# SME0806 - Estatística Computacional - Trabalho 1

21/05/2021


## Alunos:

- Aline Fernanda da Conceição, 9437275
- Diego J Talarico Ferreira, 3166561
- Matheus Victal Cerqueira, 10276661
- Murilo Henrique Soave, 10688813
- Nelson Calsolari Neto, 10277022

## Docente: Professor Dr. Mário de Castro


## Introdução

O presente documento se trata de uma solução para os exercícios propostos no Trabalho 1 da disciplina SME0806 - Estatística Computacional, oferecida pelo Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo no primeiro semestre de 2021. As temáticas abordadas são métodos computacionais para a geração de amostras aleatórias e aplicação de simulações de Monte Carlo para a resolução de problemas estatísticos.

## Desenvolvimento e Metodologia

## Exercício 1

No primeiro exercício sugerido, tem-se o objetivo de implementar um gerador de amostras pseudoaleatórias para uma variável aleatória $X$ de interessse. A função densidade de probabilidade de $X$ é dada por:

$$f(x) \propto q(x) = exp\left\{-|x|^3/3\right\}, x \in \mathbb{R}$$

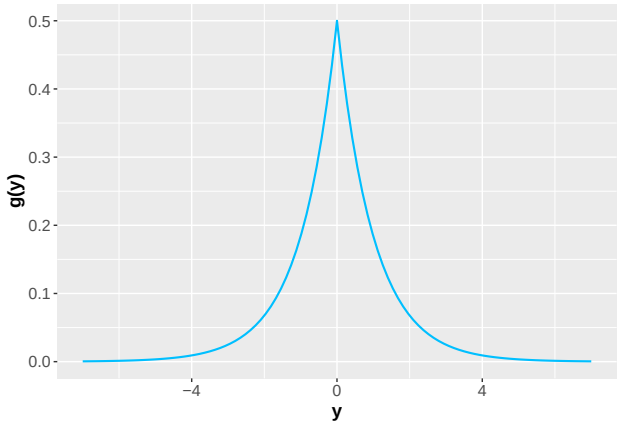
É notório que a função $q(x)$ se trata de um *kernel* para uma função de distribuição, e para obter uma função de distribuição de fato seria necerrária a multiplicação de $q(x)$ por uma contante normalizadora. Porém, pode-se gerar amostras pseudoaleatórias de $X$ apenas com o conhecimento da função *kernel* $q(x)$, e por tal motivo, não será obtido o valor de tal constante no decorrer desta solução.

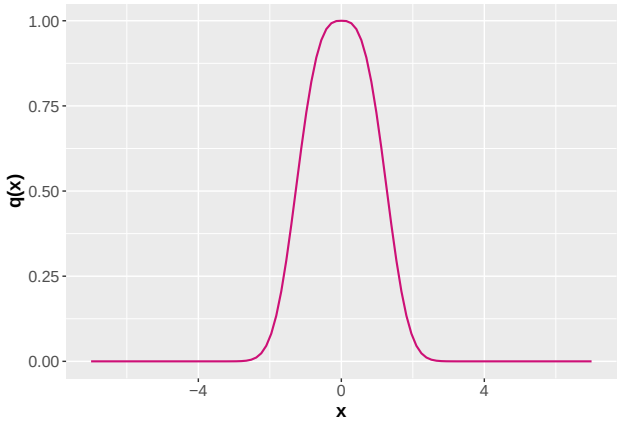
Considerando-se o comportamento da função $f(x)$, optou-se pelo método da aceitação-rejeição para obter-se a amostra de interesse. Para que o método seja implementado, é necessário a utilização de uma variável aleatória auxiliar $Y$ da qual possamos obter amostras psudoaleatórias computacionalmente. A função de densidade $g(y)$ escolhida para $Y$ foi a distribuição de Laplace padrão:

$$g(y) = \frac{1}{2} \cdot exp\{-|y|\}$$

Tal função $g(y)$ é interessante para a resolução do problema via método da aceitação-rejeição devido ao fato de podermos gerar amostras psudoaleatórias dela facilmente pelo método da inversão e pelo fato de seu comportamento permitir que ela, ao ser multiplicada por um fator $M$, seja capaz de envelopar $q(x)$, condição necessária para a aplicação do método. Abaixo encontram-se gráficos para $q(x)$ e $g(y)$. Repare que ambas possuem o mesmo domínio $\mathbb{R}$.

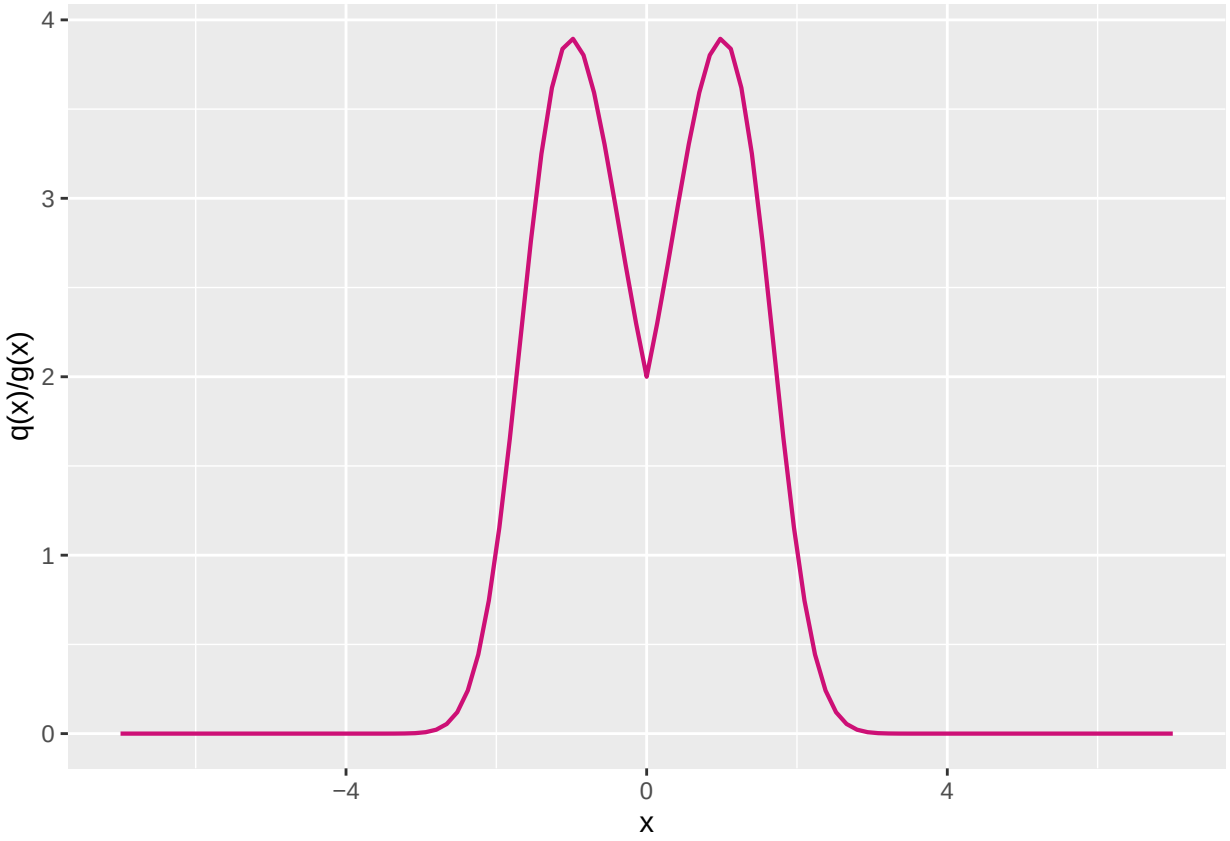
```
rm(list=ls(all=TRUE))
library(ggplot2)

# Função densidade de probabilidade de Laplace (g(y): função auxiliar)
d_lap <-function(x) {
  return(0.5* exp(-abs(x)))
}

# Função kernel de densidade da v.a. X da qual se quer obter uma amostra
q_x <-function(x) {
  return(exp(-(abs(x)^3)/3))
}

# Gráfico de g(x)
ggplot() +
  xlim(-7,7) +
  geom_function(fun = d_lap, colour = "deepskyblue1",size = 0.75) +
  xlab("y") +
  ylab("g(y)") +
  theme(axis.text=element_text(size=12),
        axis.title=element_text(size=14,face="bold"))

# Gráfico de q(x)
ggplot() +
  xlim(-7,7) +
  geom_function(fun = q_x, colour = "deeppink3",size = 0.75) +
  xlab("x") +
  ylab("q(x)") +
  theme(axis.text=element_text(size=12),
        axis.title=element_text(size=14,face="bold"))
```




Tendo-se a função auxiliar $g(y)$, pode-se dar prosseguimento à implementação do método. Obtenhamos o valor de $M$ a partir do valor máximo assumido pela função $qg(x) = q(x)/g(x)$. Tal valor corresponde ao valor de $M$ ótimo para o problema. Analisemos o gráfico para a função $qg(x)$:

```r
# Função q(x)/g(x), da qual queremos obter o valor máximo para atribuir a M
qgx <-function(x) {
  return(q_x(x)/ d_lap(x))
  }


# Gráfico de q(x)/g(x)
ggplot() +
  xlim(-7,7) +
  geom_function(fun = qgx, colour = "deeppink3",size = 0.75) +
  xlab("x") +
  ylab("q(x)/g(x)")
```



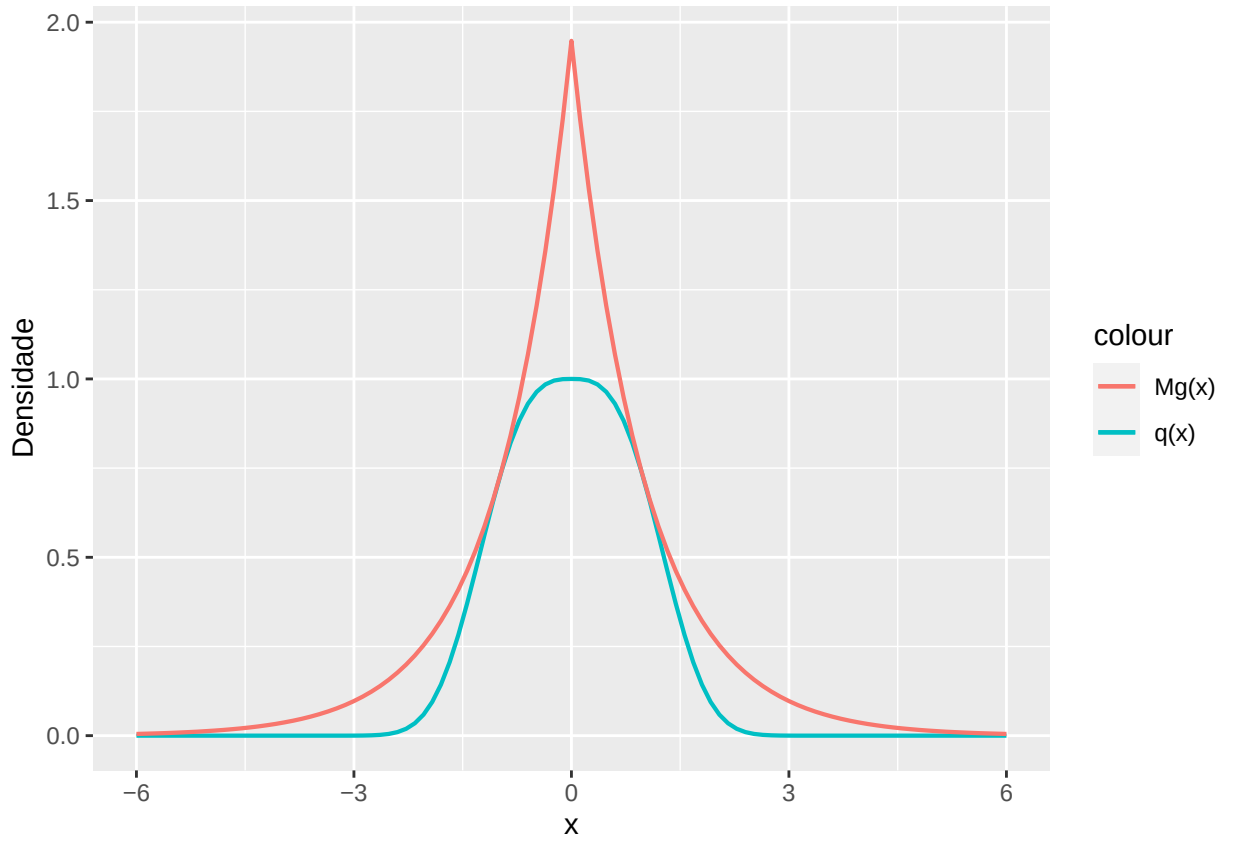

Como $q(x)$ e $g(x)$ são simétricas, a função $qg(x)$ tambpem é simétrica, como pode ser confirmado pelo gráfico acima. Pode-se notar que existem 2 máximos globais para a função acima, os quais são iguais e, devido à simetria da função, são a imagem de dois valores no domínio que são iguais em módulo. Pode-se utilizar da função *optimize* para obter o valor máquimo de $qg(x)$ e o valor positivo $x_0$ do domínio que leva a tal valor:

```r
# Obtenção do valor positivo de x que otimiza a função q(x)/g(x)
x_max <- optimize(qgx, interval=c(-7,7), maximum=TRUE)

M <- qgx(x_max$maximum)

cat("\n Valor máximo assumido por qg(x):", M,
    "\n Valor correspondente no domínio:", x_max$maximum)
```

```
##
##  Valor máximo assumido por qg(x): 3.895468
##  Valor correspondente no domínio: 1.000002
```

Assim, o valor ótimo obtido para a função (aproximadamente 3,89; obtido quando $x \in \{-1, 1\}$) será atribuido à $M$ para a obtenção da função "cobertor" $Mg(x)$. Podemos visualizar o comportamento das duas funções ($q(x)$ e $Mg(x)$) em um mesmo gráfico para observar que de fato ocorre o envelopamento de $q(x)$.

```r
# Função para Mg(x)
Mg_x <-function(x) {
  return(M*d_lap(x))
}

# Gráfico de q(x) e Mg(x)
ggplot() +
  xlim(-6,6) +
  geom_function(aes(colour = "q(x)"), fun = q_x, size = 0.75) +
  geom_function(aes(colour = "Mg(x)"), fun = Mg_x, size = 0.75) +
  xlab("x") +
  ylab("Densidade")
```




Assim sendo, podemos por fim criar uma função para obter uma amostra pseudoaleatória de $X$ por meio do método da aceitação-rejeição:

```r
sample_rej <- function(n){ # função recebe o tamanho da amostra n
```

```
  set.seed(2112) #setup da semente

  i <- n_t <- 0 # Inicialização do contador i para o tamanho da amostra e
  # do contador n_t para o número de tentativas

  a_X <- c() # vetor auxiliar para armazenamento dos valores observados de X

  while(i<n){ # laço para o tamanho da amostra, não é quebrado até o tamanho da
  # amostra ser n
   rej <- TRUE

    while(rej){ # laço para a rejeição, não é quebrado até que um valor seja  aceito
      n_t <- n_t + 1

      # obtenção de uma observação da variável auxiliar pelo método da inversão
      u <-runif(1)
      # a inversa da função de distribuição de Laplace padrão
      y <-ifelse(u<=0.5,log(2*u),-log(2*(1-u)))

      # condição de aceitação do valor observado para y: u <= q(x)/Mg(x),
      # sendo u ~ U(0,1).
      if(runif(1) <= qgx(y)/M){
        i <- i + 1
        a_X[i] <- y
        rej <- FALSE
      }
    }
  }
  cat("\n Tamanho da amostra:", i, "\n Número de tentativas:", n_t)
  return(a_X)
}
```

Agora utilizemos da função *sample_rej* para obter amostras pseudoaleatórias de $X$ de diferentes tamanhos:

```
# Amostra n = 50
aa_50 <- sample_rej(50)
```

```
##
##  Tamanho da amostra: 50
##  Número de tentativas: 65
```

```
# Amostra n = 100
aa_100 <- sample_rej(100)
```

```
##
##  Tamanho da amostra: 100
##  Número de tentativas: 147
```

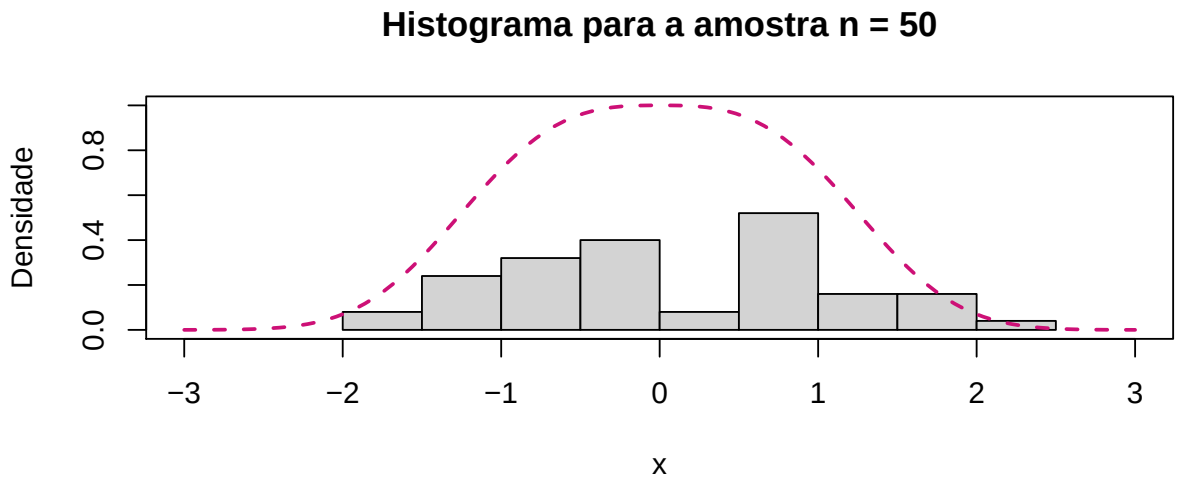
```
# Amostra n = 400
aa_400 <- sample_rej(400)
```

```
##
```

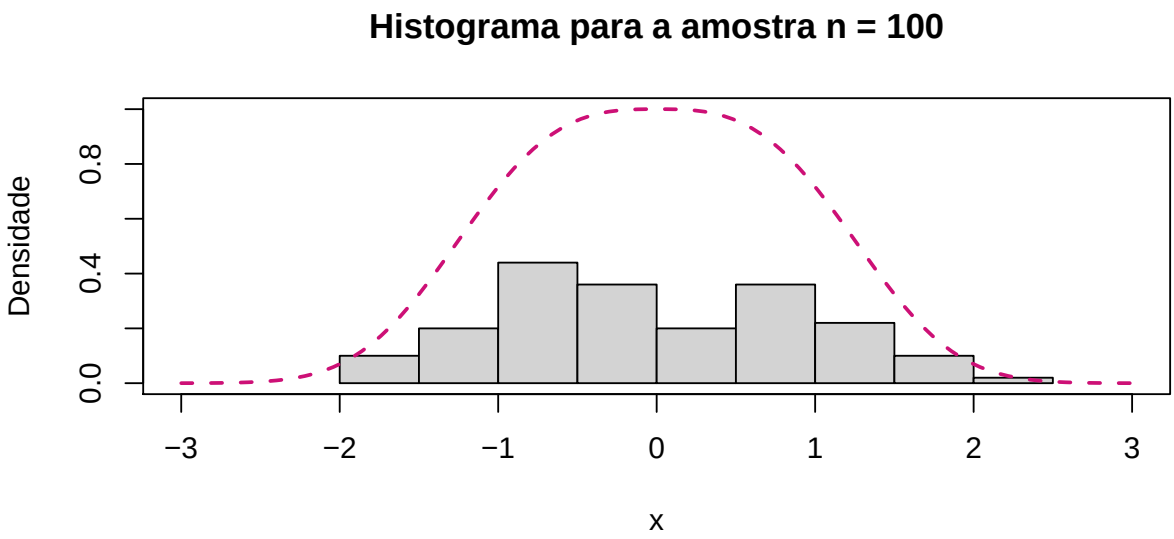
```
##  Tamanho da amostra: 400
##  Número de tentativas: 609
```

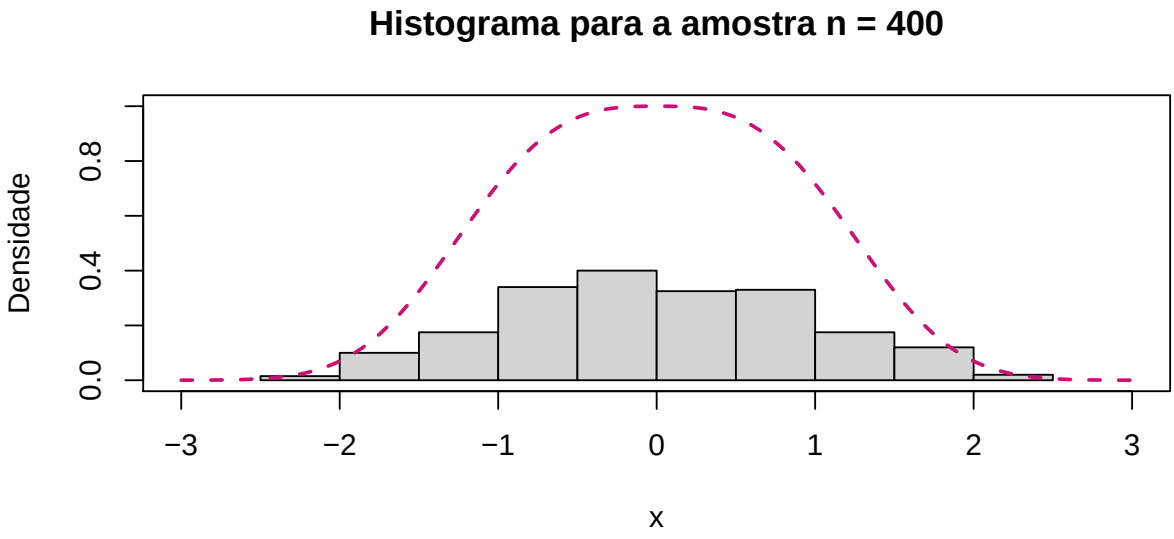
Podemos contruir os histogramas das amostras pseudoaleatórias de $X$ para visualizar a sua distribuição:

```
hist(aa_50, freq = FALSE, main = "Histograma para a amostra n = 50",
     xlab = "x", ylab = "Densidade",
     xlim = c(-3,3), ylim = c(0,1))
curve(q_x, add = TRUE, lty = 2, col = "deeppink3", lwd = 2)
box()
```




repare que os valores da função $q(x)$ (linha rosa racejada) são mais elevados do que os valores do histograma, não ocorrendo uma melhor sobreposição. Isso se deve ao fato de $q(x)$ não ser uma função de densidade, mas sim o *kernel* de uma. De qualquer forma, pode-se compreender como os histogramas deveriam se comportar comparando-se com ela. Os histogramas para as outras amostras foram gerados da mesma forma que o acima e podem ser verificados abaixo.




É notório que o histograma vai se ajustando ao comportamento de $q(x)$ conforme o número de amostras pseudoaleatórias aumenta.

## Exercício 2

No segundo exercício sugerido, existem duas variáveis aleatórias dependentes as quais queremos estudar o comportamento. As duas são modeladas por:

$$X \sim lognormal(0, 1)$$
$$log(Y) = 9 + 3log(X) + \epsilon$$

Sendo que $\epsilon \sim N(0,1)$ e é independente de $X$. Objetiva-se obter uma estimação pontual e intervalar para $E[Y/X]$.

Para obter-se as estimações de interesse, foi realizada uma abordagem por simulação de Monte Carlo. Da teoria das probabilidades, sabe-se pela lei forte dos grandes números que, sendo $W_1, \cdots, W_R$ uma amostra aleatória da variável aleatória $W$:

$$\hat{\theta} = \frac{1}{R}\sum_{j=1}^{R} W_j \to E[W], \text{ quando } R \to \infty \text{ (I)}$$

Sendo $E[W]$ a esperança estatística da variável aleatória $W$. Dessa forma, a abordagem de Monte Carlo para esse problema é da geração de amostras pseudoaleatórias de $W = Y/X$ em número grande o suficiente para obter-se uma estimação pontual para $E[Y/X]$. A partir da mesma amostra, pode-se obter o erro-padrão de Monte Carlo para o estimador $\hat{\theta}$, o que permitirá a obtenção de intervalos de confiança para nossa estimação.

Porém, para realizar o procedimento descrito, é necessária, primeiro, uma amostra aleatória de $W$. Para tal, precisamos entender qual é a distribuição estatística de $Y$. Analisemos a relação entre $Y$ e $X$:

$$log(Y) = 9 + 3log(X) + \epsilon$$

Sabe-se que $X$ possui distribuição $lognormal(0,1)$. Assim, por definição, $log(X) \sim N(0,1)$. Pelas propriedades da distribuição normal, tem-se que $log(Y) \sim N(9,10)$, o que implica que $Y \sim lognormal(9,10)$. Porém, $Y$ é dependente de $X$, isso implica que seu valor depende do valor assumido por $X$. Implementemos uma rotina em R para gerar uma amostra aleatória de $Z$.

```
set.seed(2112)
n <- 1000000 # tamanho da amostra pseudoaleatória
aa <- c() # inicialização do vetor que irá conter os valores gerados.


for(i in 1:n){
  x <- rlnorm(1,0,1) # gera uma amostra de X
  log_x <- log(x) # obtém o valor do log narural do valor assumido por X
  log_y <- 9 + 3*log_x + rnorm(1,0,1) # valor assumido por log natural de y
  y <- exp(log_y) # observação para o valor de y
  aa[i] <- y/x #atribuição do valor y/x à posição i do vetor auxiliar
}
```

Com uma amostra de $W = Y/X$, pode-se agora obter uma estimação pontual $E[W]$ por meio do resultado (I):

```
E <- mean(aa) #média simples dos valores da amostra.
cat("Estimação pontual para E[Y/X]:",E)
```

```
## Estimação pontual para E[Y/X]: 98769.65
```

Assim sendo, uma estimação pontual para $E[Y/X]$ é:

$$\hat{\theta} = 98962.12$$

Para obter um intervalo de confiança para a estimativa pontual obtida, é necessária a obtenção do erro-padrão de Monte Carlo, o qual é definido por:

$$ep(\hat{\theta}) = \sqrt{\widehat{Var}(\hat{\theta})} \text{ (II)}$$

Um estimador para a variância de $\hat{\theta}$ interessante é o seguinte. Sabe-se que a variância de $\hat{\theta}$ é dada por:

$$Var(\hat{\theta}) = Var\left(\frac{1}{R}\sum_{j=1}^{R} W_j\right) = \frac{R \cdot Var(W)}{R^2} = \frac{\sigma^2}{R} \text{ (III)}$$

Sabe-se que um bom estimador para $\sigma^2$ é a variância amostral $s^2$, dada por:

$$s^2(Z) = \frac{1}{R-1}\sum_{j=1}^{R}(Z_j - \hat{\theta})^2$$

Baseando-se em (III) e no resultado acima, podemos obter um bom estimador para a variância de $\hat{\theta}$:

$$\widehat{Var}(\hat{\theta}) = \frac{1}{R}\frac{1}{R-1}\sum_{j=1}^{R}(W_j - \hat{\theta})^2$$

,

Tendo-se tal estimador em mãos, pode-se calcular o valor do erro-padrão de Monte Carlo a partir de (II). A função para $s^2$ já está implementada em R, sendo chamada pelo comando *var*.

```
var_est <- var(aa)/n #Obtenção da variância estimaada
ep <- sqrt(var_est) # erro-padrão de Monte Carlo
cat("Valor obtido para o erro-padrão de Monte Carlo:", ep)
```

```
## Valor obtido para o erro-padrão de Monte Carlo: 892.8703
```

Finalmente, podemos obter o intervalo de confinaça para $\hat{\theta}$ a partir da relação:

$$IC[\theta, \gamma = 1 - \alpha] = [\hat{\theta} \pm Z_{(1-\alpha/2)} ep(\hat{\theta})],$$

Sendo $Z_{(1-\alpha/2)}$ o quantil $(1 - \alpha/2)$ da distribuiçãonormal padrão. Fazendo-se uma função para obter tal intervalo:

```
int_conf <- function(conf){
  a <- (1 - conf)
  cat("\nIC[",conf,"]=[",E-qnorm(1-a/2)*ep,",",E+qnorm(1-a/2)*ep,"]\n")
}
```

Basta aplicar agora a função *int_conf* para obter o intervalo de confiância para $\theta = E[Y/X]$ com o valor $\gamma$ desejado:

```
int_conf(0.8)
```

```
##
## IC[ 0.8 ]=[ 97625.39 , 99913.91 ]
```

```
int_conf(0.9)
```

```
##
## IC[ 0.9 ]=[ 97301.01 , 100238.3 ]
```

```
int_conf(0.95)
```

```
##
## IC[ 0.95 ]=[ 97019.65 , 100519.6 ]
```

```
int_conf(0.99)
```

```
##
## IC[ 0.99 ]=[ 96469.76 , 101069.5 ]
```

## Exercício 3

No terceiro exercício proposto, queremos avaliar um teste para a comparação das seguintes hipóteses:

$$H_0 : \lambda = 2$$
$$H_1 : \lambda > 2$$

com base em uma amostra aleatória de $n$ observações de uma variável aleatória $X \sim Poisson(\lambda)$.

### item a)

Primeiramente, proponhamos um teste estatístico para $H_0$ vs $H_1$. Assim, precisa-se de uma estatística de teste conveniente. Sabe-se que $\hat{\lambda} = \overline{X}$ é um Estimador de Máxima Verossmilhança para o parâmetro $\lambda$ de uma dsitribuição $X \sim Poisson(\lambda)$, assim, obtenhamos uma estatística de teste a partir dele. Sendo $X_1, \cdots, X_n$ uma amostra aleatória de $X \sim Poisson(\lambda)$, pelo Teorema Central do Limite (TCL), sabe-se que:

$$Z = \frac{\hat{\lambda}-\lambda}{\sqrt{\lambda/n}} \xrightarrow{D} N(0,1), \text{ quando } n \to \infty$$

Dessa forma, considerando-se as hipóteses de interece e o TCL, pode-se concluir que sob $H_0$:

$$W = \frac{\hat{\lambda}-2}{\sqrt{2/n}} \xrightarrow[H_0]{D} N(0,1), \text{ quando } n \to \infty$$

Dessa forma, pode-se utilizar a estatística $W$ para realizar-se um teste unilateral à direita, no qual a região crítica $RC$ é dada por:

$$RC = \left\{ \mathbf{x}; \frac{\hat{\lambda}-2}{\sqrt{2/n}} \geq k \right\},$$

sendo $\mathbf{x}$ o vetor de $n$ observações de $X$. Assim sendo, considerando-se um nível de significância $\alpha$ para o teste, pode-se obter o valor de k:

$$\alpha = P\left(\frac{\hat{\lambda}-2}{\sqrt{2/n}} \geq k \middle| \lambda = 2\right) = P(Z > k) \Rightarrow k = Z_{(1-\alpha)},$$

Sendo $Z_{(1-\alpha)}$ o quantil $1-\alpha$ da dsitribuição $Z \sim N(0,1)$. Assim sendo:

$$RC = \left\{\mathbf{x}; \frac{\hat{\lambda}-2}{\sqrt{2/n}} \geq Z_{(1-\alpha)}\right\} \Rightarrow$$
$$RC = \left\{\mathbf{x} : \hat{\lambda} \geq (\sqrt{2/n}) \cdot Z_{(1-\alpha)} - 2\right\}$$

Por fim, temos uma $RC$ para um teste de hipóteses adequado.

**item b)**

Aqui, objetiva-se analisar o comportamento do erro do tipo I (probabilidade de rejeitar-se a hipótese nula dado que ela é verdadeira) por meio de simulações de Monte Carlo. Para tal, utilizou-se o teste obtido no item anterior considerando-se um nível de significância de $\alpha = 5\%$. Assim, a região crítica fica com a seguinte formulação:

$$RC = \left\{\mathbf{x} : \hat{\lambda} \geq (\sqrt{2/n}) \cdot Z_{0,95} - 2\right\}$$

sendo $Z_{0,95}$ o quantil 95% da distribuição normal padrão.

Tendo-se a $RC$ em mãos, podemos realizar simulações de Monte Carlo, com diferentes amostras geradas de uma distribuição $X \sim Poisson(2)$ e verificar como a taxa de erro tipo I (quociente entre o número de vezes que a amostra resulta em uma rejeição da hipótese nula e o número de simulações) se comporta. A abordagem do problema foi feita da seguinte maneira:

Para cada tamanho de amostra de interesse $n = \{10, 25, 50, 100, 200\}$:

- Gerou-se 10000 amostras aleatórias de uma distribuição $X \sim Poisson(2)$ para cada tamanho $n$;
- Verificou-se para cada amostra se houve rejeição da hipótese nula, obtendo-se o número $n_{rej}$ de ocorrências de rejeição;
- Obteve-se a taxa de erro tipo I para cada tamanho de amostra a partir de:

$$taxa_I = n_{rej}/10000$$

Para tal procedimento, cirou-se a função $rc$, a qual recebe uma amostra aleatória e retorna 1 caso ocorra rejeição da hipótese nula, e 0 caso contrário:

```
rm(list=ls(all=TRUE))


rc <- function(aa){ # recebe amostra aleatória (vetor)
  n <- length(aa)

  # Aplicação dda condição de rejeição da RC obtida:
  rej <- ifelse( mean(aa) >= (sqrt(2/n)*qnorm(0.95)+2), 1, 0)
  return(rej)
}
```

Com tal função em mãos, podemos realziar a simulação:

```r
n_aa <- c(10,25,50,100,200) # tamanhos das amostras de interesse
n_sim <- 10000 # número de simulações

set.seed(2112)

for(i in n_aa){ # laço para cada tamanho de amostra
  taxa_I <- 0 # inicialização do valor da taxa de erro do tipo I

  for(j in 1:n_sim){ #laço para o número de simulações
    aa <- rpois(i, lambda = 2) # gera amostra aleatória de uma poisson(2)
    taxa_I <- taxa_I + rc(aa) # soma ao número de ocorrências de rejeição
  }
  taxa_I <- round(taxa_I/n_sim,10) #obtenção da taxa
  cat("\nTaxa de erro tipo I, aa de tamanho",i,":",taxa_I)
}
```

```
##
## Taxa de erro tipo I, aa de tamanho 10 : 0.0541
## Taxa de erro tipo I, aa de tamanho 25 : 0.0601
## Taxa de erro tipo I, aa de tamanho 50 : 0.0489
## Taxa de erro tipo I, aa de tamanho 100 : 0.0533
## Taxa de erro tipo I, aa de tamanho 200 : 0.0545
```

É notório que não há um padrão perceptível para o comportamento da taxa do erro do tipo I em relação ao tamanho da amostra analisada quando o número de simulações é elevado. A taxa para todas essas amostras ficou próxima do valor esperado $\alpha = 0.05$, que corresponde ao nível de significância do teste. Isso mostra que mesmo para amostras pequenas, quando performam-se várias simulações e analisa-se a taxa do erro, ela se comporta como o previsto pelo teste.

**item c)**

Neste item, quer-se analisar o comportamento do poder do teste obtido no item b) para tamanhos diferentes da amostra por meio de simulações de Monte Carlo. Novamente, estaremos analisando o poder de um ponto de vista frequentista, a partir da taxa do erro do tipo II, denotado por $\beta$. O poder de um teste pode ser obtido por:

$$Poder = 1 - \beta$$

O erro do tipo II é definido como a probabilidade de que $H_0$ não seja rejeitada dada que ela é falsa, ou seja, dado que a hipótese alternativa é verdadeira. Pode-se obter esta taxa da mesma forma que obteve-se a taxa para o erro do tipo I. Para tal, gerou-se 1000 amostras para cada tamanho de amostra $n$ e cada valor de $\lambda$ de interesse. A partir da taxa de erro do tipo II obtida, é possível calcular-se o poder para cada $n$ e cada $\lambda$ analisado. O procedimento de análise utilizado foi o seguinte:

```r
rm(list=ls(all=TRUE))

# modificou-se a função rc para que ela retornasse 1 quando não ocorresse rejeição

rc <- function(aa){
  n <- length(aa)
  rej <- ifelse( mean(aa) >= (sqrt(2/n)*qnorm(0.95)+2), 0, 1)
```

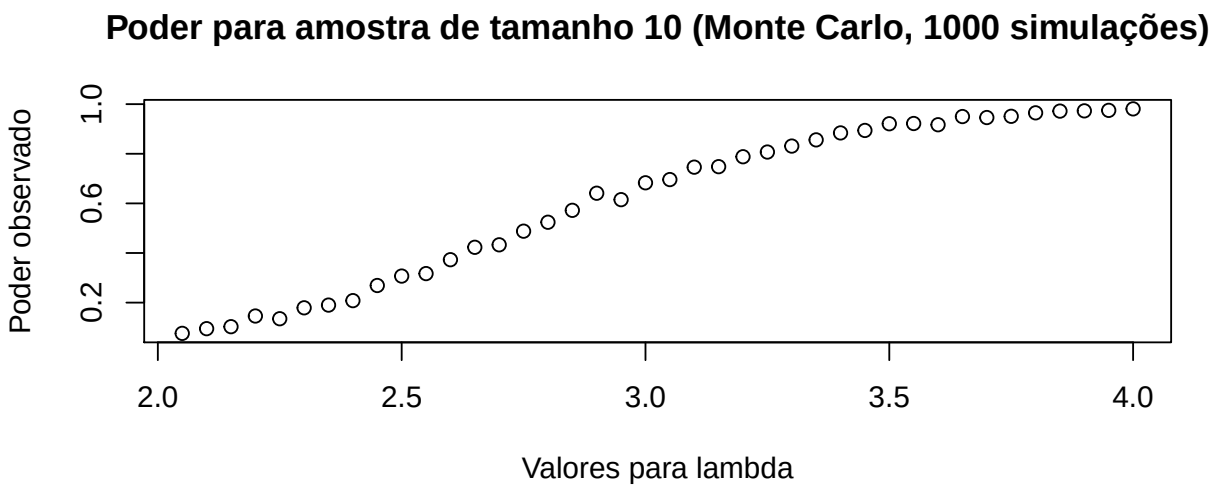
```
  return(rej)
}


n_aa <- c(10,25,50,100,200) # tamanhos das amostras de interesse
n_sim <- 1000 # número de simulações

lambdas <- seq(2.05,4, by = 0.05) # lambdas pertencentes ao subespaço
#paramétrico da hipótese alternativa que queremos analisar


df <- data.frame() # dataframe para conter os valores obtidos

set.seed(2112)

for(i in n_aa){ # laço para o tamanho das amostras
  for(j in lambdas){ #laço para os valores de lambda
    taxa_II <- 0 # inicialização do valor da taxaII
    for(k in 1:n_sim){ # laço para o número de simulações para um mesmo tamanho
      #de amostra e um mesmo lambda de interesse
      aa <- rpois(i, lambda = j) #amostra de tamanho i de uma poisson(j)
      taxa_II <- taxa_II + rc(aa) #acréssimo da ocorrência de não rejeição da
      #hipótese nula
    }
    taxa_II <- taxa_II/n_sim # obtenção da taxa para determinado tamanho de
    #amostra e determinado lambda
    poder <- (1-taxa_II) # obtenção do poder

    obs <- c(i,j,poder) # observação para o dataframe (tamanho da amostra, valor
    #do parâmetro lambda, poder)

    df <- rbind(df,obs) # inclusão da observação no dataframe
  }
}

names(df) <- c("tam","lambda", "poder") # alteração dos nomes das colunas do dataframe

head(df)
```

```
##   tam lambda poder
## 1  10   2.05 0.076
## 2  10   2.10 0.095
## 3  10   2.15 0.103
## 4  10   2.20 0.146
## 5  10   2.25 0.135
## 6  10   2.30 0.179
```

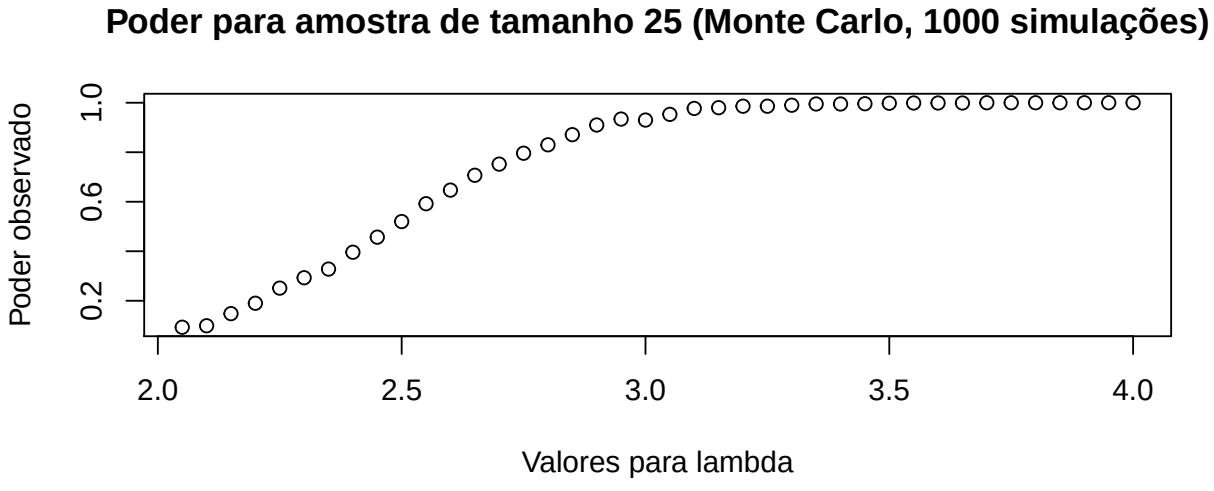
A partir do dataframe gerado, pode-se analisar o comportamento do poder para simulações feitas com tamanhos de amostras diferentes. Vejamos os gráficos:

```
plot(x = df[df$tam==10,]$lambda, y = df[df$tam==10,]$poder,
     main = "Poder para amostra de tamanho 10 (Monte Carlo, 1000 simulações)",
```

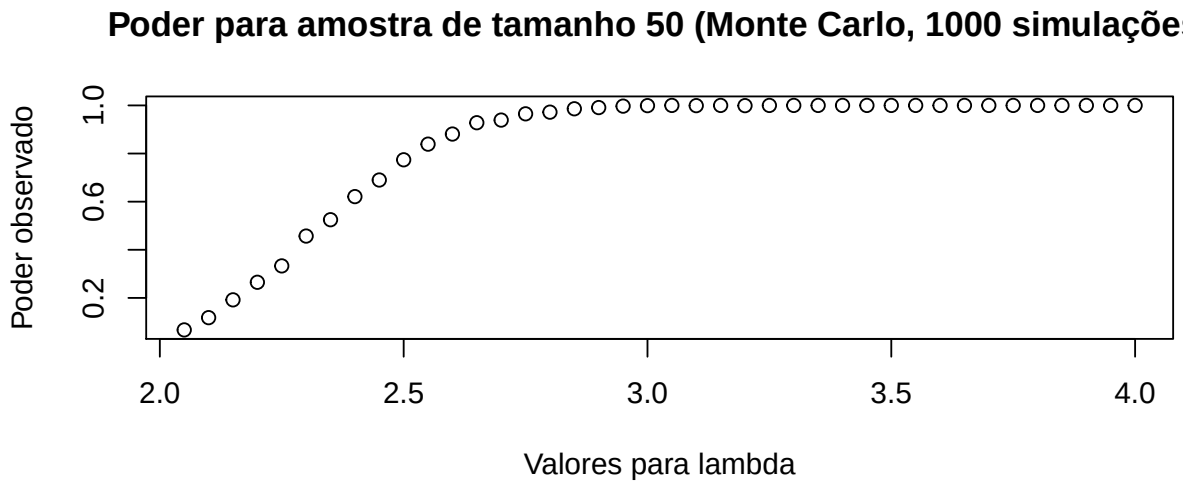
```
     xlab = "Valores para lambda",
     ylab = "Poder observado")
```



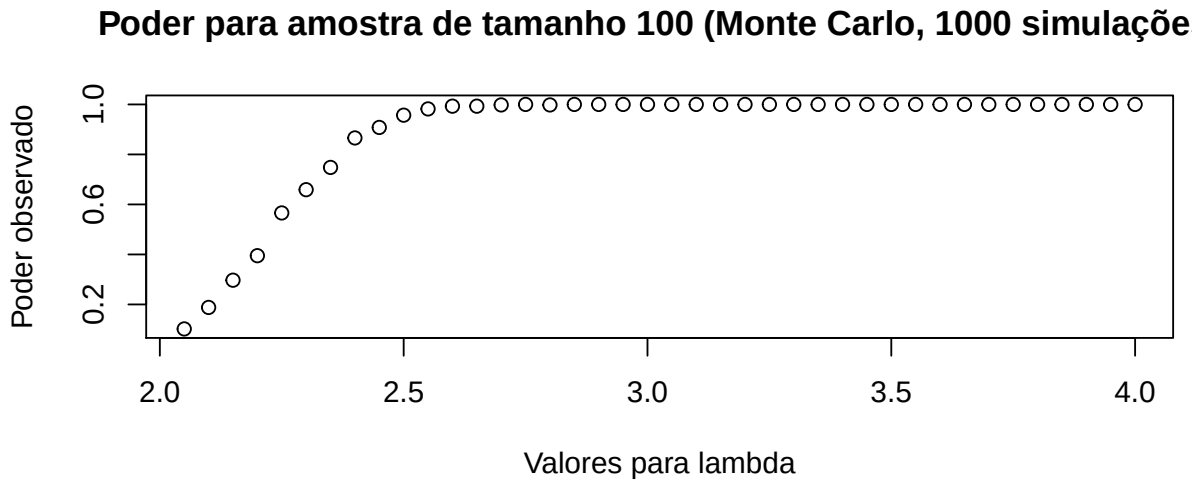

```
plot(x = df[df$tam==25,]$lambda, y = df[df$tam==25,]$poder,
     main = "Poder para amostra de tamanho 25 (Monte Carlo, 1000 simulações)",
     xlab = "Valores para lambda",
     ylab = "Poder observado")
```



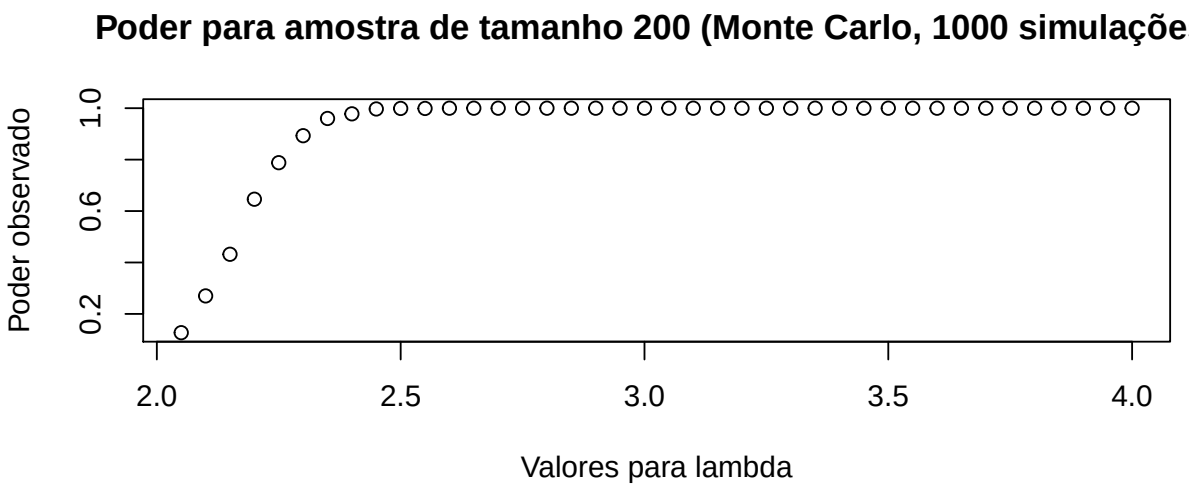

```
plot(x = df[df$tam==50,]$lambda, y = df[df$tam==50,]$poder,
     main = "Poder para amostra de tamanho 50 (Monte Carlo, 1000 simulações)",
     xlab = "Valores para lambda",
     ylab = "Poder observado")
```




```
plot(x = df[df$tam==100,]$lambda, y = df[df$tam==100,]$poder,
     main = "Poder para amostra de tamanho 100 (Monte Carlo, 1000 simulações)",
     xlab = "Valores para lambda",
     ylab = "Poder observado")
```




```
plot(x = df[df$tam==200,]$lambda, y = df[df$tam==200,]$poder,
     main = "Poder para amostra de tamanho 200 (Monte Carlo, 1000 simulações)",
     xlab = "Valores para lambda",
     ylab = "Poder observado")
```




Observando os gráficos obtidos, é notório que para amostras com valor de $n$ mais elevado, a função poder assume valores maiores para os mesmos valores de lambda. Ou seja, se a variável $X$ amostrada seguir uma distribuição $X \sim Poisson(2,5)$, por exemplo, a função poder irá assumir um valor maior quando utilizamos amostras de tamanho maior, e por conseguinte, a probabilidade de rejeitar-se a hipótse nula dada que ela é falsa, aumenta com o aumento do tamanho da amostra. Isso faz com que a função poder atinja valores próximos de 1 para valores de $\lambda$ bem menos extremos em relação à hipótese nula ($\lambda = 2$) para amostras de tamanho 200 do que para amostras de tamanho 10. Tal resultado obtido de forma simulacional concorda com o esperado caso fosse oferecida uma solução analítica para o problema em questão.