Análise da COVID-19 no Brasil

Semana 11 | Pandas

Introdução

O estudo da COVID-19 ainda é mundialmente relevante. Em especial, o Brasil tem travado batalhas políticas para a imunização de sua população, lida com questões sanitárias, educacionais, sociais e econômicas durante a pandemia. Estes desafios, infelizmente, converteram-se em centenas de milhares mortes até o presente momento.

Criar estudos de visualização da pandemia ao longo do tempo poderia potencialmente contribuir com a conscientização de algumas pessoas.

Questão Dirigida

Como podemos visualizar a disseminação da COVID-19 nos municípios brasileiros em datas específicas?

Desenvolvimento do Projeto

Fonte dos dados

Os dados sobre o número de confirmados e mortes foram obtidos da BrasilIO. Os dados sobre a latitude e longitude de cada município foram obtidos em um repositório GitHub. Esses conjunto de dados foram mesclados em um Banco de Dados SQL e os conjuntos de dados resultantes estão armazenados na pasta `dados`, disponibilizados juntamente com este caderno.

Existem dois arquivos na pasta `dados`. O arquivo com o sufixo `"_toy"` contém apenas algumas instâncias para viabilizar a aula.

Ferramentas de programação

Para a realização deste miniprojeto, utilizaremos as ferramentas a seguir.

Pandas

Pesquise e explique com suas palavras o que é o Pandas.

Numpy

Pesquise e explique com suas palavras o que é o Numpy.

Folium

Pesquise e explique com suas palavras o que é o Folium.

Plotly

Pesquise e explique com suas palavras o que é o Plotly.

Instalação

```
$ pip install plotly
```

```
$ pip install pandas
```

```
$ pip install folium
```

```
$ pip install numpy
```

Importação das bibliotecas

In [ ]:
```
import pandas as pd
import numpy as np
import folium
import plotly.express as px
```

Importação dos dados

In [ ]:
```
# Dataset toy (algumas amostras)
# nome_dataset = 'dados/dados-evolução-covid-municipios-br_toy.tsv'

# Dataset completo (100MB)
nome_dataset = 'dados/dados-evolução-covid-municipios-br.tsv'
```

In [ ]:
```
# df é do tipo DataFrame
df = pd.read_csv(nome_dataset, sep='\t')
#df
```

Ao ler um arquivo `.csv` com o Pandas, temos como retorno um `DataFrame`. Neste caso, dizemos que `DataFrame` é o tipo de dados do retorno da operação `read_csv()`. Você pode conferir executando a linha a seguir.

In [ ]:
```
type(df)
```

Exercício de fixação

- Em suas palavras, o que é um `DataFrame`?

Sua resposta

Manipulações do Dataset

Exibição das primeiras linhas

`DataFrame.head()`

A operação `head()` restringe a exibição dos valores às primeiras linhas somente.

In [ ]:
```
df.head()

# Opcionalmente, pode-se executar esta operação com um argumento inteiro
#df.head(15)
```

Exercício de fixação

- Como saber que outros métodos e funções podem ser executadas em um

DataFrame ?

Sua resposta

Seleção de linhas específicas

Como selecionar somente as linhas de uma data específica por exemplo?

`DataFrame.loc`

Suponha que se queira selecionar somente as cidades que possuem estatísticas em 25/02/2020.

Forma 1.

In [ ]:
```
df.loc[ df['data'] == '2020-02-25' ]
```

Forma 2.

In [ ]:
```
col_data = df['data']
df.loc[col_data == '2020-02-25' ]
```

Forma 3.

In [ ]:
```
col_data = df['data']
data_análise = '2020-02-25'

df.loc[col_data ==  data_análise]
```

Veja que as formas 1, 2 e 3 são equivalentes.

Exercício de fixação

- Qual é o tipo de retorno do `df.loc` ?
- Mude as datas (até 9/03/2020) e observe as mudanças.

`DataFrame.iloc`

O método `iloc` retorna uma linha do `DataFrame` a partir de um índice. A linha é do tipo `Series` .

In [ ]:
```
df.iloc[0] # Índice 0 => Primeira linha
```

In [ ]:
```
# verificando o tipo do retorno
type(df.iloc[0])
```

Podemos restringir o retorno do `iloc` para uma única coluna.

In [ ]:
```
df.iloc[0]['nome']
```

Os valores únicos de uma coluna

Como saber quais as datas compreendidas pelo dataset?

In [ ]:
```
df['data'].unique()
```

In [ ]:
```
lst_datas = list(df['data'].unique())
lst_datas
```

Dicionário de `DataFrames`

Como criar um DataFrame específico para cada data?

In [ ]:
```
datas_dic = {}
col_data = df['data']
for data in lst_datas:
    datas_dic[data] = df.loc[col_data == data]
```

In [ ]:
```
# Teste com diferentes datas
datas_dic['2020-03-07']
```

Filtrar colunas

In [ ]:
```
df.filter(items=['uf', 'nome', 'data', 'confirmados', 'mortes'])
```

Nomes das colunas

In [ ]:
```
df.columns
```

Ordenando os valores no DataFrame

In [ ]:
```
df_sorted = df.sort_values('confirmados', ascending=False)
df_sorted.head()
```

Exibição em gráfico de dispersão

Os n-municípios mais atingidos.

In [ ]:
```
df_data = datas_dic['2020-03-07']
nm_coluna_x = "nome" # Nome do município
nm_coluna_y = "confirmados"
qtde_municipios = 3
data_analise = df_data.iloc[0]['data']

df_sorted = df_data.sort_values('confirmados', ascending=False)

fig = px.scatter(
    df_sorted.head(qtde_municipios),
    x = nm_coluna_x,
    y = nm_coluna_y,
    size = nm_coluna_y,
    color = nm_coluna_x,
    hover_name = nm_coluna_x,
    size_max = 60
)

fig.update_layout(
    title = str(qtde_municipios) +" municipios mais atingidos em " + data_
    xaxis_title = "Municipios",
    yaxis_title = "Casos",
    width = 700
)
fig.show()
```

Função

E se a gente precisar criar um outro gráfico de dispersão? Teremos que escrever todo aquele código novamente?

Variáveis envolvidas:

`df` , `data_analise` , `nm_coluna_x` , `nm_coluna_y` , `qtde_municipios`

In [ ]:
```
# Declaração da função
def plotar_dispersão(df, nm_coluna_y="confirmados", nm_coluna_x="nome", qt
    df = df.sort_values(nm_coluna_y, ascending=False)
    data_analise = df_data.iloc[0]['data']
    fig = px.scatter(
        df.head(qtde_municipios), # Nome do DataFrame é o mesmo do parâmet
        x = nm_coluna_x,
        y = nm_coluna_y,
        size = nm_coluna_y,
        color = nm_coluna_x,
        hover_name = nm_coluna_x,
        size_max = 60
    )

    fig.update_layout(
        title = str(qtde_municipios) +" municipios mais atingidos em " + da
        xaxis_title = "Municipios",
        yaxis_title = "Casos",
        width = 700
    )
    fig.show()
```

In [ ]:
```
# Chamada de função
df_data = datas_dic['2020-04-08']
plotar_dispersão(df_data)
```

Exibição em gráfico de barra

In [ ]:
```
df_data = datas_dic['2020-03-08']
nm_coluna_x = "nome" # Nome do município
nm_coluna_y = "confirmados"
qtde_municipios = 3
data_analise = df_data.iloc[0]['data']

df_sorted = df_data.sort_values('confirmados', ascending=False)

fig = px.bar(
    df_sorted,
    x = nm_coluna_x,
    y = nm_coluna_y,
    color_discrete_sequence=["pink"],
    height = 500,
    width = 800
)
fig.update_layout(
    title = "Os " + str(qtde_municipios) + " municipios mais atingidos em '
    xaxis_title = "Municipios",
    yaxis_title = "Casos",
    width = 800
)

fig.show()
```

Função

In [ ]:
```
# Declaração da função
def plotar_barra(df, nm_coluna_y="confirmados", nm_coluna_x="nome", qtde_mu
    df = df.sort_values(nm_coluna_y, ascending=False)
    data_analise = df_data.iloc[0]['data']
    fig = px.bar(
        df.head(qtde_municipios),
        x = nm_coluna_x,
        y = nm_coluna_y,
        color_discrete_sequence=["pink"],
        height = 500,
        width = 800
    )
    fig.update_layout(
        title = "Os " + str(qtde_municipios) + " municipios mais atingidos
        xaxis_title = "Municipios",
        yaxis_title = "Casos",
        width = 800
    )

    fig.show()
```

In [ ]:
```
#### Chamada de função
df_data = datas_dic['2020-03-08']
plotar_barra(df_data)
```

Visualização no mapa

O mapa a seguir é interativo.

In [ ]:
```
df_mapa = datas_dic['2020-03-08']
```

In [ ]:
```
# Uma função de tamanho qualquer
def calc_raio(n):
    if (n < 0):
        return 0
    return (int((np.log(n + 1.00001))) + 0.2) * 1000
```

In [ ]:
```
def extrair_lat_lon(df, i):
    return [df_mapa.iloc[i]['lat'], df_mapa.iloc[i]['lon']]
```

In [ ]:
```
tam_circulo(5)
```

In [ ]:
```
world_map = folium.Map(
    location = [-16.1237611, -59.9219642],
    tiles = "cartodbpositron",
    zoom_start = 3,
    max_zoom = 10,
    min_zoom = 2
)

for i in range(0,len(df_mapa)):
    circulo = folium.Circle(
        location = extrair_lat_lon(df, i),
        fill = True,
        # Formula obtida na internet. Somente para reuso
        radius = calc_raio(df_mapa.iloc[i]['confirmados']),
        color = 'red',
        fill_color = 'indigo',
    )
    circulo.add_to(world_map)

world_map
```

Função

In [ ]:
```
def plotar_mapa(df):
    world_map = folium.Map(
        location = [-16.1237611, -59.9219642],
        tiles = "cartodbpositron",
        zoom_start = 4,
        max_zoom = 10,
        min_zoom = 2
    )

    for i in range(0,len(df)):
        raio = calc_raio(df.iloc[i]['mortes'])

        folium.Circle(
            location = extrair_lat_lon(df, i),
            fill = True,
            radius = raio,
            color = 'red',
            fill_color = 'indigo',
        ).add_to(world_map)
    return world_map
```

In [ ]:
```
df_mapa = datas_dic['2021-03-18']
plotar_mapa(df_mapa)
```