Lista 1 Estatistica Computacional

Davi Wentrick Feijó - 200016806

2023-05-03

Questão 1

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.

```
##### Questao 1A ----
dicionario_ptbr <- as.data.frame(read_csv("Dicionario.txt"))
colnames(dicionario_ptbr) = c("palavras")

dicionario_ptbr = dicionario_ptbr %>%
    mutate(tamanho = nchar(palavras)) %>%
    filter(tamanho == 5)
tolower(dicionario_ptbr$palavras)
```

```
# criando uma funcao para gerar as palavras e comparar elas
criar_palavras_aleatorias <- function(lista_palavras, n_amostras) {
    # Cria um vetor com todas as letras do alfabeto
    alfabeto <- letters

    # Inicializa um vetor para armazenar o número de correspondências
    # em cada amostra
    correspondencias <- numeric(n_amostras)
    amostra = c()
    for (i in seq_len(n_amostras)) {
        # Gera uma amostra aleatória de 5 letras do alfabeto
        amostra[i] <- paste(sample(alfabeto, 5, replace = TRUE), collapse = "")
    }

    # Verifica se a palavra aleatória gerada está na lista de
    # palavras
    correspondencias <- lista_palavras %in% amostra
    # Retorna o número total de correspondências encontradas em todas
    # as amostras
    return(sum(correspondencias))
}


# Cria uma lista de palavras de exemplo
```

```r
palavras <- dicionario_ptbr$palavras

start = Sys.time()
# Chama a função para gerar 100 amostras aleatórias e verificar
# quantas correspondem à lista de palavras
correspondencias_total <- criar_palavras_aleatorias(palavras, 300000)
finish = Sys.time()
```

```
## tempo = 2.79645681381
```

```
## Total de correspondências encontradas: 123 a probabilidade é de: 0.00041
```

Podemos rodar essa simulacao varias vezes e verificar o resultado obtido!

```r
# paralelizando o loop da funcao Especificando o número de núcleos a
# serem usados
n_cores <- 23
cl <- makeCluster(n_cores)
x = 100000  #numero de palvras geradas
y = 100  #numero de loops dessa simulacao
# Registrando os núcleos para uso do foreach
registerDoParallel(cl)

start = Sys.time()
# Loop externo paralelizado
matches <- foreach(n = 1:y, .combine = "mean") %dopar% {
    criar_palavras_aleatorias(palavras, x)
}

finish = Sys.time()
```

```
## tempo = 11.7124249935
```

```r
# Fechando a conexão com os núcleos
stopCluster(cl)
```

```
## A probabilidade estimada a partir de uma media da quantiade de palvras validas encontradas de 100
##  simulacoes gerando 100000 palvras aleatorias cada é: 0.00059
```

```
## A probabilidade exata de se gerar uma palvra aleatoria de 5 letras e ela existir é:
##  0.000461310205148
```

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.

```
criar_palavras_aleatorias_palindromas <- function(n_amostras) {
    # Cria um vetor com todas as letras do alfabeto
    alfabeto <- letters

    # Inicializa um vetor para armazenar o número de palavras
    # palíndromas geradas em cada amostra
    palindromos <- numeric(n_amostras)

    for (i in seq_len(n_amostras)) {
        # Gera uma amostra aleatória de 5 letras do alfabeto
        palavra_quebrada <- sample(alfabeto, 5, replace = TRUE)
        palavra = paste(palavra_quebrada, collapse = "")
        palavra_reversa = paste(rev(palavra_quebrada), collapse = "")

        # Verifica se a palavra gerada é um palíndromo
        if (palavra == palavra_reversa) {
            palindromos[i] <- 1
        } else {
            palindromos[i] <- 0
        }
    }

    # Retorna a proporção de palavras palíndromas geradas em todas as
    # amostras
    return(mean(palindromos))
}

# Chama a função para gerar 100000 amostras aleatórias e estimar a
# probabilidade de gerar uma palavra palíndroma
prob_palindromo <- criar_palavras_aleatorias_palindromas(100000)
```

```
cat("Probabilidade de gerar uma palavra palíndroma:", prob_palindromo)
```

```
## Probabilidade de gerar uma palavra palíndroma: 0.00144
```

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?

```
criar_palavras_aleatorias_vogal_consoante <- function(lista_palavras, n_amostras) {
    # Cria um vetor com todas as letras do alfabeto
    vogais <- c("a", "e", "i", "o", "u")
    consoantes <- c("b", "c", "d", "f", "g", "h", "j", "k", "l", "m", "n",
        "p", "q", "r", "s", "t", "v", "w", "x", "y", "z")

    # Inicializa um vetor para armazenar o número de correspondências
    # em cada amostra
    correspondencias <- numeric(n_amostras)
    amostra = c()
    for (i in seq_len(n_amostras)) {
        # Gera uma amostra aleatória de 5 letras do alfabeto
        palavra <- character(5)  # inicializa um vetor vazio com 10 espaços
        for (n in 1:5) {
            if (n%%2 == 0) {
                # se o índice for par, sorteia uma consoante
                letra <- sample(consoantes, 1)
            } else {
                # senão, sorteia uma vogal
                letra <- sample(vogais, 1)
            }
            palavra[n] <- letra  # adiciona a letra à palavra
        }
        amostra[i] = paste(palavra, collapse = "")
    }

    # Verifica se a palavra aleatória gerada está na lista de
    # palavras
    correspondencias <- lista_palavras %in% amostra
    # Retorna o número total de correspondências encontradas em todas
    # as amostras
    return(sum(correspondencias))
}

# Cria uma lista de palavras de exemplo
palavras <- dicionario_ptbr$palavras

start = Sys.time()
# Chama a função para gerar 100 amostras aleatórias e verificar
# quantas correspondem à lista de palavras
correspondencias_total <- criar_palavras_aleatorias_vogal_consoante(palavras,
    300000)
finish = Sys.time()
```

```
## tempo = 9.58642888069
```

```
## Total de correspondências encontradas: 504 a probabilidade estimada é: 0.00168
```

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.)

```
criar_palavras_aleatorias <- function(lista_palavras, n_amostras) {
    # Cria um vetor com todas as letras do alfabeto
    alfabeto <- letters
    probabilidade_letras = c(0.1463, 0.0104, 0.0388, 0.0499, 0.1257, 0.0102,
        0.013, 0.0128, 0.0618, 0.004, 0.0002, 0.0278, 0.0474, 0.0505, 0.1073,
        0.0252, 0.012, 0.0653, 0.0781, 0.0434, 0.0463, 0.0167, 0.0001,
        0.0021, 0.0001, 0.0047)
    # Inicializa um vetor para armazenar o número de correspondências
    # em cada amostra
    correspondencias <- numeric(n_amostras)
    amostra = c()
    for (i in seq_len(n_amostras)) {
        # Gera uma amostra aleatória de 5 letras do alfabeto
        amostra[i] <- paste(sample(alfabeto, 5, replace = TRUE, prob = probabilidade_letras),
            collapse = "")
    }

    # Verifica se a palavra aleatória gerada está na lista de
    # palavras
    correspondencias <- lista_palavras %in% amostra
    # Retorna o número total de correspondências encontradas em todas
    # as amostras
    return(sum(correspondencias))
}


# Cria uma lista de palavras de exemplo
palavras <- dicionario_ptbr$palavras

start = Sys.time()
# Chama a função para gerar 100 amostras aleatórias e verificar
# quantas correspondem à lista de palavras
correspondencias_total <- criar_palavras_aleatorias(palavras, 300000)
finish = Sys.time()
```

```
## tempo = 2.54084610939
```

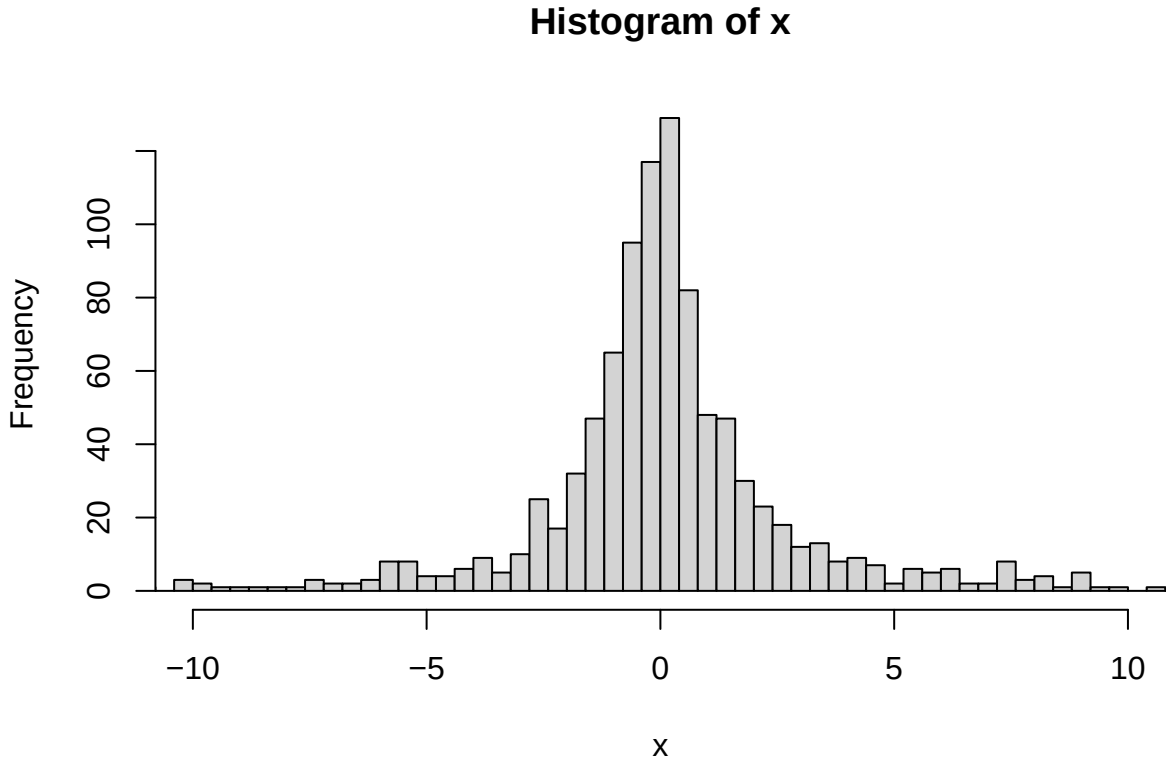
```
## Probabilidade estimada: 0.00494333333333
```

Questão 2

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

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

```r
cauchy_sample <- function(n, gamma, x0 = 0) {
    # gerar n variáveis aleatórias uniformes no intervalo [0,1]
    u <- runif(n)
    # calcular as variáveis aleatórias da distribuição Cauchy usando
    # a transformada integral inversa
    x <- x0 + gamma * tan(pi * (u - 1/2))
    return(x)
}


x <- cauchy_sample(1000, 1, 0)
hist(x, xlim = c(-10, 10), breaks = seq(-1000, 1000, 0.4))
```



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

$$p(2) = 0.2$$
$$p(3) = 0.1$$
$$p(5) = 0.2$$
$$p(7) = 0.2$$
$$p(9) = 0.3$$

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.

```
# função de distribuição acumulada inversa (ICDF)
icdf <- function(u) {
    ifelse(u <= 0.2, 2, ifelse(u <= 0.3, 3, ifelse(u <= 0.5, 5, ifelse(u <=
        0.7, 7, 9))))
}

# Gerar 1000 amostras aleatórias
u <- runif(1000)

# Aplicar a ICDF para obter as amostras aleatórias
x <- icdf(u)


tabela_ICDF <- table(x)/1000
tabela_teorica <- c(0.2, 0.1, 0.2, 0.2, 0.3)
names(tabela_ICDF) <- c(2, 3, 5, 7, 9)
names(tabela_teorica) <- c(2, 3, 5, 7, 9)

# comparando com a funcao sample do R
X2 <- sample(c(2, 3, 5, 7, 9), 1000, prob = c(0.2, 0.1, 0.2, 0.2, 0.3),
    replace = TRUE)

tabela_sample <- table(X2)/1000

rbind(tabela_teorica, tabela_ICDF, tabela_sample)
```

```
##                   2     3     5     7     9
## tabela_teorica 0.200 0.100 0.200 0.200 0.300
## tabela_ICDF    0.215 0.092 0.194 0.197 0.302
## tabela_sample  0.199 0.084 0.218 0.194 0.305
```

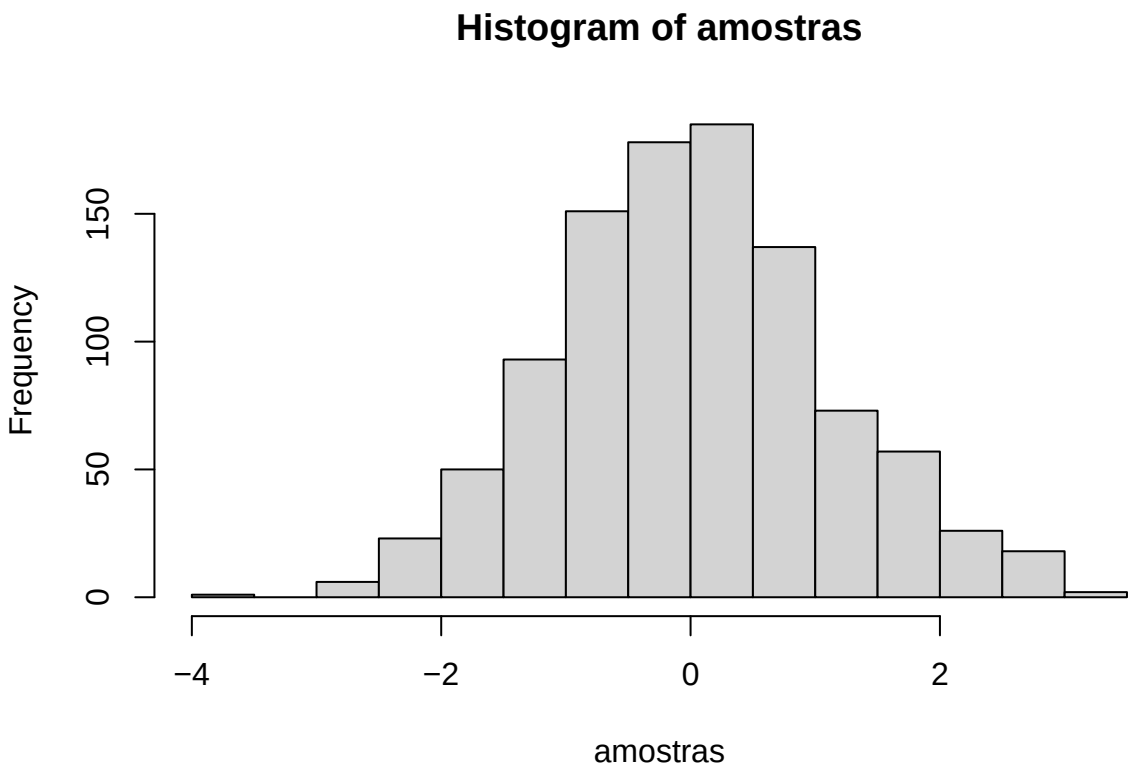
c) Escreva uma função que gere amostras da distribuição Normal padrão ($\mu = 0$, $\sigma = 1$) usando o método de aceitação e rejeição adotando como função geradora de candidatos, $g(x)$, a distribuição Cauchy padrão (isso é, com $\gamma = 1$).

A constante c é definida como a razão entre a constante de normalização da distribuição Normal e a função densidade de probabilidade da distribuição Cauchy no ponto de máximo. Assim, a constante c pode ser calculada como:

$$c = C(f)/g(0) = sqrt(2/e)$$

Onde $C(f) = 1/sqrt(2 * pi)$ é a constante de normalizacao da normal e $g(0) = (1/pi)$ é a função densidade de probabilidade da distribuição Cauchy no ponto de máximo

```
gerar_amostra_normal_aceitacao_rejeicao <- function(n) {
    f <- function(x) dnorm(x)
    g <- function(x) dcauchy(x)
    c <- sqrt(2/exp(1))

    amostras <- numeric(n)
    i <- 1

    while (i <= n) {
        x <- rcauchy(1)
        u <- runif(1)

        if (u <= f(x)/(c * g(x))) {
            amostras[i] <- x
            i <- i + 1
        }
    }

    return(amostras)
}

amostras <- gerar_amostra_normal_aceitacao_rejeicao(1000)

hist(amostras)
```


Histogram of amostras
Frequency
0
50
100
150
−4
−2
0
2
amostras