STAT 206 Lab 9_Lihua Xu

Due Monday, December 4, 5:00 PM

General instructions for labs: You are encouraged to work in pairs to complete the lab. Labs must be completed as an R Markdown file. Be sure to include your lab partner (if you have one) and your own name in the file. Give the commands to answer each question in its own code block, which will also produce plots that will be automatically embedded in the output file. Each answer must be supported by written statements as well as any code used.

Agenda: Simulate a Markov chain, adjust the behavior to ensure a fair game

Markov Chains

Suppose you have a game where the probability of winning on your first hand is 48%; each time you win, that probability goes up by one percentage point for the next game (to a maximum of 100%, where it must stay), and each time you lose, it goes back down to 48%. Assume you cannot go bust and that the size of your wager is a constant $100.

1. Is this a fair game? Simulate one hundred thousand sequential hands to determine the size of your return. Then repeat this simulation 99 more times to get a range of values to calculate the expectation.

```
Markov_C <- function(N,initial_winning,increment){
m <- 0
size <-100
first_winning <- initial_winning
for (i in 1:N){
  A <- runif(1)
  if (A <= initial_winning){
    initial_winning <- initial_winning+increment
    m <- m+1
    size <- size+100}
  else{initial_winning <- first_winning
  size <- size-100
  }
}
results <- list(winning_times=m,total_size=size)
return(results)
}

#Running for 99 times
times <- c()
size_total <- c()
for (h in 1:99)
{ times <- c(times,Markov_C(100000,0.48,0.01)$winning_times)
  size_total <- c(size_total, Markov_C(100000,0.48,0.01)$total_size)
  }
ninenine_times <- rbind(times,size_total)
ninenine_times
```

```
##              [,1]    [,2]    [,3]    [,4]    [,5]    [,6]    [,7]    [,8]
## times        48797   48823   49105   49225   48921   48916   48605   48749
## size_total -191300 -247700 -184300 -182500 -191100 -192100 -234300 -203100
```

```
##                [,9]   [,10]   [,11]   [,12]   [,13]   [,14]   [,15]   [,16]
## times         49171   49160   48911   49084   48999   49061   48769   48975
## size_total -177700 -239500 -135900 -200300 -233300 -203900 -241100 -175700
##               [,17]   [,18]   [,19]   [,20]   [,21]   [,22]   [,23]   [,24]
## times         49051   48864   49012   49053   48992   49149   48833   48693
## size_total -203500 -210900 -217500 -158900 -198300 -195500 -247100 -148100
##               [,25]   [,26]   [,27]   [,28]   [,29]   [,30]   [,31]   [,32]
## times         49051   49001   48923   49003   48891   49066   48985   48918
## size_total -218500 -149100 -167500 -199100 -201300 -198100 -200300 -145500
##               [,33]   [,34]   [,35]   [,36]   [,37]   [,38]   [,39]   [,40]
## times         48899   48790   49104   48981   48989   48998   48997   48843
## size_total -139300 -220900 -244900 -130500 -215100 -183700 -204700 -181900
##               [,41]   [,42]   [,43]   [,44]   [,45]   [,46]   [,47]   [,48]
## times         48956   48998   49038   49240   48861   48877   48783   48864
## size_total -201100 -221500 -226900 -148500 -278900 -174100 -244900 -187500
##               [,49]   [,50]   [,51]   [,52]   [,53]   [,54]   [,55]   [,56]
## times         48901   49278   48860   49178   49365   49033   48985   48994
## size_total -225300 -210500 -215100 -199100 -193300 -152700 -226100 -177900
##               [,57]   [,58]   [,59]   [,60]   [,61]   [,62]   [,63]   [,64]
## times         48762   49149   48940   49037   49097   48718   49052   48957
## size_total -202500 -183300 -217300 -161700 -268900 -218100 -164700 -230100
##               [,65]   [,66]   [,67]   [,68]   [,69]   [,70]   [,71]   [,72]
## times         48896   48757   49065   49378   48947   49355   48928   48848
## size_total -177100 -180100 -208700 -197300 -159700 -177700 -221100 -164300
##               [,73]   [,74]   [,75]   [,76]   [,77]   [,78]   [,79]   [,80]
## times         48949   49223   49051   48954   48893   49170   48862   49022
## size_total -165100 -195300 -209700 -148500 -215300 -168900 -158500 -248300
##               [,81]   [,82]   [,83]   [,84]   [,85]   [,86]   [,87]   [,88]
## times         49208   49071   49257   49246   49116   49177   49329   48748
## size_total -128700 -189700 -213900 -207100 -215500 -195700 -193700 -242700
##               [,89]   [,90]   [,91]   [,92]   [,93]   [,94]   [,95]   [,96]
## times         49148   48896   49019   49069   48937   48985   49234   49040
## size_total -236300 -200100 -192300 -203300 -202700 -225500 -173900 -236500
##               [,97]   [,98]   [,99]
## times         49054   49132   49047
## size_total -164100 -204300 -214700
```

```
#Expectations
mean_winning_time <- mean(ninenine_times[1,])
mean_size_total <- mean(ninenine_times[2,])
mean_winning_time
```

```
## [1] 49002.94
```

```
mean_size_total
```

```
## [1] -197780.8
```

2. Repeat this process but change the starting probability to a new value within 2% either way. Get the expected return after 100 repetitions. Keep exploring until you have a return value that is as fair as you can make it. Can you do this automatically?

```
#starting probability 49%
times <- c()
size_total <- c()
for (h in 1:100)
```

```
{ times <- c(times,Markov_C(100000,0.49,0.01)$winning_times)
  size_total <- c(size_total, Markov_C(100000,0.49,0.01)$total_size)
  }
ninenine_times <- rbind(times,size_total)
mean_winning_time <- mean(ninenine_times[1,])
mean_size_total <- mean(ninenine_times[2,])
mean_winning_time
```

```
## [1] 50073.6
```

```
mean_size_total
```

```
## [1] 15284
```

```
#starting probability 50%
times <- c()
size_total <- c()
for (h in 1:100)
{ times <- c(times,Markov_C(100000,0.5,0.01)$winning_times)
  size_total <- c(size_total, Markov_C(100000,0.5,0.01)$total_size)
  }
ninenine_times <- rbind(times,size_total)
mean_winning_time <- mean(ninenine_times[1,])
mean_size_total <- mean(ninenine_times[2,])
mean_winning_time
```

```
## [1] 51109.05
```

```
mean_size_total
```

```
## [1] 304388
```

```
#starting probability 47%
times <- c()
size_total <- c()
for (h in 1:100)
{ times <- c(times,Markov_C(100000,0.47,0.01)$winning_times)
  size_total <- c(size_total, Markov_C(100000,0.47,0.01)$total_size)
  }
ninenine_times <- rbind(times,size_total)
mean_winning_time <- mean(ninenine_times[1,])
mean_size_total <- mean(ninenine_times[2,])
mean_winning_time
```

```
## [1] 47961.41
```

```
mean_size_total
```

```
## [1] -405332
```

```
#starting probability 46%
times <- c()
size_total <- c()
for (h in 1:100)
{ times <- c(times,Markov_C(100000,0.46,0.01)$winning_times)
  size_total <- c(size_total, Markov_C(100000,0.46,0.01)$total_size)
  }
ninenine_times <- rbind(times,size_total)
```

```
mean_winning_time <- mean(ninenine_times[1,])
mean_size_total <- mean(ninenine_times[2,])
mean_winning_time
```

```
## [1] 46920.25
```

```
mean_size_total
```

```
## [1] -617808
```

```
#When the starting probability is 49% or 50%, the results would be more resonable.
#And I can do it automatically.
```

3. Repeat again, keeping the initial probability at 48%, but this time change the probability increment to a value different from 1%. Get the expected return after 100 repetitions. Keep changing this value until you have a return value that is as fair as you can make it.

```
#probability increment 1.1%
times <- c()
size_total <- c()
for (h in 1:100)
{ times <- c(times,Markov_C(100000,0.48,0.011)$winning_times)
  size_total <- c(size_total, Markov_C(100000,0.48,0.011)$total_size)
  }
ninenine_times <- rbind(times,size_total)
mean_winning_time <- mean(ninenine_times[1,])
mean_size_total <- mean(ninenine_times[2,])
mean_winning_time
```

```
## [1] 49321.92
```

```
mean_size_total
```

```
## [1] -113224
```

```
#probability increment 1.2%
times <- c()
size_total <- c()
for (h in 1:100)
{ times <- c(times,Markov_C(100000,0.48,0.012)$winning_times)
  size_total <- c(size_total, Markov_C(100000,0.48,0.012)$total_size)
  }
ninenine_times <- rbind(times,size_total)
mean_winning_time <- mean(ninenine_times[1,])
mean_size_total <- mean(ninenine_times[2,])
mean_winning_time
```

```
## [1] 50383.98
```

```
mean_size_total
```

```
## [1] -57246
```

```
#probability increment 1.3%
times <- c()
size_total <- c()
for (h in 1:100)
{ times <- c(times,Markov_C(100000,0.48,0.013)$winning_times)
  size_total <- c(size_total, Markov_C(100000,0.48,0.013)$total_size)
```

```
  }
ninenine_times <- rbind(times,size_total)
mean_winning_time <- mean(ninenine_times[1,])
mean_size_total <- mean(ninenine_times[2,])
mean_winning_time
```

```
## [1] 50670.09
```

```
mean_size_total
```

```
## [1] 65786
```

```
#probability increment 1.5
times <- c()
size_total <- c()
for (h in 1:100)
{ times <- c(times,Markov_C(100000,0.48,0.015)$winning_times)
  size_total <- c(size_total, Markov_C(100000,0.48,0.015)$total_size)
  }
ninenine_times <- rbind(times,size_total)
mean_winning_time <- mean(ninenine_times[1,])
mean_size_total <- mean(ninenine_times[2,])
mean_winning_time
```

```
## [1] 61436.46
```

```
mean_size_total
```

```
## [1] 2365862
```

```
#Based on the results I give above, When the probability increment is
#larger than or equal to 1.2% the results are more reasonable.
```