Web Scraping: implementação de um modelo em R para extração de notícias jornalísticas.

2023-04-02

Carlos Alberto Alves de Meneses, Prof. Dr. Pedro Rafael Diniz Marinho

Departamento de Estatística - Universidade Federal da Paraíba Cidade Universitária, s/n, João Pessoa - PB calberto@cabobranco.tv.br, Pedro.rafael.marinho @gmail.com

Carlos Alberto Alves de Meneses, Prof. Dr. Pedro Rafael Diniz Marinho

Departamento de Estatística - Universidade Federal da Paraíba Cidade Universitária, s/n, João Pessoa - PB calberto@cabobranco.tv.br, Pedro.rafael.marinho@gmail.com

1 Introdução

Na televisão, como em qualquer outro veículo de comunicação de massa, o jornalista busca a factualidade, ou seja, fatos que sejam importantes para serem passadas para seus telespectadores.

Nesse sentido, cabe ao produtor de notícias "minerar" esses fatos, localizando-os o mais rápido possível, de preferência de forma exclusiva, antes da concorrência, e enviá-los para os repórteres produzirem as matérias com o intuito de gerar índices altos de audiência.

A tarefa do produtor de notícias implica em:

- Verificar o que as empresas concorrentes estão noticiando
- Utilizar métodos de radioescuta, para obter noticias que estão sendo vinculadas nas rádios locais
- Pesquisar nas redes sociais (internet) sobre os assuntos mais comentados no momento
- Produzir as pautas do dia contendo um resumo das noticias que serão utilizadas como bases para que os repórteres preparem suas matérias

O produtor de notícias é, em geral, um profissional formado em jornalismo e faz parte de uma equipe de profissionais (chefe de redação, editores de texto, etc.) que trabalham em uma redação de jornalismo numa empresa de Televisão, por exemplo.

No entanto, nos dias atuais as notícias surgem com uma rapidez muito grande devido ao fato das pessoas possuírem equipamentos com câmeras e microfones de alta qualidade, como smartphones, que produzem dados em tempo real, gerando cotidianamente um volume de dados, principalmente nas redes sociais, gigantesco.

Com isso, faz-se necessário que a mineração de dados da WEB seja uma tarefa automatizada através da aplicação de algoritmos.

Diante disso, a proposta deste projeto é a utilização de técnicas de *web scraping* a fim de facilitar a mineração desses dados e produzir relatórios para os produtores de televisão automaticamente.

1.2 Objetivos

Geral: implementar um modelo de web scraping para raspagem de notícias da internet.

Objetivos específicos:

- Analisar as bibliotecas da linguagem R utilizadas na web scraping;
- Elaborar e documentar o código de web scraping utilizado no modelo;
- Testar e implementar o modelo proposto.

1.3 Metodologia

Em termos metodológicos, este projeto será desenvolvido para oferecer um guia rápido sobre técnicas e softwares de web scraping que podem ser usados para extrair dados de sites.

Os dados são uma parte essencial de qualquer pesquisa, seja ela acadêmica, de marketing ou cientifica.

Para Lima (2022), uma das coisas mais importantes no Campo da Ciência de Dados é a habilidade de obter dados certos para o problema que você deseja resolver.

A World Wide Web (WWW) contém todos os tipos de informações de diferentes fontes. As pessoas podem querer coletar e analisar dados de vários sites.

1.4 Fundamentação teórica

Segundo Silva (2021) existem várias maneiras de extrair informações da web, sendo o uso de APIs, provavelmente, a melhor maneira de extrair dados de um website, os quais estão disponíveis no Twitter, Facebook, Google, StackOverflow, dentre outros.

Dentre as técnicas de programação de extração de dados da web, a Web Scraping permite obter informações de qualquer página da web por ser uma técnica que foca, principalmente, na transformação de dados não estruturados (formato HTML) da Web em dados estruturados (Banco de dados, JSON ou Planilha).

A web scraping é, portanto, uma prática de coletar dados por qualquer meio que não seja um programa interagindo com uma API (ou, obviamente, por um ser humano usando um navegador web). Isso é comumente feito escrevendo um programa automatizado que consulta um servidor web, requisita dados (em geral, na forma de HTML e de outros arquivos que compõem as páginas web) e então faz análises desses dados para extrair as informações necessárias. Na prática, a web scraping engloba uma grande variedade de técnicas de programação e de tecnologias, por exemplo, análise de dados, análise de idiomas naturais e segurança de informação.

Sendo assim, independentemente da área profissional, a web scraping quase sempre oferece uma forma de orientar práticas de negócios com mais eficiência, melhorar a produtividade ou até mesmo dar origem a uma área totalmente nova (MITCHELL, 2019).

Quanto à linguagem de programação de código aberto (open source), pode-se fazer uso do software R por ter muitas bibliotecas para executar funções de Web Scraping, tais como:

- rvest
- Rselenium

2. Web Scraping

Nessa primeira fase do projeto iremos nos concentrar na obtenção dos dados de apenas um site, neste caso do *portal do Jornal da Paraíba*, deixando a implementações de outros sites de noticias para trabalhos futuros.

Executando a raspagem da web passo a passo, usando o pacote rvest R escrito por Hadley Wickham.

O *rvest* é uma biblioteca R muito útil que ajuda a coletar informações da páginas da web.

A primeira providência é baixar e carregar os pacotes necessários.

```
# Baixando e carregando o pacote
library( rvest )
library( xml2 )
library(shiny)
```

```
library(httr)
library(httr2)
```

O pacote rvest define o link da página da web como o primeiro passo. Depois disso, os rótulos apropriados devem ser definidos. A linguagem HTML edita o conteúdo usando várias tags e seletores.

Esses seletores devem ser identificados e marcados para armazenamento de seu conteúdo. Em seguida, todos os dados gravados podem ser transformados em um conjunto de dados apropriado e a análise pode ser realizada.

Coletaremos um conjunto de dados de um portal de notícias (www.jornaldaparaiba.com.br). Este site fornece informações gerais e principalmente do estado da Paraíba.

Vamos comerçar a coletar informações para descobrir quais as principais noticias que estão sendo vinculadas neste portal.

- Para coletar as informações sobre as principais manchetes do portal, usaremos a URL da página de destino do site.

```
# Link para:
url <-  " https://jornaldaparaiba.com.br/ "
url
```

```
## [1] " https://jornaldaparaiba.com.br/ "
```

Como mencionamos, estamos interessado em coletar dados sobre as materias jornalisticas publicadas no site.

Agora, a parte que mais nos importa: a coleta dos dados!

O script a seguir, fornece os seguintes procedimentos: visite o URL da página da Web, coletando nós HTML usando a função *read_html*.

Para analisar nós HTML, estamos usando as regras XPath.

```
# Ler o HTML da página da web
url <- c("https://www.jornaldaparaiba.com.br")

pagina <- read_html(url)

elementos <- pagina %>%
  html_nodes("div.container a")%>% #Aplicando as regras XPath
 # html_nodes(".titles")
   html_text2()
```

XPath lida principalmente com os nós das árvores XML 1.0 ou XML 1.1. Ë usado para representar a estrutura hierárquica de um documento XML.

XPath usa sintaxe não XML e funciona na estrutura lógica de documentos XML. XPath é projetado para ser usado embutido em uma linguagem de programação.

XPath tem sete tipos diferentes de nós: elemento, atributo, texto, namespace, instrução de processamento, comentário e nós de documento.

Para este projeto, estamos usando a função *html_nodes* e definindo nossas regras XPath , que já temos, dentro da função:

html_nodes(““div.container a”“)

O código a seguir realiza as duas funções anteriores e exibe o conteúdo da raspagem dos dados através da função *cat*.

```
# Carregar o pacote rvest
library(rvest)
library(xml2)
library(shiny)
library(httr)
library(httr2)

# Ler o HTML da página da web
url <- "https://www.jornaldaparaiba.com.br"

pagina <- read_html(url)

dt <- html_nodes(pagina, "div.container a")

# Exibir o conteúdo dos elementos selecionados
cat("Conteúdo dos elementos selecionados:\n")
```

```
## Conteúdo dos elementos selecionados:
```

```
#cat(html_text(dt), sep = "\n")
head(html_text(dt))
```

```
## [1] "\n                                                \n                                                    Economia
## [2] "\n                                                \n                                                \n
## [3] "\n                                                \n                                                \n
## [4] "\n                                          \n                                          \n
## [5] "\n                                          \n                                              Concursos e Empr
## [6] "\n                                          \n                                          \n
```

Nesse ponto, concluímos a primeira fase do nosso projeto, realizando a raspagem das manchetes principais do *portal do Jornal da Paraíba*.

2.1 Processamento dos dados

Para Moreira e Rocabado (2022), a análise de conteúdo só é possível através da transformação do texto bruto em estruturas de dados convenientes para análise.

- Temos basicamente três estruturas num texto:
- **Tokens:** O texto pode ser armazenado em n-grams ou cadeias de caracteres (*strings*).
- **Corpus:** Estruturas que armazenam tanto o conteúdo de documentos como seus metadados.

2,2 Matris de documentos e termos (DFM ou DTM):

É uma matriz esparsa com uma linha para cada documento e uma coluna para cada termo.

A análise do conteúdo do texto como dado exige versatilidade na transformação entre estruturas.

Utilizaremos especialmente os pacotes *tidytext* e *quanteda*, ambos estruturais para análise de conteúdo.

Antes de aplicarmos a tokenizaão nos nossos dados (texto) coletados, precisamos transformá-los para o formato aceito pelo *tidytext* que é um data.frame.

```
library(tidyverse)
library(tidytext)
library(quanteda)
library(dplyr)
library(rvest)
library(xml2)
library(tibble)
```

```r
texto <- (html_text(dt))
texto <- str_replace_all(texto,"\\(",")")
texto <- str_replace_all(texto, "\\)","")
texto <- str_replace_all(texto,"\n","")
texto <- str_replace_all(texto,"\t","")
texto
```

```
##  [1] "                                                                                 Economia
##  [2] "
##  [3] "
##  [4] "
##  [5] "                                                                          Concursos e Emprego
##  [6] "
##  [7] "
##  [8] "
##  [9] "
## [10] "                                                                          Concursos e Emprego
## [11] "
## [12] "
## [13] "
## [14] "
## [15] "
## [16] "                                          Conversa Política            Gervásio Maia será de
## [17] "                                          Concursos e Emprego            Inscrições em seleç
## [18] "                                          Conversa Política            Projeto das Fake News
## [19] "Carregar mais"
## [20] "
## [21] "
## [22] "
## [23] "
## [24] "
## [25] "                                          Concursos e Emprego            Inscrições em seleç
## [26] "                                          Concursos e Emprego            Inscrições no concu
## [27] "                                          Saúde            João Pessoa suspende vacinação co
## [28] "                                          Clima e Tempo            Inmet emite alertas de ac
## [29] "                                          Concursos e Emprego            Concurso e seleções
## [30] "                                          Esportes            Miullen, do Botafogo-PB, inter
## [31] "                                          Comunidade            Jovens indígenas buscam pela
## [32] "                                          Concursos e Emprego            Inscrições em seleç
## [33] "                                          Esportes            Clássico Emoção: Botafogo-PB v
## [34] "                                          Esportes            Unifacisa vence o Franca no jo
## [35] "Carregar mais"
## [36] "Qual é a Boa?Bixarte lança o álbum ‘Traviacardo’; veja entrevista"
## [37] "Qual é a Boa?Agnes Nunes fala sobre turnê internacional"
## [38] "Ver todos"
## [39] ""
## [40] "Conversa PolíticaFatos políticos de forma clara e opinativa. Angélica Nunes e Laerte Cerqueira
## [41] "Pleno PoderNotícias da cobertura política de Campina Grande e de todo o interior da Paraíba, p
## [42] "Caderno AnimalFabi e Miguel Cavalcanti abordam comportamento e saúde dos pets. Dicas, curiosid
## [43] "Sílvio OsiasMuita cultura e uma pitada de entretenimento, com abordagem de obras e questões re
## [44] "Saúde AlertaDicas e orientações para cuidar da saúde e bem estar com o médico André Telis."
## [45] "Ver mais"
## [46] ""
```

```
## [47] ""
## [48] ""
## [49] ""
## [50] ""
## [51] ""
## [52] ""
## [53] ""
## [54] ""
## [55] ""
## [56] ""
## [57] ""
```

```
#texto <- as.vector(texto)


text_df <- tibble(line=length(texto), text= texto)

text_df
```

```
## # A tibble: 57 x 2
##     line text                                                                ~
##    <int> <chr>                                                             ~
##  1    57 "                                                                 ~
##  2    57 "                                                                 ~
##  3    57 "                                                                 ~
##  4    57 "                                                                 ~
##  5    57 "                                                                 ~
##  6    57 "                                                                 ~
##  7    57 "                                                                 ~
##  8    57 "                                                                 ~
##  9    57 "                                                                 ~
## 10    57 "                                                                 ~
## # i 47 more rows
```

2,4 Armazenando dados como data.frame.

- Salvando dados em disco no formato csv

Se os dados já estiverem armazenados como um data.frame:

```
write.csv(text_df,file = "text_df.csv")
```

Um objeto *tibble* é uma classe moderna de *data.frame* dentro do R, disponível nos pacotes *dplyr* e *tibble*, que possui um método de impressão conveniente, não converte *strins* em fatores e não usa nomes de linha (MOREIRA; ROCABADO, 2022).

Porém, o objeto *tibble* ainda não está no formato aceito pelo pacote *tidytext*.

Precisaremos converter-lo em outro formato que atenda a condição *one-token-per-document-per-row*.

Portanto, cada *token unigram* (cada palavra) deve ser um valor indicado. Utilizaremos a função *unnest_tokens* do pacote *tidytext* para realizar o processo de *tokenização*.

2,5 Tokens

- O formato tidytext

Usar os principios do *tidytext* é uma maneira poderosa de tornar o processamento de dados mais ágil e eficaz. Conforme Wickham (2014), os dados organizados têm uma estrutura especifica:

- Cada variável é uma coluna;
- Cada observação é uma linha;
- Cada tipo de unidade de observação é uma tabela.

Assim, o formato do *tidytext* segue a mesma estrutura apresentada, na qual cada linha/observação possui uma unidade de texto significativa, também chamada por *token*, estes organizadas em uma coluna/variável.

O *token* pode ser uma única palavra, um conjunto de palavras, uma frase ou um parágrafo. O processo consiste em realizar a *tokenização*, em que dividimos o texto em *tokens*.

2.6 Token usando a função unnest_tokens

```
library(tidytext)

text_token <- text_df %>%
  unnest_tokens(word, text)

text_token
```

```
## # A tibble: 685 x 2
##     line word
##    <int> <chr>
##  1    57 economia
##  2    57 preço
##  3    57 do
##  4    57 botijão
##  5    57 de
##  6    57 gás
##  7    57 sofre
##  8    57 reajuste
##  9    57 a
## 10    57 partir
## # i 675 more rows
```

Iremos remover elementos da nossa base de dados (caso haja) que não agregam valor a depender da análise. esses lementos são chamados de *stopwords* e podemos removê-los utilizando o pacote *quanteda*.

```
#stopwords
library(quanteda)

stop_w <- tibble(word = stopwords(source = "stopwords-iso", language = "pt"))

#retirar do corpus as stopwords
tidy_text <- text_token %>%
  anti_join(stop_w)
```

Com a nossa base de dados no formato *tidy* podemos iniciar algumas análises como, por exemplo, a contagem da frequência de palavras ou *tokens*.

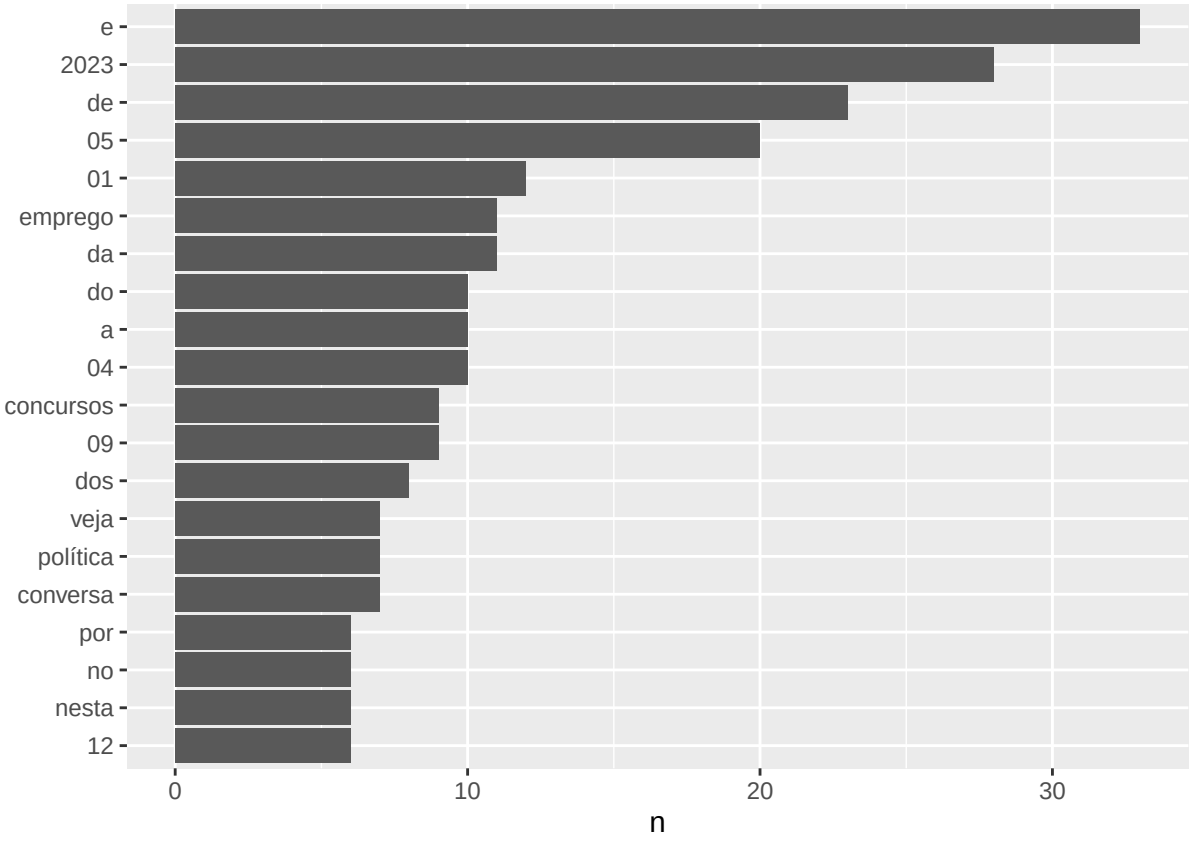
```
text_token %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 305 x 2
##    word        n
##    <chr>   <int>
##  1 e          33
##  2 2023       28
```

```
##  3 de          23
##  4 05          20
##  5 01          12
##  6 da          11
##  7 emprego     11
##  8 04          10
##  9 a           10
## 10 do          10
## # i 295 more rows
```

Também podemos visualizar as frequências calculadas:

```
text_token %>%
  count(word, sort = TRUE) %>%
  mutate(word = fct_reorder(word, n)) %>%
  slice(1:20) %>%
  ggplot(aes(word, n)) +
  geom_col() +
  coord_flip() +
  labs(x="")
```



Com a base de dados em formato *tidy*, podemos utilizar os meta-dados dos documentos em nossas análises.

2,7 Selecionando tokens

- Tokens sem pontuação e sem números

```
#Transformando o data.frame em um corpus
corp <- corpus(texto)
summary(corp, 10)
```

```
## Corpus consisting of 57 documents, showing 10 documents:
##
##    Text Types Tokens Sentences
##   text1    20     21         1
##   text2    23     24         2
##   text3    22     25         1
##   text4    12     13         1
##   text5    14     14         1
##   text6    11     11         1
##   text7    16     16         1
##   text8     8      8         2
##   text9    17     21         1
##  text10    21     22         1
```

```
#Tokens sem pontuação e sem sequência
toks <- tokens(corp, remove_punct = T, remove_numbers = T)
```

- Tokens sem stopword

```
# Removendo as stopwords
toks_nostop <- tokens_select(toks, pattern = stopwords('pt'),
                             selection = 'remove')
```

2.8 Gerando n-grams

O método de apresentado nâo respeita a ordem em que as palavras foram escritas, no entanto, para algumas análises essa ordem importa. Para garantir essa ordem, podemos criar *tokens* com *N-grams* para garantir que a ordem de palavras esteja presente no processo de *tokenização*.

Através da função *tokens_ngrams()* é possível aplicar esse método.

```
# criando n-grams
toks_ngram <- tokens_ngrams(toks, n = 1:4)
toks_ngram
```

```
## Tokens consisting of 57 documents.
## text1 :
##  [1] "Economia" "Preço"    "do"       "botijão"  "de"       "gás"
##  [7] "sofre"    "reajuste" "a"        "partir"   "desta"    "segunda"
## [ ... and 34 more ]
##
## text2 :
##  [1] "Tecnologia"   "Quais"        "profissões"   "podem"        "ser"
##  [6] "extintas"     "ou"           "criadas"      "por"          "causa"
## [11] "da"           "inteligência"
## [ ... and 42 more ]
##
## text3 :
##  [1] "Conversa"  "Política"  "Justiça"   "determina" "proibição" "de"
##  [7] "corrida"   "de"        "jegues"    "por"       "suspeita"  "de"
## [ ... and 50 more ]
##
## text4 :
##  [1] "Economia"       "Bolsa"          "Família"        "veja"
##  [5] "data"           "de"             "pagamento"      "no"
##  [9] "mês"            "de"             "maio"           "Economia_Bolsa"
## [ ... and 26 more ]
```

```
##
## text5 :
##  [1] "Concursos"   "e"           "Emprego"     "Veja"        "as"
##  [6] "vagas"       "de"          "emprego"     "disponíveis" "no"
## [11] "Sine"        "esta"
## [ ... and 34 more ]
##
## text6 :
##  [1] "Esportes"       "Série"          "C"              "onde"
##  [5] "assistir"       "regulamento"    "e"              "classificação"
##  [9] "Esportes_Série" "Série_C"        "C_onde"         "onde_assistir"
## [ ... and 14 more ]
##
## [ reached max_ndoc ... 51 more documents ]
```

2.9 Nuvem de palavras

Uma forma da visualização de frequência na análise de texto é a nuvem de palavras.

Iremos utilizar a função *textplo_wordcloud()* do pacote *quanteda*.

```
library(quanteda.textplots)
#criando uma DFM com as hashtags
dfmat_texto <- dfm(toks)
set.seed(132)
textplot_wordcloud(dfmat_texto, max_words = 100)
```

3. Análise exploratória

Existem diversas análises de dados que podem ser realizadas utilizando dados obtidos por meio de web scraping com o pacote rvest em um portal de notícias.

Vamos iniciar novamente, carregando o nosso banco com os dados obtidos através da web scraping.

```
library(rvest)
library(xml2)
library(dplyr)
library(knitr)
dt_noticias <- html_nodes(pagina,xpath=" //h1 |//h2 |//h5")%>%
html_text2()
dt_noticias
```

```
##  [1] "Preço do botijão de gás sofre reajuste a partir desta segunda (1º)"
##  [2] "Quais profissões podem ser extintas ou criadas por causa da inteligência artificial? ChatGPT re
##  [3] "Justiça determina proibição de corrida de jegues por suspeita de maus tratos e trabalho infanti
##  [4] "Bolsa Família 2023: veja data de pagamento no mês de maio"
##  [5] "Veja as 1.118 vagas de emprego disponíveis no Sine esta semana"
##  [6] "Série C 2023: onde assistir, regulamento e classificação"
##  [7] "Doenças do trabalho que mais afetam os brasileiros; veja sintomas, prevenção e tratamento"
```

```
##  [8] "Independente ou independentemente? Aprenda a diferença"
##  [9] "Bolsa Família 2023: veja data de pagamento no mês de maio"
## [10] "Veja as 1.118 vagas de emprego disponíveis no Sine esta semana"
## [11] "Série C 2023: onde assistir, regulamento e classificação"
## [12] "Doenças do trabalho que mais afetam os brasileiros; veja sintomas, prevenção e tratamento"
## [13] "Independente ou independentemente? Aprenda a diferença"
## [14] "Caso do Gari Milionário: Coriolano é absolvido e agente de limpeza da Emlur é condenado"
## [15] "Projeto das Fake News será votado nesta terça-feira; maioria dos deputados paraibanos é a favo
## [16] "Gervásio Maia será defensor do governo Lula na CPI dos Atos Golpistas"
```

```
#dt_titulos <- html_element(pagina,"section")%>%
#html_text2()
#dt_titulos

#Realizando a limpeza no texto
texto <- str_replace_all(dt_noticias,"\\(",")")
texto <- str_replace_all(dt_noticias, "\\)","")
texto <- str_replace_all(dt_noticias,"\n","")
texto <- str_replace_all(dt_noticias,"\t","")
texto
```

```
##  [1] "Preço do botijão de gás sofre reajuste a partir desta segunda (1º)"
##  [2] "Quais profissões podem ser extintas ou criadas por causa da inteligência artificial? ChatGPT re
##  [3] "Justiça determina proibição de corrida de jegues por suspeita de maus tratos e trabalho infanti
##  [4] "Bolsa Família 2023: veja data de pagamento no mês de maio"
##  [5] "Veja as 1.118 vagas de emprego disponíveis no Sine esta semana"
##  [6] "Série C 2023: onde assistir, regulamento e classificação"
##  [7] "Doenças do trabalho que mais afetam os brasileiros; veja sintomas, prevenção e tratamento"
##  [8] "Independente ou independentemente? Aprenda a diferença"
##  [9] "Bolsa Família 2023: veja data de pagamento no mês de maio"
## [10] "Veja as 1.118 vagas de emprego disponíveis no Sine esta semana"
## [11] "Série C 2023: onde assistir, regulamento e classificação"
## [12] "Doenças do trabalho que mais afetam os brasileiros; veja sintomas, prevenção e tratamento"
## [13] "Independente ou independentemente? Aprenda a diferença"
## [14] "Caso do Gari Milionário: Coriolano é absolvido e agente de limpeza da Emlur é condenado"
## [15] "Projeto das Fake News será votado nesta terça-feira; maioria dos deputados paraibanos é a favo
## [16] "Gervásio Maia será defensor do governo Lula na CPI dos Atos Golpistas"
```

```
texto_df <- tibble(line=length(texto), text= texto) %>%
  kable()

texto_df
```

line	text
16	Preço do botijão de gás sofre reajuste a partir desta segunda (1º)
16	Quais profissões podem ser extintas ou criadas por causa da inteligência artificial? ChatGPT responde
16	Justiça determina proibição de corrida de jegues por suspeita de maus tratos e trabalho infantil
16	Bolsa Família 2023: veja data de pagamento no mês de maio
16	Veja as 1.118 vagas de emprego disponíveis no Sine esta semana
16	Série C 2023: onde assistir, regulamento e classificação
16	Doenças do trabalho que mais afetam os brasileiros; veja sintomas, prevenção e tratamento
16	Independente ou independentemente? Aprenda a diferença
16	Bolsa Família 2023: veja data de pagamento no mês de maio
16	Veja as 1.118 vagas de emprego disponíveis no Sine esta semana

line	text
16	Série C 2023: onde assistir, regulamento e classificação
16	Doenças do trabalho que mais afetam os brasileiros; veja sintomas, prevenção e tratamento
16	Independente ou independentemente? Aprenda a diferença
16	Caso do Gari Milionário: Coriolano é absolvido e agente de limpeza da Emlur é condenado
16	Projeto das Fake News será votado nesta terça-feira; maioria dos deputados paraibanos é a favor
16	Gervásio Maia será defensor do governo Lula na CPI dos Atos Golpistas

3.1 Análise de sentimentos

A análise dos sentimentos ou a mineração de opinião é utilizada para extrair automaticamente informações sobre a conotação negativa ou positiva da linguagem de um documento. Embora seja uma tarefa que vem sendo utilizada há muito tempo no campo do marketing ou da política, em estudos literários ainda é uma abordagem recente e não há um método único. Além disso, há a possibilidade de extrair a polaridade dos sentimentos e também das emoções.

É importante especificar o que estamos procurando com os termos "sentimento" e "emoções", pois eles são frequentemente usados de forma intercambiável, de modo geral, mas são diferentes.

Embora não haja um acordo final sobre o número de emoções básicas, geralmente são seis: raiva, alegria, repugnância, medo, tristeza e surpresa.

Além disso, no caso do sistema automático que utilizaremos, as emoções secundárias de antecipação e confiança também aparecem.

```
# Instalar e carregar o pacote rvest
#install.packages("rvest")
library(rvest)
library(xml2)
# Definir a URL da página de notícias do G1
url <- "https://www.jornaldaparaiba.com.br/"

# Coletar os títulos e links das notícias
noticias <- read_html(url) %>%
  html_nodes(xpath=" //h1 |//h2 |//h5") %>%
  html_text() %>%
  data.frame(Título = .) %>%
  mutate(Link = read_html(url) %>%
           html_element("a") %>%
           html_attr("href"))

# Exibir as primeiras linhas do resultado
head(noticias)
```

```
##                                                                                                  Títu
## 1                                      Preço do botijão de gás sofre reajuste a partir desta segunda (
## 2 Quais profissões podem ser extintas ou criadas por causa da inteligência artificial? ChatGPT respor
## 3      Justiça determina proibição de corrida de jegues por suspeita de maus tratos e trabalho infan
## 4                                                Bolsa Família 2023: veja data de pagamento no mês de ma
## 5                                          Veja as 1.118 vagas de emprego disponíveis no Sine esta sema
## 6                                                Série C 2023: onde assistir, regulamento e classifica
##                              Link
## 1 https://jornaldaparaiba.com.br
## 2 https://jornaldaparaiba.com.br
## 3 https://jornaldaparaiba.com.br
```

```
## 4 https://jornaldaparaiba.com.br
## 5 https://jornaldaparaiba.com.br
## 6 https://jornaldaparaiba.com.br
```

Nesse código, estamos usando o read_html para carregar o conteúdo HTML da página de notícias do Jornal da Paraíba e, em seguida, o html_nodes e o html_text para coletar os títulos das notícias. Estamos também usando o html_attr para coletar os links das notícias e, por fim, criando um data.frame com os resultados.

3.2 Coletando informações sobre as noticias mais lidas e seus autores

Para obter dados sobre as notícias mais lidas e seus autores em um portal de notícias como o Jornal da Paraíba, procedemos dá seguinte forma:

- Extrair os dados das notícias mais lidas, incluindo o título, o link, o autor e os dados de publicação.

A seguir, temos o código em R que utilizaremos para coletar esses dados:

```r
# Instalar e carregar o pacote rvest
#install.packages("rvest")
library(rvest)
library(xml2)
library(tidytext)

# Definir a URL da página de notícias mais lidas do G1
url <- "https://www.jornaldaparaiba.com.br"

# Coletar os títulos, links, autores e datas das notícias mais lidas
noticias_lidas <- read_html(url) %>%
  html_nodes(xpath=" //h1 |//h2 |//h5") %>%
  html_text() %>%
  data.frame(Título = .) %>%
  mutate(Link = read_html(url) %>%
           html_element("a") %>%
           html_attr("href"),
         Autor = read_html(url) %>%
           html_element(xpath = "//div[2] |//div[1] |//section |//div |/ /h4 |//a") %>%
           html_text(),
         Data = read_html(url) %>%
          html_element(xpath = "//div[1] |//header |//div[1] |//div[1]") %>%
           html_text())

# Exibir as primeiras linhas do resultado
head(noticias_lidas)
```

```
##                                                                                  Títu
## 1                                  Preço do botijão de gás sofre reajuste a partir desta segunda (
## 2 Quais profissões podem ser extintas ou criadas por causa da inteligência artificial? ChatGPT respon
## 3      Justiça determina proibição de corrida de jegues por suspeita de maus tratos e trabalho infan
## 4                                          Bolsa Família 2023: veja data de pagamento no mês de ma
## 5                                    Veja as 1.118 vagas de emprego disponíveis no Sine esta sema
## 6                                         Série C 2023: onde assistir, regulamento e classifica
##                             Link
## 1 https://jornaldaparaiba.com.br
## 2 https://jornaldaparaiba.com.br
## 3 https://jornaldaparaiba.com.br
## 4 https://jornaldaparaiba.com.br
## 5 https://jornaldaparaiba.com.br
```

```
## 6 https://jornaldaparaiba.com.br
##
## 1 \n\n\t\t\t\t\n\t\t\n\t\t\n\t\t\n\t\t\n\n\t\t\n\t\t\t\t\n\t\t\t\n\t\t\t\n\t\t\t\n\t\t\t\tParaibano 2023\nQual é a
```

```
# Instalar e carregar os pacotes rvest e tidytext
#install.packages(c("rvest", "tidytext"))
library(rvest)
library(tidytext)
library(syuzhet)

# Definir a URL da página de notícias do Jornal da Paraíba

#url <- "https://www.jornaldaparaiba.com.br/"

# Coletar os títulos das notícias
noticias <- read_html(url) %>%
  html_nodes(xpath= "//h1 |//h2 |//h5") %>%
  html_text()

# Transformar os títulos em um data frame para a análise de sentimentos
df_noticias <- data.frame(Título = noticias, stringsAsFactors = FALSE)

# Separar cada palavra do título em uma linha
texto_palavras <- get_tokens(texto)
head(texto_palavras)
```

```
## [1] "preço"   "do"      "botijão" "de"      "gás"     "sofre"
```

No código acima, estamos coletando os títulos das notícias a partir da página do Jornal da Paraíba e envolvendo-os em um quadro de dados para a análise de sentimentos. Em seguida, estamos utilizando a função get_tokens para separar cada palavra do título em uma linha.

Como a análise que vamos realizar precisa de uma lista, seja de palavras ou de frases (aqui só prestaremos atenção a palavras individuais), precisamos de um passo intermediário entre o carregamento do texto e a extração dos valores de sentimento.

Assim, dividimos o texto (string) em uma lista de palavras (tokens). Isto é muito comum na análise distante de textos.

Agora, podemos visualizar as primeiras seis linhas do nosso token e podemos ver também quantas palavras ou tokens estão neste texto com a função length():

```
# Carregue os pacotes
library(syuzhet)
library(RColorBrewer)
```

```
library(wordcloud)
library(tm)

head(texto_palavras)
```

```
## [1] "preço"   "do"      "botijão" "de"      "gás"     "sofre"
```

```
length(texto_palavras)
```

```
## [1] 184
```

Para realizar a análise para orações, utilizamos a função get_sentences() e siguimos o mesmo processo:

```
oracoes_vetor <- get_sentences(noticias)
length(oracoes_vetor)
```

```
## [1] 19
```

3.3 Extração de dados com o NRC Sentiment Lexiconextração-de-dados-com-o-nrc-sentiment-lexicon

Agora podemos executar a função get_nrc_sentiment para obter os sentimentos contido nos títulos.

Como a função executa por padrão o vocabulário inglês, nós a escrevemos com o argumento "lang" (de language, ou idioma) para usar o vocabulário português ("portuguese"). Por sua vez, criamos um novo objeto para armazenar os dados extraídos. Este será um objeto do tipo data frame. Esta função procura a presença das oito emoções e dos dois sentimentos para cada palavra em nosso vetor, e atribui um número maior que 0 se elas existirem.

```
sentimentos_df <- get_nrc_sentiment(texto_palavras, lang="portuguese")
```

Podemos ler os resultados no novo objeto, simplesmente selecionando o objeto e executando-o. Mas para evitar "imprimir" milhares de linhas no console, também podemos usar a função head() para ver os primeiros seis tokens. No caso do texto que estamos usando, quando executarmos essa função, devemos ver o seguinte, que não é nada interessante:

```
head(sentimentos_df)
```

```
##   anger anticipation disgust fear joy sadness surprise trust negative positive
## 1     0            0       0    0   0       0        0     0        0        0
## 2     0            0       0    0   0       0        0     0        0        0
## 3     0            0       0    0   0       0        0     0        0        0
## 4     0            0       0    0   0       0        0     0        0        0
## 5     0            0       0    0   0       0        0     0        0        0
## 6     0            0       0    0   0       0        0     0        0        0
```

3.4 Resumo do texto

O que é interessante é ver um resumo de cada um dos valores que obtivemos utilizando a função geral summary(). Isto pode ser muito útil ao comparar vários textos, pois permite ver diferentes medidas, tais como a média dos resultados para cada uma das emoções e os dois sentimentos.

```
summary(sentimentos_df)
```

```
##      anger          anticipation       disgust             fear         
##  Min.   :0.000000   Min.   :0.00000   Min.   :0.000000   Min.   :0.00000  
##  1st Qu.:0.000000   1st Qu.:0.00000   1st Qu.:0.000000   1st Qu.:0.00000  
##  Median :0.000000   Median :0.00000   Median :0.000000   Median :0.00000  
##  Mean   :0.005435   Mean   :0.02174   Mean   :0.005435   Mean   :0.02717  
##  3rd Qu.:0.000000   3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu.:0.00000  
```

```
##  Max.   :1.000000   Max.   :1.00000   Max.   :1.000000   Max.   :1.00000
##       joy             sadness           surprise           trust
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.02717   Mean   :0.01087   Mean   :0.02174   Mean   :0.02717
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##     negative          positive
##  Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000
##  Mean   :0.05978   Mean   :0.07065
##  3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :2.00000   Max.   :3.00000
```

Nesse nosso exemplo, podemos ver um equilibrio, em média (mean), entre o positivo (0,05517) e o negativo (0,05517). Mas se olharmos para as emoções, parece que a tristeza (0,02759) aparece em mais momentos do que a alegria (0,02069). Vários dos valores fornecidos pela função de resumo do texto aparecem com um valor igual a 0, incluindo a mediana (median). Isto indica que poucas palavras do texto aparecem no dicionário que estamos usando (NRC) ou, inversamente, que poucas têm uma atribuição de sentimento ou emoção no dicionário.

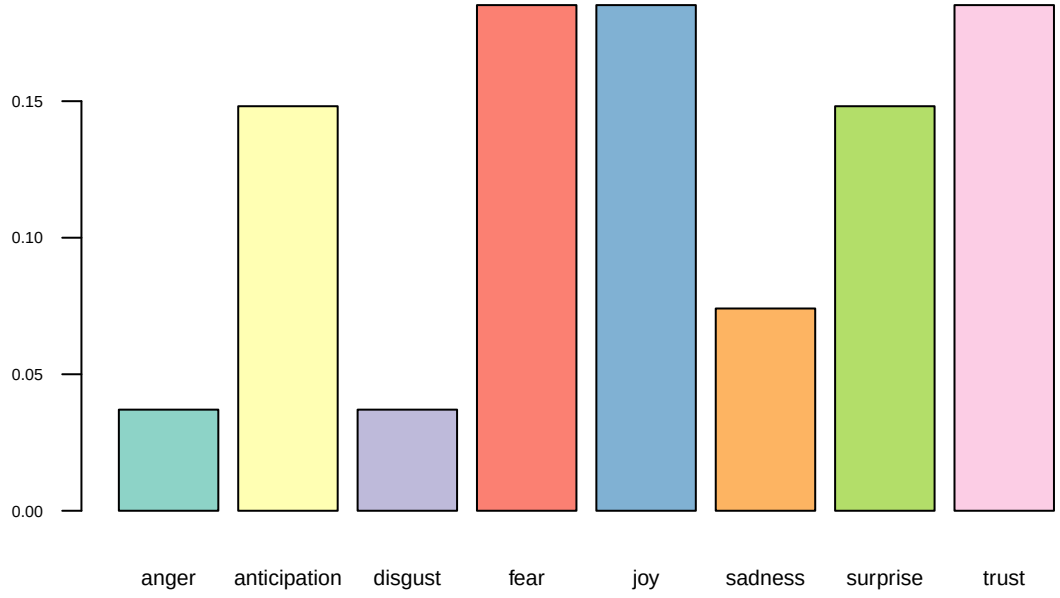
3.5 Análise das emoções em um texto

- Gráfico de barras

Para ver quais as emoções que estão mais presentes no texto, a maneira mais simples é criar um barplot. Para isso, usamos a função barplot() com o resumo das colunas 1 a 8, ou seja, as colunas de raiva (anger), antecipação (antecipation), desgosto (disgust), medo (fear), alegria (joy), tristeza (sadness), surpresa (surprise) e confiança (trust). Os resultados obtidos vêm do processamento da função prop.table() dos resultados das oito colunas com cada uma das palavras da tabela.

```
barplot(
colSums(prop.table(sentimentos_df[, 1:8])),
space = 0.2,
horiz = FALSE,
las = 1,
cex.names = 0.7,
col = brewer.pal(n = 8, name = "Set3"),
main = "Análise de sentimentos das manchetes do portal do Jornal da Paraíba",
sub = "Análise realizada utilizando a técnica de web scraping para a obtenção dos dados",
xlab="emoções", ylab = NULL, cex.axis = 0.5)
```

Análise de sentimentos das manchetes do portal do Jornal da Paraíl



A análise do gráfico indicam que a emoção de medo (fear) e antecipação (antecipation) prevalecem mais do que os demais sentimentos. Mas quais são as palavras usadas no texto que indicam "medo"? Com que frequência ela aparece no?

3.6 Contando o número de palavras com cada emoção.

A fim de realizar uma análise do texto, é muito interessante saber quais são as palavras usadas com mais frequência no texto em relação à sua identificação com cada emoção. Para isso, primeiro temos que criar um objeto de caracteres com todas as palavras que tenham um valor maior que 0 na coluna "medo" (fear).

```
palavras_medo <- texto_palavras[sentimentos_df$fear > 0]
palavras_medo
```

```
## [1] "suspeita"  "infantil"  "caso"      "condenado" "governo"
```

O conteúdo de palavras_medo nos indica que esta lista não diz muito, pois retorna apenas a listagem de palavras sem maiores informações. Para obter a contagem das vezes que cada palavra relacionada à tristeza aparece nos títulos, geramos uma tabela do primeiro conjunto de caracteres com as funções unlist e table, que depois ordenamos em ordem decrescente (se quisermos uma ordem ascendente mudamos TRUE para FALSE); criamos um novo objeto de tipo tabela e imprimimos as primeiras 10 palavras da lista com sua frequência:

```
palavras_medo_ordem <- sort(table(unlist(palavras_medo)), decreasing = TRUE)
head(palavras_medo_ordem, n = 10)
```

```
##
##      caso condenado   governo  infantil  suspeita
##         1         1         1         1         1
```

Se quisermos saber quantas palavras únicas foram relacionadas à medo, basta usar a função length no objeto que agora agrupa as palavras em ordem:

```
length(palavras_medo_ordem)
```

```
## [1] 5
```

Podemos repetir a mesma operação com o resto das emoções ou com aquela que nos interessa, assim como com os sentimentos positivos e negativos.

3.7 Nuvem de emoções

A fim de gerar uma nuvem com as palavras que correspondem a cada emoção em títulos, criaremos primeiro um vetor no qual armazenaremos todas as palavras que, nas colunas que indicamos após o símbolo $, têm um valor maior que 0. É gerado um novo objeto do tipo vetor, que contém um elemento para a lista de cada emoção.

Neste caso, devemos indicar novamente à função que temos caracteres acentuados.

```
nuvem_emocoes_vetor <- c(
paste(texto_palavras[sentimentos_df$sadness> 0], collapse = " "),
paste(texto_palavras[sentimentos_df$joy > 0], collapse = " "),
paste(texto_palavras[sentimentos_df$anger > 0], collapse = " "),
paste(texto_palavras[sentimentos_df$fear > 0], collapse = " "))
```

Uma vez gerado o vetor, deve convertê-lo em caracteres UTF-8 utilizando a função iconv.

```
nuvem_emocoes_vetor <- iconv(nuvem_emocoes_vetor, "latin1", "UTF-8")
```

Agora que temos o vetor, criamos um corpus de palavras com quatro "documentos" para a nuvem:

```
nuvem_corpus <- Corpus(VectorSource(nuvem_emocoes_vetor))
```

Em seguida, transformamos este corpus em uma matriz termo-documento com a função TermDocument-Matrix(). Com isto, agora usamos a função as.matrix() para converter o TDM em uma matriz que, como podemos ver, lista os termos no texto com um valor maior que zero para cada uma das quatro emoções que extraímos aqui. Para ver o início desta informação, use novamente a função head:

```
nuvem_tdm <- TermDocumentMatrix(nuvem_corpus)
nuvem_tdm <- as.matrix(nuvem_tdm)
head(nuvem_tdm)
```

```
##           Docs
## Terms      1 2 3 4
##   caso     1 0 0 1
##   condenado 1 0 0 1
##   infantil  0 1 1 1
##   maioria   0 1 0 0
##   trabalho  0 3 0 0
##   governo   0 0 0 1
```

Agora, atribuímos um nome a cada um dos grupos de palavras ou documentos (Docs) em nossa matriz. Usaremos os termos em português para as colunas que selecionamos para exibir na nuvem.

Mais uma vez, podemos ver a mudança feita ao executar a função head.

```
colnames(nuvem_tdm) <- c('tristeza', 'felicidade', 'raiva', 'confiança')
head(nuvem_tdm)
```

```
##           Docs
## Terms      tristeza felicidade raiva confiança
##   caso             1          0     0         1
##   condenado        1          0     0         1
```

```
##   infantil         0          1     1         1
##   maioria          0          1     0         0
##   trabalho         0          3     0         0
##   governo          0          0     0         1
```

Podemos visualizar a nuvem de palavras que estamos acostumados a ver na mídia ou em estudos académicos. O tamanho e a localização da palavra correspondem à sua maior ou menor ocorrência com valor emocional atribuído no texto. Primeiro, executamos a função set.seed() para que quando reproduzirmos o resultado visual seja o mesmo que o nosso (se não o fizer, será o mesmo, mas as palavras aparecerão em posições diferentes). E, para gerar a nuvem, vamos usar a *função comparison.cloud* do *pacote wordcloud*.

Indicamos o objeto a representar, aqui 'nuvem_tdm', indicamos uma ordem não aleatória das palavras, atribuímos uma cor para cada grupo de palavras e damos tamanhos ao título e à escala geral, e atribuímos um número máximo de termos que serão exibidos.

```
library(quanteda.textplots)
#criando uma DFM com as hashtags
dfmat_texto <- dfm(texto_palavras)
set.seed(132)
textplot_wordcloud(dfmat_texto, max_words = 100, colors = c("green", "red", "orange", "blue"))
```

O resultado da nuvem ficou com poucas palavras, isso foi motivado pelo tamanho do nosso texto ser pequeno, já que estamos utilizando os títulos das manchetes do portal.

3.8 Visualizando a evolução dos sentimentos em um texto

Para complementar a leitura isolada das emoções, estudando a flutuação dos sentimentos positivos e negativos ao longo de um texto, há uma maneira de normalizar e visualizar estas informações. Como a análise da função de extração de sentimento atribui um valor positivo tanto ao sentimento positivo quanto ao negativo, precisamos gerar dados entre um intervalo de -1 para o momento mais negativo e 1 para o mais positivo, e onde 0 é neutro. Para isso, calculamos a valência do texto multiplicando os valores na coluna de valores negativos de nosso data frame com os resultados por -1 e adicionamos o valor na coluna de valores positivos.

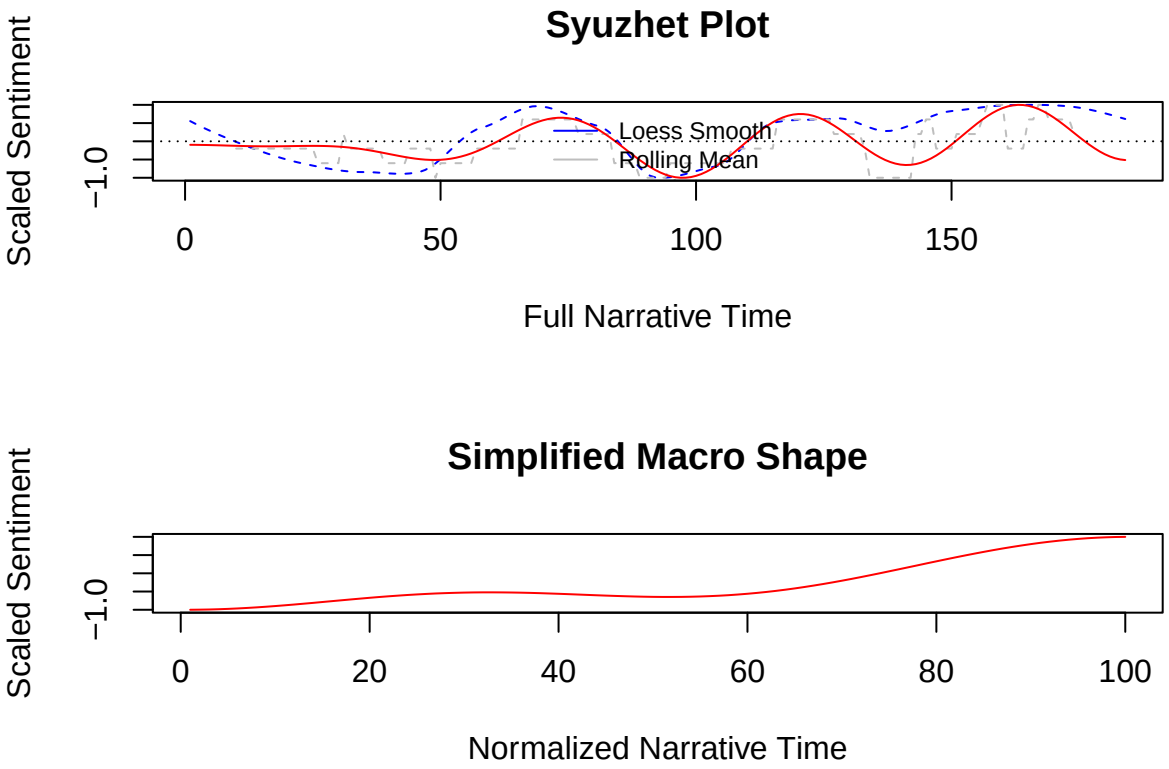
```
sentimentos_valencia <- (sentimentos_df$negative * -1) + sentimentos_df$positive
sentimentos_valencia
```

```
##   [1]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##  [26]  0  0  0  0  0  0  0  0  0 -1  0  0  0  0  2 -1  0  0  0  0  0  0 -1  0  0
##  [51]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2
##  [76]  0  0  0  0  0  0  0  0  0  0 -1  0  0  0  0  0  0  0  0  0  0  0  0 -1  0  0
## [101]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2
## [126]  0  0  0  0  0  0  0  0  0  0 -1  0  0  0  0  0 -1  0  0  0  0  0  0  0  0
## [151]  0  3  0  0  0 -2  0  0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0
## [176]  1  0 -1  0  0  0  0  0  0
```

Finalmente, podemos gerar um gráfico com a função simple_plot() integrada no pacote syuzhet, que nos dará duas imagens diferentes; a primeira, tem todas as medidas que o algoritmo calcula e, a segunda, é uma normalização das mesmas. O eixo horizontal apresenta o texto em 100 fragmentos normalizados e o eixo vertical nos informa sobre a valência do sentimento no texto.

Dependendo das características de seu computador, este gráfico pode levar até 20-30 minutos para ser gerado.

```
simple_plot(sentimentos_valencia)
```



Assim, neste caso, podemos interpretar que o texto análisado varia entre momentos positivos e negativos.

Começa de forma mais positiva, seguido e permanecendo por um momento negativo.

3.9 Salvar seus dados

Para salvar os dados para retornar a eles mais tarde, é possível fazê-lo em um ficheiro de valores separados por vírgula (CSV) com a função write.csv(). Aqui dizemos para salvar o data frame, que contém o resultado das oito emoções e os dois sentimentos de texto em um ficheiro com uma extensão .csv. Além disso, podemos acrescentar a palavra à qual cada linha de resultados corresponde, em uma coluna à esquerda usando a palavra vetor feita no início da análise.

```
write.csv(sentimentos_df, file = "analise_sent_titulos.csv", row.names = texto_palavras)
```

4. Conclusão

Como os dados obtidos através da técnica de raspagem de dados *web scraping* se limitou aos títulos das principais notícias do portal do Jornal da Paraíba, nossas análises se limitaram as análises dos textos obtidos que, por serem pequenos, ocasionaram resultados simples.

Porém, o estudo demonstra que a técnica de *web scraping* é uma ferramenta poderosa para a obtenção de dados oriundos da internet.

Para Bruce e Bruce (2019), um dos maiores desafios da ciência de dados é trabalhar essa torrent de dados brutos, transformá-la para aplicar os conceitos estatístico e assim obter informações acionável.

5. Trabalhos futuros

Como propostas de trabalhos futuros, podemos destacar a coleta, limpesa e a nálise de dados de outros sites através da técnica de *web scraping* bem como explorar outras ferramentas como utilizando o pacote *Rselenium* do R.

Referências

BRUCE, Peter; BRUCE, Andrew. Estatística Prática para Cientistas de Dados: 50 conceitos essenciais. Rio de Janeiro: Alta Books, 2019. 300 p.

LIMA, Acervo. Web Scraping usando a linguagem R. 2022. Disponível em: https://acervolima.com/web-scraping-usando-a-lingugem-r/. Acesso em: 01 abr. 2023.

MITCHELL, Ryan. Web Scraping com Python: coletando mais dados na web moderna. 2. ed. São Paulo: Oreille Novatec, 2019.

MOREIRA, Davi; ROCABADO, Mônica. Texto como Dado para Ciências Sociais: guia prático com aplicações. guia prático com aplicações. 2022. Disponível em: https://bookdown.org/davi_moreira/txt4cs/. Acesso em: 14 abr. 2023.

SILVA, Réulison. Web Scraping com Python: uma maneira de extrair dados da web. Uma maneira de extrair dados da web. 2021. Disponível em: https://reulison.com.br/web-scraping-python/. Acesso em: 28 jan. 2023.

Anexo

https://github.com/carlostvcb-ux/Web_Scraping_Inicio