## Value

List of result functions:

`initialize_attributes = function())` Constructs and returns an initialized nested list for the selected result attributes.

`initialize_replicate = function(results)` Initializes and returns nested result attributes at the start of each replicate.

`calculate_at_season = function(r, tm, season, segment_abundance, harvested, results)` Appends and calculates (non-NULL) results and returns nested result attributes at the end of each season within time step (tm) within replicate (r).

`calculate_at_timestep = function(r, tm, segment_abundance, harvested, results)` Appends and calculates (non-NULL) results and returns nested result attributes at the end of each time step (tm) within replicate (r).

`finalize_attributes = function(results)` Finalizes result calculations at the end of the simulation.

---

# disease_simulator *Stage-based seasonal spatially explicit population-level disease model.*

---

## Description

Simulates a stage-based demographic population model and returns simulation results across multiple replicate runs. Processes run at each simulation time-step include:

1. Stage transition (stochastic) calculations
2. Population growth/decline calculations
3. Disease outbreak according to a compartmental model
4. Dispersal calculations (default or user-defined)
5. Results collection

Note that the breeding season is always treated as the first season.

## Usage

```
disease_simulator(inputs)
```

## Arguments

`inputs` Nested list/object with named elements:

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

`replicates` Number of replicate simulation runs (default is 1.)

`time_steps` Number of simulation years. Required input.

`seasons` Number of seasons per year (default is 2.)

`populations` Number of populations. Required input.

`coordinates` Data frame (or matrix) of X-Y population coordinates.

`stages` Number of life cycle stages. Default: 1.

`compartments` Number of disease compartments (e.g., 3 for a SIR model). Default: 1.

`region` A `poems::Region` object defining the study region.

`initial_abundance` Array (or matrix) of initial abundances. There must be one column per population and one row per compartment/stage combination. By default, this should be in the order compartment by stage, e.g., 2 stage classes plus a SI model should be ordered as S1, S2, I1, I2. If a region object is attached, then initial abundance may be provided in the form of a raster with the same specs as the region raster and one layer per stage/compartment combination. If there is only one stage/compartment combination you may provide a vector with length `populations`. Required input.

`carrying_capacity` Array (matrix) of carrying capacity values at each population cell (`populations` rows by `time_steps` columns when across time). Required input.

`breeding_season_length` Array (matrix) of breeding season length values in days at each population cell (`populations` rows by `time_steps` columns when across time). Can also be a vector of length `populations` if the breeding season length does not change over time.

`season_lengths` Vector of season lengths in days. Length must equal `seasons`. If neither `breeding_season_length` nor `season_lengths` are provided, season lengths will default to `365/seasons`.

`correlation` List containing either an environmental correlation matrix (correlation_matrix), a pre-calculated transposed (Cholesky) decomposition matrix (t_decomposition_matrix), or a compact transposed (Cholesky) decomposition matrix (t_decomposition_compact_matrix) and a corresponding map of population indices (t_decomposition_compact_map), as per `poems::SpatialCorrelation` class attributes.

`mortality` A vector of mortality rates, one for each combination of stages and compartments. Assumed by default to be daily mortality rates unless indicated otherwise (see below). If mortality varies by season, a list of mortality vectors with the same length as `seasons` may be provided instead. Required input.

`mortality_unit` A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily. Default: all 0. A list of vectors may be provided if this varies by season.

`fecundity` A vector of fecundity rates, one for each combination of stages and compartments for which fecundity applies (see `fecundity_mask` below). If fecundity varies among seasons, a list of fecundity vectors with the same length as `seasons` may be provided. Required input.

`fecundity_unit` A vector indicating whether fecundity rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily. Default: all 0. A list of vectors may be provided if this varies by season.

`fecundity_mask` A vector indicating which stages and compartments reproduce. Must be the same length as `stages * compartments`. A list of vectors may be provided if this varies by season. If no fecundity mask is provided, then it is assumed that all stages and compartments reproduce.

`abundance_threshold` A quasi-extinction threshold at which a population becomes extinct. Default: 0.

`demographic_stochasticity` Boolean for choosing demographic stochasticity for transition, dispersal, and/or other processes (default is TRUE).

`transmission` A vector of transmission rates, one for each combination of stages and compartment for which transmission applies (see `transmission_mask` below. If transmission varies by season, a list of transmission vectors with the same length as `seasons` may be provided instead. Required input.

`transmission_unit` A vector indicating whether transmission is daily or seasonal. 1 indicates seasonal, 0 indicates daily. Default: all 0. A list of vectors may be provided if this varies by season.

`transmission_mask` A vector indicating which stages and compartments are subject to transmission (i.e., classes susceptible to infection.) Must be the same length as `compartments`. A list of vectors may be provided if this varies by season. If no transmission mask is provided, then it is assumed that all stages in the first compartment are susceptible to infection.

`recovery` A vector of recovery rates, one for each combination of stages and compartment for which recovery applies (see `recovery_mask` below.) If recovery varies by season, a list of recovery vectors the same length as `seasons` may be provided instead.

`recovery_unit` A vector indicating whether recovery rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily. Default: all 0. A list of vectors may be provided if this varies by season.

`recovery_mask` A vector indicating which compartments are subject to recovery (i.e., infected classes that can recover.) Must be the same length as `compartments`. A list of vectors may be provided if this varies by season. If no recovery mask is provided, then it is assumed that all stages in the second compartment can recover, if there is a second compartment.

`dispersal` A list that is either length 1 or the same length as `stages`. If it is length 1, the same dispersal will be applied across all stages. Within each element of the list, there should be either a function, a matrix of dispersal rates between populations (source columns to target rows) or a 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 `poems::DispersalGenerator` *dispersal_data* attribute).

`dispersal_source_n_k` Dispersal proportion (p) density dependence via source population abundance divided by carrying capacity (n/k), where p is reduced via a linear slope (defined by two list items) from n/k <= *cutoff* (p = 0) to n/k \>= *threshold* (aliases: *dispersal_n_k_cutoff* & *dispersal_n_k_threshold*).

`dispersal_target_k` Dispersal rate (r) density dependence via target population carrying capacity (k), where r is reduced via a linear slope (through the origin) when k <= *threshold* (alias: *dispersal_k_threshold*).

`dispersal_target_n` Dispersal rate (r) density dependence via target population abundance (n), where r is reduced via a linear slope (defined by two list items) from n \>= *threshold* to n <= *cutoff* (r = 0) or vice versa (aliases: *dispersal_n_threshold* & *dispersal_n_cutoff*).

`dispersal_target_n_k` Dispersal rate (r) density dependence via target population abundance divided by carrying capacity (n/k), where r is reduced via a linear slope (defined by two list items) from n/k \>= *threshold* to n/k <= *cutoff* (r = 0) or vice versa.

`season_functions` A list of population transformation functions (functions that change abundance across stages and compartments) the same length as `seasons`. The function must be in the form `function(params)`, where `params` is a list passed to the function containing:

`replicates` Number of replicate simulation runs (default is 1.)

`time_steps` Number of simulation years. Required input.

`seasons` Number of seasons per year (default is 2.)

`populations` Number of populations. Required input.

`stages` Number of life cycle stages. Default: 1.

`compartments` Number of disease compartments (e.g., 3 for a SIR model). Default: 1.

`breeding_season_length` Array (matrix) of breeding season length values in days at each population cell (`populations` rows by `time_steps` columns when across time).

`season_lengths` Vector of season lengths in days. Length must equal `seasons`.

`mortality` A vector of mortality rates, one for each combination of stages and compartments. Assumed by default to be daily mortality rates. Required input.

`mortality_unit` A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily. Default: all 0.

`fecundity` A vector of fecundity rates, one for each combination of stages and compartments for which fecundity applies (see `fecundity_mask` below). Required input.

`fecundity_unit` A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily. Default: all 0.

`fecundity_mask` A vector indicating which stages and compartments reproduce. Must be the same length as `stages * compartments`.

`abundance_threshold` A quasi-extinction threshold below which a population becomes extinct. Default: 0.

demographic_stochasticity Boolean for choosing demographic stochasticity for transition, dispersal, and/or other processes (default is TRUE).

transmission A vector of transmission rates, one for each combination of stages and compartments. Assumed by default to be daily transmission rates. Required input.

transmission_unit A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily. Default: all 0.

recovery A vector of recovery rates, one for each combination of stages and compartment for which recovery applies (see recovery_mask below.)

recovery_unit A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily. Default: all 0.

recovery_mask A vector indicating which compartments are subject to recovery (i.e., infected classes that can recover.) Must be the same length as compartments.

r Simulation replicate.

tm Simulation time step.

carrying_capacity Array of carrying capacity values for each population at time step.

segment_abundance Matrix of abundance for each combination of stage and compartment (rows) and population (columns) at time step.

occupied_indices Array of indices for populations occupied at time step.

simulator poems::SimulatorReference object with dynamically accessible *attached* and *results* lists.

additional attributes Additional attributes when the transformation is optionally nested in a list.

and returns a transformed stage abundance matrix.

simulation_order A list the same length as seasons. Each element in the list is a vector of named simulation processes in the desired order. Processes must be one of "transition", "dispersal", "season_functions", or "results." "season_functions" will be matched to the appropriate season (i.e., if "season_functions" appears in element 1 of the list, season_functions[[1]] will be called.) If the simulation processes are the same across seasons, then a single character vector may be provided. Required input.

dispersal_type A character vector that may contain "pooled" (if all individuals disperse the same), "stages", "compartments", or "segments", if different stages, compartments, or stage-compartment combinations disperse differently. If "pooled" is chosen, dispersal must be a list of length 1. If "stages" is chosen, it must be the same length as stages, if "compartments" is chosen, it must be the same length as compartments, and if "segments" is chosen, it must be the same length as stages*compartments. The default value is "pooled".

results_selection List of results selection from: "abundance" (default), "ema", "extirpation", "extinction_location", "occupancy"; "summarize" (default) or "replicate".

`results_breakdown` A string with one of these values: "segments" (default), "compartments", "stages" or "pooled." "segments" returns results for each segment (stage x compartment combination.) "compartments" returns results for each disease compartment. "stages" returns results for each life cycle stage. "pooled" returns results that are not broken down by stage or compartment.

`verbose` TRUE or FALSE, indicating if the user wants informative messages throughout the simulation process.

## Value

Selected simulation results as a nested list summarized (mean, sd, min, max) across multiple replicates (default), or 2-3D arrays including results for each replicate:

`abundance` Matrix or 3D array of simulation abundance: *populations* rows by *time_steps* columns (by *replicates* deep).

`abundance_stages` List of matrices or 3D arrays of simulation abundance for unique stage-compartment combinations when present: each *populations* rows by *time_steps* columns (by *replicates* deep).

`all$abundance` Array or matrix of total abundance across populations: *time_steps* (rows by *replicates* columns).

`all$abundance_stages` List of arrays or matrices of total abundance across populations for unique stage-compartment combinations when present: each *time_steps* (rows by *replicates* columns).

`all$ema` Array of expected minimum abundance at each time step (averaged across replicates).

`extirpation` Array or matrix of extirpation times: *populations* (rows by *replicates* columns).

`all$extirpation` Array of extirpation time across populations for each replicate.

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

`all$occupancy` Array or matrix of the number of populations occupied at each time-step: *time_steps* (rows by *replicates* columns).

`additional results` Additional results may be attached via user-defined functions (using `params$simulator$results`).

## Examples

```
inputs <- list(
 time_steps = 5,
 seasons = 2,
 populations = 25,
 stages = 2,
 compartments = 4,
 coordinates = data.frame(x = rep(seq(177.01, 177.05, 0.01), 5),
                          y = rep(seq(-18.01, -18.05, -0.01), each = 5)),
 initial_abundance = c(c(5000, 5000, 0, 1, 0, 0, 0, 0),
                       rep(c(5000, 5000, 0, 0, 0, 0, 0, 0), 24)) |>
   matrix(nrow = 8),
 carrying_capacity = matrix(100000, nrow = 25, ncol = 5),
 breeding_season_length = rep(100, 25),
```

```
 mortality = c(0.4, 0, 0.505, 0.105, 0.4, 0, 0.45, 0.05),
 mortality_unit = 1,
 fecundity = 15,
 fecundity_unit = 1,
 fecundity_mask = c(0, 1, 0, 1, 0, 1, 0, 1),
 transmission = c(0.00002, 0.00001, 7.84e-06, 3.92e-06),
 transmission_unit = 0,
 transmission_mask = c(1, 1, 0, 0, 1, 1, 0, 0),
 recovery = c(0.05714286, 0.05714286, 0.1, 0.1),
 recovery_unit = rep(0, 8),
 recovery_mask = c(0, 0, 1, 1, 0, 0, 1, 1),
 season_functions = list(siri_model_summer, siri_model_winter),
 simulation_order = c("transition", "season_functions", "results")
)
disease_simulator(inputs)
```

---

`disease_transformation` *Nested functions for a user-defined transformation of a population affected by a disease outbreak.*

---

## Description

Modular functions for the disease simulator for performing a transformation of a population across stages and disease compartments (and optionally carrying capacity) at a specified time step via a user-defined function.

## Usage

```
disease_transformation(params)
```

## Arguments

| | |
|---|---|
| `params` | A list of parameters, which must contain all of the below, except `name`, which is optional:<br>`replicates` Number of replicate simulation runs.<br>`time_steps` Number of simulation time steps.<br>`years_per_step` Number of years per time step.<br>`populations` Number of populations.<br>`seasons` Number of seasons per year.<br>`stages` Number of life cycle stages.<br>`compartments` Number of disease compartments (e.g., 3 for a SIR model).<br>`demographic_stochasticity` Boolean for optionally choosing demographic stochasticity for the transformation.<br>`density_stages` Array of booleans or numeric (0,1) for each stage to indicate which stages are affected by density. |

`abundance_threshold` A quasi-extinction threshold at which a population becomes extinct.

`mortality` A vector of mortality rates, one for each combination of stages and compartments.

`mortality_unit` A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily.

`fecundity` A vector of fecundity rates, one for each combination of stages and compartments for which fecundity applies.

`fecundity_unit` A vector indicating whether fecundity rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily.

`fecundity_mask` A vector indicating which stages and compartments reproduce.

`transmission` A vector of transmission rates, one for each combination of stages and compartment for which transmission applies (see `transmission_mask` below.

`transmission_unit` A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily.

`transmission_mask` A vector indicating which stages and compartments are subject to transmission (i.e., classes susceptible to infection.)

`recovery` A vector of recovery rates, one for each combination of stages and compartment for which recovery applies (see `recovery_mask` below.)

`recovery_unit` A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily.

`recovery_mask` A vector indicating which compartments are subject to recovery (i.e., infected classes that can recover.)

`transformation` A user-defined function (optionally nested in a list with additional attributes) for performing transformation using `params` as arguments.

`simulator` `poems::SimulatorReference` object with dynamically accessible *attached* and *results* lists.

`name` Optional name for the transformation function.

`additional attributes` Additional attributes when the transformation is optionally nested in a list.

## Value

Abundance (and capacity) transformation function: `function(r, tm, carrying_capacity, segment_abundance,occupie`
where:

`r` Simulation replicate.

`tm` Simulation time step.

`carrying_capacity` Array of carrying capacity values for each population at time step.

`segment_abundance` Matrix of abundance for each stage-compartment combo (rows) and population (columns) at time step.

`occupied_indices` Array of indices for populations occupied at time step.

`returns` List with transformed stage abundance matrix (and optionally carrying capacity).

## disease_transitions *Nested functions for stage- and compartment-based population transitions in an outbreak.*

### Description

Modular functions for the disease simulator to transition populations between stages and within disease compartments.

### Usage

```
disease_transitions(stages, compartments)
```

### Arguments

| | |
|---|---|
| stages | Number of life cycle stages. |
| compartments | Number of disease compartments. |

### Value

Transition calculation function that takes as input:

segment_abundance Matrix of (current) abundance for each stage-compartment combo (rows) and population (columns) at time step.

occupied_indices Array of indices for populations occupied at (current) time step.

## finch_region *Study region for house finch conjunctivitis simulations*

### Description

This is a raster that defines the spatial extent and resolution of the house finch conjunctivitis example in the vignette. It encompasses terrestrial North America from southern Mexico to southern Canada. It has a resolution of 46.375 by 46.375 kilometers, adding up to 6,355 possible populations of house finches. The projected coordinate system for the study region is Albers equal-area conic, with a reference longitude of 94.5 W and standard parallels at 21.5 N and 47.5 N.

### Usage

```
finch_region
```

### Format

A raster object with 1 layer and 17066 cells defining the spatial scope of a simulation.

## habitat_suitability *House finch habitat suitability*

### Description

This is a RasterStack containing data on habitat suitability for the house finch in North America from 1994 to 2016. This habitat suitability stack was generated using a species distribution model. The predictors for the SDM were the 12 bioclimatic variables, plus an urbanization index (proportion of each grid cell with urban land use.) The occurrences for the SDM came from quality-checked GBIF records.

### Usage

```
habitat_suitability
```

### Format

A RasterStack with 17066 cells and 23 layers.

### Details

The raster has the same resolution and projection as the finch_region. Habitat suitability ranges from 0 to 1, with 1 being the most suitable.

## initial_abundance *Initial house finch abundance*

### Description

This numeric matrix contains initial abundances of house finches in 1994 for the house finch conjunctivitis vignette. These abundances were themselves simulated using `epizootic` and do not represent empirical estimates of house finch abundance in 1994. The matrix has 6,355 columns, one for each population in the finch_region, and 8 rows, one for each combination of life cycle stage and disease compartment (row 1: susceptible juveniles, row 2: susceptible adults, row 3: juveniles infected for the first time, row 4: adults infected for the first time, row 5: recovered juveniles, row 6: recovered adults, row 7: re-infected juveniles, row 8: re-infected adults.) This initial abundance matrix has only zeroes in rows 3-8 because the disease has not yet broken out at the start of the simulation.

### Usage

```
initial_abundance
```

### Format

A numeric matrix with 8 rows and 6355 columns.

`siri_model_summer` *Simulate a* Mycoplasma gallisepticum *Outbreak in the Breeding Season*

## Description

Simulate a *Mycoplasma gallisepticum* outbreak during the breeding season day-by-day in a population of house finches (*Haemorhous mexicanus*). Uses a SIRI model (Susceptible-Infected 1-Recovered-Infected 2+) and includes demographic stochasticity in fecundity, mortality, and infection.

## Usage

```
siri_model_summer(inputs)
```

## Arguments

`inputs` A nested list with named elements:

`replicates` Number of replicate simulation runs.

`time_steps` Number of simulation time steps.

`populations` Number of populations.

`stages` Number of life cycle stages.

`compartments` Number of disease compartments (e.g., 3 for a SIR model).

`abundance_threshold` A quasi-extinction threshold at which a population becomes extinct.

`mortality` A vector of mortality rates, one for each combination of stages and compartments.

`mortality_unit` A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily.

`fecundity` A vector of fecundity rates, one for each combination of stages and compartments for which fecundity applies.

`fecundity_unit` A vector indicating whether fecundity rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily.

`fecundity_mask` A vector indicating which stages and compartments reproduce.

`transmission` A vector of transmission rates, one for each combination of stages and compartment for which transmission applies (see `transmission_mask` below.

`transmission_unit` A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily.

`transmission_mask` A vector indicating which stages and compartments are subject to transmission (i.e., classes susceptible to infection.)

`recovery` A vector of recovery rates, one for each combination of stages and compartment for which recovery applies (see `recovery_mask` below.)

`recovery_unit` A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily.

`recovery_mask` A vector indicating which compartments are subject to recovery (i.e., infected classes that can recover.)

`r` Simulation replicate.

`tm` Simulation time step.

`carrying_capacity` Array of carrying capacity values for each population at time step.

`breeding_season_length` Array of breeding season lengths in days for each population at time step.

`segment_abundance` Matrix of (current) abundance for each stage-compartment combo (rows) and population (columns) at time step.

`occupied_indices` Array of indices for populations occupied at (current) time step.

`simulator` `poems::SimulatorReference` object with dynamically accessible *attached* and *results* lists.

`additional attributes` Additional attributes when the transformation is optionally nested in a list.

## Details

The function can also handle the case in which there are no infected individuals. The principal difference between this function and the one for simulating an outbreak in the non-breeding season is that this one includes fecundity.

## Value

An abundance matrix with `populations` columns and `stages*compartments` rows, updated from the `segment_abundance` input in `inputs` according to demography and disease dynamics.

---

# `siri_model_winter` *Simulate a* Mycoplasma gallisepticum *Outbreak in the Non-Breeding Season*

---

## Description

Simulate a *Mycoplasma gallisepticum* outbreak during the non-breeding season day-by-day in a population of house finches (*Haemorhous mexicanus*). Uses a SIRI model (Susceptible-Infected 1-Recovered-Infected 2+) and includes demographic stochasticity in fecundity, mortality, and infection.

## Usage

```
siri_model_winter(inputs)
```

## Arguments

`inputs` A nested list with named elements:

`replicates` Number of replicate simulation runs.

`time_steps` Number of simulation time steps.

`populations` Number of populations.

`stages` Number of life cycle stages.

`compartments` Number of disease compartments (e.g., 3 for a SIR model).

`abundance_threshold` A quasi-extinction threshold below which a population becomes extinct.

`mortality` A vector of mortality rates, one for each combination of stages and compartments.

`mortality_unit` A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily.

`fecundity` A vector of fecundity rates, one for each combination of stages and compartments for which fecundity applies.

`fecundity_unit` A vector indicating whether fecundity rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily.

`fecundity_mask` A vector indicating which stages and compartments reproduce.

`transmission` A vector of transmission rates, one for each combination of stages and compartment for which transmission applies (see `transmission_mask` below.

`transmission_unit` A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily.

`transmission_mask` A vector indicating which stages and compartments are subject to transmission (i.e., classes susceptible to infection.)

`recovery` A vector of recovery rates, one for each combination of stages and compartment for which recovery applies (see `recovery_mask` below.)

`recovery_unit` A vector indicating whether mortality rates are daily or seasonal. 1 indicates seasonal, 0 indicates daily.

`recovery_mask` A vector indicating which compartments are subject to recovery (i.e., infected classes that can recover.)

`r` Simulation replicate.

`tm` Simulation time step.

`carrying_capacity` Array of carrying capacity values for each population at time step.

`breeding_season_length` Array of breeding season lengths in days for each population at time step.

`segment_abundance` Matrix of (current) abundance for each stage-compartment combo (rows) and population (columns) at time step.

`occupied_indices` Array of indices for populations occupied at (current) time step.

`simulator` `poems::SimulatorReference` object with dynamically accessible *attached* and *results* lists.

`additional attributes` Additional attributes when the transformation is optionally nested in a list.

## Details

The function can also handle the case in which there are no infected individuals. The principal difference between this function and the one for simulating an outbreak in the breeding season is that this one does not include fecundity.

## Value

An abundance matrix with `populations` columns and `stages*compartments` rows, updated from the `segment_abundance` input in `inputs` according to demography and disease dynamics.

# Index