Thomson Nguyen (tn248)
Scientific Programming (SP)
November 9, 2010

Scientific Programming
Assignment 2

# 1 Roulette (20)

**Answer.** We have a single-zero roulette wheel: 18 red numbers, 18 black numbers, one green zero. We have four betting systems (or games from here on) to model. To model these four games, we should probably model a roulette spin first. For the sake of simplicity, let's assume the only possible bets are coloured bets (red or black), or single number bets (1-36). This means no splits, streets, corners, sixes, trios, baskets, or other bets at the roulette table.

```
> roulettespin <- function(number, bet){
+         # The spin!
+         winner <- sample(0:36,1)
+
+         # In roulette, odd numbers are red and even are black in [1,10] and [19,28].
+         # In [11,18] and [29,36], odd numbers are black, and even are red.
+         # How confusing!
+
+         red <- c(1,3,5,7,9,12,14,16,18,19,21,23,25,27,30,32,34,36)
+         black <- c(2,4,6,8,10,11,13,15,17,20,22,24,26,28,29,31,33,35)
+         green <- 0
+
+         spinwin <- 0
+
+         if(winner %in% red)
+         winner[2] <- -1
+         else if(winner %in% black)
+         winner[2] <- -2
+         else if(winner %in% green)
+         winner[2] <- 0
+
+         if(number <= 0 & winner[2] == number){
+                         spinwin <- bet} # 1:1 odds
+         else if(number == winner[1]){
+                 spinwin <- bet * 35} # 35:1 odds
+         else{
+                 spinwin <- -bet}
+
+         return(spinwin)
+         }
```

Note that we've designated `number` to $-1$ for a red bet, and $-2$ for a black bet.

Now that we have our roulette spin, we can now model our four games. Note that all four of the proceeding functions take no inputs, and will output the number of bets and the amount won in a single game.

Our first game consists of a single bet of \$1, always on red. Let's call it `alwaysbetonred`:

```
> alwaysbetonred <- function(){
+         redspins <- 0
+         redwins <- 0
+
+         redwins <- roulettespin(-1,1)
+         redspins <- redspins + 1
+         return(c(redspins,redwins))
+         }
```

We can also take one bet of $1 on one number (`betonenumber`), in hopes that we get the higher 35:1 payout, but with the harder odds:

```
> betonenumber <- function(){
+         onenumberspins <- 0
+         onenumberwins <- 0
+
+         # 17 is my favorite number
+         onenumberwins <- roulettespin(17,1)
+         onenumberspins <- onenumberspins +1
+         return(c(onenumberspins,onenumberwins))
+         }
```

Our next game is the martingale. The martingale in roulette refers to a class of 18th century betting strategies made popular in France. Paul Pierre Lévy and Joseph Leo Doob introduced the martingale as a stochastic process in probability theory much later on in the early 20th century. It is said that their motivation for developing martingales was to prove the stupidity of betting systems claiming to have "beaten the house". In our implementation (`martingale`), we start with a $1 bet on red, doubling our bet after each successive loss. We reset our bet to $1 after a win, and we end the game after winning $10, or when our bet exceeds $100.

```
> martingale <- function(){
+         martingalespins <- 0
+         martingalewins <- 0
+
+         currentbet <- 1
+         while(martingalewins < 10 & currentbet < 100){
+                 spin <- roulettespin(-1,currentbet) # the actual spin
+                 martingalespins <- martingalespins + 1
+                 martingalewins <- martingalewins + spin
+
+                 if(spin < 0){
+                         currentbet <- currentbet * 2}
+
+                 else if(spin > 0){
+                         currentbet <- 1}
+
+                 #print(c(martingalespins,martingalewins,currentbet))
+                 }
+
+         return(c(martingalespins,martingalewins))
+         }
```

Lastly, we'll look at the Labouchere (split martingale) system. No one really knows who came up with this variant of the martingale, although it is said that British politician (and Cantab) Henry Labouchere

had devised the system from his fondness for gambling. In our implementation (`labouchere`), we start with the ordered list $(1,2,3,4)$. We will bet the sum of the first and last numbers of this list on red, meaning our first bet is \$5. On a win, the first and last numbers are deleted; on a loss, the sum is added to the end of the list. This process continues until we have an empty list, or until the bet exceeds \$100.

```
> labouchere <- function(){
+         laboucherespins <- 0
+         laboucherewins <- 0
+         currentbet <- 0 # to be overwritten quite soon
+         list <- c(1,2,3,4)
+
+         while(length(list) != 0 & currentbet < 100){
+                 currentbet <- list[1] + list[length(list)] #initialize
+                 spin <- roulettespin(-1,currentbet)
+                 laboucherespins <- laboucherespins + 1
+                 laboucherewins <- laboucherewins + spin
+
+                 if(spin < 0){
+                         list <- c(list, currentbet)}
+
+                 else if (spin > 0){
+                         list <- list[c(-1,-length(list))]}
+
+                 #print(list)
+                 #print(c(laboucherespins,laboucherewins,currentbet))
+                 }
+         return(c(laboucherespins,laboucherewins))
+         }
```

We'll now create a function, `simulateall`, that will run these four games a specified number of times (specified by its sole argument), and outputs a table of the expected winnings, the win percentage, and the expected number of bets (play time), and their standard deviations:

```
> simulateall <- function(numberoftimes){
+         Atrials <- t(replicate(numberoftimes,alwaysbetonred()))
+         Btrials <- t(replicate(numberoftimes,betonenumber()))
+         Ctrials <- t(replicate(numberoftimes,martingale()))
+         Dtrials <- t(replicate(numberoftimes,labouchere()))
+
+         Awins <- which(Atrials[,2] > 0)
+         Bwins <- which(Btrials[,2] > 0)
+         Cwins <- which(Ctrials[,2] > 0)
+         Dwins <- which(Dtrials[,2] > 0)
+
+         totalspins <- c(mean(Atrials[,1]),mean(Btrials[,1]),mean(Ctrials[,1]),mean(Dtrials[,1]))
+         spinssd <- c(sd(Atrials[,1]),sd(Btrials[,1]),sd(Ctrials[,1]),sd(Dtrials[,1]))
+
+         totalwins <- c(length(Awins),length(Bwins),length(Cwins),length(Dwins))/numberoftimes
+         winssd <- sqrt(totalwins * (1-totalwins))
+
+         expectedwin <- c(mean(Atrials[,2]),mean(Btrials[,2]),mean(Ctrials[,2]),mean(Dtrials[,2]))
+         winningssd <- c(sd(Atrials[,2]),sd(Btrials[,2]),sd(Ctrials[,2]),sd(Dtrials[,2]))
+
```

```
+
+         results <- cbind(expectedwin,winningssd,totalwins,winssd, totalspins,spinssd)
+         rownames(results) <- c("Bet on red","Bet one number","Martingale","Labouchere")
+         colnames(results) <- c("Expected winnings", "(SD)","Win percentage","(SD)",
+         "Expected bets","(SD)")
+         return(results)
+         }
```

For 22.2.1, let's run this for $100,000$ repetitions of each game. We'll start with setting the seed to my birthday in American date form. We'll also call `cacheSweave`[1] because the next couple of steps are computationally expensive. Table 1 has the results of our single simulation:

```
> seed <- 110685
> set.seed(seed)
> library(cacheSweave)

> results <- simulateall(100000)

> results
```

| | Expected winnings | (SD) | Win percentage | (SD) | Expected bets | (SD) |
|---|---|---|---|---|---|---|
| Bet on red | -0.03154 | 0.9995075 | 0.48423 | 0.4997512 | 1.00000 | 0.000000 |
| Bet one number | -0.02440 | 5.8455236 | 0.02710 | 0.1623748 | 1.00000 | 0.000000 |
| Martingale | -1.93885 | 37.9591254 | 0.90994 | 0.2862677 | 19.51669 | 4.520879 |
| Labouchere | -3.82102 | 77.4593735 | 0.94613 | 0.2257610 | 8.72328 | 7.524335 |

As a check (and for 22.2.2), let's calculate the expected value and proportion of games won for games A and B (i.e., betting on one colour and betting on one number). For game A, our expected value is

$$\frac{18}{37} \times 1 + \frac{19}{37} \times -1 = -0.02703,$$

with a win ratio of $\frac{18}{37} = 0.4865$. Compare with our values in `results`, $-0.03154$ and $0.4842$. Our percentage error is

$$\left| \frac{-0.02703 - (-0.03154)}{0.2703} \right| \times 100 = 1.669\%$$

for the expected value and

$$\left| \frac{0.4865 - 0.4842}{0.4865} \right| \times 100 = 0.472\%$$

for our win ratio.

For game B, our expected value is

$$\frac{1}{37} \times 35 + \frac{36}{37} \times -1 = -0.02703,$$

with a win ratio of $\frac{1}{37} = 0.02703$. Compare this with our values of $-0.02440$ and $0.02710$, respectively. This gives us an error percentage of

$$\left| \frac{-0.02703 - (-0.02440)}{0.2703} \right| \times 100 = 0.973\%$$

for the expected value and

$$\left| \frac{0.02703 - 0.02710}{0.02703} \right| \times 100 = 0.259\%$$

[1] I've attached the cache folders with this .Rnw and .pdf, as my `simulateall` function is a bit computationally expensive. I used `system.time` to time the function, and for 100k it took about 200 seconds.

for our win ratio. Not bad!

We see that for game A (betting one colour), the maximum amount we can win or lose is \$1, as our game only lasts for one bet. This is exactly the same for game B (betting one number). For the martingale, the maximum amount we can win is \$10 (our goal), but our maximum loss is if we were to lose seven times in a row, or $\sum_{i=0}^{7} 2^i = \$127$. Due to nature of continuing to spin until a list has been exhausted, for Labouchere we get that the maximum amount we can win is \$58, and the maximum loss in a single game is \$899.

We probably want to find out the variation between different runs, so now we will repeat this 100k simulation five times (22.2.3):

```
> run1 <- simulateall(100000)
> run2 <- simulateall(100000)
> run3 <- simulateall(100000)
> run4 <- simulateall(100000)
> run5 <- simulateall(100000)
```

Let's now take the min and max values of each column across the five simulations. These values are assembled in Table 2:

```
> expvals=matrix(rep(0,8),4,2)
> propwins=matrix(rep(0,8),4,2)
> expbets=matrix(rep(0,8),4,2)
> for (i in 1:4){
+ expvals[i,] <- summary(c(run1[i,1],run2[i,1],run3[i,1],run4[i,1],run5[i,1]))[c(1,6)]
+ propwins[i,] <- summary(c(run1[i,3],run2[i,3],run3[i,3],run4[i,3],run5[i,3]))[c(1,6)]
+ expbets[i,] <- summary(c(run1[i,5],run2[i,5],run3[i,5],run4[i,5],run5[i,5]))[c(1,6)]}
> tablevar<- cbind(expvals,propwins,expbets)
> rownames(tablevar) <- c("Bet on red","Bet one number","Martingale","Labouchere")
```

Here, Table 1 seems to suggest that the Labouchere system has the most variable expected winnings. We also see that the Labouchere system also has the most variable expected playing time.

□

| | **Expected Winnings** | | **Proportion Wins** | | **Expected Bets** | |
|---|---|---|---|---|---|---|
| Game | Mean | SD | Mean | SD | Mean | SD |
| Bet on red | -0.03 | 1.00 | 0.48 | 0.50 | 1.00 | 0.00 |
| Bet one number | -0.02 | 5.85 | 0.03 | 0.16 | 1.00 | 0.00 |
| Martingale | -1.94 | 37.96 | 0.91 | 0.29 | 19.52 | 4.52 |
| Labouchere | -3.82 | 77.46 | 0.95 | 0.23 | 8.72 | 7.52 |

Table 1: One 100k simulation of four different betting systems in roulette.

| | **Expected Winnings** | | **Proportion Wins** | | **Expected Bets** | |
|---|---|---|---|---|---|---|
| Game | Min | Max | Min | Max | Min | Max |
| Bet on red | -0.03 | -0.03 | 0.48 | 0.49 | 1.00 | 1.00 |
| Bet one number | -0.04 | -0.02 | 0.03 | 0.03 | 1.00 | 1.00 |
| Martingale | -2.06 | -1.78 | 0.91 | 0.91 | 19.50 | 19.54 |
| Labouchere | -4.11 | -3.80 | 0.95 | 0.95 | 8.68 | 8.72 |

Table 2: Variation of five different 100k simulations of the four betting systems.