**Introduction to Simulation of Random Variables**

([gustavoc@msu.edu)](mailto: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) are very difficult or cannot be derived analytically.

The first step in a MC study is to sample random variables from the desired distribution. In this note I review a few basic algorithms for simulating random variables and provide examples using R.

**1. Random numbers & seeds**

Computers do not generate random numbers. Everything in an algorithm is deterministic. However, there are well-established algorithms that will 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 where X is a random variable with pdf and is a one-to-one monotonic map, then

[1]

where .

Here is a transformation and is the inverse transformation.

**Example**: sampling an exponential random variable.

The pdf of an exponential random variable is given by

*Result*: if and then .

Proof:

1. here ; therefore .
2. If => , then using [1]

**Generalization**: The above results generalizes to random vectors, in which case needs to be replaced with the absolute value of the Jacobian of the transformation (i.e., the absolute value of the determinant of the matrix of first derivatives of the inverse-transformation) evaluated at . Note: this result holds for one-to-one maps. It would not hold, for instance for .

**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 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<,1,0).

Why does it work? , 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**

The previous method illustrates how uniform RVs can be used to sample discrete RVs. But what about continuous RVs?

Any continuous RV can be generated from a uniform RV using the inverse probability method. Recall that the cdf of a RV is given by . The cdf maps from values of the random variable to probabilities. The inverse cdf (or quantile function) maps from probabilities to quantiles, that is . 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 generate 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 the inverse probability method works? It works because the cdf(X) follows a uniform distribution no matter what the distribution of X is. Here are a few 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):

This implies that we can draw a sample of the vector by sampling , this yields a realization which we then use in the conditional distribution of Y given X to sample Y, that is, sample , this in turn yields a realization, , which we then use in the last conditional distribution to sample .

**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. Let’s sample X from p(X) and then sample Y from p(Y|X). From the table, and using p(Y|X)=p(Y,X)/p(X) we have: p(Y=1|X=0)=0.1/(0.1+0.1)=1/2 and p(Y=1|X=1)=0.6/(0.2+0.6)=3/4

The following example illustrates this:

n=100000

X=rep(NA,n)

Y=rep(NA,n)

pX1=0.8

pY1\_X0=1/2

pY1\_X1=3/4

# initial values

X[1]=1

Y[1]=0

# Sampling

for( i in 1:n){

X[i]=ifelse(runif(1)<pX1,1,0)

if(X[i]==0){

Y[i]=ifelse(runif(1)<pY1\_X0,1,0)

}else{

Y[i]=ifelse(runif(1)<pY1\_X1,1,0)

}

}

**4. Gibbs Sampler**

In many cases conditional distributions such as 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., is obtained by recursively sampling from fully conditional distributions, that, by sampling from:

1. and then,

Thus, the algorithm can be summarized as follows:

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

The resulting samples can be regarded as samples from the joint distribution. However, they are not mutually independent, that is, is not independent from and so on…

Let’s sample from the previous table using Gibbs. We need to find p(X|Y) and p(Y|X). From the table we have: p(X=1|Y=0)=0.2/(0.1+0.2) and p(X=1|Y=1)=0.6/0.7.

## Gibbs Sampler

  n=100000

  X=rep(NA,n); Y=rep(NA,n)

# conditional probabilities

  pY1\_X0=1/2 ; pY1\_X1=3/4

  pX1\_Y0=2/3 ; pX1\_Y1=0.6/0.7

# initial values

  X[1]=1 ; Y[1]=0

for( i in 2:n){

    # Sampling Y given X

    if(X[i-1]==0){

    Y[i]=ifelse(runif(1)<pY1\_X0,1,0)

    }else{

  Y[i]=ifelse(runif(1)<pY1\_X1,1,0)

    }

    # Sampling X given Y

    if(Y[i]==0){

    X[i]=ifelse(runif(1)<pX1\_Y0,1,0)

    }else{

  X[i]=ifelse(runif(1)<pX1\_Y1,1,0)

    }

}

**5. Multivariate Normal Distribution**

The density function of a multivariate normal (MVN) random vector with mean vector and covariance matrix

takes the following form

where:

* , or if *i=j,*  ,
* is a vector whose entries are deviations of the realized values from their corresponding means,
* is a ‘quadratic form’, a weighted sum of squares. Here is the (*ith,jth*) entry of
* and denotes the determinant of . 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, .

ii) Linear combinations of MVN vectors follow MVN distributions. In particular: follows a MVN with mean and (co)variance matrix

iii) Conditional distributions are also MVN. To illustrates this property, we introduce a partition of the MVN vector into two sub-vectors, that is: . The corresponding partition of the mean vector and (co)variance matrix are

and

The conditional distribution of given is MVN with the following mean and variance-covariance matrices

and

Above, is a matrix of regression coefficients which is equal to .

**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.

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=Z%\*%t(L)

  for(i in 1:length(mu)){ X[,i]=X[,i]+mu[i] }

var(X) # empirical (co)variance matrix should be close to S

colMeans(X) # empirical means, should be close to mu