Package ‘SimInf’

June 18, 2020

Title A Framework for Data-Driven Stochastic Disease Spread Simulations

Version 7.0.1

Description Provides an efficient and very flexible framework to conduct data-driven epidemiological modeling in realistic large scale disease spread simulations. The framework integrates infection dynamics in subpopulations as continuous-time Markov chains using the Gillespie stochastic simulation algorithm and incorporates available data such as births, deaths and movements as scheduled events at predefined time-points. Using C code for the numerical solvers and 'OpenMP' (if available) to divide work over multiple processors ensures high performance when simulating a sample outcome. One of our design goals was to make the package extendable and enable usage of the numerical solvers from other R extension packages in order to facilitate complex epidemiological research. The package contains template models and can be extended with user-defined models. For more details see the paper by Widgren, Bauer, Eriksson and Engblom (2019) <doi:10.18637/jss.v091.i12>.

Acknowledgements This work was financially supported by the Swedish Research Council within the UPMARC Linnaeus center of Excellence (Pavol Bauer, Robin Eriksson, and Stefan Engblom), the Swedish Research Council Formas (Stefan Engblom and Stefan Widgren), the Swedish Board of Agriculture (Stefan Widgren), and by the Swedish strategic research program eSSENCE (Stefan Widgren).

URL https://github.com/stewid/SimInf

BugReports https://github.com/stewid/SimInf/issues

Type Package

LazyData true

Biarch true

NeedsCompilation yes

SystemRequirements GNU Scientific Library (GSL)

Depends R(>= 3.3)

Imports digest, graphics, grDevices, methods, stats, utils, Matrix

Collate 'C-generator.R' 'check_arguments.R' 'init.R' 'valid.R' 'SimInf_events.R' 'SimInf_model.R' 'SEIR.R' 'SIR.R' 'SISe.R' 'SISe3.R' 'SISe3_sp.R' 'SISe_sp.R' 'SimInf.R' 'degree.R' 'distance.R' 'formula.R' 'mparse.R' 'openmp.R' 'package_skeleton.R' 'plot.R' 'prevalence.R' 'print.R' 'priors.R' 'punchcard.R' 'run.R' 'trajectory.R'

Encoding UTF-8

RoxygenNote 7.1.0

Author Stefan Widgren [aut, cre] (<https://orcid.org/0000-0001-5745-2284>), Robin Eriksson [aut] (<https://orcid.org/0000-0002-4291-712X>), Stefan Engblom [aut] (<https://orcid.org/0000-0002-3614-1732>), Pavol Bauer [aut] (<https://orcid.org/0000-0003-4328-7171>), Thomas Rosendal [ctb] (<https://orcid.org/0000-0002-6576-9668>), Attractive Chaos [cph] (Author of 'kvec.h'.)

Maintainer Stefan Widgren <stefan.widgren@gmail.com>

Repository CRAN

Date/Publication 2020-06-18 17:20:02 UTC

R topics documented:

`as.data.frame.SimInf_events` *Coerce to data frame*

Description

Coerce to data frame

Usage

```
## S3 method for class 'SimInf_events'
as.data.frame(x, ...)
```

Arguments

x	any R object.
...	additional arguments to be passed to or from methods.

`boxplot,SimInf_model-method`
Box plot of number of individuals in each compartment

Description

Produce box-and-whisker plot(s) of the number of individuals in each model compartment.

Usage

```
## S4 method for signature 'SimInf_model'
boxplot(x, ...)
```

Arguments

x	The `model` to plot
...	Additional arguments affecting the plot produced.

Examples

```
## Create an 'SIR' model with 10 nodes and initialise
## it with 99 susceptible individuals and one infected
## individual. Let the model run over 100 days.
model <- SIR(u0 = data.frame(S = rep(99, 10),
                             I = rep(1, 10),
                             R = rep(0, 10)),
             tspan = 1:100,
             beta = 0.16,
             gamma = 0.077)

## Run the model and save the result.
result <- run(model)

## Create a boxplot
boxplot(result)
```

C_code — *Extract the C code from a* SimInf_model *object*

Description

Extract the C code from a SimInf_model object

Usage

```
C_code(model)
```

Arguments

model	The SimInf_model object to extract the C code from.

Value

Character vector with C code for the model.

Examples

```
## Use the model parser to create a 'SimInf_model' object that
## expresses an SIR model, where 'b' is the transmission rate and
## 'g' is the recovery rate.
model <- mparse(transitions = c("S -> b*S*I/(S+I+R) -> I", "I -> g*I -> R"),
                compartments = c("S", "I", "R"),
                gdata = c(b = 0.16, g = 0.077),
                u0 = data.frame(S = 99, I = 1, R = 0),
                tspan = 1:10)

## View the C code.
C_code(model)
```

distance_matrix — *Create a distance matrix between nodes for spatial models*

Description

Calculate the euclidian distances beween coordinates for all coordinates within the cutoff.

Usage

```
distance_matrix(x, y, cutoff, min_dist = NULL)
```

Arguments

`x`	Projected x coordinate
`y`	Projected y coordinate
`cutoff`	The distance cutoff
`min_dist`	The minimum distance to separate two nodes. If the coordinates for two nodes are identical, the min_dist must be assigned or an error is raised. Default is NULL i.e. to raise an error.

Value

`dgCMatrix`

Examples

```
## Generate a grid 10 x 10 and place one node in each cell
## separated by 100m.
nodes <- expand.grid(x = (0:9) * 100, y = (0:9) * 100)
plot(y ~ x, nodes)

## Define the cutoff to only include neighbors within 300m.
d <- distance_matrix(x = nodes$x, y = nodes$y, cutoff = 301)

## View the first 10 rows and columns in the distance matrix
d[1:10, 1:10]
```

---

events — *Extract the events from a* `SimInf_model` *object*

Description

Extract the scheduled events from a `SimInf_model` object.

Usage

```
events(model)
```

Arguments

`model`	The `model` to extract the events from.

Value

`SimInf_events` object.

Examples

```
## Create an SIR model that includes scheduled events.
model <- SIR(u0     = u0_SIR(),
             tspan  = 1:(4 * 365),
             events = events_SIR(),
             beta   = 0.16,
             gamma  = 0.077)

## Extract the scheduled events from the model and display summary
summary(events(model))

## Extract the scheduled events from the model and plot them
plot(events(model))
```

---

events_SEIR *Example data to initialize events for the 'SEIR' model*

Description

Example data to initialize scheduled events for a population of 1600 nodes and demonstrate the SEIR model.

Usage

```
events_SEIR()
```

Details

Example data to initialize scheduled events (see SimInf_events) for a population of 1600 nodes and demonstrate the SEIR model. The dataset contains 466692 events for 1600 nodes distributed over 4 * 365 days. The events are divided into three types: 'Exit' events remove individuals from the population (n = 182535), 'Enter' events add individuals to the population (n = 182685), and 'External transfer' events move individuals between nodes in the population (n = 101472). The vignette contains a detailed description of how scheduled events operate on a model.

Value

A data.frame

Examples

```
## Create an 'SEIR' model with 1600 nodes and initialize
## it to run over 4*365 days. Add one infected individual
## to the first node.
u0 <- u0_SEIR()
u0$I[1] <- 1
tspan <- seq(from = 1, to = 4*365, by = 1)
model <- SEIR(u0      = u0,
              tspan   = tspan,
```

```
                events  = events_SEIR(),
                beta    = 0.16,
                epsilon = 0.25,
                gamma   = 0.01)

## Display the number of individuals affected by each event type
## per day.
plot(events(model))

## Run the model to generate a single stochastic trajectory.
result <- run(model)
plot(result)

## Summarize the trajectory. The summary includes the number of
## events by event type.
summary(result)
```

---

events_SIR *Example data to initialize events for the 'SIR' model*

Description

Example data to initialize scheduled events for a population of 1600 nodes and demonstrate the SIR model.

Usage

```
events_SIR()
```

Details

Example data to initialize scheduled events (see SimInf_events) for a population of 1600 nodes and demonstrate the SIR model. The dataset contains 466692 events for 1600 nodes distributed over 4 * 365 days. The events are divided into three types: 'Exit' events remove individuals from the population (n = 182535), 'Enter' events add individuals to the population (n = 182685), and 'External transfer' events move individuals between nodes in the population (n = 101472). The vignette contains a detailed description of how scheduled events operate on a model.

Value

A `data.frame`

Examples

```
## Create an 'SIR' model with 1600 nodes and initialize
## it to run over 4*365 days. Add one infected individual
## to the first node.
u0 <- u0_SIR()
u0$I[1] <- 1
```

```
tspan <- seq(from = 1, to = 4*365, by = 1)
model <- SIR(u0     = u0,
             tspan  = tspan,
             events = events_SIR(),
             beta   = 0.16,
             gamma  = 0.01)

## Display the number of individuals affected by each event type
## per day.
plot(events(model))

## Run the model to generate a single stochastic trajectory.
result <- run(model)
plot(result)

## Summarize the trajectory. The summary includes the number of
## events by event type.
summary(result)
```

---

events_SISe *Example data to initialize events for the 'SISe' model*

Description

Example data to initialize scheduled events for a population of 1600 nodes and demonstrate the SISe model.

Usage

```
events_SISe()
```

Details

Example data to initialize scheduled events (see SimInf_events) for a population of 1600 nodes and demonstrate the SISe model. The dataset contains 466692 events for 1600 nodes distributed over 4 * 365 days. The events are divided into three types: 'Exit' events remove individuals from the population (n = 182535), 'Enter' events add individuals to the population (n = 182685), and 'External transfer' events move individuals between nodes in the population (n = 101472). The vignette contains a detailed description of how scheduled events operate on a model.

Value

A data.frame

Examples

```
## Create an 'SISe' model with 1600 nodes and initialize
## it to run over 4*365 days. Add one infected individual
## to the first node.
u0 <- u0_SISe()
u0$I[1] <- 1
tspan <- seq(from = 1, to = 4*365, by = 1)
model <- SISe(u0 = u0, tspan = tspan, events = events_SISe(),
              phi = 0, upsilon = 1.8e-2, gamma = 0.1, alpha = 1,
              beta_t1 = 1.0e-1, beta_t2 = 1.0e-1, beta_t3 = 1.25e-1,
              beta_t4 = 1.25e-1, end_t1 = 91, end_t2 = 182,
              end_t3 = 273, end_t4 = 365, epsilon = 0)

## Display the number of individuals affected by each event type
## per day.
plot(events(model))

## Run the model to generate a single stochastic trajectory.
result <- run(model)

## Summarize the trajectory. The summary includes the number of
## events by event type.
summary(result)
```

---

events_SISe3 — *Example data to initialize events for the 'SISe3' model*

Description

Example data to initialize scheduled events for a population of 1600 nodes and demonstrate the SISe3 model.

Usage

```
data(events_SISe3)
```

Format

A data.frame

Details

Example data to initialize scheduled events (see SimInf_events) for a population of 1600 nodes and demonstrate the SISe3 model. The dataset contains 783773 events for 1600 nodes distributed over 4 * 365 days. The events are divided into three types: 'Exit' events remove individuals from the population (n = 182535), 'Enter' events add individuals to the population (n = 182685), sQuoteInternal transfer events move individuals between compartmens within one node e.g. ageing (n = 317081), and 'External transfer' events move individuals between nodes in the population (n = 101472). The vignette contains a detailed description of how scheduled events operate on a model.

Examples

```
## Create an 'SISe3' model with 1600 nodes and initialize
## it to run over 4*365 days. Add one infected individual
## to the first node.
data("u0_SISe3", package = "SimInf")
data("events_SISe3", package = "SimInf")
u0_SISe3$I_1[1] <- 1
tspan <- seq(from = 1, to = 4*365, by = 1)
model <- SISe3(u0 = u0_SISe3, tspan = tspan, events = events_SISe3,
               phi = rep(0, nrow(u0_SISe3)), upsilon_1 = 1.8e-2,
               upsilon_2 = 1.8e-2, upsilon_3 = 1.8e-2,
               gamma_1 = 0.1, gamma_2 = 0.1, gamma_3 = 0.1,
               alpha = 1, beta_t1 = 1.0e-1, beta_t2 = 1.0e-1,
               beta_t3 = 1.25e-1, beta_t4 = 1.25e-1, end_t1 = 91,
               end_t2 = 182, end_t3 = 273, end_t4 = 365, epsilon = 0)

## Display the number of individuals affected by each event type
## per day.
plot(events(model))

## Run the model to generate a single stochastic trajectory.
result <- run(model)

## Summarize the trajectory. The summary includes the number of
## events by event type.
summary(result)
```

`gdata` *Extract global data from a* `SimInf_model` *object*

Description

The global data is a numeric vector that is common to all nodes. The global data vector is passed as an argument to the transition rate functions and the post time step function.

Usage

```
gdata(model)
```

Arguments

`model`	The `model` to get global data from.

Value

a numeric vector

Examples

```
## Create an SIR model
model <- SIR(u0 = data.frame(S = 99, I = 1, R = 0),
             tspan = 1:5, beta = 0.16, gamma = 0.077)

## Set 'beta' to a new value
gdata(model, "beta") <- 2

## Extract the global data vector that is common to all nodes
gdata(model)
```

`gdata<-` *Set a global data parameter for a* `SimInf_model` *object*

Description

The global data is a numeric vector that is common to all nodes. The global data vector is passed as an argument to the transition rate functions and the post time step function.

Usage

```
gdata(model, parameter) <- value
```

Arguments

`model`	The `model` to set a global model parameter for.
`parameter`	The name of the parameter to set.
`value`	A numeric value.

Value

a `SimInf_model` object

Examples

```
## Create an SIR model
model <- SIR(u0 = data.frame(S = 99, I = 1, R = 0),
             tspan = 1:5, beta = 0.16, gamma = 0.077)

## Set 'beta' to a new value
gdata(model, "beta") <- 2

## Extract the global data vector that is common to all nodes
gdata(model)
```

indegree	*Determine in-degree for each node in a model*

Description

The number of nodes with inward *external transfer* events to each node.

Usage

```
indegree(model)
```

Arguments

model	determine in-degree for each node in the model.

Value

vector with in-degree for each node.

Examples

```
## Create an 'SIR' model with 1600 nodes and initialize
## it with example data.
model <- SIR(u0 = u0_SIR(), tspan = 1:1460, events = events_SIR(),
             beta   = 0.16, gamma  = 0.077)

## Display indegree for each node in the model.
plot(indegree(model))
```

ldata	*Extract local data from a node*

Description

The local data is a numeric vector that is specific to a node. The local data vector is passed as an argument to the transition rate functions and the post time step function.

Usage

```
ldata(model, node)
```

Arguments

model	The `model` to get local data from.
node	index to node to extract local data from.

Value

a numeric vector

Examples

```
## Create an 'SISe' model with 1600 nodes.
model <- SISe(u0 = u0_SISe(), tspan = 1:100, events = events_SISe(),
              phi = 0, upsilon = 1.8e-2, gamma = 0.1, alpha = 1,
              beta_t1 = 1.0e-1, beta_t2 = 1.0e-1, beta_t3 = 1.25e-1,
              beta_t4 = 1.25e-1, end_t1 = c(91, 101), end_t2 = c(182, 185),
              end_t3 = c(273, 275), end_t4 = c(365, 360), epsilon = 0)

## Display local data from the first two nodes.
ldata(model, node = 1)
ldata(model, node = 2)
```

---

mparse — *Model parser to define new models to run in* SimInf

Description

Describe your model in a logical way in R. mparse creates a SimInf_model object with your model definition that is ready to run.

Usage

```
mparse(
  transitions = NULL,
  compartments = NULL,
  ldata = NULL,
  gdata = NULL,
  u0 = NULL,
  v0 = NULL,
  tspan = NULL,
  events = NULL,
  E = NULL,
  N = NULL,
  pts_fun = NULL
)
```

Arguments

transitions — character vector containing transitions on the form "X -> ... -> Y". The left (right) side is the initial (final) state and the propensity is written in between the ->-signs. The special symbol @ is reserved for the empty set. For example, transitions = c("S -> k1*S*I -> I", "I -> k2*I -> R") expresses a SIR model.

compartments	contains the names of the involved compartments, for example, compartments = c("S","I","R").
ldata	optional data for the nodes. Can be specified either as a numeric matrix where column ldata[,j] contains the local data vector for the node j or as a data.frame with one row per node. If it's specified as a matrix, it must have row names to identify the parameters in the transitions. If it's specified as a data.frame, each column is one parameter. The local data vector is passed as an argument to the transition rate functions and the post time step function.
gdata	optional data that are common to all nodes in the model. Can be specified either as a named numeric vector or as as a one-row data.frame. The names are used to identify the parameters in the transitions. The global data vector is passed as an argument to the transition rate functions and the post time step function.
u0	A data.frame (or an object that can be coerced to a data.frame with as.data.frame) with the initial state i.e. the number of individuals in each compartment in each node when the simulation starts..
v0	optional data with the initial continuous state in each node. Can be specified either as a data.frame with one row per node or as a numeric matrix where column v0[,j] contains the initial state vector for the node j. If v0 is specified as a data.frame, each column is one parameter. If v0 is specified as a matrix, the row names identify the parameters. The 'v' vector is passed as an argument to the transition rate functions and the post time step function. The continuous state can be updated in the post time step function.
tspan	A vector (length >= 1) of increasing time points where the state of each node is to be returned. Can be either an integer or a Date vector. A Date vector is coerced to a numeric vector as days, where tspan[1] becomes the day of the year of the first year of tspan. The dates are added as names to the numeric vector.
events	A data.frame with the scheduled events. Default is NULL i.e. no scheduled events in the model.
E	matrix to handle scheduled events, see SimInf_events. Default is NULL i.e. no scheduled events in the model.
N	matrix to handle scheduled events, see SimInf_events. Default is NULL i.e. no scheduled events in the model.
pts_fun	optional character vector with C code for the post time step function. The C code should contain only the body of the function i.e. the code between the opening and closing curly brackets.

Value

a SimInf_model object

Examples

```
## Not run:
## Use the model parser to create a 'SimInf_model' object that
## expresses an SIR model, where 'beta' is the transmission rate
## and 'gamma' is the recovery rate.
```

```
model  <- mparse(transitions = c("S -> beta*S*I/(S+I+R) -> I",
                                 "I -> gamma*I -> R"),
                 compartments = c("S", "I", "R"),
                 gdata = c(beta = 0.16, gamma = 0.077),
                 u0 = data.frame(S = 100, I = 1, R = 0),
                 tspan = 1:100)

## Run and plot the result
set.seed(22)
result <- run(model)
plot(result)

## End(Not run)
```

`Nn` *Extract number of nodes in a model*

Description

Extract number of nodes in a model.

Usage

```
Nn(model)
```

Arguments

`model`	the `model` object to extract the number of nodes from.

Value

the number of nodes in the model.

Examples

```
## Create an 'SIR' model with 100 nodes, with 99 susceptible,
## 1 infected and 0 recovered in each node.
u0 <- data.frame(S = rep(99, 100), I = rep(1, 100), R = rep(0, 100))
model <- SIR(u0 = u0, tspan = 1:10, beta = 0.16, gamma = 0.077)

## Display the number of nodes in the model.
Nn(model)
```

nodes — *Example data with spatial distribution of nodes*

Description

Example data to initialize a population of 1600 nodes and demonstrate various models.

Usage

```
data(nodes)
```

Format

A `data.frame`

Examples

```
## Create an 'SIR' model with 1600 nodes and initialize
## it to run over 4*365 days. Add one infected individual
## to the first node.
u0 <- u0_SIR()
u0$I[1] <- 1
tspan <- seq(from = 1, to = 4*365, by = 1)
model <- SIR(u0     = u0,
             tspan  = tspan,
             events = events_SIR(),
             beta   = 0.16,
             gamma  = 0.077)

## Run the model to generate a single stochastic trajectory.
result <- run(model)

## Determine nodes with one or more infected individuals in the
## trajectory. Extract the 'I' compartment and check for any
## infected individuals in each node.
infected <- colSums(trajectory(result, ~ I, as.is = TRUE)) > 0

## Display infected nodes in 'blue' and non-infected nodes in 'yellow'.
data("nodes", package = "SimInf")
col <- ifelse(infected, "blue", "yellow")
plot(y ~ x, nodes, col = col, pch = 20, cex = 2)
```

outdegree *Determine out-degree for each node in a model*

Description

The number nodes that are connected with *external transfer* events from each node.

Usage

```
outdegree(model)
```

Arguments

model determine out-degree for each node in the model.

Value

vector with out-degree for each node.

Examples

```
## Create an 'SIR' model with 1600 nodes and initialize
## it with example data.
model <- SIR(u0 = u0_SIR(), tspan = 1:1460, events = events_SIR(),
             beta   = 0.16, gamma  = 0.077)

## Display outdegree for each node in the model.
plot(outdegree(model))
```

package_skeleton *Create a package skeleton from a* SimInf_model

Description

Describe your model in a logical way in R, then mparse creates a SimInf_model object with your model definition that can be installed as an add-on R package.

Usage

```
package_skeleton(
  model,
  name = NULL,
  path = ".",
  author = NULL,
  email = NULL,
  maintainer = NULL,
  license = "GPL-3"
)
```

Arguments

`model`	The `model` `SimInf_model` object with your model to create the package skeleton from.
`name`	Character string with the package name. It should contain only (ASCII) letters, numbers and dot, have at least two characters and start with a letter and not end in a dot. The package name is also used for the class name of the model and the directory name of the package.
`path`	Path to put the package directory in. Default is '.' i.e. the current directory.
`author`	Author of the package.
`email`	Email of the package maintainer.
`maintainer`	Maintainer of the package.
`license`	License of the package. Default is 'GPL-3'.

Value

invisible `NULL`.

References

Read the *Writing R Extensions* manual for more details.

Once you have created a *source* package you need to install it: see the *R Installation and Administration* manual, `INSTALL` and `install.packages`.

`pairs,SimInf_model-method` *Scatterplot of number of individuals in each compartment*

Description

A matrix of scatterplots with the number of individuals in each compartment is produced. The `ij`th scatterplot contains `x[,i]` plotted against `x[,j]`.

Usage

```
## S4 method for signature 'SimInf_model'
pairs(x, ...)
```

Arguments

`x`	The `model` to plot
`...`	Additional arguments affecting the plot produced.

Examples

```
## Create an 'SIR' model with 10 nodes and initialise
## it with 99 susceptible individuals and one infected
## individual. Let the model run over 100 days.
model <- SIR(u0 = data.frame(S = rep(99, 10),
                             I = rep(1, 10),
                             R = rep(0, 10)),
             tspan = 1:100,
             beta = 0.16,
             gamma = 0.077)

## Run the model and save the result.
result <- run(model)

## Create a scatter plot
pairs(result)
```

`plot,SimInf_events-method` *Display the distribution of scheduled events over time*

Description

Display the distribution of scheduled events over time

Usage

```
## S4 method for signature 'SimInf_events'
plot(x, frame.plot = FALSE, ...)
```

Arguments

`x`	The events data to plot.
`frame.plot`	Draw a frame around each plot. Default is FALSE.
`...`	Additional arguments affecting the plot

`plot,SimInf_model-method` *Display the outcome from a simulated trajectory*

Description

Plot either the median and the quantile range of the counts in all nodes, or plot the counts in specified nodes.

Usage

```
## S4 method for signature 'SimInf_model'
plot(x, compartments = NULL, node = NULL, range = 0.5, ...)
```

Arguments

x	The model to plot.
compartments	Character vector with the compartments in the model to include in the plot. Default is NULL i.e. include all compartments in the model.
node	Indices specifying the nodes to include when plotting data. Plot one line for each node. Default (node = NULL) is to extract data from all nodes and plot the median count for the specified compartments.
range	Show the quantile range of the count in each compartment. Default is to show the interquartile range i.e. the middle 50% of the count in transparent color. The median value is shown in the same color. Use range = 0.95 to show the middle 95% of the count. To display individual lines for each node, specify range = FALSE.
...	Additional arguments affecting the plot produced.

Examples

```
## Create an 'SIR' model with 100 nodes and initialise
## it with 990 susceptible individuals and 10 infected
## individuals in each node. Run the model over 100 days.
model <- SIR(u0 = data.frame(S = rep(990, 100),
                             I = rep(10, 100),
                             R = rep(0, 100)),
             tspan = 1:100,
             beta = 0.16,
             gamma = 0.077)

## Run the model and save the result.
result <- run(model)

## Plot the median and interquartile range of the number
## of susceptible, infected and recovered individuals.
plot(result)

## Plot the median and the middle 95\% quantile range of the
## number of susceptible, infected and recovered individuals.
plot(result, range = 0.95)

## Plot the median and interquartile range of the  number
## of infected individuals.
plot(result, compartments = "I")

## Plot the number of susceptible, infected
## and recovered individuals in the first
## three nodes.
plot(result, node = 1:3, range = FALSE)
```

```
## Use a stair step plot type instead.
plot(result, node = 1:3, range = FALSE, type = "s")

## Plot the number of infected individuals in the first node.
plot(result, compartments = "I", node = 1, range = FALSE)
```

---

prevalence *Calculate prevalence from a model object with trajectory data*

Description

Calculate the proportion of individuals with disease in the population, or the proportion of nodes with at least one diseased individual, or the proportion of individuals with disease in each node.

Usage

```
prevalence(
  model,
  formula,
  type = c("pop", "nop", "wnp"),
  node = NULL,
  as.is = FALSE
)
```

Arguments

`model`	The `model` with trajectory data to calculate the prevalence from.
`formula`	A formula that specifies the compartments that define the cases with a disease or that have a specific characteristic (numerator), and the compartments that define the entire population of interest (denominator). The left-hand-side of the formula defines the cases, and the right-hand-side defines the population, for example, `I~S+I+R` in a 'SIR' model (see 'Examples'). The `.` (dot) is expanded to all compartments, for example, `I~.` is expanded to `I~S+I+R` in a 'SIR' model (see 'Examples'). The formula can also contain a condition (indicated by `|`) for each node and time step to further control the population to include in the calculation, for example, `I ~ . | R == 0` to calculate the prevalence when the recovered is zero in a 'SIR' model. The condition must evaluate to `TRUE` or `FALSE` in each node and time step. Note that if the denominator is zero, the prevalence is `NaN`.
`type`	The type of prevalence measure to calculate at each time point in `tspan`: `pop` (population prevalence) calculates the proportion of the individuals (cases) in the population, `nop` (node prevalence) calculates the proportion of nodes with at least one case, and `wnp` (within-node prevalence) calculates the proportion of cases within each node. Default is `pop`.
`node`	Indices specifying the subset nodes to include in the calculation of the prevalence. Default is `NULL`, which includes all nodes.

as.is — The default (as.is = FALSE) is to generate a data.frame with one row per time-step with the prevalence. Using as.is = TRUE returns the result as a matrix, which is the internal format.

Value

A data.frame if as.is = FALSE, else a matrix.

Examples

```
## Create an 'SIR' model with 6 nodes and initialize
## it to run over 10 days.
u0 <- data.frame(S = 100:105, I = c(0, 1, 0, 2, 0, 3), R = rep(0, 6))
model <- SIR(u0 = u0, tspan = 1:10, beta = 0.16, gamma = 0.077)

## Run the model to generate a single stochastic trajectory.
result <- run(model)

## Determine the proportion of infected individuals (cases)
## in the population at the time-points in 'tspan'.
prevalence(result, I~S+I+R)

## Identical result is obtained with the shorthand 'I~.'
prevalence(result, I~.)

## Determine the proportion of nodes with infected individuals at
## the time-points in 'tspan'.
prevalence(result, I~S+I+R, type = "nop")

## Determine the proportion of infected individuals in each node
## at the time-points in 'tspan'.
prevalence(result, I~S+I+R, type = "wnp")

## Determine the proportion of infected individuals in each node
## at the time-points in 'tspan' when the number of recovered is
## zero.
prevalence(result, I~S+I+R|R==0, type = "wnp")
```

---

punchcard<- *Set a template for where to record result during a simualtion*

Description

Using a sparse result matrix can save a lot of memory if the model contains many nodes and time-points, but where only a few of the data points are of interest for post-processing.

Usage

```
punchcard(model) <- value
```

Arguments

model
: The model to set a template for where to record result.

value
: A data.frame that specify the nodes, time-points and compartments to record the number of individuals at tspan. Use NULL to reset the model to record the number of inidividuals in each compartment in every node at each time-point in tspan.

Details

Using a sparse result matrix can save a lot of memory if the model contains many nodes and time-points, but where only a few of the data points are of interest for post-processing. To use this feature, a template has to be defined for which data points to record. This is done using a data.frame that specifies the time-points (column 'time') and nodes (column 'node') to record the state of the compartments, see 'Examples'. The specified time-points, nodes and compartments must exist in the model, or an error is raised. Note that specifying a template only affects which data-points are recorded for post-processing, it does not affect how the solver simulates the trajectory.

Examples

```
## Create an 'SIR' model with 6 nodes and initialize it to run over 10 days.
u0 <- data.frame(S = 100:105, I = 1:6, R = rep(0, 6))
model <- SIR(u0 = u0, tspan = 1:10, beta = 0.16, gamma = 0.077)

## Run the model.
result <- run(model)

## Display the trajectory with data for every node at each
## time-point in tspan.
trajectory(result)

## Assume we are only interested in nodes '2' and '4' at the
## time-points '3' and '5'
df <- data.frame(time = c(3, 5, 3, 5),
                 node = c(2, 2, 4, 4),
                 S = c(TRUE, TRUE, TRUE, TRUE),
                 I = c(TRUE, TRUE, TRUE, TRUE),
                 R = c(TRUE, TRUE, TRUE, TRUE))
punchcard(model) <- df
result <- run(model)
trajectory(result)

## We can also specify to record only some of the compartments in
## each time-step.
df <- data.frame(time = c(3, 5, 3, 5),
                 node = c(2, 2, 4, 4),
                 S = c(FALSE, TRUE, TRUE, TRUE),
                 I = c(TRUE, FALSE, TRUE, FALSE),
                 R = c(TRUE, FALSE, TRUE, TRUE))
punchcard(model) <- df
result <- run(model)
trajectory(result)
```

```
## A shortcut to specify to record all of the compartments in
## each time-step is to only inlude node and time.
df <- data.frame(time = c(3, 5, 3, 5),
                 node = c(2, 2, 4, 4))
punchcard(model) <- df
result <- run(model)
trajectory(result)

## It is possible to use an empty 'data.frame' to specify
## that no data-points should be recorded for the trajectory.
punchcard(model) <- data.frame()
result <- run(model)
trajectory(result)

## Use 'NULL' to reset the model to record data for every node at
## each time-point in tspan.
punchcard(model) <- NULL
result <- run(model)
trajectory(result)
```

run *Run the SimInf stochastic simulation algorithm*

Description

Run the SimInf stochastic simulation algorithm

Usage

```
run(model, ...)

## S4 method for signature 'SimInf_model'
run(model, solver = c("ssm", "aem"), ...)

## S4 method for signature 'SEIR'
run(model, solver = c("ssm", "aem"), ...)

## S4 method for signature 'SIR'
run(model, solver = c("ssm", "aem"), ...)

## S4 method for signature 'SISe'
run(model, solver = c("ssm", "aem"), ...)

## S4 method for signature 'SISe3'
run(model, solver = c("ssm", "aem"), ...)

## S4 method for signature 'SISe3_sp'
run(model, solver = c("ssm", "aem"), ...)
```

```
## S4 method for signature 'SISe_sp'
run(model, solver = c("ssm", "aem"), ...)
```

Arguments

model	The SimInf model to run.
...	Additional arguments.
solver	Which numerical solver to utilize. Default is 'ssm'.

Value

SimInf_model object with result from simulation.

References

- Bauer P, Engblom S, Widgren S (2016) "Fast Event-Based Epidemiological Simulations on National Scales" International Journal of High Performance Computing Applications, 30(4), 438-453. doi:10.1177/1094342016635723
- Bauer P., Engblom S. (2015) Sensitivity Estimation and Inverse Problems in Spatial Stochastic Models of Chemical Kinetics. In: Abdulle A., Deparis S., Kressner D., Nobile F., Picasso M. (eds) Numerical Mathematics and Advanced Applications - ENUMATH 2013. Lecture Notes in Computational Science and Engineering, vol 103. Springer, Cham. Doi: 10.1007/978-3-319-10705-9_51

Examples

```
## Create an 'SIR' model with 10 nodes and initialise
## it to run over 100 days.
model <- SIR(u0 = data.frame(S = rep(99, 10),
                             I = rep(1, 10),
                             R = rep(0, 10)),
             tspan = 1:100,
             beta = 0.16,
             gamma = 0.077)

## Run the model and save the result.
result <- run(model)

## Plot the proportion of susceptible, infected and recovered
## individuals.
plot(result)
```

SEIR	*Create an* SEIR *model*

Description

Create an SEIR model to be used by the simulation framework.

Usage

```
SEIR(u0, tspan, events = NULL, beta = NULL, epsilon = NULL, gamma = NULL)
```

Arguments

`u0`	A `data.frame` with the initial state in each node (see 'Details').
`tspan`	A vector (length >= 1) of increasing time points where the state of each node is to be returned. Can be either an `integer` or a `Date` vector. A `Date` vector is coerced to a numeric vector as days, where `tspan[1]` becomes the day of the year of the first year of `tspan`. The dates are added as names to the numeric vector.
`events`	a `data.frame` with the scheduled events, see `SimInf_model`.
`beta`	A numeric vector with the transmission rate from susceptible to infected where each node can have a different beta value. The vector must have length 1 or `nrow(u0)`. If the vector has length 1, but the model contains more nodes, the beta value is repeated in all nodes.
`epsilon`	A numeric vector with the incubation rate from exposed to infected where each node can have a different epsilon value. The vector must have length 1 or `nrow(u0)`. If the vector has length 1, but the model contains more nodes, the epsilon value is repeated in all nodes.
`gamma`	A numeric vector with the recovery rate from infected to recovered where each node can have a different gamma value. The vector must have length 1 or `nrow(u0)`. If the vector has length 1, but the model contains more nodes, the beta value is repeated in all nodes.

Details

The SEIR model contains four compartments; number of susceptible (S), number of exposed (E) (those who have been infected but are not yet infectious), number of infectious (I), and number of recovered (R). Moreover, it has three state transitions,

$$S \xrightarrow{\beta SI/N} E$$

$$E \xrightarrow{\epsilon E} I$$

$$I \xrightarrow{\gamma I} R$$

where β is the transmission rate, ϵ is the incubation rate, γ is the recovery rate, and $N = S + E + I + R$.

The argument `u0` must be a `data.frame` with one row for each node with the following columns:

S The number of sucsceptible in each node

E The number of exposed in each node

I The number of infected in each node

R The number of recovered in each node

Value

A `SimInf_model` of class `SEIR`

Examples

```
## Create a SEIR model object.
model <- SEIR(u0 = data.frame(S = 99, E = 0, I = 1, R = 0),
              tspan = 1:100,
              beta = 0.16,
              epsilon = 0.25,
              gamma = 0.077)

## Run the SEIR model and plot the result.
set.seed(3)
result <- run(model)
plot(result)
```

`SEIR-class` *Definition of the 'SEIR' model*

Description

Class to handle the SEIR `SimInf_model`.

`select_matrix` *Extract the select matrix from a* `SimInf_model` *object*

Description

Utility function to extract `events@E` from a `SimInf_model` object, see `SimInf_events`

Usage

```
select_matrix(model)
```

Arguments

`model`	The `model` to extract the select matrix `E` from.

Value

dgCMatrix object.

Examples

```
## Create an SIR model
model <- SIR(u0 = data.frame(S = 99, I = 1, R = 0),
             tspan = 1:5, beta = 0.16, gamma = 0.077)

## Extract the select matrix from the model
select_matrix(model)
```

---

select_matrix<- *Set the select matrix for a* SimInf_model *object*

Description

Utility function to set events@E in a SimInf_model object, see SimInf_events

Usage

```
select_matrix(model) <- value
```

Arguments

model	The model to set the select matrix for.
value	A matrix.

Examples

```
## Create an SIR model
model <- SIR(u0 = data.frame(S = 99, I = 1, R = 0),
             tspan = 1:5, beta = 0.16, gamma = 0.077)

## Set the select matrix
select_matrix(model) <- matrix(c(1, 0, 0, 1, 1, 1, 0, 0, 1), nrow = 3)

## Extract the select matrix from the model
select_matrix(model)
```

set_num_threads — *Specify the number of threads that SimInf should use*

Description

Set the number of threads to be used in SimInf code that is parallelized with OpenMP (if available). The number of threads is initialized when SimInf is first loaded in the R session using optional envioronment variables (see 'Details'). It is also possible to specify the number of threads by calling `set_num_threads`. If the environment variables that affect the number of threads change, then `set_num_threads` must be called again for it to take effect.

Usage

```
set_num_threads(threads = NULL)
```

Arguments

`threads`	integer with maximum number of threads to use in functions that are parallelized with OpenMP (if available). Default is NULL, i.e. to use all available processors and then check for limits in the environment varibles (see 'Details').

Details

The `omp_get_num_procs()` function is used to determine the number of processors that are available to the device at the time the routine is called. The number of threads is then limited by `omp_get_thread_limit()` and the current values of the environmental variables (if set)

- `Sys.getenv("OMP_THREAD_LIMIT")`
- `Sys.getenv("OMP_NUM_THREADS")`
- `Sys.getenv("SIMINF_NUM_THREADS")`

Additionally, the maximum number of threads can be controlled by the `threads` argument, given that its value is not above any of the limits described above.

Value

The previous value is returned (invisible).

shift_matrix — *Extract the shift matrix from a* SimInf_model *object*

Description

Utility function to extract the shift matrix events@N from a SimInf_model object, see SimInf_events

Usage

```
shift_matrix(model)
```

Arguments

model	The model to extract the shift matrix events@N from.

Value

A mtrix.

Examples

```
## Create an SIR model
model <- SIR(u0 = data.frame(S = 99, I = 1, R = 0),
             tspan = 1:5, beta = 0.16, gamma = 0.077)

## Extract the shift matrix from the model
shift_matrix(model)
```

shift_matrix<- — *Set the shift matrix for a* SimInf_model *object*

Description

Utility function to set events@N in a SimInf_model object, see SimInf_events

Usage

```
shift_matrix(model) <- value
```

Arguments

model	The model to set the shift matrix events@N.
value	A matrix.

Value

SimInf_model object

Examples

```
## Create an SIR model
model <- SIR(u0 = data.frame(S = 99, I = 1, R = 0),
             tspan = 1:5, beta = 0.16, gamma = 0.077)

## Set the shift matrix
shift_matrix(model) <- matrix(c(2, 1, 0), nrow = 3)

## Extract the shift matrix from the model
shift_matrix(model)
```

`show,SimInf_events-method`

Brief summary of `SimInf_events`

Description

Shows the number of scheduled events.

Usage

```
## S4 method for signature 'SimInf_events'
show(object)
```

Arguments

`object`	The SimInf_events `object`

Value

None (invisible 'NULL').

`show,SimInf_model-method`

Brief summary of `SimInf_model`

Description

Brief summary of `SimInf_model`

Usage

```
## S4 method for signature 'SimInf_model'
show(object)
```

Arguments

`object`	The SimInf_model object

Value

None (invisible 'NULL').

Examples

```
## Create an 'SIR' model with 10 nodes and initialise
## it to run over 100 days.
model <- SIR(u0 = data.frame(S = rep(99, 10),
                             I = rep(1, 10),
                             R = rep(0, 10)),
             tspan = 1:100,
             beta = 0.16,
             gamma = 0.077)

## Brief summary of the model
model

## Run the model and save the result
result <- run(model)

## Brief summary of the result. Note that 'U' and 'V' are
## non-empty after running the model.
result
```

`SimInf` *A Framework for Data-Driven Stochastic Disease Spread Simulations*

Description

The SimInf package provides a flexible framework for data-driven spatio-temporal disease spread modeling, designed to efficiently handle population demographics and network data. The framework integrates infection dynamics in each subpopulation as continuous-time Markov chains (CTMC) using the Gillespie stochastic simulation algorithm (SSA) and incorporates available data such as births, deaths or movements as scheduled events. A scheduled event is used to modify the state of a subpopulation at a predefined time-point.

Details

The `SimInf_model` is central and provides the basis for the framework. A `SimInf_model` object supplies the state-change matrix, the dependency graph, the scheduled events, and the initial state of the system.

All predefined models in SimInf have a generating function, with the same name as the model, for example `SIR`.

A model can also be created from a model specification using the `mparse` method.

After a model is created, a simulation is started with a call to the run method and if execution is successful, it returns a modified SimInf_model object with a single stochastic solution trajectory attached to it.

SimInf provides several utility functions to inspect simulated data, for example, show, summary and plot. To facilitate custom analysis, it provides the trajectory and prevalence methods.

One of our design goal was to make SimInf extendable and enable usage of the numerical solvers from other R extension packages in order to facilitate complex epidemiological research. To support this, SimInf has functionality to generate the required C and R code from a model specification, see package_skeleton

---

SimInf_events *Create a SimInf_events object*

Description

The argument events must be a data.frame with the following columns:

event Four event types are supported by the current solvers: *exit*, *enter*, *internal transfer*, and *external transfer*. When assigning the events, they can either be coded as a numerical value or a character string: *exit;* 0 or 'exit', *enter;* 1 or 'enter', *internal transfer;* 2 or 'intTrans', and *external transfer;* 3 or 'extTrans'. Internally in **SimInf**, the event type is coded as a numerical value.

time When the event occurs i.e., the event is processed when time is reached in the simulation. Can be either an integer or a Date vector. A Date vector is coerced to a numeric vector as days, where t0 determines the offset to match the time of the events to the model tspan vector.

node The node that the event operates on. Also the source node for an *external transfer* event. 1 <= node[i] <= Number of nodes.

dest The destination node for an *external transfer* event i.e., individuals are moved from node to dest, where 1 <= dest[i] <= Number of nodes. Set event = 0 for the other event types. dest is an integer vector.

n The number of individuals affected by the event. n[i] >= 0.

proportion If n[i] equals zero, the number of individuals affected by event[i] is calculated by sampling the number of individuals from a binomial distribution using the proportion[i] and the number of individuals in the compartments. Numeric vector. 0 <= proportion[i] <= 1.

select To process an event[i], the compartments affected by the event are specified with select[i] together with the matrix E, where select[i] determines which column in E to use. The specific individuals affected by the event are sampled from the compartments corresponding to the non-zero entries in the specified column in E[,select[i]], where select is an integer vector.

shift Determines how individuals in *internal transfer* and *external transfer* events are shifted to enter another compartment. The sampled individuals are shifted according to column shift[i] in matrix N i.e., N[,shift[i]], where shift is an integer vector. See above for a description of N. Unused for the other event types.

Usage

```
SimInf_events(E = NULL, N = NULL, events = NULL, t0 = NULL)
```

Arguments

E	Each row corresponds to one compartment in the model. The non-zero entries in a column indicates the compartments to include in an event. For the *exit*, *internal transfer* and *external transfer* events, a non-zero entry indicate the compartments to sample individuals from. For the *enter* event, all individuals enter first non-zero compartment. `E` is sparse matrix of class `dgCMatrix`.
N	Determines how individuals in *internal transfer* and *external transfer* events are shifted to enter another compartment. Each row corresponds to one compartment in the model. The values in a column are added to the current compartment of sampled individuals to specify the destination compartment, for example, a value of `1` in an entry means that sampled individuals in this compartment are moved to the next compartment. Which column to use for each event is specified by the `shift` vector (see below). `N` is an integer matrix.
events	A `data.frame` with events.
t0	If `events$time` is a `Date` vector, then `t0` determines the offset to match the time of the events to the model `tspan` vector, see details. If `events$time` is a numeric vector, then `t0` must be `NULL`.

Value

S4 class `SimInf_events`

Examples

```
## Let us illustrate how movement events can be used to transfer
## individuals from one node to another.  Use the built-in SIR
## model and start with 2 nodes where all individuals are in the
## first node (100 per compartment).
u0 <- data.frame(S = c(100, 0), I = c(100, 0), R = c(100, 0))

## Then create 300 movement events to transfer all individuals,
## one per day, from the first node to the second node. Use the
## fourth column in the select matrix where all compartments
## can be sampled with equal weight.
events <- data.frame(event      = rep("extTrans", 300),
                     time       = 1:300,
                     node       = 1,
                     dest       = 2,
                     n          = 1,
                     proportion = 0,
                     select     = 4,
                     shift      = 0)

## Create an SIR model without disease transmission to
## demonstrate the events.
```

```
model <- SIR(u0     = u0,
             tspan  = 1:300,
             events = events,
             beta   = 0,
             gamma  = 0)

## Run the model and plot the number of individuals in
## the second node.  As can be seen in the figure, all
## indivuduals have been moved to the second node when
## t = 300.
plot(run(model), node = 1:2, range = FALSE)

## Let us now double the weight to sample from the 'I'
## compartment and rerun the model.
model@events@E[2, 4] <- 2
plot(run(model), node = 1:2, range = FALSE)

## And much larger weight to sample from the I compartment.
model@events@E[2, 4] <- 10
plot(run(model), node = 1:2, range = FALSE)

## Increase the weight for the R compartment.
model@events@E[3, 4] <- 4
plot(run(model), node = 1:2, range = FALSE)
```

SimInf_events-class *Class* "SimInf_events"

Description

Class to hold data for scheduled events to modify the discrete state of individuals in a node at a pre-defined time t.

Slots

E Each row corresponds to one compartment in the model. The non-zero entries in a column indicates the compartments to include in an event. For the *exit*, *internal transfer* and *external transfer* events, a non-zero entry indicate the compartments to sample individuals from. For the *enter* event, all individuals enter first non-zero compartment. E is sparse matrix of class dgCMatrix.

N Determines how individuals in *internal transfer* and *external transfer* events are shifted to enter another compartment. Each row corresponds to one compartment in the model. The values in a column are added to the current compartment of sampled individuals to specify the destination compartment, for example, a value of 1 in an entry means that sampled individuals in this compartment are moved to the next compartment. Which column to use for each event is specified by the shift vector (see below). N is an integer matrix.

event Type of event: 0) *exit*, 1) *enter*, 2) *internal transfer*, and 3) *external transfer*. Other values are reserved for future event types and not supported by the current solvers. Integer vector.

`time` Time of when the event occurs i.e., the event is processed when time is reached in the simulation. `time` is an integer vector.

`node` The node that the event operates on. Also the source node for an *external transfer* event. Integer vector. 1 <= `node[i]` <= Number of nodes.

`dest` The destination node for an *external transfer* event i.e., individuals are moved from `node` to `dest`, where 1 <= `dest[i]` <= Number of nodes. Set `event = 0` for the other event types. `dest` is an integer vector.

`n` The number of individuals affected by the event. Integer vector. n[i] >= 0.

`proportion` If `n[i]` equals zero, the number of individuals affected by `event[i]` is calculated by sampling the number of individuals from a binomial distribution using the `proportion[i]` and the number of individuals in the compartments. Numeric vector. 0 <= proportion[i] <= 1.

`select` To process `event[i]`, the compartments affected by the event are specified with `select[i]` together with the matrix `E`, where `select[i]` determines which column in `E` to use. The specific individuals affected by the event are proportionally sampled from the compartments corresponding to the non-zero entries in the specified column in `E[,select[i]]`, where `select` is an integer vector.

`shift` Determines how individuals in *internal transfer* and *external transfer* events are shifted to enter another compartment. The sampled individuals are shifted according to column `shift[i]` in matrix `N` i.e., `N[,shift[i]]`, where `shift` is an integer vector. See above for a description of `N`. Unsued for the other event types.

`SimInf_model` *Create a* `SimInf_model`

Description

Create a `SimInf_model`

Usage

```
SimInf_model(
  G,
  S,
  tspan,
  events = NULL,
  ldata = NULL,
  gdata = NULL,
  U = NULL,
  u0 = NULL,
  v0 = NULL,
  V = NULL,
  E = NULL,
  N = NULL,
  C_code = NULL
)
```

Arguments

G
: Dependency graph that indicates the transition rates that need to be updated after a given state transition has occured. A non-zero entry in element `G[i,i]` indicates that transition rate `i` needs to be recalculated if the state transition `j` occurs. Sparse matrix ($Nt \times Nt$) of object class `dgCMatrix`.

S
: Each column corresponds to a transition, and execution of state transition `j` amounts to adding the `S[,j]` to the state vector of the node where the state transition occurred. Sparse matrix ($Nc \times Nt$) of object class `dgCMatrix`.

tspan
: A vector (length >= 1) of increasing time points where the state of each node is to be returned. Can be either an `integer` or a `Date` vector. A `Date` vector is coerced to a numeric vector as days, where `tspan[1]` becomes the day of the year of the first year of `tspan`. The dates are added as names to the numeric vector.

events
: A `data.frame` with the scheduled events.

ldata
: local data for the nodes. Can either be specified as a `data.frame` with one row per node. Or as a matrix where each column `ldata[,j]` contains the local data vector for the node `j`. The local data vector is passed as an argument to the transition rate functions and the post time step function.

gdata
: A numeric vector with global data that is common to all nodes. The global data vector is passed as an argument to the transition rate functions and the post time step function.

U
: The result matrix with the number of individuals in each disease state in every node ($N_n N_c \times$ `length(tspan)`). `U[,j]` contains the number of individuals in each disease state at `tspan[j]`. `U[1:Nc,j]` contains the state of node `1` at `tspan[j]`. `U[(Nc + 1):(2 * Nc),j]` contains the state of node `2` at `tspan[j]` etc.

u0
: The initial state vector. Either a matrix ($N_c \times N_n$) or a a `data.frame` with the number of individuals in each compartment in every node.

v0
: The initial continuous state vector in every node. (`dim(ldata)[1]` $\times N_N$). The continuous state vector is updated by the specific model during the simulation in the post time step function.

V
: The result matrix for the real-valued continous compartment state (N_n`dim(ldata)[1]` $\times$ `length(tspan)`). `V[,j]` contains the real-valued state of the system at `tspan[j]`.

E
: Sparse matrix to handle scheduled events, see `SimInf_events`.

N
: Sparse matrix to handle scheduled events, see `SimInf_events`.

C_code
: Character vector with optional model C code. If non-empty, the C code is written to a temporary C-file when the `run` method is called. The temporary C-file is compiled and the resulting DLL is dynamically loaded. The DLL is unloaded and the temporary files are removed after running the model.

Value

SimInf_model

SimInf_model-class *Class* "SimInf_model"

Description

Class to handle data for the SimInf_model.

Slots

G Dependency graph that indicates the transition rates that need to be updated after a given state transition has occured. A non-zero entry in element G[i,i] indicates that transition rate i needs to be recalculated if the state transition j occurs. Sparse matrix ($Nt \times Nt$) of object class dgCMatrix.

S Each column corresponds to a state transition, and execution of state transition j amounts to adding the S[,j] column to the state vector u[,i] of node *i* where the transition occurred. Sparse matrix ($Nc \times Nt$) of object class dgCMatrix.

U The result matrix with the number of individuals in each compartment in every node. U[,j] contains the number of individuals in each compartment at tspan[j]. U[1:Nc,j] contains the number of individuals in node 1 at tspan[j]. U[(Nc + 1):(2 * Nc),j] contains the number of individuals in node 2 at tspan[j] etc. Integer matrix ($N_n N_c \times$ length(tspan)).

U_sparse If the model was configured to write the solution to a sparse matrix (dgCMatrix) the U_sparse contains the data and U is empty. The layout of the data in U_sparse is identical to U. Please note that U_sparse is numeric and U is integer.

V The result matrix for the real-valued continuous state. V[,j] contains the real-valued state of the system at tspan[j]. Numeric matrix ($N_n$dim(ldata)[1] $\times$ length(tspan)).

V_sparse If the model was configured to write the solution to a sparse matrix (dgCMatrix) the V_sparse contains the data and V is empty. The layout of the data in V_sparse is identical to V.

ldata A matrix with local data for the nodes. The column ldata[,j] contains the local data vector for the node j. The local data vector is passed as an argument to the transition rate functions and the post time step function.

gdata A numeric vector with global data that is common to all nodes. The global data vector is passed as an argument to the transition rate functions and the post time step function.

tspan A vector of increasing time points where the state of each node is to be returned.

u0 The initial state vector ($N_c \times N_n$) with the number of individuals in each compartment in every node.

v0 The initial value for the real-valued continuous state. Numeric matrix (dim(ldata)[1] $\times N_n$).

events Scheduled events SimInf_events

C_code Character vector with optional model C code. If non-empty, the C code is written to a temporary C-file when the run method is called. The temporary C-file is compiled and the resulting DLL is dynamically loaded. The DLL is unloaded and the temporary files are removed after running the model.

SIR *Create an* SIR *model*

Description

Create an SIR model to be used by the simulation framework.

Usage

```
SIR(u0, tspan, events = NULL, beta = NULL, gamma = NULL)
```

Arguments

`u0`	A `data.frame` with the initial state in each node (see 'Details').
`tspan`	A vector (length >= 1) of increasing time points where the state of each node is to be returned. Can be either an `integer` or a `Date` vector. A `Date` vector is coerced to a numeric vector as days, where `tspan[1]` becomes the day of the year of the first year of `tspan`. The dates are added as names to the numeric vector.
`events`	a `data.frame` with the scheduled events, see `SimInf_model`.
`beta`	A numeric vector with the transmission rate from susceptible to infected where each node can have a different beta value. The vector must have length 1 or `nrow(u0)`. If the vector has length 1, but the model contains more nodes, the beta value is repeated in all nodes.
`gamma`	A numeric vector with the recovery rate from infected to recovered where each node can have a different gamma value. The vector must have length 1 or `nrow(u0)`. If the vector has length 1, but the model contains more nodes, the beta value is repeated in all nodes.

Details

The SIR model contains three compartments; number of susceptible (S), number of infectious (I), and number of recovered (R). Moreover, it has two state transitions,

$$S \stackrel{\beta SI/N}{\longrightarrow} I$$

$$I \stackrel{\gamma I}{\longrightarrow} R$$

where β is the transmission rate, γ is the recovery rate, and $N = S + I + R$.

The argument `u0` must be a `data.frame` with one row for each node with the following columns:

S The number of sucsceptible in each node

I The number of infected in each node

R The number of recovered in each node

Value

A SimInf_model of class SIR

Examples

```
## Create an SIR model object.
model <- SIR(u0 = data.frame(S = 99, I = 1, R = 0),
             tspan = 1:100,
             beta = 0.16,
             gamma = 0.077)

## Run the SIR model and plot the result.
set.seed(22)
result <- run(model)
plot(result)
```

SIR-class   *Definition of the SIR model*

Description

Class to handle the SIR SimInf_model.

Details

The SIR model contains three compartments; number of susceptible (S), number of infectious (I), and number of recovered (R). Moreover, it has two state transitions,

$$S \xrightarrow{\beta SI/N} I$$

$$I \xrightarrow{\gamma I} R$$

where β is the transmission rate, γ is the recovery rate, and $N = S + I$.

Examples

```
## Create an SIR model object.
model <- SIR(u0 = data.frame(S = 99, I = 1, R = 0),
             tspan = 1:100,
             beta = 0.16,
             gamma = 0.077)

## Run the SIR model and plot the result.
set.seed(22)
result <- run(model)
plot(result)
```

SISe *Create a SISe model*

Description

Create an 'SISe' model to be used by the simulation framework.

Usage

```
SISe(
  u0,
  tspan,
  events = NULL,
  phi = NULL,
  upsilon = NULL,
  gamma = NULL,
  alpha = NULL,
  beta_t1 = NULL,
  beta_t2 = NULL,
  beta_t3 = NULL,
  beta_t4 = NULL,
  end_t1 = NULL,
  end_t2 = NULL,
  end_t3 = NULL,
  end_t4 = NULL,
  epsilon = NULL
)
```

Arguments

u0	A data.frame with the initial state in each node (see 'Details').
tspan	A vector (length >= 1) of increasing time points where the state of each node is to be returned. Can be either an integer or a Date vector. A Date vector is coerced to a numeric vector as days, where tspan[1] becomes the day of the year of the first year of tspan. The dates are added as names to the numeric vector.
events	a data.frame with the scheduled events, see SimInf_model.
phi	A numeric vector with the initial environmental infectious pressure in each node. Will be repeated to the length of nrow(u0). Default is NULL which gives 0 in each node.
upsilon	Indirect transmission rate of the environmental infectious pressure
gamma	The recovery rate from infected to susceptible
alpha	Shed rate from infected individuals
beta_t1	The decay of the environmental infectious pressure in interval 1.
beta_t2	The decay of the environmental infectious pressure in interval 2.

beta_t3	The decay of the environmental infectious pressure in interval 3.
beta_t4	The decay of the environmental infectious pressure in interval 4.
end_t1	vector with the non-inclusive day of the year that ends interval 1 in each node. Will be repeated to the length of nrow(u0).
end_t2	vector with the non-inclusive day of the year that ends interval 2 in each node. Will be repeated to the length of nrow(u0).
end_t3	vector with the non-inclusive day of the year that ends interval 3 in each node. Will be repeated to the length of nrow(u0).
end_t4	vector with the non-inclusive day of the year that ends interval 4 in each node. Will be repeated to the length of nrow(u0).
epsilon	The background environmental infectious pressure

Details

The 'SISe' model contains two compartments; number of susceptible (S) and number of infectious (I). Additionally, it contains an environmental compartment to model shedding of a pathogen to the environment. Consequently, the model has two state transitions,

$$S \xrightarrow{\upsilon \varphi S} I$$

$$I \xrightarrow{\gamma I} S$$

where the transition rate per unit of time from susceptible to infected is proportional to the concentration of the environmental contamination φ in each node. Moreover, the transition rate from infected to susceptible is the recovery rate γ, measured per individual and per unit of time. Finally, the environmental infectious pressure in each node is evolved by,

$$\frac{d\varphi(t)}{dt} = \frac{\alpha I(t)}{N(t)} - \beta(t)\varphi(t) + \epsilon$$

where α is the average shedding rate of the pathogen to the environment per infected individual and $N = S + I$ the size of the node. The seasonal decay and removal of the pathogen is captured by $\beta(t)$. It is also possible to include a small background infectious pressure ϵ to allow for other indirect sources of environmental contamination. The environmental infectious pressure $\varphi(t)$ in each node is evolved each time unit by the Euler forward method. The value of $\varphi(t)$ is saved at the time-points specified in `tspan`.

The argument `u0` must be a `data.frame` with one row for each node with the following columns:

S The number of sucsceptible in each node

I The number of infected in each node

Value

`SISe`

Beta

The time dependent beta is divided into four intervals of the year

```
where 0 <= day < 365

Case 1: END_1 < END_2 < END_3 < END_4
INTERVAL_1 INTERVAL_2     INTERVAL_3     INTERVAL_4     INTERVAL_1
[0, END_1) [END_1, END_2) [END_2, END_3) [END_3, END_4) [END_4, 365)

Case 2: END_3 < END_4 < END_1 < END_2
INTERVAL_3 INTERVAL_4     INTERVAL_1     INTERVAL_2     INTERVAL_3
[0, END_3) [END_3, END_4) [END_4, END_1) [END_1, END_2) [END_2, 365)

Case 3: END_4 < END_1 < END_2 < END_3
INTERVAL_4 INTERVAL_1     INTERVAL_2     INTERVAL_3     INTERVAL_4
[0, END_4) [END_4, END_1) [END_1, END_2) [END_2, END_3) [END_3, 365)
```

SISe-class — *Definition of the* SISe *model*

Description

Class to handle the SISe SimInf_model.

SISe3 — *Create a* SISe3 *model*

Description

Create a SISe3 model to be used by the simulation framework.

Usage

```
SISe3(
  u0,
  tspan,
  events = NULL,
  phi = NULL,
  upsilon_1 = NULL,
  upsilon_2 = NULL,
  upsilon_3 = NULL,
  gamma_1 = NULL,
  gamma_2 = NULL,
  gamma_3 = NULL,
  alpha = NULL,
```

```
  beta_t1 = NULL,
  beta_t2 = NULL,
  beta_t3 = NULL,
  beta_t4 = NULL,
  end_t1 = NULL,
  end_t2 = NULL,
  end_t3 = NULL,
  end_t4 = NULL,
  epsilon = NULL
)
```

Arguments

u0	A data.frame with the initial state in each node (see 'Details').
tspan	A vector (length >= 1) of increasing time points where the state of each node is to be returned. Can be either an integer or a Date vector. A Date vector is coerced to a numeric vector as days, where tspan[1] becomes the day of the year of the first year of tspan. The dates are added as names to the numeric vector.
events	a data.frame with the scheduled events, see SimInf_model.
phi	A numeric vector with the initial environmental infectious pressure in each node. Will be repeated to the length of nrow(u0). Default is NULL which gives 0 in each node.
upsilon_1	Indirect transmission rate of the environmental infectious pressure in age category 1
upsilon_2	Indirect transmission rate of the environmental infectious pressure in age category 2
upsilon_3	Indirect transmission rate of the environmental infectious pressure in age category 3
gamma_1	The recovery rate from infected to susceptible for age category 1
gamma_2	The recovery rate from infected to susceptible for age category 2
gamma_3	The recovery rate from infected to susceptible for age category 3
alpha	Shed rate from infected individuals
beta_t1	The decay of the environmental infectious pressure in interval 1.
beta_t2	The decay of the environmental infectious pressure in interval 2.
beta_t3	The decay of the environmental infectious pressure in interval 3.
beta_t4	The decay of the environmental infectious pressure in interval 4.
end_t1	vector with the non-inclusive day of the year that ends interval 1 in each node. Will be repeated to the length of nrow(u0).
end_t2	vector with the non-inclusive day of the year that ends interval 2 in each node. Will be repeated to the length of nrow(u0).
end_t3	vector with the non-inclusive day of the year that ends interval 3 in each node. Will be repeated to the length of nrow(u0).

`end_t4`	vector with the non-inclusive day of the year that ends interval 4 in each node. Will be repeated to the length of nrow(u0).
`epsilon`	The background environmental infectious pressure

Details

The `SISe3` model contains two compartments in three age categories; number of susceptible (S_1, S_2, S_3) and number of infectious (I_1, I_2, I_3). Additionally, it contains an environmental compartment to model shedding of a pathogen to the environment. Consequently, the model has six state transitions,

$$S_1 \stackrel{\upsilon_1 \varphi S_1}{\longrightarrow} I_1$$

$$I_1 \stackrel{\gamma_1 I_1}{\longrightarrow} S_1$$

$$S_2 \stackrel{\upsilon_2 \varphi S_2}{\longrightarrow} I_2$$

$$I_2 \stackrel{\gamma_2 I_2}{\longrightarrow} S_2$$

$$S_3 \stackrel{\upsilon_3 \varphi S_3}{\longrightarrow} I_3$$

$$I_3 \stackrel{\gamma_3 I_3}{\longrightarrow} S_3$$

where the transition rate per unit of time from susceptible to infected is proportional to the concentration of the environmental contamination φ in each node. Moreover, the transition rate from infected to susceptible is the recovery rate $\gamma_1, \gamma_2, \gamma_3$, measured per individual and per unit of time. Finally, the environmental infectious pressure in each node is evolved by,

$$\frac{d\varphi(t)}{dt} = \frac{\alpha \left(I_1(t) + I_2(t) + I_3(t)\right)}{N(t)} - \beta(t)\varphi(t) + \epsilon$$

where α is the average shedding rate of the pathogen to the environment per infected individual and $N = S_1 + S_2 + S_3 + I_1 + I_2 + I_3$ the size of the node. The seasonal decay and removal of the pathogen is captured by $\beta(t)$. It is also possible to include a small background infectious pressure ϵ to allow for other indirect sources of environmental contamination. The environmental infectious pressure $\varphi(t)$ in each node is evolved each time unit by the Euler forward method. The value of $\varphi(t)$ is saved at the time-points specified in `tspan`.

The argument `u0` must be a `data.frame` with one row for each node with the following columns:

S_1 The number of sucsceptible in age category 1

I_1 The number of infected in age category 1

S_2 The number of sucsceptible in age category 2

I_2 The number of infected in age category 2

S_3 The number of sucsceptible in age category 3

I_3 The number of infected in age category 3

Value

`SISe3`

Beta

The time dependent beta is divided into four intervals of the year

```
where 0 <= day < 365

Case 1: END_1 < END_2 < END_3 < END_4
INTERVAL_1 INTERVAL_2     INTERVAL_3     INTERVAL_4     INTERVAL_1
[0, END_1) [END_1, END_2) [END_2, END_3) [END_3, END_4) [END_4, 365)

Case 2: END_3 < END_4 < END_1 < END_2
INTERVAL_3 INTERVAL_4     INTERVAL_1     INTERVAL_2     INTERVAL_3
[0, END_3) [END_3, END_4) [END_4, END_1) [END_1, END_2) [END_2, 365)

Case 3: END_4 < END_1 < END_2 < END_3
INTERVAL_4 INTERVAL_1     INTERVAL_2     INTERVAL_3     INTERVAL_4
[0, END_4) [END_4, END_1) [END_1, END_2) [END_2, END_3) [END_3, 365)
```

SISe3-class *Definition of the 'SISe3' model*

Description

Class to handle the SISe3 `SimInf_model` model.

SISe3_sp *Create an* `SISe3_sp` *model*

Description

Create an `SISe3_sp` model to be used by the simulation framework.

Usage

```
SISe3_sp(
  u0,
  tspan,
  events = NULL,
  phi = NULL,
  upsilon_1 = NULL,
  upsilon_2 = NULL,
  upsilon_3 = NULL,
```

```
  gamma_1 = NULL,
  gamma_2 = NULL,
  gamma_3 = NULL,
  alpha = NULL,
  beta_t1 = NULL,
  beta_t2 = NULL,
  beta_t3 = NULL,
  beta_t4 = NULL,
  end_t1 = NULL,
  end_t2 = NULL,
  end_t3 = NULL,
  end_t4 = NULL,
  distance = NULL,
  coupling = NULL
)
```

Arguments

u0	A data.frame with the initial state in each node (see 'Details').
tspan	A vector (length >= 1) of increasing time points where the state of each node is to be returned. Can be either an integer or a Date vector. A Date vector is coerced to a numeric vector as days, where tspan[1] becomes the day of the year of the first year of tspan. The dates are added as names to the numeric vector.
events	a data.frame with the scheduled events, see SimInf_model.
phi	A numeric vector with the initial environmental infectious pressure in each node. Will be repeated to the length of nrow(u0). Default is NULL which gives 0 in each node.
upsilon_1	Indirect transmission rate of the environmental infectious pressure in age category 1
upsilon_2	Indirect transmission rate of the environmental infectious pressure in age category 2
upsilon_3	Indirect transmission rate of the environmental infectious pressure in age category 3
gamma_1	The recovery rate from infected to susceptible for age category 1
gamma_2	The recovery rate from infected to susceptible for age category 2
gamma_3	The recovery rate from infected to susceptible for age category 3
alpha	Shed rate from infected individuals
beta_t1	The decay of the environmental infectious pressure in interval 1.
beta_t2	The decay of the environmental infectious pressure in interval 2.
beta_t3	The decay of the environmental infectious pressure in interval 3.
beta_t4	The decay of the environmental infectious pressure in interval 4.
end_t1	vector with the non-inclusive day of the year that ends interval 1 in each node. Will be repeated to the length of nrow(u0).

`end_t2`	vector with the non-inclusive day of the year that ends interval 2 in each node. Will be repeated to the length of nrow(u0).
`end_t3`	vector with the non-inclusive day of the year that ends interval 3 in each node. Will be repeated to the length of nrow(u0).
`end_t4`	vector with the non-inclusive day of the year that ends interval 4 in each node. Will be repeated to the length of nrow(u0).
`distance`	The distance matrix between neighboring nodes
`coupling`	The coupling between neighboring nodes

Details

The `SISe3_sp` model contains two compartments in three age categories; number of susceptible (S_1, S_2, S_3) and number of infectious (I_1, I_2, I_3). Additionally, it contains an environmental compartment to model shedding of a pathogen to the environment. Moreover, it also includes a spatial coupling of the environmental contamination among proximal nodes to capture between-node spread unrelated to moving infected individuals. Consequently, the model has six state transitions,

$$S_1 \stackrel{\upsilon_1 \varphi S_1}{\longrightarrow} I_1$$

$$I_1 \stackrel{\gamma_1 I_1}{\longrightarrow} S_1$$

$$S_2 \stackrel{\upsilon_2 \varphi S_2}{\longrightarrow} I_2$$

$$I_2 \stackrel{\gamma_2 I_2}{\longrightarrow} S_2$$

$$S_3 \stackrel{\upsilon_3 \varphi S_3}{\longrightarrow} I_3$$

$$I_3 \stackrel{\gamma_3 I_3}{\longrightarrow} S_3$$

where the transition rate per unit of time from susceptible to infected is proportional to the concentration of the environmental contamination φ in each node. Moreover, the transition rate from infected to susceptible is the recovery rate $\gamma_1, \gamma_2, \gamma_3$, measured per individual and per unit of time. Finally, the environmental infectious pressure in each node is evolved by,

$$\frac{d\varphi_i(t)}{dt} = \frac{\alpha \left(I_{i,1}(t) + I_{i,2}(t) + I_{i,3}(t)\right)}{N_i(t)} + \sum_k \frac{\varphi_k(t) N_k(t) - \varphi_i(t) N_i(t)}{N_i(t)} \cdot \frac{D}{d_{ik}} - \beta(t)\varphi_i(t)$$

where α is the average shedding rate of the pathogen to the environment per infected individual and $N = S_1 + S_2 + S_3 + I_1 + I_2 + I_3$ the size of the node. Next comes the spatial coupling among proximal nodes, where D is the rate of the local spread and d_{ik} the distance between holdings i and k. The seasonal decay and removal of the pathogen is captured by $\beta(t)$. The environmental infectious pressure $\varphi(t)$ in each node is evolved each time unit by the Euler forward method. The value of $\varphi(t)$ is saved at the time-points specified in `tspan`.

The argument `u0` must be a `data.frame` with one row for each node with the following columns:

S_1 The number of sucsceptible in age category 1

I_1 The number of infected in age category 1

S_2 The number of sucsceptible in age category 2

I_2 The number of infected in age category 2

S_3 The number of sucsceptible in age category 3

I_3 The number of infected in age category 3

Value

`SISe3_sp`

Beta

The time dependent beta is divided into four intervals of the year

```
where 0 <= day < 365

Case 1: END_1 < END_2 < END_3 < END_4
INTERVAL_1 INTERVAL_2     INTERVAL_3     INTERVAL_4     INTERVAL_1
[0, END_1) [END_1, END_2) [END_2, END_3) [END_3, END_4) [END_4, 365)

Case 2: END_3 < END_4 < END_1 < END_2
INTERVAL_3 INTERVAL_4     INTERVAL_1     INTERVAL_2     INTERVAL_3
[0, END_3) [END_3, END_4) [END_4, END_1) [END_1, END_2) [END_2, 365)

Case 3: END_4 < END_1 < END_2 < END_3
INTERVAL_4 INTERVAL_1     INTERVAL_2     INTERVAL_3     INTERVAL_4
[0, END_4) [END_4, END_1) [END_1, END_2) [END_2, END_3) [END_3, 365)
```

`SISe3_sp-class` *Definition of the 'SISe3_sp' model*

Description

Class to handle the SISe3_sp `SimInf_model` model.

SISe_sp *Create a SISe_sp model*

Description

Create a SISe_sp model to be used by the simulation framework.

Usage

```
SISe_sp(
  u0,
  tspan,
  events = NULL,
  phi = NULL,
  upsilon = NULL,
  gamma = NULL,
  alpha = NULL,
  beta_t1 = NULL,
  beta_t2 = NULL,
  beta_t3 = NULL,
  beta_t4 = NULL,
  end_t1 = NULL,
  end_t2 = NULL,
  end_t3 = NULL,
  end_t4 = NULL,
  coupling = NULL,
  distance = NULL
)
```

Arguments

u0	A data.frame with the initial state in each node (see 'Details').
tspan	A vector (length >= 1) of increasing time points where the state of each node is to be returned. Can be either an integer or a Date vector. A Date vector is coerced to a numeric vector as days, where tspan[1] becomes the day of the year of the first year of tspan. The dates are added as names to the numeric vector.
events	a data.frame with the scheduled events, see SimInf_model.
phi	A numeric vector with the initial environmental infectious pressure in each node. Will be repeated to the length of nrow(u0). Default is NULL which gives 0 in each node.
upsilon	Indirect transmission rate of the environmental infectious pressure
gamma	The recovery rate from infected to susceptible
alpha	Shed rate from infected individuals
beta_t1	The decay of the environmental infectious pressure in interval 1.

beta_t2	The decay of the environmental infectious pressure in interval 2.
beta_t3	The decay of the environmental infectious pressure in interval 3.
beta_t4	The decay of the environmental infectious pressure in interval 4.
end_t1	vector with the non-inclusive day of the year that ends interval 1 in each node. Will be repeated to the length of nrow(u0).
end_t2	vector with the non-inclusive day of the year that ends interval 2 in each node. Will be repeated to the length of nrow(u0).
end_t3	vector with the non-inclusive day of the year that ends interval 3 in each node. Will be repeated to the length of nrow(u0).
end_t4	vector with the non-inclusive day of the year that ends interval 4 in each node. Will be repeated to the length of nrow(u0).
coupling	The coupling between neighboring nodes
distance	The distance matrix between neighboring nodes

Details

The SISe_sp model contains two compartments; number of susceptible (S) and number of infectious (I). Additionally, it contains an environmental compartment to model shedding of a pathogen to the environment. Moreover, it also includes a spatial coupling of the environmental contamination among proximal nodes to capture between-node spread unrelated to moving infected individuals. Consequently, the model has two state transitions,

$$S \xrightarrow{\upsilon \varphi S} I$$

$$I \xrightarrow{\gamma I} S$$

where the transition rate per unit of time from susceptible to infected is proportional to the concentration of the environmental contamination φ in each node. Moreover, the transition rate from infected to susceptible is the recovery rate γ, measured per individual and per unit of time. Finally, the environmental infectious pressure in each node is evolved by,

$$\frac{d\varphi_i(t)}{dt} = \frac{\alpha I_i(t)}{N_i(t)} + \sum_k \frac{\varphi_k(t)N_k(t) - \varphi_i(t)N_i(t)}{N_i(t)} \cdot \frac{D}{d_{ik}} - \beta(t)\varphi_i(t)$$

where α is the average shedding rate of the pathogen to the environment per infected individual and $N = S + I$ the size of the node. Next comes the spatial coupling among proximal nodes, where D is the rate of the local spread and d_{ik} the distance between holdings i and k. The seasonal decay and removal of the pathogen is captured by $\beta(t)$. The environmental infectious pressure $\varphi(t)$ in each node is evolved each time unit by the Euler forward method. The value of $\varphi(t)$ is saved at the time-points specified in tspan.

The argument u0 must be a data.frame with one row for each node with the following columns:

S The number of sucsceptible

I The number of infected

Value

SISe_sp

Beta

The time dependent beta is divided into four intervals of the year

```
where 0 <= day < 365

Case 1: END_1 < END_2 < END_3 < END_4
INTERVAL_1 INTERVAL_2     INTERVAL_3     INTERVAL_4     INTERVAL_1
[0, END_1) [END_1, END_2) [END_2, END_3) [END_3, END_4) [END_4, 365)

Case 2: END_3 < END_4 < END_1 < END_2
INTERVAL_3 INTERVAL_4     INTERVAL_1     INTERVAL_2     INTERVAL_3
[0, END_3) [END_3, END_4) [END_4, END_1) [END_1, END_2) [END_2, 365)

Case 3: END_4 < END_1 < END_2 < END_3
INTERVAL_4 INTERVAL_1     INTERVAL_2     INTERVAL_3     INTERVAL_4
[0, END_4) [END_4, END_1) [END_1, END_2) [END_2, END_3) [END_3, 365)
```

SISe_sp-class — *Definition of the* SISe_sp *model*

Description

Class to handle the SISe_sp SimInf_model.

---

summary,SimInf_events-method — *Detailed summary of a* SimInf_events *object*

Description

Shows the number of scheduled events and the number of scheduled events per event type.

Usage

```
## S4 method for signature 'SimInf_events'
summary(object, ...)
```

Arguments

object	The SimInf_events object
...	Additional arguments affecting the summary produced.

Value

None (invisible 'NULL').

`summary,SimInf_model-method`
Detailed summary of a `SimInf_model` *object*

Description

Detailed summary of a `SimInf_model` object

Usage

```
## S4 method for signature 'SimInf_model'
summary(object, ...)
```

Arguments

`object`	The `SimInf_model` object
`...`	Additional arguments affecting the summary produced.

Value

None (invisible 'NULL').

`trajectory` *Extract data from a simulated trajectory*

Description

Extract the number of individuals in each compartment in every node after generating a single stochastic trajectory with `run`.

Usage

```
trajectory(model, compartments = NULL, node = NULL, as.is = FALSE)
```

Arguments

- `model` — the `model` to extract the result from.
- `compartments` — specify the names of the compartments to extract data from. The compartments can be specified as a character vector e.g. `compartments = c('S','I','R')`, or as a formula e.g. `compartments = ~S+I+R` (see 'Examples'). Default (`compartments=NULL`) is to extract the number of individuals in each compartment i.e. the data from all discrete state compartments in the model. In models that also have continuous state variables e.g. the `SISe` model, use `~.` instead of `NULL` to also include these.
- `node` — indices specifying the subset of nodes to include when extracting data. Default (`node = NULL`) is to extract data from all nodes.
- `as.is` — the default (`as.is = FALSE`) is to generate a `data.frame` with one row per node and time-step with the number of individuals in each compartment. Using `as.is = TRUE` returns the result as a matrix, which is the internal format (see 'Details').

Value

A `data.frame` if `as.is = FALSE`, else a matrix.

Internal format of the discrete state variables

Description of the layout of the internal matrix (`U`) that is returned if `as.is = TRUE`. `U[,j]` contains the number of individuals in each compartment at `tspan[j]`. `U[1:Nc,j]` contains the number of individuals in node 1 at `tspan[j]`. `U[(Nc + 1):(2 * Nc),j]` contains the number of individuals in node 2 at `tspan[j]` etc, where `Nc` is the number of compartments in the model. The dimension of the matrix is $N_n N_c \times$ `length(tspan)` where N_n is the number of nodes.

Internal format of the continuous state variables

Description of the layout of the matrix that is returned if `as.is = TRUE`. The result matrix for the real-valued continuous state. `V[,j]` contains the real-valued state of the system at `tspan[j]`. The dimension of the matrix is N_n`dim(ldata)[1]` $\times$ `length(tspan)`.

Examples

```
## Create an 'SIR' model with 6 nodes and initialize
## it to run over 10 days.
u0 <- data.frame(S = 100:105, I = 1:6, R = rep(0, 6))
model <- SIR(u0 = u0, tspan = 1:10, beta = 0.16, gamma = 0.077)

## Run the model to generate a single stochastic trajectory.
result <- run(model)

## Extract the number of individuals in each compartment at the
## time-points in 'tspan'.
trajectory(result)

## Extract the number of recovered individuals in the first node
## at the time-points in 'tspan'.
trajectory(result, compartments = "R", node = 1)
```

```
## Extract the number of recovered individuals in the first and
## third node at the time-points in 'tspan'.
trajectory(result, compartments = "R", node = c(1, 3))

## Create an 'SISe' model with 6 nodes and initialize
## it to run over 10 days.
u0 <- data.frame(S = 100:105, I = 1:6)
model <- SISe(u0 = u0, tspan = 1:10, phi = rep(0, 6),
    upsilon = 0.02, gamma = 0.1, alpha = 1, epsilon = 1.1e-5,
    beta_t1 = 0.15, beta_t2 = 0.15, beta_t3 = 0.15, beta_t4 = 0.15,
    end_t1 = 91, end_t2 = 182, end_t3 = 273, end_t4 = 365)

## Run the model
result <- run(model)

## Extract the continuous state variable 'phi' which represents
## the environmental infectious pressure.
trajectory(result, "phi")
```

---

u0_SEIR *Example data to initialize the 'SEIR' model*

Description

Example data to initialize a population of 1600 nodes and demonstrate the SEIR model.

Usage

```
u0_SEIR()
```

Details

A `data.frame` with the number of individuals in the 'S', 'E', 'I' and 'R' compartments in 1600 nodes. Note that the 'E', 'I' and 'R' compartments are zero.

Value

A `data.frame`

Examples

```
## Create an 'SEIR' model with 1600 nodes and initialize it to
## run over 4*365 days and record data at weekly time-points.
## Add ten infected individuals to the first node.
u0 <- u0_SEIR()
u0$I[1] <- 10
tspan <- seq(from = 1, to = 4*365, by = 7)
model <- SEIR(u0      = u0,
              tspan   = tspan,
```

```
                events  = events_SEIR(),
                beta    = 0.16,
                epsilon = 0.25,
                gamma   = 0.01)

## Run the model to generate a single stochastic trajectory.
result <- run(model)
plot(result)

## Summarize trajectory
summary(result)
```

---

u0_SIR *Example data to initialize the 'SIR' model*

Description

Example data to initialize a population of 1600 nodes and demonstrate the SIR model.

Usage

```
u0_SIR()
```

Details

A data.frame with the number of individuals in the 'S', 'I' and 'R' compartments in 1600 nodes. Note that the 'I' and 'R' compartments are zero.

Value

A data.frame

Examples

```
## Create an 'SIR' model with 1600 nodes and initialize
## it to run over 4*365 days. Add one infected individual
## to the first node.
u0 <- u0_SIR()
u0$I[1] <- 1
tspan <- seq(from = 1, to = 4*365, by = 1)
model <- SIR(u0     = u0,
             tspan  = tspan,
             events = events_SIR(),
             beta   = 0.16,
             gamma  = 0.01)

## Run the model to generate a single stochastic trajectory.
result <- run(model)
plot(result)
```

```
## Summarize trajectory
summary(result)
```

u0_SISe *Example data to initialize the 'SISe' model*

Description

Example data to initialize a population of 1600 nodes and demonstrate the SISe model.

Usage

```
u0_SISe()
```

Details

A data.frame with the number of individuals in the 'S' and 'I' compartments in 1600 nodes. Note that the 'I' compartment is zero.

Value

A data.frame

Examples

```
## Create an 'SISe' model with 1600 nodes and initialize it to
## run over 4*365 days and record data at weekly time-points.

## Load the initial population and add ten infected individuals to
## the first node.
u0 <- u0_SISe()
u0$I[1] <- 10

## Define 'tspan' to run the simulation over 4*365 and record the
## state of the system at weekly time-points.
tspan <- seq(from = 1, to = 4*365, by = 7)

## Load scheduled events for the population of nodes with births,
## deaths and between-node movements of individuals.
events <- events_SISe()

## Create an 'SISe' model
model <- SISe(u0 = u0, tspan = tspan, events = events_SISe(),
              phi = 0, upsilon = 1.8e-2, gamma = 0.1, alpha = 1,
              beta_t1 = 1.0e-1, beta_t2 = 1.0e-1, beta_t3 = 1.25e-1,
              beta_t4 = 1.25e-1, end_t1 = 91, end_t2 = 182,
              end_t3 = 273, end_t4 = 365, epsilon = 0)

## Run the model to generate a single stochastic trajectory.
result <- run(model)
```

```
## Summarize trajectory
summary(result)

## Plot the proportion of nodes with at least one infected
## individual.
plot(prevalence(result, I~S+I, "nop"), type = "l")
```

---

u0_SISe3 *Example data to initialize the 'SISe3' model*

Description

Example data to initialize a population of 1600 nodes and demonstrate the SISe3 model.

Usage

```
data(u0_SISe3)
```

Format

A data.frame

Details

A data.frame with the number of individuals in the 'S_1', 'S_2', 'S_3', 'I_1', 'I_2' and 'I_3' compartments in 1600 nodes. Note that the 'I_1', 'I_2' and 'I_3' compartments are zero.

Examples

```
## Create an 'SISe3' model with 1600 nodes and initialize it to
## run over 4*365 days and record data at weekly time-points.

## Load the initial population and add ten infected individuals to
## I_1 in the first node.
u0 <- u0_SISe3
u0$I_1[1] <- 10

## Define 'tspan' to run the simulation over 4*365 and record the
## state of the system at weekly time-points.
tspan <- seq(from = 1, to = 4*365, by = 7)

## Load scheduled events for the population of nodes with births,
## deaths and between-node movements of individuals.
events <- events_SISe3

## Create a 'SISe3' model
model <- SISe3(u0 = u0, tspan = tspan, events = events,
               phi = rep(0, nrow(u0)), upsilon_1 = 1.8e-2,
               upsilon_2 = 1.8e-2, upsilon_3 = 1.8e-2,
```

```
                 gamma_1 = 0.1, gamma_2 = 0.1, gamma_3 = 0.1,
                 alpha = 1, beta_t1 = 1.0e-1, beta_t2 = 1.0e-1,
                 beta_t3 = 1.25e-1, beta_t4 = 1.25e-1, end_t1 = 91,
                 end_t2 = 182, end_t3 = 273, end_t4 = 365, epsilon = 0)

## Run the model to generate a single stochastic trajectory.
result <- run(model)

## Summarize trajectory
summary(result)

## Plot the proportion of nodes with at least one infected
## individual.
plot(prevalence(result, I_1 + I_2 + I_3 ~ ., "nop"), type = "l")
```

Index