Econ 5023: Statistics for Decision Making

Univariate Statistics (VIII): Intro to Monte Carlo Simulation

Le Wang

November 8, 2017

Itinerary:

1. The Purpose of Monte Carlo Simulation and The Very Brief Account of Its Historical Development
2. Philosophical Debate over Randomness (Frequency vs. Subjective)
3. Randomness in R
   - 3.1 Simulate a Fair Coin in R
   - 3.2 Simulate an Unfair coin in R
   - 3.3 How do we visualize the Law of Large Number?
   - 3.4 How shoud we play craps?

Original purpose:

Find approximate solutions to certain mathematical equations.

Monte Carlo Simulation : generating random data (samples) from a probability distribution.

Creation of Uncertainty/Randomness

Interpretation of Randomness: **Frequency** vs **Subjective**

Frequency Interpretation: You judge a sample by the way it turns out.

Subjective Interpretation: You judge a sample by the way it is produced. We know how it is produced (on our computer) in our computer!

Algorithms to produce randomness have been refined over the years.

We now understand that truly random events do occur on the atomic level. Today, cutting-edge quantum generators produce truly random numbers from the toss of Nature's perfect quantum dice.

Randomness in R

We safely leave this philosophical debate and practical examination of randomness behind us. Assume that it has been done correctly for our purpose.

When we say that a random variable in R *simulates* (or represents) some unknown quantity in real life, we mean that

Any event for this simulated random variable is, from the perspective of our current information and beliefs, just as likely as the same event for the real unknown quantity.

For example,

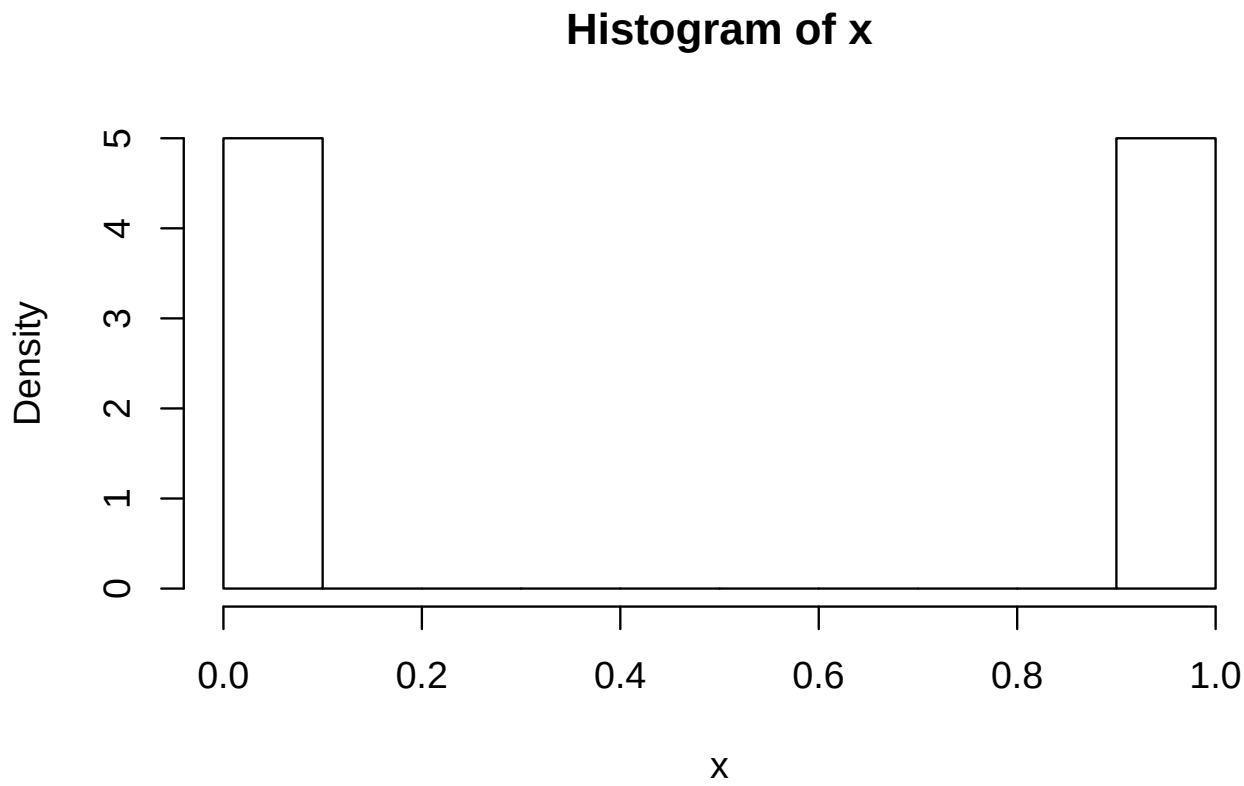
If the unknown outcome of a coin toss has a probability 0.50 of equaling 1 (or tails), then the random variable in R should also have probability .50 of equaling 1 after the next recalculation of the computer.

Example 1: Simulate a (fair) Coin Toss in R

```r
x <- sample(0:1, 100, replace = TRUE)
x
```

```
##   [1] 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 1 0 0 1 0 1 1 0 1 1 0 1 0 0 1 1 0 1 1 1
##  [36] 0 0 1 0 0 0 1 1 0 1 0 0 0 0 1 0 0 1 0 0 1 1 1 1 1 0 1 1 1 1 0 0 0 0 1
##  [71] 1 1 1 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 0 0 0 0 1 0 0 0 0 1 0 0
```

```r
hist(x,probability = TRUE)
```




```r
mean(x)
```

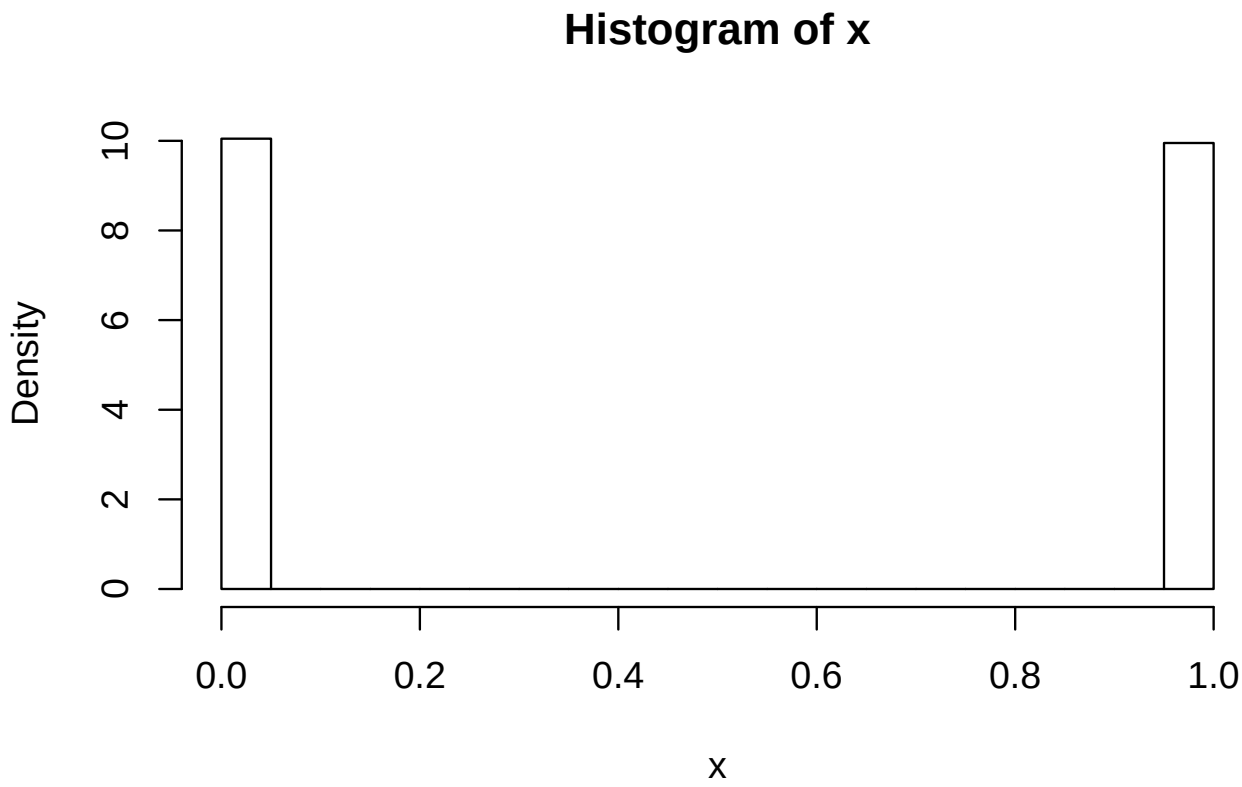
```
## [1] 0.5
```

```r
abs(mean(x)-0.5)
```

```
## [1] 0
```

Two More Refinements: Increase the sample size

```
x <- sample(0:1, 100000, replace = TRUE)
hist(x,probability = TRUE)
```




```
mean(x)
```

```
## [1] 0.4976
```

```
abs(mean(x)-0.5)
```

Two More Refinements: Reproducible Results

```
set.seed(123456)
x <- sample(0:1, 100000, replace = TRUE)
mean(x)
```

```
## [1] 0.49873
```

```
abs(mean(x)-0.5)
```

```
## [1] 0.00127
```

Question: How do you simulate a unfair coin toss?

```r
set.seed(123456)
x <- sample(0:1, 100000, replace = TRUE, prob = c(0.8,0.2))
mean(x)
```

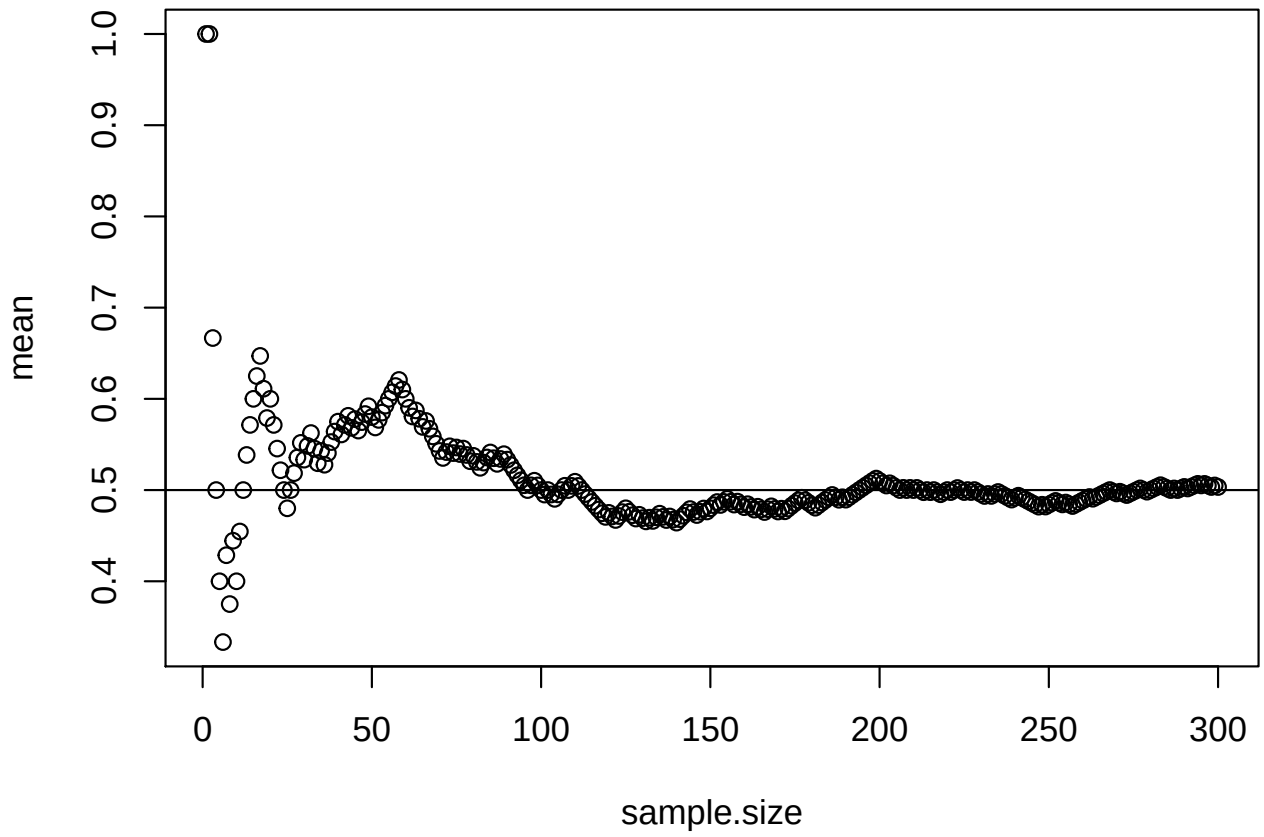
```
## [1] 0.2005
```

What else do we learn from these experiments?

Suppose that these are really random, then we know that

1. When the sample size is large enough, we are getting closer to the truth.
2. Even with the random variables being drawing from a true distribution, the sample estiamtes are arbitrarily close to the truth, but never exact!

Question: How do we visualize the Law of Large Number?


mean
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
50
100
150
200
250
300
sample.size

```r
set.seed(123456)
x<-sample(0:1, 300, replace = TRUE)
sum <- cumsum(x)
sample.size<-c(1:300)
mean <- sum/sample.size
data<-data.frame(cbind(x, sum, sample.size, mean))
head(data)
```

```
##   x sum sample.size      mean
## 1 1   1           1 1.0000000
## 2 1   2           2 1.0000000
## 3 0   2           3 0.6666667
## 4 0   2           4 0.5000000
## 5 0   2           5 0.4000000
## 6 0   2           6 0.3333333
```

Question: How do you play craps?

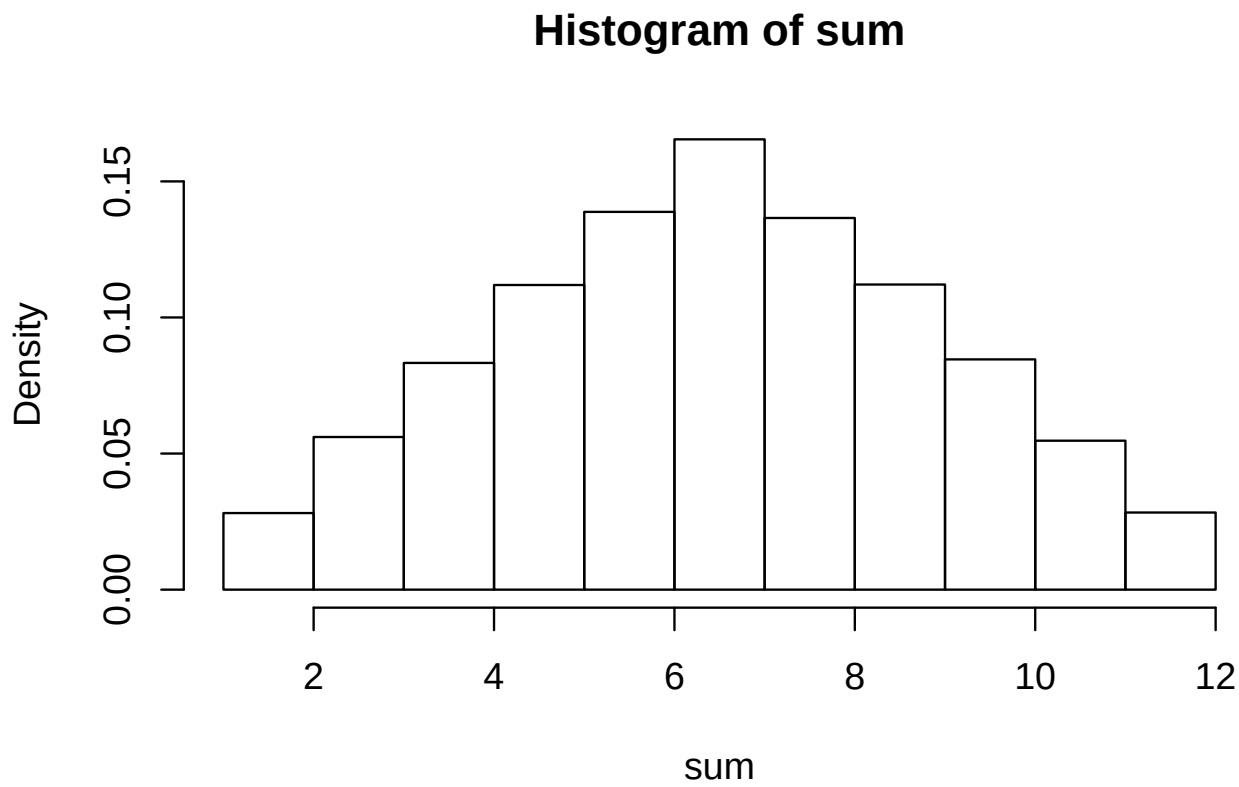
Theoretical Values:

$\Pr[x = 2] = \frac{1}{36}, \Pr[x = 3] = \frac{2}{36}, \ldots, \Pr[x = 7] = \frac{6}{36}, \Pr[x = 8] = \frac{5}{36}$

```r
set.seed(123456)
dice1 <- sample(1:6,100000,replace=T)
dice2 <- sample(1:6,100000,replace=T)

sum <- dice1 + dice2

hist(sum, breaks=c(1:12), probability = TRUE)
```



Applications in Financial Engineering

A fundamental implication of asset pricing theory is that under certain circumstances, the price of a derivative security can be usefully represented as an expected value.

Valuing a derivative security by Monte Carlo typically involves simulating paths of stochastic processes used to desribe the evolution of underlying asset prices, interest rates, model parameters, and other factors relevant to the security in question.

Rather than simply drawing points randomly from an interval, we seek to sample from a space of paths. [Glasserman, Monte Carlo Methods in Financial Engineering]