Stochastic processes 101 (in R)

May 16, 2005

Remember: Type `q()` (not just `q` by itself), or go to `File/Quit` in the menu bar, to quit R. In R, # signifies a comment. Any line beginning with #, or anything after # on a line, is simply ignored.

1 Waiting times and demographic stochasticity

A process running Recall from lecture 1 that a simple birh death process gives rise to exponentially distributed waiting times until the next birth/death. plot exponential distribution, with a rate of 0.5

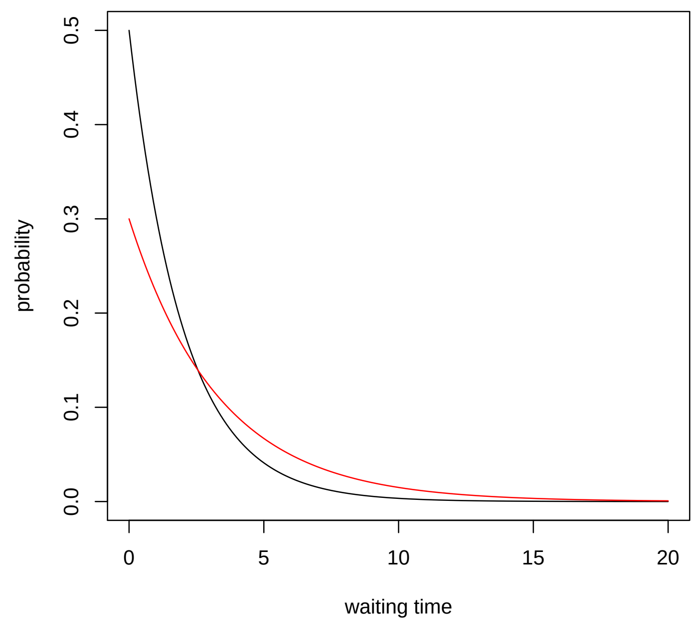
Sequence of candidate waiting times:

```
> wtime = seq(0, 20, by = 0.1)
```

Calculate the probability of each waiting time given a rate of 0.5 (i.e., 50% probability per unit time of occurrence) or of 0.3:

```
> prob1 = dexp(wtime, rate = 0.5)
> prob2 = dexp(wtime, rate = 0.3)
```

```
> plot(wtime, prob1, type = "l", xlab = "waiting time", ylab = "probability")
> lines(wtime, dexp(wtime, 0.3), col = 2)
```



The second line is equivalent to any of the following:

```
> lines(wtime, prob2, col = 2)
> lines(wtime, prob2, col = "red")
> curve(dexp(-0.3 * x), add = TRUE, col = 2)
```

Problem 1: Assume density-independent population growth in a population of 100 individuals, where per capita birth rate is 0.5 and per capita death rate is 0.3. (a) Use rexp() to calculate the probability that the next event is a birth. (b) what would the death rate need to be for there to be a 30% chance that the next event is a birth. (hint loop over a sequence % of death rates)

Simple epidemic under demographic stochasticity:

Simulate a simple epidemic for 10,000 events under demographic stochasticity:

```
> n.events <- 1000
> S = rep(NA, n.events)
> I = rep(NA, n.events)
> R = rep(NA, n.events)
> timing = rep(NA, n.events)
```

Set the initial population size and time (0):

```
> S[1] = 100
> I[1] = 2
```

```
> R[1] = 0
> timing[1] = 0
```

Set parameters:

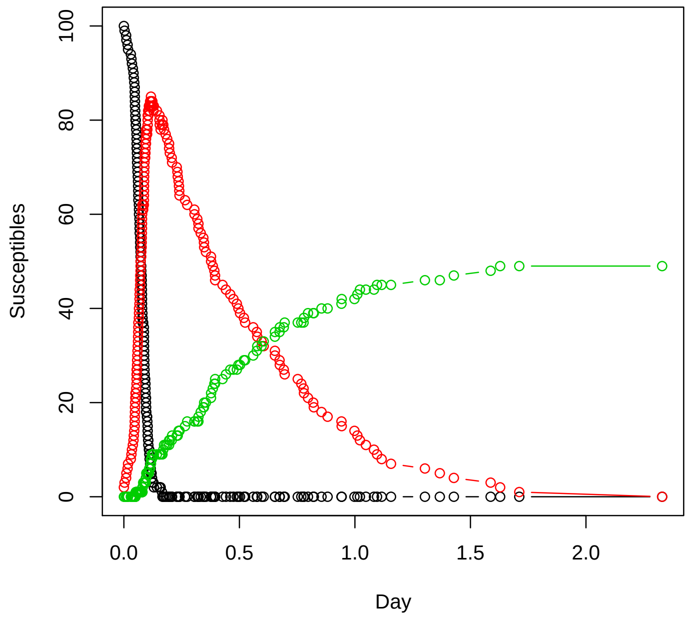
```
> beta = 0.5
> alpha = 1
> gamma = 1
```

Simulate the time series (starting at time step 2, ending at n.events)

```
> for (t in 2:n.events) {
+     if (I[t - 1] == 0) {
+         warning("epidemic ended before max time steps")
+         break
+     }
+     if (S[t - 1] == 0) {
+         inf.rate = 0
+         inf.time = Inf
+     }
+     else {
+         inf.rate = beta * S[t - 1] * I[t - 1]
+         inf.time = rexp(1, inf.rate)
+     }
+     I.death.rate = alpha * I[t - 1]
+     I.death.time = rexp(1, I.death.rate)
+     recovery.rate = gamma * I[t - 1]
+     recovery.time = rexp(1, recovery.rate)
+     elapsed.t = min(inf.time, I.death.time, recovery.time)
+     if (elapsed.t == inf.time) {
+         S[t] = S[t - 1] - 1
+         I[t] = I[t - 1] + 1
+         R[t] = R[t - 1]
+     }
+     else if (elapsed.t == I.death.time) {
+         S[t] = S[t - 1]
+         I[t] = I[t - 1] - 1
+         R[t] = R[t - 1]
+     }
+     else {
+         S[t] = S[t - 1]
+         I[t] = I[t - 1] - 1
+         R[t] = R[t - 1] + 1
+     }
+     timing[t] = timing[t - 1] + elapsed.t
+ }
```

Plot the results:

```
> plot(timing, S, type = "b", xlab = "Day", ylab = "Susceptibles")
> lines(timing, I, type = "b", col = 2)
> lines(timing, R, type = "b", col = 3)
```

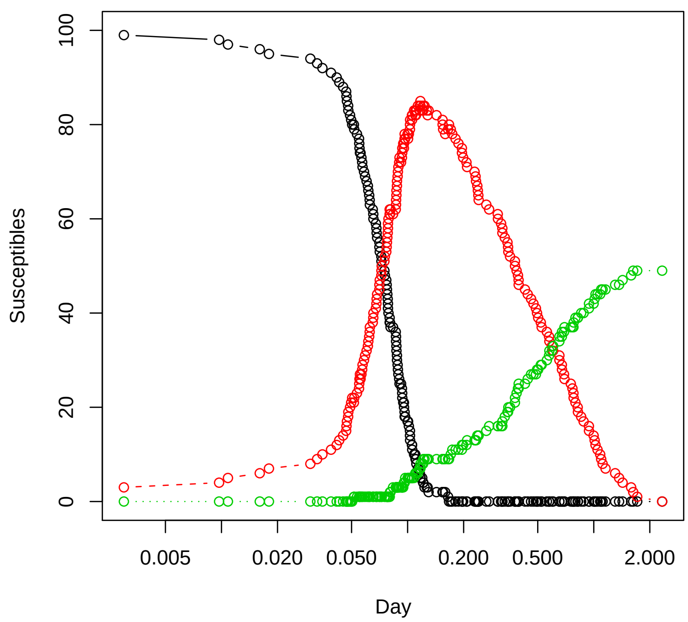


This is equivalent to:

```
> matplot(timing, cbind(S, I, R), type = "b", xlab = "Day", ylab = "Susceptibles",
+     pch = 1)
```

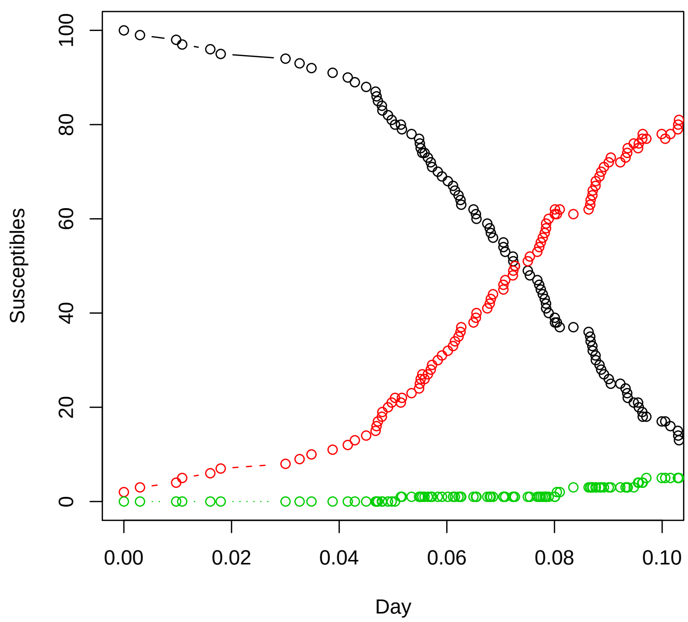
Plotting with a logarithmic time scale shows the initial stages of the epidemic more clearly:

```
> matplot(timing, cbind(S, I, R), type = "b", log = "x", xlab = "Day",
+     ylab = "Susceptibles", pch = 1)
```



Of course, we could also just explicitly restrict

```
> matplot(timing, cbind(S, I, R), type = "b", xlim = c(0, 0.1),
+     xlab = "Day", ylab = "Susceptibles", pch = 1)
```


Susceptibles
100
80
60
40
20
0
0.00
0.02
0.04
0.06
0.08
0.10
Day