Package ‘hybridModels’

September 27, 2017

Version 0.2.15

Title Stochastic Hybrid Models in Dynamic Networks

Date 2017-09-26

License GPL (>= 2)

Depends R (>= 3.3.1),

Imports doParallel, 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 or its approximations (Gillespie DT (2007) <doi:10.1146/annurev.physchem.58.032806.104637>) and individual-based approach, respectively. Movement among nodes are data based and can be irregular. Equations that models spread within nodes are customizable and there are two link types among nodes: migration and influence (commuting).

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 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-09-27 09:18:47 UTC
buildModelClass

R topics documented:

buildModelClass .................................................. 2
findContactChain ................................................... 3
hybridModel ........................................................ 4
hybridModels ......................................................... 7
networkSample ....................................................... 8
nodesCensus .......................................................... 8
plot ................................................................. 9
simHM .............................................................. 10
summary ............................................................. 11

Index 13

buildModelClass  It builds an object of a pre-specified class.

Description

buildModelClass is generic function that calls a method to create a object base on model's name.

Usage

buildModelClass(x, var.names, init.cond, model parms, 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 reciever node,
the time when each connection between donor to the reciever 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.
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.
**findContactChain**

Finding elements in contact chains of a dynamic network.

**Description**

Parallel function to find outgoing and ingoing contact chain elements.

**Usage**

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

**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 huge 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**

a data.frame with ingoing and outgoing contact chains size. or 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, var.names = NULL, link.type = "migration", model = "custom", init.cond, fill.time = F, model.params, prop.func = NULL, state.var = NULL, infl.var = NULL, state.change.matrix = NULL, ssa.method = list(method = "D", epsilon = 0.03, nc = 10, dtf = 10, nd = 100), nodesCensus = NULL, sim.number = 1, pop.correc = TRUE, num.cores = "max")
**Arguments**

- **network**: a data.frame with variables that describe the donor node, the reciever node, the time when each connection between donor to the reciever happened and the weight of these connection.
- **var.names**: a list with variable names of the network: the donor node, the reciever node, the time when each connection between donor to the reciever 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.
- **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(136811 = 10, 136812 = 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"))

# 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, # 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(136811 = 10, 136812 = 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 dynamics 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.

**Modelling 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 5 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
Summary plots for hybrid Models

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'), 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.

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
```
simHM runs the chosen hybrid model.

Description

simHM is a generic function that calls a method to run the simulation based 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 weight of these connections.
- **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.

References

.
Summary

The `summary` function provides 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, printing nodes' final states.

Description

`summary.HM` is a method to print a summary of nodes' states at a specific time (the time must be present in the network data). The default value is Null, meaning it prints nodes' final states.

Usage

```r
## 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.

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))))
```
networkSample$destinationID)))
names(init.cond) <- paste('S', unique(c(networkSample$originID,
      networkSample$destinationID)), sep = '')
init.cond <- c(init.cond, c(136811 = 10, 136812 = 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

*Topic **datasets**
  networkSample, 8
  nodesCensus, 8

buildModelClass, 2

character, 3, 5

data.frame, 3, 5, 10

findContactChain, 3

GillespieSSA, 6, 11

hybridModel, 4
hybridModels, 7
hybridModels-package (hybridModels), 7

integer, 3

list, 2, 3, 5

matrix, 2, 5

networkSample, 8
nodesCensus, 8

plot, 9

simHM, 10
summary, 11

vector, 2, 3, 5, 11