Atividade 1 - 200053507

Questão 1

Simulando computacionalmente o gerador de Babel.

Todo seu destino, a cura do câncer e até o que vai acontecer no fim do mundo. Todas essas respostas já estão escritas na Biblioteca de Babel. Essa biblioteca proposta por Jorge Luís Borges é composta por um número infinito de galerias, contendo todos os livros possíveis.

" [...] Um (livro) constava das letras M C V malevolamente repetidas da primeira linha até a última. Outro é um simples labirinto de letras mas a penúltima página diz 'ó tempo tuas pirâmides'."

A maior parte dos livros não tem qualquer significado. Entretanto, embora improváveis, certos textos resultam em grandes obras, como o Bhagavad Gita. Considerando as afirmações acima e a lista de palavras existentes na língua portuguesa (disponível no arquivo "Dicionario.txt"), responda aos itens a seguir. Não faça distinção entre letras maiúsculas e minúsculas.

a) Estime via simulação computacional (*Monte Carlo*) a probabilidade de se gerar uma palavra *válida* (isso é, do dicionário) ao sortear ao acaso sequências de 5 letras (todas com a mesma probabilidade). Em seguida, calcule analiticamente tal probabilidade e faça um gráfico indicando se a estimativa obtida se aproxima do valor teórico conforme a amostra aumenta. **Atenção**: utilize somente as letras do alfabeto sem carateres especiais.

```
pacman::p_load(tidyverse)#carregando a(s) biblioteca(s)
set.seed(123)#escolhando a semente
n<-1000000#número de simulações
dicionario <- read_table(file = "Dicionario.txt", col_names = "Palavra")#Lendo o arquivo
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   Palavra = col_character()
## )
```

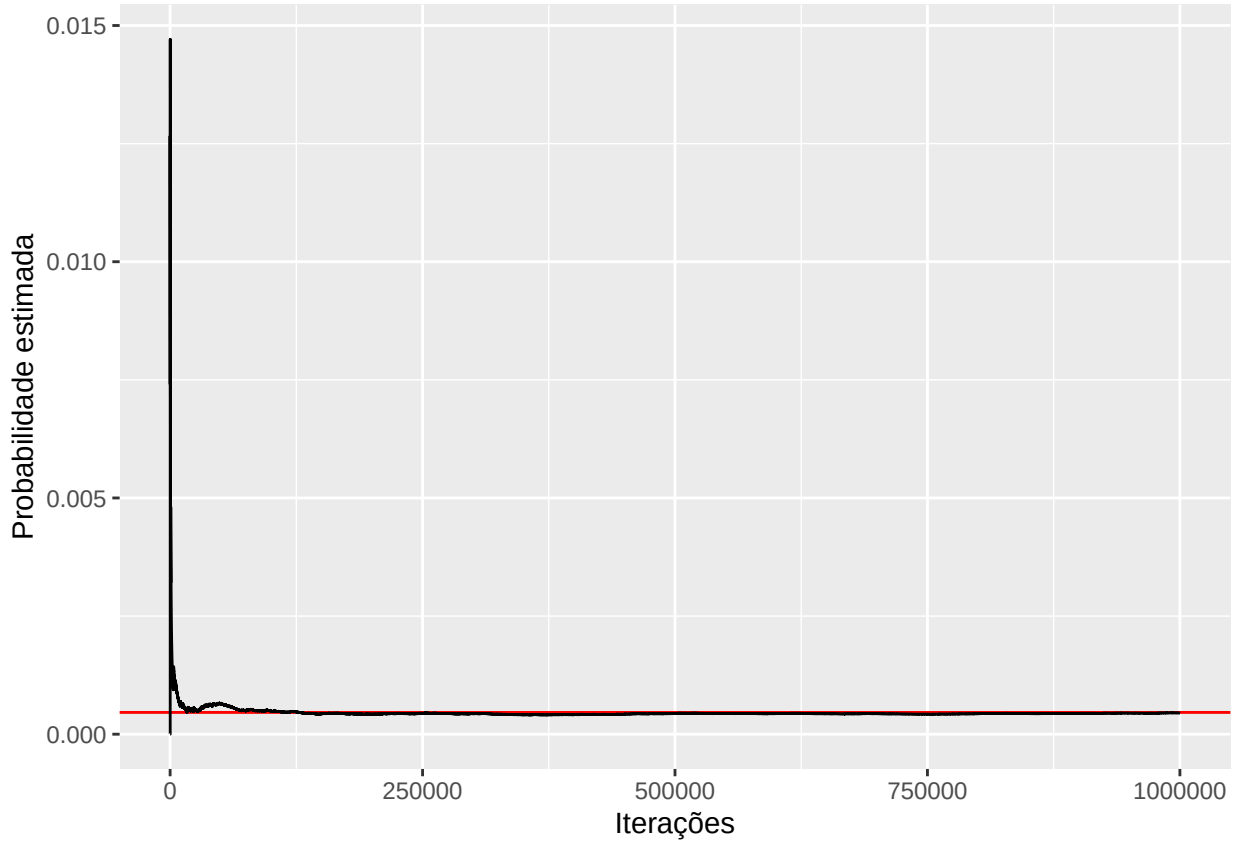
```
letra_5<-dicionario%>%
  filter(str_count(Palavra) == 5) %>%
  pull(Palavra)#filtrando as palavras com 5 letras

gerador <-function(n_letras) {
  sample(letters, n_letras, replace = TRUE) %>%
    str_c(collapse = "")}#Função que gera

palavras_geradas <-tibble(palavra = map_chr(rep(5, n), gerador)) %>%
  mutate(valida = palavra %in% letra_5) %>%
  mutate(probabilidade = cummean(valida))#tibble criado com um laço para repetir n vezes a funcao gerad

(prob_exata <-length(letra_5)/(26**5))#probabilidade exata
```

```
## [1] 0.0004613102
```

```
(prob_aprox <-last(palavras_geradas$probabilidade))#probabilidade simulada
```

```
## [1] 0.000449
```

```
ggplot(palavras_geradas, aes(x =1:n, y=probabilidade)) +
  geom_hline(yintercept = prob_exata, color = "red") + #prob exata
  geom_line() +
  labs(x = "Iterações", y = "Probabilidade estimada")
```



b) Estime a probabilidade da sequência gerada ser um palíndromo (ou seja, pode ser lida, indiferentemente, da esquerda para direita ou da direita para esquerda). Compare o resultado com a probabilidade exata, calculada analiticamente.

```
strReverse <- function(x)
  sapply(lapply(strsplit(x, NULL), rev), paste, collapse="")
# funcao criada, pois a funcao do r não funciounou de maneira correta

(mean(strReverse(palavras_geradas$palavra) == palavras_geradas$palavra)) #prob_aproximada
```

```
## [1] 0.001506
```

```
(26**3 / 26**5)#prob exata
```

```
## [1] 0.00147929
```

c) Construa um gerador que alterne entre consoantes e vogais (se uma letra for uma vogal, a próxima será uma consoante e vice-versa). Qual a probabilidade de gerar uma palavra válida com este novo gerador?

```
gerador_vogal <-function(){
  inicial <-sample(letters, 1)
  vogais <-c("a", "e", "i", "o", "u")
  consoantes <-setdiff(letters, vogais)
  proximas_vogais <-sample(vogais, 2, replace = TRUE)
  proximas_consoantes<-sample(consoantes, 2, replace = TRUE)
  if(inicial %in% consoantes){
    sequencia <-paste0(inicial,
                       proximas_vogais[1],
                       proximas_consoantes[1],
                       proximas_vogais[2],
                       proximas_consoantes[2])
  }else{
    sequencia <-paste0(inicial,
                       proximas_consoantes[1],
                       proximas_vogais[1],
                       proximas_consoantes[2],
                       proximas_vogais[2])
  }

  sequencia #sequencia criada começando por consoante/vogal
}

palavras <- replicate(n, gerador_vogal())#replicando a função n vezes
palavras[palavras %in% dicionario$Palavra]%>%
  length() / n #probabilidade aumenta
```

```
## [1] 0.006757
```

d) Considere um processo gerador de sequências de 5 caracteres no qual cada letra é sorteada com probabilidade proporcional à sua respectiva frequência na língua portuguesa (veja essa página). Suponha que esse processo gerou uma sequência com ao menos um "a". Neste caso, estime a probabilidade dessa sequência ser uma palavra válida. **Dica**: Use a função `sample` e edite o parâmetro `prob`. **Para pensar**: Você consegue calcular essa probabilidade analiticamente? (Não precisa responder.)

```
freq<-c(14.63, 1.04,3.88, 4.99, 12.57,1.02, 1.30, 1.28, 6.18, 0.40,
        0.02,2.78, 4.74, 5.05, 10.73, 2.52, 1.20, 6.53,7.81, 4.34,
        4.63, 1.67, 0.01, 0.21, 0.01,0.47)#frequencias coletadas manualmente

gerador_d <-function() {
  sequencia <-sample(letters, 5, prob = freq)
  while(!"a" %in% sequencia) {
    sequencia <-sample(letters, 5, prob = freq)}
  str_c(sequencia, collapse = "")}
tibble(palavra =replicate(n, gerador_d())) %>%
  summarise(probabilidade =mean(palavra %in% dicionario$Palavra))
```

```
## # A tibble: 1 x 1
##   probabilidade
##           <dbl>
## 1        0.0104
```

Questão 2

Gerando números pseudo-aleatórios.

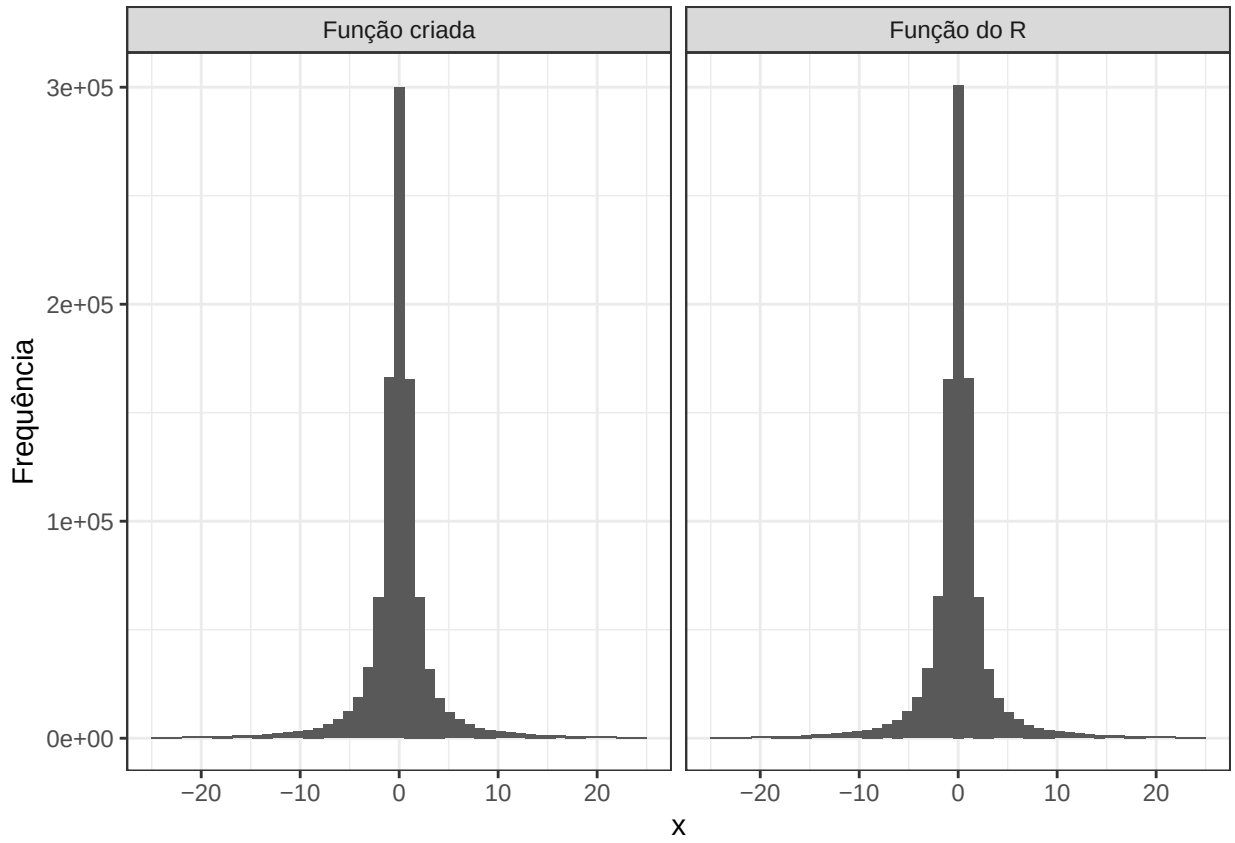
a) Escreva uma função que gere, a partir do método da transformada integral, uma amostra aleatória de tamanho n da distribuição Cauchy para n e γ arbitrários. A densidade da Cauchy(γ) é dada por

$$f(x) = \frac{1}{\pi\gamma(1 + (x/\gamma)^2)}.$$

Dica: Veja essa página.

```
n_c<-runif(n)#gerando números uniformes
inverse_cauchy<-function(u, sigma, mu) {
  sigma * tan(pi * (u - (1/2))) + mu# densidade de cauchy
}

sigma <-1
mu <-0

xsim<-map_dbl(n_c, inverse_cauchy, sigma = sigma, mu = mu)#laço para repetir

rcauchy_vals <-rcauchy(n, location = 0, scale = 1)#funcao do R para comparar

tibble(`Função criada` =xsim, "Função do R" = rcauchy_vals) %>%
  pivot_longer(everything(), names_to = "method") %>%
  ggplot(aes(value)) +
  geom_histogram(bins = 50) +
  scale_x_continuous(limits = c(-25, 25)) +
  facet_wrap(~ method) +
  theme_bw() +
  labs(x = "x", y = "Frequência")#teste de comparação
```

```
## Warning: Removed 50723 rows containing non-finite values (‘stat_bin()‘).
```



b) Uma variável aleatória discreta X tem função massa de probabilidade

$$\begin{aligned} p(2) &= 0.3 \\ p(3) &= 0.1 \\ p(5) &= 0.1 \\ p(7) &= 0.2 \\ p(9) &= 0.3 \end{aligned}$$

Use o método de transformação inversa para gerar uma amostra aleatória de tamanho 1000 a partir da distribuição de X. Construa uma tabela de frequência relativa e compare as probabilidades empíricas com as teóricas. Repita usando a função *sample* do R.

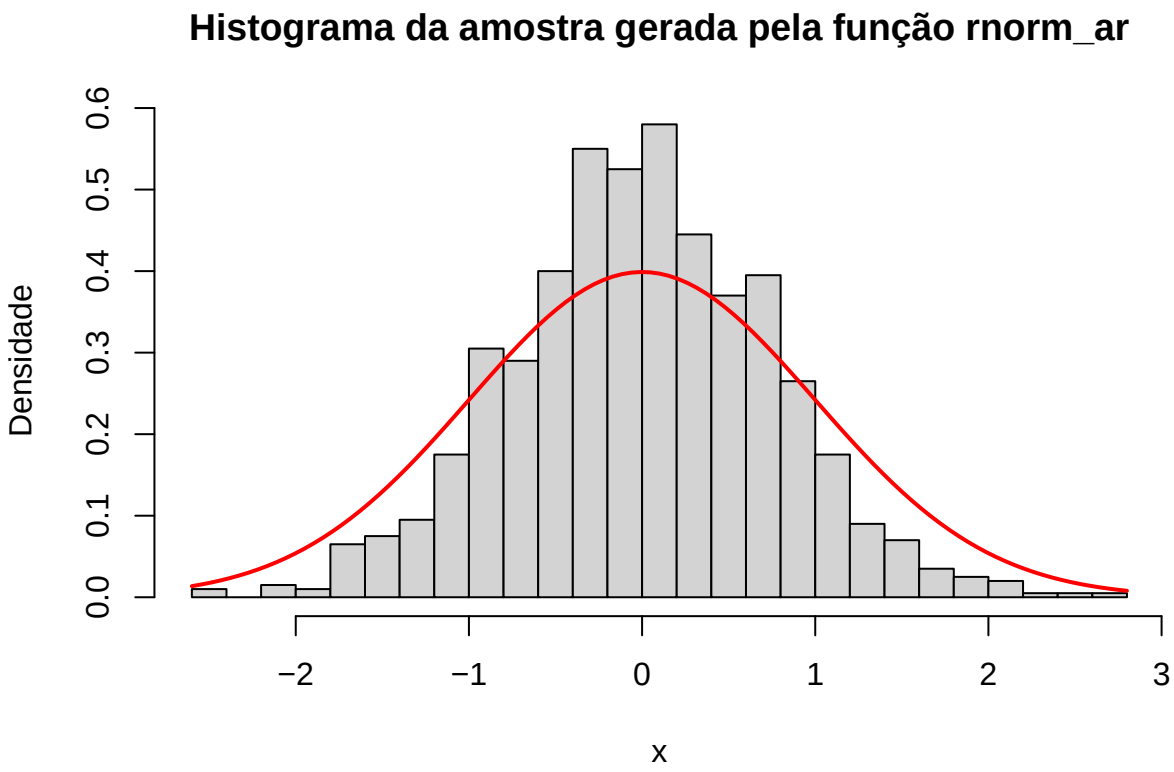
```
u <-runif(1000) # gerar valores aleatórios uniformes entre 0 e 1
x <-numeric(1000) # vetor para armazenar a amostra gerada

for (i in 1:1000) {
  if (u[i] < 0.3) {
    x[i] <-2
  } else if (u[i] < 0.4) {
    x[i] <-3
  } else if (u[i] < 0.5) {
    x[i] <-5
  } else if (u[i] < 0.7) {
    x[i] <-7
```

```
  } else {
    x[i] <-9
  }}


p_empiricas <-table(x) / length(x)
p_teoricas <-c(0.3, 0.1, 0.1, 0.2, 0.3)#p exata
cbind(p_empiricas, p_teoricas)#comparação das duas probabilidades
```

```
##   p_empiricas p_teoricas
## 2       0.290        0.3
## 3       0.096        0.1
## 5       0.109        0.1
## 7       0.216        0.2
## 9       0.289        0.3
```

c) Escreva uma função que gere amostras da distribuição Normal padrão ($\mu = 0, \sigma = 1$) usando o método de aceitação e rejeição adotando como função geradora de candidatos, $g(x)$, a distribuição Cauchy padrão (isso é, com $\gamma = 1$).

```
rnorm_ar <-function(n) {
  x <- numeric(n)
  i <- 1
  while(i <= n) {
    y <-rcauchy(1)#usando a função do R
    c <-sqrt(2 * exp(1) / pi)
    if(runif(1) < c * exp(-(y^2)/2)) {
      x[i] <- y
      i <- i + 1
    }
  }
  return(x)
} # função criada para gerar os números

x <-rnorm_ar(1000)#gerando 1000 numeros

hist(x, breaks = 30, freq = FALSE, main = "Histograma da amostra gerada pela função rnorm_ar",
     xlab = "x", ylab = "Densidade")#Histograma para mostrar o resultado
curve(dnorm(x), add = TRUE, col = "red", lwd = 2)
```


Histograma da amostra gerada pela função rnorm_ar
Densidade
0.0
0.1
0.2
0.3
0.4
0.5
0.6
−2
−1
0
1
2
3
x