**CBS810 Lab 5: Simulate Stochastic Models Using Gillespie’s Algorithm**

Shi Chen schen31@ncsu.edu

**Learning objectives:**

**Understand the advantages of stochastic models**

**Know the idea and steps of Gillespie’s algorithm**

**Implement the Gillespie’s algorithm for stochastic models**

Event-driven approach for demographic stochasticity: explicit consideration of events in the system

**The core idea of Gillespie’s algorithm:**

**WHEN** does the next event happen?

**WHAT** is the next event?

SIS model: most simple case:

**S-> I** (infection) and **I->S** (recovery)

|  |  |  |
| --- | --- | --- |
| **Event** | **Transition** | **Associated Rate** |
| Infection | (S,I)→(??) |  |
| Recovery | (S,I)→(??) |  |

**The recipe for the Gillespie (as simple as 1-2-3!)**

**1. Calculate the TOTAL rate of ALL events (**+**)**

**2. Simulate the time until the next events – rate is reciprocal of time!**

**3. Simulate WHAT event actually occurs**

****

p1= / (+)

p2=/ (+)

If Unifis between 0 and P1, infection occurs

If Unifis between P1 and P1 + P2 (=1), recovery occurs

**Q:** What if we have more than 2 events?

**The sample code for SIS model**

# The Gillespie function

gillesp <- function(start,ratefun,trans,pars,times) {

t0 <- times[1] ## set time to starting time

ntimes <- length(times) ## total time duration

X <- start ## set state to starting state

res <- matrix(nrow=length(times),ncol=length(start),dimnames=list(times,names(start)))

## matrix for results

for (ctr in 1:(ntimes-1)) { ## loop over reporting times

res[ctr,] <- X ## record current state

while (t0<times[ctr+1]) { ## using while instead of for

rates <- ratefun(X,pars,t0) ## calculate current rates

if (all(rates==0)) break ## extinction

totrate <- sum(rates)

elapsed <- rexp(1,totrate) ## sample elapsed time

which.trans <- sample(1:nrow(trans),size=1,prob=rates) ## pick transition

t0 <- t0+elapsed ## update time

X <- X+trans[which.trans,] ## add transition values to current state

} }

cbind(times,res)

}

## starting condition (1 infected, 99 susceptible)

start=c(S=99,I=1)

# specify rate function: beta\*S\*I and gamma\*I for infection/transmission and recovery

ratefun.SIS = function(X,pars,time) {

vals = c(as.list(pars),as.list(X)) ## attach state and pars as lists

rates = with(vals, ##allows reference to states and parameters by name

c(infection=beta\*S\*I, recovery=gamma\*I))

}

statenames.SIS = c("S","I") ## state variable names

transnames.SIS = c("infection","recovery") # transition names

# transmission matrix

trans.SIS = matrix(c(-1,1,1,-1),

byrow=TRUE, ## default is by column

ncol=2, ## number of columns = number of state variables

dimnames=list(transnames.SIS,statenames.SIS))

# specify parameters

pars.SIS= c(beta=0.05,gamma=1)

# specify simulation time

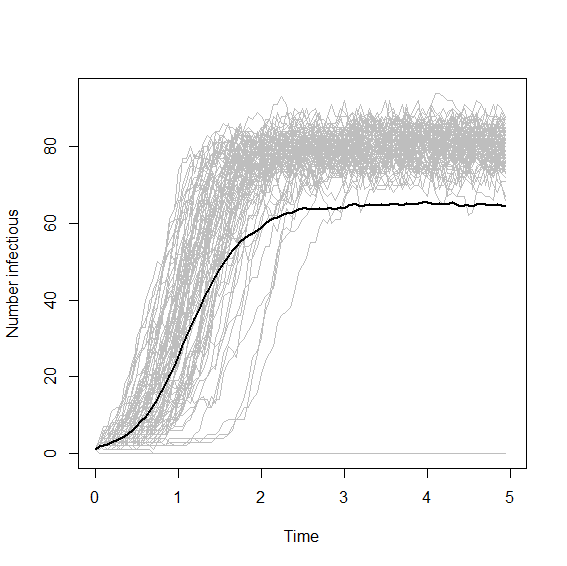
times= seq(0, 5, by = 0.05)

# run the simulation

G.SIS.mult= replicate(100, gillesp(start = start, times = times, ratefun = ratefun.SIS, trans = trans.SIS, pars = pars.SIS)[, "I"])

matplot(times,G.SIS.mult,type="l",col="gray",lty=1, xlab="Time", ylab="Number infectious")

lines(times,rowMeans(G.SIS.mult),lwd=2)

**Q:** Why is the mean (solid black line) substantially lower than most of the simulations (grey lines)?

**Exercise:**

Simulate the SIR model (with or without disease-induced death)

**Hint:** write down the transition matrix, and determine the rate of changes.