Lecture 7: Birth-death models

Fugo Takasu
Dept. Information and Computer Sciences
Nara Women's University
takasu@ics.nara-wu.ac.jp

7 June 2006

1 A deterministic model of birth and death

Consider a population and imagine that each individual gives birth to an offspring or dies according a certain rule. Let the population size be N and we focus on the dynamics of N as a function of time t. If both birth and death occur at a constant rate, say, β and δ, respectively, the net change of the population size N within a short time interval Δt is given as

$$\Delta N = N(t + \Delta t) - N(t) = \beta \Delta t N - \delta \Delta t N$$

because there are N individuals and each individual gives birth and dies with the rate $\beta \Delta t$ and $\delta \Delta t$ respectively.

By arranging this equation and letting $\Delta t \to 0$ we have a differential equation

$$\frac{dN}{dt} = (\beta - \delta)N \tag{1}$$

The solution is

$$N(t) = N(0) \exp[(\beta - \delta)t] \tag{2}$$

where $N(0)$ is initial population size.

It is obvious that the population size N exponentially increases when the birth rate exceeds the death rate, $\beta > \delta$, and it exponentially decreases toward zero when $\beta < \delta$. If we take the logarithm of $N(t)$, it increases or decreases linearly with time t and the slop is given by $\beta - \delta$ because $\log N(t) = \log N(0) + (\beta - \delta)t$.

This is a deterministic model of birth and death. $\beta - \delta$ is the net rate of increase per individual. As in the previous deterministic immigration-emigration model the population size should be interpreted as "density", not the number of individuals as non-negative integer. The difference is that α and β in the immigration-emigration model are now replaced with βN and δN where N is the population size as density. We now want to derive a stochastic model that corresponds to this deterministic model.

2 A stochastic model

We assume that the birth rate β is the probability that an individual gives birth to an offspring and that the death rate δ the probability that the individual dies within a unit time. We also assume that within a short time interval Δt, only one of the following three cases occurs mutually exclusively; an individual 1) gives birth to an offspring, 2) dies, or 3) neither gives birth nor dies. This stochastic birth-death process could be implemented using the algorithm

For all individuals repeat 1) Give birth to a new individual with probability $\beta\Delta t$. 2) Remove this individual with probability $\delta\Delta t$.

Here is an outline of the program that simulates this stochastic process.

```
#define BIRTH_RATE 0.03
#define DEATH_RATE 0.02
#define DT  0.1
#define INTV 10
main()
{
  int pop_size, new_indiv, dead_indiv, i, step;
  double prob_birth, prob_death, ran;

  prob_birth = BIRTH_RATE* DT;
  prob_death = DEATH_RATE * DT;

  pop_size = 10;   /* initial population size */

  for(step=0; step<5000; step++){    /* advance the time by DT */

     if( step%INTV == 0) printf("%d ", pop_size);
     new_indiv = 0;
     dead_indiv = 0;
     for(i=1; i<=pop_size; i++){  /* for all individuals */
       ran = ran2(seed);
       if( ran < prob_birth )
         new_indiv++;
       else if( prob_birth < ran && ran < prob_birth + prob_death )
         dead_indiv++;

    } /* end of for i */

    pop_size += (new_indiv - dead_indiv) ;
  } /* end of for step */
}
```

3 Waiting time

As in the stochastic immigration-emigration model, it would be better to introduce waiting time to run simulation. Either birth or death takes place with the rate $\beta N + \delta N$, and waiting time to the next event (either birth or death) w is given as an exponential distributed random variable those p.d.f is

$$f(w) = \lambda \exp[-\lambda w]$$

where $\lambda = \beta N + \delta N$. A birth occurs with the conditional probability $\beta N/(\beta N + \delta N)$ and a death likewise. For the sake of calculating average population size later, it is convenient to output the population size N with an equal time interval $\Delta T = 1$.

4 Simulation

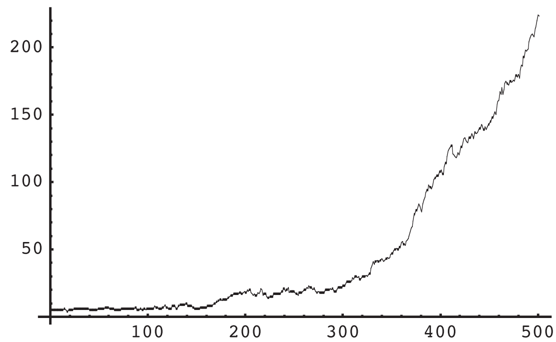
1. Write a C program to carry out simulation of the stochastic birth-death process. In the simulation we start with an initial population size, say, $n(0) = 10$ and repeat the dynamics with the same initial condition for several times. The dynamics of the population size should be written into a file. The data should be separated by a white space and write them in one line in the following format (assuming the time interval is 1).

Trial 1:	$n(0)$	$n(1)$	$n(2)$	$\cdots$	$n(100)$
Trial 2:	$n(0)$	$n(1)$	$n(2)$	$\cdots$	$n(100)$
Trial 3:	$n(0)$	$n(1)$	$n(2)$	$\cdots$	$n(100)$
$\vdots$					

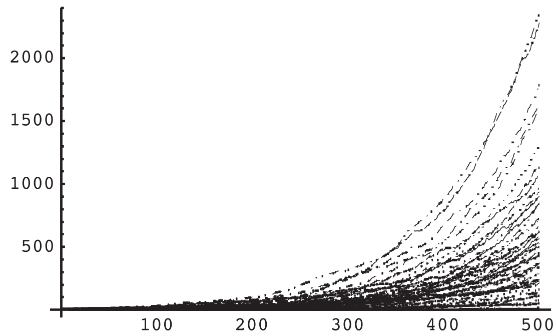
2. Using *Mathematica*, draw the simulated dynamics both in normal and logarithmic scale to see how the population size changes. Observe that in some trial the population size N can become zero at some time t and and hereafter it remains zero. Consider why the population size N remains zero once it reached zero.

Stochastic birth-death process by C

```
In[114]:= << Graphics`Graphics`

In[115]:= << Graphics`MultipleListPlot`

In[116]:= SetDirectory["/Users/takasu/home/情報科学科の仕事/講義/平成１８年度/
             H18 大学院講義/Birth-death model/birth-death/build/Development/"]

Out[116]= /Users/takasu/home/情報科学科の仕事/講義/平成１８年度/
            H18 大学院講義/Birth-death model/birth-death/build/Development

In[121]:= data = ReadList["data_eq_intv", Real, RecordLists ? True];
          Length[data]

Out[122]= 50

In[123]:= ListPlot[data[[1]], PlotJoined ? True]
```

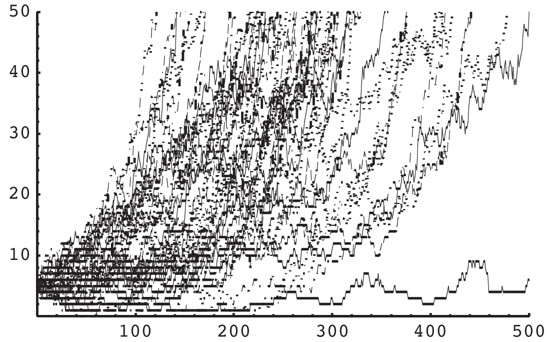



```
Out[123]= - Graphics -

In[124]:= MultipleListPlot[data, PlotJoined ? True, SymbolShape ? None, PlotRange ? All]
```



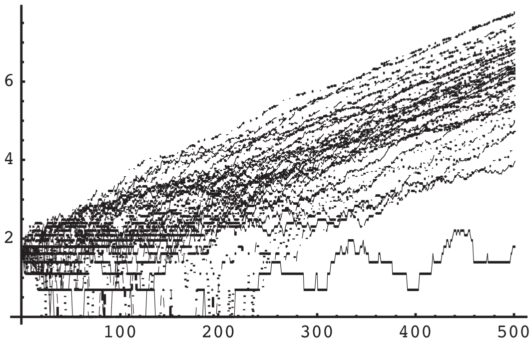

```
Out[124]= - Graphics -
```

```
In[125]:= MultipleListPlot[data, PlotJoined ? True,
            SymbolShape ? None, PlotRange ? {{0, 500}, {0, 50}}]
```



```
Out[125]= - Graphics -
```

```
In[132]:= MultipleListPlot[ Log[data], PlotJoined ? True,
            SymbolShape ? None, PlotRange ? {All, All}]
```



```
Out[132]= - Graphics -
```

```
In[136]:= Fit[ Take[ Log[data[[1]]], -100], {1, t}, t]

Out[136]= 4.69288 + 0.00665416 t
```

```
In[138]:= Do[
            tmp  = Take[ Log[data[[i]]], -20];
            regress = Fit[tmp, {1, t}, t] ;
            Print[regress], {i, 1, Length[data]}
           ]

5.21152 + 0.00992187 t

3.90098 + 0.0190778 t

7.26946 + 0.0106285 t

6.76606 + 0.010213 t

Indeterminate

1.2639 + 0.0195746 t

5.60861 + 0.0105491 t

7.20476 + 0.00974918 t

4.38565 + 0.0179139 t

6.6896 + 0.00953472 t

6.61389 + 0.00919948 t
```