Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Estatística

Eduardo Moreira Araújo
José Vítor Barreto Porfirio
Stefan Zurman Gonçalves

Processos Estocásticos: Markov Chain Monte Carlo e o uso do algoritmo Metropolis-Hastings aplicado em criptografia

Brasília, DF
04 de Dezembro de 2023

Conteúdo

Resumo

A criptografia é um mecanismo fundamental para proteção de mensagens e senhas. A estatística surge nesse contexto como ferramenta para aumentar a eficácia dos métodos criptográficos ou no caminho inverso, auxiliando na decodificação de mensagens. O objetivo deste trabalho é aplicar o algoritmo Metropolis-Hastings para decodificar textos em língua portuguesa, considerando sua relação teórica com Cadeias de Markov. O software R foi utilizado para implementação do algoritmo, tendo como base um texto em português e a criação de um score que mede a proximidade de textos em relação ao idioma. Os resultados indicam uma convergência eficiente do algoritmo e a capacidade de tal técnica estatística em quebrar criptografias.

Palavras-Chave: Criptografia; Cadeia de Markov; Algoritmo Metropolis-Hastings; Probabilidade.

1 Introdução

Criptografia é uma forma de esconder o conteúdo de mensagens para que somente indivíduos que deveriam conhecer a forma de descriptografar tenham acesso ao seu conteúdo. É extremamente útil para proteger dados sensíveis, evitando que senhas e informações sigilosas caíam nas mãos de pessoas mal intencionadas.

Atualmente possuir bons mecanismos de criptografia é diferencial no armazenamento de senhas e demais credenciais de serviços online, o surgimento dos computadores quânticos no mercado trouxe o tema à tona, devido à capacidade dessa nova tecnologia de quebrar completamente o sistema mais moderno de criptografia, que envolve uma técnica de 3 passos *hashing*, *salting* e *peppering* [7].

Surgir um novo tipo de computador capaz de quebrar os melhores mecanismos de criptografia da atualidade é uma grande semelhança com a origem dos computadores. Durante a Segunda Guerra Mundial, os alemães possuíam um dispositivo de criptografia chamado "Enigma", os demais países não sabiam como descriptografar as mensagens, ainda que pudessem intercepta-las. É sob essas condições que Alan Turing, um cientista britânico que trabalhava para quebrar essa criptografia, criou uma máquina capaz de executar algoritmos apenas com números binários e inventou um algoritmo que descriptografava as mensagens do "Enigma", essa máquina ficou conhecida como o 1º computador e Alan Turing conhecido como o pai da computação [6].

Esse estudo não é tão ambicioso para tentar quebrar as maiores técnicas da criptografia contemporânea, mas foca no trabalho didático da utilização do algoritmo de Metropolis-Haistings para a quebra de criptografias mais simples e lúdicas, aplicadas sobre textos e poemas. Esta é uma das principais técnicas na área de Markov Chain Monte Carlo (MCMC) e se baseia teoricamente e computacionalmente em um processo estocástico conhecido como Cadeias de Markov.

O trabalho está dividido em três grandes partes: No Referencial Teórico explicamos brevemente alguns conceitos e propriedades sobre processos estocásticos, Cadeias de Markov, MCMC e o algoritmo Metropolis-Hastings. Na Metogologia discutimos a concepção geral sobre criptografia, descriptografia e como a implementação do método foi pensada em termos de linguagem. Por fim, na parte de Resultados apresentamos o código em R e a explicação da aplicação do algorimo Metropolis-Hastings.

2 Referencial Teórico

A seguir são apresentadas as teorias e técnicas estatísticas que serão abordadas no presente relatório.

2.1 Processos Estocásticos

Um processo estocástico $\{X(t), t \in T\}$ ou $\{X_t, t \in T\}$ pode ser definido como uma sequência de variáveis aleatórias, onde para cada $t \in T$, $X(t)$ é uma variável aleatória, com espaço de probabilidade $(\Omega, \mathcal{A}, P)$ [3]. A partir dessa definição, podem se dividir os processos estocásticos como em tempo discreto, se T for um conjunto enumerável de elementos, ou a tempo contínuo, se T for um conjunto não-enumerável de elementos. As variáveis são definidas em um espaço comum $\mathcal{S}$, em que se o espaço for enumerável ou seja, as variáveis aleatórias são discretas, diz-se que o processo estocástico possui espaço de estados enumerável, enquanto se o espaço for não-enumerável, ou seja, as variáveis aleatórias são contínuas, diz-se que o processo estocástico possui espaço de estados geral. Assim, um processo estocástico pode ser caracterizado por seu índice t e seu espaço de estados, além da relação de dependência entre suas variáveis aleatórias.

2.2 Cadeias de Markov

Uma cadeia de Markov pode ser definida como um processo estocástico $\{X(t)\}$ com espaço de estado enumerável e tempo discreto tal que a distribuição condicional de qualquer estado futuro $X(t+1)$ dado todos os estados passado $X(0), X(1), \ldots, X(t-1)$ e o estado presente $X(t)$ independe dos estados passados e depende apenas do estado presente. Essa relação pode ser dada por

$$P(X_{t+1} = j | X_0 = x_0, X_1 = x_1, \ldots, X_{t-1} = x_{t-1}, X_t = i) = P(X_{t+1} = j | X_t = i)$$

Uma cadeia de Markov é considerada ergódica se conforme aumentam os índices t, a probabilidade da cadeia estar em qualquer espaço independe do espaço onde ela começou. A distribuição resultante é chamada de distribuição estacionária. Para trabalharmos com essa ideia, precisamos calcular a probabilidade de transição de t passos da cadeia, logo, incorporamos a ideia de matriz de transição:

$$P^{(t)} = [P_{ij}^t]_{ij \in S} \text{ com } \sum_{j=1}^{t} P_{ij} = 1$$

2.3 Markov Chain Monte Carlo (MCMC)

O principal propósito do MCMC é, a partir de uma dada distribuição de probabilidades $\boldsymbol{\pi}$, estimar uma variável aleatória com essa distribuição de probabilidades. O método consiste em criar uma cadeia de Markov ergódica na qual a distribuição estacionária equivale à distribuição de probabilidades $\boldsymbol{\pi}$. O processo consiste então de iterar a cadeia até a sua convergência, e então pegar os últimos elementos como uma amostra da distribuição $\boldsymbol{\pi}$. A partir desse método, é possível estimar qualquer momento de uma variável aleatória X com distribuição de probabilidades $\boldsymbol{\pi}$.

As estimativas são possíveis devido a lei forte dos grandes números para cadeias de Markov, definidos pelo seguinte teorema [1]:

Teorema 2.1 *Assuma que $X_0, X_1, \ldots,$ é uma cadeia de Markov ergódica com distribuição estacionária $\boldsymbol{\pi}$. Seja r uma função limitada. Seja também X uma variável aleatória com distribuição $\boldsymbol{\pi}$. Então, com probabilidade 1,*

$$\lim_{t\to\infty} \frac{r(X_1) + r(X_2) + \cdot + r(X_n)}{n} = E(r(X)),$$

onde $E(r(X)) = \sum_j r(j)\pi_j$

2.4 O algoritmo Metropolis-Hastings

O algoritmo de Metropolis-Hastings é um dos métodos mais utilizados em MCMC. A partir dele, é possível gerar uma sequência $X_0, X_1, \ldots,$ de uma cadeia de Markov ergódica com distribuição estacionária $\boldsymbol{\pi}$, ou seja, é possível gerar uma amostra da distribuição de interesse. O seu processo de amostragem é descrito a seguir:

Seja $\boldsymbol{\pi}$ uma distribuição de probabilidades discreta de uma variável X que queremos amostrar. Seja também $\mathbf{T}$ a matriz de transição de uma cadeia de Markov irredutível com mesmo espaço de estado que $\boldsymbol{\pi}$, na qual é possível tirar amostras. A partir dessa cadeia de Markov, serão gerados elementos candidatos de uma sequência, os quais serão aceitos ou não pelo algoritmo para serem partes da cadeia final X_t. O processo de aceitação de novos elementos se da pelo seguinte procedimento:

1. Amostra-se um candidato j para o próximo estado a partir da matriz de transição $\mathbf{T}$, dado o estado atual i. Essa matriz de transição com as probabilidades da cadeia deve ser irredutível;

2. Calcula-se a função de aceitação $a(i,j) = \frac{\pi_j T_{ji}}{\pi_i T_{ij}}$;

3. Se $a(i,j) \geq 1$, aceitar j como o próximo estado da cadeia, caso contrário, aceitar j como o próximo estado com probabilidade $a(i,j)$ ou i como o próximo estado com probabilidade $1 - a(i,j)$.

A principal importância do método reside em sua capacidade de gerar amostras de distribuições complexas, as quais a geração de amostras de maneiras convencional é difícil ou impossível. O método apresenta a maior vantagem por não exigir a computação direta da função de densidade de probabilidade, permitindo a exploração de distribuições complexas sem depender de cálculos analíticos difíceis ou impossíveis de executar. A partir do método, é possível gerar também amostras de distribuições multivariadas, e o método tem suma importância na inferência Bayesiana, para a amostragem de distribuições a posteriori complexas, para então serem feitas inferências sobre tal distribuição [4].

3 Metodologia

3.1 Criptografia

Suponha uma mensagem qualquer com um número arbitrário de carácteres, sem erros de digitação, então é possível criar uma função bijetiva que mapeia os carácteres originais em novos carácteres, isto é, uma função que leva cada carácter em outro ou nele mesmo, mas que cada letra é mapeada em exatamente 1 novo carácter e esse novo carácter só é obtido por esse mapeamento. Um exemplo é o mapeamento que leva uma letra na próxima seguindo o alfabeto de forma modular

$$\begin{cases} a \to b \\ b \to c \\ \vdots \\ y \to z \\ z \to a \end{cases}$$

Segundo essa criptografia, preservando os espaços em branco, pode-se ter a seguinte mensagem antes e depois de ser criptografada

Antes : Passei em processos com SS

Depois : Qbttfj fn qspdfttpt dpn TT

depois da criptografia a mensagem não pode ser compreendida, o que cumpre o objetivo do remetente de não permitir que pessoas que não possuem uma forma de decriptar a mensagem leiam-na.

3.2 Descriptografia

No exemplo da seção 3.1 nos partimos do ponto de vista em que temos a mensagem original e a função de criptografia. Para decriptar a mensagem bastaria inverter a função de criptografia, a letra “b” passaria a corresponder à letra “a”, “c” a “b” e assim sucessivamente.

Suponha que a seguinte mensagem tenha sido recebida “ariivb vu azslviisi lsu ii". Qual foi a criptografia utilizada e qual era a mensagem original? Caso tentássemos todas as criptografias possíveis, teríamos $26! \approx 10^8$ combinações possíveis para tentar, pode-se mostrar que mesmo em um supercomputador, esse processo demoraria anos para ser concluído [2].

Uma maneira mais eficiente para responder essas perguntas é utilizar o algoritmo de Metropolis-Haistings [1, 2]. Para tanto precisa-se de uma matriz de transição T de uma letra para outra, dessa forma cada par de letras recebe uma pontuação ba-

seada em sua probabilidade de ocorrência, ao tentar uma nova inversão de função de criptografia tem-se uma nova pontuação para a decriptação proposta, caso a pontuação aumente, então aceita-se a mudança, do contrário, soteia-se uma probabilidade uniforme que decide se a nova decriptação é aceita ou não.

A cada passo do algortimo, deve-se escolher dois carácteres e trocar o seu mapeamento um pelo outro, dessa forma, a cada iteração o algoritmo consegue progredir de forma markoviana para solucionar o problema, concluindo a caracterização do algoritmo de Metropolis-Haistings.

É possível formalizar esse raciocínio por meio de uma sequência de passos [1], a sequência descreve todo o processo de aceitação das funções

1. **Calcula-se a pontuação S_0 da frase decriptada atual**
2. **Sorteiam-se duas letras L_1 e L_2 que devem ter suas posições trocadas na função de decriptação**
3. **Calcula-se a pontuação S_1 da frase sob a nova decriptação**
4. **Calcula-se a razão das pontuações $R = \frac{S_1}{S_0}$**
5. $\begin{cases} \textbf{Se } R \geq 1\textbf{, retorne a nova função de decriptação} \\ \textbf{Se } R < 1\textbf{, sorteie } U \begin{cases} \textbf{Se } U \geq R\textbf{, Retorne a nova função de decriptação} \\ \textbf{Se } U < R\textbf{, Retorne a antiga função de decriptação} \end{cases} \end{cases}$

em que $U \sim Unforme(0, 1)$.

3.3 Linguagem

Embora as aplicações que inspiraram este trabalho estejam em inglês e consequentemente as matrizes de transição tenham sido construídas com base na língua inglesa [1, 2], compreende-se relevante construir bases que permitam a aplicação das técnicas de tradução em português.

Para atingir esse objetivo, a primeira proposta era utilizar as palavras de um dicionário online para compor a matriz de transição, entretanto essa abordagem peca na transição de espaços para letras e na quantidade de palavras. Na prática, essa abordagem se mostrou mal sucedida.

Para corrigir os problemas da abordagem do dicionário, foi feito o processamento automatizado dos livros “Cosmos”, “Contato” e “O Mundo Assombrado pelos Demônios” de Carl Sagan. Dessa forma o algoritmo varreu uma quantidade muito maior de palavras e também contabilizou as transições dos espaços para letra. Essa base de informação foi essencial para criarmos um score que mede a proximidade de um texto qualquer em relação à língua portuguesa. Essa medida é baseado na ideia de log-verossimilhança e é fundamental na aceitação ou rejeição de amostras aleatórias na aplicação do algoritmo Metropolis-Hastings.

4 Resultados

4.1 Preparação do código em R

Para a implementação do algoritmo de Metropolis-Hastings, precisamos realizar algumas etapas prévias no processo de elaboração do código. Em um primeiro momento, se faz necessário ler uma lista grande de palavras em português a partir de um dicionário ou um livro, por exemplo. Além disso, precisamos nos certificar que todas as palavras possuem letras minúsculas, sem acentos ou símbolos e caracteres especiais. Além das letras, o único elemento textual mantido corresponde aos espaços em branco entre as palavras, uma vez que algumas não terão um número par de letras e precisamos considerar as letras mais prováveis para começar e finalizar palavras. Os livros de referência em língua portuguesa foram alocados em um único arquivo *txt* e são do autor Carl Sagan: *Cosmos* (1980), *Contato* (1985) e *O Mundo Assombrado pelos Demônios* (1995).

```
# PACOTES UTILIZADOS ----

library(tidyverse)
library(readr)
library(stringi)
library(stringr)
library(purrr)
library(glue)


# LEITURA DO TEXTO DE REFERÊNCIA ----
livros_referencia <- read_file("livros_Carl_Sagan.txt")



# MANIPULAÇÃO DO TEXTO ----
livros_referencia <- livros_referencia |>
                 str_to_lower() |> # letra minúscula
                 gsub(pattern = "[^A-Za-z ]+", replacement = "", x=_) |>
                 stri_trans_general(id = "Latin-ASCII")
```

A próxima ação é tentar estimar a probabilidade de transição de uma letra do alfabeto para outra seguinte em uma palavra. Dentro de um conjunto de possibilidades, qual é a probabilidade de termos uma letra “a” seguida de uma letra “b”, por exemplo? Uma forma de fazermos esse cálculo é juntar a lista de palavras do livro em uma única *string*, dividir em um vetor com pares de letras ou letra mais espaço, gerar uma tabela com os pares de letras encontrados e estimar a probabilidade a

partir da divisão entre a frequência que cada par foi encontrado e a frequência total de todos os pares existentes no banco de dados.

```
# PARES DE LETRAS ----
pares_letras <- function(texto){
  indice_inicial <- 1 : (nchar(texto) - 1)
  indice_final <- indice_inicial + 1
  return(stri_sub(texto, from = indice_inicial,
                         to = indice_final))
}


texto_ref_pares_letras <- pares_letras(livros_referencia)

# Verificando alguns pares obtidos
texto_ref_pares_letras[10000:10030]
```

```
##  [1] "rm" "ma" "ar" "rm" "mo" "os" "s " " " u" "um" "ma" "a " " " c" "co" "om" "mp"
## [16] "pa" "an" "nh" "hi" "ia" "a " " " d" "de" "e " " " p" "pr" "ro" "od" "du" "uc"
## [31] "co"
```

```
# Combinações mais prováveis
matriz_probabilidade <-
  table(texto_ref_pares_letras) / length(texto_ref_pares_letras)


# Probabilidade estimada - Top 10
matriz_probabilidade  %>%
  sort(decreasing = TRUE)  %>%
  head(10)
```

```
## texto_ref_pares_letras
##                    a          o          s          e          d          e
## 0.11324879 0.02929575 0.02825492 0.02799453 0.02697660 0.01778220 0.01517284
##          a         de         os
## 0.01484553 0.01484517 0.01388289
```

Esse tipo de raciocínio pode ser entendido como a construção de uma matriz de transição sob a perspectiva de uma Cadeia de Markov, uma vez que estamos estimando a probabilidade de uma cadeia ficar ou sair de um estado e ir para outro. Nesse caso, o conjunto de estados envolve a sequência de todas as letras possíveis e o espaço em branco. Exemplo: “processos estocásticos” - transição de “p” para “r”, de “r” para “o”, de “o” para “c”, e assim sucessivamente. Uma manipulação adequada envolve garantir que pares não encontrados no texto de referência não apresentem valores ausentes na probabilidade estimada quando pegarmos um novo texto, logo, incorporamos uma probabilidade mínima maior que 0.

```r
# INCLUSÃO DE COMBINAÇÕES NÃO ENCONTRADAS ----
matriz_probabilidade_final <- function(par_letras){
  probabilidades <- matriz_probabilidade[par_letras]

  if (is.na(probabilidades)) {
    return(1 / length(texto_ref_pares_letras))
  } else{
    return(probabilidades)
  }
}
```

Após estimar a probabilidade de cada combinação de caracteres em Português, podemos definir o quanto um novo texto, mensagem ou frase é similar à língua portuguesa. Nesse sentido, podemos incorporar a ideia de verossimilhança, na qual dividimos o novo texto em pares de letras, pegamos a probabilidade de ocorrência dos pares encontrados e multiplicamos. Para facilitar o processamento e a leitura da escala, utilizamos o logaritmo das probabilidades. Nessa perspectiva, teremos um score dado pela log-verossimilhança da distribuição de probabilidade dos pares de caracteres. Essa etapa pode ser dividida da seguinte forma:

- Divisão do novo texto em pares de caracteres (letras e espaço em branco);
- Coleta das probabilidades de cada par encontrado a partir da matriz de transição gerada pelo texto de referência (livros em txt);
- Cálculo do logaritmo das probabilidades;
- Soma do logaritmo das probabilidades para obtenção da log-verossimilhança do texto.

```r
# LOG-VEROSSIMILHANÇA - SCORE
log_vero_texto <- function(texto){
    texto  %>%
    pares_letras()  %>%
    map_dbl(matriz_probabilidade_final)  %>%
    log()  %>%
    sum()
}
```

```
log_vero_texto("texto em lingua portuguesa")
```

```
## [1] -138.1152
```

```
log_vero_texto("fghrx gh wghdfr etfsasrcva")
```

```
## [1] -219.6002
```

O próximo passo consiste em gerar três funções. A primeira é um gerador automático de *cipher* (criptografia). Poderíamos pegar um já existente em sites da Internet ou criar um manualmente, entretanto, a prática de deixar esse processo automático é mais adequada. Nessa função, pegamos as letras do alfabeto e permutamos de forma aleatória, dessa forma, cada letra de uma mensagem estará atrelada a uma outra letra única. A segunda função nos ajuda a pegar um texto e codificar a partir do *cipher* criado previamente. A terceira função nos ajuda a reverter o processo, ou seja, pegamos a mensagem embaralhada e sem significado, associamos a um *cipher* verdadeiro e a mensagem é decodificada. No algoritmo de Metropolis Hastings essa última função é importante para incorporarmos todos os *ciphers* aceitos no processo e que cada vez mais se aproximam do verdadeiro.

```
# GERAÇÃO DE UMA CODIFICAÇÃO CIPHER ----

# Gerando um cipher permutando as letras do alfabeto
gerar_cipher <- function() sample(letters, replace = FALSE)

# Codificando um texto usando um cipher
codificar_texto <- function(texto, cipher) {
    chartr(x = texto,
    old = paste(letters, collapse = ""),
    new = paste(cipher, collapse = "")
  )
}

# Decodificando um texto a partir de um cipher fornecido
decodificar_texto <- function(texto_codificado, cipher) {
  chartr( x = texto_codificado,
    old = paste(cipher, collapse = ""),
    new = paste(letters, collapse = "")
  )
}
```

Por fim, podemos desenvolver uma função que nos auxiliará no algoritmo. Como a ideia é dar um chute inicial para o *cipher* que decodifica a mensagem e depois ir atualizando até que haja uma convergência para o *cipher* verdadeiro, então, precisamos atualizá-lo segundo algum critério. Nessa perspectiva, a cada iteração podemos pegar duas letras e mudá-las de posição de forma aleatória. Essa etapa é fundamental, visto que vamos alterar a posição das letras no *cipher* até que os resultados obtidos sejam satisfatórios e revelem a mensagem do texto.

```r
# FUNÇÃO PARA ATUALIZAÇÃO DA CODIFICAÇÃO ATUAL ----

atualizacao <- function(x){
  # Select two distinct indices
  indices_aleatorizacao <- sample(1:length(x), size = 2, replace=FALSE)
  elemento_1 <- x[indices_aleatorizacao[1]]
  elemento_2 <- x[indices_aleatorizacao[2]]

  x[indices_aleatorizacao[1]] <- elemento_2
  x[indices_aleatorizacao[2]] <- elemento_1

  return(x)
}
```

4.2 Aplicação do algoritmo Metropolis-Hastings

A concepção geral do algoritmo de Metropolis-Hastings é gerar uma Cadeia de Markov que converge para uma distribuição estacionária. No contexto da aplicação, queremos que a cadeia transite pelo conjunto de estados definido como a combinação de possibilidades de criptografias (*ciphers* = 26!). O caminho percorrido pela Cadeia de Markov deve ser a mais eficiente possível, no qual os *ciphers* com os melhores scores devem ser buscados e os *ciphers* que apresentam textos mais distantes da língua portuguesa devem ser evitados [2].

Geramos um *cipher* (criptografia) de forma aleatória para darmos um ponto inicial para a cadeia. Posteriormente, desenvolvemos um código que segue a seguinte configuração:

- Pegue o *cipher* atual e permute duas letras de forma aleatória. Esse *cipher* será a nossa proposta de atualização;
- Calcule o score para o *cipher* atual e o *cipher* proposto após permutação das letras;
- Faça a razão entre os dois scores (score do *cipher* proposto / score do *cipher* atual). Nesse caso, como utilizamos o logaritmo no cálculo dos scores, essa divisão se torna uma diferença;
- Se a razão for maior que 1 (diferença maior que 0), então a nova criptografia proposta se torna o *cipher* atual. Isso ocorre, pois a criptografia proposta produz um texto decodificado mais próximo da língua portuguesa do que a anterior.
- Se a razão for menor que 1 (diferença menor que 0), então a nova criptografia proposta será aceita com probabilidade igual a razão entre os scores e será rejeitada com probabilidade do evento complementar;
- Repetir o processo até convergência da cadeia em um comando *for*.

Quanto mais próximo da língua portuguesa o texto gerado pelo *cipher* proposto estiver, maior é a probabilidade de aceitação. A Cadeia de Markov, nesse sentido, viaja pelos *ciphers* que produzem resultados mais adequados e ignora aqueles que estão mais distantes do objetivo de traduzir a verdadeira mensagem por trás do texto. Em suma, o algoritmo está retirando amostras de forma aproximada da distribuição das criptografias dado um texto codificado [2].

```
# ALGORITMO METROPOLIS-HASTINGS - Exemplo 1 ----

# Mensagem que queremos obter
mensagem <- "olhem de novo esse ponto. e aqui, e a nossa casa, somos
nos. nele, todos a quem ama, todos a quem conhece, qualquer um sobre
quem voce ouviu falar, cada ser humano que ja existiu, viveram as su
as vidas. nao ha, talvez, melhor demonstracao da tola presuncao huma
na do que esta imagem distante do nosso minusculo mundo. para mim, d
estaca a nossa responsabilidade de sermos mais amaveis uns com os ou
tros, e para preservarmos e protegermos o palido ponto azul, o unico
lar que conhecemos ate hoje."


# Gerar um cipher aleatorio para ser o verdadeiro cipher
cipher_original <- gerar_cipher()
cipher_original
```

```
##  [1] "i" "m" "o" "e" "t" "p" "u" "j" "w" "c" "a" "n" "h" "z" "v"
"q" "r" "y" "k"
## [20] "x" "l" "s" "f" "b" "d" "g"
```

```
# Codificando a mensagem
texto_codificado <- codificar_texto(texto = mensagem,
                                    cipher = cipher_original)
texto_codificado
```

```
## [1] "vnjth et zvsv tkkt qvzxv. t irlw, t i zvkki oiki, kvhvk zvk.
ztnt, xvevk i rlth ihi, xvevk i rlth ovzjtot, rlinrlty lh kvmyt rlth
svot vlswl piniy, oiei kty jlhizv rlt ci tbwkxwl, swstyih ik klik sw
eik. ziv ji, xinstg, htnjvy ethvzkxyioiv ei xvni qytklzoiv jlhizi ev
rlt tkxi whiuth ewkxizxt ev zvkkv hwzlkolnv hlzev. qiyi hwh, etkxioi
i zvkki ytkqvzkimwnweiet et ktyhvk hiwk ihistwk lzk ovh vk vlxyvk, t
qiyi qytktysiyhvk t qyvxtutyhvk v qinwev qvzxv igln, v lzwov niy rlt
ovzjtothvk ixt jvct."
```

```
# Criacao de um cipher aleatorio para iniciar a Cadeia de Markov
cipher_atual <- gerar_cipher()
cipher_atual
```

```
##  [1] "d" "h" "o" "q" "f" "z" "c" "a" "r" "e" "y" "g" "p" "x" "n"
"i" "k" "s" "j"
## [20] "v" "b" "w" "t" "l" "m" "u"
```

```r
# Contagem de aceitacao
i <- 0

for (iter in 1:1000) {

  # Proposta de um novo cipher - atualização de duas letras no cipher atual
  cipher_proposta <- atualizacao(cipher_atual)

  # texto decodificado do cipher proposto
  texto_decodificado_proposta <- decodificar_texto(texto_codificado,
                                      cipher = cipher_proposta)

  # texto decodificado do cipher atual
  texto_decodificado_atual <- decodificar_texto(texto_codificado,
                                     cipher = cipher_atual)

  # Log-verossilhança - texto decodificado do cipher proposto
  log_vero_proposta <- log_vero_texto(texto_decodificado_proposta)

  # Log-verossilhança - texto decodificado do cipher atual
  log_vero_atual <- log_vero_texto(texto_decodificado_atual)
```

```r
  # Log-verossilhança - texto decodificado do cipher atual
  log_vero_atual <- log_vero_texto(texto_decodificado_atual)

  # Probabilidade de aceitação - Metropolis Hastings
  prob_aceitacao <- min(1, exp(log_vero_proposta - log_vero_atual))

  # aceitacao or not with probability given by `prob_aceitacao`
  aceitacao <- sample(c(TRUE, FALSE),
                      size=1, prob = c(prob_aceitacao, 1-prob_aceitacao))

  # Se aceitarmos, então o cipher atual é atualizado com o cipher proposto
  if (aceitacao) {
    cipher_atual <- cipher_proposta

    # Print do texto codificado quando houver aceitacao
    print(glue("Iteração {i}: {texto_decodificado_proposta}"))

    # Incremento na contagem de aceitacao
    i <- i + 1
  }
}
```

Transição 50: esfur tu cege ummu vecpe. u anoi, u a cemma dama, merem cem. cusu, petem a nour ara, petem a nour decfudu, noasnoul or mexlu nour gedu eogio jasal, data mul forace nou wa uhimpio, gigular am moam gitam. cae fa, pasguz, rusfel turecmpladae ta pesa vlumocdae foraca te nou umpa iraqur timpacpu te cemme ricomdose rocte. vala rir, tumpada a cemma lumvecmaxisitatu tu mulrem raim araguim ocm der em eoplem, u vala vlumulgalrem u vlepuqulrem e vasite vecpe azos, e ocide sal nou decfudurem apu fewu.

Transição 75: entam da sefe arra pesve. a ogiu, a o serro coro, remer ser. sana, veder o giam omo, veder o giam cestaca, giongial im rexla giam feca eifui jonol, codo ral timose gia ho aqurvui, fufalom or rior fudor. soe to, vonfab, mantel damesrvlocoe do veno plariscoe timoso de gia arvo umozam durvosva de serre musircine misde. polo mum, darvoco o serro larpesroxunudoda da ralmer mour omofaur isr cem er eivler, a polo plaralfolmer a plevazalmer e ponude pesve obin, e isuce nol gia cestacamer ova teha.

Transição 100: olgem de novo esse ponto. e aqui, e a nossa casa, somos nos. nele, todos a quem ama, todos a quem congece, qualquer um sobre quem voce ouviu halar, cada ser gumano que fa existiu, viveram as suas vidas. nao ga, talvez, melgor demonstracao da tola presuncao gumana do que esta imajem distante do nosso minusculo mundo. para mim, destaca a nossa responsabilidade de sermos mais amaveis uns com os outros, e para preservarmos e protejermos o palido ponto azul, o unico lar que congecemos ate gofe.

Transição 150: olhem de novo esse ponto. e aqui, e a nossa casa, somos nos. nele, todos a quem ama, todos a quem conhece, qualquer um sobre quem voce ouviu walar, cada ser humano que fa existiu, viveram as suas vidas. nao ha, talveg, melhor demonstracao da tola presuncao humana do que esta imazem distante do nosso minusculo mundo. para mim, destaca a nossa responsabilidade de sermos mais amaveis uns com os outros, e para preservarmos e protezermos o palido ponto agul, o unico lar que conhecemos ate hofe.

Transição 4.788: olhem de novo esse ponto. e aqui, e a nossa casa, somos nos. nele, todos a quem ama, todos a quem conhece, qualquer um sobre quem voce ouviu falar, cada ser humano que ja existiu, viveram as suas vidas. nao ha, talvez, melhor demonstracao da tola presuncao humana do que esta imagem distante do nosso minusculo mundo. para mim, destaca a nossa responsabilidade de sermos mais amaveis uns com os outros, e para preservarmos e protegermos o palido ponto azul, o unico lar que conhecemos ate hoje.

“Olhem de novo esse ponto. É aqui, é a nossa casa, somos nós. Nele, todos a quem ama, todos a quem conhece, qualquer um sobre quem você ouviu falar, cada ser humano que já existiu, viveram as suas vidas. (...) Não há, talvez, melhor demonstração da tola presunção humana do que esta imagem distante do nosso minúsculo mundo. Para mim, destaca a nossa responsabilidade de sermos mais amáveis uns com os outros, e para preservarmos e protegermos o pálido ponto azul, o único lar que conhecemos até hoje.” [5]

Figura 1: Pálido Ponto Azul: imagem da Terra feita a uma distância de 6 bilhões de km pela sonda Voyager 1 - NASA, fevereiro de 1990.

Ao observarmos os resultados acima, verificamos que houve uma convergência para o texto original (sem acentuação e letras maiúsculas). No início, o algoritmo ainda não consegue nos retornar uma decodificação satisfatória, entretanto, a partir da centésima transição da cadeia a mensagem começa a ficar cada vez mais clara. A partir da transição de número 150 a cadeia ainda apresenta uma leve dificuldade no processo, uma vez que a letra “g” está trocada com a letra “z” e a letra “f” com a letra “j”, por exemplo. A partir desse limite, a cadeia tende a se estabilizar alternando entre algumas configurações que chegam próximo de decodificar a mensagem, entretanto, a distribuição estacionária só atinge o resultado correto após um número maior de iterações. O processo demora, em média, meia hora com cerca de 400.000

iterações e quase 5000 passos dados pela cadeia das criptografias, considerando os caminhos que tendem a produzir resultados cada vez mais satisfatórios. Caso a intenção seja alcançar um nível regular de acertos em que identificamos a mensagem mas não encontramos o resultado exato, então, com cerca de 3.000 iterações e 150 transições (aceitação no algoritmo) já alcançamos um processo eficiente.

No exemplo acima utilizamos um texto do próprio Carl Sagan do livro *Pálido Ponto Azul: uma visão do futuro da humanidade no espaço*, de 1994. O que acontece se utilizarmos o texto de outro autor e, mais interessante, um texto com outro estilo de escrita. Nesse sentido, utilizamos o famoso poema de Gonçalves Dias, *Canção do Exílio* de 1843.

Transição 10:

lduri fpnni fpl hivlpdnia
suwp eiufi s aizdi,
ia ikpa, btp ibtd ysnxpdil,
uis ysnxpdil esls vi.

usaas ept fpl lida pafnpvia,
usaaia kincpia fpl lida ovsnpa,
usaasa zsabtpa fpl lida kdwi,
usaai kdwi lida ilsnpa.

pl edalin, ascdurs, i usdfp,
lida hnicpn puesufns pt vi;
lduri fpnni fpl hivlpdnia,
suwp eiufi s aizdi.

lduri fpnni fpl hndlsnpa,
btp fida uãs puesufns pt ei;
pl edalin – ascdurs, i usdfp –
lida hnicpn puesufns pt vi;
lduri fpnni fpl hivlpdnia,
suwp eiufi s aizdi.

uãs hpnldfi wpta btp pt lsnni,
apl btp pt ksvfp hini vi;
apl btp wpaontfp sa hndlsnpa
btp uis puesufns hsn ei;
apl bt'duwi ikdafp ia hivlpdnia,
suwp eiufi s aizdi.

Transição 312:

minha terra tem palmeiras
onde canta o sabia,
as aves, que aqui jorxeiam,
nao jorxeiam como la.

nosso ceu tem mais estrelas,
nossas vargeas tem mais flores,
nossos bosques tem mais vida,
nossa vida mais amores.

em cismar, soginho, a noite,
mais prager encontro eu la;
minha terra tem palmeiras,
onde canta o sabia.

minha terra tem primores,
que tais não encontro eu ca;
em cismar – soginho, a noite –
mais prager encontro eu la;
minha terra tem palmeiras,
onde canta o sabia.

nao permita deus que eu morra,
sem que eu volte para la;
sem que desfrute os primores
que nao encontro por ca;
sem qu'inda aviste as palmeiras,
onde canta o sabia.

Transição 1.322:

minha terra tem palmeiras
onde canta o sabia,
as aves, que aqui gorjeiam,
nao gorjeiam como la.

nosso ceu tem mais estrelas,
nossas varzeas tem mais flores,
nossos bosques tem mais vida,
nossa vida mais amores.

em cismar, sozinho, a noite,
mais prazer encontro eu la;
minha terra tem palmeiras,
onde canta o sabia.

minha terra tem primores,
que tais não encontro eu ca;
em cismar – sozinho, a noite –
mais prazer encontro eu la;
minha terra tem palmeiras,
onde canta o sabia.

nao permita deus que eu morra,
sem que eu volte para la;
sem que desfrute os primores
que nao encontro por ca;
sem qu'inda aviste as palmeiras,
onde canta o sabia.

A cadeia se tornou estacionária com 1.322 passos. Em suma, o algoritmo se torna mais eficiente quanto maior for o texto de referência para cálculo da matriz de transição entre letras e maior for o texto que queremos decodificar. A consequência é que obtemos mais informação em cada log-verossimilhança calculada como score de proximidade em relação à língua portuguesa. Além disso, o chute inicial que damos para o *cipher* que decodifica a mensagem também impacta no tempo para o alcance da estacionariedade da distribuição. Se o início da cadeia estiver mais próximo da verdadeira chave que revela a mensagem, então, a cadeia viaja menos e o algoritmo converge mais rapidamente.

5 Considerações Finais

Nesse estudo, foi visto uma aplicação do algoritmo de Metropolis-Hastings para a quebra de encriptação de textos em português. O algoritmo consiste em, a partir de uma matriz de transição que indica a probabilidade de uma letra i ser seguida de uma letra j, e uma métrica de log-verossimilhança das frases, testar várias funções de decrepitação até obter uma com o maior valor para a métrica, por um método de aceitação e rejeição análogo ao algoritmo de Metropolis-Hastings. Uma vez que essa métrica é maximizada, tem-se a frase mais provável de ocorrer pelas probabilidades de transição das letras.

Nas aplicações, foi visto que o algoritmo foi capaz de descifrar as mensagens, entretanto demorando um tempo longo para descriptografar-las. Vale notar também que alguns aspectos diferentes podem afetar no número de passos até a convergência do algoritmo, sendo eles o chute inicial do *cipher*, o tamanho do texto de referência e o tamanho do texto a ser traduzido.

Vale notar que esse algoritmo de quebra de criptografia se aplica apenas para o caso específico de criptografia mencionado em 3.1, e algoritmos de criptografia utilizados atualmente possuem graus de complexidade muito maiores, o que necessitaria de um algoritmo ainda mais poderoso para fazer a quebra da criptografia.

Referências

[1] Robert P. Dobrow. *Introduction to Stochastic Processes with R.* Wiley, 2016.

[2] Maximilian Rohde. Code-breaking with Markov Chain Monte Carlo (MCMC), 2022.

[3] Sheldon M Ross. *Introduction to probability models.* Academic press, 2014.

[4] Sheldon M Ross. *Simulation.* Academic press, sixth edition, 2023.

[5] Carl Sagan. *Pálido Ponto Azul: uma visão do futuro da humanidade no espaço.* Random House, 1994.

[6] Timeline - World History Documentaries. Alan Turing: The Scientist Who Saved The Allies | Man Who Cracked The Nazi Code | Timeline, 2022.

[7] Veritassium. How Quantum Computers Break The Internet... Starting Now, 2023.