Lista 4 - MAE399

Samira Tokunaga, Yiying Lin, André Vinícius

May 12, 2019

1. Considere $\{S_n\}$ como sendo o passeio aleatório simples (inicialmente $p \in [0,1]$) em Z.

a. Mostre que $E(\{S_n\}) = n(p-q)$ e $Var(\{S_n\}) = 4npq$. Lembre que $q = (1-p)$.

Seja $p \in [0,1]$ e X_i uma variável aleatória definida por:

$$X_i = \begin{cases} +1 & \text{com probabilidade} \quad p \\ -1 & \text{com probabilidade} \quad q = (1-p) \end{cases}$$

Então, a Série $\{S_n\}$ é chamada Passeio Aleatório Simples em $\mathbb{Z}$, onde:

$$S_n = \sum_{i=1}^{n} X_i$$

Além disso, a Esperança e a Variância para a Série são determinadas por:

$$\begin{aligned} E(S_n) = E(\sum_{i=1}^{n} X_i) = \sum_{i=1}^{n} E(X_i) = \sum_{i=1}^{n}\sum_{j=1}^{k} x_j p_j = \\ = \sum_{i=1}^{n}(1)(p) + (-1)(q) = \sum_{i=1}^{n} p - q = n(p-q) \end{aligned}$$

$$\begin{aligned} Var(S_n) = Var(\sum_{i=1}^{n} X_i) = \sum_{i=1}^{n} Var(X_i) = \sum_{i=1}^{n}[E(X_i^2) - E^2(X_i)] = \\ = \sum_{i=1}^{n} E(X_i^2) - \sum_{i=1}^{n} E^2(X_i) = \sum_{i=1}^{n} E(X_i^2) - \sum_{i=1}^{n}(p-q)^2 = \\ = \sum_{i=1}^{n}\sum_{j=1}^{k} x_j^2 p_j - \sum_{i=1}^{n}(p-q)^2 = \sum_{i=1}^{n}(p+q) - \sum_{i=1}^{n}(p-q)^2 = \\ = \sum_{i=1}^{n}(p+q) - (p-q)^2 = \sum_{i=1}^{n} p(1-p) + q(1-q) + 2pq = \\ = \sum_{i=1}^{n} p(q) + q(p) + 2pq = \sum_{i=1}^{n} 4pq = 4npq \end{aligned}$$

b. Considere $S_0 = 0$ e o passeio aleatório simétrico, ou seja, $p = 1/2$. Faça 1.000 simulações para as trajetórias de S_n com $n \in [0, 1, ..., 10.000]$. Para os instantes $n \in [1.000, 2.000, ..., 10.000]$, encontre os valores x_{inf} e x_{sup} que coloquem respectivamente abaixo e acima deles, 1% e 99% dos valores S_n simulados.

```
PASS1D <- function(n){

  #função que gera um passeio aleatório simples simétrico de tamanho "n"
  steps <- sample(c(-1, 1), (n), replace = TRUE) #passos
```

```r
  walk <- numeric(1) #inicializando o vetor do passeio
  for(s in 1:length(steps)){
    #gerando o passeio
    walk <- c(walk, walk[s] + steps[s])
  }
  #retorna o passeio
  return(walk)
}

run1D <- function(runs, n, type = PASS1D){#or type = BrowBridge1D

  #função que gera uma matriz de passeios aleatórios de tamanho "n"
  #cada linha da matriz se refere a um passeio aleatório diferente
  #a quantidade de passeios é especificada pelo parâmetro "runs"

  result <- matrix(nrow = 0, ncol = n + 1)#inicializa a matriz
  #pb <- txtProgressBar(min = 0, max = runs, style = 3)#barra de progresso
  for(i in 1:runs){
    result <- rbind(result, type(n)) #preenche a linha da matriz
    #setTxtProgressBar(pb, i)#incrementa a barra de progresso
  }
  #close(pb)#fecha barra de progresso
  return(result)#retorna a matriz
}

#gerando a matriz dos dados
matriz <- run1D(1000, 10000)

#inicializando os vetores dos x_inf, x_sup
x_inf_simulado <- numeric(0)
x_sup_simulado <- numeric(0)

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

  #coletando os dados das colunas da matriz
  xzes <- quantile(matriz[,i], probs = c(0.01, 0.99))
  x_inf_simulado <- c(x_inf_simulado, xzes[1])
  x_sup_simulado <- c(x_sup_simulado, xzes[2])

}

#renomeando as posições do vetor
names(x_inf_simulado) <- c(1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000)
names(x_sup_simulado) <- c(1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000)

#mostrando os resultados
x_inf_simulado
```

```
##    1000    2000    3000    4000    5000    6000    7000    8000    9000
##  -77.00  -99.00 -131.02 -143.04 -159.04 -177.06 -181.04 -197.02 -209.00
##   10000
## -251.02
```

```
x_sup_simulado
```

```
##    1000    2000    3000    4000    5000    6000    7000    8000    9000   10000
##   71.00   99.00 129.04 147.00 157.00 171.04 197.04 211.02 249.06 241.00
```

c. Baseado no Teorema Central do Limite, encontre os limites de variação centrados na média teórica, ou seja, $n(p-q)=0$, contendo 98% dos valores possíveis (análogo a x_{inf} e x_{sup} do item anterior), para o passeio aleatório, nos instantes $n \in [1.000, 2.000, ..., 10.000]$.

Como temos $p=q=1/2$, então:

$$E(\{S_n\}) = n(p-q) = n(0) = 0$$
$$Var(\{S_n\}) = 4npq = 4n/4 = n$$

Partindo disto, e usando a tabela normal, chegamos a:

```
#cada um dos n
pontos <- c(1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000)

#x_inf
x_inf_teorico <- -2.33*sqrt(pontos)

#x_sup
x_sup_teorico <- 2.33*sqrt(pontos)
```

d. Faça uma tabela comparativa com os valores x_{inf} e x_{sup} obtidos nos itens b e c.

```
rbind(x_inf_simulado, x_inf_teorico, x_sup_simulado, x_sup_teorico)
```

```
##                     1000      2000      3000      4000      5000      6000
## x_inf_simulado -77.00000  -99.0000 -131.0200 -143.0400 -159.0400 -177.060
## x_inf_teorico  -73.68107 -104.2008 -127.6194 -147.3621 -164.7559 -180.481
## x_sup_simulado  71.00000   99.0000  129.0400  147.0000  157.0000  171.040
## x_sup_teorico   73.68107  104.2008  127.6194  147.3621  164.7559  180.481
##                     7000      8000      9000     10000
## x_inf_simulado -181.0400 -197.0200 -209.0000 -251.02
## x_inf_teorico  -194.9418 -208.4015 -221.0432 -233.00
## x_sup_simulado  197.0400  211.0200  249.0600  241.00
## x_sup_teorico   194.9418  208.4015  221.0432  233.00
```

e. Faça 5 simulações para as trajetórias de $\{S_n\}$ com $n \in [0, 1, ..., 10.000]$ e plote as trajetórias em um gráfico onde apareçam também os valores x_{inf} e x_{sup} análogos aos obtidos no item c, mas todos os valores de $n \in [0, 1, ..., 10.000]$.

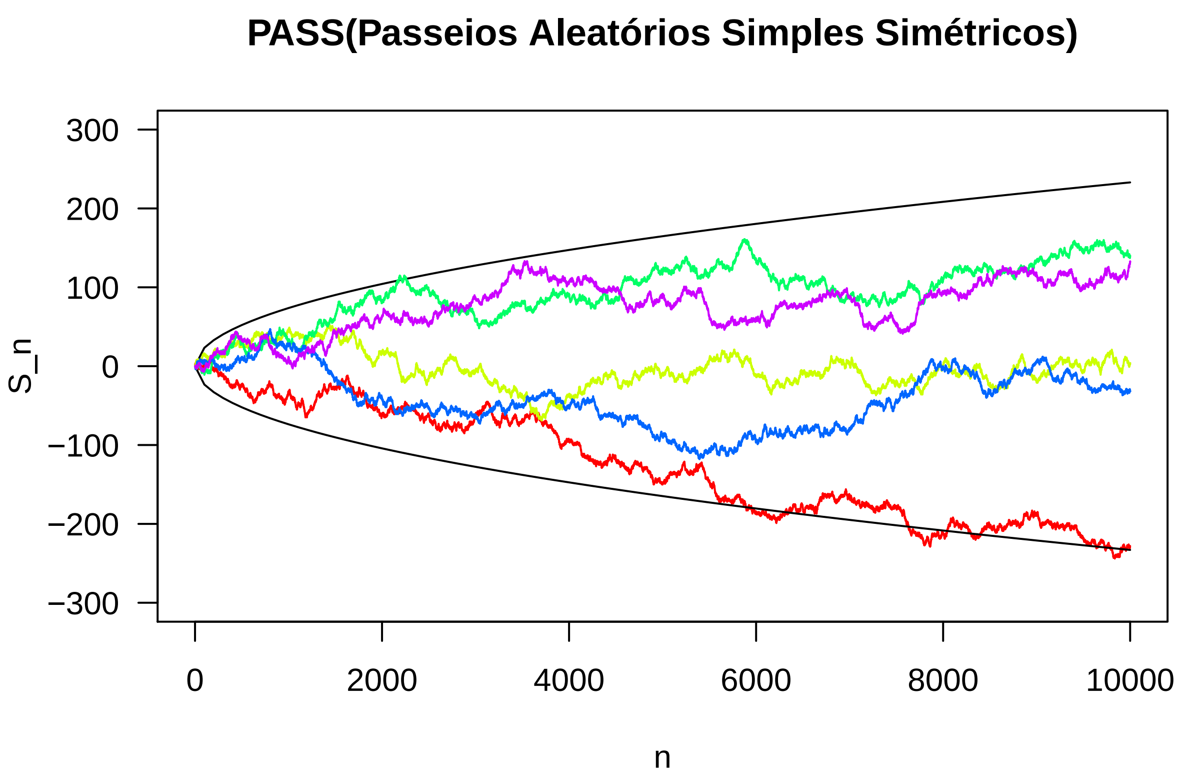
```
plot1D <- function(matriz){

  #esta função plota os dados da matriz gerada pela função run1D
  n <- ncol(matriz) #tamanho do passeio aleatório
  s <- sqrt(n) #desvio padrão do passeio aleatório
  cores <- rainbow(nrow(matriz))
  plot(0:(n-1), #xizes
       matriz[1,], #yzes
       main = "PASS(Passeios Aleatórios Simples Simétricos)",
       xlab = "n", #label dos xzes
       ylab = expression("S_n"), #label dos yzes
       las = 1,
       type = "l", #gráfico de linhas
```

```
      ylim = c(-3*s, 3*s), #limites superior e inferior do gráfico
      col = cores[1]) #cores das linhas
  curve(2.33*(sqrt(x)), add = TRUE) #limite de 99% dos dados
  curve(-2.33*(sqrt(x)), add = TRUE) #limite de 1% dos dados
  #inicializando barra de progresso
  #pb <- txtProgressBar(min = 0, max = nrow(matriz), style = 3)
  for(i in 2:nrow(matriz)){

    #iterando sobre os passeios aleatórios da matriz
    lines(matriz[i,], col = cores[i]) #plotando cada uma das linhas
    #setTxtProgressBar(pb, i) #atualiza barra de progresso
  }
  #close(pb) #fecha barra de progresso
}

cincosimulacoes <- run1D(5, 10000)
plot1D(cincosimulacoes)
```



2. Considere agora uma ponte com o passeio aleatório simples, ou seja, S_n com $n \in 0, 1, ..., 10.000$ tal que, $S_0 = 0$ e $S_{10000} = 0$.

a. Faça 1.000 simulações para as trajetórias de S_n. Para os instantes $n \in 0, 1, ..., 10.000$, encontre os valores x_{inf} e x_{sup} que coloquem respectivamente abaixo e acima deles, 1% e 99% dos valores S_n simulados. Baseado nas trajetórias indique uma função que modele a evolução dos valores encontrados para x_{inf} e x_{sup}.

```
BrowBridge1D <- function(n){

  #esta função gera uma ponte browniana de tamanho "n"
  steps <- sample(c(rep(-1, n/2), rep(1, n/2)), n, replace = FALSE)
```

```r
  walk <- numeric(1)#inicializando o vetor
  for(s in 1:length(steps)){
    #gerando o passeio
    walk <- c(walk, walk[s] + steps[s])
  }
  #retornando os dados
  return(walk)
}

#gerando a matriz dos dados
browmatriz <- run1D(1000, 10000, type = BrowBridge1D)

#inicializando os vetores dos x_inf, x_sup
x_inf_bsimulado <- numeric(0)
x_sup_bsimulado <- numeric(0)

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

  #coletando os dados das colunas da matriz
  xzes <- quantile(browmatriz[,i], probs = c(0.01, 0.99))
  x_inf_bsimulado <- c(x_inf_bsimulado, xzes[1])
  x_sup_bsimulado <- c(x_sup_bsimulado, xzes[2])

}

#renomeando as posições do vetor
names(x_inf_bsimulado) <- c(1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000)
names(x_sup_bsimulado) <- c(1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000)

#mostrando os resultados
x_inf_bsimulado
```

```
##    1000    2000    3000    4000    5000    6000    7000    8000    9000
##  -73.02 -101.00 -103.02 -105.00 -111.02 -103.00 -105.00  -95.02  -69.00
##   10000
##   -1.00
```

```r
x_sup_bsimulado
```

```
##   1000   2000   3000   4000   5000   6000   7000   8000   9000  10000
##  69.02  95.04 115.02 115.00 113.02 105.00 101.06  87.02  69.00   1.00
```

A curva descrita pela variância da ponte browniana é uma parábola, gerada por uma função quadrática. Pesquisando, encontramos a fórmula $Var(B_t) = \frac{t(T-t)}{T}$, onde t é um instante qualquer tal que $0 \leq t \leq T$, sendo T o ponto final.

b. Tome 5 simulações para as trajetórias de S_n com $n \in 0, 1, ..., 10.000$ e plote as trajetórias em um gráfico onde apareçam também os valores x_{inf} e x_{sup} obtidos no item anterior.

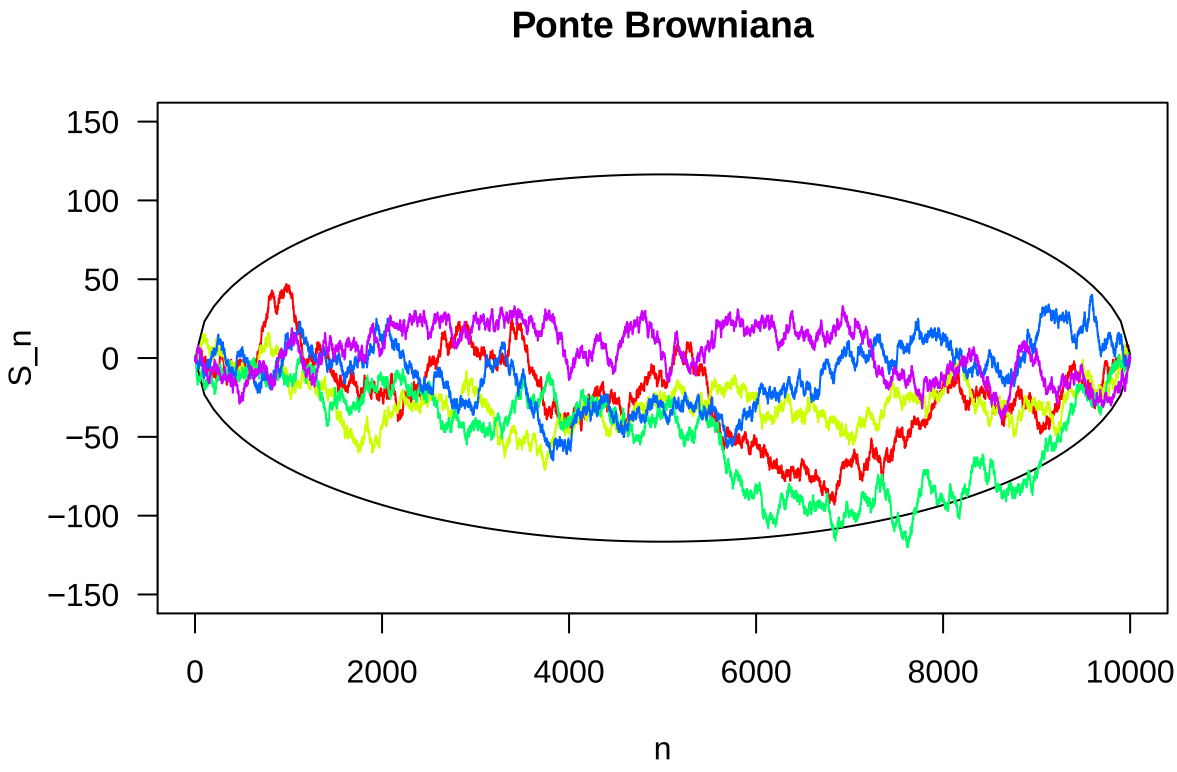
```r
plot1D <- function(matriz){

  #esta função plota os dados da matriz gerada pela função run1D
  n <- ncol(matriz) #tamanho do passeio aleatório
  s <- sqrt(((n/2)^2)/n) #desvio padrão da ponte browniana
  cores <- rainbow(nrow(matriz))
  plot(0:(n-1), #xizes
```

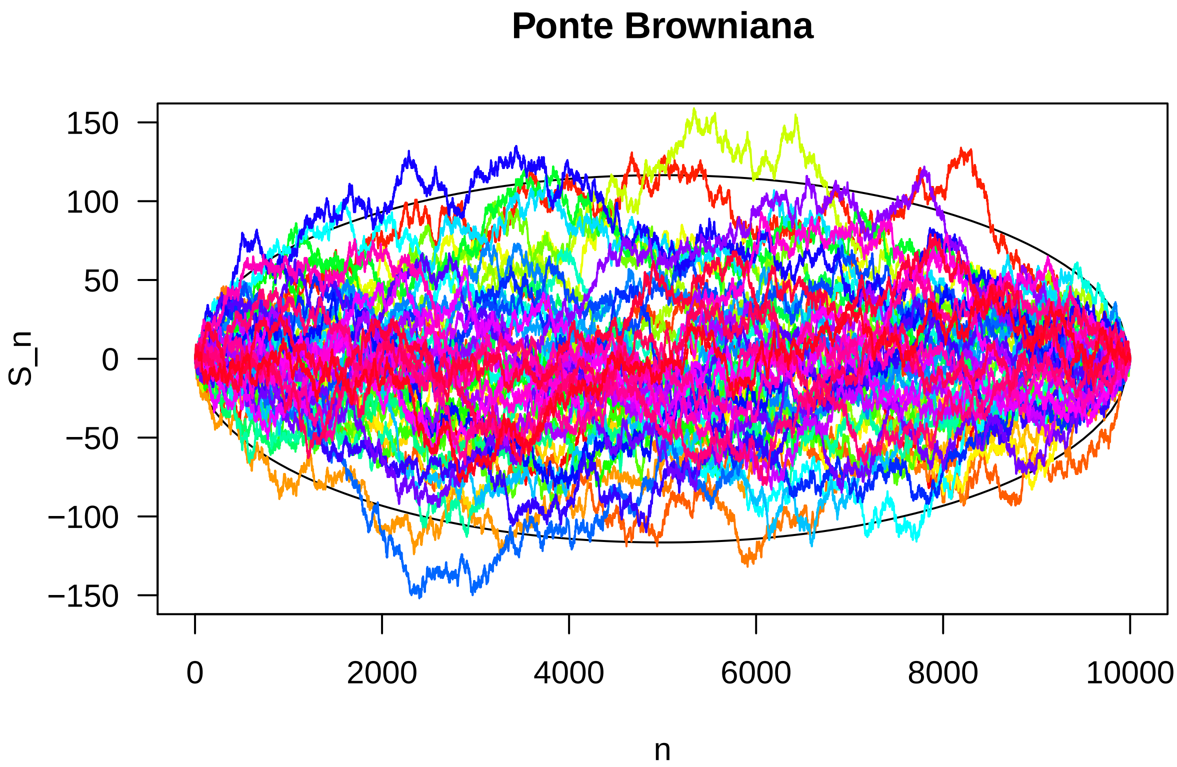
```
       matriz[1,], #yzes
       main = "Ponte Browniana",
       xlab = "n", #label dos xzes
       ylab = expression("S_n"), #label dos yzes
       las = 1,
       type = "l", #gráfico de linhas
       ylim = c(-3*s, 3*s), #limites superior e inferior do gráfico
       col = cores[1]) #cores das linhas
  curve(2.33*sqrt((x*abs(x-n))/n), add = TRUE) #limite de 1% dos dados
  curve(-2.33*sqrt((x*abs(x-n))/n), add = TRUE) #limite de 99% dos dados
  #inicializando barra de progresso
  #pb <- txtProgressBar(min = 0, max = nrow(matriz), style = 3)
  for(i in 2:nrow(matriz)){

    #iterando sobre os passeios aleatórios da matriz
    lines(matriz[i,], col = cores[i]) #plotando cada uma das linhas
    #setTxtProgressBar(pb, i) #atualiza barra de progresso
  }
  #close(pb) #fecha barra de progresso
}

cincosimulacoesb <- run1D(5, 10000, type = BrowBridge1D)
plot1D(cincosimulacoesb)
```




```
cincoentasimulacoesb <- run1D(50, 10000, type = BrowBridge1D)
plot1D(cincoentasimulacoesb)
```


Ponte Browniana
150
100
50
0
−50
−100
−150
S_n
0
2000
4000
6000
8000
10000
n