Package: GillespieSSA (via r-universe)

October 31, 2024

Type Package

Title Gillespie's Stochastic Simulation Algorithm (SSA)

Version 0.6.2

Description Provides a simple to use, intuitive, and extensible interface to several stochastic simulation algorithms for generating simulated trajectories of finite population continuous-time model. Currently it implements Gillespie's exact stochastic simulation algorithm (Direct method) and several approximate methods (Explicit tau-leap, Binomial tau-leap, and Optimized tau-leap). The package also contains a library of template models that can be run as demo models and can easily be customized and extended. Currently the following models are included, 'Decaying-Dimerization' reaction set, linear chain system, logistic growth model, 'Lotka' predator-prey model, Rosenzweig-MacArthur predator-prey model, 'Kermack-McKendrick' SIR model, and a 'metapopulation' SIRS model. Pineda-Krch et al. (2008) <doi:10.18637/jss.v025.i12>.

Depends R (>= 2.0.0)

Imports grDevices, graphics, methods, stats, utils

Suggests knitr, rmarkdown, testthat

URL https://github.com/rcannood/GillespieSSA

License GPL (>= 3)

RoxygenNote 7.1.2

Encoding UTF-8

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Repository https://rcannood.r-universe.dev

RemoteUrl https://github.com/rcannood/gillespiessa

RemoteRef HEAD

RemoteSha b532ff92067307e2438e1663a4a932f7f055a068

Contents

`GillespieSSA-package` *Gillespie Stochastic Simulation Algorithm package*

Description

Package description and overview of basic SSA theory

GillespieSSA is a versatile and extensible framework for stochastic simulation in R and provides a simple interface to a number of Monte Carlo implementations of the stochastic simulation algorithm (SSA). The methods currently implemented are: the Direct method, Explicit tau-leaping (ETL), Binomial tau-leaping (BTL), and Optimized tau-leaping (OTL). The package also provides a library of ecological, epidemiological, and evolutionary continuous-time (demo) models that can easily be customized and extended. Currently the following models are included, Decaying-Dimerization Reaction Set, Linear Chain System, single-species logistic growth model, Lotka predator-prey model, Rosenzweig-MacArthur predator-prey model, Kermack-McKendrick SIR model, and a metapopulation SIRS model.

The stochastic simulation algorithm

The stochastic simulation algorithm (SSA) is a procedure for constructing simulated trajectories of finite populations in continuous time. If $X_i(t)$ is the number of individuals in population i ($i = 1, \ldots, N$) at time t the SSA estimates the state vector $\mathbf{X}(t) \equiv (X_1(t), \ldots, X_N(t))$, given that the system initially (at time t_0) was in state $\mathbf{X}(t_0) = \mathbf{x_0}$. Reactions, single instantaneous events changing at least one of the populations (e.g. birth, death, movement, collision, predation, infection, etc), cause the state of the system to change over time. The SSA procedure samples the time τ to the next reaction R_j ($j = 1, \ldots, M$) and updates the system state $\mathbf{X}(t)$ accordingly. Each reaction R_j is characterized mathematically by two quantities; its state-change vector $\boldsymbol{\nu}_j \equiv (\nu_{1j}, \ldots, \nu_{Nj})$, where ν_{ij} is the change in the number of individuals in population i caused by one reaction of type j and its propensity function $a_j(\mathbf{x})$, where $a_j(\mathbf{x})dt$ is the probability that a particular reaction j will occur in the next infinitesimal time interval $[t, t + dt]$.

SSA implementations

There are numerous exact Monte Carlo procedures implementing the SSA. Perhaps the simplest is the Direct method of Gillespie (1977. The Direct method is an exact continuous-time numerical realization of the corresponding stochastic time-evolution equation. Because the Direct method

simulates one reaction at a time it is often, however, computationally too slow for practical applications.

Approximate implementations of the SSA sacrifices exactness for large improvements in computational efficiency. The most common technique used is tau-leaping where reaction-bundles are attempted in coarse-grained time increments τ. Speed-ups of several orders of magnitude compared to the Direct method are common. Tau-leaping must be used with care, however, as it is not as foolproof as the Direct method.

Example models

Individual demo models can be run by issuing `demo(<model name>)`, alternatively all of the demo models can be run using `demo(GillespieSSA)`. The following example models are available:

Decaying-Dimerization Reaction Set (Gillespie, 2001)
```
vignette("decaying_dimer", package = "GillespieSSA")
```

SIRS metapopulation model (Pineda-Krch, 2008)
```
vignette("epi_chain", package = "GillespieSSA")
```

Linear Chain System (Cao et al., 2004)
```
vignette("linear_chain", package = "GillespieSSA")
```

Pearl-Verhulst Logistic growth model (Kot, 2001, Pineda-Krch, 2008)
```
vignette("logistic_growth", package = "GillespieSSA")
```

Lotka predator-prey model (Gillespie, 1977; Kot, 2001)
```
vignette("lotka_predator_prey", package = "GillespieSSA")
```

Radioactive decay model (Gillespie, 1977)
```
vignette("radioactive_decay", package = "GillespieSSA")
```

Rosenzweig-MacArthur predator-prey model (Pineda-Krch et al., 2007, Pineda-Krch, 2008)
```
vignette("rm_predator_prey", package = "GillespieSSA")
```

Kermack-McKendrick SIR model (Brown & Rothery, 1993)
```
vignette("sir", package = "GillespieSSA")
```

Acknowledgements

- Heinrich zu Dohna for many caffein induced discussions on the package and reference manual, and for providing comments on the vignette documentation.
- Ben Bolker for comments on the initial release of the package and for providing a hint for how to more elegantly handle model parameters as arguments to the ssa() function.
- Josh Obrien for copy editing and feedback on the JSS manuscript.
- Thomas Petzoldt for comments on the package, the JSS manuscript and for preparing version 0.5-4.
- Three anonymous referees whose comments substantially improved some of the functionality.

References

- Brown D. and Rothery P. 1993. Models in biology: mathematics, statistics, and computing. John Wiley & Sons.
- Cao Y., Li H., and Petzold L. 2004. Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. J. Chem. Phys. 121:4059-4067. doi:10.1063/1.1778376
- Cao Y., Gillespie D.T., and Petzold L.R. 2006. Efficient step size selection for the tau-leaping method. J. Chem. Phys. 124:044109. doi:10.1063/1.2159468
- Cao Y., Gillespie D.T., and Petzold L.R. 2007. Adaptive explicit tau-leap method with automatic tau selection. J. Chem. Phys. 126:224101. doi:10.1063/1.2745299
- Chatterjee A., Vlachos D.G., and Katsoulakis M.A. 2005. Binomial distribution based tau-leap accelerated stochastic simulation. J. Chem. Phys. 122:024112. doi:10.1063/1.1833357
- Gillespie D.T. 1977. Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem. 81:2340. doi:10.1021/j100540a008
- Gillespie D.T. 2001. Approximate accelerated stochastic simulation of chemically reacting systems. J. Chem. Phys. 115:1716-1733. doi:10.1063/1.1378322
- Gillespie D.T. 2007. Stochastic simulation of chemical kinetics. Annu. Rev. Chem. 58:35 doi:10.1146/annurev.physchem.58.032806.104637
- Kot M. 2001. Elements of mathematical ecology. Cambridge University Press. doi:10.1017/CBO9780511608520
- Pineda-Krch M. 2008. Implementing the stochastic simulation algorithm in R. Journal of Statistical Software 25(12): 1-18. doi:10.18637/jss.v025.i12
- Pineda-Krch M., Blok H.J., Dieckmann U., and Doebeli M. 2007. A tale of two cycles — distinguishing quasi-cycles and limit cycles in finite predator-prey populations. Oikos 116:53-64. doi:10.1111/j.2006.00301299.14940.x

See Also

`ssa()`, `ssa.d()`, `ssa.etl()`, `ssa.btl()`, `ssa.otl()`, `ssa.plot()`

`ssa` *Invoking the stochastic simulation algorithm*

Description

Main interface function to the implemented SSA methods. Runs a single realization of a predefined system.

Usage

```
ssa(
                    x0,            # initial state vector
                     a,            # propensity vector
                    nu,            # state-change matrix
                 parms = NULL,     # model parameters
                    tf,            # final time
                method = ssa.d(),  # SSA method
               simName = "",
                   tau = 0.3,      # deprecated
                     f = 10,       # deprecated
               epsilon = 0.03,     # deprecated
                    nc = 10,       # deprecated
                   hor = NA_real_, # deprecated
                   dtf = 10,       # deprecated
                    nd = 100,      # deprecated
  ignoreNegativeState = TRUE,
      consoleInterval = 0,
       censusInterval = 0,
              verbose = FALSE,
          maxWallTime = Inf
)
```

Arguments

x0	numerical vector of initial states where the component elements must be named using the same notation as the corresponding state variable in the propensity vector, a.
a	character vector of propensity functions where state variables correspond to the names of the elements in x0.
nu	numerical matrix of change if the number of individuals in each state (rows) caused by a single reaction of any given type (columns).
parms	named vector of model parameters.
tf	final time.
method	an SSA method, the valid options are: • ssa.d() — Direct method (default method), • ssa.etl() - Explicit tau-leap, • ssa.btl() — Binomial tau-leap, or • ssa.otl() — Optimized tau-leap.
simName	optional text string providing an arbitrary name/label for the simulation.
tau	[DEPRECATED], see ssa.etl()
f	[DEPRECATED], see ssa.btl()
epsilon	[DEPRECATED], see ssa.otl()
nc	[DEPRECATED], see ssa.otl()

`hor` [DEPRECATED], see `ssa.otl()`

`dtf` [DEPRECATED], see `ssa.otl()`

`nd` [DEPRECATED], see `ssa.otl()`

`ignoreNegativeState`
: boolean object indicating if negative state values should be ignored (this can occur in the `etl` method). If `ignoreNegativeState=TRUE` the simulation finishes gracefully when encountering a negative population size (i.e. does not throw an error). If `ignoreNegativeState=FALSE` the simulation stops with an error message when encountering a negative population size.

`consoleInterval`
: (approximate) interval at which `ssa` produces simulation status output on the console (assumes `verbose=TRUE`). If `consoleInterval=0` console output is generated each time step (or tau-leap). If `consoleInterval=Inf` no console output is generated. Note, `verbose=FALSE` disables all console output. **Console output drastically slows down simulations.**

`censusInterval` (approximate) interval between recording the state of the system. If `censusInterval=0` (t, x) is recorded at each time step (or tau-leap). If `censusInterval=Inf` only (t_0, x_0) and (t_f, x_t) is recorded. Note, the size of the time step (or tau-leaps) ultimately limits the interval between subsequent recordings of the system state since the state of the system cannot be recorded at a finer time interval the size of the time steps (or tau-leaps).

`verbose` boolean object indicating if the status of the simulation simulation should be displayed on the console. If `verbose=TRUE` the elapsed wall time and (t, x) is displayed on the console every `consoleInterval` time step and a brief summary is displayed at the end of the simulation. If `verbose=FALSE` the simulation runs *entirely* silent (overriding `consoleInterval`). **Verbose runs drastically slows down simulations.**

`maxWallTime` maximum wall time duration (in seconds) that the simulation is allowed to run for before terminated. This option is useful, in particular, for systems that can end up growing uncontrolably.

Details

Although `ssa` can be invoked by only specifying the system arguments (initial state vector `x0`, propensity vector `a`, state-change matrix `nu`), the final time (`tf`), and the `SSA` method to use, substantial improvements in speed and accuracy can be obtained by adjusting the additional (and optional) `ssa` arguments. By default `ssa` (tries to) use conservative default values for the these arguments, prioritizing computational accuracy over computational speed. These default values are, however, **not** fool proof for the approximate methods, and occasionally one will have to hand tweak them in order for a stochastic model to run appropriately.

Value

Returns a list object with the following elements,

- `data`: a numerical matrix object of the simulation time series where the first column is the time vector and subsequent columns are the state frequencies.

- stats: sub-list object with elements containing various simulation statistics. The of the sub-list are:
- stats$startWallTime: start wall clock time (YYYY-mm-dd HH:MM:SS).
- stats$endWallTime: end wall clock time (YYYY-mm-dd HH:MM:SS).
- stats$elapsedWallTime: elapsed wall time in seconds.
- stats$terminationStatus: string vector listing the reason(s) for the termination of the realization in 'plain words'. The possible termination statuses are:
  - finalTime = if the simulation reached the maximum simulation time tf,
  - extinction = if the population size of all states is zero,
  - negativeState = if one or several states have a negative population size (can occur in the ETL method),
  - zeroProp = if all the states have a zero propensity function,
  - maxWallTime = if the maximum wall time has been reached. Note the termination status may have more than one message.
- 'stats$nSteps" total number of time steps (or tau-leaps) executed.
- stats$meanStepSize: mean step (or tau-leap) size.
- stats$sdStepSize: one standard deviation of the step (or tau-leap) size.
- stats$SuspendedTauLeaps: number of steps performed using the Direct method due to OTL suspension (only applicable for the OTL method).
- arg$...: sub-list with elements containing all the arguments and their values used to invoke ssa (see Usage and Arguments list above).

Preparing a run

In order to invoke SSA the stochastic model needs at least four components, the initial state vector (x0), state-change matrix (nu), propensity vector (a), and the final time of the simulation (tf). The initial state vector defines the population sizes in all the states at $t = 0$, e.g. for a system with two species X1 and X2 where both have an initial population size of 1000 the initial state vector is defined as `x0 <- c(X1=1000,X2=1000)`. The elements of the vector have to be labelled using the same notation as the state variables used in the propensity functions. The state-change matrix defines the change in the number of individuals in each state (rows) as caused by one reaction of a given type (columns). For example, the state-change matrix for system with the species S_1 and S_2 with two reactions

$$S_1 \xrightarrow{c_1} S_2$$

$$S_2 \xrightarrow{c_2} 0$$

is defined as `nu <- matrix(c(-1,0,+1,-1),nrow=2,byrow=TRUE)` where c_1 and c_2 are the per capita reaction probabilities. The propensity vector, a, defines the probabilities that a particular reaction will occur over the next infinitesimal time interval $[t, t + dt]$. For example, in the previous example the propensity vector is defined as `a <- c("c1*X1","c2*X2")`. The propensity vector consists of character elements of each reaction's propensity function where each state variable requires the corresponding named element label in the initial state vector (x0).

Example

Irreversible isomerization: Perhaps the simplest model that can be formulated using the SSA is the irreversible isomerization (or radioactive decay) model. This model is often used as a first pedagogic example to illustrate the SSA (see e.g. Gillespie 1977). The deterministic formulation of this model is

$$\frac{dX}{dt} = -cX$$

where the single reaction channel is

$$S \xrightarrow{c} 0$$

By setting $X_0 = 1000$ and $c = 0.5$ it is now simple to define this model and run it for 10 time steps using the Direct method,

```
out <- ssa(x0=c(X=1000),a=c("c*X"),nu=matrix(-1),parms=c(c=0.5),tf=10)
```

The resulting time series can then be displayed by,

```
ssa.plot(out)
```

Note

Selecting the appropriate SSA method is a trade-off between computational speed, accuracy of the results, and which SSA actually works for a given scenario. This depends on the characteristics of the defined system (e.g. number of reaction channels, number of species, and the absolute and relative magnitude of the propensity functions). **Not all methods are appropriate for all models.** When selecting a SSA method all of these factors have to be taken into consideration. The various tau-leap methods accept a number of additional arguments. While the default values of these arguments may work for some scenarios they may have to be adjusted for others. The default values for the tau-leap methods are conservative in terms of computational speed and substantial increase in efficiency may be gained by optimizing their values for a specific system.

See Also

GillespieSSA-package, `ssa.d()`, `ssa.etl()`, `ssa.btl()`, `ssa.otl()`

Examples

```
## Irreversible isomerization
## Large initial population size (X=1000)
parms <- c(c=0.5)
x0  <- c(X=10000)
a   <- c("c*X")
nu  <- matrix(-1)
out <- ssa(x0,a,nu,parms,tf=10,method=ssa.d(),simName="Irreversible isomerization") # Direct method
plot(out$data[,1],out$data[,2]/10000,col="red",cex=0.5,pch=19)

## Smaller initial population size (X=100)
```

```
x0  <- c(X=100)
out <- ssa(x0,a,nu,parms,tf=10,method=ssa.d()) # Direct method
points(out$data[,1],out$data[,2]/100,col="green",cex=0.5,pch=19)

## Small initial population size (X=10)
x0  <- c(X=10)
out <- ssa(x0,a,nu,parms,tf=10,method=ssa.d()) # Direct method
points(out$data[,1],out$data[,2]/10,col="blue",cex=0.5,pch=19)

## Logistic growth
parms <- c(b=2, d=1, K=1000)
x0  <- c(N=500)
a   <- c("b*N", "(d+(b-d)*N/K)*N")
nu  <- matrix(c(+1,-1),ncol=2)
out <- ssa(x0,a,nu,parms,tf=10,method=ssa.d(),maxWallTime=5,simName="Logistic growth")
ssa.plot(out)

## Kermack-McKendrick SIR model
parms <- c(beta=0.001, gamma=0.1)
x0  <- c(S=499,I=1,R=0)
a   <- c("beta*S*I","gamma*I")
nu  <- matrix(c(-1,0,+1,-1,0,+1),nrow=3,byrow=TRUE)
out <- ssa(x0,a,nu,parms,tf=100,method=ssa.d(),simName="SIR model")
ssa.plot(out)

## Lotka predator-prey model
parms <- c(c1=10, c2=.01, c3=10)
x0  <- c(Y1=1000,Y2=1000)
a   <- c("c1*Y1","c2*Y1*Y2","c3*Y2")
nu  <- matrix(c(+1,-1,0,0,+1,-1),nrow=2,byrow=TRUE)
out <- ssa(x0,a,nu,parms,tf=100,method=ssa.etl(),simName="Lotka predator-prey model")
ssa.plot(out)
```

`ssa.btl` *Binomial tau-leap method (BTL)*

Description

Binomial tau-leap method implementation of the SSA as described by Chatterjee et al. (2005). Should be passed as `method` argument for `ssa()`.

Usage

```
ssa.btl(f = 10)
```

Arguments

`f`	coarse-graining factor (see page 4 in Chatterjee et al. 2005).

Details

Performs one time step using the Binomial tau-leap method. Intended to be invoked by `ssa()`.

References

Chatterjee et al. (2005)

See Also

GillespieSSA-package, `ssa()`

Examples

```
ssa.btl(f = 40)
```

`ssa.d` *Direct method (D)*

Description

Direct method implementation of the SSA as described by Gillespie (1977). Should be passed as `method` argument for `ssa()`.

Usage

```
ssa.d()
```

References

Gillespie (1977)

See Also

GillespieSSA-package, `ssa()`

Examples

```
ssa.d()
```

ssa.etl *Explicit tau-leap method (ETL)*

Description

Explicit tau-leap method implementation of the SSA as described by Gillespie (2001). Should be passed as `method` argument for `ssa()`.

Usage

```
ssa.etl(tau = 0.3)
```

Arguments

`tau`	tau-leap.

Details

Performs one time step using the Explicit tau-leap method. Intended to be invoked by `ssa()`.

References

Gillespie (2001)

See Also

GillespieSSA-package, `ssa()`

Examples

```
ssa.etl(tau = .1)
```

ssa.otl *Optimized tau-leap method (OTL)*

Description

Optimized tau-leap method implementation of the SSA as described by Cao et al. (2006). Should be passed as `method` argument for `ssa()`.

Usage

```
ssa.otl(epsilon = 0.03, nc = 10, hor = NA_real_, dtf = 10, nd = 100)
```

Arguments

epsilon	error control parameter.
nc	number of critical reactions threshold parameter.
hor	Highest order reaction vector. There must be one entry per species in x. Must be one of 1: first-order, 2: second-order or 22: homo-dimer. If hor is NA, defaults are all second-order.
dtf	Direct method threshold factor for temporarily suspending the OTL method.
nd	number of Direct method steps to perform during an OTL suspension.

Note

Third order-reactions ($S_1 + S_2 + S_3 \rightarrow \ldots$) are not supported currently since they are approximations to sets of coupled first- and second-order reactions). See Cao et al. (2006) for more details.

References

Cao et al. (2006)

See Also

GillespieSSA-package, ssa()

Examples

```
ssa.otl(
  hor = 1,
  nc = 10,
  epsilon = .03,
  dtf = 10,
  nd = 100
)
```

---

ssa.plot *Simple plotting of ssa output*

Description

Provides basic functionally for simple and quick time series plot of simulation output from ssa().

Usage

```
ssa.plot(
  out = stop("requires simulation output object"),
  file = "ssaplot",
  by = 1,
  plot.from = 2,
  plot.to = ncol(out$data),
```

```
  plot.by = 1,
  show.title = TRUE,
  show.legend = TRUE
)
```

Arguments

out	data object returned from ssa().
file	name of the output file (only applicable if plot.device!="x11".
by	time increment in the plotted time series
plot.from	first population to plot the time series for (see note)
plot.to	last population to plot the time series for (see note)
plot.by	increment in the sequence of populations to plot the time series for (see note)
show.title	boolean object indicating if the plot should display a title
show.legend	boolean object indicating if the legend is displayed

Note

The options by, plot.from, plot.to, and plot.by can be used to plot a sparser sequence of data points. To plot the population sizes using a larger time interval the by option can be set, e.g. to plot only every 10th time point by=10. To plot only specific populations the plot.from, plot.to, and plot.by options can be set to subset the state vector. Note that the indexing of the populations is based on the $(t, \mathbf{X})$ vector, i.e. the first column is the time vector while the first population is index by 2 and the last population by $N + 1$. Display of a plot title above the plot and legend is optional (and are set with the arguments show.title and show.legend. Above the plot panel miscellaneous information for the simulation are displayed, i.e. method, elapsed wall time, number of time steps executed, and the number of time steps per data point.

See Also

GillespieSSA-package, ssa()

Examples

```
## Define the Kermack-McKendrick SIR model and run once using the Direct method
parms <- c(beta=.001, gamma=.100)
x0 <- c(S=500, I=1, R=0)                          # Initial state vector
nu <- matrix(c(-1,0,1,-1,0,1),nrow=3,byrow=TRUE) # State-change matrix
a  <- c("beta*S*I", "gamma*I")                    # Propensity vector
tf <- 100                                         # Final time
simName <- "Kermack-McKendrick SIR"
out <- ssa(x0,a,nu,parms,tf,method="D",simName,verbose=TRUE,consoleInterval=1)

## Basic ssa plot
ssa.plot(out)

# Plot only the infectious class
ssa.plot(out,plot.from=3,plot.to=3)
```

Index