week4hw609

Charley Ferrari

Wednesday, September 16, 2015

Page 191, Question 3

Using Monte Carlo simulation, write an algorthm to calculate an approximation to π by considering the number of random points selected inside the quarter circle:

$$Q : x^2 + y^2 = 1, x \geq 0, y \geq 0$$

This is a unit circle, so I'm going to set my M box to be x=1 and y=1. I'll draw random numbers from $0 \leq x \leq 1$ and $0 \leq y \leq 1$.

```
montecarlox <- runif(100000, 0, 1)
montecarloy <- runif(100000, 0, 1)
montecarlotest <- montecarlox^2 + montecarloy^2 <= 1

proportion <- length(montecarlotest[montecarlotest])/length(montecarlotest)
```

The area of the bounds set by M is 1, and the proportion of the "hits" that resulted from the Monte Carlo test is stated above, along with the area of the quarter circle.

The volume of the whole circle $= \pi r^2$, and from our result above, equals 4 times the area we just calculated:

```
calculatedarea <- 4*proportion
```

r=1 in our case here, so what we're left with is an approximation for π

```
calculatedarea
```

```
## [1] 3.1412
```

Page 194, Question 1

Use the middle-square method to generate

a. 10 random numbers using $x_0 = 1009$

```
a <- 1009^2
```

01018081

```
a <- c(a, 180^2)
r <- 180
```

00032400

```
a <- c(a, 324^2)
r <- c(r, 324)
```

00104976

```
a <- c(a, 1049^2)
r <- c(r, 1049)
```

01100401

```
a <- c(a, 1004^2)
r <- c(r, 1004)
```

01008016

```
a <- c(a, 80^2)
r <- c(r, 80)
```

00006400

```
a <- c(a, 64^2)
r <- c(r, 64)
```

00004096

```
a <- c(a, 40^2)
r <- c(r, 40)
```

00001600

```
a <- c(a, 16^2)
r <- c(r, 16)
```

00000256

```
a <- c(a, 2^2)
r <- c(r, 2)
```

00000004

It appears we've ended up at 0.

Going through this process with this seed, and 8 digits seemed to have highlighted a problem. I assumed that once smaller numbers began appearing, by random chance I'd be back at higher numbers if I continued this algorithm. It would appear that instead there is occassionally a tendency to end up at 0. Here's our full list of random numbers with this seed:

```
r
```

```
## [1]  180  324 1049 1004   80   64   40   16    2
```

Page 201, Question 4

Construct and perform a Monte Carlo simulation of a horse race.

It appears that the odds add up to more than 1. A quick google search revealed this is simply due to the bookie taking a cut:

```
(1/7)+(1/5)+(1/9)+(1/12)+(1/4)+(1/35)+(1/15)+(1/4)
```

```
## [1] 1.13254
```

So, I'll mutiply each of these percentages by 1/1.13254 to get the underlying probabilities of each horse winning:

```
m <- 1/((1/7)+(1/5)+(1/9)+(1/12)+(1/4)+(1/35)+(1/15)+(1/4))

ef <- m/7
ll <- m/5
nl <- m/9
cc <- m/12
pp <- m/4
lh <- m/35
ss <- m/15
dd <- m/4
```

Now I'll compute the cumulative probabilities to make my Monte Carlo simulation easier:

```
ef <- m/7
ll <- m/5 + ef
nl <- m/9 + ll
cc <- m/12 + nl
pp <- m/4 + cc
lh <- m/35 + pp
ss <- m/15 + lh
dd <- m/4 + ss

cumprob <- c(ef,ll,nl,cc,pp,lh,ss,dd)
```

I'll run an iterated simulation, generating random numbers within a for loop. For each random number, I'll implement a vectorized solution to find out where along the Cumulative probability distribution the number belongs, and hence which horse won.

```
rnlist <- runif(10000,0,1)

horselist <- c()

for(i in rnlist){
  horselist <- c(horselist,sum(i < cumprob))
}
```

In this simulation, the numbers correspond to the horse names in reverse order (8=ef, 7=ll, etc). We could plot a histogram of the horselist if we wanted to visualize the results:

```r
hist(horselist, breaks=8)
```