Lecture 1: Generating random numbers

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

19 April 2006

1 Random variable

A random variable X has an associated probability with it. It can be either discrete or continuous.

An example of a discrete random variable is toss of coin (0 for tail and 1 for head with equal probability 1/2) and dice (1 through 6 with equal probability 1/6), or anything that follows binomial, geometric and poisson distribution. Let us now denote the probability of drawing a certain value, say i, as P_i. The it satisfies the following properties.

$$0 \leq P_i \leq 1$$
$$\sum_i P_i = 1$$

where summation is taken for all possible i. P_i is called probability distribution.

An example of a continuous random variable is the time for which you must be patient in a waiting line, or anything that follows uniform, exponential and gauss (normal) distribution. For continuous random variable X, the probability that X lies within a short interval x and $x + dx$ is defined as

$$\text{Prob}[x < X < x + dx] = P(x)dx$$

where $P(x)$ is called probability density function (pdf). It satisfies

$$0 \leq P(x)$$
$$\int P(x)dx = 1$$

where integral is taken for possible range of x.

Once a probability distribution (for discrete) or a probability density function (for continuous) is given, the expected value (mean, average) and the variance of the random variable are calculated as follows.

For discrete random variable,

$$\mathrm{E}[X] = \sum_i i \times P_i$$
$$\mathrm{Var}[X] = \mathrm{E}[(X - \mathrm{E}[X])^2] = \sum_i (i - \mathrm{E}[X])^2 \times P_i$$

For continuous random variable,

$$\mathrm{E}[X] = \int x \times P(x) dx$$
$$\mathrm{Var}[X] = \mathrm{E}[(X - \mathrm{E}[X])^2] = \int (x - \mathrm{E}[X])^2 \times P(x) dx$$

Variance can also be calculated as

$$\mathrm{Var}[X] = \mathrm{E}[X^2] - \mathrm{E}[X]^2$$

Here below I list some continuous random variables often used in many sciences.

1.1 Uniform random variable

An uniform random variable X takes a value x within a certain range with equal probability. Let the range be $a < x < b$. Then the probability density function (pdf) is given as

$$P(x) = \frac{1}{b-a} \quad \text{for} \quad a < x < b$$
$$= 0 \quad \text{otherwise}$$

And naturally $\int_a^b P(x)dx = 1$. Derive the expected value and the variance. Try.

1.2 Exponential random variable

An exponential random variable X takes a non-negative value x ($0 < x < \infty$). The pdf contains a parameter $\gamma > 0$ and it is defined as

$$P(x) = \gamma \exp[-\gamma x]$$

Confirm that $\int_0^\infty P(x)dx = 1$. Derive the expected value and the variance.

1.3 Gauss random variable

A gauss random variable (Gaussian) X takes a value x ($-\infty < x < \infty$). The pdf is

$$P(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left[-\frac{(x-m)^2}{2\sigma^2}\right]$$

and it contains two parameters m and σ. This pdf is often presented as $N(m, \sigma^2)$. m is the expected value and σ^2 is the variance.

1.4 Exercise

1. Derive the expected value and the variance of an uniform random variable X whose range is $a < X < b$.
2. Derive the expected value and the variance of an exponential random variable X whose pdf is given as $P(x) = \gamma \exp[-\gamma x]$.
3. Check that the expected value and the variance of a random variable X that follows $N(m, \sigma^2)$ is m and σ^2, respectively, i.e., $\int_{-\infty}^{\infty} xP(x)dx = m$ and $\int_{-\infty}^{\infty} x^2P(x) - m^2 = \sigma^2$.

2 Generating random numbers

Most part of this session follows "Numerical Recipes in C", a popular book for introducing numerical methods using C languages. The content can be obtained for free from the web site http://www.library.cornell.edu/nr/bookcpdf.html.

Among the three random variables listed above, the uniform random variable is most important because the other two random variable can be derived and transformed from a uniform random variable. Let us now focus on an uniform random variable whose range is $0 < x < 1$.

2.1 Generating uniform random numbers

There are several algorithms to generate uniform random numbers. But some of them produces rather poor quality (e.g., built-in random number generator in C). Here we use a function introduced in the Numerical Recipe as `ran2()` that returns a real value between 0 and 1 with equal probability. I have typed the code and it is available for use. Note that any random number generated by some algorithmic method is nothing more than pseudo random number. But we will not step into a philosophically deep question "Can any algorithm produce real random number?" any more.

To generate random numbers in most algorithms, we need a "seed" to specify a series of random numbers. If we start generating a series of random numbers with the same seed, we have always the same series of random numbers.

Example of series of random numbers generated using the same seed value.

Trial 1: $\{0.201669, 0.411992, 0.146775, 0.236965, 0.773501, \dots\}$

Trial 2: $\{0.201669, 0.411992, 0.146775, 0.236965, 0.773501, \dots\}$

Trial 3: $\{0.201669, 0.411992, 0.146775, 0.236965, 0.773501, \dots\}$

This doesn't make sense in simulation except for debugging program. For choosing a different seed value every time simulation is run, the time of run (day, hour, minute, second, which can be readily obtained in C program) can be a good seed to produce different series of random numbers.

Here is an example how to generate uniform random number ($0 < x < 1$) in C.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>  /* This is necessary to get time */

#define STEP 100

extern double ran2(long);   /* ran2() is defined external in another file */

int main(void)
{
  long seed; /* seed is long integer */

  double rand1;
  int i;
  FILE *fp_uniform;

  seed = (long)time(NULL);   /* seed should be chosen different for every new  trial */

  /* Initialize the random generator ran2 with NEGATIVE argument */
  ran2(-seed);      /* This must be done only once before generating random number! */

  printf("Seed is %ld\n", seed);
  fp_uniform = fopen("uniform.out","w");

  for(i=0; i<STEP; i++){

    /* Uniform random number is generated and returned by calling ran2() */
    rand1 = ran2(seed);
    printf("%f\n", rand1)
    fprintf(fp_uniform, "%f\n", rand1);

  }
```

```
  fclose(fp_uniform);

  return 0;
}
```

2.2 Checking the uniform random numbers

In *Mathematica* we 1) read a set of random numbers saved in a file, 2) count the number of random numbers within a certain interval, and 3) make the frequency distribution to see if generated random numbers reflect original pdf.

In the notebook, we first read a package to do statistics, Statistics‘DataManipulation‘ at the beginning. This should be carried out only once after *Mathematica* is launched. We then designate the directory where our target file resides using the command SetDirectory. Note that *Mathematica*'s commands start with capital letter and they are case sensitive. Below is a sample *Mathematica* notebook.

```
<<Statistics`DataManipulation`
```

In[2]:=
```
SetDirectory["/Volumes/home/Users/takasu/Random variables
/random variables/build/"]
```

Out[2]= Volumes home 情報科学科の仕事 講義 平成１６年度
H16 大学院講義 Random variables random variables build

In[3]:=
```
data = ReadList["uniform.out",Real];
len = Length[data]
maxdata = Max[data]
```

Out[4]= 10000

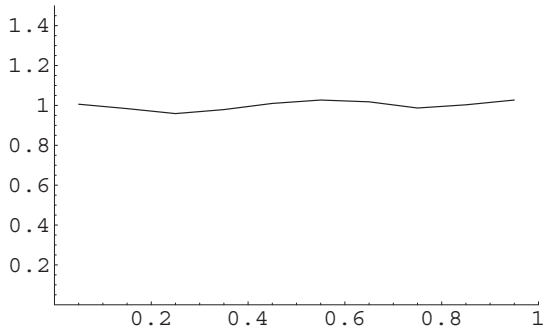
Out[5]= 0.999896

In[6]:=
```
dx = 0.1;
categories = 1/dx;
counts = BinCounts[ data,{0,maxdata,dx}]
midpoints = Table[x-dx/2,{x,dx,maxdata+dx,dx}];

dist2 = Transpose[ {midpoints, counts/len*categories}];
```

Out[8]= 1006, 984, 959, 979, 1010, 1027, 1018, 987, 1003, 1027

In[13]:=
```
g = ListPlot[ dist2,PlotJoined->True, PlotRange->{{0,1},{0,1.5}}]
```



Out[13]= Graphics

In[14]:=
```
meanUniform = Apply[ Plus, data]/len
```

Out[14]= 0.50279

In[15]:=
```
varianceUniform = Apply[Plus, (data - meanUniform)^2]/len
```

Out[15]= 0.0832519

In[16]:=
```
meanUniform = Integrate[ x ,{x,0,1}]
```

Out[16]= $\frac{1}{2}$

In[17]:=
```
Integrate[ (x-meanUniform)^2,{x,0,1}]
```

Out[17]= $\frac{1}{12}$

2.3 Exercise

1. Complete a C program that generates 100 uniform random numbers. Write the series of random numbers into a file.

2. By viewing the saved file, check and judge if these numbers follow uniform distribution. Does the simulated distribution look uniform if we increase the number of random numbers to 1000, or more?