

Probabilities & Data

Week 12: Monte Carlo Simualtions

DR NICK FEWSTER-YOUNG

Topics to be covered

- *Probability Through Simulation*
- *Understanding Limit Theorems through Simulation*
- *Approximation methods for integration*
- *The Accept-Reject Technique*
- *Application to Bayesian Inference*

Motivation

Simulation of observations from random experiments has always been of profound interest. Generation of random numbers has always been crucial for statisticians, both for theory and applications. The emergence of the computer and its speed has greatly benefited from the endurance of random numbers.

The applications are extensive and apply to approximation of numbers and values to proving theories on the small scale work in the large practical setting by being random and unpredictable like the real world with an ounce of certainty.

Generation of random numbers has a lot of applications in real life. The 14-digit number that is written on your mobile recharge voucher, numbers on be-lucky-to-win scratch cards, match your mobile numbers on those given in a series of draws of newspapers, etc., are all some scenarios which require generation of random numbers.



Probability through simulation

It is possible to solve some problems using a simulation approach. We will explore a few problems as examples to see how this can be achieved 😊.

- The Birthday Problem
- Buffon's Needle Problem

The Birthday Problem

The number of cells (dates) is $n = 365$. Alternatively imagine them as 365 balls of different colors in an urn.

For a fixed k , draw a ball at random and note its color before replacing it in the urn. The urn is shaken well to ensure that the balls are again in random order. This exercise is repeated k times, and then record if two balls of the same color are repeated.

If yes, note 1, if no, note 0. The exercise up to this point is repeated a large number of times, say B . The average of B can then be taken as the probability of having at least two of the same birthdays in a group of k people. This approach will give us the simulated birthday probabilities.

Birthday Problem R code

```
## This R program for Simulated Birthday Probabilities
```

```
> diy <- 1:365; B <- 1000; bins <- c(2,5,10,20,30,40,50)
```

```
> simprob <- 1:length(bins)*0
```

```
> for(j in 1:length(bins)) {
```

```
+   y<-1:B*0;
```

```
+   for(i in 1:B) {
```

```
+     x <- sample(diy,bins[j],replace=TRUE)
```

```
+     if(dim(table(x))<bins[j]) y[i] <- 1
```

```
+   }
```

```
+   simprob[j]<-mean(y)
```

```
+ }  
> simprob
```

```
[1] 0.003 0.019 0.126 0.401 0.713 0.892 0.970
```

What is happening

The sample function, is a draw of k , **R** variable bins, balls is carried out from diy , days in year, with replacement (option `replace=TRUE`).

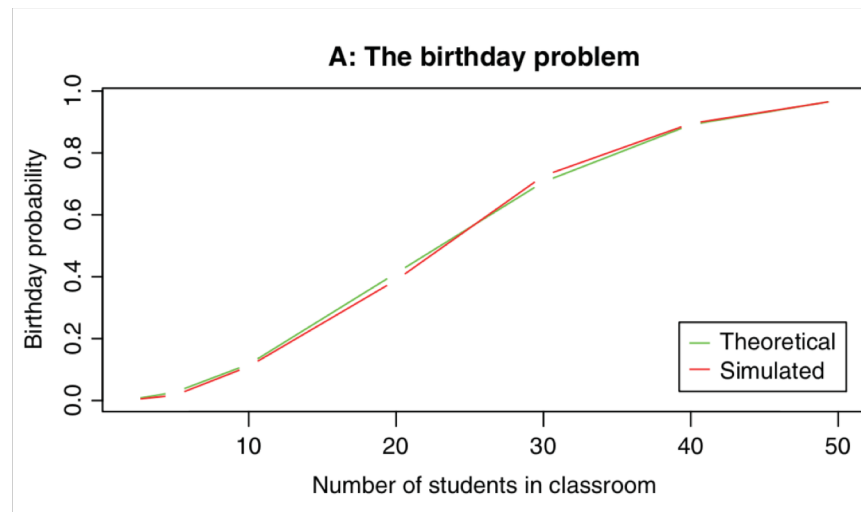
To check if any birthday is selected more than once, we check if `dim(table(x))<bins[j]` holds true.

If the condition is satisfied, then `y[i]` is assigned a value of 1, which is otherwise zero.

Size of k	Theoretical Probability	Simulated Birthday Probability
2	0.0027	0.003
5	0.0271	0.019
10	0.1169	0.126
20	0.4114	0.401
30	0.7063	0.713
40	0.8912	0.892
50	0.9704	0.970

Birthday Problem

Here is a simulation and the actual probabilities plotted next to each other.




Buffon's Needle Problem

Georges-Louis Leclerc, Comte de Buffon posed a problem in the 18th century, in what is now popularly referred to as *Buffon's needle problem*.

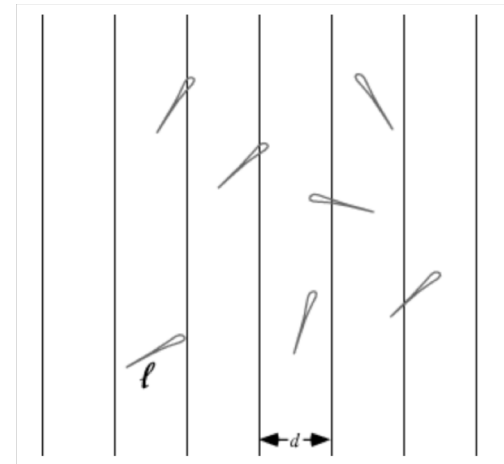
Consider equally spaced strips of parallel lines and drop a needle onto the surface. The lines are at a distance d apart from each other. Shonkwiler and Mendivil (2010) have explained the Buffon's needle problem as the first experiment where simulation has been used to estimate the π value.

Assume that the length of the needle l is less than d . Let X denote the distance from the center of the line to the next nearest line. Let θ be the acute angle that the needle makes with respect to the parallel lines.



Then $0 \leq X \leq d/2$ and $0 \leq \theta \leq \pi/2$. The probability of the needle crossing a line

$$P(\text{needle crossing a line}) = \frac{2l}{\pi d}$$



In the next simulation study we will examine the number of times the needle crosses a line and use the *empirical probability* for the estimation of π in the above expression. The required R program now follows.

Buffon's R code

```
# L: Needle Length; d: Distance between two lines; Hn: of hits in n throws  
# Simulating if needle crosses the line  
# We needed simulated values of sin(theta). Simulate values in the interval 0-1 and use it as  
sin(theta)  
> l <- 10; d <- 25; n <- 1e5  
> theta <- runif(n,0,22/14)  
> sinTheta <- sin(theta)  
> x <- runif(n,0,d/2)  
> SimHits <- ifelse((x<=(l/2)*sinTheta),1,0)  
# We can now estimate the value of pi  
> piEstimate <- 2*l/(mean(SimHits)*d)
```

Conclusion

[1] 3.158934

That is pretty close to π !!!

We are 2 decimals places correct here!

If we kept going then we quickly get a number which is more accurate.

Understanding Limit Theorems

When we first came across some limit theorems, they were presented to us as theories and by using the expectation and variance of the random sample X_1, X_2, \dots .

Now we will revisit some of the limit theorems and a simulation approach will be taken to generate the data, that is, instead of looking at the quantities related with the random sample, we will use the pseudo random observations x_1, x_2, \dots and estimated sample means and variance from there.

The Weak Law of Large Numbers

Understanding the Weak Law of Large Numbers (WLLN).

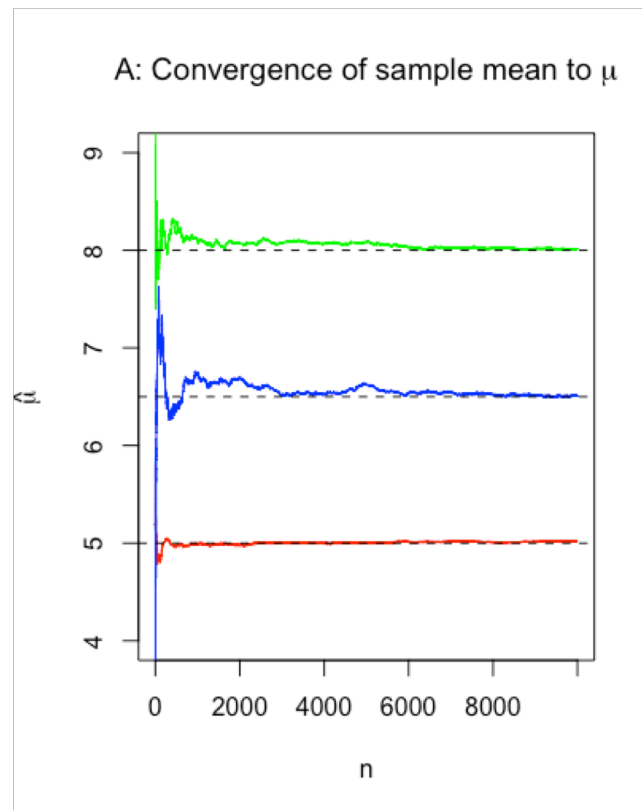
To understand the WLLN through simulation, consider three distributions Normal, Exponential, and Gamma, for which the variance is theoretically known to be finite.

Then generate 10 000 values from each of these distributions with means respectively at 5, 6.5, and 8. The next piece of code along with the graph gives a clear understanding of how true the WLLN holds in reality

R code

```
> n <- 10000;  
> xnorm <- rnorm(n,5,1); xexp <- rexp(n,1/6.5)  
> xgamma <- rgamma(n,4,1/2)  
> plot(1:n,cumsum(xnorm[1:n])/1:n, type="l",xlab="n", ylab=expression(hat(mu)),  
main=expression(paste("A: Convergence of sample mean to ", mu)), col = "red",ylim=c(4,9))  
> lines(1:n,cumsum(xexp[1:n])/1:n,type="l",col="blue")  
> lines(1:n,cumsum(xgamma[1:n])/1:n,type="l",col="green")  
> abline(h=c(5,6.5,8),lty=2)
```

Convergence Pictures



Central Limit Theorem

The CLT is a very powerful technique theorem.

The conditions were stated when identically and independent sample averages converged to the standard Normal distribution. We will consider different distributions which are significantly different from the Normal distribution, and then observations are simulated from it.

Then averages of the simulated observations are computed and the module is repeated, the (simulation) experiment, a large number of times, and finally a histogram is plotted of the standardized values and the standard normal curve is fitted to inspect if the convergence is true or not!

Central Limit Theorem

Discrete Uniform Distribution. If we throw a fair dice with N sides, it is known that the probability distribution will be a discrete uniform distribution. This produces a flat probability mass function. That is, we assume X is from a discrete uniform distribution $(1, \dots, N)$, $i = 1, 2, \dots, n$.

Define $\bar{X} = \sum_i^n \frac{X_i}{n}$.

As n increases, the CLT tells us that the asymptotic distribution of

$$\frac{\bar{X} - E(X)}{\sqrt{\text{Var}(X)}} \text{ is } N(0, 1).$$

Fix $N = 100$ and consider simulating a batch of 20 observations first and note the average. This step will then be repeated a large number of times, $B = 1000$. The 1000 averages will then be standardized and a histogram will be obtained to examine the appropriateness of the CLT.

R code

```
> xmean <- NULL
> B <- 1000
> for(i in 1:B){xmean[i]=mean(sample.int(100,size=200,replace=TRUE)) }

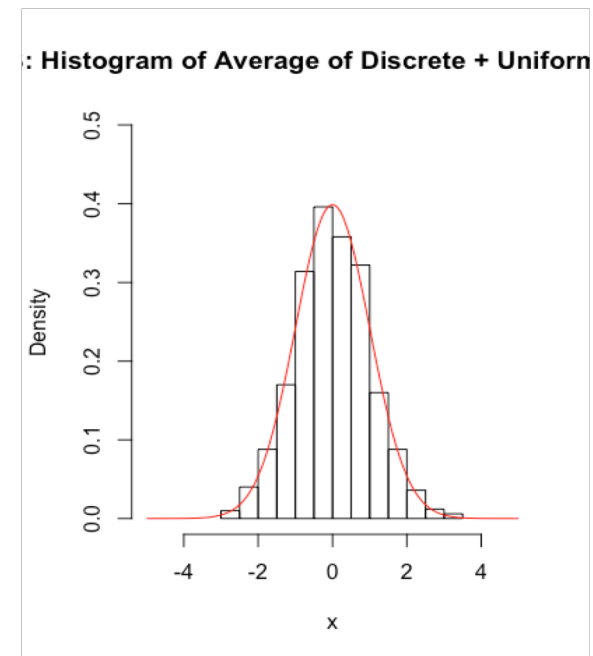
> xstan <- (xmean-mean(xmean))/sd(xmean)

> hist(xstan,prob=TRUE,main="B: Histogram of Average of Discrete + Uniform
RVs",xlab="x",xlim=c(-5,5),ylim=c(0,0.5))

> curve(dnorm(x,mean=0,sd=1),add=TRUE,col="red")
```

The graph

The histogram plot from the simulation:



Monte Carlo Integration

Random numbers have been used since early days for evaluation of integrals. In a certain way, the Buffon's needle problem is such an example. Its simplicity also helps lay the foundation for complex algorithms. Let us begin with the algorithm and some interesting applications.

If we consider the integral

$$I = \int_0^1 g(x) dx.$$

If we include the uniform probability distribution then the integral can be rewritten as

$$I = \int_0^1 f_U(x)g(x) dx.$$

It follows that

$$E(g(U)) = I \quad \text{and} \quad \sum_j^k \frac{g(U_i)}{k} \rightarrow E(g(U)) \quad \text{as } k \rightarrow \infty.$$

Integrating

Suppose that we want to evaluate the integral:

$$I = \int_0^1 e^{e^x} dx$$

```
> u <- runif(1000) #random sample from the U(0,1)
```

```
> int1 <- mean(exp(exp(u)))
```

```
> int1; integrate(function(x) exp(exp(x)),0,1)
```

```
[1] 6.271236
```

```
6.316564 with absolute error < 7e-14
```

More Monte Integration

Let us consider the integral,

$$I = \int_{-2}^2 e^{x+x^2} dx.$$

Clearly, we need to use the substitution $y = (x + 2)/4$ to convert to an integral between 0 and 1.

$$I = \frac{1}{4} \int_0^1 e^{16y^2-12y+2} dy.$$

```
> int3 <- 4* mean(exp(16*u^2-12*u+2))
> int3; integrate(function(x) exp(x+x^2),-2,2)
[1] 90.75416
93.16275 with absolute error < 0.00062
> u <- runif(105) # Increasing the sample points

> int3 <- 4* mean(exp(16*u^2-12*u+2))
> int3
[1] 92.87321
```

Bayesian Inference

Bayesian inference has been greatly enhanced during the previous two to three decades, and especially the use of Monte Carlo techniques has boosted its use and applications. In the previous cases we saw that the target probability density function f need not be completely specified, and it is this aspect which becomes very useful for Bayesian inference.

The *histogram prior* offers a brute-force solution for the present circumstance. For the sake of simplicity, assume that the unknown parameter is a scalar and the probability model and prior are respectively denoted by $p(x|\theta)$ and $\pi(\theta)$. The general algorithm which helps implement the histogram prior approach is then given in the following slide.

The steps to the algorithm

- Consider a grid of legitimate points for θ from the posterior distribution.
- For the data \mathbf{x} compute the likelihood $L(\theta|\mathbf{x})$.
- Obtain the product $L(\theta|\mathbf{x}) \pi(\theta)$ for all the points specified in the first step.
- A random sample $L(\theta|\mathbf{x}) \pi(\theta)$ from the previous step will be a good approximation of the posterior distribution.

Example

Example. A Histogram Prior for the Proportion of Heavy Sleepers. Consider an experiment done to find out if the students sleep for a recommended eight hours per day. We will consider p be the proportion of students who get 8 hours. Suppose it belongs to equally spaced intervals with prior weights given by the prior probability *prior_weights*

`prior_weights = (1, 5.2, 8, 7.2, 4.6, 2.1, 0.7, 0.1, 0, 0)`

is extended over a prior grid as required in this approach. A very elementary function *getc* helps in this task along with the *sapply* function. The likelihood function is computed in the usual way. The *posterior_grid* as given in the **R** code accomplishes the third step of the approach, while the sample function in the final step is obtained.

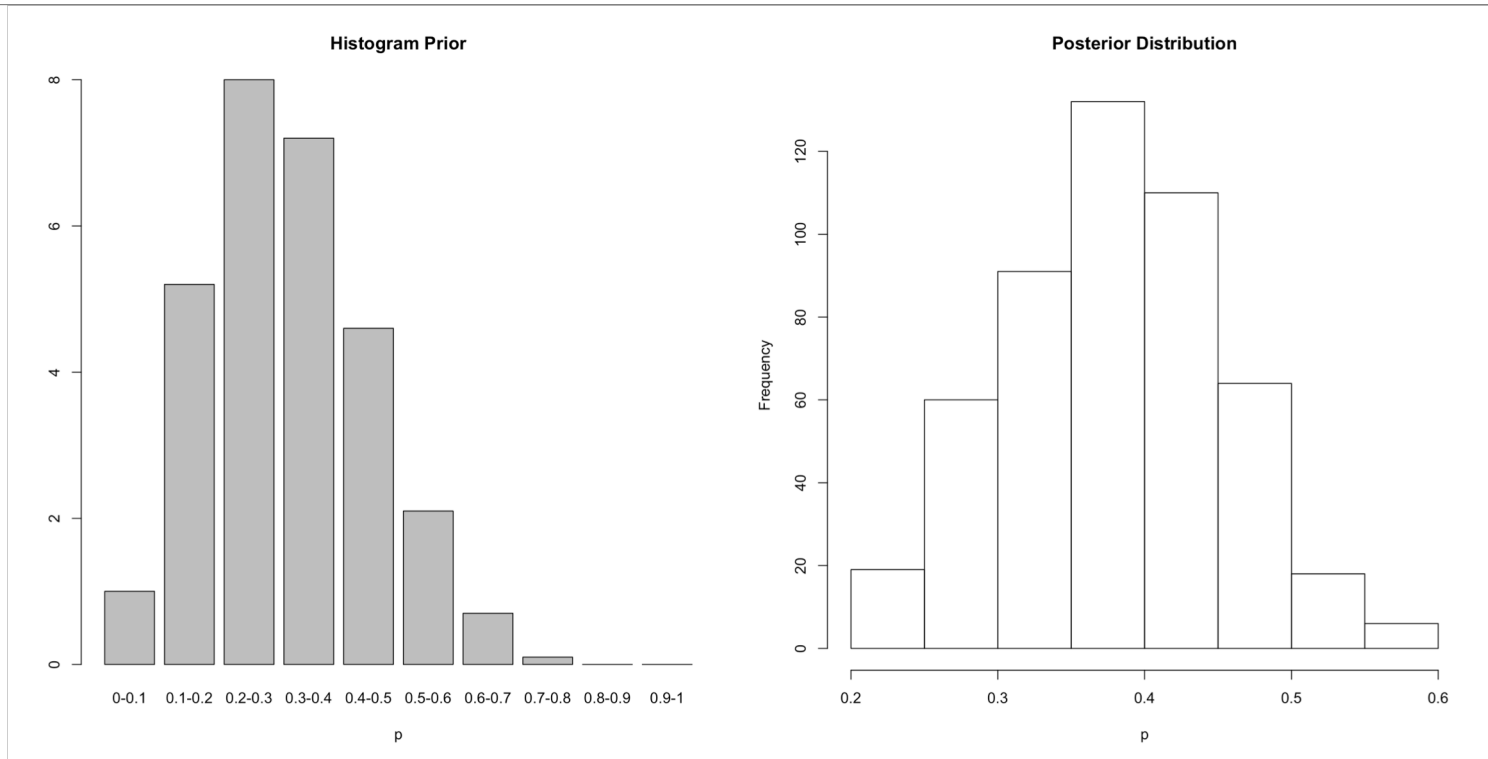
R code

```
> p <- seq(0,1,length=100)
x <- 11; n <- 27
lower_limit <- seq(0,0.9,0.1); upper_limit <- seq(0.1,1,0.1)

prior_intervals <- cbind(lower_limit,upper_limit)
prior_weights <- c(1, 5.2, 8, 7.2, 4.6, 2.1, 0.7, 0.1, 0, 0)

names(prior_weights) <- paste(prior_intervals[,1],"-", prior_intervals[,2],sep="")
prior_interval_weights <- prior_weights/sum(prior_weights)
getc <- function(a,b,c) c[a>= b[,1] & a<=b[,2]]
p_prior <- unlist(sapply(p,getc,b=prior_intervals, c=prior_interval_weights))
likelihood <- p^x*(1-p)^(n-x)
posterior_grid <- p_prior*likelihood
posterior_grid <- posterior_grid/sum(posterior_grid)
posterior_sample <- sample(p,size=500,replace=TRUE,prob=posterior_grid)
par(mfrow=c(1,2))
barplot(prior_weights,main="Histogram Prior",xlab="p")
hist(posterior_sample,main="Posterior Distribution",xlab="p")
```

Pretty Pictures





- Next week we will work on recapping the course and looking at examples in the class from everyone.
- Its been great having you all!
- Good luck in the exam and finishing off the final assignment!
- The Assignment is DUE this Tuesday at 11:55pm!