Lecture 6

Random Number Generation.

In what follows we assume that R~U(0,1) are available in the software:

RAND() in Excel,
RND in Visual Basic
Or an appropriate function in another language.

The methods for generating these are the domain of numerical analysis and only marginally of interest to a statistics course.

Statistical packages – Data Desk, MINITAB, S-plus, GPSS provide random values from a broader range of distributions and these may be used for *static* simulations.

Static simulations are used to investigate properties of samples from distributions that cannot be obtained by analytical methods. For example distributions of "unusual" data summaries.

A special case is bootstrapping where distributions of simple summaries such as mean and variance are estimated without any model assumptions – using only the numerical data available.

Dynamic simulations are computer representations of processes that evolve in time.

Essentially a dynamic simulation is a program that branches randomly.
Almost all computer games are dynamic simulations.

For static simulations we can use statistical packages or other software. In dynamic simulations the random values are part of the program itself so we need to know how to get them.

We have already described how random Bernoulli numbers can be generated:

```
Function Rbernoulli(p)
If  RND≤p then
Z=1
Else
Z=0
Endif
Return(Z)
```

We have also explained that we can exploit relationships between variables to generate variables from other distributions:

```
Function Rgeometric(p)
X=0
Fail=false
While Fail=false
      X=X+1
If Rbernoulli(p)=1 then Fail=True
Wend
```

There are however two generic methods

Inverse distribution function method.

Consider a random variable X transformed by its own cumulative distribution function.

$Y=F_X(X)$ where X has cdf $F_X(x)$.

What is the cdf of Y?

$$F_Y(y) = P(Y \le y)$$
$$F_Y(y) = P(F_X(X) \le y) = P(X \le F_X^{-1}(y))$$

If X is cts its cdf is strictly increasing (monotonic) so the inverse is clearly defined.

But $P(X \le x) = F_X(x)$ so $F_Y(y) = F_X(F_X^{-1}(y)) = y$

So Y is Uniform(0,1) – this result has more implications than just random number generation. But we can use it as follows:

$Y=RND$
$F_X^{-1}(Y)$ has distribution function F_X.

Defining the inverse function is rather a formal way of looking at this. In practice to get a random value X from a distribution with cdf F_X solve:

$$F_X(X) = RND$$

Example: A single server queue.

Customers arrive at a queue according to a PP with rate 0.5 customers/minute.
We simulate such a process by generating the inter-arrival times. Gaps between arrivals.

The inter-arrival times (T) are exponential with $\lambda = 0.5$.

$$F_T(t) = 1 - e^{-0.5t}$$

so

$$\begin{aligned}
1 - e^{-0.5T} &= RND \\
e^{-0.5T} &= 1 - RND \\
-0.5T &= \log(1 - RND) \\
T &= -2\log(1 - RND).
\end{aligned}$$

The processing of each customer time is Weibull with shape a=2, scale b=1.

S has distribution function $F_S(s) = 1 - e^{-(s/b)^a}$

So $S = \left(-b\log(1 - RND)\right)^{1/a}$ here a=2, b=1 – clearly demonstrating the relationship between the exponential and the Weibull.

We can now simulate this queuing system.

For example in Excel:

Customer	R arrival	Gap	Arr Time	Start Serv	R service	serv time	depart
1	0.789971	3.121023	3.121023	3.121023	0.657899	1.035689	4.156712
2	0.52036	1.469438	4.590461	4.590461	0.442251	0.764098	5.354559
3	0.664064	2.181668	6.772129	6.772129	0.553677	0.898172	7.6703
4	0.981448	7.974386	14.74651	14.74651	0.875307	1.44288	16.1894
5	0.01494	0.030106	14.77662	16.1894	0.605714	0.964717	17.15411
6	0.937934	5.559103	20.33572	20.33572	0.927933	1.621776	21.9575

Queue mechanics mean that:

Arr Time (n) = Gap(n) +Arr Time(n-1)
Depart = Start Serv+ Serv Time
Start Serv (n) = maximum(Arr Time(n),Depart(n-1))

Information collected would be:
Waiting Time = Start Serv – Arr Time
Number in Queue (although this would be tricky in Excel)

Excel provides values of NORMSINV(p) so N(0,1) can be generated as:

NORMSINV(RAND())

Other inverse distribution functions are also available.

rand()	Normsinv
0.023862	-1.9798
0.585257	0.2154
0.050949	-1.6357
0.530378	0.0762
0.910734	1.3453
0.310936	-0.4932
0.458008	-0.1055

The normal values have been rounded to 4 decimal places.

For general normals we can use the property:

$$N(\mu,\sigma^2) = \mu + \sigma N(0,1)$$

A t-distribution has more probability in the tails (far away from 0) than the normal otherwise very similar. One parameter, n.

t = TINV(Rand(),n)

will generate positive values from this distribution.
To get the full distribution :
If $R_2<0.5$ then stick a "-" in front of the number.

The method cannot be used directly for discrete data. An extension of the method for Bernoulli but using the ideas above is as follows:

Subdivide the (0,1) range into the probabilities of X, generate RND check within which range it falls return X.

Best illustrated by example:

x	Poisson(x,2,0)	Poisson(x,2,1)
0	0.1353	0.1353
1	0.2707	0.4060
2	0.2707	0.6767
3	0.1804	0.8571
4	0.0902	0.9473
5	0.0361	0.9834
6	0.0120	0.9955
7	0.0034	0.9989
8	0.0009	0.9998

Only a probability of 0.0002 x>8 is ignored.

Method
Find largest x so that R < Poisson(x,λ,1) $=F_X(x)$
The function VLOOKUP(r,table,c) does the job.

R=0.2370 would result in a value of x=1

R=0.8345 -> x=2

and so on.

The rejection method

This is a different approach that works directly on the pdf rather than the cdf – the remarkable thing is that it works.

The method is simplest for a finite number of outcomes:

1. Move forward	probability 0.6
2. Turn to left	probability 0.3
3. Turn to right.	probability 0.1.

The method involves two steps:

1. Select a possibility with equal probability:

   X=1+int(n*RND1)   here n=3.

2. If p(X)≤RND2 then Return(X) else try again.

Thus in one pass the method only sometimes returns a value.

Implementation:

```
Function  Rmove
Local X
  generated=false
   While generated=false
       X=1+int(3*RND)
       If p(X)≤RND  Then   ‘a 2nd call to function RND
              Z=X
              generated=True
       endif
   wend
Return(Z)
```

To show that the method works we have to show that:

$$P(Z = X \mid Generated) = p(X)$$

This is an exercise in conditional probability.

However let us first consider an improved (more efficient) version:

Let H=max(p(x))

We then use the criterion:

```
…
If  p(X)≤H*RND2 Then
…
```

This way we avoid the cases where RND2 is too big to be accepted irrespective of the candidate X. Also the method is more general – while $p(X) \geq 0$ they no longer have to be ≤ 1 nor do they have to sum to 1.

$$P(X = x \mid generated) = \frac{p(x)}{\sum_{allx} p(x)}$$

Proof: X takes values 1..n .

$$P(X = x \mid generated) = \frac{P((X = x) \cap generated)}{P(generated)}$$

$$P((X = x) \cap generated) = P(generated \mid X = x) \times P(X = x)$$

$$P(generated \mid X = x) = P(H \times RND2 \le p(x)) =$$
$$= P(RND2 \le \frac{p(x)}{H}) = \frac{p(x)}{H}$$ because RND2 is U(0,1)

$P(X = x) = \frac{1}{n}$ because we chose X by an equally likely method.

So $Numerator = \frac{p(x)}{nH}$.

$$P(generated) = \sum_{x=1}^{n} P(generated \mid X = x) \times P(X = x)$$
$$= \sum_{x=1}^{n} \frac{p(x)}{H} \times \frac{1}{n} = \frac{1}{nH} \sum_{x=1}^{n} p(x)$$
The nH cancel giving the desired result.

The method efficiency=
$P(generate \text{ in one pass}) = q = \frac{1}{nH} \sum_{x=1}^{n} p(x)$ (for the unmodified method H=1, if p(x) are probabilities then H=max(p(x))≤ 1 and so this probability is smaller).

The number of passes required is Geometric(q) and so the average number of passes = 1/q and twice that many calls to RND are required.

This method also works for continuous random variables, in the proof replace summations by integrals. The criterion becomes:

If $H * RND2 \le f(x)$ then …

Where f(x) is the pdf and H is its maximum value. The candidate X is U(a,b) ((a,b) is the range of X). Further f(x) doesn't have to integrate to 1 (as p(x) do not need to sum to 1) so we can drop the awkward constants such as beta functions :
For Beta dist use $f(x) = x^{a-1}(1-x)^{b-1}$ - in fact the method is used where the integral cannot be done. All you need is the maximum of the integrand.

You don't even need that exactly!
The method will work with $H \geq \max(p(x))$ but efficiency will suffer with increasing H.

The only draw back is that range of X must be finite – we have no way of selecting the candidate X.

By hand example

Move forward	probability 0.6
Turn to left	probability 0.3
Turn to right.	probability 0.1.

We take H=0.6. and random numbers from RAND()

```
0.1863110.539556  0.86746
0.0647370.1156560.177365
0.809145  0.282720.007338
0.6836160.4974520.626544
```

Using these row-wise

1+INT(3*0.186..)=1
    0.6>H (=0.6) *0.539..   So return: Move forward.

1+INT(3*0.867..)=3
    0.1> 0.6*0.0647   So return: Turn Right

The third pair gives another Move forward.

The fourth pair

1+INT(3*0.809..)=3

    but 0.1<0.6*0.2827.. so do not return anything try again…
The fifth pair = another move forward and so on

For a cts distribution try Normal truncated to (-3,+3)

$f(x) = e^{-0.5x^2}$ H=1 at x=0 is maximum.

Candidate X = -3+6*RAND()

X	f(x)	RND2	Result
2.480312	0.289339	0.566307	try again
-0.80041	1.492134	0.762565	-0.80041
0.205306	0.90244	0.522587	0.205306
2.313498	0.314507	0.482606	try again
-0.89333	1.563089	0.518801	-0.89333
1.706226	0.426086	0.911269	try again
2.193992	0.333872	0.383632	try again

Method efficiency – $\int_{-3}^{3} e^{-0.5x^2} = \sqrt{2\pi}$ =2.506 (practically).

The area in the rectangle = 6*1= 6

So efficiency ==2.506/6= 0.4177.

The real advantage of the method occurs when probabilities (likelinesses) change dynamically during the run. All we have to do is to alter the specific value(s) and check if H needs to be modified. With the inverse CDF method a recalculation of the CDF is required.