Making Change at Random

Saumitra Mazumder

July 2022

1 Introduction

We begin first with a general discussion about stopping times.

Definition 1.1. *Let $\tau : \Omega \to \mathbb{N} \cup \{\infty\}$ be a random variable, which is defined on a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_n)_{n\in\mathbb{N}})$. τ is called a stopping time to the filtration $\mathbb{F} = ((\mathcal{F}_n))$ if*

$$\{\tau = n\} \in \mathcal{F}_n \text{ for every } n$$

That is, we are searching for an event to occur, but we don't know when that even will occur. What we do know is that the event can/will occur at some point in the future. Clearly, the occurrence time of this event is random and is a stopping time if at any point in time, we can decided whether the event has occurred or not.

2 The Problem

Fix some price, denoted P. Suppose we are independently sampling uniformly from the a population $X \sim U(0.01, 0.05, 0.10, 0.25, 0.5)$ and put $S_n = \sum_{i=1}^{n} X_n$, where X_n are independent draws of X. Our decision to stop will be when $\{S_n \geq P\}$. By the above definition, for each draw, we can tell whether or not to stop drawing. That is, such a process is a stopping time.

Once we have made the decision to stop, we can create a new random variable $S_{N_0} - P$ which will give us the difference in price and our randomly picked change.

3 Algorithm

We simulate the above process via a Monte Carlo experiment and exploit the law of large numbers to give back a good approximation as follows:

1. Define a domain of possible inputs, $C = \{0.01, 0.05, 0.10, 0.25, 0.5\}$
2. Sample uniformly from inputs, $S_n = \sum_{i=1}^{n} X_n$

3. Make a decision from the input: When $S_n \geq P$, put $CB_n = S_n - P$, otherwise sample again.

4. By the LLN, as the number of samples becomes very large, $\hat{\mu}_{S_n} \to \mathbb{E}[S_n]$

3.1 Code

We create a function that lets us choose the price and the number of realizations.

```
options(digits = 10)                          ## 10 decimal places
set.seed(1)                                   ## to replicate

# Notice that each draw is independent and with equal probability
# (ie, proability does not change with each draw)
# so we can simply use the sample function on a set,
# say C = {1, 5, 10, 25, 50}

changeBack <- function(P,N){
  n <- as.numeric(N);                         ## number of realizations
  p <- as.numeric(P)
  C <- c(0.01, 0.05, 0.10, 0.25, 0.5);        ## init coin vector
  CB <-matrix(0, n, 1)                        ## Init change back vector
  DS <- matrix(0, 3, 1)                       ## Init Descriptive stats

    for(i in 1:n){
      S <- 0;
      while (p > S){S = S + sample(C,1,replace = TRUE)}
      CB[i] = S- p
    }
    DS[1] <-mean(CB)                          ## sample mean
    DS[2] <-sd(CB)                            ## sample std deviation
    DS[3] <- sd(CB)/sqrt(n)                   ## std error of the mean is
                                              ## estimated via the std error of std. dev
    return(DS)
}
```

The descriptive statistics for the above code are as follows:

```
> changeBack(0.25,1000)
               [,1]
[1,] 0.162950000000
[2,] 0.144879907812
[3,] 0.004581504959
> changeBack(1,1000)
               [,1]
[1,] 0.154440000000
[2,] 0.141741232452
```

```
[3,] 0.004482251329
> changeBack(10,1000)
              [,1]
[1,] 0.18585000000
[2,] 0.14603014176
[3,] 0.00461787855
```

Here, the first row are the mean values of the change back given 1000 runs of the above algorithm for $P = 0.25, 1, 10$. Likewise, the second row and third row are the sample standard deviation and the estimate for the standard error of the mean respectively.

4 Pretty Graph

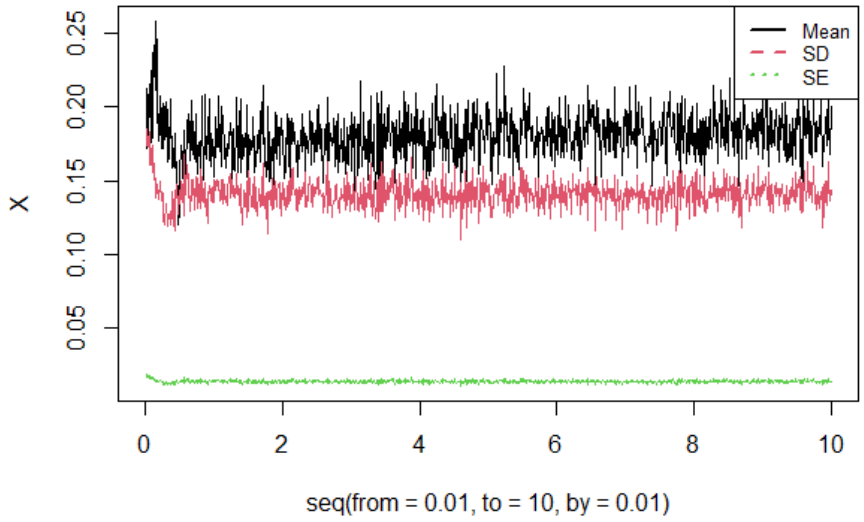
For obtaining the descriptive statistics for $0.01 \leq P \leq 10$, we can run a loop over the above function as follows:

```
X <- matrix(0,3,1000);
Ps <- seq(from = 0.01, to = 10, by = 0.01);
for ( i in 1:length(Ps)) {
     X[,i] <- changeBack(Ps[i],100);             ## Give my poor Raspberry Pi a break.
}
```

Calling the following:

```
> X <-as.data.frame(t(X));
> colnames(X) <- c("Mean", "SD", "SE");
> matplot(x = seq(from = 0.01, to = 10, by = 0.01), y= X, type="l", lty=1)
> legend("topright", colnames(X), col=seq_len(3), cex=0.8, lty=seq_len(3), lwd=2)
```

Outputs the following plot:



What we can tell from the above plots are that for price values between 0.01 and 10.00, the change back descriptive statistics tend to cluster around the same values. There tends not to be any trending other than for very small prices.