Lecture 3: Immigration-emigration models

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

10 May 2006

1 A deterministic model of immigration and emigration

Consider a population and imagine that immigration into and emigration out of the population occur. Let the population size be N and we focus on the dynamics of N as a function of time t. If both immigration and emigration occur at a constant rate, say, α and β, respectively, the net change of the population size N during a short time interval Δt is given as

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

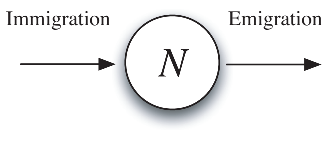
Arranging this equation and letting $\Delta t \to 0$ we have an ordinary differential equation ODE

$$\frac{dN}{dt} = \alpha - \beta \tag{1}$$

The solution is

$$N(t) = (\alpha - \beta)t + C \tag{2}$$

where C is a constant that should be determined with initial condition $C = N(0)$.



It is obvious that the population size linearly increases when immigration rate is larger than emigration rate, $\alpha > \beta$, and it linearly decreases when $\alpha < \beta$. In the latter case N continues to decline and it could be negative. But in biological sense "negative" population size makes no sense and we should interpret that the population size N has reached and remain zero, i.e., no individuals can emigrate from an empty population.

This is a deterministic model of immigration and emigration. This is "deterministic" because once initial condition $N(0)$ is given, the dynamics of $N(t)$ is uniquely determined as the solution (2). Here the population size should be interpreted as "density", not the number of individuals as integer. We now want to derive a stochastic model that corresponds to this deterministic model.

2 A stochastic model

There are a number of ways that extend the deterministic model into stochastic model. The immigration rate α and emigration rate β could be random variables associated with certain distributions. In this case the equation (1) becomes a stochastic differential equation because the parameter α and β themselves vary stochastically. Analysis of such a stochastic differential equation is not simple and we look for another way of extending the deterministic model.

In the immigration and emigration process, it would make sense to assume that the immigration rate α is the probability that an individual immigrates into the population within a short time interval Δt and that the emigration rate β the probability that an individual goes out of the population.

This stochastic process is summarized as follows. With probability $\alpha\Delta t$, an individual joins the population and the population size is increased by one. With probability $\beta\Delta t$, an individual goes out of the population and the population size is decreased by one. In this formulation the population size is "integer", not density represented by floating point number. This stochastic process is easily implemented using the following rule.

An event A that occurs with probability P

is equivalent to

1) Generate a random number X that distributes uniformly between 0 and 1.
2) If $X < P$, we assume the event A has occurred. Otherwise, the event A did not occur.

We also assume that the change of the population size within time interval Δt is at most ± 1. This means that only one of the following three cases occurs mutually exclusively during Δt; 1) an individual joins, 2) an individual quits, 3) no change, with probability $\alpha\Delta t$, $\beta\Delta t$, $1 - \alpha\Delta t - \beta\Delta t$, respectively. This would be satisfied if we assume the interval Δt is so small that the population size changes ± 1.

According to the above rule, 1) we draw a uniformly distributed random number X, 2) If X falls between 0 and $\alpha\Delta t$ the population size is incremented by one, or 3) If X falls between $\alpha\Delta t$ and $\alpha\Delta t + \beta\Delta t$, it is decremented by one, and 4) Otherwise, the population size does not change.

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

```
#define ALPHA   0.3
#define BETA  0.2
#define DT  0.1
#define INTV  10

main()
{
  int pop_size;    /* population size */
  int step;    /* for count of time */
  double prob1, prob2, rand_uniform;    /* for probabilities */

  /* initialization of ran2()  */

  prob1 = ALPHA * DT;
  prob2 = BETA * DT;

  pop_size = 10;   /* initial population size */

  for(step=0; step<1000; step++){    /* advance the time by DT */

    if( step % INTV == 0)
      printf("%d\n", pop_size);

    rand_uniform = ran2(seed);
    if( rand_uniform < prob1 )
      pop_size++;
    eise if( prob1 < rand_uniform && rand_uniform < prob1 + prob2 )
      pop_size--;

    if( pop_size < 0 ) pop_size = 0;   /* pop. size cannot be negative */

  } /* end of for */

}
```

3 Waiting time to the next event

We have implemented the stochastic immigration-emigration process using a fixed time interval Δt which has been assumed to be small enough. But to what extent should it be small? Ideally we can set Δt as small as possible. But practically, setting Δt small like $\Delta t = 0.001$ needs a large number of iterations to reach a certain end-time.

A better and more rigorous way to implement such a stochastic process is to introduce "waiting time" to the next event. When an event occurs randomly at a rate λ (λ times within a unit time on average), the number of actual events N follows a Poisson process with the parameter λ, i.e.,

$P(N = n) = \exp[-\lambda]\lambda^n/n!$. The distribution of the waiting time W to the next event, $F(w)$, is

$$\begin{aligned} F(w) &= \text{Prob}(W \le w) = 1 - \text{Prob}(W > w) \\ &= 1 - \text{Prob}(\text{no events in } [0, w]) \\ &= 1 - \exp[-\lambda w] \end{aligned}$$

and the p.d.f. is given as

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

That is, the waiting time to the next event is exponentially distributed in Poisson process. The average of the waiting time is easily calculated as

$$T_w = \int_0^\infty w f(w) dw = \frac{1}{\lambda}$$

It is the inverse of the rate λ and is intuitively reasonable.

In the immigration-emigration process, either immigration (population size incremented by one) or emigration (population size decremented by one) occurs at the rate $\alpha + \beta$. Thus the waiting time W to *any* change of population size follows exponential distribution with $\lambda = \alpha + \beta$. Conditional probability that immigration has occurred is $\alpha/(\alpha+\beta)$ and that emigration has occurred is $\beta/(\alpha+\beta)$.

The new algorithm is

```
1) Generate a waiting time $W$ and make proceed the time by $W$.
2) Generate a uniform distributed random number $X$.
 If $X < \alpha/(\alpha + \beta)$, immigration has occurred. Otherwise, emigration has occurred.
3) Repeat 1) and 2).
```

In this algorithm, the time proceeds with a variable interval (waiting time). But it is often convenient to output the dynamics of N with a fixed time interval.

4 Simulation

1. Write a C program to carry out the simulation with a fixed time interval Δt. In the simulation we start the stochastic process 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 at $t = 0, 1, 2, \cdots$ should be written into a file in the following format with a pair of t and $n(t)$ separated by white space.

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

2. Using *Mathematica*, draw the dynamics in figures by reading the time sequence of the population size.
3. Write another C program to run the simulation with variable time interval (waiting time). And compare the dynamics with that of a fixed time interval.

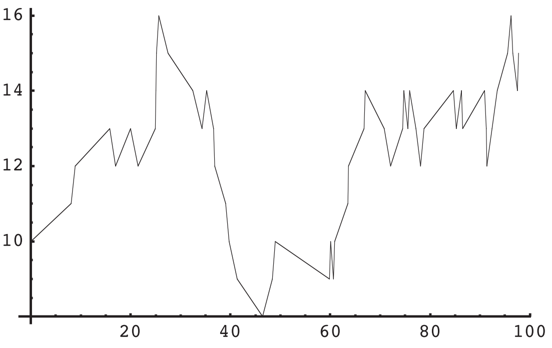
Stochastic immigration-emigration process: Simulation variable time interval

```
In[3]:= << Graphics`MultipleListPlot`

In[4]:= << Statistics`DescriptiveStatistics`
```

■ Simulation by C

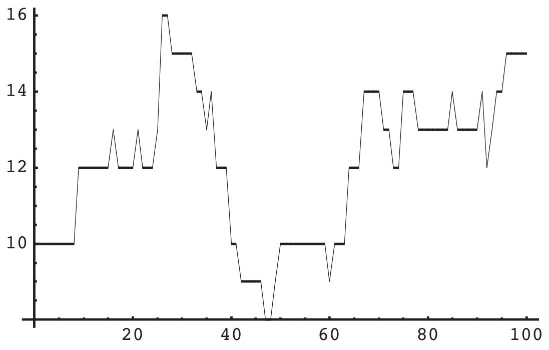
```
In[5]:= SetDirectory["/Users/takasu/home/情報科学科の仕事/講義/平成１８年度/H18 大学院講義
        /Immigration model/immigration-migration/build/Development"]

Out[5]= /Users/takasu/home/情報科学科の仕事/講義/平成１８年度/H18
          大学院講義/Immigration model/immigration-migration/build/Development

In[6]:= data = ReadList["data_var_intv",Real,RecordLists->True];
        len = Length[data]
        data2 = Map[Partition[#,2]&,data];

Out[7]= 100

In[9]:= dataEqIntv = ReadList["data_eq_intv", Real, RecordLists → True];
        len = Length[data]
        dataEqIntv2 = Map[Partition[#, 2] &, dataEqIntv];

Out[10]= 100

In[12]:= data2[[1]]

Out[12]= {{0., 10.}, {8.08089, 11.}, {8.89152, 12.}, {15.8257, 13.}, {16.9576, 12.},
          {20.0459, 13.}, {21.4251, 12.}, {24.9252, 13.}, {25.085, 14.}, {25.2216, 15.},
          {25.6454, 16.}, {27.5009, 15.}, {32.4232, 14.}, {34.3067, 13.}, {35.2519, 14.},
          {36.5654, 13.}, {36.8391, 12.}, {39.0561, 11.}, {39.7345, 10.}, {41.2968, 9.},
          {46.4579, 8.}, {48.4338, 9.}, {49.0135, 10.}, {59.8277, 9.}, {60.06, 10.},
          {60.6768, 9.}, {60.8851, 10.}, {63.5101, 11.}, {63.67, 12.}, {66.7634, 13.},
          {66.9816, 14.}, {70.7441, 13.}, {72.0135, 12.}, {74.5065, 13.},
          {74.7272, 14.}, {75.572, 13.}, {75.8765, 14.}, {77.1662, 13.}, {78.1438, 12.},
          {78.7558, 13.}, {84.6708, 14.}, {85.2426, 13.}, {86.2878, 14.}, {86.4987, 13.},
          {90.8578, 14.}, {91.2066, 13.}, {91.3511, 12.}, {92.4123, 13.}, {93.4795, 14.},
          {95.5561, 15.}, {96.1695, 16.}, {96.5772, 15.}, {97.4561, 14.}, {97.717, 15.}}

In[13]:= dataEqIntv2[[1]]

Out[13]= {{0., 10.}, {1., 10.}, {2., 10.}, {3., 10.}, {4., 10.}, {5., 10.}, {6., 10.},
          {7., 10.}, {8., 10.}, {9., 12.}, {10., 12.}, {11., 12.}, {12., 12.}, {13., 12.},
          {14., 12.}, {15., 12.}, {16., 13.}, {17., 12.}, {18., 12.}, {19., 12.}, {20., 12.},
          {21., 13.}, {22., 12.}, {23., 12.}, {24., 12.}, {25., 13.}, {26., 16.}, {27., 16.},
          {28., 15.}, {29., 15.}, {30., 15.}, {31., 15.}, {32., 15.}, {33., 14.},
          {34., 14.}, {35., 13.}, {36., 14.}, {37., 12.}, {38., 12.}, {39., 12.},
          {40., 10.}, {41., 10.}, {42., 9.}, {43., 9.}, {44., 9.}, {45., 9.}, {46., 9.},
          {47., 8.}, {48., 8.}, {49., 9.}, {50., 10.}, {51., 10.}, {52., 10.}, {53., 10.},
          {54., 10.}, {55., 10.}, {56., 10.}, {57., 10.}, {58., 10.}, {59., 10.}, {60., 9.},
          {61., 10.}, {62., 10.}, {63., 10.}, {64., 12.}, {65., 12.}, {66., 12.}, {67., 14.},
          {68., 14.}, {69., 14.}, {70., 14.}, {71., 13.}, {72., 13.}, {73., 12.}, {74., 12.},
          {75., 14.}, {76., 14.}, {77., 14.}, {78., 13.}, {79., 13.}, {80., 13.}, {81., 13.},
          {82., 13.}, {83., 13.}, {84., 13.}, {85., 14.}, {86., 13.}, {87., 13.}, {88., 13.},
          {89., 13.}, {90., 13.}, {91., 14.}, {92., 12.}, {93., 13.}, {94., 14.},
          {95., 14.}, {96., 15.}, {97., 15.}, {98., 15.}, {99., 15.}, {100., 15.}}
```

```
g1 = ListPlot[data2[[1]],PlotJoined->True,PlotRange->All]
```



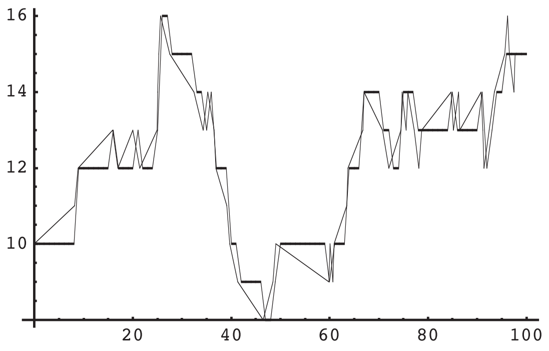
```
- Graphics -
```

```
g2 = ListPlot[dataEqIntv2[[1]], PlotJoined → True]
```



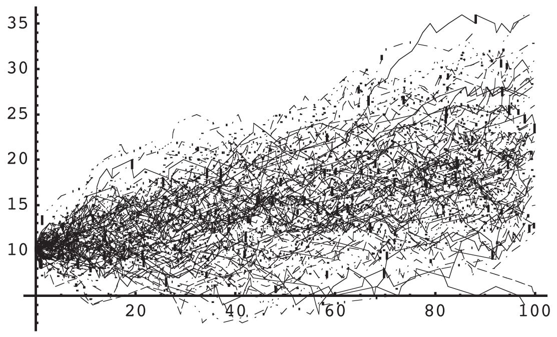
```
- Graphics -
```

```
Show[g1, g2]
```



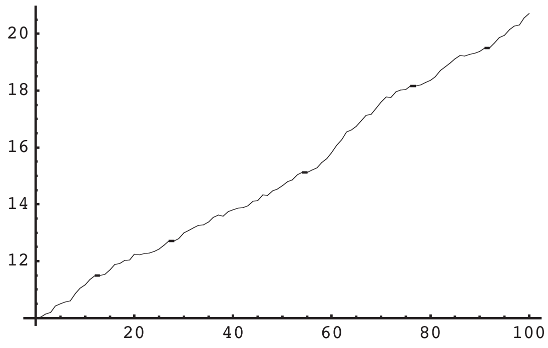
```
- Graphics -
```

```
gSim = MultipleListPlot[data2, PlotJoined → True, SymbolShape → None, PlotRange → All]
```

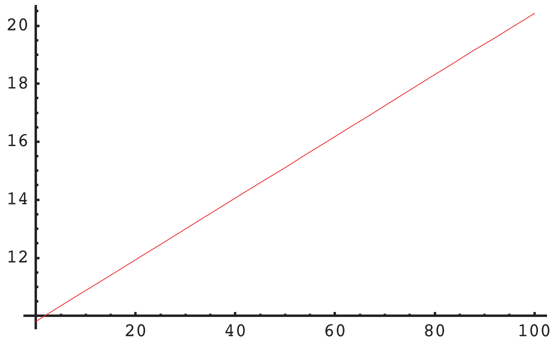


```
- Graphics -

popSeqList = {};

Do[
 tmp = Transpose[ dataEqIntv2[[i]] ] ;
 AppendTo[popSeqList, tmp[[2]] ], {i, 1, Length[dataEqIntv]}
]

timeSeq = Transpose[dataEqIntv2[[1]]][[1]];
popSeqAverage = Map[Mean, Transpose[popSeqList]];
seqAverage = {timeSeq, popSeqAverage} // Transpose

{{0., 10.}, {1., 10.03}, {2., 10.14}, {3., 10.2}, {4., 10.41}, {5., 10.49},
 {6., 10.56}, {7., 10.59}, {8., 10.86}, {9., 11.05}, {10., 11.17}, {11., 11.36},
 {12., 11.49}, {13., 11.5}, {14., 11.54}, {15., 11.7}, {16., 11.87}, {17., 11.92},
 {18., 12.01}, {19., 12.03}, {20., 12.23}, {21., 12.22}, {22., 12.25}, {23., 12.28},
 {24., 12.35}, {25., 12.42}, {26., 12.56}, {27., 12.7}, {28., 12.7}, {29., 12.78},
 {30., 12.99}, {31., 13.07}, {32., 13.18}, {33., 13.26}, {34., 13.27}, {35., 13.37},
 {36., 13.53}, {37., 13.62}, {38., 13.57}, {39., 13.73}, {40., 13.8}, {41., 13.87},
 {42., 13.88}, {43., 13.94}, {44., 14.1}, {45., 14.12}, {46., 14.32}, {47., 14.31},
 {48., 14.47}, {49., 14.53}, {50., 14.65}, {51., 14.79}, {52., 14.86}, {53., 15.04},
 {54., 15.11}, {55., 15.11}, {56., 15.21}, {57., 15.27}, {58., 15.47}, {59., 15.6},
 {60., 15.8}, {61., 16.06}, {62., 16.27}, {63., 16.54}, {64., 16.61}, {65., 16.73},
 {66., 16.94}, {67., 17.13}, {68., 17.16}, {69., 17.39}, {70., 17.6}, {71., 17.78},
 {72., 17.75}, {73., 17.96}, {74., 18.02}, {75., 18.05}, {76., 18.15}, {77., 18.17},
 {78., 18.21}, {79., 18.29}, {80., 18.36}, {81., 18.48}, {82., 18.71}, {83., 18.84},
 {84., 18.97}, {85., 19.12}, {86., 19.23}, {87., 19.22}, {88., 19.27}, {89., 19.32},
 {90., 19.37}, {91., 19.49}, {92., 19.5}, {93., 19.68}, {94., 19.86}, {95., 19.95},
 {96., 20.15}, {97., 20.27}, {98., 20.3}, {99., 20.56}, {100., 20.72}}
```

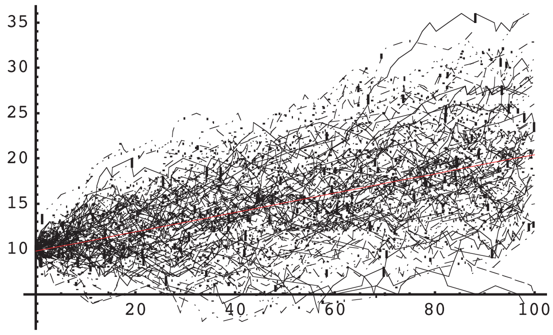
```
ListPlot[seqAverage, PlotJoined → True]
```



```
- Graphics -

fitted = Fit[seqAverage, {1, t}, t]

9.81317 + 0.106198 t

gFitted = Plot[fitted, {t, 0, 100}, PlotStyle → RGBColor[1, 0, 0]]
```



```
- Graphics -

Show[gSim, gFitted]
```



```
- Graphics -
```