Chapter 10

Simulation: An Introduction
Definition

- A simulation is an imitation of some real thing, state of affairs, or process.
- The act of simulating something generally entails representing certain key characteristics or behaviors of a selected physical or abstract system.


Use of Simulation

- Simulation is used in many contexts, including the modeling of natural systems or human systems in order to gain insight into their functioning.

- Simulation can be used to show the eventual real effects of alternative conditions and courses of action.
Example 1: Checking distribution theory

Theory:

- A sample of size 4 is taken from a normal distribution with mean 0.
- Consider the statistics

\[ t = \frac{\bar{X}}{\hat{\sigma}/\sqrt{4}} \]

- The statistic \( t \) follows a t-distribution with 3 degrees of freedom.
Example 1

Simulation

- Generate 1000 random samples each of size 4 from a normal distribution.
- For each sample, compute the value of the statistic $t$.
- Construct a histogram for these 1000 realizations of the statistic $t$. 
Example 1

Result:

- The histogram matches well with the $t(3)$ distribution.
Example 2: Comparing estimators

We have learnt several robust estimators of location.

Question: Which estimator is better, the trimmed mean or the Winsorized mean?

Secondary question:

• Under what conditions (in terms of the underlying distributions) is the estimator better?

• How to measure the performance? We may use the Mean Square Error (MSE = $Bias^2 + Variance$)

• It may be intractable to compute the MSE for each of the estimators under different underlying distributions,

• Simulation is an alternative solution

• How to design such a simulation study?
Example 3: Buffon’s needle experiment

- A needle of length $l$ is thrown randomly onto a grid of parallel lines with distance $d (> l)$
Example 3

- What is the probability that the needle intersects a line?
- Answer: \((2l)/\pi d\)
- Can we get the answer through simulations?
- A website gives the visualization of the experiment http://www.metablake.com/pi.swf
Example 3

The experiment

- Simulate the throwing of a needle into a grid of parallel lines, say \( N \) times

- Count the number of times the needle intersects a line, say \( n \) times

- Then \( n/N \) gives estimate of the probability that the needle intersects a line
How to do the simulation

• Step 1: Generate the position and the inclination of the needle
  – Generate a random number, \( x \), from Uniform \((0,d/2)\).
    This number represents the location of the centre of the needle.
  – Generate a random number, \( \theta \), from Uniform \((0, \pi)\).
    This number represents the angle of between the needle and the parallel lines.
How to do the simulation

• Step 2: Check if the needle cuts a line.
  – The needle cuts a line if \( x < (l/2)\sin(\theta) \)
  – Create a variable \( w = 1 \) if \( x < (l/2)\sin(\theta) \) and 0 otherwise.

• Step 3: Repeat Steps 1 and 2 \( n \) times. Count the number of times that \( w = 1 \), let say \( N \) times. Then an estimate of the probability of the needle intersects a line is given by \( n/N \).
Example 3: R Code

```r
# Buffon’s needle
# X~U(0,d/2), t~U(0,pi) where d is the distance between 2 parallel lines
# A needle of length L cut one of the lines if x<L/2*sin(t)
# Theoretical Probability=2*L/(pi*d)

ns=50000 ; d=2
L=1 ; d2=d/2
#Theoretical answer
2*L/(pi*d)
[1] 0.3183099

x=runif(ns,0,d2)
t=runif(ns,0,pi)
length(x[x<L/2*sin(t)])/ns
[1] 0.3186
```
Random number generator

Definition

• A sequence of pseudo-random number \( \{U_i\} \) is a deterministic sequence of number in [0,1] having the same relevant statistical properties as a sequence of random numbers.
Congruential generators

- Congruential generators are defined by
  \[ X_i = (aX_{i-1} + c) \mod M \]
  for a multiplier a, shift c, and modulus M.
- a, c and M are all integers
- Uniform random numbers are obtained by \( U_i = X_i / M \)
- To initialize, we have to provide a seed, \( X_0 \)
- If \( c = 0 \), generators having the form
  \[ X_i = aX_{i-1} \mod M \]
  are called multiplicative congruential generator.
- if \( c > 0 \), they are called linear congruential generator.
Remarks

• $M + 1$ values \{$X_0, X_1, \ldots, X_M$\} cannot be distinct and at least one value must occur twice, as $X_i$ and $X_{i+k}$, say.

• $X_i, X_{i+1}, \ldots X_{i+k-1}$ is repeated as $X_{i+k}, X_{i+k+1}, \ldots X_{i+2k-1}$.

• The sequence $X_i$ is periodic with period $k \leq M$.

• For multiplicative generators, the \textit{maximal} period is $M - 1$.

• If 0 ever occurs, it is repeated indefinitely.

• One of our primary objectives is to use a generator with as large period as possible.

• However, a large period does not guarantee a good generator.
R function for congruential generators

> #Linear congruential generators
> lcg=function(n,a,m,c,x0){
+ ran=NULL
+ for (i in 1:n){
+ x1=(a*x0+c)%%m
+ x0=x1
+ ran=c(ran, x1/m)
+ ran}
> lcg(10,397204094,2^31-1,0,1234)
[1] 0.24381116 0.08947511 0.38319371 0.72800325
[4] 0.72792771 0.17503827
[7] 0.40680994 0.90923700 0.34271140 0.55960111

17
Generate uniform random numbers: SAS

* Generate Uniform random numbers;

```sas
data try1;
seed=1234;
do i =1 to 10;
x=ranuni(seed);
output;
end;
keep x;
run;
```

```sas
proc print data=try1;
var x;
run;
```
Generate uniform random numbers: R

> # Generate 1000 random numbers from U(0,1) distribution
> x=runif(1000)

In general, “runif(n,a,b)” generates a vector of n random numbers from a uniform distribution between a and b.
Generate non-uniform random numbers

Inversion method

- If $X$ has a continuous distribution function $F(x)$ (i.e. $\Pr(X \leq x)$), then $F(X) \sim \text{Uniform } (0,1)$

Algorithm:

- Generate $U$ from Uniform $(0,1)$
- Set $X = F^{-1}(U)$ provided the inverse exists.
Exponential distribution

- If $X$ follows an exponential distribution with parameter $\lambda$ (i.e. $E(X) = \lambda$), then

$$F(x) = Pr(X \leq x) = \int_0^x \frac{1}{\lambda} e^{\frac{t}{\lambda}} dt = \int_0^{x/\lambda} e^{-y} dy = 1 - e^{-x/\lambda}$$

Solving $u = F(x) = 1 - e^{-x/\lambda}$ for $x$, we have

$x = F^{-1}(u) = -\lambda \times \log(1 - u)$. Then

- Generate $U$ from Uniform $(0,1)$
- Set $X = -\lambda \times \log(1 - U)$ or $X = -\lambda \times \log(U)$
Weibull distribution

- If X follows a Weibull distribution with parameter $\beta$, then it can be shown that

$$F(x) = 1 - \exp(-x^\beta) \text{ on } (0, \infty)$$

Note: $f(x) = \beta x^{\beta-1} \exp(-x^\beta)$ for $x$ in $(0, \infty)$

Solving $u = F(x) = 1 - \exp(-x^\beta)$ for $x$, we have

$$x = (-\log(1 - u))^{1/\beta}.$$  Then

- Generate U from Uniform (0,1)
- Set $X = (-\log(1 - U))^{1/\beta}$ or $X = (-\log(U))^{1/\beta}$. 

Cauchy distribution

- If $X$ follows a Cauchy distribution with parameter $\mu$ and $\sigma$, then it can be shown that

\[
F(x) = \frac{1}{2} + \frac{1}{\pi} \tan^{-1}\left(\frac{x - \mu}{\sigma}\right)
\]

for $x$ in $(-\infty, \infty)$ Note: \(f(x) = \frac{1}{\pi \sigma (1 + (\frac{x-\mu}{\sigma})^2)}\).

Solving \(u = F(x) = \frac{1}{2} + \frac{1}{\pi} \tan^{-1}\left(\frac{x-\mu}{\sigma}\right)\), we have
\[x = \sigma \tan[\pi(u - 0.5)] + \mu.\]

- Generate U from Uniform (0,1)
- Set $X = \sigma \tan[\pi(U - 0.5)] + \mu.$
**Algorithm to generate a normal random variable**

Box-Muller Algorithm

- Generate $U_1$ and $U_2$ from Uniform $(0,1)$
- Set $\theta = 2\pi U_1$ and $R = (-2\log U_2)^{1/2}$
- Set $X = R \cos(\theta)$ and $Y = R \sin(\theta)$

Then $X$ and $Y$ are independent standard normal variables. Often, only $X$ or $Y$ is used.
Algorithm to generate a normal random variable

Polar algorithm (Modified Box-Muller algorithm)

- Generate $U_1, U_2 \sim \text{Uniform}(-1, 1)$ until $U_1^2 + U_2^2 < 1$
- Set $W = U_1^2 + U_2^2$ and $c = \sqrt{-2\log(W)/W}$
- Set $X = cU_1$ and $Y = cU_2$

Then $X$ and $Y$ are independent standard normal variables.

Remark: Polar algorithm uses rejection to avoid calculating two trigonometric functions and so is usually substantially faster compared to Box-Muller algorithm. However, using $(2^n)$ uniform random numbers will not generate $(2^n)$ standard normal random numbers.
Generate a random variable from other r. v.

Cauchy distribution

- If $Y$ and $Z$ are independent and follow $N(0,1)$, then $X = Y/Z$ follows a Cauchy$(0,1)$ distribution

- If $Y \sim N(\mu, \sigma^2)$ and $Z \sim N(0,1)$, and are independent, then $X = Y/Z$ follows a Cauchy $(\mu, \sigma^2)$ distribution
Generate a random variable from other r. v.

Chi-square distribution

- If $Y$ follows a normal distribution, then $X = Y^2$ follows a Chi-square distribution with 1 degree of freedom.

- If $Y_1, Y_2, \cdots, Y_n$ are independent and identically distributed standard normal variables, then

$$X = Y_1^2 + Y_2^2 + \cdots + Y_n^2$$

follows $\chi^2(n)$, a Chi-square distribution with $n$ degrees of freedom.
Generate a random variable from other r. v.

Student’s t-distribution

- If $Y \sim N(0, 1)$ and $Z \sim \chi^2(p)$, then

$$X = \frac{Y}{\sqrt{Z/p}}$$

follows a Student’s t distribution with $p$ degrees of freedom
Generate a random variable from other r. v.

F distribution

- If $Y \sim \chi^2(m)$ and $Z \sim \chi^2(n)$, then
  \[
  X = \frac{Y/m}{Z/n}
  \]
  follows a F distribution with degrees of freedom $m$ and $n$. 
**Function to generate uniform distribution r. v.**

To generate random numbers from Uniform \((a,b)\)

\[
f(x) = \frac{1}{(b - a)} \quad \text{for} \quad a < x < b
\]

In R

\[
\begin{align*}
> & \# \text{ Generate uniform r. v.} \\
> & n=100 \\
> & a=0 \\
> & b=100 \\
> & x=\text{runif}(n,a,b) \\
> & x
\end{align*}
\]
**Function to generate uniform distribution r. v.**

In SAS

```sas
data unif;
seed=1234;
n=100;a=0;b=10;
do i=1 to n;
x=a+(b-a)* ranuni(seed);
output;
end;
keep x;
run;
```
Function to generate Normal distribution r. v.

To generate random numbers from Normal \((\mu, \sigma^2)\) for \(-\infty < x < \infty\)

\[
f(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right) \text{ for } -\infty < x < \infty
\]

In R
> # Generate normal r. v.
> n=100
> mu=0
> sigma=1
> x=rnorm(n,mean=mu,sd=sigma)
> x
Function to generate Normal distribution r. v.

In SAS

data norm;
seed=1234;
N=100;mu=0;sigma=1;
do i=1 to n;
x=mu+sigma*rannor(seed);
output;
end;
keep x;
run;
Function to generate Expo distribution r. v.

To generate random numbers from Exponential ($\lambda$)

$$f(x) = \frac{1}{\lambda} e^{-\frac{x}{\lambda}} \text{ for } x > 0$$

In R

```R
# Generate exponential r. v.
> n=100
> lambda=5
> x=rexp(n,rate=lambda)
> x
```
Function to generate Expo distribution r. v.

In SAS

```
data expno;
  seed=1234;
  n=100;lambda=5;
  do i=1 to n;
    x=lambda*ranexp(seed);
    output;
  end;
  keep x;
run;
```
Function to generate Gamma distribution r. v.

To generate random numbers from Gamma \((\alpha, \beta)\)

\[
f(x) = \frac{1}{\beta^\alpha \Gamma(\alpha)} x^{\alpha - 1} \exp\left(-\frac{x}{\beta}\right) \text{ for } x > 0
\]

In R

\>
\> # Generate gamma r. v.
\>
\> n=100
\>
\> alpha=1
\>
\> beta=2
\>
\> x=rgamma(n,shape=alpha,scale=beta)
\>
\> x
Function to generate Gamma distribution r. v.

In SAS

data gammano;
seed=1234;
n=100;alpha=1;beta=2;
do i=1 to n;
x=beta*rangam(seed,alpha);
output;
end;
keep x;
run;
Function to generate $\chi^2$ distribution r. v.

To generate random numbers from $\chi^2(p)$

$$f(x) = \frac{1}{2^{\frac{p}{2}} \Gamma\left(\frac{p}{2}\right)} x^{\frac{p}{2}-1} \exp\left(-\frac{x}{2}\right) \text{ for } x > 0$$

In R

> # Generate Chi-square r. v.
> n=100
> p=10
> x=rchisq(n,df=p)
> x
Function to generate $\chi^2$ distribution r. v.

In SAS

```sas
data chisqno;
seed=1234;
n=100;df=10;alpha=df/2;
do i=1 to n;
  x=2*rangam(seed,alpha);
output;
end;
keep x;
run;
```
Function to generate Beta distribution r. v.

To generate random numbers from Beta \((\alpha, \beta)\)

\[
f(x) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha-1}(1 - x)^{\beta-1} \text{ for } 0 < x < 1
\]

In R

```r
> # Generate Beta r. v.
> n=100
> a=2
> b=3
> x=rbeta(n,shape1=a,shape2=b)
> x
```
Function to generate Beta distribution r. v.

In SAS

data betano;
seed=1234;
n=100;alpha=2;beta=3;
do i=1 to n;
y1=rangam(seed,alpha);
y2=rangam(seed,beta);
x=y1/(y1+y2);
output;
end;
keep x;
run;
**Function to generate t-distribution r. v.**

To generate random numbers from $t(k)$

$$f(x) = \frac{\Gamma\left(\frac{k+1}{2}\right)}{\Gamma\left(\frac{k}{2}\right)} \frac{1}{\sqrt{k\pi}} \frac{1}{\left(1 + \frac{x^2}{k}\right)^{\frac{k+1}{2}}} \text{ for } -\infty < x < \infty$$

In R

```r
# Generate t r. v.
> n=100
> k=5
> x=rt(n,df=k)
> x
```
Function to generate t distribution r. v.

In SAS

```sas
data tno;
seed=1234;
n=100;df=5;alpha=df/2;
do i=1 to n;
y1=rannor(seed);
y2=rangam(seed,alpha);
x=y1/sqrt(y2/df);
output;
end;
keep x;
run;
```
Function to generate F-distribution r.v.

To generate random numbers from F(m,n)

\[ f(x) = \frac{\Gamma \left( \frac{n_1 + n_2}{2} \right)}{\Gamma \left( \frac{n_1}{2} \right) \Gamma \left( \frac{n_2}{2} \right)} \left( \frac{n_1}{n_2} \right)^{\frac{n_1}{2}} x^{\frac{n_1}{2} - 1} \left(1 + \frac{n_1}{n_2} x\right)^{-\frac{n_1 + n_2}{2}} \text{ for } 0 < x < \infty \]

In R

> # Generate F r. v.
> n=100
> n1=5
> n2=10
> x=rf(n,df1=n1,df2=n2)
> x
Function to generate F distribution r.v.

In SAS

```sas
data fno;
seed=1234;
n=100; df1=5; df2=10;
do i=1 to n;
y1=2*rangam(seed,df1/2);
y2=2*rangam(seed,df2/2);
x=(y1/df1)/(y2/df2);
output;
end;
keep x;
run;
```
Function to generate Binomial distribution r.v.

To generate random numbers from Binomial(n,p)

\[ f(x) = \binom{n}{x} p^x (1 - p)^{n-x} \text{ for } x = 0, 1, 2, \cdots, n \]

In R

> # Generate Binomial r.v.
> nn=100
> n=10
> p=0.3
> x=rbinom(100,size=n,prob=p)
> x
Function to generate Binomial distribution r.v.

In SAS

data binomno;
seed=1234;
ns=100;n=10;p=0.3;
do i=1 to ns;
x=ranbin(seed,n,p);
output;
end;
keep x;
run;
**Function to generate Poisson distribution r.v.**

To generate random numbers from poisson(\(\lambda\))

\[
f(x) = \frac{e^{-\lambda} \lambda^x}{x!} \quad \text{for } x = 0, 1, 2, \ldots
\]

In R

```r
> # Generate Poisson r. v.
> n=100
> lambda=3
> x=rpois(100,lambda)
> x
```
Function to generate Poisson distribution r.v.

In SAS

```sas
data poisno;
seed=1234;
n=100;lambda=3;
do i=1 to n;
x=ranpoi(seed,lambda);
output;
end;
keep x;
run;
```
Function to generate Hypergeometric r.v.

To generate random numbers from Hypergeometric distribution 
\((n, N, S)\)

\[
f(x) = \frac{\binom{S}{x} \binom{N - S}{n - x}}{\binom{N}{n}} \text{ for } x = 0, 1, 2, \cdots, \min(n, S)
\]

In R

\[
> \# \text{ Generate hypergeometric r.v.} \\
> \text{ns=100; n=10} \\
> \text{S=20; N=50} \\
> \text{x=rhyper(ns,S,N,n)} \\
> \text{x}
\]
**Function to generate Nega-Binomial distr. r.v.**

To generate random numbers from $\text{NBinom}(r, p)$

$$f(x) = \binom{r + x - 1}{x} p^r (1 - p)^x \text{ for } x = 0, 1, 2, \cdots$$

In R

```r
> # Generate Negative Binomial r. v.
> n=100
> r=10
> p=0.3
> x=rnbinom(n,size=r,prob=p)
> x
```