`adaptivetau`: efficient stochastic simulations in R

Philip Johnson

Abstract

Stochastic processes underlie all of biology, from the large-scale processes of evolution to the fine-scale processes of biochemical interactions. Consequently, the analysis of biological data frequently necessitates the use of Markov models. While these models sometimes yield analytic solutions, simulations can provide intuition when analytic results are intractable. This vignette presents a few examples of `adaptivetau` applied to biological problems.

Introduction

Fundamentally, stochastic effects become critical to take into account when the quantities in a system are small enough that extinction becomes probable. This vignette presents three such biological systems:

1. (ecology) Lotka-Volterra predator-prey dynamics predict oscillations in the population sizes of predator and prey species for certain parameters [Lotka, 1920]. However, when the population of either species nears zero, chance effects can lead to extinction of that species and thus a halt to the oscillations.

2. (evolution) When a new mutation arises in a population of size N, it will eventually be either lost (reaching 0% frequency) or fix (reaching 100% frequency). If the allele is selectively neutral, then with probability $(N-1)/N$, the stochastic process of genetic drift will lead to the new allele disappearing from the population instead of reaching fixation [Kimura, 1957].

3. (epidemiology) In general, emerging zoonotic infections are poorly adapted to humans and thus unlikely to be transmitted from the initial infected person to significantly more people. However, continued contact between humans and an animal reservoir will give the pathogen

more chances to become human adapted, leading to a larger epidemic [Lloyd-Smith et˜al., 2009].

Crucially, these stochastic processes are all Markovian — the future state of the process depends only on the present state (i.e., it is independent of the history).

Here, we focus on simulating trajectories from a continuous-time Markov process for which the transition rates are not a function of time (i.e. time-homogeneous). The straightforward (and exact) method of simulating such a process requires many iterations of 1) drawing a random exponentially-distributed waiting time until the next transition and 2) randomly selecting the identity of the transition, weighted by the relative transition rates. This solution, which is known as the Gillespie algorithm in the chemical physics literature [Gillespie, 1976], becomes impractical as any one transition rate grows large. Instead, models with larger transition rates require an approximate approach that increases simulation speed while still maintaining reasonable accuracy. One way to perform this approximation is to reduce the number of iterations by treating transition rates as constant over time periods for which this approximation leads to little error.

The `adaptivetau` package in R implements both an exact solution and an approximate solution known as the "adaptive tau-leaping algorithm" [Cao et˜al., 2007]. Similar functionality exists in the publicly-available `GillespieSSA` R package [Pineda-Krch, 2008]; however, our new implementation is much faster, due in part to its underlying implementation in C++.

Methods

IMPORTANT: This section can be skipped if you just want to learn to use the package, but the approximations involved in the adaptive tau-leaping algorithm should be understood before interpreting simulation results!

First we define our notation and use it to describe the type of model simulated by this package. Then we provide basic intuition for the adaptive tau-leaping algorithm, leaving a more detailed description to Cao et˜al. [2007].

Let the state of the Markov process at time t, $X(t)$, be completely described by a vector of $n \geq 1$ random variables $X(t) := [X_1(t), \ldots, X_n(t)]$, each of which is defined on the non-negative integers. Transitions between states occur according to the instantaneous rate matrix, Q. In general, this matrix will be extremely sparse, so we do not attempt to use it directly. Instead, for each allowable transition, j, we define a rate, λ_j, and a vector

of n integers, $\Delta_j := [\delta_{j,1}, \ldots, \delta_{j,n}]$, that reflects the change in state if this transition were followed: $X(t) + \Delta_j$. An arbitrary number of transitions may exist: $j \in \mathbb{Z}$. Because we are modeling a time-homogeneous process, a transition rate λ_j cannot depend on t, although it may depend on the current state, $X(t)$. For a given state $X(t)$, transitions that would lead to any of the n state variables becoming negative must have rate 0.

Thus the stochastic model is completely defined by a vector giving the initial state ($X(0)$), a set of allowable transitions ($\{\Delta_j\}$) and a function to calculate transition rates given the state ($\lambda(X)$).

Given a model, the package simulates a trajectory from time 0 to a user-supplied stopping time, t_f. Intuitively, the algorithm identifies time periods during which all transition rates will remain approximately constant and all n state variables will remain greater than zero with probability ~ 1. Then the simulator can "leap" over such a period of length τ and, for each transition, add the net effect of the Poisson-distributed number of transitions that should have occurred during this period: $X(t + \tau) \approx X(t) + \sum_j y_j \Delta_j$ where $y_j \sim \text{Poisson}(\tau \lambda_j)$. The challenge arises in handling models for which transition rates frequently change and in balancing efficiency with accuracy when selecting these time periods to leap over. Here we implement the algorithm of Cao et˜al. [2007].

Example 1: ecology

Consider a simple Lotka-Volterra ecological model in which a prey species (X_1) interacts with a predator species (X_2). Traditionally, this would be written using two differential equations:

$$dX_1/dt = rX_1 - \alpha X_1 X_2$$
$$dX_2/dt = \beta X_1 X_2 - \delta X_2$$

However, these equations ignore stochastic variance and the possibility of species extinction. We can reformulate them in a stochastic model with state variables $X(t) = [X_1(t), X_2(t)]$ and three transitions:

1. growth of prey: $\Delta_1 = [+1, 0]$ with rate $\lambda_1 = rX_1$
2. predation: $\Delta_2 = [-2, +1]$ with rate $\lambda_2 = \beta X_1 X_2$ (effectively, $\alpha = 2\beta$)
3. death of predators: $\Delta_3 = [0, -1]$ with rate $\lambda_3 = \delta X_2$

Now we can write this model in R for use with

```
> library(adaptivetau)
```

First we specify our transitions (Δ):

```
> transitions = cbind(c( 1, 0), # prey grow (+1 prey, no change predator)
+                     c(-2, 1), # predation (-2 prey, +1 predator)
+                     c( 0,-1)) # predator dies (no change prey, -1 predator)
```

Next we write a function to calculate the rates of each transition (λ):

```
> lvRateF <- function(x, params, t) {
+   return(c(params$r * x["prey"],               # rate of prey growing
+            params$beta * x["prey"]*x["pred"] * # rate of predation
+              (x["pred"] >= 2),
+            params$delta * x["pred"]))          # rate of predators dying
+ }
```

Note that the conditional `(x["pred"] >= 2)` above ensures that the predation transition can never create a negative number of prey (since 2 are subtracted). Finally we run the simulations setting initial conditions (`init.values`), parameters (`params`), and the simulation stop time (`tf`):

```
> set.seed(4) # set random number generator seed to be reproducible
> simResults = ssa.adaptivetau(init.values = c(prey = 1000, pred = 500),
+                              transitions, lvRateF,
+                              params = list(r=10, beta=0.01, delta=10),
+                              tf=12)
```

We plot the results in Figure 1 using the following code:

```
> matplot(simResults[,"time"], simResults[,c("prey","pred")], type='l',
+         xlab='Time', ylab='Counts (log scale)', log='y')
> legend("bottomleft", legend=c("prey", "predator"), lty=1:2, col=1:2)
```

Exact vs approximate algorithms

The above code uses the adaptive tau-leaping approximation for simulation. An exact simulation requires more computational time, but captures more fine-scale variance, as you can see by running another simulation using the `ssa.exact` command:

```
> simResults = ssa.exact(init.values = c(prey = 1000, pred = 500),
+                        transitions, lvRates,
+                        params = list(r=10, beta=0.01, delta=10),
+                        tf=12)
```

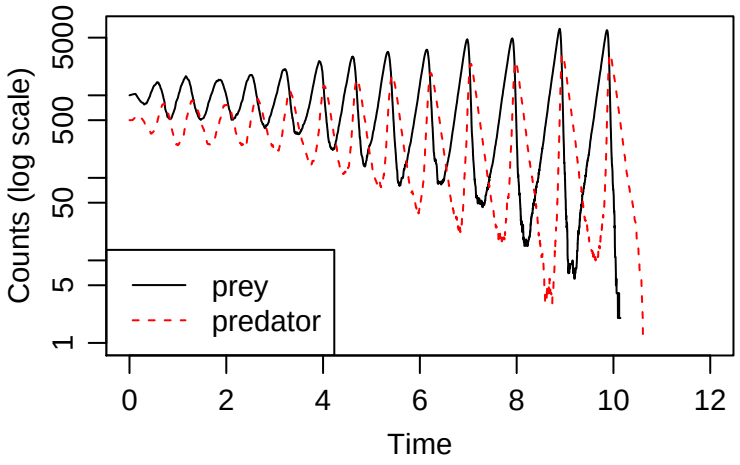

Figure 1: Simulated trajectory from the Lotka-Voltera model under the adaptive tau-leaping approximation.

Alternatively, you can improve the approximation by reducing the ϵ variable in the tau-leaping parameters (`tl.params`) when calling `ssa.adaptivetau` (in the earlier example, ϵ takes its default value of 0.05):

```
> simResults = ssa.adaptivetau(init.values = c(prey = 1000, pred = 500),
+                              transitions, lvRates,
+                              params = list(r=10, beta=0.01, delta=10),
+                              tf=12,
+                              tl.params = list(epsilon = 0.005))
```

Note we have explicitly linked the death of prey to growth in the predator population by placing these two actions in a single transition and incorporating the parameter α into the Δ_2 vector. Alternatively, these actions could have been split into two independent transitions with state change vectors $[-1, 0]$ and $[0, +1]$ and rates $\alpha X_1 X_2$ and $\beta X_1 X_2$. This formulation would yield slightly increased variance in the stochastic process since these actions are uncoupled.

With the parameters and initial conditions used in this example, the deterministic model predicts a steady state with $dX_1/dt = dX_2/dt = 0$, while our stochastic trajectory has a dramatically different outcome with prey becoming extinct around time 11. If we repeat the simulation many times, we find the prey go extinct in approximately 1/3 of the simulations (followed shortly by extinction of the predator) while the predator become

extinct in the other 2/3 of the simulations (and prey grow infinitely, revealing a fundamental unrealistic aspect of this model).

Example 2: genetic drift

Consider a Moran model for the evolution of a population of size N in which we track the number of individuals with a novel mutant allele (X) versus the number of individuals with the ancestral allele ($N - X$). Since the population size is a constant, this model has only one independent variable (X). Under the Moran model, evolution occurs when one individual is chosen to reproduce and, simultaneously, one individual is chosen to die. Thus the model allows only "one-step" transitions in which X either decreases or increases by one ($\Delta_1 = [-1]$ and $\Delta_2 = [+1]$). Both transitions occur at the same rate: $\lambda_1 = \lambda_2 = \frac{X}{N}\frac{N-X}{N}$.

Although this model has only one variable, we have two "critical values" for this variable: $X = 0$ and $X = N$ are both absorbing boundaries. The adaptive tau-leaping approximation always careful near 0 (the $X = 0$ boundary), but it is unaware of other critical values. We work around this problem by creating an additional variable for the ancestral allele, such that $X_2 = 0$ when $X_1 = N$.

```
> library(adaptivetau)
> transitions = cbind(c(-1,+1), # num individuals w/ derived allele decreases
+                     c(+1,-1)) # num individuals w/ derived allele increases
> driftRateF <- function(x, params, t) {
+   rep(x[1] * x[2]/sum(x)^2, 2)
+ }
> set.seed(1) # set random number generator seed to be reproducible
> r=ssa.adaptivetau(c(500,500), transitions, driftRateF, params=NULL, tf=Inf)
```

Simulation results are plotted in Figure 2. The above code also demonstrates an alternative stopping condition for the simulation: when `tf=Inf`, the simulation will run until all transition rates are 0 (in this case, when either allele fixes in the population).

Example 3: epidemiology

We want to model the stuttering transmission of an infectious disease in the initial stages of its movement from a non-human animal reservoir to a human

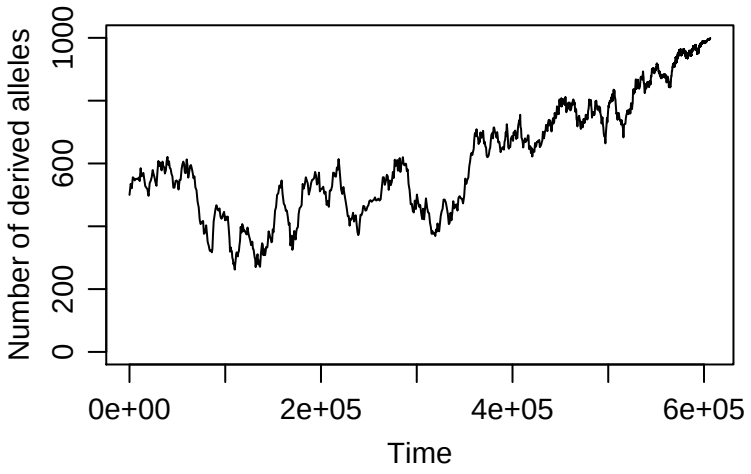

Figure 2: Genetic drift starting at frequency of 50% in a population of size 1000.

population. Our model is a slight modification on the standard Kermack-McKendrick SIR compartment model: susceptible (S), infected with non-human-adapted pathogen (I_1), infected with human-adapted pathogen (I_2), and recovered (this last class is also immune) (R).

First, the differential equation representation:

$$\begin{aligned} dS/dt &= -S(z + \beta_1 I_1 + \beta_1 \mu I_1 + \beta_2 I_2) \\ dI_1/dt &= S(z + \beta_1 I_1) - I_1(\delta_1 + \gamma_1) \\ dI_2/dt &= S(\beta_1 \mu I_1 + \beta_2 I_2) - I_2(\delta_2 + \gamma_2) \\ dR/dt &= \gamma_1 I_1 + \gamma_2 I_2 \end{aligned}$$

where we assume transmissible contact with infected animals occurs at a constant rate z, human-to-human transmissible contact occurs with rates $\beta_{1,2}$, human adaptation occurs with rate μ, death due to infection with rates $\delta_{1,2}$ and recovery with rates $\gamma_{1,2}$.

Now we will make a stochastic version of this model in R. Since this model has a few more variable than our earlier examples, we will use the `ssa.maketrans` helper function which eases creation of the transition matrix:

```
> library(adaptivetau)
> init.values = c(
+   S = 10^5, # susceptible humans
```

```
+   I1 = 0,   # infected humans
+   I2 = 0,   # infected humans
+   R = 0)    # recovered (and immune) humans
> transitions = ssa.maketrans(names(init.values),
+   rbind('S', -1, 'I1', +1), # infection (animal-adapted strain)
+   rbind('S', -1, 'I2', +1), # infection (human-adapted strain)
+   rbind('I1', -1),          # death due to infection
+   rbind('I2', -1),
+   rbind('I1', -1, 'R', +1), # recovery
+   rbind('I2', -1, 'R', +1)
+   )
> SIRrateF <- function(x, p, t) {
+   return(c(x["S"] * (p$zoonotic + p$beta[1]*x["I1"]), # infection rate
+            x["S"] * (p$beta[2]*x["I2"] + p$mu*p$beta[1]*x["I1"]),
+            params$delta[1]*x["I1"], # infected death rate
+            params$delta[2]*x["I2"],
+            params$gamma[1]*x["I1"], # recovery rate
+            params$gamma[2]*x["I2"]))
+ }
> params = list(zoonotic=1e-6, beta=c(1e-7, 1e-5), mu=0.1,
+               delta=c(1e-2,1e-5), gamma=c(0.1, 0.1))
> set.seed(3) # set random number generator seed to be reproducible
> r=ssa.adaptivetau(init.values, transitions, SIRrateF, params, tf=1000)
```

Output from this simulation appears in Figure 3.

Conclusion

While the above examples are simple, the `adaptivetau` package maintains quick runtimes on models with hundreds of variables and thousands of transitions. Due to the approximations inherent in the adaptive tau-leaping algorithm, the precise runtime depends strongly on the exact nature of the model structure and parameters. However, this hybrid R/C++ implementation appears to be approximately 50 times faster than the pure R implementation in the `GillespieSSA` package.

Acknowledgment

Funding: This work supported in part by National Science Foundation DBI-0906041 to Philip Johnson and National Institutes of Health R01-

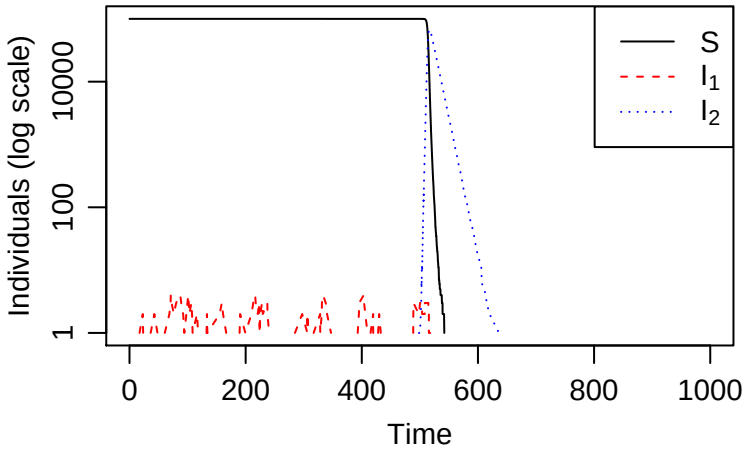

Figure 3: Simulated trajectory from SIIR model of stuttering zoonotic transmission. Zoonotic transmission occurs many times (red dashed line) before the pathogen becomes human-adapted and creates a serious epidemic (blue dotted line).

AI049334 to Rustom Antia.

References

Y.˜Cao, D.˜T. Gillespie, and L.˜R. Petzold. Adaptive explicit-implicit tau-leaping method with automatic tau selection. *J Chem Phys*, 126 (22):224101, 2007. URL `http://eutils.ncbi.nlm.nih.gov/entrez/eutils/elink.fcgi?cmd=prlinks&dbfrom=pubmed&retmode=ref&id=17581038`.

Daniel˜T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J Comput Phys*, 22(4):403 – 434, 1976. ISSN 0021-9991. doi: DOI:10.1016/0021-9991(76)90041-3. URL `http://www.sciencedirect.com/science/article/B6WHY-4DD1NC9-CP/2/43ade5f11fb949602b3a2abdbbb29f0e`.

M.˜Kimura. Some problems of stochastic processes in genetics. *Ann Math Stat*, 28(4):882–901, 1957.

J.˜O. Lloyd-Smith, D.˜George, K.˜M. Pepin, V.˜E. Pitzer, J.˜R. Pulliam, A.˜P. Dobson, P.˜J. Hudson, and B.˜T. Grenfell. Epidemic dynam-

ics at the human-animal interface. *Science*, 326(5958):1362–7, 2009. URL http://eutils.ncbi.nlm.nih.gov/entrez/eutils/elink.fcgi?cmd=prlinks&dbfrom=pubmed&retmode=ref&id=19965751.

A.˜J. Lotka. Analytical note on certain rhythmic relations in organic systems. *Proc Natl Acad Sci U S A*, 6(7):410–5, 1920. URL http://eutils.ncbi.nlm.nih.gov/entrez/eutils/elink.fcgi?cmd=prlinks&dbfrom=pubmed&retmode=ref&id=16576509.

Mario Pineda-Krch. GillespieSSA: implementing the Gillespie stochastic simulation algorithm in R. *Journal of Statistical Software*, 25(12):1–18, 4 2008. ISSN 1548-7660. URL http://www.jstatsoft.org/v25/i12.