Thomson Nguyen (tn248)
Scientific Programming (SP)
November 8, 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)
+         }
```

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` because the next couple of steps are computationally expensive:

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

> simulateall(10000)

               Expected winnings       (SD) Win\n\t percentage      (SD)
Bet on red               -0.0166  0.9999122             0.4917 0.4999311
Bet one number            0.0836  6.1513572             0.0301 0.1708625
Martingale               -2.1547 38.2890790             0.9084 0.2884605
Labouchere               -4.3623 78.9708635             0.9446 0.2287593
               Expected bets     (SD)
Bet on red            1.0000 0.000000
Bet one number        1.0000 0.000000
Martingale           19.5269 4.562933
Labouchere            8.8130 7.559031
```

□