BRASÍLIA

2021

PEDRO VITOR VALENÇA MIZUNO

RASPAGEM, INFOGRAFIA E TUTORIAIS
TRABALHO FINAL DA DISCIPLINA

Professores: Benedito Medeiros Neto e Edison Ishikawa

Brasília, 19 de maio de 2021
Versão 1.0
Universidade de Brasília

Sumário

1. Cliente / Área Solicitante

Alunos da disciplina Jornalismo de Dados, juntamente com Larissa de Jesus Silva, Virgínia Barros, aluna de mestrado da Universidade do Maranhão (UFMA), Daumildo Alves de Oliveira Junior e Wanessa Alves Pereira, alunos do curso de Jornalismo da Universidade de Brasília (UnB).

2. Descrição da Demanda

A fim de continuar o trabalho dos alunos de semestres anteriores, Alex Nascimento Souza e Guilherme da Silva Fontes Lopes, foi implementado um sistema de raspagem no Twitter capaz de varrer uma conta de um usuário em busca de *tweets* que citam uma palavra-chave pedida por aquele que utiliza o programa, de forma que seja possível obter de forma simples informações acerca de um tema.

Como o trabalho do jornalista envolve também a apresentação das informações que ele obtém, foi criado um programa pelo qual se possa obter infográficos baseados em dados que um usuário insira, permitindo diversos formatos para apresentá-los.

E, em conjunto com o aluno Pedro Chaves, foram feitos tutoriais acerca de ferramentas comumente utilizadas por alunos e jornalistas que têm interesse na área de jornalismo de dados. Assim, estudantes de jornalismo de dados poderão obter informações acerca das ferramentas por meio dos tutoriais e ver um projeto que aplica algumas delas.

3. Patrocinadores

Nome		Cargo
Benedito Medeiros Neto		Professor Adjunto
Telefone	Endereço Eletrônico	Lotação
(61) 99968-0789	beneditomedeirosneto@gmail.com	Departamento de Ciência da Computação

Nome		Cargo
Edison Ishikawa		Professor Adjunto
Telefone	Endereço Eletrônico	Lotação
(61) 3107-2270	ishikawa@unb.br	Departamento de Ciência da Computação

4. Gerentes do Projeto

Nome		Cargo
Benedito Medeiros Neto		Professor Adjunto
Telefone	Endereço Eletrônico	Lotação
(61) 99968-0789	beneditomedeirosneto@gmail.com	Departamento de Ciência da Computação

Nome		Cargo
Edison Ishikawa		Professor Adjunto
Telefone	Endereço Eletrônico	Lotação
(61) 3107-2270	ishikawa@unb.br	Departamento de Ciência da Computação

5. Justificativa

Alunos e profissionais de jornalismo precisam obter informações de forma eficaz e rápida. Para isso, um método de raspagem é uma boa forma de ilustrar como é possível criar meios que permitam atingir esse objetivo.

Além de obter dados eficientemente, alunos e profissionais de jornalismo necessitam que os dados que obtenham sejam apresentados para uma fácil leitura. Assim, utilizar um sistema que cria infográficos dá essa capacidade ao usuário.

Por fim, por meio dos tutoriais, os alunos poderão entender o básico das ferramentas utilizadas por jornalistas que trabalham com dados, de forma que saibam como, quando e por que utilizá-las.

6. Meta do Projeto

Os objetivos deste projeto foram:

1. desenvolver um sistema que execute raspagens de dados na conta de Twitter de um usuário, retornando *tweets* que contenham as palavras chaves solicitadas (baseado no sistema do Twitter de 2021);
2. desenvolver um sistema que crie infográficos de acordo com os dados inseridos pelo cliente;
3. criar tutoriais que ensinam o básico das ferramentas utilizadas por jornalistas de dados.

7. Produto ou Serviço

Para o desenvolvimento do produto foram utilizadas as ferramentas:

- GitHub: onde o código foi armazenado;
- Python versão 3.8: para criar os sistemas de raspagem e infografia;
- vsCode: o editor de texto;
- snscrape: biblioteca utilizada para realizar raspagem no Twitter;
- Numpy, pandas e Matplotlib: bibliotecas utilizadas para gerar os gráficos;
- Django: o framework utilizado para a plataforma onde estão os tutoriais.

8. Objetivo Estratégico

Foram feitos sistemas de raspagem de dados e infografia e tutoriais de ferramentas voltadas para a manipulação de dados. Por meio destes, alunos

e profissionais de jornalismo podem obter conhecimentos sobre a área de jornalismo de dados relevantes no ano de 2021.

9. Resultados

Houveram alguns empecilhos quanto à hospedagem das ferramentas na WikiJour. Primeiramente, houveram problemas de compatibilidade quanto aos estilos utilizados no portal dos alunos dos semestres anteriores e com o Django CMS utilizado na wiki atual. Além disso, foi descoberto o fato de que os servidores do Twitter bloqueiam IPs provenientes da AWS (Amazon Web Services), de forma que seja impossível utilizar a ferramenta de raspagem hospedada em provedores gratuitos ou de baixo custo. Assim, foi optado atualizar o portal criado pelos alunos de semestres anteriores, adicionando um tutorial de como utilizar o código de raspagem localmente e a ferramenta de infografia. O novo portal atualizado pode ser acessado pelo seguinte link: http://portaljornalismodados.herokuapp.com/

Quanto aos tutoriais, neste documento é possível lê-los por completo na seção 9.3 e na WikiJour.

9.1) **Raspagem de Dados:**

Seguindo a proposta apresentada, foi realizada uma nova versão do sistema de raspagem feita pelos alunos Alex Nascimento Souza e Guilherme da Silva Fontes Lopes. Devido a uma atualização da API do Twitter, a biblioteca utilizada para a raspagem (twitterscraper) não funcionava mais. Então foi adotada a biblioteca snscrape (https://github.com/JustAnotherArchivist/snscrape), que possibilitou a volta do funcionamento do sistema.

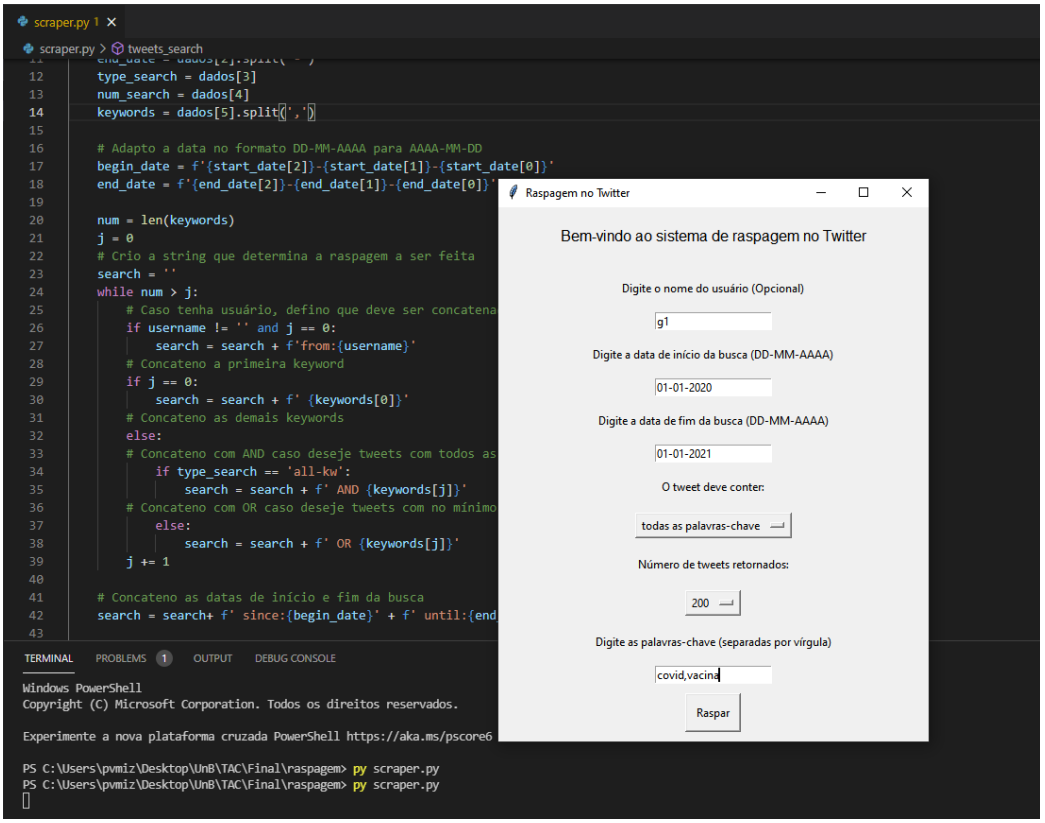
Além da mudança da biblioteca, foram feitas alterações na forma de realizar a pesquisa. Mantiveram-se as datas de início e fim, a conta de usuário (não obrigatória) e a listagem das palavras-chave. Entretanto, adicionou-se a possibilidade de escolher uma pesquisa que busque por

tweets que contenham todas ou apenas algumas palavras-chave e uma forma de limitar o número de tweets retornados.

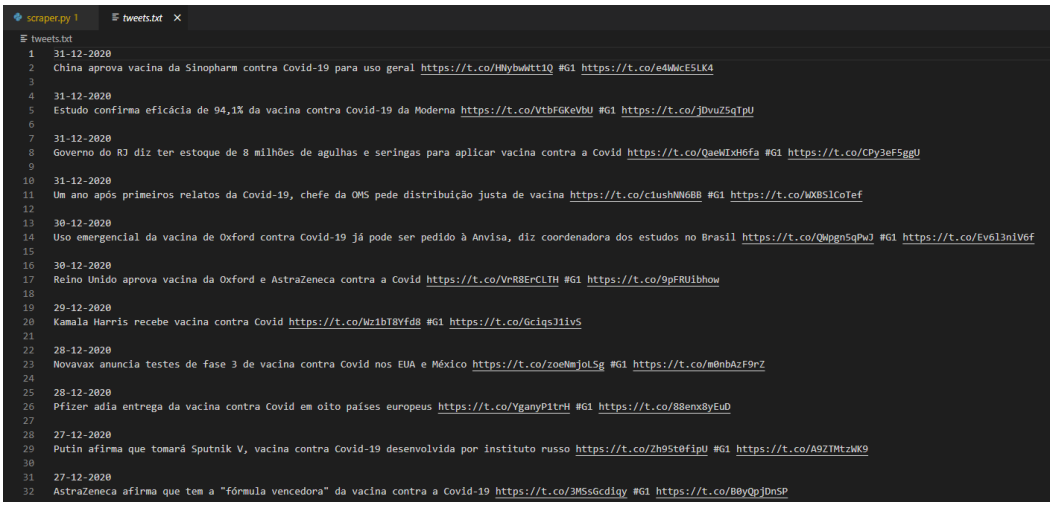
Como explicado na seção 9, não é possível acoplar a ferramenta ao portal. Assim, foi decidido criar tutoriais por texto e vídeo que detalhassem como utilizar o código da raspagem localmente. O tutorial, então, foi disponibilizado na área “Busca Twitter” do portal.

Seguindo o tutorial apresentado, é possível executar o código de raspagem, que retorna uma GUI para utilizar a ferramenta, como apresentado na figura abaixo. Para o exemplo de sua utilização foram especificados os valores de data de início e fim, a conta a ser raspada (o Twitter do site de notícias g1), que devem ser retornados no máximo

200 tweets com todas as palavras-chave e, por fim, as palavras-chave “covid” e “vacina”.



Após ser realizada a raspagem, é gerado no mesmo diretório onde o código está armazenado um arquivo txt contendo os tweets raspados.

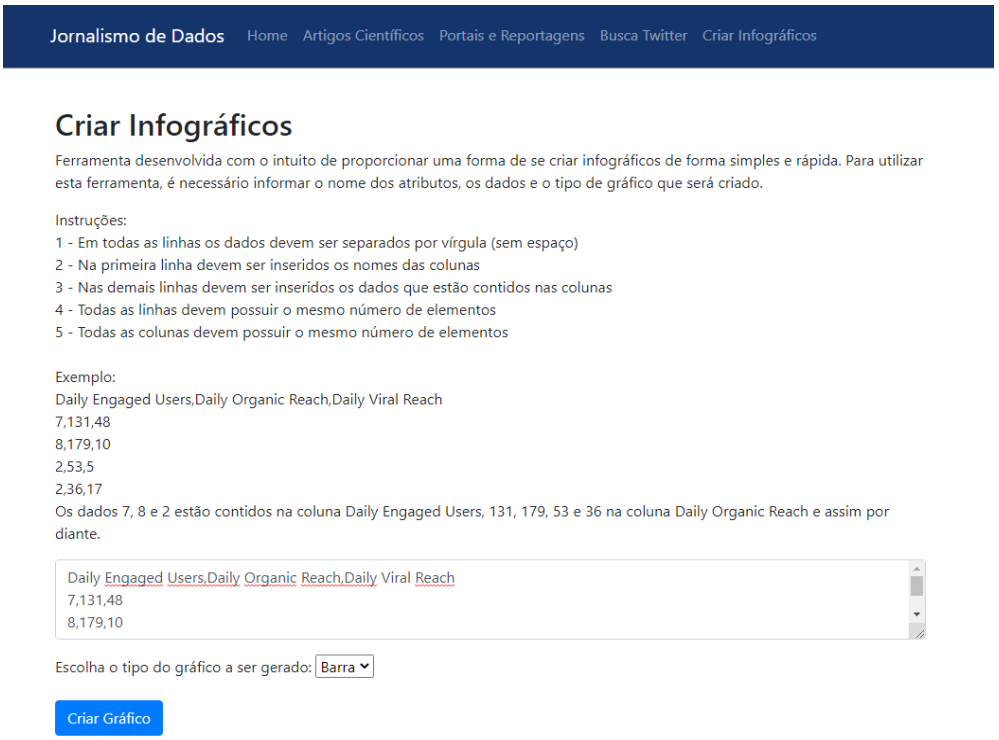


9.2) Infografia:

Para criar a ferramenta que cria infográficos, foram necessárias três bibliotecas Python: Numpy, pandas e Matplotlib. O Numpy e o pandas

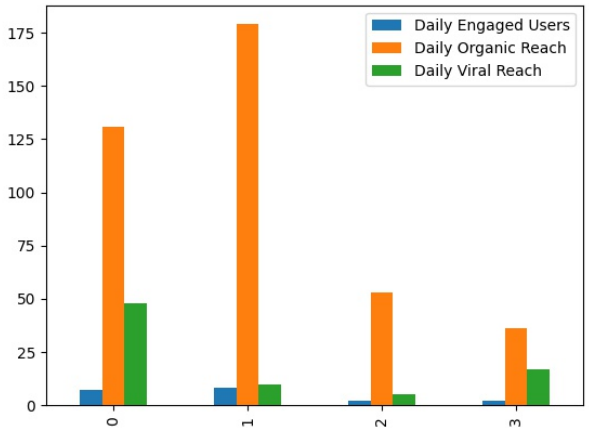
foram utilizados para estruturar os dados lidos e, por meio deles, o Matplotlib foi capaz de gerar os infográficos.

Para a utilização correta do gerador de infográficos, foram criadas regras, presentes na imagem abaixo. Na caixa de texto são digitados os dados, em que na primeira linha estão presentes os nomes das colunas e nas linhas seguintes são listados os dados das colunas. Por fim, é possível escolher o tipo do dado gerado, podendo ser: gráfico em barra, área, pizza ou linha.



Assim, após ser gerado o gráfico, é apresentada uma página que contém a imagem no infográfico.

Gráfico Resultante



9.3) Tutoriais de ferramentas jornalismo de dados:

A seguir são apresentados os tutoriais das ferramentas.

- **Numpy:**

Numpy é uma biblioteca muito utilizada para computação científica e análise de dados utilizando Python. Por meio dela é possível trabalhar com arrays, álgebra linear, matrizes e mais.

No Python já existe uma estrutura de comportamento similar ao array: a lista. No entanto, os arrays de Numpy apresentam um processamento muito mais rápido que a lista, sendo assim, uma escolha muito interessante para sistemas que exijam um bom desempenho.

Instalação do Numpy:

https://numpy.org/install/

Obs: os exemplos a seguir foram realizados no shell do Python3.

Para utilizar esse pacote, devemos importar por meio de:

```
>>> import numpy as np
```

A peça fundamental do numpy são os N-dimensional array, também chamados de ndarrays, que permite você realizar operações de forma rápida

e flexível em conjuntos de dados. Dados esses que podem ser inteiros, floats, booleanos, strings, números complexos e muitos outros.
Podemos utilizar conjuntos de dados como listas para criar ndarrays. No seguinte código, criamos um array de 4 elementos, sendo eles 123, 454, 434 e 12.

```
>>> np.array([10,12,5,3])
array([10, 12,  5,  3])
```

Para armazenar o array em uma variável, basta atribuí-lo:

```
>>> a = np.array([[1,4],[5,6]])
```

Além disso, como é autoexplicativo no nome, podemos ter múltiplas dimensões nesse array. Por exemplo:
2 dimensões:

```
>>> np.array([[2,5,6],[9,7,8]])
array([[2, 5, 6],
       [9, 7, 8]])
```

3 dimensões:

```
>>> np.array([[[123, 12], [156, 346]], [[87, 45], [12, 54]], [[5, 4], [9, 7]]])
array([[[123,  12],
        [156, 346]],

       [[ 87,  45],
        [ 12,  54]],

       [[  5,   4],
        [  9,   7]]])
```

Também podemos iniciar um array sem necessitar de uma lista. Por exemplo:
Array de zeros com 2 linhas e 3 colunas:

```
>>> np.zeros((2, 3))
array([[0., 0., 0.],
       [0., 0., 0.]])
```

Array de uns com 4 linhas e 2 colunas:

```
>>> np.ones((4, 2))
array([[1., 1.],
       [1., 1.],
       [1., 1.],
       [1., 1.]])
```

Array de zeros com 3 linhas, 4 colunas e 2 repetições disso (3 dimensões):

```
>>> np.zeros((2, 3, 4))
array([[[0., 0., 0., 0.],
        [0., 0., 0., 0.],
        [0., 0., 0., 0.]],

       [[0., 0., 0., 0.],
        [0., 0., 0., 0.],
        [0., 0., 0., 0.]]])
```

Array com 10 elementos, de 0 a 9:

```
>>> np.arange(10)
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Array de 4 números aleatórios:

```
>>> np.random.randn(6)
array([ 1.25115478, -0.62599618, -0.6802084 ,  1.15048219, -2.42585027,
       -0.19863413])
```

É possível realizar operações aritméticas com arrays, como adição, subtração, multiplicação e muitas outras:

```
>>> a = np.array([[1,4],[5,6]])
>>> b = np.array([[3,1],[6,9]])
>>> a + b
array([[ 4,  5],
       [11, 15]])
>>> a - b
array([[-2,  3],
       [-1, -3]])
>>> a * b
array([[ 3,  4],
       [30, 54]])
>>> a / b
array([[0.33333333, 4.        ],
       [0.83333333, 0.6666667]])
>>> a ** b
array([[       1,        4],
       [   15625, 10077696]])
>>> 1 / a
array([[1.        , 0.25      ],
       [0.2       , 0.16666667]])
```

Note que as operações realizadas não são operações de matrizes de álgebra linear.

De forma similar a listas em Python, podemos acessar dados de arrays por meio de índices e slices:

```
>>> a = np.array([[1,4],[5,6]])
>>> a[1]
array([5, 6])
>>> a = np.arange(10)
>>> a[3]
3
>>> a[2:6]
array([2, 3, 4, 5])
```

No Numpy existem diversos métodos matemáticos e estatísticos. No seguinte exemplo foi feita a média, o desvio padrão e a soma de a:

```
>>> a = np.arange(10)
>>> a.mean()
4.5
>>> a.std()
2.8722813232690143
>>> a.sum()
45
```

Também é possível utilizar cálculos de álgebra linear. No exemplo a seguir foi realizada a multiplicação de matrizes de álgebra linear:

```
>>> a = np.array([[1,4],[5,6]])
>>> b = np.array([[3,1],[6,9]])
>>> np.dot(a,b)
array([[27, 37],
       [51, 59]])
```

Para guardar um array gerado para futuros usos, é possível armazená-lo em um arquivo npy. Também podemos salvar arrays por meio de:

np.save('nome_array', a)

Além disso, ao invés de colocarmos manualmente os dados em linhas de código, podemos obtê-los por meio de leitura de arquivos. Dado um arquivo texto, podemos ler os dados e armazenar em um array para, então, trabalhar com eles.

```
>>> a
array([[1, 4],
       [5, 6]])
>>> a = np.array([[1,4],[5,6]])
>>> np.save('nome_array', a)
>>> b = np.load('nome_array.npy')
>>> b
array([[1, 4],
       [5, 6]])
```

Lista de exercícios de Numpy:
https://www.w3resource.com/python-exercises/numpy/index.php

- **pandas:**

O pandas é uma biblioteca de Python montada com base no Numpy que contém estruturas de dados de alto-nível e ferramentas que permitem a análise de dados de forma fácil e rápida. Para tal, o Pandas oferece estruturas e operações para manipular tabelas.
Instalação do Pandas:
https://pandas.pydata.org/pandas-docs/stable/getting_started/install.html
Para importar a biblioteca é necessário ter a linha:

```
>>> import pandas as pd
```

As duas estruturas de dados mais relevantes dessa biblioteca são: Series e DataFrame.
Series é um array de uma dimensão que contém qualquer tipo de dado existente no Numpy e que cada elemento é localizado por um índice, de forma similar as listas do Python. No exemplo é criada uma Series de 4 elementos. Podemos ver os valores por meio de values e como o índice funciona por meio de index.

```
>>> b = pd.Series([5,87,4,-3])
>>> b.values
array([ 5, 87,  4, -3])
>>> b.index
RangeIndex(start=0, stop=4, step=1)
```

É possível também definir os índices da series. No exemplo estamos criando a series com índices a, b, c e d.

```
>>> b = pd.Series([5, 87, 4, -3], index=['a', 'b', 'c', 'd'])
>>> b
a     5
b    87
c     4
d    -3
dtype: int64
```

Podemos realizar diversas operações, como filtragens, operações matemáticas e operações lógicas. Tais operações serão detalhadas na seção a seguir.

O dataframe pode ser interpretado como uma tabela contendo um conjunto de colunas, em que cada uma pode ter um tipo de dados diferente (string, inteiro, float, etc). O dataframe possui tanto um índice de coluna quanto de linha.

Assim, como em Series, os valores retornados de dataframe é um ndarray, no entanto, de duas dimensões.

Existem diversas formas de criar um dataframe, mas a mais comum é por meio de um dicionário de listas ou arrays de NumPy, todos com o mesmo tamanho.

```
>>> data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'], 'year
': [2000, 2001, 2002, 2001, 2002], 'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}
>>> tabela = pd.DataFrame(data)
>>> tabela
    state  year  pop
0    Ohio  2000  1.5
1    Ohio  2001  1.7
2    Ohio  2002  3.6
3  Nevada  2001  2.4
4  Nevada  2002  2.9
```

Podemos obter dados de uma coluna acessando por meio do nome das colunas (onde se obtém os dados da coluna) e pelo índice (onde se obtém os dados da linha):

```
>>> data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'], 'year': [2000, 2001, 2002, 2001,
2002], 'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}
>>> frame = pd.DataFrame(data)
>>> frame['state']
0      Ohio
1      Ohio
2      Ohio
3    Nevada
4    Nevada
Name: state, dtype: object
>>> frame['state'][3]
'Nevada'
```

Podemos atribuir valores a colunas por meio de:

```
>>> frame
    state  year  pop
0    Ohio  2000  1.5
1    Ohio  2001  1.7
2    Ohio  2002  3.6
3  Nevada  2001  2.4
4  Nevada  2002  2.9
>>> frame.loc[3, 'pop'] = 5
>>> frame
    state  year  pop
0    Ohio  2000  1.5
1    Ohio  2001  1.7
2    Ohio  2002  3.6
3  Nevada  2001  5.0
4  Nevada  2002  2.9
```

Outra forma comum de criar dataframes é por meio de dicionários aninhados, em que as chaves dos dicionários externos se tornam os nomes das colunas e as chaves dos dicionários internos se tornam os índices:

```
>>> pop = {'Nevada': {2001: 2.4, 2002: 2.9}, 'Ohio': {2000: 1.5, 2001: 1.7, 2002: 3.6}}
>>> frame = pd.DataFrame(pop)
>>> frame
      Nevada  Ohio
2001     2.4   1.7
2002     2.9   3.6
2000     NaN   1.5
```

Note que como não especificamos um índice 2000 para ‘Nevada’, ao imprimir o dataframe o valor nesse setor é NaN (Not a Number).

Entre as ferramentas mais importantes para mexer com series e dataframes, temos:

Reindex: permite criar um novo objeto com dados organizados por um novo índice.

```
>>> obj = pd.Series([4.5, 7.2, -5.3, 3.6], index=['d', 'b', 'a', 'c'])
>>> obj2 = obj.reindex(['a', 'b', 'c', 'd', 'e'])
>>> obj2
a   -5.3
b    7.2
c    3.6
d    4.5
e    NaN
dtype: float64
>>> obj = pd.Series([4.5, 7.2, -5.3, 3.6], index=['d', 'b', 'a', 'c'])
>>> obj
d    4.5
b    7.2
a   -5.3
c    3.6
dtype: float64
>>> obj2 = obj.reindex(['a', 'b', 'c', 'd', 'e'])
>>> obj2
a   -5.3
b    7.2
c    3.6
d    4.5
e    NaN
dtype: float64
```

Drop: permite criar um novo objeto sem índices especificados na chamada.

```
>>> obj = pd.Series(np.arange(5.), index=['a', 'b', 'c', 'd', 'e'])
>>> obj
a    0.0
b    1.0
c    2.0
d    3.0
e    4.0
dtype: float64
>>> obj2
a    0.0
b    1.0
d    3.0
e    4.0
dtype: float64
>>> obj = pd.Series(np.arange(5.), index=['a', 'b', 'c', 'd', 'e'])
>>> obj
a    0.0
b    1.0
c    2.0
d    3.0
e    4.0
dtype: float64
>>> obj2 = obj.drop('c')
>>> obj2
a    0.0
b    1.0
d    3.0
e    4.0
dtype: float64
```

Indexação e filtragem: podemos acessar os elementos por meio dos índices e fazer operações que filtram os dados.

```
>>> data = pd.DataFrame(np.arange(16).reshape((4, 4)), index=['Ohio', 'Colorado', 'Utah', 'New York'
], columns=['one', 'two', 'three', 'four'])
>>> data[['three', 'one']]
          three  one
Ohio          2    0
Colorado      6    4
Utah         10    8
New York     14   12
>>> data[data['three'] > 5]
          one  two  three  four
Colorado    4    5      6     7
Utah        8    9     10    11
New York   12   13     14    15
```

Operações aritméticas: podemos realizar operações aritméticas em series e dataframes. O fill_value=0 determina que caso o tamanho dos dataframes sejam diferentes, aqueles valores que o menor não tem devem ser preenchidos com 0. Caso não fosse explicitado, seriam valores NaN.

```
>>> df1 = pd.DataFrame(np.arange(12.).reshape((3, 4)), columns=list('abcd'))
>>> df2 = pd.DataFrame(np.arange(20.).reshape((4, 5)), columns=list('abcde'))
>>> df1.add(df2, fill_value=0)
      a     b     c     d     e
0   0.0   2.0   4.0   6.0   4.0
1   9.0  11.0  13.0  15.0   9.0
2  18.0  20.0  22.0  24.0  14.0
3  15.0  16.0  17.0  18.0  19.0
```

Podemos também salvar dataframes e series em arquivos CSV com o seguinte código:

```
>>> obj = pd.Series(np.arange(5.), index=['a', 'b', 'c', 'd', 'e'])
>>> obj.to_csv('df.csv', index=False)
```

Outra ferramenta importante de pandas é a leitura de arquivos de tabelas, como o csv e excel. Com ela podemos armazenar os valores lidos em um dataframe, de forma que seja fácil transferir dados de um arquivo de planilhas para um programa Python.

```
>>> frame = pd.read_csv('nome_arquivo.csv')
```

```
>>> frame = pd.read_excel('nome_arquivo.xlsx')
```

Lista de exercícios de pandas:
https://www.w3resource.com/python-exercises/pandas/index.php

- **Matplotlib:**

Matplotlib é uma biblioteca de Python muito utilizada para gerar gráficos, desde linhas, histogramas, barras, gráficos em pizza e muitos outros tipos mais complexos. No seguinte link são exemplificados os gráficos que a biblioteca é capaz de gerar.
https://matplotlib.org/stable/tutorials/introductory/sample_plots.html
Instalação do Matplotlib:
https://matplotlib.org/stable/users/installing.html
Para importar a biblioteca ao seu programa, é necessário incluir a linha:

```
>>> import matplotlib.pyplot as plt
```

Os gráficos do matplotlib residem no objeto Figure. Para criá-lo, basta usar a seguinte linha:

```
>>> fig = plt.figure()
```

Em uma figura é possível existir vários subgráficos. Para tal, utilizamos o a função add_subplot, em que os dois primeiros argumentos definem o tamanho x e y do grid e o terceiro mostra sobre qual subplot se está definindo.

```
>>> graf1 = fig.add_subplot(2, 2, 1)
>>> graf2 = fig.add_subplot(2, 2, 2)
```

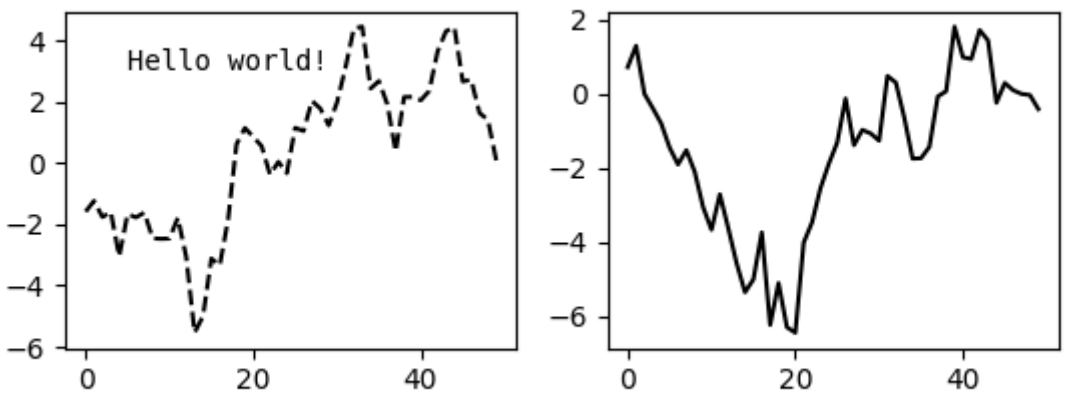
É possível estilizar seus gráficos. Por exemplo, nas seguintes linhas nós criamos gráficos de números aleatórias, em que um tem linha pontilhada e outro tem linha contínua. Além disso, no gráfico 1 decidimos adicionar uma mensagem ‘Hello world!’ no quadrante 5,3.

```
>>> graf1.plot(np.random.randn(50).cumsum(), 'k--')
[<matplotlib.lines.Line2D object at 0x7fa711994c70>]
>>> graf2.plot(np.random.randn(50).cumsum(), 'k-')
[<matplotlib.lines.Line2D object at 0x7fa716886490>]
>>> graf1.text(5, 3, 'Hello world!', family='monospace', fontsize=10)
Text(5, 3, 'Hello world!')
```

Por fim, podemos gerar o gráfico ao salvar a figura com o seguinte comando:

```
>>> fig.savefig('grafico1.png')
```

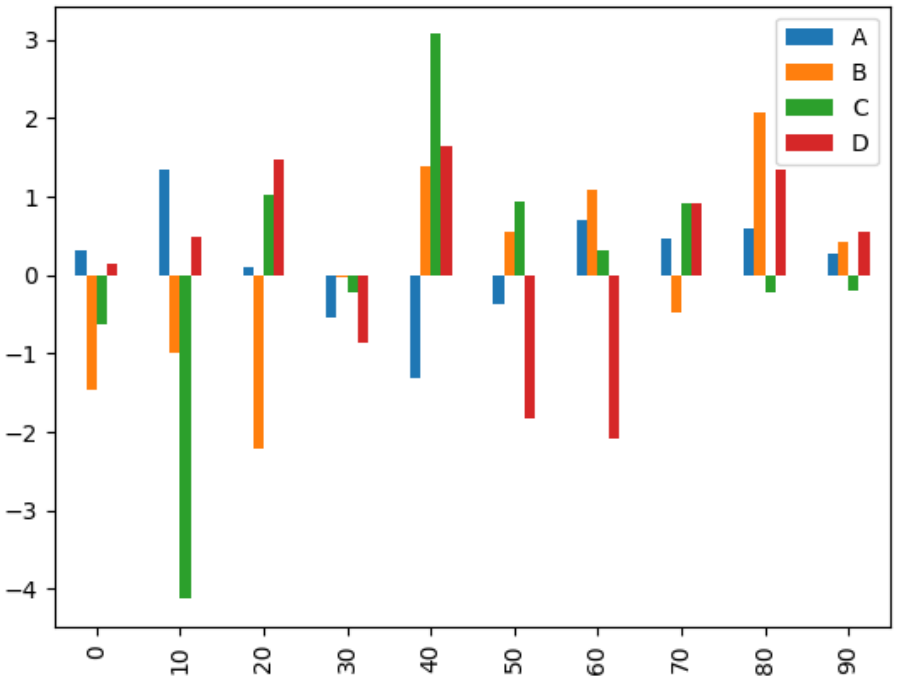
Após o comando, é gerada a figura com os dois gráficos:



Trabalhando em conjunto com o pandas, é possível gerar gráficos a partir de series dataframes. No exemplo a seguir, criamos um dataframe de números aleatórios de 10 linhas e 4 colunas. Além disso, o índice é definido de 0 a 100, mas com “pulos” de 10. Por fim, é utilizada a função plot para gerar o gráfico em barra no objeto fig e a figura é salva.

```
>>> df = pd.DataFrame(np.random.randn(10, 4), columns=['A', 'B', 'C', 'D'], index=np.arange(0, 100, 10))
>>> fig = df.plot(kind = 'bar').get_figure()
>>> fig.savefig('grafico2.png')
```

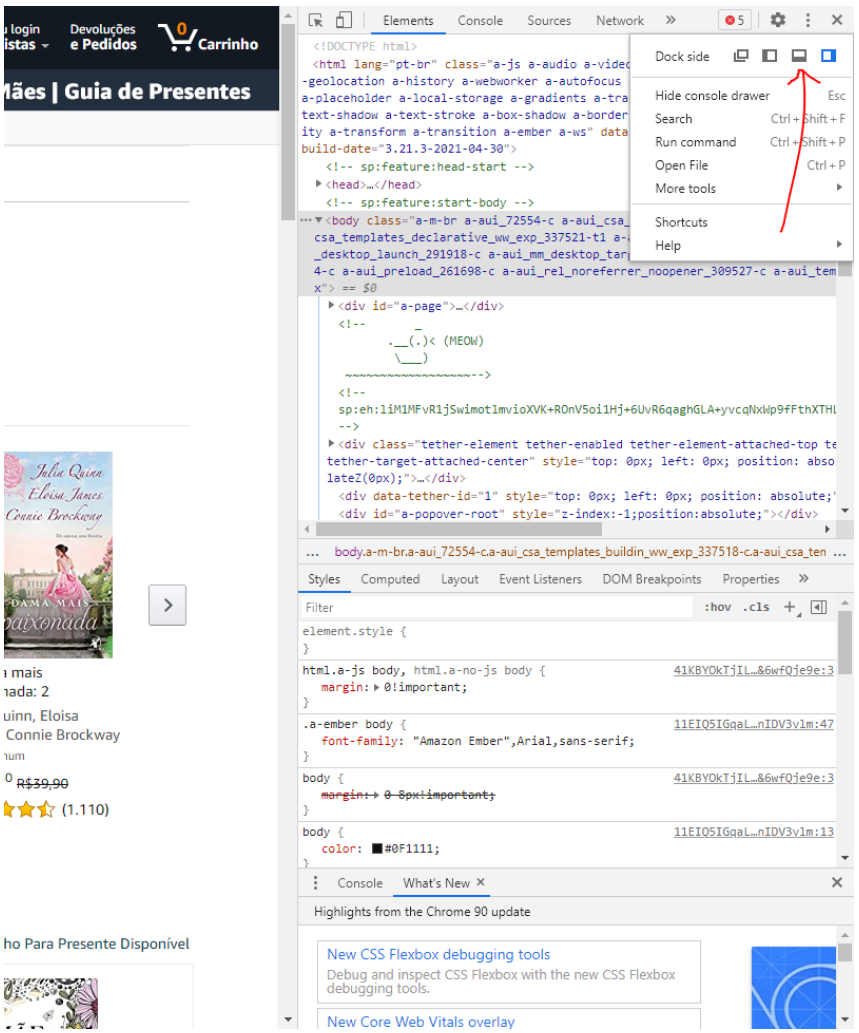
O gráfico gerado foi o seguinte:



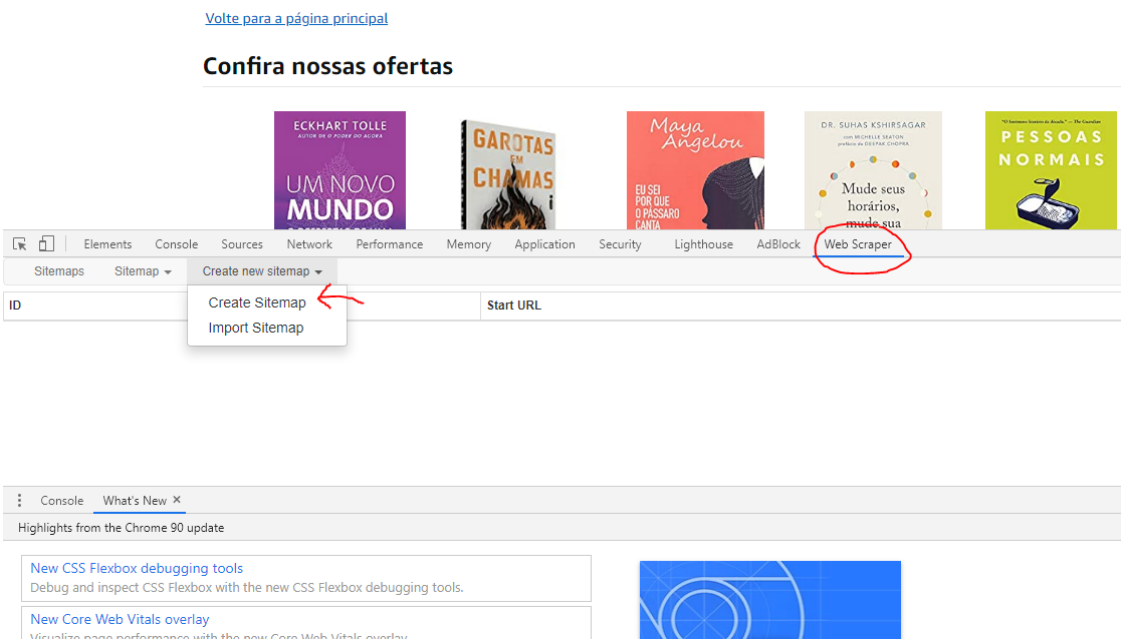
Lista de exercícios de Matplotlib:
https://www.w3resource.com/graphics/matplotlib/

- **Webscraper.io:**

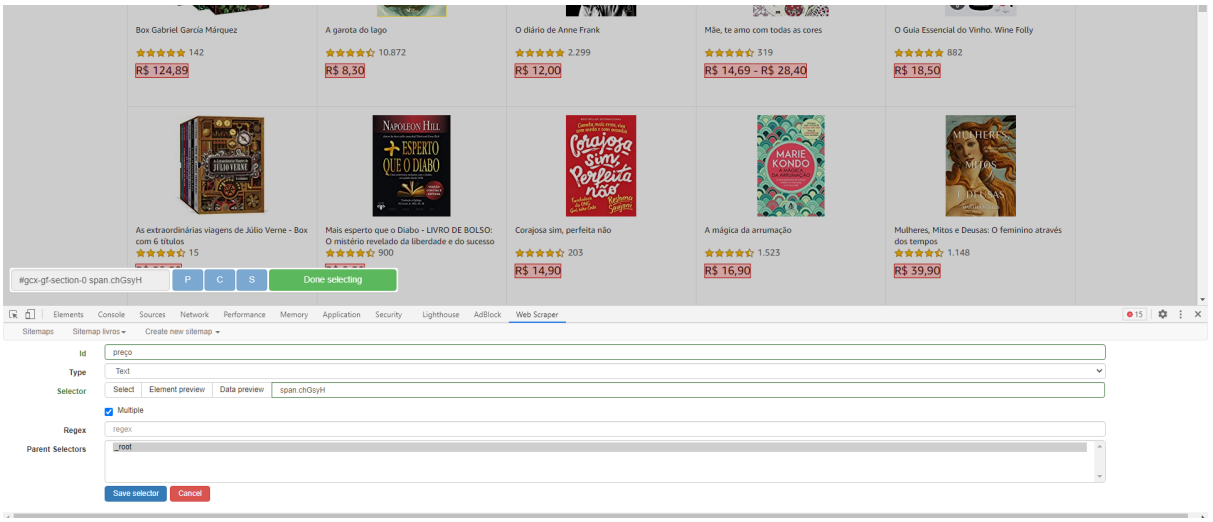
O webscraper.io é um programa gratuito em forma de plugin, mas que pode ser pago para utilizá-lo em formas mais complexas. Ele permite realizar raspagem de dados de forma simples, apenas clicando em ícones, textos e outros elementos de páginas HTML.
Mais informações acerca dessa ferramenta podem ser vistas pelo seguinte link:
https://webscraper.io/
Para utilizar o webscraper.io, é necessário utilizar o navegador Google Chrome e instalar o plugin no link:
https://chrome.google.com/webstore/detail/web-scraper-free-web-scra/jnhgnonknehpejjnehehllkliplmbmhn?hl=en
Concluída a instalação, seguimos os seguintes passos:
Abra o modo desenvolvedor em uma página web (clique F12). Caso o modo desenvolvedor esteja na vertical, clique no símbolo de três pontos verticais e clique na opção apontada pela seta na imagem:



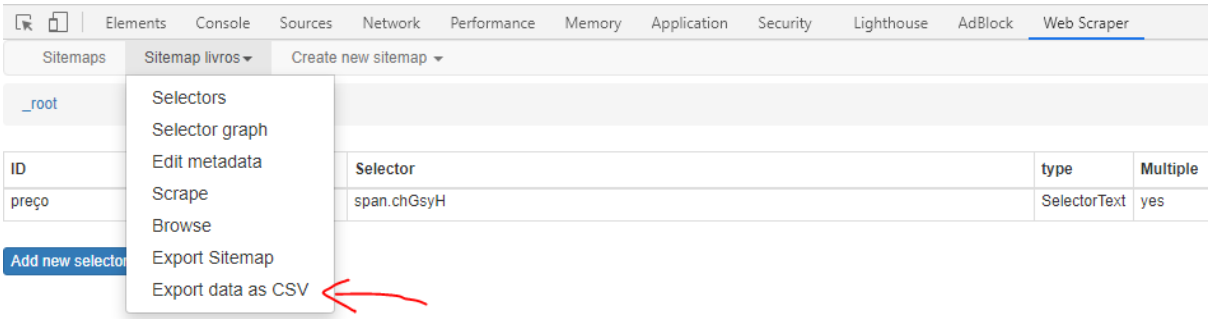
Com modo desenvolvedor na horizontal, clique em “Web Scraper”, “Create new sitemap” e “Create Sitemap”:



Em “Sitemap name” crie um nome para seu sitemap. E em “Start URL” cole o link da página que você deseja raspar. Por fim, clique em “Create Sitemap”.
Agora, clique em “Add new selector”. Na caixa de texto “Id”, crie um nome para o seletor que você está criando. Na caixa de texto “Type”, selecione o tipo de dado que será raspado (normalmente é tipo Text). Marque a caixa “Multiple” e clique em “Select”, presente na seção “Selector”.
Após clicar em “Select”, clique em alguns elementos que você queira raspar (geralmente após clicar em 2, o plugin já reconhece os outros). No exemplo a seguir, desejamos raspar os preços dos livros na página:



Com os elementos agora marcados em vermelho, clique em “Done selecting”. Por fim, clique em “Save selector”.
Então, no botão “Data Preview” é possível ver os dados raspados pelo plugin.
Caso você queira baixar os dados em formato CSV, clique em “Sitemap *nome do seu Sitemap*” e clique em “Export data as CSV”



Clique em “Download now!” e pronto, você baixou os dados raspados em formato CSV.

10. Melhorias Futuras:

Como melhorias futuras, seria recomendável trazer o tutorial de raspagem e a ferramenta de infografia presentes no Portal Jornalismo de Dados para o WikiJour. Além disso, caso seja possível hospedar o Portal ou a WikiJour em um servidor não AWS, seria interessante adaptar o código de raspagem como uma aplicação web, sendo assim não mais necessário que o programa de raspagem seja executado localmente.

Modificações do Documento				
Data	Versão	Descrição	Autor	Aprovado por
19/05/2021	1.0	Lançamento do Documento	Pedro Vitor Valença Mizuno	

Apêndice:

Planejamento do TAP
Raspagem, Infografia e Tutoriais
Pedro Vitor Valença Mizuno
Cliente / Área Solocitante
Descrição da Demanda
Patrocinadores e Gerentes do Projeto
Justificativa
Meta do Projeto / Expectativas do Cliente
Produto ou Serviço
Objetivo Estratégico
Alunos de Jornalismo de Dados / Universidade do Maranhão / Larissa de Jesus Silva
Criar um sistema de raspagem capaz de varrer os tweets de uma conta sobre um assunto
Criar um sistema que gere infográficos de acordo com os dados informados.
Criar tutoriais de ferramentas que auxiliem na área de Jornalismo de Dados
Benedito Medeiros Neto e Edison Ishikawa
Alunos e profissionais de jornalismo precisam obter informações de forma eficaz
Alunos e profissionais de jornalismo precisam representar dados visualmente
Alunos e profissionais de jornalismo precisam entender e conhecer ferramentas voltadas para a manipulação de dados
Permitir uma melhor obtenção de informações de tweets
Gerar infográficos de forma simples
Obter, de forma simples e objetiva, conhecimentos acerca de ferramentas úteis para Jornalismo de Dados
Serão utilizadas as ferramentas:
GitHub
Python 3.X
VSCode
Django
Bibliotecas de Raspagem e de Geração de Gráficos
Criar plataformas de raspagem e infografia que auxiliem na obtenção e exposição de informações. Criar também tutoriais de ferramentas que manipulam dados.
Tweets
Raspagem
Informações
Dados
Gerador
Infográfico