LAB100

Week 19: Monte Carlo evaluation

Your aim for this workshop is to learn how to use simulation as a way of evaluating properties of random variables or to estimate unknown quantities. Upon completing this workshop you should be able to:

- Perform calculations with random numbers
- Use the `runif` command
- Perform basic Monte-Carlo integration

Remember that you need to change the working directory and open a new R script at the start of each workshop.

1 Functions with a Coin

Recall from the previous workshop that we can simulate 10 tosses of a fair coin using the `sample` command:

```
SampleSpace <- c( "H", "T" )
ProbSet <- c( 0.5, 0.5 )
sample( SampleSpace, size=10, replace=TRUE, ProbSet)
```

Instead of generating heads and tails, we may be interested in the number of times the coin landed on a head out of ten tosses. This event can easily be simulated by a logical statement and the `sum` command:

```
CoinTosses <- sample( SampleSpace, size=10, replace=TRUE, ProbSet)
sum( CoinTosses == "T" )
```

This works because the logical constants `TRUE` and `FALSE` have numerical values of 1 and 0 respectively. Therefore, when a `"T"`ail is simulated the logical statement returns `TRUE` and adds a value of 1 to the summation.

To understand the features of this random number, such as the mean and variance, we need to generate many samples. To do this we can use a for loop to repeat the above code and store the result in a vector. Type the following into your editor, making sure you understand each line and execute:

```
SampleSize <- 10
Trials <- rep( 0, SampleSize )
for( i in 1:SampleSize ){
    CoinTosses <- sample( SampleSpace, size=10, replace=TRUE, ProbSet)
    Trials[i] <- sum( CoinTosses == "T" )
}
```

The vector `Trials` now contains 10 simulated realisations of the number of tails that occurred from the toss 10 fair coins. We can obtain information from this vector by calculating some summary statistics and by plotting the data:

```
mean( Trials )
var( Trials )
barplot( table(Trials), main="Number of Tails with Ten Fair Coins")
```

If you conduct the simulations with a seeding number of 234 your simulations should have a mean of `4.2` and a variance of `1.955556`.

Increase the the value of `SampleSize` to 100 and then to 1000. As the number of samples increase, what would you conclude about the shape of the barplot? In order to reach the same conclusion, would you be prepared to toss a coin 10,000 times?

2 Functions with Two Dice

Recall from the previous workshop that the sample space and probability set for a dice is:

```
SampleSpace <- 1:6
ProbSet <- rep( 1/6, times=6 )
```

The roll of two dice can be simulated as two realisations using the `sample` command:

```
Rolls <- sample( SampleSpace, 2, replace=TRUE, ProbSet )
Rolls
```

Craps is a gambling game involving two dice where the players places wagers on the outcome of the dice's total. To simulate a realisation of this game we simply need to add the observations from the two dice:

```
sum( Rolls )
```

To gain a better understanding of this random variable we need to simulate many more totals of two dice. This can be done in a similar manner as in the previous section:

```
SampleSize <- 1000
Trials <- rep( 0, SampleSize )
for( i in 1:SampleSize ){
    Rolls <- sample( SampleSpace, size=2, replace=TRUE, ProbSet)
    Trials[i] <- sum( Rolls )
}
barplot( table( Trials ), main="Total Number on Two Dice")
```

You should notice a triangular shape to your barplot.

3 The runif command

Most scientific calculators have a RND or RAND button that generate a random number between zero and one. The command in RStudio that produces a similar result is the runif command (pronounced r-u-nif):

```
runif( 1 )
```

The single argument dictates the total number of simulations that you wish to generate. As before, we can calculate statistical summaries and plot a histogram.

```
data <- runif( 100 )
hist( data, col=5 )
mean( data )
```

Note that this histogram is flat, therefore these numbers are random generations from a standard uniform distribution. The simulation range can be adjusted by the inclusion of the min and max arguments. The following example simulates 100 numbers uniformly within the range $[-2, 2]$.

```
runif( 1, min=-2, max=2 )
data <- runif( 100, min=-2, max=2 )
hist( data, col=5 )
mean( data )
```

4 Monte Carlo Integration

One application of Monte Carlo simulation is to approximate integrals. Suppose we wish to evaluate

$$\int_a^b f(x)dx$$

for some non-negative function $f(x)$. Suppose that we also know that $\max f(x) \leq k$, for some value k. We can therefore define a rectangle $[a,b] \times [0,k]$ within which the integrand we wish to evaluate will be enclosed. The idea of Monte Carlo Integration is that we can simulate a large number of points uniformly on the rectangle $[a,b] \times [0,k]$. This rectangle has area $(b-a) \times k$. For each simulated point (x^*, y^*) we can assess whether $f(x^*) > y^*$. The probability that a simulated point satisfies $f(x^*) > y^*$ is equal to

$$p = \frac{\int_a^b f(x)dx}{(b-a) \times k}.$$

If we simulate a large number of points then the observed proportion of times $f(x^*) > y^*$, $\hat{p}$, will be a good estimate of p and hence we can estimate the integrand by

$$\int_a^b f(x)dx \approx (b-a) \times k \times \hat{p}$$

The following function can use Monte Carlo integration to approximate the integral of a supplied function `f` from `a` to `b` assuming the function lies between 0 and `k`, using `N` simulated points.

```
MC.integrate <- function(f,a,b,k,N) {
   simx <- runif(N,a,b)
   simy <- runif(N,0,k)
   p <- sum(f(simx) > simy)/N
   return(p * k * (b-a))
}
```

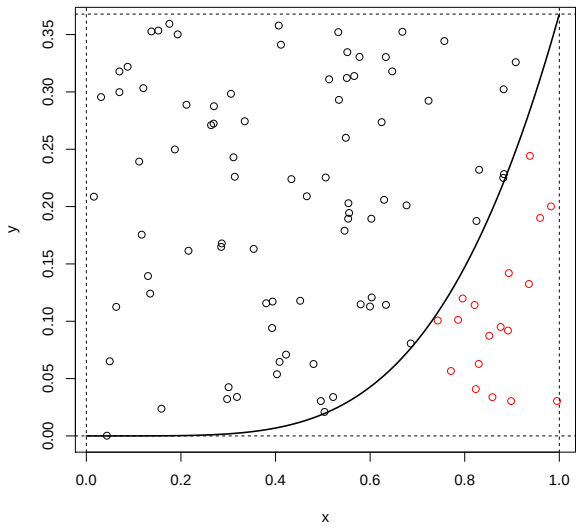
Note that we can simulate points uniformly on a rectangle by independently simulating uniformly distributed x and y co-ordinates.

Suppose we want to estimate $\int_0^1 x^5 \exp(-x)dx$, and we know that $x^5 \exp(-x) \leq \exp(-1)$ on the interval $[0,1]$. The true value of the integrand is $120 - 326\exp(-1) = 0.0713$.

```
f <- function(x) x^5*exp(-x)
set.seed(103)
I <- MC.integrate(f,a=0,b=1,k=exp(-1),N=100)
```

We see that from $N = 100$ and this particular seed we get $I = 0.0662$.

The following plot shows how the estimated has arisen. The estimate is based on the proportion of red points times by the area of the rectangle.



If we repeat the process but with $N = 100000$,

```
set.seed(103)
I <- MC.integrate(f,a=0,b=1,k=exp(-1),N=10000)
```

we get a much better approximation $I = 0.072$.

Quiz 1: Monte Carlo Integration 1

Which of the following commands should produce a value close to 1 if a large enough value of `N` has been specified?

(**A**) `MC.integrate(sin,a=0,b=1,k=1,N=N)`

(**B**) `MC.integrate(sin,a=0,b=pi,k=1,N=N)`

(**C**) `MC.integrate(sin,a=0,b=1,k=pi,N=N)`

(**D**) `MC.integrate(sin,a=0,b=pi/2,k=1,N=N)`

(**E**) `MC.integrate(sin,a=0,b=pi/2,k=1/2,N=N)`

Quiz 2: Variance of the sum of two dice

Adapt the code in Section 2 to obtain a Monte-Carlo estimate of the variance of the sum of the faces from two rolled dice. Which of the following is nearest to the answer you obtain?

(**A**) 7.83 (**B**) 5.83 (**C**) 4.83 (**D**) 6.83 (**E**) 3.83

[Hint: You may want to increase the `SampleSize` to 10000 or 100000 to get a more accurate result.]

Quiz 3: Monte Carlo Integration 2

Using the seeding number 134, use the `MC.integrate` function to approximate the definite integral

$$I = \int_0^1 x^2(1-x)^3 dx$$

with `k` chosen as the maximum value that the integrand attains for $0 \leq x \leq 1$ and `N` taken as 10000. What is the estimated value of I to 5 d.p.?

(**A**) 0.01821 (**B**) 0.01667 (**C**) 0.01677 (**D**) 0.01610 (**E**) 0.01540

Quiz 4: Monte Carlo Integration 3

Investigate the accuracy of using Monte Carlo Integration to approximate the integral

$$I = \int_0^1 \frac{1-x}{1+x} dx$$

by using the `MC.integrate` function with `k=1` and a range of value of `N` from `N=100` to `N=100000`. Let $\hat{I}_N$ denote an estimate of I based on N simulations. Based on your investigations, which of the following formulae best describes the apparent relationship between the average discrepancy between the true and estimated value of I for large N?

(**A**) $\log|\hat{I}_N - I| \approx -2\log N + c$

(**B**) $\log|\hat{I}_N - I| \approx -0.5\log N + c$

(**C**) $\log|\hat{I}_N - I| \approx \log N + c$

(**D**) $\log|\hat{I}_N - I| \approx -\log N + c,$

where c is some constant.