Package ‘nosoi’

April 3, 2020

Type Package
Title A Forward Agent-Based Transmission Chain Simulator
Version 1.0.2
Description The aim of ‘nosoi’ (pronounced no.si) is to provide a flexible agent-based stochastic transmission chain/epidemic simulator (Lequime et al. bioRxiv 2020.03.03.973107). It is named after the daimones of plague, sickness and disease that escaped Pandora’s jar in the Greek mythology. ‘nosoi’ is able to take into account the influence of multiple variable on the transmission process (e.g. dual-host systems (such as arboviruses), within-host viral dynamics, transportation, population structure), alone or taken together, to create complex but relatively intuitive epidemiological simulations.

URL https://github.com/slequime/nosoi
Language en-US
License GPL-3
Encoding UTF-8
LazyData true
RoxygenNote 7.0.2

biocViews

Depends data.table (>= 1.12.0), R (>= 3.5.0)
Imports stats (>= 3.5.2), methods (>= 3.5.2), dplyr (>= 0.8.0), magrittr (>= 1.5), stringr (>= 1.4.0), reshape2 (>= 1.4.0), raster (>= 2.8-19)
Suggests testthat (>= 2.1.0), knitr, rmarkdown, igraph, tidyverse, ggplot2, ggnetwork, intergraph, viridis, gifski, png, gganimate, ape (>= 5.3), tidytree (>= 0.3.3), treeio (>= 1.10.0), ggtree (>= 2.0.2)

VignetteBuilder knitr
NeedsCompilation no

Author Sebastian Lequime [aut, cre] (<https://orcid.org/0000-0002-3140-0651>), Paul Bastide [aut] (<https://orcid.org/0000-0002-8084-9893>), Simon Dellicour [aut] (<https://orcid.org/0000-0001-9558-1052>), Philippe Lemey [aut] (<https://orcid.org/0000-0003-2826-5353>), Guy Baele [aut] (<https://orcid.org/0000-0002-1915-7732>)
**dualContinuous**

**Maintainer**  Sebastian Lequime <sebastian.lequime@gmail.com>

**Repository**  CRAN

**Date/Publication**  2020-04-03 14:10:19 UTC

**R topics documented:**

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>dualContinuous</td>
<td>2</td>
</tr>
<tr>
<td>dualDiscrete</td>
<td>10</td>
</tr>
<tr>
<td>dualNone</td>
<td>16</td>
</tr>
<tr>
<td>getCumulative</td>
<td>20</td>
</tr>
<tr>
<td>getDynamic</td>
<td>21</td>
</tr>
<tr>
<td>getHostData</td>
<td>21</td>
</tr>
<tr>
<td>getR0</td>
<td>23</td>
</tr>
<tr>
<td>getTableHosts</td>
<td>23</td>
</tr>
<tr>
<td>getTableState</td>
<td>24</td>
</tr>
<tr>
<td>getTransmissionTree</td>
<td>25</td>
</tr>
<tr>
<td>nosoiSim</td>
<td>27</td>
</tr>
<tr>
<td>nosoiSummary</td>
<td>28</td>
</tr>
<tr>
<td>sampleTransmissionTree</td>
<td>30</td>
</tr>
<tr>
<td>sampleTransmissionTreeFromExiting</td>
<td>32</td>
</tr>
<tr>
<td>singleContinuous</td>
<td>34</td>
</tr>
<tr>
<td>singleDiscrete</td>
<td>38</td>
</tr>
<tr>
<td>singleNone</td>
<td>41</td>
</tr>
</tbody>
</table>

**Index**  44

---

**dualContinuous**  *Dual-host pathogen in structured (continuous) hosts populations*

**Description**

This function runs a dual-host transmission chain simulation, with structured hosts populations (such as spatial features) in a shared continuous space. The simulation stops either at the end of given time (specified by *length.sim*) or when the number of hosts infected threshold (*max.infected*) is passed. The movement of hosts on the continuous space map is a random walk (Brownian motion) that can be modified towards a biased random walk where hosts tend to be attracted to higher values of the environmental variable defined by the raster.

**Usage**

```r

dualContinuous(
  length.sim,
  max.infected.A,
  max.infected.B,
  init.individuals.A,
  init.individuals.B,
```
init.structure.A,
init.structure.B,
structure.raster.A,
structure.raster.B,
pExit.A,
param.pExit.A,
timeDep.pExit.A = FALSE,
diff.pExit.A = FALSE,
hostCount.pExit.A = FALSE,
pMove.A,
param.pMove.A,
timeDep.pMove.A = FALSE,
diff.pMove.A = FALSE,
hostCount.pMove.A = FALSE,
sdMove.A,
param.sdMove.A,
diff.sdMove.A = FALSE,
timeDep.sdMove.A = FALSE,
hostCount.sdMove.A = FALSE,
attracted.by.raster.A = FALSE,
nContact.A,
param.nContact.A,
timeDep.nContact.A = FALSE,
diff.nContact.A = FALSE,
hostCount.nContact.A = FALSE,
pTrans.A,
param.pTrans.A,
timeDep.pTrans.A = FALSE,
diff.pTrans.A = FALSE,
hostCount.pTrans.A = FALSE,
prefix.host.A = "H",
pExit.B,
param.pExit.B,
timeDep.pExit.B = FALSE,
diff.pExit.B = FALSE,
hostCount.pExit.B = FALSE,
pMove.B,
param.pMove.B,
timeDep.pMove.B = FALSE,
diff.pMove.B = FALSE,
hostCount.pMove.B = FALSE,
hostCount.sdMove.B = FALSE,
applied.by.raster.B = FALSE,
nContact.B,
param.nContact.B,
timeDep.nContact.B = FALSE,
diff.nContact.B = FALSE,
hostCount.nContact.B = FALSE,
pTrans.B,
param.pTrans.B,
timeDep.pTrans.B = FALSE,
diff.pTrans.B = FALSE,
hostCount.pTrans.B = FALSE,
prefix.host.B = "V",
print.progress = TRUE,
print.step = 10
)

Arguments

length.sim specifies the length (in unit of time) over which the simulation should be run.

max.infected.A specifies the maximum number of individual hosts A that can be infected in the simulation.

max.infected.B specifies the maximum number of individual hosts B that can be infected in the simulation.

init.individuals.A number of initially infected individuals (hosts A).

init.individuals.B number of initially infected individuals (hosts B).

init.structure.A in which location the initially infected host-A individuals are located. A vector of coordinates in the same coordinate space as the raster (NA if init.individual.A is 0).

init.structure.B in which location the initially infected host-B individuals are located. A vector of coordinates in the same coordinate space as the raster (NA if init.individual.B is 0).

structure.raster.A raster object defining the environmental variable for host-type A.

structure.raster.B raster object defining the environmental variable for host B.

pExit.A function that gives the probability to exit the simulation for an infected host A (either moving out, dying, etc.).

param.pExit.A parameter names (list of functions) for the pExit for host-type A.

timeDep.pExit.A is pExit of host-type A dependent on the absolute time of the simulation (TRUE/FALSE)?

diff.pExit.A does pExit of host-type A depend on the environmental variable (set by the raster) (TRUE/FALSE).

hostCount.pExit.A does pExit of host-type A vary with the host count (of either host-type A or B) in each raster cell? (TRUE/FALSE); if TRUE, diff.pExit.A should be TRUE.
**dualContinuous**

- **pMove.A** function that gives the probability of a host moving as a function of time for host-type A.
- **param.pMove.A** parameter names (list of functions) for the pMove for host-type A.
- **timeDep.pMove.A** is pMove of host-type A dependent on the absolute time of the simulation (TRUE/FALSE)?
- **diff.pMove.A** does pMove of host-type A depend on the environmental variable (set by the raster) (TRUE/FALSE).A.
- **hostCount.pMove.A** does pMove of host-type A vary with the host count (of either host-type A or B) in each raster cell? (TRUE/FALSE); if TRUE, diff.pMove.A should be TRUE.
- **sdMove.A** function that gives the distance traveled for host-type A (based on coordinates); output is the standard deviation value for the Brownian motion.
- **param.sdMove.A** parameter names (list of functions) for sdMove for host-type A.
- **diff.sdMove.A** does sdMove of host-type A depend on the environmental variable (set by the raster) (TRUE/FALSE).
- **timeDep.sdMove.A** is sdMove of host-type A dependent on the absolute time of the simulation (TRUE/FALSE)?
- **hostCount.sdMove.A** does sdMove vary with the host count (of either host-type A or B) in each raster cell? (TRUE/FALSE); diff.sdMove.A should be TRUE.
- **attracted.by.raster.A** should the host-type A be attracted by higher values in the environmental raster? (TRUE/FALSE).
- **nContact.A** function that gives the number of potential transmission events per unit of time for host-type A.
- **param.nContact.A** parameter names (list of functions) for param.nContact for host-type A.
- **timeDep.nContact.A** is nContact of host-type A dependent on the absolute time of the simulation (TRUE/FALSE)?
- **diff.nContact.A** does nContact of host-type A depend on the environmental variable (set by the raster) (TRUE/FALSE).
- **hostCount.nContact.A** does nContact vary with the host count (of either host-type A or B) in each raster cell? (TRUE/FALSE); diff.nContact.A should be TRUE.
- **pTrans.A** function that gives the probability of transmit a pathogen as a function of time since infection for host A.
- **param.pTrans.A** parameter names (list of functions) for the pExit for host A.
- **timeDep.pTrans.A** is pTrans of host-type A dependent on the absolute time of the simulation (TRUE/FALSE)?
- **diff.pTrans.A** does pTrans of host-type A depend on the environmental variable (set by the raster) (TRUE/FALSE).
hostCount.pTrans.A does pTrans vary with the host count (of either host-type A or B) in each raster cell? (TRUE/FALSE); diff.pTrans.A should be TRUE.

prefix.host.A character(s) to be used as a prefix for the host A identification number.

pExit.B function that gives the probability to exit the simulation for an infected host B (either moving out, dying, etc.).

param.pExit.B parameter names (list of functions) for the pExit for host-type B.

timeDep.pExit.B is pExit of host-type B dependent on the absolute time of the simulation (TRUE/FALSE)?

diff.pExit.B does pExit of host-type B depend on the environmental variable (set by the raster) (TRUE/FALSE).

hostCount.pExit.B does pExit of host-type B vary with the host count (of either host-type A or B) in each raster cell? (TRUE/FALSE); if TRUE, diff.pExit.B should be TRUE.

pMove.B function that gives the probability of a host moving as a function of time for host-type B.

param.pMove.B parameter names (list of functions) for the pMove for host-type B.

timeDep.pMove.B is sdMove of host-type B dependent on the absolute time of the simulation (TRUE/FALSE) for host-type B.

diff.pMove.B does pMove of host-type B depend on the environmental variable (set by the raster) (TRUE/FALSE).

hostCount.pMove.B does pMove of host-type B vary with the host count (of either host-type A or B) in each raster cell? (TRUE/FALSE); if TRUE, diff.pMove.B should be TRUE.

sdMove.B function that gives the distance traveled for host-type B (based on coordinates); output is the standard deviation value for the Brownian motion.

param.sdMove.B parameter names (list of functions) for sdMove for host-type B.

diff.sdMove.B does sdMove of host-type B depend on the environmental variable (set by the raster) (TRUE/FALSE).

timeDep.sdMove.B is sdMove of host-type B dependent on the absolute time of the simulation (TRUE/FALSE)?

hostCount.sdMove.B does sdMove of host-type B vary with the host count (of either host-type A or B) in each raster cell? (TRUE/FALSE); if TRUE, diff.sdMove.B should be TRUE.

attracted.by.raster.B should the host-type B be attracted by higher values in the environmental raster? (TRUE/FALSE)

nContact.B function that gives the number of potential transmission events per unit of time for host B.

param.nContact.B parameter names (list of functions) for param.nContact for host-type B.
timeDep.nContact.B

is nContact of host-type B dependent on the absolute time of the simulation (TRUE/FALSE)?

diff.nContact.B

does nContact of host-type B depend on the environmental variable (set by the raster) (TRUE/FALSE).

hostCount.nContact.B

does nContact of host-type B vary with the host count (of either host-type A or B) in each raster cell? (TRUE/FALSE); if TRUE, diff.nContact.B should be TRUE.

pTrans.B

function that gives the probability of transmit a pathogen as a function of time since infection for host B.

param.pTrans.B

parameter names (list of functions) for the pExit for host-type B.

timeDep.pTrans.B

is pTrans of host-type B dependent on the absolute time of the simulation (TRUE/FALSE)?

diff.pTrans.B

does pTrans of host-type B depend on the environmental variable (set by the raster) (TRUE/FALSE).

hostCount.pTrans.B

does pTrans of host-type B vary with the host count (of either host-type A or B) in each raster cell? (TRUE/FALSE); if TRUE, diff.pTrans.B should be TRUE.

prefix.host.B

character(s) to be used as a prefix for the host B identification number.

print.progress

if TRUE, displays a progress bar (current time/length.sim).

print.step

print.progress is TRUE, step with which the progress message will be printed.

Details

The pExit and pTrans functions should return a single probability (a number between 0 and 1), and nContact a positive natural number (positive integer) or 0.

The param arguments should be a list of functions or NA. Each item name in the parameter list should have the same name as the argument in the corresponding function.

The use of timeDep (switch to TRUE) makes the corresponding function use the argument prestime (for "present time").

Value

An object of class nosoiSim, containing all results of the simulation.

Raster

The structure raster(s) provided provided should of class raster. High values of the environmental variable can attract hosts if attracted.by.raster is TRUE. Raster have to share the same space (i.e. also the same cell size and ID).

Order of Arguments

The user specified function’s arguments should follow this order: t (mandatory), prestime (optional, only if timeDep is TRUE), current.env.value (optional, only if diff is TRUE), host.count.A or host.count.B (optional, only if hostCount is TRUE) and parameters specified in the list.
**Structure Parameters**

The `pMove` function should return a single probability (a number between 0 and 1), and `sdMove` a real number (keep in mind this number is related to your coordinate space).

The use of `diff` (switch to `TRUE`) makes the corresponding function use the argument `current.env.value` (for "current environmental value").

The use of `hostCount` (switch to `TRUE`) makes the corresponding function use the argument `host.count`.

**Suffixes**

The suffix `.A` or `.B` specifies if the considered function or parameter concerns host type A or B.

**See Also**

For simulations with a discrete structure, see `dualDiscrete`. For simulations without any structures, see `dualNone`.

**Examples**

```r
library(raster)

#Generating a raster the for movement
set.seed(860)

test.raster <- raster(nrows=100, ncols=100, xmn=-50, xmx=50, ymn=-50, ymx=50)
test.raster[] <- runif(10000, -80, 180)
test.raster <- focal(focal(test.raster, w=matrix(1, 5, 5), mean), w=matrix(1, 5, 5), mean)

t_incub_fct <- function(x){rnorm(x,mean = 5,sd=1)}
p_max_fct <- function(x){rbeta(x,shape1 = 5,shape2=2)}
p_Move_fct <- function(t){return(0.1)}

sdMove_fct = function(t,current.env.value){return(100/(current.env.value+1))}

p_Exit_fct <- function(t){return(0.08)}

proba <- function(t,p_max,t_incub){
  if(t <= t_incub){p=0}
  if(t >= t_incub){p=p_max}
  return(p)
}

time_contact = function(t){round(rnorm(1, 3, 1), 0)}

start.pos <- c(0,0)
set.seed(805)
test.nosoi <- nosoiSim(type="dual", popStructure="continuous",
  length.sim=200,
  max.infected.A=500,
  max.infected.B=500,
  max.infected.B=500,
)
```

---

*dualContinuous*
init.individuals.A=1,
init.individuals.B=0,
init.structure.A=start.pos,
init.structure.B=NA,
structure.raster.A=test.raster,
structure.raster.B=test.raster,
pExit.A=p_Exit_fct,
param.pExit.A=NA,
timeDep.pExit.A=FALSE,
diff.pExit.A=FALSE,
pMove.A=p_Move_fct,
param.pMove.A=NA,
timeDep.pMove.A=FALSE,
diff.pMove.A=FALSE,
diff.sdMove.A=TRUE,
sdMove.A=sdMove_fct,
param.sdMove.A=NA,
attracted.by.raster.A=TRUE,
nContact.A=time_contact,
param.nContact.A=NA,
timeDep.nContact.A=FALSE,
diff.nContact.A=FALSE,
pTrans.A=proba,
param.pTrans.A=list(p_max=p_max_fct,
t_incub=t_incub_fct),
timeDep.pTrans.A=FALSE,
diff.pTrans.A=FALSE,
prefix.host.A="H",
pExit.B=p_Exit_fct,
param.pExit.B=NA,
timeDep.pExit.B=FALSE,
diff.pExit.B=FALSE,
pMove.B=p_Move_fct,
param.pMove.B=NA,
timeDep.pMove.B=FALSE,
diff.pMove.B=FALSE,
diff.sdMove.B=TRUE,
sdMove.B=sdMove_fct,
param.sdMove.B=NA,
attracted.by.raster.B=TRUE,
nContact.B=time_contact,
param.nContact.B=NA,
timeDep.nContact.B=FALSE,
diff.nContact.B=FALSE,
pTrans.B=proba,
param.pTrans.B=list(p_max=p_max_fct,
t_incub=t_incub_fct),
timeDep.pTrans.B=FALSE,
diff.pTrans.B=FALSE,
prefix.host.B="V")
dualDiscrete

Dual-host pathogen in structured (discrete) hosts populations

Description

This function, that can be wrapped within `nosoiSim`, runs a dual-host transmission chain simulation, with discrete hosts populations structures (e.g. spatial, socio-economic, etc.). The simulation stops either at the end of given time (specified by `length.sim`) or when the number of hosts infected threshold (`max.infected`) is crossed.

Usage

dualDiscrete(
  length.sim,
  max.infected.A,
  max.infected.B,
  init.individuals.A,
  init.individuals.B,
  init.structure.A,
  init.structure.B,
  structure.matrix.A,
  structure.matrix.B,
  pExit.A,
  param.pExit.A,
  timeDep.pExit.A = FALSE,
  diff.pExit.A = FALSE,
  hostCount.pExit.A = FALSE,
  pMove.A,
  param.pMove.A,
  timeDep.pMove.A = FALSE,
  diff.pMove.A = FALSE,
  hostCount.pMove.A = FALSE,
  nContact.A,
  param.nContact.A,
  timeDep.nContact.A = FALSE,
  diff.nContact.A = FALSE,
  hostCount.nContact.A = FALSE,
  pTrans.A,
  param.pTrans.A,
  timeDep.pTrans.A = FALSE,
  diff.pTrans.A = FALSE,
  hostCount.pTrans.A = FALSE,
  prefix.host.A = "H",
  pExit.B,
  param.pExit.B,
  timeDep.pExit.B = FALSE,
  diff.pExit.B = FALSE,
  ...
dualDiscrete

```r
dualDiscrete

  hostCount.pExit.B = FALSE,
  pMove.B,
  param.pMove.B,
  timeDep.pMove.B = FALSE,
  diff.pMove.B = FALSE,
  hostCount.pMove.B = FALSE,
  nContact.B,
  param.nContact.B,
  timeDep.nContact.B = FALSE,
  diff.nContact.B = FALSE,
  hostCount.nContact.B = FALSE,
  pTrans.B,
  param.pTrans.B,
  timeDep.pTrans.B = FALSE,
  diff.pTrans.B = FALSE,
  hostCount.pTrans.B = FALSE,
  prefix.host.B = "V",
  print.progress = TRUE,
  print.step = 10
```

**Arguments**

- `length.sim` specifies the length (in unit of time) over which the simulation should be run.
- `max.infected.A` specifies the maximum number of individual hosts A that can be infected in the simulation.
- `max.infected.B` specifies the maximum number of individual hosts B that can be infected in the simulation.
- `init.individuals.A` number of initially infected individuals (hosts A).
- `init.individuals.B` number of initially infected individuals (hosts B).
- `init.structure.A` in which state (e.g. location) the initially infected individuals of host-type A are located (NA if init.individual.A is 0)?
- `init.structure.B` in which state (e.g. location) the initially infected individuals of host-type B are located (NA if init.individual.B is 0)?
- `structure.matrix.A` transition matrix (probabilities) to go from location A (row) to B (column) for host-type A.
- `structure.matrix.B` transition matrix (probabilities) to go from location A (row) to B (column) for host-type B.
- `pExit.A` function that gives the probability to exit the simulation for an infected host A (either moving out, dying, etc.).
- `param.pExit.A` parameter names (list of functions) for the pExit for host-type A.
timeDep.pExit.A is pExit of host-type A dependent on the absolute time of the simulation (TRUE/FALSE)?
diff.pExit.A is pExit of host-type A different between states of the structured population (TRUE/FALSE)?
hostCount.pExit.A does pExit of host-type A vary with the host count (of either host-type A or B) in the state? (TRUE/FALSE); diff.pExit.A should be TRUE.
pMove.A function that gives the probability of a host moving as a function of time for host-type A.
param.pMove.A parameter names (list of functions) for the pMove for host-type A.
timeDep.pMove.A is pMove of host-type A dependent on the absolute time of the simulation (TRUE/FALSE)?
diff.pMove.A is pMove of host-type A different between states of the structured population (TRUE/FALSE)?
hostCount.pMove.A does pMove of host-type A vary with the host count (of either host A or B) in the state? (TRUE/FALSE); diff.pMove.A should be TRUE.
nContact.A function that gives the number of potential transmission events per unit of time for host-type A.
param.nContact.A parameter names (list of functions) for param.nContact for host-type A.
timeDep.nContact.A is nContact of host-type A dependent on the absolute time of the simulation (TRUE/FALSE)?
diff.nContact.A is nContact of host-type A different between states of the structured population (TRUE/FALSE)?
hostCount.nContact.A does nContact of host-type A vary with the host count (of either host A or B) in the state? (TRUE/FALSE); diff.nContact.A should be TRUE.
pTrans.A function that gives the probability of transmit a pathogen as a function of time since infection for host A.
param.pTrans.A parameter names (list of functions) for the pExit for host A.
timeDep.pTrans.A is pTrans of host-type A dependent on the absolute time of the simulation (TRUE/FALSE)?
diff.pTrans.A is pTrans of host-type A different between states of the structured population (TRUE/FALSE)?
hostCount.pTrans.A does pTrans of host-type A vary with the host count (of either host A or B) in the state? (TRUE/FALSE); diff.pTrans.A should be TRUE.

prefix.host.A character(s) to be used as a prefix for the host A identification number.
pExit.B function that gives the probability to exit the simulation for an infected host B (either moving out, dying, etc.).
param.pExit.B parameter names (list of functions) for the pExit for host-type B.
**dualDiscrete**

- **timeDep.pExit.B**
  - is pExit of host-type B dependent on the absolute time of the simulation (TRUE/FALSE)?

- **diff.pExit.B**
  - is pExit of host-type B different between states of the structured population (TRUE/FALSE)?

- **hostCount.pExit.B**
  - does pExit of host-type B vary with the host count (of either host A or B) in the state? (TRUE/FALSE); diff.pExit.B should be TRUE.

- **pMove.B**
  - function that gives the probability of a host moving as a function of time for host-type B.

- **param.pMove.B**
  - parameter names (list of functions) for the pMove for host-type B.

- **timeDep.pMove.B**
  - is sdMove of host-type B dependent on the absolute time of the simulation (TRUE/FALSE) for host-type B.

- **diff.pMove.B**
  - is pMove of host-type B different between states of the structured population (TRUE/FALSE)?

- **hostCount.pMove.B**
  - does pMove of host-type B vary with the host count (of either host A or B) in the state? (TRUE/FALSE); diff.pMove.B should be TRUE.

- **nContact.B**
  - function that gives the number of potential transmission events per unit of time for host B.

- **param.nContact.B**
  - parameter names (list of functions) for param.nContact for host-type B.

- **timeDep.nContact.B**
  - is nContact of host-type B dependent on the absolute time of the simulation (TRUE/FALSE)?

- **diff.nContact.B**
  - is nContact of host-type B different between states of the structured population (TRUE/FALSE)?

- **hostCount.nContact.B**
  - does nContact of host-type B vary with the host count (of either host A or B) in the state? (TRUE/FALSE); diff.nContact.B should be TRUE.

- **pTrans.B**
  - function that gives the probability of transmit a pathogen as a function of time since infection for host B.

- **param.pTrans.B**
  - parameter names (list of functions) for the pExit for host-type B.

- **timeDep.pTrans.B**
  - is pTrans of host-type B dependent on the absolute time of the simulation (TRUE/FALSE)?

- **diff.pTrans.B**
  - is pTrans host-type B different between states of the structured population (TRUE/FALSE)?

- **hostCount.pTrans.B**
  - does pTrans of host-type B vary with the host count (of either host A or B) in the state? (TRUE/FALSE); diff.pTrans.B should be TRUE.

- **prefix.host.B**
  - character(s) to be used as a prefix for the host B identification number.

- **print.progress**
  - if TRUE, displays a progress bar (current time/length.sim).

- **print.step**
  - print.progress is TRUE, step with which the progress message will be printed.
Details

The pExit and pTrans functions should return a single probability (a number between 0 and 1), and nContact a positive natural number (positive integer) or 0.

The param arguments should be a list of functions or NA. Each item name in the parameter list should have the same name as the argument in the corresponding function.

The use of timeDep (switch to TRUE) makes the corresponding function use the argument prestime (for "present time").

Value

An object of class nosoiSim, containing all results of the simulation.

Structure Matrix

The structure/transition matrix provided should of class matrix, with the same number of rows and columns, rows representing departure state and column the arrival state. All rows should add to 1. Probability values can be different for hosts A and B (so two different matrices), but the name of the column and the rows should be shared.

Order of Arguments

The user specified function's arguments should follow this order: t (mandatory), prestime (optional, only if timeDep is TRUE), current.in (optional, only if diff is TRUE), host.count.A or host.count.B (optional, only if hostCount is TRUE) and parameters specified in the list.

Structure Parameters

The pMove function should return a single probability (a number between 0 and 1).

The use of diff (switch to TRUE) makes the corresponding function use the argument current.in (for "currently in"). Your function should in that case give a result for every possible discrete state.

The use of hostCount (switch to TRUE) makes the corresponding function use the argument host.count.

Suffixes

The suffix .A or .B specifies if the considered function or parameter concerns host type A or B.

See Also

For simulations with a structure in continuous space, see dualContinuous. For simulations without any structures, see dualNone.

Examples

```r
#Host A
t_infectA_fct <- function(x){rnorm(x,mean = 12,sd=3)}
pTrans_hostA <- function(t,t_infectA){
  if(t/t_infectA <= 1){p=sin(pi*t/t_infectA)}
  if(t/t_infectA > 1){p=0}
```

return(p)
}

p_Move_fctA <- function(t){return(0.1)}

p_Exit_fctA <- function(t,t_infectA){
  if(t/t_infectA <= 1){p=0}
  if(t/t_infectA > 1){p=1}
  return(p)
}

time_contact_A = function(t){sample(c(0,1,2),1,prob=c(0.2,0.4,0.4))}

t_incub_fct_B <- function(x){rnorm(x,mean = 5,sd=1)}
p_max_fct_B <- function(x){rbeta(x,shape1 = 5,shape2=2)}

p_Exit_fct_B <- function(t,current.in){
  if(current.in=="A"){return(0.1)}
  if(current.in=="B"){return(0.2)}
  if(current.in=="C"){return(1)}
}

pTrans_hostB <- function(t,p_max,t_incub){
  if(t <= t_incub){p=0}
  if(t >= t_incub){p=p_max}
  return(p)
}

time_contact_B = function(t){round(rnorm(1, 3, 1), 0)}

transition.matrix = matrix(c(0,0.2,0.4,0.5,0,0.6,0.5,0.8,0),
nrow = 3, ncol = 3,
dimnames=list(c("A","B","C"),c("A","B","C")))

set.seed(6262)
test.nosoi <- nosoiSim(type="dual", popStructure="discrete",
  length.sim=40,
  max.infected.A=100,
  max.infected.B=200,
  init.individuals.A=1,
  init.individuals.B=0,
  init.structure.A="A",
  init.structure.B=NA,
  structure.matrix.A=transition.matrix,
  structure.matrix.B=transition.matrix,
  pExit.A = p_Exit_fctA,
  param.pExit.A = list(t_infectA = t_infectA_fct),
  pMove.A=p_Move_fctA,
  param.pMove.A=NA,
  timeDep.pMove.A=FALSE,
  diff.pMove.A=FALSE,
  timeDep.pExit.A=FALSE,
  nContact.A = time_contact_A,
  param.nContact.A = NA,
dualNone

**Dual-host pathogen in homogeneous hosts populations**

**Description**

This function, that can be wrapped within `nosoiSim`, runs a dual-host transmission chain simulation, without any structure features in both hosts populations. The simulation stops either at the end of given time (specified by `length.sim`) or when the number of hosts infected threshold (`max.infected`) is crossed.

**Usage**

dualNone(
  length.sim,
  max.infected.A,
  max.infected.B,
  init.individuals.A,
  init.individuals.B,
  pExit.A,
  param.pExit.A,
  timeDep.pExit.A = FALSE,
  nContact.A,
  param.nContact.A,
  timeDep.nContact.A = FALSE,
  pTrans.A = pTrans_hostA,
  param.pTrans.A = list(t_infectA=t_infectA_fct),
  timeDep.pTrans.A=FALSE,
  prefix.host.A="H",
  param.pExit.B = NA,
  timeDep.pExit.B=FALSE,
  diff.pExit.B=TRUE,
  pMove.B=NA,
  param.pMove.B=NA,
  timeDep.pMove.B=FALSE,
  diff.pMove.B=FALSE,
  nContact.B = time_contact_B,
  param.nContact.B = NA,
  timeDep.nContact.B=FALSE,
  pTrans.B = pTrans_hostB,
  param.pTrans.B = list(p_max=p_max_fct_B,
                       t_incub=t_incub_fct_B),
  timeDep.pTrans.B=FALSE,
  prefix.host.B="V")
pTrans.A,  
param.pTrans.A,  
timeDep.pTrans.A = FALSE,  
pExit.B,  
param.pExit.B,  
timeDep.pExit.B = FALSE,  
nContact.B,  
param.nContact.B,  
timeDep.nContact.B = FALSE,  
pTrans.B,  
param.pTrans.B,  
timeDep.pTrans.B = FALSE,  
prefix.host.B = "V",  
print.progress = TRUE,  
print.step = 10  
)

**Arguments**

- **length.sim** specifies the length (in unit of time) over which the simulation should be run.
- **max.infected.A** specifies the maximum number of individual hosts A that can be infected in the simulation.
- **max.infected.B** specifies the maximum number of individual hosts B that can be infected in the simulation.
- **init.individuals.A** number of initially infected individuals (hosts A).
- **init.individuals.B** number of initially infected individuals (hosts B).
- **pExit.A** function that gives the probability to exit the simulation for an infected host A (either moving out, dying, etc.).
- **param.pExit.A** parameter names (list of functions) for the pExit for host-type A.
- **timeDep.pExit.A** is pExit of host-type A dependent on the absolute time of the simulation (TRUE/FALSE)?
- **nContact.A** function that gives the number of potential transmission events per unit of time for host-type A.
- **param.nContact.A** parameter names (list of functions) for param.nContact for host-type A.
- **timeDep.nContact.A** is nContact of host-type A dependent on the absolute time of the simulation (TRUE/FALSE)?
- **pTrans.A** function that gives the probability of transmit a pathogen as a function of time since infection for host A.
- **param.pTrans.A** parameter names (list of functions) for the pExit for host A.
- **timeDep.pTrans.A** is pTrans of host-type A dependent on the absolute time of the simulation (TRUE/FALSE)?
prefix.host.A  character(s) to be used as a prefix for the host A identification number.
pExit.B      function that gives the probability to exit the simulation for an infected host B (either moving out, dying, etc.).
param.pExit.B parameter names (list of functions) for the pExit for host-type B.
timeDep.pExit.B is pExit of host-type B dependent on the absolute time of the simulation (TRUE/FALSE)?
nContact.B   function that gives the number of potential transmission events per unit of time for host B.
param.nContact.B parameter names (list of functions) for param.nContact for host-type B.
timeDep.nContact.B is nContact of host-type B dependent on the absolute time of the simulation (TRUE/FALSE)?
pTrans.B    function that gives the probability of transmit a pathogen as a function of time since infection for host B.
param.pTrans.B parameter names (list of functions) for the pExit for host-type B.
timeDep.pTrans.B is pTrans of host-type B dependent on the absolute time of the simulation (TRUE/FALSE)?
prefix.host.B character(s) to be used as a prefix for the host B identification number.
print.progress if TRUE, displays a progress bar (current time/length.sim).
print.step    print.progress is TRUE, step with which the progress message will be printed.

Details

The pExit and pTrans functions should return a single probability (a number between 0 and 1), and nContact a positive natural number (positive integer) or 0.

The param arguments should be a list of functions or NA. Each item name in the parameter list should have the same name as the argument in the corresponding function.

The use of timeDep (switch to TRUE) makes the corresponding function use the argument prestime (for "present time").

Value

An object of class nosoiSim, containing all results of the simulation.

Suffixes

The suffix .A or .B specifies if the considered function or parameter concerns host type A or B.

Order of Arguments

The user specified function’s arguments should follow this order: t (mandatory), prestime (optional, only if timeDep is TRUE), parameters specified in the list.
See Also

For simulations with a discrete structured host population, see dualDiscrete. For simulations with a structured population in continuous space, dualContinuous

Examples

```r
#Host A
t_infectA_fct <- function(x){rnorm(x,mean = 12,sd=3)}
pTrans_hostA <- function(t,t_infectA){
  if(t/t_infectA <= 1){p=sin(pi*t/t_infectA)}
  if(t/t_infectA > 1){p=0}
  return(p)
}
p_Exit_fctA <- function(t,t_infectA){
  if(t/t_infectA <= 1){p=0}
  if(t/t_infectA > 1){p=1}
  return(p)
}
time_contact_A = function(t){sample(c(0,1,2),1,prob=c(0.2,0.4,0.4))}

#Host B
t_incub_fct_B <- function(x){rnorm(x,mean = 5,sd=1)}
p_max_fct_B <- function(x){rbeta(x,shape1 = 5,shape2=2)}
p_Exit_fct_B <- function(t,prestime){(sin(prestime/12)+1)/5}
pTrans_hostB <- function(t,p_max,t_incub){
  if(t <= t_incub){p=0}
  if(t >= t_incub){p=p_max}
  return(p)
}
time_contact_B = function(t){round(rnorm(1, 3, 1), 0)}

set.seed(90)
test.nosoi <- nosoiSim(type="dual", popStructure="none",
length.sim=40,
max.infected.A=100,
max.infected.B=200,
init.individuals.A=1,
init.individuals.B=0,
pExit.A = p_Exit_fctA,
param.pExit.A = list(t_infectA = t_infectA_fct),
timeDep.pExit.A=FALSE,
nContact.A = time_contact_A,
param.nContact.A = NA,
timeDep.nContact.A=FALSE,
pTrans.A = pTrans_hostA,
param.pTrans.A = list(t_infectA=t_infectA_fct),
```

getCumulative

```r

test.nosoi
```

**getCumulative**

*Gets the cumulative number of infected hosts for the full length of the simulation*

**Description**

This function computes from the output of a nosoiSim simulation the cumulative count of infected hosts at each time step of the simulation. The output is a data.table.

**Usage**

```
getCumulative(nosoi.output)
```

**Arguments**

- `nosoi.output` Output of a nosoi simulation (object of class nosoiSim).

**Value**

The output is a data.table with the following structure:

- **t** Time-step (integer).
- **Count** Cumulative number of infected hosts at given time-step.
- **type** Host-type, identified by its user-defined prefix.

**See Also**

summary.nosoiSim
**getDynamic**

*Gets the current number of infected hosts for the full length of the simulation*

**Description**

This function computes from the output of a nosoiSim simulation the dynamic count of infected hosts at each time step (and each state if discrete structure) of the simulation. The output is a data.table.

**Usage**

`getDynamic(nosoi.output)`

**Arguments**

- `nosoi.output`: Output of a nosoi simulation (object of class `nosoiSim`).

**Value**

The output is a data.table with the following structure:

- **state** (only when discrete structure) Given state
- **Count** Current number of infected hosts at given time-step.
- **type** Host-type, identified by its user-defined prefix.
- **t** Time-step (integer).

**See Also**

`summary.nosoiSim`

---

**getHostData**

*Extracts specific data from a nosoiSim object*

**Description**

This function extracts data user-defined data (i.e. table.hosts, N.infected, table.state or popStructure) from a nosoiSim object.

**Usage**

```r
getHostData(
    nosoi.output,
    what = c("table.hosts", "N.infected", "table.state", "popStructure"),
    pop = "A"
)
```
Arguments

- `nosoi.output` an object of class `nosoiSim`
- `what` the data to get, among `table.hosts`, `N.infected`, `table.state` or `popStructure`.
- `pop` the population to be extracted (one of "A" or "B")

Value

Returns a `data.table` with the requested data.

See Also

To directly extract `table.hosts` or `table.state`, you can also use `getTableHosts` and `getTableState` respectively.

Examples

```r
t_incub_fct <- function(x){rnorm(x,mean = 5,sd=1)}
p_max_fct <- function(x){rbeta(x,shape1 = 5,shape2=2)}
p_Exit_fct <- function(t){return(0.08)}

proba <- function(t,p_max,t_incub){
  if(t <= t_incub){p=0}
  if(t >= t_incub){p=p_max}
  return(p)
}

time_contact <- function(t){round(rnorm(1, 3, 1), 0)}

test.nosoi <- nosoiSim(type="single", popStructure="none",
length=40,
max.infected=100,
init.individuals=1,
nContact=time_contact,
param.nContact=NA,
pTrans = proba,
param.pTrans = list(p_max=p_max_fct,
t_incub=t_incub_fct),
pExit=p_Exit_fct,
param.pExit=NA)

data.extracted <- getHostData(test.nosoi, "table.hosts", "A")
```
getR0

**Description**

Gets an estimate of secondary cases (what R0 usually tries to estimate) and its distribution from the output of a `nosoiSim` simulation. The actual calculation is based on inactive hosts at the end of the simulation to avoid bias introduced by hosts that have not finished their transmission potential.

**Usage**

```r
getR0(nosoi.output)
```

**Arguments**

- `nosoi.output` Output of a nosoi simulation (object of class `nosoiSim`).

**Value**

A list with the following items:

- `N.inactive` Number of inactive hosts at the end of the simulation.
- `R0.mean` Mean R0 based on the distribution (see below).
- `R0.dist` Distribution for each host of the secondary cases it generated (in case of dual-hosts, then the secondary cases of the same host-type).

**See Also**

- `summary.nosoiSim`

getTableHosts

**Description**

This function extracts the `table.hosts` for the request host-type from a `nosoiSim` object.

**Usage**

```r
getableHosts(nosoi.output, pop = "A")
```

**Arguments**

- `nosoi.output` an object of class `nosoiSim`
- `pop` the host-type to be extracted (either "A" or "B", if not dual-host, then "A")
Value

Returns a data.table with the requested data. The table.hosts (class data.table) contains informations about each host that has been simulated (one row is one host). The structure of the table is the following:

- **hosts.ID**: Unique identifier for the host, based on user-defined prefix and an integer.
- **inf.by**: Unique identifier for the host that infected the current one.
- **inf.in**: (only if structure is present) State or coordinates (in that case inf.in.x and inf.in.y) in which the host was infected.
- **current.in**: (only if structure is present) State or coordinates (in that case current.in.x and current.in.y) in which the host is at the end of the simulation.
- **current.env.value**: (only if continuous structure is present) Environmental value (raster cell value) in which the host is at the end of the simulation.
- **current.cell.raster**: (only if continuous structure is present) Raster cell numeric ID in which the host is at the end of the simulation.
- **host.count**: (only if structure is present) Host count in the current state or raster cell (beware, updated only if used).
- **inf.time**: When did the host enter the simulation (infection time).
- **out.time**: When did the host exit the simulation (NA if still active).
- **active**: Is the host still active at the end of the simulation (TRUE for YES, FALSE for NO)?
- **parameters**: The remaining columns are the sampled values for the individual-based parameters (if any) specified by the user.

---

**getTableState**

*Extracts table.state from a nosoiSim object*

**Description**

This function extracts the table.state for the request host-type from a nosoiSim object. table.state is present only if there is any structure (discrete or continuous) used.

**Usage**

```
getTableState(nosoi.output, pop = "A")
```

**Arguments**

- `nosoi.output`: an object of class nosoiSim
- `pop`: the host-type to be extracted (either "A" or "B", if not dual-host, then "A")
getTransmissionTree

Value

Returns a data.table with the requested data. The table.state (class data.table) contains information about the location of each host during time (one row is one host at one location). The structure of the table is the following:

hosts.ID Unique identifier for the host, based on user-defined prefix and an integer.
state State or coordinates (in that case state.x and state.y) in which the host is during that time interval.
current.env.value (only if continuous structure is present) Environmental value (raster cell value) in which the host is at the end of the simulation.
current.cell.raster (only if continuous structure is present) Raster cell numeric ID in which the host is at the end of the simulation.
time.from Time-step at which the host moved to the location.
time.to Time-step at which the host exited the location (either by exiting the simulation or moving somewhere else).

getTransmissionTree

Description

From a nosoi simulated epidemics, this function extracts the full transmission tree in a form mimicking a phylogenetic tree.

Usage

getTransmissionTree(nosoiInf)

Arguments

nosoiInf an object of class nosoiSim

Details

This function uses packages tidytree and treeio, that rely on ape.

Value

A tree of class treedata, containing a phylogenetic tree based on the transmission chain and the mapped data at all the nodes.

See Also

For exporting the annotated tree to other software packages, see functions in treeio (e.g. write.beast). To sub-sample this tree, see functions sampleTransmissionTree and sampleTransmissionTreeFromExiting.
Examples

```r
t_incub_fct <- function(x){rnorm(x,mean = 5,sd=1)}
p_max_fct <- function(x){rbeta(x,shape1 = 5,shape2=2)}
p.Exit_fct <- function(t){return(0.08)}
p.Move_fct <- function(t){return(0.1)}

proba <- function(t,p_max,t_incub){
  if(t <= t_incub){p=0}
  if(t >= t_incub){p=p_max}
  return(p)
}

time_contact = function(t){round(rnorm(1, 3, 1), 0)}

transition.matrix = matrix(c(0, 0.2, 0.4, 0.5, 0, 0.6, 0.5, 0.8, 0),
nrow = 3, ncol = 3,
dimnames = list(c("A", "B", "C"), c("A", "B", "C")))

set.seed(805)
test.nosoi <- nosoiSim(type="single", popStructure="discrete",
  length=20,
  max.infected=100,
  init.individuals=1,
  init.structure="A",
  structure.matrix=transition.matrix,
  pMove=p_Move_fct,
  param.pMove=NA,
  nContact=time_contact,
  param.nContact=NA,
  pTrans = proba,
  param.pTrans = list(p_max=p_min_fct,
    t_incub=t_incub_fct),
  pExit=p_Exit_fct,
  param.pExit=NA
)

## Make sure all needed packages are here
if (requireNamespace("ape", quietly = TRUE)
  || requireNamespace("tidytree", quietly = TRUE)
  || requireNamespace("treeio", quietly = TRUE)) {
  library(ape)
  library(tidytree)
  library(treeio)
}

# Full transmission tree
ttreedata <- getTransmissionTree(test.nosoi)
plot(ttreedata@phylo)

## Sampling "non dead" individuals
hID <- c("H-1", "H-7", "H-15", "H-100")
samples <- data.table(hosts = hID,
```
times = c(5.2, 9.3, 10.2, 16),
labels = paste0(hID, "-s"))

sampledTree <- sampleTransmissionTree(test.nosoi, ttreedata, samples)
plot(sampledTree@phylo)

## Sampling "dead" individuals
sampledDeadTree <- sampleTransmissionTreeFromExiting(ttreedata, hID)
plot(sampledDeadTree@phylo)
}

nosoiSim

Top-level function to use nosoi.

Description
This function determines which general settings the user wants to use for his simulation. All other arguments are passed down to the chosen simulator itself, such as singleNone, singleDiscrete, singleContinuous, dualNone, dualDiscrete or dualContinuous.

Usage
nosoiSim(type = "single", popStructure = "none", ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>specifies which type of pathogen we are interested in, either &quot;single&quot; or &quot;dual&quot;-host (e.g. arboviruses).</td>
</tr>
<tr>
<td>popStructure</td>
<td>specifies if the population in which the transmission is to occur is structured (&quot;none&quot;, &quot;discrete&quot; or &quot;continuous&quot;).</td>
</tr>
<tr>
<td>...</td>
<td>arguments to be passed on to the chosen simulator itself, such as singleNone, singleDiscrete, singleContinuous, dualNone, dualDiscrete or dualContinuous.</td>
</tr>
</tbody>
</table>

Value
An object of class nosoiSim, containing all results of the simulation. Class nosoiSim object have the following slots:

- **total.time** Number of time steps the simulation ran (integer).
- **type** String giving the simulation type ("single" or "dual" host).
- **host.info.A** object of class nosoiSimOne
  - **N.infected** Number of infected hosts (integer).
  - **table.hosts** Table containing the results of the simulation (see `getTableHosts` for more details on the table).
- **table.state** Table containing the results of the simulation, focusing on the movement history of each host (see `getTableState` for more details on the table).
**prefix.host** String containing the prefix used to name hosts (character string).

**popStructure** String giving the population structure (one of "none", "discrete" or "continuous").

**host.info.B:** object of class `nosoiSimOne` Same structure as `host.info.A`, but for host B (if it exists).

### See Also

- Individual simulation functions: `singleNone`, `singleDiscrete`, `singleContinuous`, `dualNone`, `dualDiscrete` and `dualContinuous`

- Functions to extract the results: `getTableHosts`, `getTableState`

- Summary statistics functions: `nosoiSummary`, `getCumulative`, `getDynamic`, `getR0`

### Examples

```r
t_incub_fct <- function(x){rnorm(x, mean = 5, sd=1)}
p_max_fct <- function(x){rbeta(x, shape1 = 5, shape2=2)}
p_Exit_fct <- function(t){return(0.08)}

proba <- function(t,p_max,t_incub){
  if(t <= t_incub){p=0}
  if(t >= t_incub){p=p_max}
  return(p)
}

time_contact = function(t){round(rnorm(1, 3, 1), 0)}

test.nosoi <- nosoiSim(type="single", popStructure="none",
                       length=40,
                       max.infected=100,
                       init.individuals=1,
                       nContact=time_contact,
                       param.nContact=NA,
                       pTrans = proba,
                       param.pTrans = list(p_max=p_max_fct,
                       t_incub=t_incub_fct),
                       pExit=p_Exit_fct,
                       param.pExit=NA)

test.nosoi
```

---

**nosoiSummary** **Summarizes the epidemiological features of a nosoi simulation**

### Description

This function provides summary information about the simulation (number of infected hosts, R0, etc.) as a list.
Usage

nosoiSummary(object)

## S3 method for class 'nosoiSim'
summary(object, ...)

Arguments

object Output of a nosoi simulation (object of class nosoiSim).

... further arguments passed to or from other methods.

Value

All computed data is provided in a list:

- **R0** Provides a sublist with number of inactive hosts at the end of the simulation N.inactive, mean R0 R0.mean, and R0 distribution R0.dist. For more details, see getR0.

- **dynamics** data.table with the count of currently infected (i.e. active) hosts at each time step of the simulation (by state if the simulation was in a discrete structured host population). For more details, see getDynamic.

- **cumulative** data.table with the cumulative count of infected hosts at each time step of the simulation. For more details, see getCumulative.

See Also

You can directly compute each elements of the list without using the summarise function. See getR0, getDynamic and getCumulative.

Examples

```r
t_incub_fct <- function(x){rnorm(x,mean = 5,sd=1)}
p_max_fct <- function(x){rbeta(x,shape1 = 5,shape2=2)}
p_Exit_fct <- function(t){return(0.08)}

proba <- function(t,p_max,t_incub){
  if(t <= t_incub){p=0}
  if(t >= t_incub){p=p_max}
  return(p)
}

time_contact <- function(t){round(rnorm(1, 3, 1), 0)}

test.nosoi <- nosoiSim(type="single", popStructure="none",
  length=40,
  max.infected=100,
  init.individuals=1,
  nContact=time_contact,
  param.nContact=NA,
  pTrans = proba,
```
param.pTrans = list(p_max=p_max_fct,
   t_incub=t_incub_fct),
   pExit=p.Exit_fct,
   param.pExit=NA)

nosoiSummary(test.nosoi)

---

sampleTransmissionTree

*Sample the transmission tree (phylogenetic tree-like)*

**Description**

Sample a full transmission tree. This function allows for sampling multiple times on the same lineage. When this happens, the sampled ancestor is a tip with length zero.

**Usage**

```r
sampleTransmissionTree(nosoiInf, tree, samples)
```

**Arguments**

- **nosoiInf**: an object of class `nosoiSim`
- **tree**: a `treedata` object created by function `getTransmissionTree`
- **samples**: a `data.table` object with the following entries:
  - `hosts`: Host ID of the individuals to be sampled
  - `times`: Times at which each host is sampled
  - `labels`: label for the corresponding tip in the tree

**Details**

The tree needs to be produced by function `getTransmissionTree` applied on the same `nosoiSim` object.

**Value**

A tree of class `treedata`, containing a phylogenetic tree based on the transmission chain and the mapped data at all the nodes.

**See Also**

For exporting the annotated tree to other software packages, see functions in `treeio` (e.g. `write.beast`).
To get the full transmission matrix, see `getTransmissionTree`.
For sampling only dead individuals, see `sampleTransmissionTreeFromExiting`.
Examples

t_incub_fct <- function(x){rnorm(x, mean = 5, sd=1)}
p_max_fct <- function(x){rbeta(x, shape1 = 5, shape2=2)}
p_Exit_fct <- function(t){return(0.08)}
p_Move_fct <- function(t){return(0.1)}

proba <- function(t, p_max, t_incub){
  if(t <= t_incub){p=0}
  if(t >= t_incub){p=p_max}
  return(p)
}

time_contact = function(t){round(rnorm(1, 3, 1), 0)}

transition.matrix = matrix(c(0, 0.2, 0.4, 0.5, 0, 0.6, 0.5, 0.8, 0),
nrow = 3, ncol = 3,
dimnames = list(c("A", "B", "C"), c("A", "B", "C")))

set.seed(805)
test.nosoi <- nosoiSim(type="single", popStructure="discrete",
length=20,
max.infected=100,
init.individuals=1,
init.structure="A",
structure.matrix=transition.matrix,
pMove=p_Move_fct,
param.pMove=NA,
nContact=time_contact,
param.nContact=NA,
pTrans = proba,
param.pTrans = list(p_max=p_max_fct,
t_incub=t_incub_fct),
pExit=p_Exit_fct,
param.pExit=NA)

## Make sure all needed packages are here
if (requireNamespace("ape", quietly = TRUE)
  || requireNamespace("tidytree", quietly = TRUE)
  || requireNamespace("treeio", quietly = TRUE)) {
library(ape)
library(tidytree)
library(treeio)
}

 hashtags

Full transmission tree
ttreedata <- getTransmissionTree(test.nosoi)
plot(ttreedata@phylo)

## Sampling “non dead” individuals
hID <- c("H-1", "H-7", "H-15", "H-100")
samples <- data.table(hosts = hID,
sampleTransmissionTreeFromExiting

Sample the transmission tree (phylogenetic tree-like) among the exited hosts

Description
Sample a full transmission tree. This function allows for sampling only exited (i.e. inactive) individuals (e.g. when the sampling procedure is destructive or cuts the hosts from the population). Beware because it does not influence the epidemiological process, it only means that the host has been sampled when exiting the simulation.

Usage
sampleTransmissionTreeFromExiting(tree, hosts)

Arguments
- tree: a tree data object created by function getTransmissionTree
- hosts: a vector of dead hosts to sample

Details
The tree needs to be produced by function getTransmissionTree applied on the same nosoiSim object.

Value
A tree of class treedata, containing a phylogenetic tree based on the transmission chain and the mapped data at all the nodes.

See Also
For exporting the annotated tree to other software packages, see functions in treeio (e.g. write.beast).
To get the full transmission matrix, see getTransmissionTree.
For sampling among non-dead individuals, see sampleTransmissionTree.
Examples

t_incub_fct <- function(x){rnorm(x,mean=5,sd=1)}
p_max_fct <- function(x){rbeta(x,shape1=5,shape2=2)}
p_Exit_fct <- function(t){return(0.08)}
p_Move_fct <- function(t){return(0.1)}

proba <- function(t,p_max,t_incub){
  if(t <= t_incub){p=0}
  if(t >= t_incub){p=p_max}
  return(p)
}

time_contact = function(t){round(rnorm(1, 3, 1), 0)}

transition.matrix = matrix(c(0, 0.2, 0.4, 0.5, 0, 0.6, 0.5, 0.8, 0),
nrow = 3, ncol = 3,
dimnames = list(c("A", "B", "C"), c("A", "B", "C")))

set.seed(805)
test.nosoi <- nosoiSim(type="single", popStructure="discrete",
  length=20,
  max.infected=100,
  init.individuals=1,
  init.structure="A",
  structure.matrix=transition.matrix,
  pMove=p_Move_fct,
  param.pMove=NA,
  nContact=time_contact,
  param.nContact=NA,
  pTrans = proba,
  param.pTrans = list(p_max=p_max_fct,
    t_incub=t_incub_fct),
  pExit=p_Exit_fct,
  param.pExit=NA)

## Make sure all needed packages are here
if (requireNamespace("ape", quietly = TRUE)
  || requireNamespace("tidytree", quietly = TRUE)
  || requireNamespace("treeio", quietly = TRUE)) {
  library(ape)
  library(tidytree)
  library(treeio)

  #' ## Full transmission tree
  ttreedata <- getTransmissionTree(test.nosoi)
  plot(ttreedata@phylo)

  ## Sampling “non dead” individuals
  hID <- c("H-1", "H-7", "H-15", "H-100")
  samples <- data.table(hosts = hID,
singleContinuous

Description
This function runs a single-host transmission chain simulation, with a structured host population (such as spatial features) in a continuous space. The simulation stops either at the end of given time (specified by length.sim) or when the number of hosts infected threshold (max.infected) is passed. The movement of hosts on the continuous space map is a random walk (Brownian motion) that can be modified towards a biased random walk where hosts tend to be attracted to higher values of the environmental variable defined by the raster.

Usage

```r
singleContinuous(
  length.sim,
  max.infected,
  init.individuals,
  init.structure,
  structure.raster,
  diff.pExit = FALSE,
  timeDep.pExit = FALSE,
  hostCount.pExit = FALSE,
  pExit,
  param.pExit,
  diff.pMove = FALSE,
  timeDep.pMove = FALSE,
  hostCount.pMove = FALSE,
  pMove,
  param.pMove,
  diff.sdMove = FALSE,
  timeDep.sdMove = FALSE,
  hostCount.sdMove = FALSE,
  sdMove,
  param.sdMove,
)```

attracted.by.raster = FALSE,
diff.nContact = FALSE,
timeDep.nContact = FALSE,
hostCount.nContact = FALSE,
nContact,
param.nContact,
diff.pTrans = FALSE,
timeDep.pTrans = FALSE,
hostCount.pTrans = FALSE,
pTrans,
param.pTrans,
prefix.host = "H",
print.progress = TRUE,
print.step = 10
)

Arguments

length.sim specifies the length (in unit of time) over which the simulation should be run.
max.infected specifies the maximum number of hosts that can be infected in the simulation.
init.individuals number of initially infected individuals.
init.structure in which location the initially infected individuals are located. A vector of coor-
dinates in the same coordinate space as the raster.
structure.raster raster object defining the environmental variable.
diff.pExit does pExit depend on the environmental variable (set by the raster) (TRUE/FALSE).
timeDep.pExit is pExit dependent on the absolute time of the simulation? (TRUE/FALSE)
hostCount.pExit does pExit vary with the host count in each raster cell? (TRUE/FALSE); if
TRUE, diff.pExit should be TRUE.
pExit function that gives the probability to exit the simulation for an infected host
(either moving out, dying, etc.).
param.pExit parameter names (list of functions) for the pExit.
diff.pMove does pMove depend on the environmental variable (set by the raster) (TRUE/FALSE).
timeDep.pMove does pMove depend on the absolute time of the simulation (TRUE/FALSE).
hostCount.pMove does pMove vary with the host count in each raster cell? (TRUE/FALSE); if
TRUE, diff.pMove should also be TRUE.
pMove function that gives the probability of a host moving as a function of time.
param.pMove parameter names (list of functions) for the pMove.
diff.sdMove does sdMove depend on the environmental variable (set by the raster) (TRUE/FALSE).
timeDep.sdMove does sdMove depend on the absolute time of the simulation (TRUE/FALSE).
`hostCount.sdMove` does `sdMove` vary with the host count in each raster cell? (TRUE/FALSE); if TRUE, `diff.sdMove` should be TRUE.

`sdMove` function that gives the distance traveled (based on coordinates); output is the standard deviation value for the Brownian motion.

`param.sdMove` parameter names (list of functions) for `sdMove`.

`attracted.by.raster` should the hosts be attracted by higher values in the environmental raster? (TRUE/FALSE).

`diff.nContact` does nContact depend on the environmental variable (set by the raster) (TRUE/FALSE).

`timeDep.nContact` is nContact dependent on the absolute time of the simulation? (TRUE/FALSE)

`hostCount.nContact` does nContact vary with the host count in each raster cell? (TRUE/FALSE); if TRUE, `diff.nContact` should be TRUE.

`nContact` function that gives the number of potential transmission events per unit of time.

`param.nContact` parameter names (list of functions) for `param.nContact`.

`diff.pTrans` does pTrans depend on the environmental variable (set by the raster) (TRUE/FALSE).

`timeDep.pTrans` is pTrans dependent on the absolute time of the simulation? (TRUE/FALSE)

`hostCount.pTrans` does pTrans vary with the host count in each raster cell? (TRUE/FALSE); if TRUE, `diff.pTrans` should be TRUE.

`pTrans` function that gives the probability of transmit a pathogen as a function of time since infection.

`param.pTrans` parameter names (list of functions) for the `pExit`.

`prefix.host` character(s) to be used as a prefix for the hosts identification number.

`print.progress` if TRUE, displays a progress bar (current time/length.sim).

`print.step` print.progress is TRUE, step with which the progress message will be printed.

**Details**

The `pExit` and `pTrans` functions should return a single probability (a number between 0 and 1), and `nContact` a positive natural number (positive integer) or 0.

The `param` arguments should be a list of functions or NA. Each item name in the parameter list should have the same name as the argument in the corresponding function.

The use of `timeDep` (switch to TRUE) makes the corresponding function use the argument `prestime` (for "present time").

**Value**

An object of class `nosoiSim`, containing all results of the simulation.

**Raster**

The structure raster provided provided should of class `raster`. High values of the environmental variable can attract hosts if `attracted.by.raster` is TRUE.
**Structure Parameters**

The `pMove` function should return a single probability (a number between 0 and 1), and `sdMove` a real number (keep in mind this number is related to your coordinate space).

The use of `diff` (switch to `TRUE`) makes the corresponding function use the argument `current.env.value` (for "current environmental value").

The use of `hostCount` (switch to `TRUE`) makes the corresponding function use the argument `host.count`.

**Order of Arguments**

The user specified function’s arguments should follow this order: `t` (mandatory), `prestime` (optional, only if `timeDep` is `TRUE`), `current.env.value` (optional, only if `diff` is `TRUE`), `host.count` (optional, only if `hostCount` is `TRUE`) and parameters specified in the list.

**See Also**

For simulations with a discrete structure, see `singleDiscrete`. For simulations without any structures, see `singleNone`.

**Examples**

```r
library(raster)
#Generating a raster the for movement
set.seed(860)

test.raster <- raster(nrows=100, ncols=100, xmn=-50, xmx=50, ymn=-50, ymx=50)
test.raster[] <- runif(10000, -80, 180)
test.raster <- focal(focal(test.raster, w=matrix(1, 5, 5), mean), w=matrix(1, 5, 5), mean)
plot(test.raster)

t_incub_fct <- function(x){rnorm(x, mean = 5, sd=1)}
p_max_fct <- function(x){rbeta(x, shape1 = 5, shape2=2)}
p_Move_fct <- function(t){return(0.1)}

sdMove_fct = function(t, current.env.value){return(100/(current.env.value+1))}

p_Exit_fct <- function(t){return(0.08)}

proba <- function(t,p_max,t_incub){
  if(t < t_incub)(p=0)
  if(t >= t_incub)(p=p_max)
  return(p)
}

time_contact = function(t){round(rnorm(1, 3, 1), 0)}

start.pos <- c(0,0)

test.nosoiA <- nosoiSim(type="single", popStructure="continuous",
	length=200,
```
max.infected=500,
init.individuals=1,
init.structure=start.pos,
structure.raster=test.raster,
pMove=p_Move_fct,
param.pMove=NA,
diff.sdMove=TRUE,
sdMove=sdMove_fct,
param.sdMove=NA,
attracted.by.raster=TRUE,
nContact=time_contact,
param.nContact=NA,
pTrans = proba,
param.pTrans = list(p_max=p_max_fct,
            t_incub=t_incub_fct),
pExit=p_Exit_fct,
param.pExit=NA)

singleDiscrete  Single-host pathogen in a structured (discrete) host population

Description
This function, that can be wrapped within nosoiSim, runs a single-host transmission chain simulation, with a discrete host population structure (e.g. spatial, socio-economic, etc.). The simulation stops either at the end of given time (specified by length.sim) or when the number of hosts infected threshold (max.infected) is crossed.

Usage
singleDiscrete(
    length.sim,
    max.infected,
    init.individuals,
    init.structure,
    structure.matrix,
    diff.pExit = FALSE,
    timeDep.pExit = FALSE,
    hostCount.pExit = FALSE,
    pExit,
    param.pExit,
    diff.pMove = FALSE,
    timeDep.pMove = FALSE,
    hostCount.pMove = FALSE,
    pMove,
    param.pMove,
    diff.nContact = FALSE,
timeDep.nContact = FALSE,
hostCount.nContact = FALSE,
nContact,
param.nContact,
diff.pTrans = FALSE,
timeDep.pTrans = FALSE,
hostCount.pTrans = FALSE,
pTrans,
param.pTrans,
prefix.host = "H",
print.progress = TRUE,
print.step = 10
)

Arguments

length.sim specifies the length (in unit of time) over which the simulation should be run.
max.infected specifies the maximum number of hosts that can be infected in the simulation.
init.individuals number of initially infected individuals.
init.structure in which state (e.g. location) the initially infected individuals are located.
structure.matrix transition matrix (probabilities) to go from location A (row) to B (column)
diff.pExit is pExit different between states of the structured population (TRUE/FALSE)
timeDep.pExit is pExit dependent on the absolute time of the simulation? (TRUE/FALSE)
hostCount.pExit does pExit varies with the host count in the state? (TRUE/FALSE); diff.pExit should be TRUE.
pExit function that gives the probability to exit the simulation for an infected host (either moving out, dying, etc.).
param.pExit parameter names (list of functions) for the pExit.
diff.pMove is pMove different between states of the structured population (TRUE/FALSE)
timeDep.pMove is pMove dependent on the absolute time of the simulation (TRUE/FALSE)
hostCount.pMove does pMove varies with the host count in the state? (TRUE/FALSE); diff.pMove should be TRUE.
pMove function that gives the probability of a host moving as a function of time.
param.pMove parameter names (list of functions) for the pMove.
diff.nContact is nContact different between states of the structured population (TRUE/FALSE)
timeDep.nContact is nContact dependent on the absolute time of the simulation? (TRUE/FALSE)
hostCount.nContact does nContact varies with the host count in the state? (TRUE/FALSE); diff.nContact should be TRUE.
nContact function that gives the number of potential transmission events per unit of time.

param.nContact parameter names (list of functions) for param.nContact.

diff.pTrans is pTrans different between states of the structured population (TRUE/FALSE)

timeDep.pTrans is pTrans dependent on the absolute time of the simulation? (TRUE/FALSE)

hostCount.pTrans does pTrans varies with the host count in the state? (TRUE/FALSE); diff.pTrans should be TRUE.

pTrans function that gives the probability of transmit a pathogen as a function of time since infection.

param.pTrans parameter names (list of functions) for the pExit.

prefix.host character(s) to be used as a prefix for the hosts identification number.

print.progress if TRUE, displays a progress bar (current time/length.sim).

print.step print.progress is TRUE, step with which the progress message will be printed.

Details

The pExit and pTrans functions should return a single probability (a number between 0 and 1), and nContact a positive natural number (positive integer) or 0.

The param arguments should be a list of functions or NA. Each item name in the parameter list should have the same name as the argument in the corresponding function.

The use of timeDep (switch to TRUE) makes the corresponding function use the argument prestime (for "present time").

Value

An object of class nosoiSim, containing all results of the simulation.

Structure Matrix

The structure matrix provided should of class matrix, with the same number of rows and columns, rows representing departure state and column the arrival state. All rows should add to 1.

Structure Parameters

The pMove function should return a single probability (a number between 0 and 1).

The use of diff (switch to TRUE) makes the corresponding function use the argument current.in (for "currently in"). Your function should in that case give a result for every possible discrete state.

The use of hostCount (switch to TRUE) makes the corresponding function use the argument host.count.

Order of Arguments

The user specified function’s arguments should follow this order: t (mandatory), prestime (optional, only if timeDep is TRUE), current.in (optional, only if diff is TRUE), host.count (optional, only if hostCount is TRUE) and parameters specified in the list.
singleNone

See Also

For simulations with a structure in continuous space, see singleContinuous. For simulations without any structures, see singleNone.

Examples

t_incub_fct <- function(x){rnorm(x,mean = 5,sd=1)}
p_max_fct <- function(x){rbeta(x,shape1 = 5,shape2=2)}
p_Exit_fct <- function(t){return(0.08)}
p_Move_fct <- function(t){return(0.1)}

proba <- function(t,p_max,t_incub){
  if(t <= t_incub){p=0}
  if(t > t_incub){p=p_max}
  return(p)
}

time_contact = function(t){round(rnorm(1, 3, 1), 0)}

transition.matrix = matrix(c(0,0.2,0.4,0.5,0,0.6,0.5,0.8,0),
  nrow = 3, ncol = 3,
  dimnames=list(c("A","B","C"),c("A","B","C")))

set.seed(805)
test.nosoiA <- nosoiSim(type="single", popStructure="discrete",
  length=20,
  max.infected=100,
  init.individuals=1,
  init.structure="A",
  structure.matrix=transition.matrix,
  pMove=p_Move_fct,
  param.pMove=NA,
  nContact=time_contact,
  param.nContact=NA,
  pTrans = proba,
  param.pTrans = list(p_max=p_max_fct,
                      t_incub=t_incub_fct),
  pExit=p_Exit_fct,
  param.pExit=NA)

---

singleNone

Single-host pathogen in a homogeneous host population

Description

This function, that can be wrapped within nosoiSim, runs a single-host transmission chain simulation, without any structure features in the host population. The simulation stops either at the end of given time (specified by length.sim) or when the number of hosts infected threshold (max.infected) is crossed.
Usage

singleNone(
  length.sim,
  max.infected,
  init.individuals,
  pExit,
  param.pExit,
  timeDep.pExit = FALSE,
  nContact,
  param.nContact,
  timeDep.nContact = FALSE,
  pTrans,
  param.pTrans,
  timeDep.pTrans = FALSE,
  prefix.host = "H",
  print.progress = TRUE,
  print.step = 10
)

Arguments

length.sim specifies the length (in unit of time) over which the simulation should be run.
max.infected specifies the maximum number of hosts that can be infected in the simulation.
init.individuals number of initially infected individuals.
pExit function that gives the probability to exit the simulation for an infected host (either moving out, dying, etc.).
param.pExit parameter names (list of functions) for the pExit.
timeDep.pExit is pExit dependent on the absolute time of the simulation? (TRUE/FALSE)
nContact function that gives the number of potential transmission events per unit of time.
param.nContact parameter names (list of functions) for param.nContact.
timeDep.nContact is nContact dependent on the absolute time of the simulation? (TRUE/FALSE)
pTrans function that gives the probability of transmit a pathogen as a function of time since infection.
param.pTrans parameter names (list of functions) for the pExit.
timeDep.pTrans is pTrans dependent on the absolute time of the simulation? (TRUE/FALSE)
prefix.host character(s) to be used as a prefix for the hosts identification number.
print.progress if TRUE, displays a progress bar (current time/length.sim).
print.step print.progress is TRUE, step with which the progress message will be printed.
Details

The pExit and pTrans functions should return a single probability (a number between 0 and 1), and nContact a positive natural number (positive integer) or 0.

The param arguments should be a list of functions or NA. Each item name in the parameter list should have the same name as the argument in the corresponding function.

The use of timeDep (switch to TRUE) makes the corresponding function use the argument prestime (for "present time").

Value

An object of class nosoiSim, containing all results of the simulation.

Order of Arguments

The user specified function’s arguments should follow this order: t (mandatory), prestime (optional, only if timeDep is TRUE), parameters specified in the list.

See Also

For simulations with a discrete structured host population, see singleDiscrete. For simulations with a structured population in continuous space, singleContinuous

Examples

t_incub_fct <- function(x){rnorm(x,mean = 5, sd=1)}
p_max_fct <- function(x){rbeta(x,shape1 = 5,shape2=2)}
p_Exit_fct <- function(t){return(0.08)}

proba <- function(t,p_max,t_incub){
  if(t <= t_incub){p=0}
  if(t >= t_incub){p=p_max}
  return(p)
}

time_contact <- function(t){round(rnorm(1, 3, 1), 0)}

test.nosoi <- nosoiSim(type="single", popStructure="none",
  length=40,
  max.infected=100,
  init.individuals=1,
  nContact=time_contact,
  param.nContact=NA,
  pTrans = proba,
  param.pTrans = list(p_max=p_max_fct,
    t_incub=t_incub_fct),
  pExit=p_Exit_fct,
  param.pExit=NA)

test.nosoi
Index

ape, 25

data.table, 20–22, 24, 25, 29, 30
dualContinuous, 2, 14, 19, 27, 28
dualDiscrete, 8, 10, 19, 27, 28
dualNone, 8, 14, 16, 27, 28
getCumulative, 20, 28, 29
getDynamic, 21, 28, 29
getHostData, 21
getR0, 23, 28, 29
getTableHosts, 22, 23, 27, 28
getTableState, 22, 24, 27, 28
getTransmissionTree, 25, 30, 32

nosoiSim, 7, 10, 14, 16, 18, 20, 21, 23–25, 27, 29, 30, 36, 38, 40, 41, 43

nosoiSummary, 28, 28

sampleTransmissionTree, 25, 30, 32
sampleTransmissionTreeFromExiting, 25, 30, 32

singleContinuous, 27, 28, 34, 41, 43
singleDiscrete, 27, 28, 37, 38, 43

singleNone, 27, 28, 37, 41, 41

summary.nosoiSim, 20, 21, 23
summary.nosoiSim(nosoiSummary), 28

treedata, 25, 30, 32

write.beast, 25, 30, 32