Lista05 _MAE399

Samira Tokunaga, Yiying Lin, André Vinícius

1. Considere S_n como sendo o passeio aleatório simples simétrico ($p = \frac{1}{2}$) em $\mathbb{Z}$.

a. Dizemos que há uma troca de liderança quando, para algum k, observamos $S_k = 0$, mas $S_{k-1} \neq S_{k+1}$. Sabemos que probabilidade de que, até o instante $2n + 1$, ocorram exatamente r trocas de liderança é igual a:

$$\binom{2n+1}{n-r}/2^{2n}$$

Considere agora 1.000 simulações para as trajetórias de S_n para $n \in \{0, 1, ..., 100\}$. Encontre para cada uma destas 1.000 simulações o número de trocas de lideranças. Com estes dados estime a distribuição do número de trocas de lideranças em trajetorias de tamanho 101. Compare em uma tabela estas estimativas com os valores obtidos pela fórmula descrita no início deste item.

```
passo <- function(n){

  #função que gera os dados de um passeio aleatório de tamanho (n)

  #sorteia os passos
  steps <- sample(c(-1, 1), n, replace = TRUE)
  #inicializa o vetor(passeio aleatório) com zero
  walk <- numeric(1)
  for (s in 1:length(steps)){
    #soma os passos
    walk <- c(walk, walk[s] + steps[s])
  }
  #retorna os dados
  return(walk)
}

run <- function(nsimul, n){

  #função que gera uma matriz com simulações de passeios aleatórios
  #cada linha da matriz é uma simulação de passeio aleatório
  #cada elemento de uma linha é um passo do passeio aleatório

  #inicializa a matriz vazia
  m <- matrix(nrow = 0, ncol = n + 1)
  #barra de progresso
  #pb <- txtProgressBar(min = 0, max = nsimul, style = 3)
  for(i in 1:nsimul){
    #preenche a matriz com os passeios
    m <- rbind(m, passo(n))
    #atualiza a barra de progresso
    #setTxtProgressBar(pb, i)
  }
  #fecha a barra de progresso
  #close(pb)
  #retorna os dados
  return(m)
```

```r
}

trocaslidera <- function(matriz){

  #função que observa as trocas de liderança de uma matriz de passeios aleatórios
  #o vetor 'ntrocas' conta a quantidade de trocas
  #o vetor 'ultima' registra a última troca observada

  #inicializando os vetores vazios
  ntrocas <- c()
  ultima <- c()

  #iterando sobre as linhas da matriz
  for (j in 1:nrow(matriz)){
    #inicializa os dados com zero
    cont = 0
    ultima[j] = 0
    #iterando sobre os elementos da linha(passeio aleatório)
    for (i in 2:(ncol(matriz)-1)){
      #se encontrar um zero
      if(matriz[j,i] == 0){
        #verifica se o eixo foi cruzado
        if(matriz[j, i+1] != matriz[j, i-1]){
            #se sim, adiciona ao contador e registra a última troca
            cont = cont + 1
            ultima[j] <- i - 1
          }
        }
    }
      #ao final da linha, registra a quantidade de trocas
      ntrocas[j] <- cont
  }
  #monta a matriz com os dados
  dados <- rbind(ntrocas, ultima)
  #retorna os dados
  return(dados)
}

#gerando dados para observação
passeios <- run(1000, 100)
trocas <- trocaslidera(passeios)

#calculando a proporção de trocas observadas
trocasobserv <- table(trocas[1,])/1000
#função teórica da distribuição de trocas
teoriatrocas <- function(r, n) (choose(2*n+1, n-r))/2^(2*n)
#gerando um vetor com os dados teóricos para comparação
trocasteoric <- numeric(0)
for(i in 0:(length(trocasobserv)-1)){
  trocasteoric <- c(trocasteoric, teoriatrocas(i, 50))
}
names(trocasteoric) <- c(0:(length(trocasobserv)-1))
```

```r
#tabela de comparação do observado com o teórico
trocastabela <- rbind(trocasobserv, trocasteoric)
trocastabela
```

```
##                      0         1         2         3         4          5
## trocasobserv 0.1560000 0.1460000 0.1400000 0.1300000 0.0990000 0.08900000
## trocasteoric 0.1576179 0.1515557 0.1401175 0.1245489 0.1064327 0.08742686
##                      6          7          8          9        10
## trocasobserv 0.0710000 0.06000000 0.05100000 0.02400000 0.0140000
## trocasteoric 0.0690212 0.05236091 0.03816134 0.02671294 0.0179546
##                      11          12          13          14          15
## trocasobserv 0.00700000 0.006000000 0.002000000 0.001000000 0.002000000
## trocasteoric 0.01158361 0.007170808 0.004257667 0.002423595 0.001321961
##                        16           17
## trocasobserv 0.0010000000 0.0010000000
## trocasteoric 0.0006905766 0.0003452883
```

b. Sabemos também que a probabilidade de que até o instante $2n$(inclusive) a última visita à origem tenha ocorrido no instante $2k$ é igual a:

$$\binom{2k}{k}\binom{2n-2k}{n-k}/2^{2n}$$

Considere agora 1.000 simulações para as trajetórias de S_n para $n \in \{0, 1, ..., 100\}$. Encontre para cada uma destas 1. 000 simulações o instante da última visita à origem. Com estes dados, estime a distribuição do instante da última à origem em trajetórias de tamanho 101. Compare em uma tabela as estimativas encontradas com os valores obtidos pela fórmula descrita no início deste item.

```r
#calculando a proporção das últimas trocas observadas
ultimaobserv <- table(trocas[2,])/1000
#função teórica da distribuição das últimas trocas
teoriaultima <- function(k, n) (choose(2*k, k)*choose(2*n-2*k, n-k))/2^(2*n)
#gerando um vetor com os dados teóricos para comparação
ultimateoric <- numeric(0)
for(k in 0:(length(ultimaobserv)-1)){
  ultimateoric <- c(ultimateoric, teoriaultima(k, 50))
}
names(ultimateoric) <- c(seq(0, 2*(length(trocasobserv)-1), 2))

#tabela de comparação do observado com o teórico
ultimastabela <- rbind(ultimaobserv, ultimateoric)
ultimastabela
```

```
##                       0          2          4          6          8
## ultimaobserv 0.15600000 0.03800000 0.02100000 0.02400000 0.02600000
## ultimateoric 0.07958924 0.04019658 0.03045824 0.02564904 0.02268423
##                      10         12         14         16         18
## ultimaobserv 0.02800000 0.01900000 0.01300000 0.02300000 0.00900000
## ultimateoric 0.02064016 0.01913273 0.01797032 0.01704537 0.01629237
##                      20         22         24         26         28
## ultimaobserv 0.00900000 0.01500000 0.01000000 0.01600000 0.01000000
## ultimateoric 0.01566883 0.01514593 0.01470336 0.01432635 0.01400394
##                     30         32         34         36        38
## ultimaobserv 0.0120000 0.01400000 0.01600000 0.01900000 0.0090000
## ultimateoric 0.0137278 0.01349155 0.01329018 0.01311979 0.0129773
```

```
##                       40         42         44         46         48
## ultimaobserv 0.0130000 0.01300000 0.01200000 0.01800000 0.00700000
## ultimateoric 0.0128603 0.01276688 0.01269561 0.01264543 0.01261561
##                        50         52         54         56         58
## ultimaobserv 0.01400000 0.02100000 0.01300000 0.01300000 0.01500000
## ultimateoric 0.01260571 0.01261561 0.01264543 0.01269561 0.01276688
##                       60        62         64         66         68
## ultimaobserv 0.0100000 0.0080000 0.01300000 0.01200000 0.01300000
## ultimateoric 0.0128603 0.0129773 0.01311979 0.01329018 0.01349155
##                       70         72         74         76         78
## ultimaobserv 0.0120000 0.01100000 0.01400000 0.01800000 0.01900000
## ultimateoric 0.0137278 0.01400394 0.01432635 0.01470336 0.01514593
##                        80         82         84         86         88
## ultimaobserv 0.01100000 0.02300000 0.02200000 0.02600000 0.01900000
## ultimateoric 0.01566883 0.01629237 0.01704537 0.01797032 0.01913273
##                        90         92         94         96         98
## ultimaobserv 0.02400000 0.02200000 0.03000000 0.02500000 0.04200000
## ultimateoric 0.02064016 0.02268423 0.02564904 0.03045824 0.04019658
```

2. Vamos pensar agora no problema conhecido como **Ruína do jogador**, ou seja, um passeio aleatório simples ($p \in [0,1]$) em $\mathbb{Z}$, partindo de x ($S_0 = x$), com barreiras absorventes em c e d, com $c \leq x \leq d$. Inicialmente definimos $T_c = inf_n\{S_n = c\}$ e $T_d = inf_n\{S_n = d\}$. Da teoria dos processos estocásticos, sabemos que para $p \neq \frac{1}{2}$, $P(T_d = T_c = \text{inf}) = 0$ e

$$P_x(T_d < T_c) = \frac{1-(q/p)^{x-c}}{1-(q/p)^{d-c}}$$

a. Considere o um passeio onde p seja igual à probabilidade de vencer uma rodada na roleta americana, ou seja, $p = 18/38$(portanto $q = 20/38$). Imagine um jogador que começa com $x = \{10, 20, 30, 40\}$ e decide jogar até chegar em $50,00 ou até perder tudo. Faça 1.000 simulações para cada uma das fortunas iniciais $x = \{10, 20, 30, 40\}$ de modo a comparar os resultados simulados com os obtidos pela fórmula acima.

```
gambler_ruin <- function(pwin,
                         bet,
                         winvalue,
                         losevalue = 0){

  #função que simula um apostador jogando roleta
  #o parâmetro (pwin) define a probabilidade de vitória
  #o parâmetro (bet) define a aposta inicial
  #o parâmetro (winvalue) define quanto o apostador deseja acumular para parar
  #o parâmetro (losevalue) define o limite de perdas do apostador

  #variável que conta a quantidade de rodadas até atingir um dos limites
  #quando gerarmos os dados da simulação, faremos um resumo dessa variável
  spins <- 0
  #inicialização das variáveis a serem retornadas
  result <- data.frame(WIN = logical(), SPINS = numeric())

  #inicializa a soma acumulada com a aposta inicial
  soma <- bet
  #iterar até acumular o desejado ou perder tudo
  while(soma != winvalue){
    spins <- spins + 1
```

```r
    try <- sample(c(1, -1), 1, prob = c(pwin, (1-pwin)))
    soma <- soma + try
    if(soma == 0){
      #se perder tudo
      result[1,1] <- FALSE
      result[1,2] <- spins
      return(result)
    }
  }
  result[1,1] <- TRUE
  result[1,2] <- spins
  return(result)
}

roulette <- function(n, pwin, bet, winvalue, losevalue = 0){

  #função que simula (n) tentativas da função gambler_ruin
  #a função retorna um dataframe com os seguintes resultados:
  #
  # WIN - variável lógica: TRUEs são vitórias, FALSEs são derrotas
  # SPINS - variável quantitativa: quantidade de rodadas para uma tentativa
  # (é interessante notar o comportamento dessa variável:
  #  Aparentemente, ela está associada a uma relação entre a distância do
  #  valor inicial para os limites de parada e a probabilidade de vitória.
  #  Seu valor máximo parece representar igual probabilidade de resultado final)

  results <- data.frame(WIN = logical(),
                        SPINS = numeric())
  for(i in 1:n) results[i,] <- gambler_ruin(pwin, bet, winvalue, losevalue)
  return(results)
}

#x = 10
#esperado
(1-(20/18)^10)/(1-(20/18)^50)
```

```
## [1] 0.009676981
```

```r
#observado
summary(roulette(1000, 18/38, 10, 50))
```

```
##     WIN              SPINS
##  Mode :logical   Min.   :  10.0
##  FALSE:987       1st Qu.:  54.0
##  TRUE :13        Median : 108.0
##                  Mean   : 191.3
##                  3rd Qu.: 246.0
##                  Max.   :1810.0
```

```r
#x = 20
#esperado
(1-(20/18)^20)/(1-(20/18)^50)
```

```
## [1] 0.03743029
```

```r
#observado
summary(roulette(1000, 18/38, 20, 50))
```

```
##     WIN              SPINS
##  Mode :logical   Min.   :  34.0
##  FALSE:959       1st Qu.: 140.0
##  TRUE :41        Median : 248.0
##                  Mean   : 329.1
##                  3rd Qu.: 424.0
##                  Max.   :1808.0
```

```r
#x = 30
#esperado
(1-(20/18)^30)/(1-(20/18)^50)
```

```
## [1] 0.117026
```

```r
#observado
summary(roulette(1000, 18/38, 30, 50))
```

```
##     WIN              SPINS
##  Mode :logical   Min.   :  52.0
##  FALSE:882       1st Qu.: 242.0
##  TRUE :118       Median : 388.0
##                  Mean   : 454.7
##                  3rd Qu.: 580.0
##                  Max.   :2400.0
```

```r
#x = 40
#esperado
(1-(20/18)^40)/(1-(20/18)^50)
```

```
## [1] 0.3453043
```

```r
#observado
summary(roulette(1000, 18/38, 40, 50))
```

```
##     WIN              SPINS
##  Mode :logical   Min.   :  12.0
##  FALSE:688       1st Qu.: 171.5
##  TRUE :312       Median : 381.0
##                  Mean   : 438.0
##                  3rd Qu.: 614.0
##                  Max.   :2324.0
```

b. Considere agora o passeio simétrico, ou seja, $p = \frac{1}{2}$. Faça o mesmo que foi pedido no item acima, plotando as probabilidades obtidas pela simulação (tambem para $x = \{10, 20, 30, 40\}$) e inferindo sobre qual função melhor se ajusta ao conjunto de pontos obtidos. Fato: $P_x(T_d < T_c)$ é linear em x (fortuna inicial).

```r
#x = 10
#esperado
10/50
```

```
## [1] 0.2
```

```r
#observado
summary(roulette(1000, 1/2, 10, 50))
```

```
##      WIN               SPINS
##  Mode :logical   Min.   :  12.0
##  FALSE:809       1st Qu.:  78.0
##  TRUE :191       Median : 207.0
##                  Mean   : 397.1
##                  3rd Qu.: 562.5
##                  Max.   :3970.0
```

```r
#x = 20
#esperado
20/50
```

```
## [1] 0.4
```

```r
#observado
summary(roulette(1000, 1/2, 20, 50))
```

```
##      WIN               SPINS
##  Mode :logical   Min.   :  34.0
##  FALSE:597       1st Qu.: 238.0
##  TRUE :403       Median : 433.0
##                  Mean   : 594.7
##                  3rd Qu.: 818.0
##                  Max.   :3908.0
```

```r
#x = 30
#esperado
30/50
```

```
## [1] 0.6
```

```r
#observado
summary(roulette(1000, 1/2, 30, 50))
```

```
##      WIN               SPINS
##  Mode :logical   Min.   :  42.0
##  FALSE:390       1st Qu.: 240.0
##  TRUE :610       Median : 466.0
##                  Mean   : 629.5
##                  3rd Qu.: 851.0
##                  Max.   :4310.0
```

```r
#x = 40
#esperado
40/50
```

```
## [1] 0.8
```

```r
#observado
summary(roulette(1000, 1/2, 40, 50))
```

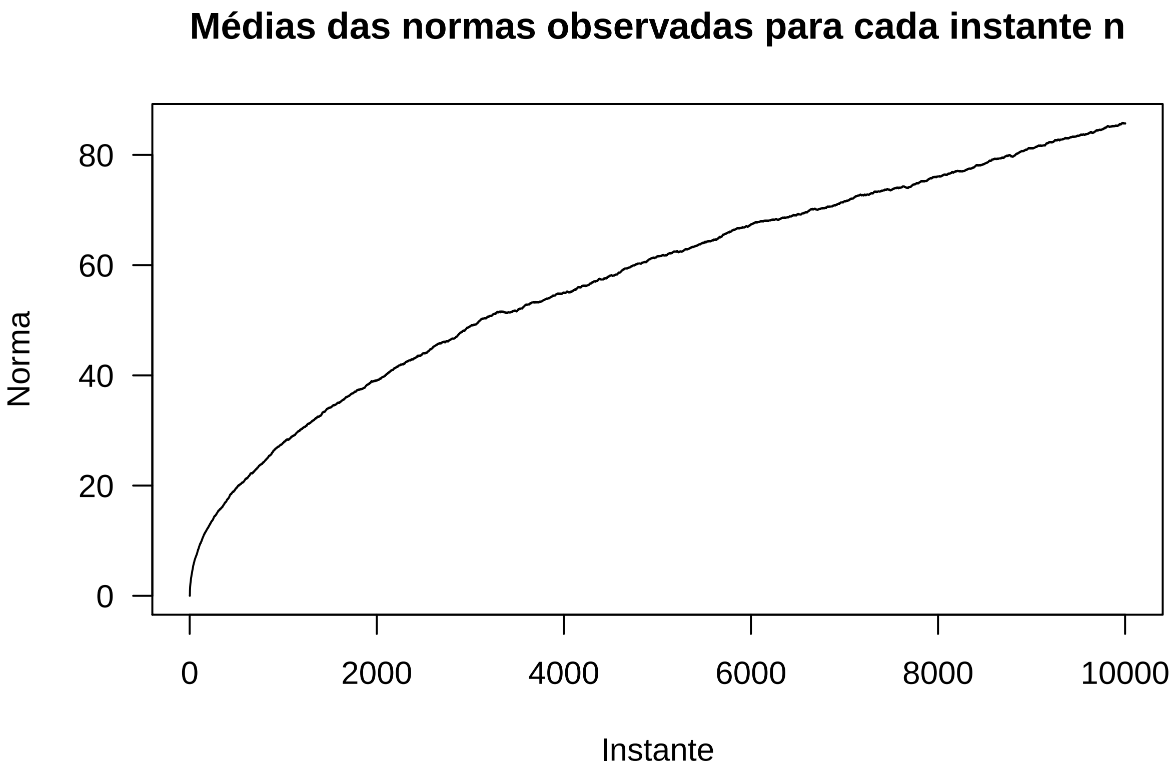
```
##      WIN               SPINS
##  Mode :logical   Min.   :  10.0
##  FALSE:196       1st Qu.:  73.5
##  TRUE :804       Median : 191.0
##                  Mean   : 380.0
##                  3rd Qu.: 500.0
##                  Max.   :3078.0
```

3. Considere agora S_n como sendo o passeio aleatório simples simétrico em $\mathbb{Z}^{\not\vDash}$, partindo da origem, ou

seja, $S_0 = (0,0)$. Calcule a média da distância Euclidiana em relação à origem do passeio observada nos instantes $n = \{1.000, 2.000, ..., 10.000\}$ para 1.000 simulações(para $S_n = (x_n, y_n)$, definimos a distância Euclidiana como $||Sn||_2 = \sqrt{x_n^2 + y_n^2}$). Plote estes pontos em um gráfico, indicando uma função para imitar o comportamento de $E[||Sn||_2]$.

```r
PASS2D <- function(n){

  #função que gera os dados de um passeio aleatório em Z² de tamanho (n)

  #passos possíveis
  orderedpairs <- rbind(c(1,0),c(-1,0),c(0,1),c(0,-1))
  #gerando os passos
  steps <- orderedpairs[sample(nrow(orderedpairs), size=n, replace=TRUE),]
  #inicializando a matriz do passeio vazia
  walk <- matrix(c(0, 0), ncol = 2)
  for(s in 1:(nrow(steps))){
    #preenchendo a matriz do passeio
    walk <- rbind(walk, walk[s,] + steps[s,])
  }
  return(walk)
}

#biblioteca para tratar com matrizes de mais de 2 dimensões
library(abind)

run2D <- function(runs, n){

  #função que gera uma matriz em 3 dimensões com simulações de
  #passeios aleatórios em Z².
  #a primeira dimensão da matriz é o tamanho do passeio aleatório (n)
  #a segunda dimensão da matriz são os pares ordenados em Z² (2)
  #a terceira dimensão da matriz são as simulações (runs)

  #inicializando matriz em 3 dimensões
  result <- array(dim = c(n+1, 2, 0))
  #barra de progresso
  #pb <- txtProgressBar(min = 0, max = runs, style = 3)
  #gerar um passeio aleatório para cada simulação
  for(i in 1:runs){
    #adicionar o passeio à matriz
    result <- abind(PASS2D(n), result, along = 3)
    #atualizar barra de progresso
    #setTxtProgressBar(pb, i)
  }
  #fechar barra de progresso
  #close(pb)
  #retornar dados
  return(result)
}

#gerando os dados para análise
dados <- run2D(1000, 10000)

#função para calcular a média das normas observadas
```

```r
observednorm <- function(x) mean(sqrt(colSums((dados[x, ,])^2)))

#calcular a norma média observada para todos os pontos
norms <- numeric(0)
for(i in 0:10001) norms <- c(norms, observednorm(i))

#plotar as normas médias
plot(norms,
     type = "l",
     main = "Médias das normas observadas para cada instante n",
     xlab = "Instante",
     ylab = "Norma",
     las = 1)
```




```r
#dado que as normas médias observadas apresentam a curva de uma
#função raiz, vamos descobrir a constante 'c' que ajusta a curva aos dados
roots <- numeric(0)
for(i in 0:10001) roots <- c(roots, sqrt(i))
c <- norms/roots
c <- mean(c, na.rm = TRUE)

#estimamos o valor da constante
c
```

```
## [1] 0.8669153
```

```r
#gerando tabela comparativa dos dados com a distribuição teórica
observado <- numeric(0)
teorico <- numeric(0)
```

```r
for(i in seq(1000, 10000, 1000)){

  observado <- c(observado, observednorm(i))
  teorico <- c(teorico, c*sqrt(i))

}

tabela <- rbind(observado, teorico)
tabela
```

```
##               [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
## observado 27.76441 39.13914 48.90550 54.97676 61.58513 67.41232 71.57267
## teorico   27.41427 38.76963 47.48291 54.82854 61.30017 67.15097 72.53134
##               [,8]     [,9]    [,10]
## observado 76.09738 81.23048 85.73399
## teorico   77.53926 82.24281 86.69153
```