Package 'samc'

April 1, 2021

Type Package

Title Spatial Absorbing Markov Chains

Version 1.3.0

Description Implements functions for working with absorbing Markov chains. The implementation is based on the framework described in ``Toward a unified framework for connectivity that disentangles movement and mortality in space and time'' by Fletcher et al. (2019) <doi:10.1111/ele.13333>, which applies them to spatial ecology. This framework incorporates both resistance and absorption with spatial absorbing Markov chains (SAMC) to provide several short-term and long-term predictions for metrics related to connectivity in landscapes. Despite the ecological context of the framework, this package can be used in any application of absorbing Markov chains.

License GPL (>= 3)

URL https://andrewmarx.github.io/samc/

BugReports https://github.com/andrewmarx/samc/issues/

Encoding UTF-8

LazyData true

Depends R (>= 3.3.0)

Imports methods, gdistance, Matrix, raster

RoxygenNote 7.1.1

Suggests knitr, rmarkdown, testthat, viridisLite

VignetteBuilder knitr

Collate 'RcppExports.R' 'samc-class.R' 'check.R' 'location-class.R' 'cond_passage.R' 'data.R' 'visitation.R' 'dispersal.R' 'distribution.R' 'internal-functions.R' 'locate.R' 'map.R' 'mortality.R' 'pairwise.R' 'samc.R' 'survival.R'

LinkingTo Rcpp (>= 1.0.1), RcppEigen (>= 0.3.3.5.0)

NeedsCompilation yes

Author Andrew Marx [aut, cre, cph] (<https://orcid.org/0000-0002-7456-1631>),
Robert Fletcher [ctb] (<https://orcid.org/0000-0003-1717-5707>),
Miguel Acevedo [ctb] (<https://orcid.org/0000-0002-8289-1497>),
Jorge Sefair [ctb] (<https://orcid.org/0000-0002-5887-8938>),
Chao Wang [ctb] (<https://orcid.org/0000-0003-2706-5122>)

Maintainer Andrew Marx <andrewjmarx@ufl.edu>

Repository CRAN

Date/Publication 2021-04-01 20:00:02 UTC

R topics documented:

---

check *Check landscape data*

Description

Check that landscape inputs have valid values and matching properties.

Usage

```
check(a, b)

## S4 method for signature 'RasterLayer,missing'
check(a)

## S4 method for signature 'matrix,missing'
check(a)
```

```
## S4 method for signature 'RasterLayer,RasterLayer'
check(a, b)

## S4 method for signature 'matrix,matrix'
check(a, b)

## S4 method for signature 'samc,RasterLayer'
check(a, b)

## S4 method for signature 'samc,matrix'
check(a, b)
```

Arguments

a	A samc-class, matrix, or RasterLayer-class object
b	A matrix or RasterLayer-class object

Details

This function is used to ensure that landscape inputs (resistance, absorption, fidelity, and occupancy) have valid values and the same properties. This includes checking the CRS (if using RasterLayer inputs), dimensions, and locations of cells with NA data. It can be used to directly compare two matrices or two RasterLayers, or it can be used to check a samc-class object against a matrix or RasterLayer.

The function returns TRUE if the inputs have matching properties. Otherwise, it will stop execution and print the error message generated by the compareRaster() function from the raster package. This error will provide some details about the difference between the two inputs.

Note that the package assumes the different landscape inputs will be the same type, either matrices or RasterLayers. Mixing RasterLayer data and matrix data is not supported.

Value

See *Details* section.

Examples

```
# "Load" the data. In this case we are using data built into the package.
# In practice, users will likely load raster data using the raster() function
# from the raster package.
res_data <- samc::ex_res_data
abs_data <- samc::ex_abs_data
occ_data <- samc::ex_occ_data


# Make sure our data meets the basic input requirements of the package using
# the check() function.
check(res_data, abs_data)
check(res_data, occ_data)
```

```
# Create a `samc-class` object with the resistance and absorption data using
# the samc() function. We use the recipricol of the arithmetic mean for
# calculating the transition matrix. Note, the input data here are matrices,
# not RasterLayers. If using RasterLayers, the latlon parameter must be set.
samc_obj <- samc(res_data, abs_data, tr_fun = function(x) 1/mean(x))


# Convert the occupancy data to probability of occurrence
occ_prob_data <- occ_data / sum(occ_data, na.rm = TRUE)


# Calculate short- and long-term metrics using the analytical functions
short_mort <- mortality(samc_obj, occ_prob_data, time = 50)
short_dist <- distribution(samc_obj, origin = 3, time = 50)
long_disp <- dispersal(samc_obj, occ_prob_data)
visit <- visitation(samc_obj, dest = 4)
surv <- survival(samc_obj)


# Use the map() function to turn vector results into RasterLayer objects.
short_mort_map <- map(samc_obj, short_mort)
short_dist_map <- map(samc_obj, short_dist)
long_disp_map <- map(samc_obj, long_disp)
visit_map <- map(samc_obj, visit)
surv_map <- map(samc_obj, surv)
```

---

cond_passage *Conditional Mean First Passage Time*

Description

Calculate the mean number of steps to first passage

Usage

```
cond_passage(samc, origin, dest)

## S4 method for signature 'samc,missing,location'
cond_passage(samc, dest)

## S4 method for signature 'samc,location,location'
cond_passage(samc, origin, dest)
```

Arguments

samc	A samc-class object. This should be output from the samc function.

origin	A positive integer or named location representing a cell in the landscape. Corresponds to row *i* of matrix *P* in the samc-class object. When paired with the dest parameter, multiple values may be provided as a vector.
dest	A positive integer or named location representing a cell in the landscape. Corresponds to column *j* of matrix *P* in the samc-class object. When paired with the origin parameter, multiple values may be provided as a vector.

Details

$\tilde{t} = \tilde{B}_j^{-1}\tilde{F}\tilde{B}_j \cdot 1$

- **cond_passage(samc, dest)**
  The result is a vector where each element corresponds to a cell in the landscape, and can be mapped back to the landscape using the map function. Element *i* is the mean number of steps before absorption starting from location *i* conditional on absorption into *j*
- **cond_passage(samc, origin, dest)**
  The result is a numeric value representing the mean number of steps before absorption starting from a given origin conditional on absorption into *j*.

WARNING: This function will crash when used with data representing a disconnected graph. This includes, for example, isolated pixels or islands in raster data. This is a result of the transition matrix for disconnected graphs leading to some equations being unsolvable. Different options are being explored for how to best identify these situations in data and handle them accordingly.

Value

A numeric vector or a single numeric value

Performance

Any relevant performance information about this function can be found in the performance vignette:
`vignette("performance", package = "samc")`

Examples

```
# "Load" the data. In this case we are using data built into the package.
# In practice, users will likely load raster data using the raster() function
# from the raster package.
res_data <- samc::ex_res_data
abs_data <- samc::ex_abs_data
occ_data <- samc::ex_occ_data


# Make sure our data meets the basic input requirements of the package using
# the check() function.
check(res_data, abs_data)
check(res_data, occ_data)


# Create a `samc-class` object with the resistance and absorption data using
# the samc() function. We use the recipricol of the arithmetic mean for
```

```
# calculating the transition matrix. Note, the input data here are matrices,
# not RasterLayers. If using RasterLayers, the latlon parameter must be set.
samc_obj <- samc(res_data, abs_data, tr_fun = function(x) 1/mean(x))


# Convert the occupancy data to probability of occurrence
occ_prob_data <- occ_data / sum(occ_data, na.rm = TRUE)


# Calculate short- and long-term metrics using the analytical functions
short_mort <- mortality(samc_obj, occ_prob_data, time = 50)
short_dist <- distribution(samc_obj, origin = 3, time = 50)
long_disp <- dispersal(samc_obj, occ_prob_data)
visit <- visitation(samc_obj, dest = 4)
surv <- survival(samc_obj)


# Use the map() function to turn vector results into RasterLayer objects.
short_mort_map <- map(samc_obj, short_mort)
short_dist_map <- map(samc_obj, short_dist)
long_disp_map <- map(samc_obj, long_disp)
visit_map <- map(samc_obj, visit)
surv_map <- map(samc_obj, surv)
```

`dispersal` *Calculate dispersal metrics*

Description

Calculates the probability of individuals visiting locations

Usage

```
dispersal(samc, occ, origin, dest, time)

## S4 method for signature 'samc,missing,missing,location,numeric'
dispersal(samc, dest, time)

## S4 method for signature 'samc,RasterLayer,missing,location,numeric'
dispersal(samc, occ, dest, time)

## S4 method for signature 'samc,matrix,missing,location,numeric'
dispersal(samc, occ, dest, time)

## S4 method for signature 'samc,missing,missing,missing,missing'
dispersal(samc)

## S4 method for signature 'samc,missing,location,missing,missing'
dispersal(samc, origin)
```

```
## S4 method for signature 'samc,missing,missing,location,missing'
dispersal(samc, dest)

## S4 method for signature 'samc,missing,location,location,missing'
dispersal(samc, origin, dest)

## S4 method for signature 'samc,RasterLayer,missing,missing,missing'
dispersal(samc, occ)

## S4 method for signature 'samc,matrix,missing,missing,missing'
dispersal(samc, occ)

## S4 method for signature 'samc,RasterLayer,missing,location,missing'
dispersal(samc, occ, dest)

## S4 method for signature 'samc,matrix,missing,location,missing'
dispersal(samc, occ, dest)
```

Arguments

samc	A samc-class object. This should be output from the samc function.
occ	A RasterLayer-class or matrix. The input type must match the input type used to create the samc-class object, and must have the same properties as the rest of the landscape data. See the check function for more details.
origin	A positive integer or named location representing a cell in the landscape. Corresponds to row *i* of matrix *P* in the samc-class object. When paired with the dest parameter, multiple values may be provided as a vector.
dest	A positive integer or named location representing a cell in the landscape. Corresponds to column *j* of matrix *P* in the samc-class object. When paired with the origin parameter, multiple values may be provided as a vector.
time	A positive integer or a vector of positive integers representing time steps. Vectors must be ordered and contain no duplicates. Vectors may not be used for metrics that return dense matrices. The maximum time step value is capped at 10,000.

Details

$\tilde{D}_{jt} = (\sum_{n=0}^{t-1} \tilde{Q}^n)\tilde{q}_j$

- **dispersal(samc, dest, time)**

  The result is a vector (single time step) or a list of vectors (multiple time steps) where each element corresponds to a cell in the landscape, and can be mapped back to the landscape using the map function. Element *k* is the probability of ever visiting a given destination, if starting at any other location, within *t* or fewer time steps.

$\psi^T \tilde{D}_{jt}$

- **dispersal(samc, occ, dest, time)**
  The result is a numeric (single time step) or a list of numerics (multiple time steps) that is the unconditional probability of visiting a given destination within *t* or fewer time steps.

$D = (F - I)diag(F)^{-1}$

- **dispersal(samc)**
  The result is a matrix where element (*i*,*j*) is the probability that location *j* is visited when starting in location *i*.
  The returned matrix will always be dense and cannot be optimized. Must enable override to use.
- **dispersal(samc, origin)**
  This function has not been optimized yet, and will not run.
- **dispersal(samc, dest)**
  The result is a vector where each element corresponds to a cell in the landscape, and can be mapped back to the landscape using the `map` function. Element *i* is the probability that the destination is visited when starting in location *i*.
- **dispersal(samc, origin, dest)**
  The result is a numeric value that is the probability that an individual starting at the origin visits the destination.

$\psi^T D$

- **dispersal(samc, occ)**
  The result is a vector where each element corresponds to a cell in the landscape, and can be mapped back to the landscape using the `map` function. Element *j* is the unconditional probability distribution of ever visiting location *j*, regardless of the initial location.
- **dispersal(samc, occ, dest)**
  The result is a numeric value that is the unconditional probability distribution of ever visiting a given destination, regardless of the initial location.

Value

A matrix, a vector, a list of vectors, or a numeric

Performance

Any relevant performance information about this function can be found in the performance vignette:
`vignette("performance", package = "samc")`

Examples

```
# "Load" the data. In this case we are using data built into the package.
# In practice, users will likely load raster data using the raster() function
# from the raster package.
res_data <- samc::ex_res_data
abs_data <- samc::ex_abs_data
occ_data <- samc::ex_occ_data
```

```
# Make sure our data meets the basic input requirements of the package using
# the check() function.
check(res_data, abs_data)
check(res_data, occ_data)


# Create a `samc-class` object with the resistance and absorption data using
# the samc() function. We use the recipricol of the arithmetic mean for
# calculating the transition matrix. Note, the input data here are matrices,
# not RasterLayers. If using RasterLayers, the latlon parameter must be set.
samc_obj <- samc(res_data, abs_data, tr_fun = function(x) 1/mean(x))


# Convert the occupancy data to probability of occurrence
occ_prob_data <- occ_data / sum(occ_data, na.rm = TRUE)


# Calculate short- and long-term metrics using the analytical functions
short_mort <- mortality(samc_obj, occ_prob_data, time = 50)
short_dist <- distribution(samc_obj, origin = 3, time = 50)
long_disp <- dispersal(samc_obj, occ_prob_data)
visit <- visitation(samc_obj, dest = 4)
surv <- survival(samc_obj)


# Use the map() function to turn vector results into RasterLayer objects.
short_mort_map <- map(samc_obj, short_mort)
short_dist_map <- map(samc_obj, short_dist)
long_disp_map <- map(samc_obj, long_disp)
visit_map <- map(samc_obj, visit)
surv_map <- map(samc_obj, surv)
```

`distribution` *Calculate distribution metrics*

Description

Calculate the probability of finding an individual at a given location at a specific time.

Usage

```
distribution(samc, occ, origin, dest, time)

## S4 method for signature 'samc,missing,missing,missing,numeric'
distribution(samc, time)

## S4 method for signature 'samc,missing,location,missing,numeric'
distribution(samc, origin, time)
```

```
## S4 method for signature 'samc,missing,missing,location,numeric'
distribution(samc, dest, time)

## S4 method for signature 'samc,missing,location,location,numeric'
distribution(samc, origin, dest, time)

## S4 method for signature 'samc,RasterLayer,missing,missing,numeric'
distribution(samc, occ, time)

## S4 method for signature 'samc,matrix,missing,missing,numeric'
distribution(samc, occ, time)
```

Arguments

samc	A samc-class object. This should be output from the samc function.
occ	A RasterLayer-class or matrix. The input type must match the input type used to create the samc-class object, and must have the same properties as the rest of the landscape data. See the check function for more details.
origin	A positive integer or named location representing a cell in the landscape. Corresponds to row *i* of matrix *P* in the samc-class object. When paired with the dest parameter, multiple values may be provided as a vector.
dest	A positive integer or named location representing a cell in the landscape. Corresponds to column *j* of matrix *P* in the samc-class object. When paired with the origin parameter, multiple values may be provided as a vector.
time	A positive integer or a vector of positive integers representing time steps. Vectors must be ordered and contain no duplicates. Vectors may not be used for metrics that return dense matrices. The maximum time step value is capped at 10,000.

Details

Q^t

- **distribution(samc, time)**
  The result is a matrix where element (i,j) is the probability of being at location j after t time steps if starting at location i.
  The returned matrix will always be dense and cannot be optimized. Must enable override to use.
- **distribution(samc, origin, time)**
  The result is a vector (single time step) or a list of vectors (multiple time steps) where element j is the probability of being at location j after t time steps if starting at a given origin.
- **distribution(samc, dest, time)**
  The result is a vector where element i is the probability of being at a given destination after t time steps if starting at location i.
- **distribution(samc, origin, dest, time)**
  The result is a numeric value (single time step) or a list of numeric values (multiple time steps) that is the probability of being at a given destination after t time steps when beginning at a given origin.

$\psi^T Q^t$

- **distribution(samc, occ, time)**
  The result is a vector (single time step) or a list of vectors (multiple time steps) where each element corresponds to a cell in the landscape, and can be mapped back to the landscape using the map function. Element *i* is the unconditional probability of finding an individual (or expected number of individuals) in location *i* after *t* time steps.

Value

A vector

Performance

Any relevant performance information about this function can be found in the performance vignette:
`vignette("performance", package = "samc")`

Examples

```
# "Load" the data. In this case we are using data built into the package.
# In practice, users will likely load raster data using the raster() function
# from the raster package.
res_data <- samc::ex_res_data
abs_data <- samc::ex_abs_data
occ_data <- samc::ex_occ_data


# Make sure our data meets the basic input requirements of the package using
# the check() function.
check(res_data, abs_data)
check(res_data, occ_data)


# Create a `samc-class` object with the resistance and absorption data using
# the samc() function. We use the recipricol of the arithmetic mean for
# calculating the transition matrix. Note, the input data here are matrices,
# not RasterLayers. If using RasterLayers, the latlon parameter must be set.
samc_obj <- samc(res_data, abs_data, tr_fun = function(x) 1/mean(x))


# Convert the occupancy data to probability of occurrence
occ_prob_data <- occ_data / sum(occ_data, na.rm = TRUE)


# Calculate short- and long-term metrics using the analytical functions
short_mort <- mortality(samc_obj, occ_prob_data, time = 50)
short_dist <- distribution(samc_obj, origin = 3, time = 50)
long_disp <- dispersal(samc_obj, occ_prob_data)
visit <- visitation(samc_obj, dest = 4)
surv <- survival(samc_obj)
```

```
# Use the map() function to turn vector results into RasterLayer objects.
short_mort_map <- map(samc_obj, short_mort)
short_dist_map <- map(samc_obj, short_dist)
long_disp_map <- map(samc_obj, long_disp)
visit_map <- map(samc_obj, visit)
surv_map <- map(samc_obj, surv)
```

---

ex_abs_data *Example absorption data*

Description

A fabricated dataset containing landscape absorption probability values.

Usage

```
ex_abs_data
```

Format

A matrix with 34 rows and 202 columns.

Source

Fletcher et al (2019) <doi:10.1111/ele.13333>

---

ex_occ_data *Example occupancy data*

Description

A fabricated dataset containing landscape occupancy values.

Usage

```
ex_occ_data
```

Format

A matrix with 34 rows and 202 columns.

Source

Fletcher et al (2019) <doi:10.1111/ele.13333>

ex_res_data *Example resistance data*

Description

A fabricated dataset containing landscape resistance values.

Usage

```
ex_res_data
```

Format

A matrix with 34 rows and 202 columns.

Source

Fletcher et al (2019) <doi:10.1111/ele.13333>

locate *Get cell numbers*

Description

Get cell numbers from raster data

Usage

```
locate(samc, xy)

## S4 method for signature 'samc,missing'
locate(samc)

## S4 method for signature 'samc,ANY'
locate(samc, xy)
```

Arguments

samc	A samc-class object
xy	Any valid input to the y argument of the extract function in the raster package.

Details

This function is used to get cell numbers from raster data. The numbers used for origin and destination values in many samc metrics refer to column/row numbers of the P matrix. For a P matrix derived from raster data, these numbers would normally line up with the cell numbers of the raster, but this is not always true. This is the case when the raster contains NA data; the cells associated with this data are excluded from the P matrix. This causes issues trying to determine the cell numbers that should be used in analyses.

The `locate` function operates more-or-less like the `cellFromXY` function in the raster package, but unlike `cellFromXY`, locate properly accounts for NA cells in identifying cell numbers from coordinate data.

This function can also be used if the samc object was created from matrix inputs for the resistance, absorption, and fidelity parameters. In this case, the values in the xy coordinate parameter can be column-row values with the caveat that (1,1) is the bottom left corner.

The xy parameter can also be excluded. In this case, the function returns a raster where the values of the cells contains the cell number.

Internally, this function relies on the `extract` function from the raster package, and any valid input for the y argument of that function is valid here.

Value

A rasterlayer or a vector

Examples

```
library(samc)
library(raster)


# Load example data
res_data <- samc::ex_res_data
abs_data <- samc::ex_abs_data
occ_data <- samc::ex_occ_data


# Create samc-class object
samc_obj <- samc(res_data, abs_data, tr_fun = function(x) 1/mean(x))


# We can use locate() to return a raster with the cell numbers encoded as data
# in the cells
cell_raster <- locate(samc_obj)
plot(cell_raster)


# We can use a variety of spatial inputs to get cell numbers using locate()
# The simplest is a two-column data.frame
coords <- data.frame(x = c(50, 79, 22),
                     y = c(25, 11, 19))
print(coords)
```

```
locate(samc_obj, coords)

# You will get an error if you input a coordinate that does not correspond
# to a non-NA cell
coords <- data.frame(x = c(1),
                     y = c(1))
print(coords)
try(locate(samc_obj, coords))
```

`location-class` *location class*

Description

Union class for location inputs

Details

The location class is a union class of the "numeric" and "character" classes. Users generally do not need to worry about it except to know that any method parameter with "location" as the type can have either an integer or a character name provided as input.

`map` *Map vector data*

Description

Map vector data to a RasterLayer

Usage

```
map(samc, vec)

## S4 method for signature 'samc,numeric'
map(samc, vec)

## S4 method for signature 'samc,list'
map(samc, vec)
```

Arguments

`samc`	Spatial absorbing Markov chain object. This should be output from the samc() function.
`vec`	Vector data to fill into the map.

Details

This is a convenience function to ensure that vector data is properly mapped back to the original landscape data. The reason this is needed is that the package supports both matrices and RasterLayers, which differ in the order that data is read and written (R matrices are column-major order, whereas the raster package uses row-major order). Internally, the package uses only a single order, regardless of the original data. This can cause issues with mapping vector results if care is not taken, and this function is provided to simplify the process. It also correctly maps results for landscape data that has NA cells, which are another potential source of error if not careful.

The only requirement of the vec input is that the number of elements in it matches the number of non-NA cells in the landscape data that was used to create the samc object.

Value

A RasterLayer object

Examples

```
# "Load" the data. In this case we are using data built into the package.
# In practice, users will likely load raster data using the raster() function
# from the raster package.
res_data <- samc::ex_res_data
abs_data <- samc::ex_abs_data
occ_data <- samc::ex_occ_data


# Make sure our data meets the basic input requirements of the package using
# the check() function.
check(res_data, abs_data)
check(res_data, occ_data)


# Create a `samc-class` object with the resistance and absorption data using
# the samc() function. We use the recipricol of the arithmetic mean for
# calculating the transition matrix. Note, the input data here are matrices,
# not RasterLayers. If using RasterLayers, the latlon parameter must be set.
samc_obj <- samc(res_data, abs_data, tr_fun = function(x) 1/mean(x))


# Convert the occupancy data to probability of occurrence
occ_prob_data <- occ_data / sum(occ_data, na.rm = TRUE)


# Calculate short- and long-term metrics using the analytical functions
short_mort <- mortality(samc_obj, occ_prob_data, time = 50)
short_dist <- distribution(samc_obj, origin = 3, time = 50)
long_disp <- dispersal(samc_obj, occ_prob_data)
visit <- visitation(samc_obj, dest = 4)
surv <- survival(samc_obj)


# Use the map() function to turn vector results into RasterLayer objects.
```

```
short_mort_map <- map(samc_obj, short_mort)
short_dist_map <- map(samc_obj, short_dist)
long_disp_map <- map(samc_obj, long_disp)
visit_map <- map(samc_obj, visit)
surv_map <- map(samc_obj, surv)
```

---

mortality *Calculate mortality metrics*

Description

Calculates the probability of experiencing mortality at specific locations.

Usage

```
mortality(samc, occ, origin, dest, time)

## S4 method for signature 'samc,missing,missing,missing,numeric'
mortality(samc, time)

## S4 method for signature 'samc,missing,location,missing,numeric'
mortality(samc, origin, time)

## S4 method for signature 'samc,missing,missing,location,numeric'
mortality(samc, dest, time)

## S4 method for signature 'samc,missing,location,location,numeric'
mortality(samc, origin, dest, time)

## S4 method for signature 'samc,RasterLayer,missing,missing,numeric'
mortality(samc, occ, time)

## S4 method for signature 'samc,matrix,missing,missing,numeric'
mortality(samc, occ, time)

## S4 method for signature 'samc,missing,missing,missing,missing'
mortality(samc)

## S4 method for signature 'samc,missing,location,missing,missing'
mortality(samc, origin)

## S4 method for signature 'samc,missing,missing,location,missing'
mortality(samc, dest)

## S4 method for signature 'samc,missing,location,location,missing'
mortality(samc, origin, dest)
```

```
## S4 method for signature 'samc,RasterLayer,missing,missing,missing'
mortality(samc, occ)

## S4 method for signature 'samc,matrix,missing,missing,missing'
mortality(samc, occ)
```

Arguments

samc	A samc-class object. This should be output from the samc function.
occ	A RasterLayer-class or matrix. The input type must match the input type used to create the samc-class object, and must have the same properties as the rest of the landscape data. See the check function for more details.
origin	A positive integer or named location representing a cell in the landscape. Corresponds to row *i* of matrix *P* in the samc-class object. When paired with the dest parameter, multiple values may be provided as a vector.
dest	A positive integer or named location representing a cell in the landscape. Corresponds to column *j* of matrix *P* in the samc-class object. When paired with the origin parameter, multiple values may be provided as a vector.
time	A positive integer or a vector of positive integers representing time steps. Vectors must be ordered and contain no duplicates. Vectors may not be used for metrics that return dense matrices. The maximum time step value is capped at 10,000.

Details

$\tilde{B}_t = (\sum_{n=0}^{t-1} Q^n)\tilde{R}$

- **mortality(samc, time)**

  The result is a matrix where element (i,j) is the probability of experiencing mortality at location j within t or fewer steps if starting at location i.

  The returned matrix will always be dense and cannot be optimized. Must enable override to use.

- **mortality(samc, origin, time)**

  The result is a vector (single time step) or a list of vectors (multiple time steps) where each element corresponds to a cell in the landscape, and can be mapped back to the landscape using the map function. Element j is the probability of experiencing mortality at location j within t or fewer steps if starting at a given origin.

- **mortality(samc, dest, time)**

  The result is a vector (single time step) or a list of vectors (multiple time steps) where each element corresponds to a cell in the landscape, and can be mapped back to the landscape using the map function. Element i is the probability of experiencing mortality at a given destination within t or fewer steps if starting at location i.

- **mortality(samc, origin, dest, time)**

  The result is a numeric value (single time step) or a list of numeric values (multiple time steps) that is the probability of experiencing mortality at a given destination within t or fewer steps if starting at a given origin.

$\psi^T \tilde{B}_t$

- **mortality(samc, occ, time)**
  The result is a vector (single time step) or a list of vectors (multiple time steps) where each element corresponds to a cell in the landscape, and can be mapped back to the landscape using the map function. Element j is the unconditional probability of experiencing mortality at location j within t or fewer time steps.

$B = F\tilde{R}$

- **mortality(samc)**
  The result is a matrix where element (i,j) is the probability of experiencing mortality at location j if starting at location i.
  The returned matrix will always be dense and cannot be optimized. Must enable override to use.
- **mortality(samc, origin)**
  The result is a vector where each element corresponds to a cell in the landscape, and can be mapped back to the landscape using the map function. Element j is the probability of experiencing mortality at location j if starting at a given origin.
- **mortality(samc, dest)**
  The result is a vector where each element corresponds to a cell in the landscape, and can be mapped back to the landscape using the map function. Element i is the probability of experiencing mortality at a given destination if starting at location i.
- **mortality(samc, origin, dest)**
  The result is a numeric value that is the probability of experiencing mortality at a given destination if starting at a given origin

$\psi^T B$

- **mortality(samc, occ)**
  The result is a vector where each element corresponds to a cell in the landscape, and can be mapped back to the landscape using the map function. Element j is the unconditional probability of experiencing mortality at location j, regardless of the initial state.

Value

A matrix, vector, or numeric

Performance

Any relevant performance information about this function can be found in the performance vignette:
`vignette("performance", package = "samc")`

Examples

```
# "Load" the data. In this case we are using data built into the package.
# In practice, users will likely load raster data using the raster() function
# from the raster package.
res_data <- samc::ex_res_data
```

```
abs_data <- samc::ex_abs_data
occ_data <- samc::ex_occ_data


# Make sure our data meets the basic input requirements of the package using
# the check() function.
check(res_data, abs_data)
check(res_data, occ_data)


# Create a `samc-class` object with the resistance and absorption data using
# the samc() function. We use the recipricol of the arithmetic mean for
# calculating the transition matrix. Note, the input data here are matrices,
# not RasterLayers. If using RasterLayers, the latlon parameter must be set.
samc_obj <- samc(res_data, abs_data, tr_fun = function(x) 1/mean(x))


# Convert the occupancy data to probability of occurrence
occ_prob_data <- occ_data / sum(occ_data, na.rm = TRUE)


# Calculate short- and long-term metrics using the analytical functions
short_mort <- mortality(samc_obj, occ_prob_data, time = 50)
short_dist <- distribution(samc_obj, origin = 3, time = 50)
long_disp <- dispersal(samc_obj, occ_prob_data)
visit <- visitation(samc_obj, dest = 4)
surv <- survival(samc_obj)


# Use the map() function to turn vector results into RasterLayer objects.
short_mort_map <- map(samc_obj, short_mort)
short_dist_map <- map(samc_obj, short_dist)
long_disp_map <- map(samc_obj, long_disp)
visit_map <- map(samc_obj, visit)
surv_map <- map(samc_obj, surv)
```

`pairwise` *Pairwise analyses*

Description

Analysis for pairwise combinations locations

Usage

```
pairwise(fun, samc, origin, dest)

## S4 method for signature '`function`,samc,location,location'
pairwise(fun, samc, origin, dest)
```

```
## S4 method for signature '`function`,samc,location,missing'
pairwise(fun, samc, origin)
```

Arguments

fun	A samc analytical function with signature fun(samc, origin, dest)
samc	A samc-class object
origin	A vector of locations
dest	A vector of locations. Can be excluded to reuse the origin parameter

Details

When providing vector inputs for the 'origin' and 'dest' parameters to analytical functions, the package assumes that users are providing pairs of 'origin' and 'dest'. That is, 'origin[1]' is paired with 'dest[1]', 'origin[2]' is paired 'dest[2]', etc. Another way to think about it is that these two vector inputs can be treated as columns in the same dataframe. The result of the analytical function then is a vector of the same length as the input. This behavior works for any situation, so it is the default for the package.

However, some users may wish to run an analytical function for all the pairwise combinations of the values in the input vectors. That is, 'origin[1]' is paired with 'dest[1]','dest[2]', 'dest[3]', etc, before moving on to the next elements in 'origin'. This approach has the advantage of potentially reducing the amount of code needed for an analysis, and the results can be represented as a pairwise matrix, but it is not suitable for all situations. To enable this second approach more easily, the 'pairwise()' function runs all the combinations of the 'origin' and 'dest' parameters for an analytical function and returns the results in a 'long' format data.frame. This data.frame can then be reshaped into a pairwise matrix or 'wide' format data.frame using tools like the reshape2 or tidyr packages.

This function is not intended to be used with other inputs such as 'occ' or 'time'

Value

A 'long' format data.frame

Examples

```
library(samc)

# Load example data
res_data <- samc::ex_res_data
abs_data <- samc::ex_abs_data
occ_data <- samc::ex_occ_data


# Create samc-class object
samc_obj <- samc(res_data, abs_data, tr_fun = function(x) 1/mean(x))

# pairwise() example
pw <- pairwise(cond_passage, samc_obj, origin = 1:4, dest = 5)
print(pw)
```

```
# pairwise() example without dest
pw <- pairwise(dispersal, samc_obj, origin = c(2, 7))
print(pw)
```

---

samc *Create an samc object*

Description

Create an samc object that contains the absorbing Markov chain data

Usage

```
samc(resistance, absorption, fidelity, latlon, tr_fun, p_mat, ...)

## S4 method for signature
## 'RasterLayer,RasterLayer,RasterLayer,logical,`function`,missing'
samc(
  resistance,
  absorption,
  fidelity,
  latlon,
  tr_fun,
  override = FALSE,
  directions = 8
)

## S4 method for signature
## 'RasterLayer,RasterLayer,missing,logical,`function`,missing'
samc(resistance, absorption, latlon, tr_fun, override = FALSE, directions = 8)

## S4 method for signature 'matrix,matrix,matrix,missing,`function`,missing'
samc(
  resistance,
  absorption,
  fidelity,
  tr_fun,
  override = FALSE,
  directions = 8
)

## S4 method for signature 'matrix,matrix,missing,missing,`function`,missing'
samc(resistance, absorption, tr_fun, override = FALSE, directions = 8)

## S4 method for signature 'missing,missing,missing,missing,missing,dgCMatrix'
```

```
samc(p_mat, override = FALSE)

## S4 method for signature 'missing,missing,missing,missing,missing,matrix'
samc(p_mat, override = FALSE)
```

Arguments

resistance	A RasterLayer-class or matrix
absorption	A RasterLayer-class or matrix
fidelity	A RasterLayer-class or matrix
latlon	Logical (TRUE or FALSE) indicating whether the rasters use latitude/longitude
tr_fun	A function to calculate the transition values in the transition function
p_mat	A base R matrix object or Matrix package dgCMatrix sparse matrix
...	Placeholder
override	Optional flag to prevent accidentally running memory intensive functions. Defaults to FALSE
directions	Optional param. Must be set to either 4 or 8 (default is 8)

Details

This function is used to create a samc-class object. There are two options for creating this object.

Option 1: Raster and Matrix Inputs

The samc-class object can be created from a combination of resistance, absorption, and fidelity data. These different landscape data inputs must be the same type (a matrix or RasterLayer), and have identical properties, including dimensions, location of NA cells, and CRS (if using RasterLayers). Some of the inputs are mandatory, whereas others are optional.

The resistance and absorption inputs are always mandatory, whereas the fidelity input is optional. If the fidelity input is not provided, then it it is assumed that there is no site fidelity (i.e., individuals will always move to an adjacent cell each time step).

The latlon parameter is required if the landscape data inputs are RasterLayer objects. The package does not attempt to determine this automatically, and it does not assume a default. Users must set it to TRUE if they are using latitude and longitude data.

The tr_fun parameter is mandatory. It used when calculating the values for the transition matrix. Internally, this is passed to the transition function in the gdistance package to create the transition matrix.

Option 2: P Matrix Input

The p_mat parameter can be used to create a samc-class object directly from a preconstructed P matrix. This matrix must be either a base R matrix, or a sparse matrix (dgCMatrix format) from the Matrix package. It must meet the requirement of a P matrix described in Fletcher et al. (2019). This includes:

- The number of rows must equal the number of columns (a square matrix)
- The last row must contain all 0's, except the last element, which must be 1
- Each row must sum to 1

- All values must be in the range of 0-1

Additionally, the columns and rows of the P matrix may be named (e.g., using dimnames(), rowname(), colnames(), etc). When specifying `origin` or `dest` inputs to metrics, these names may be used instead of cell numbers. This has the advantage of making the code for an analysis easier to read and interpret, which may also help to eliminate unintentional mistakes. There are two requirements for naming the rows/cols of a P matrix. First, since the P matrix represents a pairwise matrix, the row and column names must be the same. Second, there must be no duplicate names. The exception to these rules is the very last column and the very last row of the P matrix. Since these are not part of the pairwise transition matrix, they may have whatever names the user prefers.

Other Parameters

The `override` parameter is optional. To prevent users from unintentionally running memory intensive versions of functions that could make their systems non-responsive or crash software, it is set to `FALSE` by default. For various reasons, it can be set to `TRUE`. In particular, a user might do this if they are using a very small landscape dataset, or perhaps for a moderately sized dataset if they have access to a system with exceptionally large amounts of RAM. Before setting this to `TRUE`, users should read the Performance vignette/ article to understand the expected memory requirements. They should also consider starting with scaled down version of their data and then gradually scaling back up while monitoring their memory usage as a means to gauge what is reasonable for their system.

The `directions` parameter is optional. When constructing the P matrix from matrix or raster data, the `samc()` function must decide how adjacent cells are connected. This value can be set to either 4 or 8. When set to 4, nodes are connected horizontally and vertically (similar to the directions of how a rook moves in chess). When set to 8, nodes are connected diagonally in addition to horizontally and vertically (queen movement in chess). When not specified, the `samc()` function defaults to a value of 8. When using large datasets to construct a P matrix, setting the directions to 4 may lead to significant improvements in computation time and the amount of memory needed to perform an analysis.

Additional Information

Depending on the data used to construct the samc-class object, some metrics may cause crashes. This is a result of the underlying P matrix having specific properties that make some equations unsolvable. One known case is a P matrix that represents a disconnected graph, which can lead to the `cond_passage()` function crashing. In terms of raster/matrix inputs, a disconnected graph occurs when one or more pixels/cells are unreachable from other pixels/cells due to the presence of a full barrier made up of NA values. In a raster, these may be obvious as islands, but can be as inconspicuous as a rogue isolated pixel. There may be other currently unknown situations that lead to unsolvable metrics.

Future work is planned towards identifying these issues during creation of the samc-class object and handling them appropriately to prevent inadvertent crashes.

Value

A spatial absorbing Markov chain object

Examples

```
# "Load" the data. In this case we are using data built into the package.
# In practice, users will likely load raster data using the raster() function
```

```
# from the raster package.
res_data <- samc::ex_res_data
abs_data <- samc::ex_abs_data
occ_data <- samc::ex_occ_data


# Make sure our data meets the basic input requirements of the package using
# the check() function.
check(res_data, abs_data)
check(res_data, occ_data)


# Create a `samc-class` object with the resistance and absorption data using
# the samc() function. We use the recipricol of the arithmetic mean for
# calculating the transition matrix. Note, the input data here are matrices,
# not RasterLayers. If using RasterLayers, the latlon parameter must be set.
samc_obj <- samc(res_data, abs_data, tr_fun = function(x) 1/mean(x))


# Convert the occupancy data to probability of occurrence
occ_prob_data <- occ_data / sum(occ_data, na.rm = TRUE)


# Calculate short- and long-term metrics using the analytical functions
short_mort <- mortality(samc_obj, occ_prob_data, time = 50)
short_dist <- distribution(samc_obj, origin = 3, time = 50)
long_disp <- dispersal(samc_obj, occ_prob_data)
visit <- visitation(samc_obj, dest = 4)
surv <- survival(samc_obj)


# Use the map() function to turn vector results into RasterLayer objects.
short_mort_map <- map(samc_obj, short_mort)
short_dist_map <- map(samc_obj, short_dist)
long_disp_map <- map(samc_obj, long_disp)
visit_map <- map(samc_obj, visit)
surv_map <- map(samc_obj, surv)
```

`samc-class` *samc class*

Description

S4 class to manage SAMC data.

Details

The samc class is used to help ensure that the package is used correctly and to minimize the possibility for users to accidentally produce nonsensical results that may not be obviously incorrect. This

class contains the p matrix necessary for all the calculations in the package, and enforces its type so that users are less likely to inadvertently alter it in a way that will cause issues in calculations.

The class also contains a RasterLayer object derived from the input data. This object is used for checking inputs and mapping vector data in other functions.

Finally, an override flag is used to help ensure that users do not accidentally run memory intensive versions of functions that can cause their systems to become non-responsive or for software to crash.

The samc function is used to create samc-class objects.

Slots

p The transition probability matrix *P*.

source Information about the data source for the P matrix

map Used to verify landscape inputs and mapping of vector data.

clumps Number of discontinuous regions in data

override Used to prevent accidental use of memory intensive functions.

---

survival *Calculate survival metrics*

Description

Calculates the expected amount of time that individuals survive in the landscape.

Usage

```
survival(samc, occ)

## S4 method for signature 'samc,missing'
survival(samc)

## S4 method for signature 'samc,RasterLayer'
survival(samc, occ)

## S4 method for signature 'samc,matrix'
survival(samc, occ)
```

Arguments

samc	A samc-class object. This should be output from the samc function.
occ	A RasterLayer-class or matrix. The input type must match the input type used to create the samc-class object, and must have the same properties as the rest of the landscape data. See the check function for more details.

Details

$z = (I - Q)^{-1} \cdot 1 = F \cdot 1$

- **survival(samc)**
  The result is a vector where each element corresponds to a cell in the landscape, and can be mapped back to the landscape using the map function. The value of element *i* is the expected amount of time that individuals survive when starting at location *j*.

$\psi^T z$

- **survival(samc, occ)**
  The result is a numeric that represents the expected time that any individual stays in the landscape before death, regardless of the initial location.

Value

A vector or a numeric

Performance

Any relevant performance information about this function can be found in the performance vignette:
`vignette("performance", package = "samc")`

Examples

```
# "Load" the data. In this case we are using data built into the package.
# In practice, users will likely load raster data using the raster() function
# from the raster package.
res_data <- samc::ex_res_data
abs_data <- samc::ex_abs_data
occ_data <- samc::ex_occ_data


# Make sure our data meets the basic input requirements of the package using
# the check() function.
check(res_data, abs_data)
check(res_data, occ_data)


# Create a `samc-class` object with the resistance and absorption data using
# the samc() function. We use the recipricol of the arithmetic mean for
# calculating the transition matrix. Note, the input data here are matrices,
# not RasterLayers. If using RasterLayers, the latlon parameter must be set.
samc_obj <- samc(res_data, abs_data, tr_fun = function(x) 1/mean(x))


# Convert the occupancy data to probability of occurrence
occ_prob_data <- occ_data / sum(occ_data, na.rm = TRUE)


# Calculate short- and long-term metrics using the analytical functions
short_mort <- mortality(samc_obj, occ_prob_data, time = 50)
```

```
short_dist <- distribution(samc_obj, origin = 3, time = 50)
long_disp <- dispersal(samc_obj, occ_prob_data)
visit <- visitation(samc_obj, dest = 4)
surv <- survival(samc_obj)


# Use the map() function to turn vector results into RasterLayer objects.
short_mort_map <- map(samc_obj, short_mort)
short_dist_map <- map(samc_obj, short_dist)
long_disp_map <- map(samc_obj, long_disp)
visit_map <- map(samc_obj, visit)
surv_map <- map(samc_obj, surv)
```

`visitation` *Calculate visitation metrics*

Description

Calculates the number of times that individuals from each location visit each location in the landscape before death.

Usage

```
visitation(samc, origin, dest)

## S4 method for signature 'samc,missing,missing'
visitation(samc)

## S4 method for signature 'samc,location,missing'
visitation(samc, origin)

## S4 method for signature 'samc,missing,location'
visitation(samc, dest)

## S4 method for signature 'samc,location,location'
visitation(samc, origin, dest)
```

Arguments

`samc`	A `samc-class` object. This should be output from the `samc` function.
`origin`	A positive integer or named location representing a cell in the landscape. Corresponds to row *i* of matrix *P* in the `samc-class` object. When paired with the dest parameter, multiple values may be provided as a vector.
`dest`	A positive integer or named location representing a cell in the landscape. Corresponds to column *j* of matrix *P* in the `samc-class` object. When paired with the origin parameter, multiple values may be provided as a vector.

Details

$$F = (I - Q)^{-1}$$

- **visitation(samc)**
  The result is a matrix where element (*i*,*j*) is the expected number of times an individual that starts in *i* uses *j* before it dies.
  The returned matrix will always be dense and cannot be optimized. Must enable override to use.
- **visitation(samc, origin)**
  The result is a vector where each element corresponds to a cell in the landscape, and can be mapped back to the landscape using the `map` function. Element *j* is the number of times that an individual starting at the origin visits location *j* before it dies.
- **visitation(samc, dest)**
  The result is a vector where each element corresponds to a cell in the landscape, and can be mapped back to the landscape using the `map` function. Element *i* is the number of times that an individual starting at location *i* visits the destination before it dies.
- **visitation(samc, origin, dest)**
  The result is a numeric value that is the expected number of times an individual starting at the origin visits the destination before it dies.

Value

A matrix, a vector, or a numeric

Performance

Any relevant performance information about this function can be found in the performance vignette:
`vignette("performance", package = "samc")`

Examples

```
# "Load" the data. In this case we are using data built into the package.
# In practice, users will likely load raster data using the raster() function
# from the raster package.
res_data <- samc::ex_res_data
abs_data <- samc::ex_abs_data
occ_data <- samc::ex_occ_data


# Make sure our data meets the basic input requirements of the package using
# the check() function.
check(res_data, abs_data)
check(res_data, occ_data)


# Create a `samc-class` object with the resistance and absorption data using
# the samc() function. We use the recipricol of the arithmetic mean for
# calculating the transition matrix. Note, the input data here are matrices,
# not RasterLayers. If using RasterLayers, the latlon parameter must be set.
```

```
samc_obj <- samc(res_data, abs_data, tr_fun = function(x) 1/mean(x))


# Convert the occupancy data to probability of occurrence
occ_prob_data <- occ_data / sum(occ_data, na.rm = TRUE)


# Calculate short- and long-term metrics using the analytical functions
short_mort <- mortality(samc_obj, occ_prob_data, time = 50)
short_dist <- distribution(samc_obj, origin = 3, time = 50)
long_disp <- dispersal(samc_obj, occ_prob_data)
visit <- visitation(samc_obj, dest = 4)
surv <- survival(samc_obj)


# Use the map() function to turn vector results into RasterLayer objects.
short_mort_map <- map(samc_obj, short_mort)
short_dist_map <- map(samc_obj, short_dist)
long_disp_map <- map(samc_obj, long_disp)
visit_map <- map(samc_obj, visit)
surv_map <- map(samc_obj, surv)
```

Index