Package 'dizzysNEWYANN'

July 20, 2015

Type Package

Title Simulating SEIR/SIR models

Version 0.3

Date 2015-30-01

Author Tran Thi Cam Giang

Maintainer Tran Thi Cam Giang <camgiang2010@gmail.com>

Description This package permit us to simulate infectious diseases by using the SIR/SEIR models. To do that, we implement the direct algorithm of Gillespie in 1977 as described in the book ``Modeling Infectious Diseases IN HUMANS AND ANIMALS'' by Keeling and Pejman Rohani (2008), and the adaptive tau leaping to approximate the trajectory of a continuous-time stochastic process as described by Cao et al. (2007) The Journal of Chemical Physics. This package is based upon work supported by the Institute de Recherche pour Developpement at Paris in France and by Jean-Daniel Zucker and Marc Choisy.

Imports base,stats,graphics,methods, utils

License 0.4

Repository CRAN

LazyLoad true

Collate 'zzz.R' 'seir.r' 'equilibrium.r' 'seir.plot.r' 'seir.generic.r' 'seir.simul.r' 'seir.adaptivetau.r' 'equilibriumYANN.r' 'seirYANN.r'

R topics documented:

dizzys-package 2
coef.seir 3
confint.pers.rate 4
equilibrium 5
lines.seir 6
pers.rate.obj 8
persistence 9
plot.pers 10
plot.seir 11

pop.seir 13
print.seir 14
seir-class 16
simul 17
str.seir 20
summary.seir 21

Index **22**

`dizzys-package` *Modelling of Infectious Diseases*

Description

Permiting to simulate infectious diseases by using the SIR/SEIR models under deterministic and stochastic models.

Details

Package:	dizzys
Type:	Package
Version:	1.0
Date:	2013-05-02
License:	R
Depends:	methods, deSolve, rgl, survival, KMsurv

Author(s)

TRAN Thi Cam Giang <camgiang2010@gmail.com>

References

Matt J. Keeling and Pejman Rohani (2008) Modeling Infectious Diseases IN HUMANS AND ANIMALS. Princeton University Press.

Cao Y, Gillespie DT, Petzold LR,The Journal of Chemical Physics, 2007.

Norman Matloff (2009), The Art of R Programming.

See Also

package 'dizzyz'

`coef.seir` *Coefficient of seir Object*

Description

coef is a generic function which extracts coefficients of parameters from objects returned by modeling functions.

Usage

```
coef.seir(object, ...)
```

Arguments

`object`	an object for which the extraction of model coefficients is meaningful.
`...`	other arguments.

Details

The function uses an object of class 'seir' for extracting coefficients of parameters. There are two parameter types to show. One is the initial values of state variables. The other is the values of parameters used in simulation.

Value

'coef.seir' showing the initial values of state variables and the values of parameters used in simulation for all cities.

Author(s)

TRAN Thi Cam Giang

References

Extract Model Coefficients in R.

See Also

The generic function 'coef' in R

Examples

```
seirobj1<-seir(N=1e7)
coef(seirobj1)

seirobj2<-seir(nbVilles=3, N=c(1e7,1e6))
coef(seirobj2)
```

`confint.pers.rate` *Confidence interval for the estimated global disease persistence rate in a metapopulation.*

Description

This function permits us to find the confidence interval for the estimated global disease persistence rate in a metapopulation.

Usage

```
confint.pers.rate<-function(object, level)
```

Arguments

`object`	a seir object with the condition, this object has to have the value of the parameter 'persistence'. Because, based on this parameter, we can estimated the survival rate.
`level`	the confidence level required.

Details

For this function, in the step 1, we also use the function 'survreg' of the package 'survival' to estimate the global persistence rate for a metapopulation. Then, we use the generic function 'confint' to find the confidence interval for the estimated global disease persistence rate in the metapopulation.

Value

Result returned is a number.

Author(s)

TRAN Thi Cam Giang

See Also

'confint.pers.rate' function in the 'dizzys' package.

Examples

```
#STO, STO
sto<-seir(N=10e5,type="stoch",duration=30*365,beta1=0.1,nbVilles=10)
objper<- persistence(sto)
pers.obj<-pers.rate.obj(objper)
confint.pers.rate(pers.obj)
pause()
```

`equilibrium` *Finding Endemic Equilibrium of a seasonally-forced SEIR/SIR model*

Description

'equilibrium' returns the values of the state variables of an SEIR/SIR model at endemic equilibrium.

Usage

```
equilibrium(duration, unitTIME, N, mu, beta0, beta1, sigma, gamma, phi, T)
```

Arguments

`duration`	time values over which to perform the numerical integration.
`unitTIME`	time unit of simulation.
`N`	number of population.
`mu`	per capita birth and death rates (per day).
`beta0`	mean value of contact rate (per day).
`beta1`	amplitude of contact rate.
`sigma`	transition rate from exposed (E) to infected (I), per day (inverse of the average duration of the latency period.
`gamma`	recovery rate (per day).
`phi`	phase of the contact rate (in radians).
`T`	period of the contact rate (in days).

Details

The host population is supposed to be at demographic equilibrium with the births balancing the deaths (births and death occur with the same rate 'mu') and a total population size constant and equal to 1. The contact rate is forced by a sinusoid of period 'T' and phase 'phi'. The SEIR epidemiological model is defined by the following set of differential equations:

dS/dt = mu - beta0 * (1 + beta1 * cos(2 * pi * t / T + phi)) * S * I - mu * S

dE/dt = beta0 * (1 + beta1 * cos(2 * pi * t / T + phi)) * S * I - mu * E - sigma * E

dI/dt = sigma * E - mu * I - gamma * I

dR/dt = gamma * I - mu * R

'beta0' is the mean value of the contact rate and 'beta1' is the amplitude (in percentage of the mean) around the mean. The endemic equilibrium value of this sinusoidally-forced epidemiological system is found in two steps. First the endemic equilibium point (S*,E*,I*,R*) of the unforced system ('beta1 = 0') is found from the following analytical equations:

S* = (gamma + mu)*(sigma + mu)/(beta0 * sigma)

E* = mu*((1/(sigma + mu))-((gamma + mu)/(beta0 * sigma)))

I* = mu*((beta0*sigma - (gamma + mu)*(sigma + mu))/(beta0*(gamma + mu)*(sigma + mu)))

R* = 1 - S* - E* - I*

This endemic equilibium point is used as a starting point for numerical integration of the system of differential equations. This is done by the 'seir.det' function that uses the 'ode' function of the 'deSolve' package. The last values of the of the state variable are returned. These correspond to a given point on the equilibrium limit cycle (assuming that this equilibrium limit cycle is reached) and the value of this point will depend on the phase 'phi'.

Value

'equilibrium' returns a named vector of 3 numerical proportions containing the values of the state variables S, E and I on one point of the equilibrium limit cycle (provided that this limit cycle is reached). In case this limit cycle is reached, the value of the S, E and I state variables will depend on the input phase value 'phi'.

Author(s)

TRAN Thi Cam Giang

References

Anderson RM & May RM (1991) Infectious Diseases of Humans - Dynamics and Control. Oxford University Press.

See Also

'seir' in package 'dizzyz' and 'ode' in package 'deSolve'.

Examples

```
## The point on the limit cycle depends on the input phase value 'phi':
equilibrium(mu=.000456, beta0=1000/365, beta1=.1, sigma=1/7, gamma=1/7, phi=0, T=365,duration=100,unitTIME=1)
equilibrium(mu=.000456, beta0=1000/365, beta1=.1, sigma=1/7, gamma=1/7, phi=pi/2, T=365, duration=100,unitTIME=7
```

`lines.seir` *Add Connected Line Segments of an seir object to a Plot 2D/3D*

Description

A generic function taking coordinates given of the slot 'pop' of a seir object in various ways and joining the corresponding points with line segments.

Usage

```
lines.seir(object, x, y, z, pop, col, type, unitTIME, proj,...)
```

Arguments

`object`	a seir object, this is the important parameter.
`x, y, z`	the names or the positions of the columns in the slot 'pop' of the seir object. This arguments are corresponding to the x, y and z coordinates of the plot. Normally, if 'z' is NULL, we only can add a line 2D to a plot 2D, on the other side, if 'z' is one of the names or the positions of the columns ("time","S","E","P","R","N") or (1,2,3,4,5,6), we can add a line 3D to a plot 3D.
`pop`	numeric. What subpopulations are chosen to add from the subpopulation set. The subpopulation set is a series of natural number from one to n, where n is the number of subpopulation in a metapopualtion of n subpopulation. Normally, 'pop' is NULL, it allows us to add all lines of the n subpopulations to the plot. In the other side, 'pop' is a good numeric, it means that 'pop' is in the series (1,...,n), we only can add the lines of the subpopulation in 'pop' to the plot.
`col`	the colors for lines. Multiple colors can be specified so that each line can be given its own color. If there are fewer colors than number of lines they are recycled in the standard fashion. Lines will all be plotted in the first colour specified.
`unitTIME`	the unit of time. It is 'day' corresponding to unitTIME=1, is 'week' corresponding to unitTIME=7, is 'month' corresponding to unitTIME=30 and is 'year' corresponding to unitTIME=365. Normally, unitTIME is equal to 1, however we can change it.
`proj`	list of the plane names. This argument is only used in the plot 3D. It allows us to add the projection of the lines 3D on the planes. It is necessary to give the value of 'proj' corresponding to the x, y and z axes.
`...`	further graphical parameters such as 'lwd', 'lty'.

Details

The function 'lines' is the partial analogue of plot.seir with the argument add=T. Moreover, for the x, y, and z coordinates, the slot 'pop' of the seir object is a list of the data frames corresponding to each subpopulation in a metapopulation of n subpopulations. The each data frame often has six columns that are 'time' being a time column, at time t, 'S' being the number of susceptibles, 'E' being the number of exposed individuals, 'P' being the number of infectives, 'R' being the number of removed individuals with immunity and 'N' being the number of population, it is the sum of S, E, P, R.

Author(s)

TRAN Thi Cam Giang

References

Murrell, P. (2005) R Graphics. Chapman & Hall/CRC Press.

Examples

```
#creating a plot
#adding a line to the plot
seir(nbVilles=2)->obj
seir(nbVilles=1)->obj1
#2D
plot(obj,col="red")
lines(obj1,col="blue",lwd=2)
#3D
plot(obj,z="S",col="red",proj=list(c("time","P")))
lines(obj1,z="S",col="blue",proj=list(c("time","P")))
```

`pers.rate.obj` *Calculating the global disease persistence rate in a metapopulation.*

Description

Calculating the global disease persistence rate in a metapopulation, by using the Kaplan–Meier survival curve of the metapopulation.

Usage

```
pers.rate.obj<-function(object)
```

Arguments

`object`	a seir object with the condition, this object has to have the value of the parameter 'persistence'. Because, based on this parameter, we can estimated the survival rate.

Details

For this function, we have a set of the survival time of all subpopulation in the metapopulation, due to the value of the parameter 'persistence'. Based on this value, we use the function 'survreg' of the package 'survival' to estimate the global persistence rate for a metapopulation. The function 'survival' is used for the Parametric Survival Model. Therefore, we get the estimated global persistence rate.

Value

A matrix (or vector) with columns giving lower and upper confidence limits for the estimated value. These will be labelled as (1-level)/2 and 1 - (1-level)/2 in

Author(s)

TRAN Thi Cam Giang

See Also

'pers.rate.obj' function in the 'dizzys' package.

Examples

```
#STO, STO
sto<-seir(N=10e5,type="stoch",duration=30*365,beta1=0.1,nbVilles=10)
objper<- persistence(sto)
pers.obj<-pers.rate.obj(objper)
pause()
```

`persistence` *Persistence in a Metapopulation*

Description

Generic function for calculing number of subpopulations in a metapopulation not extincts at time t.

Usage

```
persistence(object)
```

Arguments

`object`	a seir object.

Details

The function 'persistence' gives us the number of subpopulations, in a metapopualtion, in which the disease is not extinct at time t by using the parameter 'object' given.

Value

'persistence' returns a seir object with the value of the slot 'persistence'. The slot 'persistence' is a data frame. This data frame has five columns, 'time' means being the time at which there are some extinct subpopulations, at time t, we have, 'nbVilles' is the number of not extinct subpopulations before the time t, 'ndie' means being the number of died subpopulations, 'dieVille' gives us which subpopulation is died, 'remain' is the number of not extinct subpopulations after the time t.

Author(s)

TRAN Thi Cam Giang

References

Matt J. Keeling and Pejman Rohani (2008) Modeling Infectious Diseases IN HUMANS AND ANIMALS. Princeton University Press.

Examples

```
obj1<-seir.stoch(nbVilles=10,N=1e5)
objper<-persistence(obj1)
objper@persistence
```

`plot.pers` *Plotting Kaplan Meier Survival Curve*

Description

Plotting Kaplan–Meier survival curve based on persistence value or adding one or more straight lines through the current plot based on the slot 'pop' of seir object.

Usage

```
plot.pers(object, x, y, type, col, xlim, ylim, curvetype, vilabline,...)
```

Arguments

`object`	a seir object with the value of the slot "persistence".
`x, y`	the names or the positions of the columns in the slot 'pop' of the seir object. This arguments are corresponding to the x and y coordinates of the plot. In this function, we only plot lines 2D, 'x, y' are one of the names or the positions of the columns ("time","S","E","P","R","N") or (1,2,3,4,5,6)
`type`	what type of plot should be drawn. Possible types are "p", "l", "b", "c", ect.
`col`	the colors for lines. Multiple colors can be specified so that each line can be given its own color. If there are fewer colors than number of lines they are recycled in the standard fashion. Lines will all be plotted in the first colour specified.
`xlim`	the x limits of the plot.
`ylim`	the y limits of the plot.
`curvetype`	type of curve for plot. There are two value for 'curvetype'. If curvetype=="KM" (Kaplan–Meier), it means that we will plot Kaplan–Meier survival curve based on the slot 'persistence' of the object. In the other side, if curvetype=="population", it means that we will plot all lines by using the slot 'pop' of the object with straight lines.
`vilabline`	numeric. What subpopulations are chosen to add straight lines to plot, from the subpopulation set. The subpopulation set is a series of natural number from one to n, where n is the number of subpopulation in a metapopualtion of n subpopulation. Normally, 'vilabline' is NULL, we don't add any straight lines to the plot. In the other side, 'vilabline' is a good numeric, it means that 'vilabline' is in the series (1,...,n), we only can add the lines of the subpopulation in 'vilabline' to the plot.
`...`	other arguments.

Details

This function allows us to plot two types of curve to the plot. Plotting Kaplan–Meier survival curve based on persistence value or adding one or more straight lines through the current plot based on the slot 'pop' of seir object.

Author(s)

TRAN Thi Cam Giang

References

David G. Kleinbaum and Mitchel Klein, Survival Analysis.

See Also

Kaplan–Meier survival curve

Examples

```
p<-persistence(seir(type="sto",nbVilles=15,N=1e5))
plot.pers(p)
x11()
plot.pers(p,curvetype="pop",col=c("green","blue"),vilabline=c(1,3))
```

`plot.seir` *Plotting 2D/3D a seir Object*

Description

Generic function of 'seir' class for plotting 2D/3D of seir objects.

Usage

```
plot.seir(object,x="time",y=4,z=NULL,pop=c(),col="black",type="l",unitTIME=1,
proj=list(),add=F,xlim=NULL,ylim=NULL,zlim=NULL,xlab=x,ylab=y,zlab=z)
```

Arguments

`object`	seir object.
`x, y, z`	the names or the positions of the columns in the slot 'pop' of the seir object. This arguments are corresponding to the x, y and z coordinates of the plot. Normally, if 'z' is NULL, we only can add a line 2D to a plot 2D, on the other side, if 'z' is one of the names or the positions of the columns ("time","S","E","P","R","N") or (1,2,3,4,5,6), we can add a line 3D to a plot 3D.

pop	numeric. What subpopulations are chosen to add from the subpopulation set. The subpopulation set is a series of natural number from one to n, where n is the number of subpopulation in a metapopualtion of n subpopulation. Normally, 'pop' is NULL, it allows us to add all lines of the n subpopulations to the plot. In the other side, 'pop' is a good numeric, it means that 'pop' is in the series (1,...,n), we only can add the lines of the subpopulation in 'pop' to the plot.
col	the colors for lines. Multiple colors can be specified so that each line can be given its own color. If there are fewer colors than number of lines they are recycled in the standard fashion. Lines will all be plotted in the first colour specified.
type	what type of plot should be drawn. Possible types are "p", "l", "b", "c", ect.
unitTIME	the unit of time. It is 'day' corresponding to unitTIME=1, is 'week' corresponding to unitTIME=7, is 'month' corresponding to unitTIME=30 and is 'year' corresponding to unitTIME=365. Normally, unitTIME is equal to 1, however we can change it.
proj	list of the plane names. This argument is only used in the plot 3D. It allows us to add the projection of the lines 3D on the planes. It is necessary to give the value of 'proj' corresponding to the x, y and z axes.
add	adding lines of the object to a current plot, if add=T. In contrast, if add=F, we plot lines of the object to a new plot. By default, add=F.
xlim, ylim, zlim	limits to use for the coordinates.
xlab, ylab, zlab	labels for the coordinates.
...	further graphical parameters such as 'lwd', 'lty'

Details

Moreover, for the x, y, and z coordinates, the slot 'pop' of the seir object is a list of the data frames corresponding to each subpopulation in a metapopulation of n subpopulations. The each data frame often has six columns that are 'time' being a time column, at time t, 'S' being the number of susceptibles, 'E' being the number of exposed individuals, 'P' being the number of infectives, 'R' being the number of removed individuals with immunity and 'N' being the number of population, it is the sum of S, E,P,R.

In the set of arguments, there are some arguments we can give each subpopulation each value's.

S, E, I, R, N: the initial values of state variables for 'nbVilles' subpopulations. Multiple values can be specified so that each subpopulation can be given its own state variables. If there are fewer values than subpopulations they are recycled in the standard fashion. Subpopulations will all be simulated in the first value specified.

mu, beta0, beta1, sigma, gamma, phi: the parameters of simulation for 'nbVilles' subpopulations. Multiple values can be specified so that each subpopulation can be given its own parameters. If there are fewer values than subpopulations they are recycled in the standard fashion. Subpopulations will all be simulated in the first value specified.

Author(s)

TRAN Thi Cam Giang

References

Generic X-Y Plotting in R. plot.default in R.

See Also

'plot.seir' in 'dizzys' package.

Examples

```
 obj<-seir(nbVilles=3, N=5e5)
plot(obj,col=c("red","blue"),lwd=2,xlab="time (day)", ylab="number of infectives")
plot(obj,z="S",col=c("red","blue"),lwd=2,xlab="time (day)", ylab="number of infectives",zlab="number of suscepti
plot(obj,z="S",col=c("red","blue"),lwd=2,proj=list(c("time","P"),c("time","S")),box=F,xlab="time (day)", ylab="
```

`pop.seir` *Extract Values of State Variables of each City according to Time.*

Description

Generic function allowing to extract number of S, I, R individuals of each population in a metapopulation of n subpopulations according to time.

Usage

```
pop.seir(object, subset, fct,...)
```

Arguments

`object`	a seir object.
`subset`	numeric. What subpopulations are chosen to extract from the subpopulation set. The subpopulation set is a series of natural number from one to n, where n is the number of subpopulation in a metapopualtion of n subpopulation. Normally, 'subset' is NULL, it allows us to extract all n subpopulations. In the other side, 'subset' is a good numeric, it means that 'subset' is in the series (1,...,n), we only can extract the subpopulations in 'subset' argument.
`fct`	this is a function such as "sum", "mean",etc. With this function, we can do it with the extracted data from subpopulations.
`...`	other parameters.

Details

Result returned is dependent on the values of the parameter 'subset' and 'fct'.

if subset=NULL and fct=NULL, the result is the list of the 'pop' slot in the object given.

if subset=NULL and fct is a function, the result is a matrix after doing the 'fct' function on the list of the 'pop' slot in the object given.

if subset!=NULL and fct=NULL, the result is a list of the subpopulations extracted from the list of the 'pop' slot in the object given.

if subset!=NULL and fct is a function, the result is a matrix after doing the 'fct' function on the list of the subpopulations extracted in the object given.

Value

Viewing in detail in the 'detail' part.

Author(s)

TRAN Thi Cam Giang

References

Norman Matloff (2009), The Art of R Programming.

See Also

'pop.seir' fucntion in 'dizzys' package.

Examples

```
obj<-seir(nbVilles=3, N=1e7)
tpobj<-pop(obj)
class(tpobj)
tpobj<-pop(obj,fct="sum")
class(tpobj)
tpobj<-pop(obj,subset=c(1,2),fct="sum")
class(tpobj)
```

---

print.seir *Printing seir Object*

Description

Generic function allowing us to print a seir object.

Usage

```
print.seir(object, ...)
```

Arguments

object	a seir object.
...	further arguments.

Details

The function 'print' allows us to print all slots of a seir object of the 'seir' class. It contains in the following:

'pop' is a list of the values of the state variales according to time.

'duration' is time to do simulation.

'S, E, I, R' are the initial values of the state variables, number of susceptible, exposed, infected, recovered individuals, respectively.

'N' is the initial number of population.

'T' is period of the contact rate (in days).

'mu' is per capita birth and death rates (per day).

'beta0' is mean value of contact rate (per day).

'beta1' is amplitude of contact rate.

'sigma' is transition rate from exposed (E) to infected (I), per day (inverse of the average duration of the latency period.

'gamma' is recovery rate (per day).

'unitTIME' is unit of time.

'phi' is phase of the contact rate (in radians).

'nbVilles' is number of subpopulation in metapopulation.

'type' is type of simulation, 'deterministic' or 'stochastic'.

'epsilon' is proportion of infections by contacts with infected from other subpopulations.

'rho' is coupling rate between subpopulations i and j.

'seed' is seed number of random number generator.

'rng' is random number generator, 'good' or 'fast'.

'method' is simulation algorithm,'direct' or 'adaptivetau'.

'persistence' is data frame about the information of the persistence of the metapopulation.

Author(s)

TRAN Thi Cam Giang

See Also

'print.seir' function in 'dizzys' package.

Examples

```
obj<-seir(nbVilles=3, N=1e7)
print(obj)
```

seir-class *seir Class* "seir"

Description

This package permit us to simulate infectious diseases by using the SIR/SEIR models.To do that, we implement the direct algorithm of Gillespie in 1977 as described in the book 'Modeling Infectious Diseases IN HUMANS AND ANIMALS' by Keeling and Pejman Rohani (2008), and the adaptive tau leaping to approximate the trajectory of a continuous-time stochastic process as described by Cao et al. (2007) The Journal of Chemical Physics.

Objects from the Class

Objects can be created by calls of the form `new("seir", ...)`.

Slots

`pop`: list. Object of class

`duration`: numeric. time values over which to perform the numerical integration.

`S`: numeric. Initial value of the state variable 'S', this is the number of susceptible individuals at the time t0.

`E`: numeric. Initial value of the state variable 'E', this is the number of exposed individuals at the time t0.

`I`: numeric. Initial value of the state variable 'I', this is the number of infected individuals at the time t0.

`R`: numeric. Initial value of the state variable 'R', this is the number of recovered individuals at the time t0.

`N`: numeric. Initial value of the state variable 'N', this is the number of population at the time t0.

`T`: numeric. Period of the contact rate (in days).

`mu`: numeric. Per capita birth and death rates (per day).

`beta0`: numeric. Mean value of contact rate (per day).

`beta1`: numeric. Amplitude of contact rate.

`sigma`: numeric. Transition rate from exposed (E) to infected (I), per day (inverse of the average duration of the latency period).

`gamma`: numeric. Recovery rate (per day).

`unitTIME`: numeric. Unit of time.

`phi`: numeric. Phase of the contact rate (in radians).

`nbVilles`: numeric. Number of subpopulations in metapopulation.

`type`: character. Type of simulation, 'deterministic' or 'stochastic'.

`epsilon`: numeric. Proportion of infections by contacts with infected from other subpopulations. The value of epsilon between subpopulation i and j is equal to the value of epsilon between subpopulation j and i. epsilon is always greater than or equal to zero and less than or equal to one,specially, epsilon, of the subpopulation i and i, is equal to zero.

`rho`: numeric. Coupling rate between subpopulations i and j. rho between subpopulation i and j is equal to rho between subpopulation j and i. rho is always greater than or equal to zero and less than or equal to one, specially, rho, of the subpopulation i and i, is equal to one.

`seed`: numeric. Random seed for random number generator.

`rng`: character. Random number generator, 'good' or 'fast'. If we chose rng="good", it means that we use the random number generator in C++. In the other side, we chose rng="fast", it means that we use the random number generator of the professor Yann Chevaleyre at university of Paris 13. With the random generator's Yann Chevaleyre, we find that it is faster than the random generator in C++, in contrast, its accuracy is less than that of the random generator's Yann Chevaleyre.

`method`: character. Simulation algorithm,'direct' or 'adaptivetau'. If we chose method="direct", it means that we implement the direct algorithm of Gillespie in 1977. In contrast, if we chose method="adaptivetau", it means that we implement the adaptive tau-leaping approximation for simulating the trajectory of a continuous-time Markov process.

`persistence`: data.frame. Data frame about the information of the persistence of the metapopulation.

Methods

No methods defined with class "seir" in the signature.

Author(s)

TRAN Thi Cam Giang

References

Matt J. Keeling and Pejman Rohani (2008) Modeling Infectious Diseases IN HUMANS AND ANIMALS. Princeton University Press. Cao Y, Gillespie DT, Petzold LR,The Journal of Chemical Physics, 2007. Norman Matloff (2009), The Art of R Programming.

See Also

'seir' class in 'dizzys' package.

Examples

```
showClass("seir")
```

`simul` *Redoing or Continuing a Simulation.*

Description

Redoing or continuing a simulation by using values of parameters given.

Usage

```
simul(object,type="deter",continue=F,duration=5*365,method="direct",unitTIME=1,mu=1/(70*365),beta0=
T=365,phi=0,nbVilles=1,epsilon=0.0,rho=0.0,seed=as.numeric(Sys.time()),rng="good",append=TRUE,t0=NU
```

Arguments

object	a seir object. If missing this object, we create a new seir object by using the values of state variables and of parameters given.
type	type of simulation, 'deterministic' or 'stochastic'.
continue	logical (T/F). If continue=F, it means to redo silulation. Else, if continue=T, it means to continue to do silulation.
duration	time values over which to perform the numerical integration.
method	simulation algorithm,'direct' or 'adaptivetau'. If we chose method="direct", it means that we implement the direct algorithm of Gillespie in 1977. In contrast, if we chose method="adaptivetau", it means that we implement the adaptive tau-leaping approximation for simulating the trajectory of a continuous-time Markov process.
unitTIME	unit of time.
mu	per capita birth and death rates (per day).
beta0	mean value of contact rate (per day).
beta1	amplitude of contact rate.
sigma	transition rate from exposed (E) to infected (I), per day (inverse of the average duration of the latency period).
gamma	recovery rate (per day).
T	period of the contact rate (in days).
phi	phase of the contact rate (in radians).
nbVilles	number of subpopulations in the metapopulation.
epsilon	proportion of infections by contacts with infected from other subpopulations. The value of epsilon between subpopulation i and j is equal to the value of epsilon between subpopulation j and i. epsilon is always greater than or equal to zero and less than or equal to one,specially, epsilon, of the subpopulation i and i, is equal to zero.
rho	coupling rate between subpopulations i and j. rho between subpopulation i and j is equal to rho between subpopulation j and i. rho is always greater than or equal to zero and less than or equal to one, specially, rho, of the subpopulation i and i, is equal to one.
seed	random seed for random number generator.
rng	random number generator, 'good' or 'fast'. If we chose rng="good", it means that we use the random number generator in C++. In the other side, we choose rng="fast", it means that we use the random number generator of the professor Yann Chevaleyre at university of Paris 13. With the random generator's Yann Chevaleyre, we find that it is faster than the random generator in C++, in contrast, its accuracy is less than that of the random generator's Yann Chevaleyre.

S	initial value of the state variable 'S', this is the number of susceptible individuals at the time t0.
E	initial value of the state variable 'E', this is the number of exposed individuals at the time t0.
I	initial value of the state variable 'I', this is the number of infected individuals at the time t0.
R	initial value of the state variable 'R', this is the number of recovered individuals in immunity at the time t0.
N	number of population.
append	logical(T/F). It is available when continue=T. If continue=T and append=T, it means that we want our new object contains the old data + the new. If continue=T and append=F, we only get the new.
t0	numeric. It means that if we want the start time after what has already been simulated ("t0 = NULL") or if we prefer that time restarts any value (for example "t0 = 1").

Details

For this function, to get the values of arguments, first, for the arguments missed, we get its values from the seir object, in contrast, we get its given values. Moreover, in the set of arguments, there are some arguments we can give each subpopulation each value's. **S, E, I, R, N**: the initial values of state variables for 'nbVilles' subpopulations. Multiple values can be specified so that each subpopulation can be given its own state variables. If there are fewer values than subpopulations they are recycled in the standard fashion. Subpopulations will all be simulated in the first value specified. **mu, beta0, beta1, sigma, gamma, phi**: the parameters of simulation for 'nbVilles' subpopulations. Multiple values can be specified so that each subpopulation can be given its own parameters. If there are fewer values than subpopulations they are recycled in the standard fashion. Subpopulations will all be simulated in the first value specified.

Value

Result returned is an object of the 'seir' class.

Author(s)

TRAN Thi Cam Giang

See Also

'simul' function in the 'dizzys' package.

Examples

```
#STO, STO
sto<-seir(N=10e5,type="stoch",duration=5*365,beta1=0.1,nbVilles=2)
plot(simul(sto,type="stoch",continue=TRUE,duration=5*365,beta1=0,phi=c(pi/2,0)))
pause()
#DET, DET
```

```
det<-seir(N=10e5,type="deter",duration=50*365)
plot(simul(det,type="deter",continue=TRUE,duration=50*365,beta1=0,phi=pi/2))
pause()
```

`str.seir` *Describe the Structure of a seir Object.*

Description

Describe the structure of a seir object.

Usage

```
str(object, ...)
```

Arguments

`object`	a seir object.
`...`	further arguments.

Details

Compactly display the internal structure of an object, of the 'seir' class. It displays the names and the values of all parameters for each subpopulation in a metapopulation.

Value

'str.seir' does not return anything, for efficiency reasons. The obvious side effect is output to the terminal.

Author(s)

TRAN Thi Cam Giang

See Also

'str.seir' function in the 'dizzys' package.

Examples

```
obj<-seir()
str(obj)
```

summary.seir *Object Summaries*

Description

generic function used to display the internal structure of an object of 'seir' class.

Usage

```
summary.seir(object, ...)
```

Arguments

object	an objecct of the 'seir' class.
...	further arguments.

Details

Display in detail the internal structure of an object of the 'seir' class. It displays the names, the values, the signification of all parameters for each subpopulation in a metapopulation.

Value

'str.seir' does not return anything, for efficiency reasons. The obvious side effect is output to the terminal.

Author(s)

Tran Thi Cam Giang.

See Also

'summary.seir' function of the 'dizzys' package.

Examples

```
obj<-seir()
summary(obj)
```

Index

∗Topic **Kaplan–Meier curve**
plot.pers, 10
∗Topic **Kaplan–Meier estimator**
plot.pers, 10
∗Topic **R object**
str.seir, 20
∗Topic **R package**
confint.pers.rate, 4
pers.rate.obj, 8
simul, 17
∗Topic **SEIR/SIR model**
equilibrium, 5
∗Topic **classes**
seir-class, 16
∗Topic **coefficients**
coef.seir, 3
∗Topic **limit cycle equilibrium**
equilibrium, 5
∗Topic **lines**
lines.seir, 6
∗Topic **metapopulation**
persistence, 9
∗Topic **package**
dizzys-package, 2
∗Topic **persistence**
persistence, 9
∗Topic **projection on plane**
lines.seir, 6
∗Topic **seir class**
summary.seir, 21
∗Topic **seir model**
confint.pers.rate, 4
pers.rate.obj, 8
plot.seir, 11
simul, 17
str.seir, 20
∗Topic **seir object**
coef.seir, 3
∗Topic **summary**
summary.seir, 21

coef.seir, 3
confint.pers.rate, 4

dizzys *(*dizzys-package*)*, 2
dizzys-package, 2

equilibrium, 5

lines.seir, 6

pers.rate.obj, 8
persistence, 9
plot.pers, 10
plot.seir, 11
pop.seir, 13
print.seir, 14

seir-class, 16
simul, 17
str.seir, 20
summary.seir, 21