Introdução

Este tutorial é baseado nos trabalhos de José Robeto Ayala Solares [1] e de Kevin Markham [2]. O objetivo deste material é apresentar os comando básicos do R para coleta de dados em páginas da WEB. Aqui será analisada um artigo com baseado no artigo Trump's Lies[3]. O interesse desse trabalho é obter

- a data da mentira
- a mentira
- a explicação do porque é mentira
- a URL para um artigo que sustenta a explicação.

Leitura de página web com R

Para ler uma página da web com o R é possível com o uso do pacote `rvest` criado por Hadley Wickham. Este pacote é inspirado em bibliotecas como BeautifulSoup, para facilitar a coleta de dados de páginas web em html. A primeira função importante é a `read_html`, a qual retorna um documento XML que contem todas as informações sobe a página web.

Selecionando os Registros

Os registros com as informações procurados apresentam a seguinte estrutura em código HTML:

```
<span class="short-desc">
    <strong> DATA </strong>
    MENTIRA
    <span class="short-truth">
    <a  href="URL"> EXPLICACAO </a>
    </span>
</span>
```

Portanto, para coletar todas as mentiras, é preciso identificar todas as tags que pertencem à classe = "short-desc". A função projetada para isso é `html_nodes`, requer o documento XML retornado pela função `read_html`.

[1] https://towardsdatascience.com/web-scraping-tutorial-in-r-5e71fd107f32
[2] https://www.dataschool.io/python-web-scraping-of-president-trumps-lies/
[3] https://www.nytimes.com/interactive/2017/06/23/opinion/trumps-lies.html

```
> resultado <- html_nodes(webpage,".short-desc")
> resultado
{xml_nodeset (180)}
 [1] <span class="short-desc"><strong>Jan. 21 </strong>I wasn't ...
 [2] <span class="short-desc"><strong>Jan. 21 </strong>A reporter for  ...
```

Isso retorna uma lista com 180 nós XML que contêm as informações para cada um dos 180 está na página da web.

Extraindo a data

Vamos começar de forma simples e focar na extração de todos os detalhes necessários da primeira mentira. Podemos então estender isso para todos os outros facilmente. Lembre-se de que a estrutura geral de um único registro é:

```
<span class="short-desc">
    <strong> DATA </strong>
    MENTIRA
    <span class="short-truth">
    <a  href="URL"> EXPLICACAO </a>
    </span>
</span>
```

Observe que a data está incorporada na tag . Para selecioná-lo, podemos usar a função html_nodes () usando o seletor "strong".

```
first_result <- results[1]
first_result %>% html_nodes("strong")
#> {xml_nodeset (1)}
#> [1] <strong>Jan. 21 </strong>
```

Em seguida, precisamos usar a função `html_text` para extrair apenas o texto, com o argumento trim ativo para aparar espaços iniciais e finais. Finalmente, usamos o pacote stringr para adicionar o ano à data extraída.

```
 data <- html_nodes(resultado[i],"strong")
 data <- html_text(data,trim = TRUE)
library(stringr)
data <- str_c(data,",2017")
#> [1] "Jan. 21, 2017"
```

Extraindo a mentira

Para selecionar a mentira, precisamos fazer uso da função `xml_contents` que faz parte do pacote xml2 (este pacote é requerido pelo pacote rvest, então não é necessário carregá-lo). A função retorna uma lista com os nós que fazem parte do first_result.

```
xml_contents(first_result)
#> {xml_nodeset (3)}
```

Observe que há um par extra de aspas () ao redor da mentira. Para se livrar deles, simplesmente usamos a função `str_sub` do pacote stringr para selecionar apenas a mentira.

```
lie <- xml_contents(first_result)[2] %>% html_text(trim = TRUE)
str_sub(lie, 2, -2)
#> [1] "I wasn't a fan of Iraq. I didn't want to go into Iraq."
```

Observe que há um par extra de aspas () ao redor da mentira. Para se livrar deles, simplesmente usamos a função `str_sub` do pacote stringr para selecionar apenas a mentira.

Extraindo a explicação

Espero que agora não seja muito complicado ver que para extrair a explicação basta selecionar o texto dentro da tag que pertence a class = ". Short-truth". Isso extrairá o texto junto com os parênteses de abertura e fechamento, mas podemos facilmente nos livrar deles.

```
explicacao <- html_nodes(resultado[i],".short-truth")
explicacao <- html_text(explicacao, trim = TRUE)
explicacao <- str_sub(explicacao, 2, -2)
```

Extraindo a URL

Finalmente, para obter o URL, observe que este é um atributo dentro da tag <a>. Nós simplesmente selecionamos este nó com a função `html_nodes`, e então selecionamos o atributo href com a função `html_attr`.

```
caminho <- html_nodes(resultado[i],"a")
caminho <- html_attr(caminho,"href")
```

Formando o dataset

Encontramos uma maneira de extrair cada uma das 4 partes do primeiro registro. Podemos estender esse processo para todo o resto usando um loop for. No final, queremos ter um quadro de dados com 180 linhas (uma para cada registro) e 4 colunas (para manter a data, a mentira, a explicação e a URL). Uma maneira de fazer isso é criar um quadro de dados vazio e simplesmente adicionar uma nova linha à medida que cada novo registro é processado.

```
for (i in seq_along(resultado)) {
    data <- html_nodes(resultado[i],"strong")
    data <- html_text(data,trim = TRUE)
    data <- str_c(data,",2017")
    mentira <- xml_contents(resultado[i])[2]
    mentira <- html_text(mentira,trim = TRUE)
    mentira <- str_sub(mentira, 2, -2)
    explicacao <- html_nodes(resultado[i],".short-truth")
    explicacao <- html_text(explicacao, trim = TRUE)
    explicacao <- str_sub(explicacao, 2, -2)
    caminho <- html_nodes(resultado[i],"a")
    caminho <- html_attr(caminho,"href")
    registro <- rbind(registro,
                      data.frame(data, mentira, explicacao, caminho))
  }
```

Exportando o dataframe para um arquivo csv

Se você deseja exportar seu conjunto de dados, você pode usar a função write.csv () que vem por padrão com R.

```
write.csv(df, "trump_mente.csv")
```

Da mesma forma, para recuperar seu conjunto de dados, você pode usar a função padrão `read.csv` .