Simulation of Random Variables

(Gustavo de los Campos, gustavoc@msu.edu)

Monte Carlo (MC) Methods play a central role in statistical analysis, including power and type-I error rate analysis as well as Likelihood and Bayesian inference. The basic idea underlying MC methods is very simple: **we use samples** from the distribution of random variables to evaluates features of that distribution (e.g., expected value, variance, quantiles) that are very difficult or cannot be derived analytically.

The first step in a MC study is to sample random variables from the desired distribution. This note provides an overview of basic algorithms for simulating random variables and provide examples using R.

1. Random numbers & seeds

Computers do not generate random numbers; computer algorithms are deterministic. However, there are well-established algorithms that generate sequences of numbers that resemble IID sequences from uniform random variables. A random number generator implements an algorithm that, starting from a seed (an integer), generates a deterministic sequence of numbers that resemble IID samples from uniform distributions. Given the seed the sequence is deterministic but we regarded it as random. Try in R the following code

```
set.seed(1923)
runif(5)
runif(4)
set.seed(1923)
runif(5)
runif(4)
```

1. Transformations of random variables

Suppose that $Y = g(X)$ where X is a random variable with pdf $f_X(x)$ and $g(X)$ is a one-to-one monotonic, continuous, and differentiable map, then

$$f_Y(y) = f_X(X(y))\left|\frac{dX(y)}{dy}\right| \qquad [1]$$

where $X(y) = g^{-1}(y)$.

Here $Y = g(X)$ is a transformation and $X(Y) = g^{-1}(Y)$ is the inverse transformation.

Example: sampling an exponential random variable.

The pdf of an exponential random variable is given by

$$f_Y(y) = \lambda e^{-\lambda y} \quad \lambda, y > 0$$

Result: if $X \sim U(0,1)$ and $Y = -\frac{\log(X)}{\lambda}$ then $Y \sim Exponential(\lambda)$.

Proof:

(1) here $Y = g(X) = -\frac{\log(X)}{\lambda}$; therefore $X = g^{-1}(Y) = e^{-\lambda Y}$.
(2) If $X \sim U(0,1)$=> $f_X(x) = 1(0 \leq x \leq 1)$, then using [1]
(3) $f_Y(y) = 1(g^{-1}(Y) \in [0,1]) \left|\frac{dX(y)}{dy}\right| = 1(y > 0)e^{-\lambda y}|-\lambda| = \lambda e^{-\lambda y} \quad y > 0$

Generalization: The above results generalizes to random vectors, in which case $\left|\frac{dX(y)}{dy}\right|$ needs to be replaced with the absolute value of the determinant of the matrix of first derivatives of the inverse-transformation evaluated at y. Note: this result holds for one-to-one maps. It would not hold, for instance for $g(X) = X^2$.

Example:

```
lambda=5;n=1e5
x=runif(n)
y=-log(x)/lambda
hist(y,n)
mean(y); 1/lambda
var(y); 1/(lambda^2)
y2=rexp(n=n,rate=lambda)
par(mfrow=c(1,2))
hist(y,50); hist(y2,50)
```

[See map of distributions from Casella and Berger in our website]

2. Sampling Bernoulli Random Variables

A Bernoulli random variable (X) with success probability θ can be sampled as follows: sample `U~uniform(n=1,min=0,max=1),` set X to be 0 if U<p, 1 otherwise, that is `X=ifelse(U<`θ`,1,0).`

Why does it work? $p(X=0)=p(U<\theta)=\int_0^\theta 1d\theta=\theta$, that is the cdf of the uniform is the 45 degree line in the unit square, that is `p=qunif(p)` for all values of p in [0,1].

4. The inverse probability method

Continuous RVs can be generated from uniform RV using the inverse probability method. Recall that the cdf of a RV is given by $p(X<q)=\Phi(q)=\int_{-\infty}^{q}p(x)dx$. The cdf maps from values of the random variable to probabilities. The inverse cdf (or quantile function) maps from probabilities to quantiles, that is $\Phi^{-1}(\theta)=q:\Phi(q)=\theta$. In R the cdf of a random variable has prefix 'p' (e.g., pnorm) and the inverse cdf has prefix 'q' (e.g., qnorm).

The inverse probability method generates a random draw for an arbitrary RV, with known inverse-cdf, by: (i) drawing a uniform [0,1] RV (say U~runif(1)) and then evaluating the inverse cdf of X at U. The following example illustrates this:

```
mu=5
SD=4
N=1e5

u=runif(n)
Y1=qnorm(u, sd=SD,mean=mu)
Y2=rnorm(n,sd=SD,mean=mu)
# Compare the empirical quantiles of Y1 and Y2…
```

Why does it work? Because the CDF has uniform distribution, to see this note that by definition $Y\in[0,1]$. Furthermore, if $Y=\Phi(X)$, then $P(Y<k)=P(\Phi_x(X)<k)=P(X<\Phi_X^{-1}(k))=\Phi_X\left(\Phi^{-1}(k)\right)=k$. The only RV that has support on the [0,1] and has a CDF $P(Y<k)=k$ is the uniform!

Here are a few additional examples:

```
N=1e5

 X=rexp(n=N,rate=4)
 hist(pexp(X,rate=4),50)

 X=rnorm(n=N,mean=12,sd=5)
 hist(pnorm(X,mean=12,sd=5),50)
```

3. Composition Sampling

So far, we have considered univariate distributions. We now turn into sampling from multivariate distributions.

Recall that any multivariate distribution can be factored out as the product of marginal times conditional distributions, that is (note that any order of factorization is valid):

$$p(X,Y,Z) = p(X)p(Y|X)p(Z|X,Y)$$

This implies that we can draw a sample of the vector (X,Y,Z) by sampling $X \sim p(X)$, this yields a realization x_i which we then use in the conditional distribution of Y given X to sample Y, that is, sample $Y \sim p(Y|X = x_i)$, this in turn yields a realization, y_i, which we then use in the last conditional distribution to sample $Z \sim p(Z|X = x_i, Y = y_i)$.

Example 1: let X, Y be Bernoulli RVs with the following joint distribution

	Y=0	Y=1
X=0	0.1	0.1
X=1	0.2	0.6

Task: develop a composition sampling algorithm to generate 1000 samples of (X,Y) from its joint distribution.

Box 1: outline of composition sampling algorithm:

Preliminaries:

- Derive from the table p(Y=1|X=0) and p(Y=1|X=1) and store these in a vector
- Initialize vectors (X and Y) with length equal to the desired sample size (e.g., N=10000), e.g., X<-rep(NA,N)

Sampling:

- In a for loop, with i in 1 to N:
  - Sample X[i] from its marginal distribution
  - Sample Y[i] from its conditional distribution p(Y=1|X=X[i])

Did the algorithm work? To assess this check:

mean(X) and mean(Y) should be close to the marginal probability
mean((X==0)&(Y==0)), mean((X==1)&(Y==0)), mean((X==0)&(Y==1)), and mean((X==1)&(Y==1)) should all be close to the joint probabilities in Example 1.

4. Gibbs Sampler

In many cases conditional distributions such as $p(Z|X,Y)$ may not have a closed form. In those cases, composition sampling is not feasible. An alternative strategy is to use a Gibbs sampler. In a Gibbs sampler a draw from the joint distribution of a set of random variables, e.g., (X,Y,Z) is obtained by recursively sampling from fully conditional distributions, that, by sampling from:

(i) $X_i \sim p(X|Y = y_{i-1}, Z = z_{i-1}) => x_i$
(ii) $Y_i \sim p(Y|X = x_i, Z = z_{i-1}) => y_i$ and then,
(iii) $Z_i \sim p(Y|X = x_i, Y = y_i) => z_i$

Thus, the algorithm can be summarized as follows:

(0) Initialized with valid values (i.e., values that have non-zero probability)
(1) for i in 1-n, sample from (i)-(ii)-(iii) to complete a draw (x_i, y_i, z_i)

The resulting samples can be regarded as samples from the joint distribution.

Box 2: outline of Gibbs sampling algorithm:

Preliminaries:

- Derive from the table the four conditional probabilities needed: p(Y=1|X=0), p(Y=1|X=1), p(X=1|Y=0), p(X=1|Y=1), store them in a matrix.
- Initialize vectors (X and Y) with length equal to the desired sample size (e.g., N=10000), e.g., X<-rep(NA,N)
- Initialize X[1] with a 0 or 1 value (could be a sample from it's marginal distribution
- Sample Y[1] from p(Y|X=X[2])

For loop:

- In a for loop, with i in 2 to N:
  - Sample X[i] from p(X| Y=Y[i-1])
  - Sample Y[i] from p(Y| X=X[i-1])

Effective number of independent samples: Samples collected using a Gibbs sampler are not mutually independent, that is, (x_i, y_i, z_i) is not independent from $(x_{i-1}, y_{i-1}, z_{i-1})$.

```
plot(X[1:100],type='o')
library(coda)
length(X)        # total number of samples
effectiveSize(X) # approximate number of independent samples
```

5. Multivariate Normal Distribution

The density function of a multivariate normal (MVN) random vector $\boldsymbol{X} = \left(X_1, \ldots, X_p\right)'$ with mean vector $E[\boldsymbol{X}] = \boldsymbol{\mu}' = \left(\mu_1, .., \mu_p\right)'$ and covariance matrix

$$Cov[\boldsymbol{X}] = \boldsymbol{\Sigma} = \begin{bmatrix} \Sigma_{11} & \cdots & \Sigma_{1p} \\ \vdots & \ddots & \vdots \\ \Sigma_{p1} & \cdots & \Sigma_{pp} \end{bmatrix}$$

takes the following form

$$f(\boldsymbol{x}) = |\boldsymbol{\Sigma}|^{-\frac{1}{2}} Exp\{(\boldsymbol{x} - \boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\}$$

where:

- $\Sigma_{ij} = Cov(X_i, X_j)$, or if *i=j*, $\Sigma_{ii} = Var(X_i)$,
- $(\boldsymbol{x} - \boldsymbol{\mu}) = \left(x_1 - \mu_1, \ldots, x_p - \mu_i\right)'$ is a vector whose entries are deviations of the realized values from their corresponding means,
- $(\boldsymbol{x} - \boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}) = \sum_{i=1}^{p}\sum_{j=1}^{p}(x_i - \mu_i)\left(x_j - \mu_j\right)\Sigma^{ij}$ is a 'quadratic form', a weighted sum of squares. Here Σ^{ij} is the (i^{th},j^{th}) entry of $\boldsymbol{\Sigma}^{-1}$,
- and $|\boldsymbol{\Sigma}|$ denotes the determinant of $|\boldsymbol{\Sigma}|$. Covariance matrices are positive-definite; therefore, the determinant is guaranteed to be positive.

Properties of the Multivariate Normal Distribution

i) All the marginal distributions are normal, specifically, $X_i \sim N(\mu_i, \Sigma_{ii})$.

ii) Linear combinations of MVN vectors follow MVN distributions. In particular: $\boldsymbol{Y} = \boldsymbol{\alpha} + \boldsymbol{T}\boldsymbol{X}$ follows a MVN with mean $E[\boldsymbol{Y}] = \boldsymbol{\alpha} + \boldsymbol{T}\boldsymbol{\mu}$ and (co)variance matrix $Cov[\boldsymbol{Y}] = \boldsymbol{T}\boldsymbol{\Sigma}\boldsymbol{T}'$

iii) Conditional distributions are also MVN. To illustrates this property, consider a partition of the MVN vector $\boldsymbol{X}$ into two sub-vectors, that is: $\boldsymbol{X} = (\boldsymbol{X}_1', \boldsymbol{X}_2')'$. The corresponding partition of the mean vector and (co)variance matrix are

$$E[\boldsymbol{X}] = \boldsymbol{\mu} = \begin{bmatrix} E(\boldsymbol{X}_1) \\ E(\boldsymbol{X}_2) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix} \text{ and } Cov[\boldsymbol{X}] = \begin{bmatrix} Var(\boldsymbol{X}_1) & Cov(\boldsymbol{X}_1, \boldsymbol{X}_2') \\ Cov(\boldsymbol{X}_2, \boldsymbol{X}_1') & Var(\boldsymbol{X}_2) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}$$

The conditional distribution of $\boldsymbol{X}_2$ given $\boldsymbol{X}_1$ is MVN with the following mean and variance-covariance matrices

$$E(\boldsymbol{X}_2|\boldsymbol{X}_1) = \boldsymbol{\mu}_2 + \boldsymbol{B}_{21}(\boldsymbol{X}_1 - \boldsymbol{\mu}_1) \text{ and } Cov(\boldsymbol{X}_2|\boldsymbol{X}_1) = \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}$$

Above, $\boldsymbol{B}_{21}$ is a matrix of regression coefficients which is equal to $\boldsymbol{B}_{21} = \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}$.

Simulating MVN random variables

The functions `mvrnorm()` of the MASS package generate random draws from MVN distributions. How are these samples generated? The sampling scheme uses the fact that linear combinations of MVN random variables follow MVN distribution. The algorithm works as follows: (a) sample p IID N(0,1) RVs, (b) take a linear combination of these RVs that would generate the target mean and (co)variance matrix.

Example: Using Cholesky Decomposition

```
mu=c(2,10,3) # mean vector
S=diag(c(1,2,1.2))  # variances
S[1,2]=S[2,1]=.5    # covariance 1,2
S[1,3]=S[3,1]=.2    # covariance 1,3
S[3,2]=S[2,3]=.4    # covariance 2,3

n=1e6

L=t(chol(S))        # lower-triangular Cholesky S=LL'

z=rnorm(length(mu))
x=mu+L%*%z  # note E[x]=mu+LE[z]=mu and Cov(x)=LCov(z)L'=LL'=X

# Now let's generate 5000 IID samples
 Z=matrix(nrow=5000,ncol=length(mu),rnorm(5000*length(mu)))
 X=matrix(nrow=nrow(Z),ncol=ncol(Z),NA)
 for(i in 1:nrow(X)){
     X[i,]=mu+ L%*%Z[i,]
 }
 colMeans(X) # compare with mu
 cov(X)      # compare with S

# to avoid a loop you can also use this
 W=Z%*%t(L)
 for(i in 1:length(mu)){ W[,i]=W[,i]+mu[i] }

 var(W)
 colMeans(W) ]
```

Note: Another strategy would be to use either composition sampling or Gibbs sampling. These algorithms will require to sample from conditional distributions which can be derived using the formulae presented above.