Package ‘hybridModels’

October 13, 2022

Version 0.3.7
Title An R Package for the Stochastic Simulation of Disease Spreading in Dynamic Networks
Date 2020-06-25
License GPL (>= 2)
Depends R (>= 3.3.1),
Imports doParallel, doRNG, foreach, ggplot2, GillespieSSA, parallel, reshape2, stats, stringr, grid
Description Simulates stochastic hybrid models for transmission of infectious diseases in dynamic networks. It is a metapopulation model in which each node in the network is a sub-population and disease spreads within nodes and among them, combining two approaches: stochastic simulation algorithm (<doi:10.1146/annurev.physchem.58.032806.104637>) and individual-based approach, respectively. Equations that models spread within nodes are customizable and there are two link types among nodes: migration and influence (commuting). More information in Fernando S. Marques, Jose H. H. Grisi-Filho, Marcos Amaku et al. (2020) <doi:10.18637/jss.v094.i06>.

LazyLoad true
LazyData true
Author Fernando S. Marques [aut, cre],
      Jose H. H. Grisi-Filho [aut],
      Marcos Amaku [aut]
Maintainer Fernando S. Marques <fernandosix@gmail.com>
URL https://github.com/fernandosm/hybridModels
BugReports https://github.com/fernandosm/hybridModels/issues
RoxygenNote 7.1.0
NeedsCompilation no
Repository CRAN
Date/Publication 2020-06-26 14:50:11 UTC
R topics documented:

- buildModelClass ........................................ 2
- findContactChain ......................................... 3
- hybridModel .............................................. 5
- hybridModels ............................................. 8
- networkSample ........................................... 9
- nodesCensus ............................................. 9
- plot ..................................................... 10
- simHM .................................................... 11
- summary .................................................. 12

Index ......................................................... 14

---

buildModelClass

It builds an object of a pre-specified class.

Description

buildModelClass is a generic function that calls a method to create an object based on the model’s name.

Usage

buildModelClass(
  x,
  var.names,
  init.cond,
  model.parms,
  probWeights,
  emigrRule,
  prop.func = NULL,
  state.var = NULL,
  infl.var = NULL,
  state.change.matrix = NULL
)

Arguments

- `x` is an empty object of a class requested.
- `var.names` a list with variable names of the network: the donor node, the receiver node, the time when each connection between donor to the receiver happened and the weight of these connection. The variables names must be "from", "to", "Time" and "arc", respectively.
- `init.cond` a named vector with initial conditions.
- `model.parms` a named vector with model’s parameters.
findContactChain

probWeights a named vector (optional and for migration type only) mapping state variables to migration probability weights based on state variables. These argument can be used to give weights for sampling individuals from node. They need not sum to one, they should be non-negative and not zero. For more information on the sampling method sample.

eemigrRule a string (optional and for migration type only) stating how many individual emigrate based on state variables. It requires that the network have weights instead of number of individuals that migrate.

prop.func a character vector with propensity functions of a generic node. See references for more details.

state.var a character vector with the state variables of the propensity functions.

infl.var a named vector mapping state variables to influence variables.

state.change.matrix is a state-change matrix. See references for more details.

Value
An object of the class requested.

References
.

findContactChain

Finding elements in contact chains of a dynamic network.

Description
Parallel function to find outgoing and ingoing contact chain elements.

Usage
findContactChain(
  Data,
  from,
  to,
  Time,
  selected.nodes,
  type = "size",
  numberOfcores = NULL
)
findContactChain

Arguments

Data data.frame with network information: node ID, origin node, destination node, and the time in which the link was established.

from character, variable name (column name) for origin node.

to character, variable name (column name) for destination node.

Time character, variable name (column name) for the time in which the link was established between two nodes.

selected.nodes vector, the function will find the contact chain of the nodes present in the selected.nodes vector.

type character, of returned result. type = 'size' (default), will return the size of 'outgoing' and 'ingoing' contact chains. Type = 'chain' will return also the nodes in each chain (might be slow for big data sets).

numberOfcores integer, number of cores used to calculate the contact chain (default is NULL, that will lead the algorithm to use the max number of cores).

Details

This is a function that find elements of a contact chain from a dynamic network.

Value

setting type = 'size', it returns a data.frame with ingoing and outgoing contact chains size, add 1 to include the selected.nodes. Setting type = 'chain', it returns a list with the data frame and elements of ingoing and outgoing chains.

References


Examples

# Loading data
data(networkSample) # help("networkSample"), for more info.

# contact chain function
selected.nodes <- c(37501, 36811, 36812)
contact.chain <- findContactChain(Data = networkSample, from = 'originID',
to = 'destinationID', Time = 'Day', selected.nodes,
type = 'chain', numberOfcores = 2)

hybridModel

Hybrid model simulation.

Description

hybridModel function runs hybrid models simulations.

Usage

hybridModel(
  network = stop("undefined 'network'"),
  var.names = NULL,
  link.type = "migration",
  model = "custom",
  probWeights = NULL,
  emigrRule = NULL,
  init.cond = stop("undefined 'initial conditions'"),
  fill.time = F,
  model.parms = stop("undefined 'model parameters'"),
  prop.func = NULL,
  state.var = NULL,
  infl.var = NULL,
  state.change.matrix = NULL,
  ssa.method = NULL,
  nodesCensus = NULL,
  sim.number = 1,
  pop.correc = TRUE,
  num.cores = "max"
)
Arguments

network a data.frame with variables that describe the donor node, the receiver node, the time when each connection between donor to the receiver happened and the number of individual or weight of these connection.

var.names a list with variable names of the network: the donor node, the receiver node, the time when each connection between donor to the receiver happened and the weight of these connection. The variables names must be "from", "to", "Time" and "arc", respectively.

link.type a character describing the link type between nodes. There are two types: 'migration' and 'influence'. In the migration link type there are actual migration between nodes. In the influence link type individuals does not migrate, just influences another node.

model a character describing model’s name.

probWeights a named vector (optional and for migration type only) mapping state variables to migration probability weights based on state variables. These argument can be used to give weights for sampling individuals from node. They need not sum to one, they should be non-negative and not zero. For more information on the sampling method sample.

emigrRule a string (optional and for migration type only) stating how many individual emigrate based on state variables. It requires that the network have weights instead of number of individuals that migrate.

init.cond a named vector with initial conditions.

fill.time It indicates whether to return all dates or just the dates when nodes get connected.

model.parms a named vector with model’s parameters.

prop.func a character vector with propensity functions of a generic node. See references for more details

state.var a character vector with the state variables of the propensity functions.

infl.var a named vector mapping state variables to influence variables.

state.change.matrix is a state-change matrix. See references for more details

ssa.method a list with SSA parameters. The default method is the direct method. See references for more details

nodesCensus a data.frame with the first column describing nodes’ ID, the second column with the number of individuals and the third describing the day of the census.

sim.number Number of repetitions. The default value is 1

pop.correc Whether hybridModel function tries to balance the number of individuals or not. The default value is TRUE.

num.cores number of threads/cores that the simulation will use. the default value is num.cores = 'max', the Algorithm will use all threads/cores available.

Value

Object containing a data.frame (results) with the number of individuals through time per node and per state.
References


See Also

GillespieSSA.

Examples

# Migration model
# Parameters and initial conditions for an SIS model
# loading the data set
data(networkSample) # help("networkSample"), for more info
networkSample <- networkSample[which(networkSample$Day < "2012-03-20"),]

var.names <- list(from = 'originID', to = 'destinationID', Time = 'Day', arc = 'num.animals')

prop.func <- c('beta * S * I / (S + I)', 'gamma * I')
state.var <- c('S', 'I')
state.change.matrix <- matrix(c(-1, 1, # S
                                1, -1), # I
                              nrow = 2, ncol = 2, byrow = TRUE)

model.parms <- c(beta = 0.1, gamma = 0.01)

init.cond <- rep(100, length(unique(c(networkSample$originID, networkSample$destinationID))))

names(init.cond) <- paste('S', unique(c(networkSample$originID, networkSample$destinationID)), sep = '')
init.cond <- c(init.cond, c(I36811 = 10, I36812 = 10)) # adding infection

# running simulations, check the number of cores available (num.cores)
sim.results <- hybridModel(network = networkSample, var.names = var.names,
                            model.parms = model.parms, state.var = state.var,
                            prop.func = prop.func, init.cond = init.cond,
                            state.change.matrix = state.change.matrix,
                            sim.number = 2, num.cores = 2)

# default plot layout (plot.types: 'pop.mean', 'subpop', or 'subpop.mean')
plot(sim.results, plot.type = 'subpop.mean')

# changing plot layout with ggplot2 (example)
# uncomment the lines below to test new layout exemple
#library(ggplot2)
#plot(sim.results, plot.type = 'subpop') + ggtitle('New Layout') +
# theme_bw() + theme(axis.title = element_text(size = 14, face = "italic"))
# Influence model
# Parameters and initial conditions for an SIS model
# loading the data set
data(networkSample) # help("networkSample"), for more info
networkSample <- networkSample[which(networkSample$Day < "2012-03-20"),]

var.names <- list(from = 'originID', to = 'destinationID', Time = 'Day',
                  arc = 'num.animals')

prop.func <- c('beta * S * (I + i) / (S + I + s + i)', 'gamma * I')
state.var <- c('S', 'I')
infl.var <- c(S = "s", I = "i") # mapping influence
state.change.matrix <- matrix(c(-1, 1, 1, -1), nrow = 2, ncol = 2, byrow = TRUE)

model.parms <- c(beta = 0.1, gamma = 0.01)

init.cond <- rep(100, length(unique(c(networkSample$originID,
                                       networkSample$destinationID))))
names(init.cond) <- paste('S', unique(c(networkSample$originID,
                                       networkSample$destinationID)), sep = '')
init.cond <- c(init.cond, c(I36811 = 10, I36812 = 10)) # adding infection

# running simulations, check num of cores available (num.cores)
# Uncomment to run
# sim.results <- hybridModel(network = networkSample, var.names = var.names,
#                              model.parms = model.parms, state.var = state.var,
#                              infl.var = infl.var, prop.func = prop.func,
#                              init.cond = init.cond,
#                              state.change.matrix = state.change.matrix,
#                              sim.number = 2, num.cores = 2)

# default plot layout (plot.types: 'pop.mean', 'subpop', or 'subpop.mean')
# plot(sim.results, plot.type = 'subpop.mean')

---

**hybridModels**

**hybridModels**: an R package for stochastic simulation of disease spreading in dynamic networks.

### Description

The hybridModels package provides functions to simulate stochastic models in dynamic networks, using two processes in different scales: 1 Global scale to simulate the transmission from one node to another 2 Local scale to simulate the transmission inside the node

### Modeling transmission of diseases

'Framework to run n simulations in dynamic network and plot results'
networkSample

*Daily record of animal’s movement (from 2012 to 2013).*

**Description**

One dataset containing the number of animals that were moved from one node to another.

**Usage**

`networkSample`

**Format**

A data frame with 78 rows and 4 variables:

- **Day**: The day when the movement occurs
- **originID**: The ID of the origin premises
- **destinationID**: The ID of the destination premises
- **num.animals**: The number of animals traded

**Source**

ADAGRO

---

nodesCensus

*Information about animal premises (from 2012 to 2013).*

**Description**

A dataset containing animal premises’ identification and census.

**Usage**

`nodesCensus`

**Format**

A data frame with 507 rows and 2 variables:

- **nodes.ID**: The ID of the premises
- **pop**: premises’s population size
Description

plot.HM is a method to plot hybrid models from this package.

Usage

```r
## S3 method for class 'HM'
plot(x, sim = 1, plot.type = "subpop", facet.scales = "free_y", ...)  
```

Arguments

- `x`: HM object
- `sim`: indicates which simulation to plot.
- `plot.type`: plots the mean number of each state variable for the whole population ('pop.mean'), or the subpopulations of a particular simulation ('subpop', default value), or the mean of each subpopulation ('subpop.mean').
- `facet.scales`: should scales be fixed ("free_y", the default), free ("free"), or free in one dimension ("free_x", "free_y"). See ggplot2 package for more details.
- `...`: arguments to be passed to methods.

References


Examples

```r
# Parameters and initial conditions for an SIS model
# loading the data set
data(networkSample) # help("networkSample"), for more info
networkSample <- networkSample[which(networkSample$Day < "2012-03-20"),]

var.names <- list(from = 'originID', to = 'destinationID', Time = 'Day',
                 arc = 'num.animals')

prop.func <- c('beta * S * I / (S + I)', 'gamma * I')
state.var <- c('S', 'I')
state.change.matrix <- matrix(c(-1, 1, # S
                                1, -1), # I
                               nrow = 2, ncol = 2, byrow = TRUE)

model.parms <- c(beta = 0.1, gamma = 0.01)
```
init.cond <- rep(100, length(unique(c(networkSample$originID, networkSample$destinationID))))
names(init.cond) <- paste('S', unique(c(networkSample$originID, networkSample$destinationID)), sep = '')
init.cond <- c(init.cond, c(I36811 = 10, I36812 = 10)) # adding infection

# running simulations, check num of cores available (num.cores)
sim.results <- hybridModel(network = networkSample, var.names = var.names, model.parms = model.parms, state.var = state.var, prop.func = prop.func, init.cond = init.cond, state.change.matrix = state.change.matrix, sim.number = 2, num.cores = 2)

# default plot layout (plot.types: 'pop.mean', 'subpop', or 'subpop.mean')
plot(sim.results, plot.type = 'subpop.mean')

# changing plot layout with ggplot2 (example)
# uncomment the lines below to test new layout exemple
#library(ggplot2)
#plot(sim.results, plot.type = 'subpop') + ggtitle('New Layout') +
# theme_bw() + theme(axis.title = element_text(size = 14, face = "italic"))

---

**simHM**  
*It runs the chosen hybrid model.*

---

**Description**

**simHM** is generic function that calls a method to run the simulation base on object's class

**Usage**

`simHM(x, network, sim.number, num.cores = "max", fill.time)`

**Arguments**

- **x** of a specific class of model.
- **network** a `data.frame` with variables that describe the donor node, the receiver node, the time when each connection between donor to the receiver happened and the number of individual or weight of these connection.
- **sim.number** Number of repetitions. The default value is 1
- **num.cores** number of threads/cores that the simulation will use. The default value is num.cores = 'max', the Algorithm will use all threads/cores available.
- **fill.time** It indicates whether to return all dates or just the dates when nodes get connected.

**Value**

A `data.frame` with the number of individuals through time per node, per state and per simulation.
summary

References
.

See Also

GillespieSSA.

summary

summary for hybrid models

Description

summary.HM is a method to print a summary with basic description of nodes’ states at a specific
time (the time must be present in the network data). The default value is Null, that means it prints
nodes’ final states.

Usage

## S3 method for class 'HM'
summary(object, at = NULL, stateVars = NULL, nodes = NULL, ...)

Arguments

object  
HM object

at  
the date (as character) that will be used to print the summary

stateVars  
vector containing the state variable to summarize. The default value is NULL,
which will print a summary with all states.

nodes  
vector containing the nodes of interest. The default value is NULL, which will
print a summary with all nodes.

...  
arguments to be passed to methods.

References

Package for the Stochastic Simulation of Disease Spreading in Dynamic Network. In: Journal of

Examples

# Parameters and initial conditions for an SIS model
# loading the data set
data(networkSample) # help("networkSample"), for more info
networkSample <- networkSample[which(networkSample$Day < "2012-03-20"),]

var.names <- list(from = 'originID', to = 'destinationID', Time = 'Day',
                 arc = 'num.animals')
prop.func <- c('beta * S * I / (S + I)', 'gamma * I')
state.var <- c('S', 'I')
state.change.matrix <- matrix(c(-1, 1, # S
                               1, -1), # I
                               nrow = 2, ncol = 2, byrow = TRUE)
model.parms <- c(beta = 0.1, gamma = 0.01)
init.cond <- rep(100, length(unique(c(networkSample$originID,
                                     networkSample$destinationID))))
names(init.cond) <- paste('S', unique(c(networkSample$originID,
                                           networkSample$destinationID)), sep = '')
init.cond <- c(init.cond, c(I36811 = 10, I36812 = 10)) # adding infection

# running simulations, check num of cores available (num.cores)
sim.results <- hybridModel(network = networkSample, var.names = var.names,
                           model.parms = model.parms, state.var = state.var,
                           prop.func = prop.func, init.cond = init.cond,
                           state.change.matrix = state.change.matrix,
                           sim.number = 4, num.cores = 2)

summary(sim.results, stateVars = c('S', 'I'), nodes = c(36812, 36813))
Index

* datasets
  networkSample, 9
  nodesCensus, 9
buildModelClass, 2
character, 4, 6
data.frame, 4, 6, 11
findContactChain, 3
GillespieSSA, 7, 12
hybridModel, 5
hybridModels, 8
integer, 4
list, 2, 4, 6
matrix, 3, 6
networkSample, 9
nodesCensus, 9
plot, 10
sample, 3, 6
simHM, 11
summary, 12
vector, 2–4, 6, 12