Python para Engenharia de Dados

Projeto Final

COVID-19

Anne Karoline Fortunato do Carmo

2023

Introdução

Recentemente, o mundo passou por uma pandemia, conhecida como pandemia da COVID-19. Essa pandemia impactou o mundo inteiro desde o seu surgimento em dezembro de 2019 e o Brasil registrou um alto número de casos e de óbitos ao longo dos últimos três anos.

Este documento é referente a uma análise, onde foram explorados os dados relacionados à pandemia no Brasil. Os dados utilizados foram obtidos a partir do portal Brasil.IO, que disponibiliza informações detalhadas sobre os casos confirmados, casos de óbitos e outros indicadores relacionados à doença.

Os dados disponibilizados permitiram obter insights extremamente relevantes sobre a propagação e impacto da COVID-19 no país.

Objetivo

O objetivo desta análise é investigar e obter insights relevantes sobre como foi a evolução da COVID-19 no Brasil.

Foram realizadas explorações analíticas que foram capazes de fornecer como foi a evolução dos casos no país e quais foram os padrões e as tendências.

Metodologia

Para realizar esta análise, utilizou-se a linguagem de programação Python e algumas de suas bibliotecas de análise de dados como o Pandas e o Matplotlib.

Os dados disponibilizados pelo Brasil I.O foram carregados em um Dataframe do Pandas para facilitar a análise dos dados e a sua manipulação.

Os procedimentos da análise foram:

1. Carregamentos dos dados.
2. Exploração dos dados
3. Limpeza e pré-processamento dos dados.
4. Análise Exploratória dos dados.
5. Apresentação dos resultados utilizando gráficos e tabelas.

Carregamento dos dados

Para o carregamentos dos dados foi utilizado o link fornecido (https://data.brasil.io/dataset/covid19/caso_full.csv.gz) e carregado em um Dataframe do Pandas para a análise.

```
import pandas as pd

caminho = './Dados/caso_full.csv'
df = pd.read_csv(caminho)
```
✓ 15.5s

Disponível no notebook trabalho-final.ipynb

Os 5 primeiros resultados estão exibidos no dataframe abaixo:

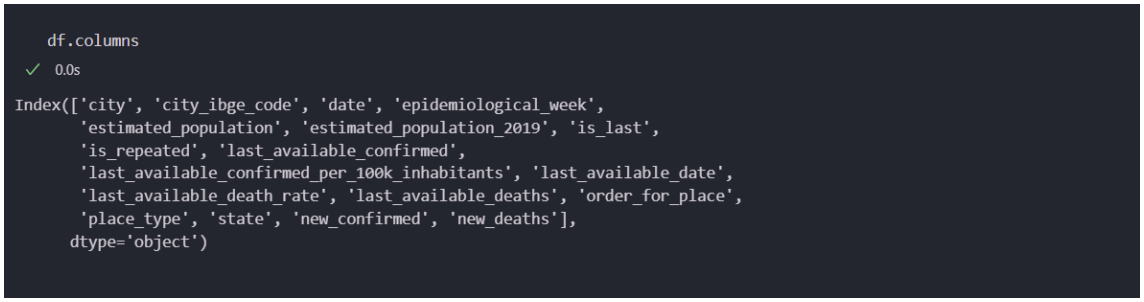
```
df.head()
```
✓ 0.2s Python

	city	city_ibge_code	date	epidemiological_week	estimated_population	estimated_population_2019	is_last	is_repeated	last_available_confirmed	last_available_confirmed_per_100k_inhabitants	l
0	Rio Branco	1200401.0	2020-03-17	202012	413418.0	407319.0	False	False	3	0.72566	
1	NaN	12.0	2020-03-17	202012	894470.0	881935.0	False	False	3	0.33539	
2	Rio Branco	1200401.0	2020-03-18	202012	413418.0	407319.0	False	False	3	0.72566	
3	NaN	12.0	2020-03-18	202012	894470.0	881935.0	False	False	3	0.33539	
4	Rio Branco	1200401.0	2020-03-19	202012	413418.0	407319.0	False	False	4	0.96754	

Disponível no notebook trabalho-final.ipynb

Exploração dos dados

Foi realizado alguns comandos para conhecer a base de dados. O df.columns retornou todas as colunas do Dataframe.



Disponível no notebook trabalho-final.ipynb

A descrição das colunas do Dataframe se encontra abaixo:

- **city:** nome do município (pode estar em branco quando o registro é referente ao estado, pode ser preenchido com Importados/Indefinidos também).
- **city_ibge_code:** código IBGE do local.
- **date:** data de coleta dos dados no formato YYYY-MM-DD.
- **epidemiological_week:** número da semana epidemiológica no formato YYYYWW.
- **estimated_population:** população estimada para esse município/estado em 2020, segundo o IBGE. (acesse o script que faz o download e conversão dos dados de população).
- **estimated_population_2019:** população estimada para esse município/estado em 2019, segundo o IBGE. ATENÇÃO: essa coluna possui valores desatualizados, prefira usar a coluna estimated_population.
- **is_last**: campo pré-computado que diz se esse registro é o mais novo para esse local, pode ser True ou False (caso filtre por esse campo, use is_last=True ou is_last=False, não use o valor em minúsculas).
- **is_repeated:** campo pré-computado que diz se as informações nesse registro foram publicadas pela Secretaria Estadual de Saúde no dia date ou se o dado é repetido do último dia em que o dado está disponível (igual

ou anterior a date). Isso ocorre pois nem todas as secretarias publicam boletins todos os dias. Veja também o campo last_available_date.

- **last_available_confirmed:** número de casos confirmados do último dia disponível igual ou anterior à data date.
- **last_available_confirmed_per_100k_inhabitants:** número de casos confirmados por 100.000 habitantes (baseado em estimated_population) do último dia disponível igual ou anterior à data date.
- **last_available_date:** data da qual o dado se refere.
- **last_available_death_rate:** taxa de mortalidade (mortes / confirmados) do último dia disponível igual ou anterior à data date.
- **last_available_deaths:** número de mortes do último dia disponível igual ou anterior à data date.
- **order_for_place:** número que identifica a ordem do registro para este local. O registro referente ao primeiro boletim em que esse local aparecer será contabilizado como 1 e os demais boletins incrementarão esse valor.
- **place_type:** tipo de local que esse registro descreve, pode ser city ou state.
- **state:** sigla da unidade federativa, exemplo: SP.
- **new_confirmed:** número de novos casos confirmados desde o último dia (note que caso is_repeated seja True, esse valor sempre será 0 e que esse valor pode ser negativo caso a SES remaneje os casos desse município para outro).
- **new_deaths:** número de novos óbitos desde o último dia (note que caso is_repeated seja True, esse valor sempre será 0 e que esse valor pode ser negativo caso a SES remaneje os casos desse município para outro).

O df.describe() retornar um resumo estatísticos dos dados como contagem, média, desvio padrão, valores mínimos e máximos. Com essas informações também é possível identificar se há valores inconsistentes ou ausentes no Dataframe.

```
df.describe()
```
✓ 1.9s — Python

	city_ibge_code	epidemiological_week	estimated_population	estimated_population_2019	last_available_confirmed	last_available_confirmed_per_100k_inhabitants	last_available_death_rate	last_avai
count	3.840002e+06	3.853648e+06	3.840002e+06	3.840002e+06	3.853648e+06	3.824482e+06	3.853648e+06	3
mean	3.228321e+06	2.021052e+05	8.116488e+04	8.054619e+04	4.777532e+03	5.984968e+03	2.650386e-02	1
std	1.009499e+06	5.755437e+01	9.057809e+05	8.992380e+05	6.180363e+04	5.358505e+03	4.975375e-02	1
min	1.100000e+01	2.020090e+05	7.760000e+02	7.810000e+02	0.000000e+00	2.160000e-03	0.000000e+00	0
25%	2.509107e+06	2.020440e+05	5.602000e+03	5.603000e+03	1.420000e+02	1.571626e+03	1.140000e-02	3
50%	3.144508e+06	2.021160e+05	1.219500e+04	1.213900e+04	4.800000e+02	4.675325e+03	1.970000e-02	9
75%	4.117297e+06	2.021400e+05	2.689900e+04	2.670000e+04	1.395000e+03	9.208211e+03	2.990000e-02	3
max	5.300108e+06	2.022130e+05	4.628933e+07	4.591905e+07	5.232374e+06	9.048099e+05	1.000000e+00	1

Disponível no notebook trabalho-final.ipynb

Limpeza e pré-processamento dos dados

Após conhecer um pouco sobre a base de dados que será trabalhada, foi utilizado o comando df.isnull() que retorna True para valores nulos ou False para campos preenchidos. É possível perceber que há campos nulos no Dataframe estudado.

```
df.isnull()
```
Python

	city	city_ibge_code	date	epidemiological_week	estimated_population	estimated_population_2019	is_last	is_repeated	last_available_confirmed	last_available_confirmed_per_100k_inhabitant
0	False	False	False	False	False	False	False	False	False	Fals
1	True	False	False	False	False	False	False	False	False	Fals
2	False	False	False	False	False	False	False	False	False	Fals
3	True	False	False	False	False	False	False	False	False	Fals
4	False	False	False	False	False	False	False	False	False	Fals
...	...	...	...	...	...	...	...	...	...	.
3853643	False	False	False	False	False	False	False	False	False	Fals
3853644	False	False	False	False	False	False	False	False	False	Fals
3853645	False	False	False	False	False	False	False	False	False	Fals
3853646	False	False	False	False	False	False	False	False	False	Fals
3853647	True	False	False	False	False	False	False	False	False	Fals

3853648 rows × 18 columns

Disponível no notebook trabalho-final.ipynb

Foi realizado uma etapa de limpeza para remoção de dados ausentes ou inconsistentes afim de garantir a qualidade dos dados utilizados.

Para verificar os dados ausentes, foi utilizado a função df.isnull().sum() para ver a soma de valores ausentes no Dataframe para casa coluna.

A próxima imagem mostra que as colunas city, city_ibge_cod, estimated_population, estimated_population_2019 e last_available_confirmed_per_100k_inhabitants possuem valores nulo. Após essa verificação, havia duas opções: deletar os registros que possuem valores ausentes(usando o comando df.dropna()) ou preencher os valores ausentes com algum valor adequado (usando o comando df.fillna()).

```
df.isnull().sum()
✓ 6.2s
```

```
city                                          20119
city_ibge_code                                13646
date                                              0
epidemiological_week                              0
estimated_population                          13646
estimated_population_2019                     13646
is_last                                           0
is_repeated                                       0
last_available_confirmed                          0
last_available_confirmed_per_100k_inhabitants 29166
last_available_date                               0
last_available_death_rate                         0
last_available_deaths                             0
order_for_place                                   0
place_type                                        0
state                                             0
new_confirmed                                     0
new_deaths                                        0
dtype: int64
```

Disponível no notebook trabalho-final.ipynb

De acordo com as definições das colunas, o campo **city** é referente ao nome do município e pode estar em branco quando o registro é referente ao estado. Desta forma, pode ser preenchido com Importados/Indefinidos. Foi inserido indefinidos nos campos nulos.

```
df['city'].fillna('Indefinidos', inplace=True)
✓ 0.1s
```

```
df.isnull().sum()
✓ 2.6s
```

```
city                                          0
city_ibge_code                            13646
date                                          0
epidemiological_week                          0
estimated_population                      13646
estimated_population_2019                 13646
is_last                                       0
is_repeated                                   0
last_available_confirmed                      0
last_available_confirmed_per_100k_inhabitants 29166
last_available_date                           0
last_available_death_rate                     0
last_available_deaths                         0
order_for_place                               0
place_type                                    0
state                                         0
new_confirmed                                 0
new_deaths                                    0
dtype: int64
```

Disponível no notebook trabalho-final.ipynb

A coluna city_ibge_code, quando nula, recebeu valor de 0.

```
df['city_ibge_code'].fillna(0, inplace=True)
✓ 0.0s
```

```
df.isnull().sum()
✓ 2.3s
```

```
city                                          0
city_ibge_code                                0
date                                          0
epidemiological_week                          0
estimated_population                      13646
estimated_population_2019                 13646
is_last                                       0
is_repeated                                   0
last_available_confirmed                      0
last_available_confirmed_per_100k_inhabitants 29166
last_available_date                           0
last_available_death_rate                     0
last_available_deaths                         0
order_for_place                               0
place_type                                    0
state                                         0
new_confirmed                                 0
new_deaths                                    0
dtype: int64
```

Disponível no notebook trabalho-final.ipynb

Como a coluna **estimated_population_2019** está desatualizada, ela foi removida do Dataframe.

```
# Como a coluna de 2019 estava desatualizada, será removida.
df = df.drop("estimated_population_2019", axis=1)
✓ 0.2s
```

```
df.isnull().sum()
✓ 2.6s
```

```
city                                          0
city_ibge_code                                0
date                                          0
epidemiological_week                          0
estimated_population                      13646
is_last                                       0
is_repeated                                   0
last_available_confirmed                      0
last_available_confirmed_per_100k_inhabitants 29166
last_available_date                           0
last_available_death_rate                     0
last_available_deaths                         0
order_for_place                               0
place_type                                    0
state                                         0
new_confirmed                                 0
new_deaths                                    0
dtype: int64
```

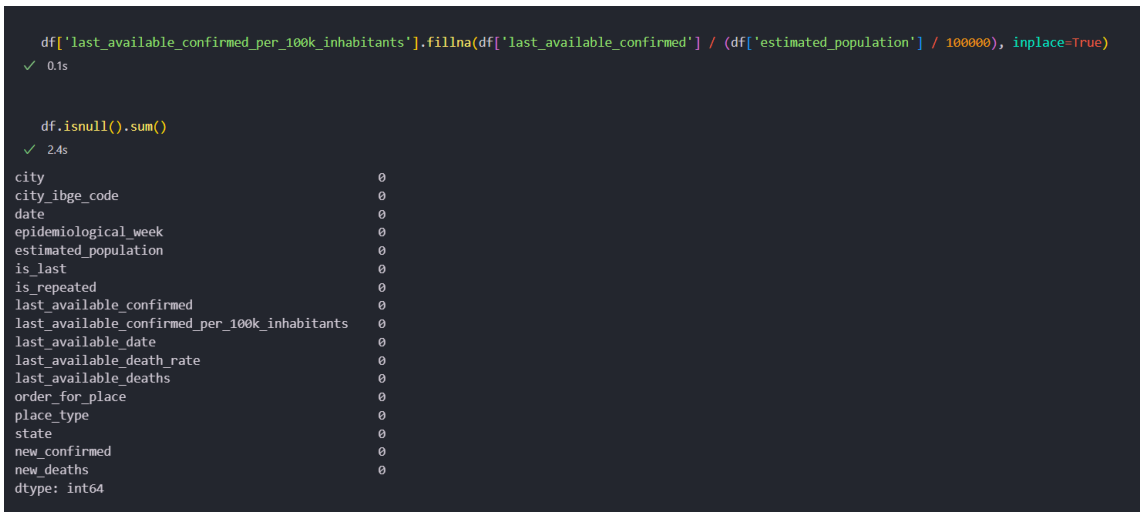
Disponível no notebook trabalho-final.ipynb

Os registros que possuíam 'estimated_population' nulo foram removidos do Dataframe.

```
df.dropna(subset=['estimated_population'], inplace=True)
✓ 0.5s

df.isnull().sum()
✓ 2.4s

city                                            0
city_ibge_code                                  0
date                                            0
epidemiological_week                            0
estimated_population                            0
is_last                                         0
is_repeated                                     0
last_available_confirmed                        0
last_available_confirmed_per_100k_inhabitants   15520
last_available_date                             0
last_available_death_rate                       0
last_available_deaths                           0
order_for_place                                 0
place_type                                      0
state                                           0
new_confirmed                                   0
new_deaths                                      0
dtype: int64
```

Disponível no notebook trabalho-final.ipynb

De acordo com a descrição das colunas, last_available_confirmed_per_100k_inhabitants é número de casos confirmados por 100.000 habitantes (baseado em estimated_population) do último dia disponível igual ou anterior à data date. Foi realiza esse tratamento nos dados para os campos que possuíam valores nulos.



```
df['last_available_confirmed_per_100k_inhabitants'].fillna(df['last_available_confirmed'] / (df['estimated_population'] / 100000), inplace=True)
✓ 0.1s

df.isnull().sum()
✓ 2.4s

city                                            0
city_ibge_code                                  0
date                                            0
epidemiological_week                            0
estimated_population                            0
is_last                                         0
is_repeated                                     0
last_available_confirmed                        0
last_available_confirmed_per_100k_inhabitants   0
last_available_date                             0
last_available_death_rate                       0
last_available_deaths                           0
order_for_place                                 0
place_type                                      0
state                                           0
new_confirmed                                   0
new_deaths                                      0
dtype: int64
```

Disponível no notebook trabalho-final.ipynb

Após realizar a limpeza dos registros nulos, a próxima etapa foi converter os tipos de dado, quando necessário. Para isso, foi necessário verificar o schema atual do Dataframe com o comando df.dtypes().

```
df.dtypes
✓ 0.1s
city                                          object
city_ibge_code                               float64
date                                          object
epidemiological_week                           int64
estimated_population                         float64
estimated_population_2019                    float64
is_last                                         bool
is_repeated                                     bool
last_available_confirmed                       int64
last_available_confirmed_per_100k_inhabitants float64
last_available_date                           object
last_available_death_rate                    float64
last_available_deaths                          int64
order_for_place                                int64
place_type                                    object
state                                         object
new_confirmed                                  int64
new_deaths                                     int64
dtype: object
```

Disponível no notebook trabalho-final.ipynb

De posse dessas informações, as únicas colunas que optou para converter o tipo de dado foi a coluna ‘date’ e ‘last_available_date’ que se encontra como tipo object (string) e é do tipo date e a coluna ‘estimated_population’ que se encontrava no tipo float e é do tipo int.

```
df['date'] = pd.to_datetime(df['date'])
df['last_available_date'] = pd.to_datetime(df['last_available_date'])
df['estimated_population'] = df['estimated_population'].astype(int)
✓ 0.3s
```

Disponível no notebook trabalho-final.ipynb

A nova configuração do schema ficou:

```
df.dtypes
✓ 0.1s

city                                                object
city_ibge_code                                     float64
date                                        datetime64[ns]
epidemiological_week                                int64
estimated_population                                int32
is_last                                              bool
is_repeated                                          bool
last_available_confirmed                            int64
last_available_confirmed_per_100k_inhabitants      float64
last_available_date                         datetime64[ns]
last_available_death_rate                          float64
last_available_deaths                               int64
order_for_place                                     int64
place_type                                         object
state                                              object
new_confirmed                                       int64
new_deaths                                          int64
dtype: object
```

Disponível no notebook trabalho-final.ipynb

last_available_date é a data da qual o dado se refere. Portando, iremos criar três novas colunas no dataframe, referente a ano, mes e dia.

```
# Extraindo ano, mes e dia em colunas separadas.
df['ano'] = df['last_available_date'].dt.year
df['mes'] = df['last_available_date'].dt.month
df['dia'] = df['last_available_date'].dt.day
```

Disponível no notebook trabalho-final.ipynb

Para realizar algumas análises, também foi realizado a divisão do df em dois dataframes. Um para place_type igual a city e outro para place_type igual a state.

```
df_city = df[df['place_type'] == 'city']
df_state = df[df['place_type'] == 'state']
✓ 0.9s
```

Disponível no notebook trabalho-final.ipynb

```
print("DataFrame de cidades:")
df_city.head()
✓ 0.0s                                                                    Python

DataFrame de cidades:
```

	city	city_ibge_code	date	epidemiological_week	estimated_population	is_last	is_repeated	last_available_confirmed	last_available_confirmed_per_100k_inhabitants	last_available_date	last_availa
0	Rio Branco	1200401.0	2020-03-17	202012	413418	False	False	3	0.72566	2020-03-17	
2	Rio Branco	1200401.0	2020-03-18	202012	413418	False	False	3	0.72566	2020-03-18	
4	Rio Branco	1200401.0	2020-03-19	202012	413418	False	False	4	0.96754	2020-03-19	
6	Rio Branco	1200401.0	2020-03-20	202012	413418	False	False	7	1.69320	2020-03-20	
8	Rio Branco	1200401.0	2020-03-21	202012	413418	False	False	11	2.66075	2020-03-21	

Disponível no notebook trabalho-final.ipynb

```
print("DataFrame de estados:")
df_state.head()
```

✓ 0.0s Python

DataFrame de estados:

	city	city_ibge_code	date	epidemiological_week	estimated_population	is_last	is_repeated	last_available_confirmed	last_available_confirmed_per_100k_inhabitants	last_available_date	last_a
1	Indefinidos	12.0	2020-03-17	202012	894470	False	False	3	0.33539	2020-03-17	
3	Indefinidos	12.0	2020-03-18	202012	894470	False	False	3	0.33539	2020-03-18	
5	Indefinidos	12.0	2020-03-19	202012	894470	False	False	4	0.44719	2020-03-19	
7	Indefinidos	12.0	2020-03-20	202012	894470	False	False	7	0.78259	2020-03-20	
9	Indefinidos	12.0	2020-03-21	202012	894470	False	False	11	1.22978	2020-03-21	

Disponível no notebook trabalho-final.ipynb

Análise Exploratória dos dados

Abaixo estão as análises realizadas:

Distribuição dos casos por estado

Abaixo estão as análises realizadas para obter a quantidade de casos confirmados por estado.

```
df_casos_por_estado_org = df_state[['state', 'new_confirmed']].groupby('state').sum()
df_casos_por_estado_org.sort_values('new_confirmed', ascending=False)
```
✓ 0.0s

Disponível no notebook trabalho-final.ipynb

Também foi realizado um código de validação, para verificar se a consulta estava sendo realizada corretamente.

```
df_estado_validacao = df[df['city'] == 'Indefinidos'].groupby('state')['new_confirmed'].sum().reset_index()
df_estado_validacao
```
✓ 0.2s

Disponível no notebook trabalho-final.ipynb

Após obter o Dataframe, foi criado um gráfico para visualizar os resultados.

```
import matplotlib.pyplot as plt

plt.figure(figsize=(15, 10))
plt.bar(df_estado_validacao['state'], df_estado_validacao['new_confirmed'])
plt.xlabel('Estados')
plt.ylabel('Casos Confirmados')
plt.title('Distribuição de Casos Confirmados por Estado')
plt.show()
```

Disponível no notebook trabalho-final.ipynb

Evolução de casos de COVID-19 por data

Para obter os casos confirmados por data, foi utilizado a coluna de nossos casos confirmados e a coluna de data correspondente ao registro.

```
df_casos_confirmados_por_data = df_city.groupby('last_available_date')['new_confirmed'].sum()
df_casos_confirmados_por_data
```
✓ 0.5s

Disponível no notebook trabalho-final.ipynb

Ordenando os dados do maior para o menor:

```
df_casos_confirmados_por_data_tabela = df_casos_confirmados_por_data.reset_index()
df_casos_confirmados_por_data_tabela.columns = ['Data', 'Casos Confirmados']
df_casos_confirmados_por_data_tabela.sort_values('Casos Confirmados', ascending=False)
```

✓ 0.0s

Disponível no notebook trabalho-final.ipynb

Código utilizado para a criação do gráfico:

```
import matplotlib.pyplot as plt

plt.figure(figsize=(15, 7))
plt.plot(df_casos_confirmados_por_data.index, df_casos_confirmados_por_data.values)
plt.xlabel('Data')
plt.ylabel('Casos Confirmados')
plt.title("Evolução dos casos confirmados de COVID-19 no Brasil")
```

✓ 0.3s

Disponível no notebook trabalho-final.ipynb

Total de mortes por COVID

Realizando a soma dos registros da coluna new_deaths para obter o total de pessoas que vieram a óbito pela doença:

```
total_mortes = df_state['new_deaths'].sum()
total_mortes
```

[118] ✓ 0.0s

```
659159
```

```
print(f'Total de mortes por covid: {total_mortes} pessoas.')
```

[119] ✓ 0.0s

```
Total de mortes por covid: 659159 pessoas.
```

Disponível no notebook trabalho-final.ipynb

Total de mortes por ano

Obtendo o total de mortes, separando os resultados pelos anos:

```
mortes_por_ano = df_state.groupby('ano')['new_deaths'].sum()
mortes_por_ano
```

✓ 0.0s

```
ano
2020    195072
2021    424263
2022     39824
Name: new_deaths, dtype: int64
```

Disponível no notebook trabalho-final.ipynb

Código utilizado para a criação do gráfico:

```
import matplotlib.pyplot as plt

plt.bar(mortes_por_ano.index, mortes_por_ano.values)
plt.xlabel('Ano')
plt.ylabel('Número de Mortes')
plt.title('Número de Mortes por Ano')
plt.show()
```

Disponível no notebook trabalho-final.ipynb

Taxa de mortalidade por estado

Obtendo a taxa de mortalidade por estado do país:

```
df_mortalidade_por_estado = df_state.groupby('state')['last_available_death_rate'].mean()
df_mortalidade_por_estado
```

✓ 0.1s

Disponível no notebook trabalho-final.ipynb

Ordenando os registros em ordem crescente:

```
df_mortalidade_por_estado = df_mortalidade_por_estado.sort_values(ascending=False)
df_mortalidade_por_estado
```

✓ 0.0s

Disponível no notebook trabalho-final.ipynb

Código utilizado para a criação do gráfico:

```
import matplotlib.pyplot as plt

plt.figure(figsize=(15, 9))
plt.barh(df_mortalidade_por_estado.index, df_mortalidade_por_estado.values)
plt.xlabel('Taxa de Mortalidade')
plt.ylabel('Estados')
plt.title("Taxa de mortalidade de COVID-19 por estado")
```

✓ 0.3s

Disponível no notebook trabalho-final.ipynb

Análise regional do Brasil

Para fazer uma análise por região do brasil, é necessário criar uma nova coluna no dataframe.
Desta forma, foi possível agrupar os dados por região geográfica baseado na sigla dos estados. é possível realizar uma comparação de casos confirmados óbito.

```
estado_regiao = {
    'AC': 'Norte',
    'AL': 'Nordeste',
    'AP': 'Norte',
    'AM': 'Norte',
    'BA': 'Nordeste',
    'CE': 'Nordeste',
    'DF': 'Centro-Oeste',
    'ES': 'Sudeste',
    'GO': 'Centro-Oeste',
    'MA': 'Nordeste',
    'MT': 'Centro-Oeste',
    'MS': 'Centro-Oeste',
    'MG': 'Sudeste',
    'PA': 'Norte',
    'PB': 'Nordeste',
    'PR': 'Sul',
    'PE': 'Nordeste',
    'PI': 'Nordeste',
    'RJ': 'Sudeste',
    'RN': 'Nordeste',
    'RS': 'Sul',
    'RO': 'Norte',
    'RR': 'Norte',
    'SC': 'Sul',
    'SP': 'Sudeste',
    'SE': 'Nordeste',
    'TO': 'Norte'
}

df_city['region'] = df_city['state'].map(estado_regiao)
```

Disponível no notebook trabalho-final.ipynb

Agrupando os registros por região e obtendo a soma dos casos confirmados, soma dos óbitos e a media da taxa de mortalidade:

```
df_regiao = df_city.groupby('region').agg({
    'last_available_confirmed': 'sum',
    'last_available_deaths': 'sum',
    'last_available_death_rate': 'mean'
}).reset_index().head()
df_regiao.head()
```

✓ 0.6s

	region	last_available_confirmed	last_available_deaths	last_available_death_rate
0	Centro-Oeste	943253839	22627734	0.026320
1	Nordeste	2078957306	51866731	0.026927
2	Norte	862471350	21405231	0.023430
3	Sudeste	3358069907	115088379	0.029856
4	Sul	1725609019	36151416	0.022257

Disponível no notebook trabalho-final.ipynb

Ordenando os registros obtidos:

```
df_regiao = df_regiao.sort_values('last_available_confirmed', ascending=False)
df_regiao
```

✓ 0.0s

	region	last_available_confirmed	last_available_deaths	last_available_death_rate
3	Sudeste	3358069907	115088379	0.029856
1	Nordeste	2078957306	51866731	0.026927
4	Sul	1725609019	36151416	0.022257
0	Centro-Oeste	943253839	22627734	0.026320
2	Norte	862471350	21405231	0.023430

Disponível no notebook trabalho-final.ipynb

Código utilizado para a criação do gráfico:

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.bar(df_regiao['region'], df_regiao['last_available_confirmed'], label='Casos Confirmados')
plt.bar(df_regiao['region'], df_regiao['last_available_deaths'], label='Óbitos')
plt.xlabel('Região')
plt.ylabel('Quantidade')
plt.title('Casos Confirmados e Óbitos por Região')
plt.legend()
plt.show()
```

✓ 0.1s

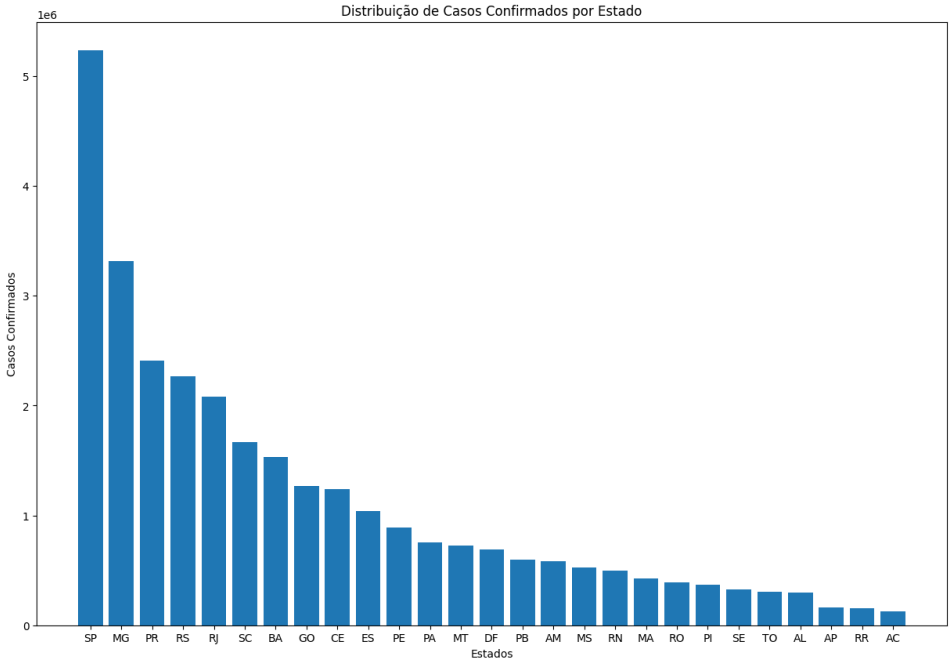
Disponível no notebook trabalho-final.ipynb

Resultados

Os resultados da análise estão apresentados em forma de gráficos e comentários descritivos. Os insights obtidos com a propagação da COVID-19 no brasil indica que:

Distribuição dos casos por estado

Com base no gráfico gerado, percebe-se que alguns estados brasileiros apresentam um número significativamente maior de casos em comparação com outros. Pode-se concluir que a propagação da COVID-19 não foi uniformemente distribuída pelo país.



Disponível no notebook trabalho-final.ipynb

Total de mortes por ano

O primeiro registro foi realizado em 23/02/2020 e o último registro do dataset foi realizado em 27/03/2022.

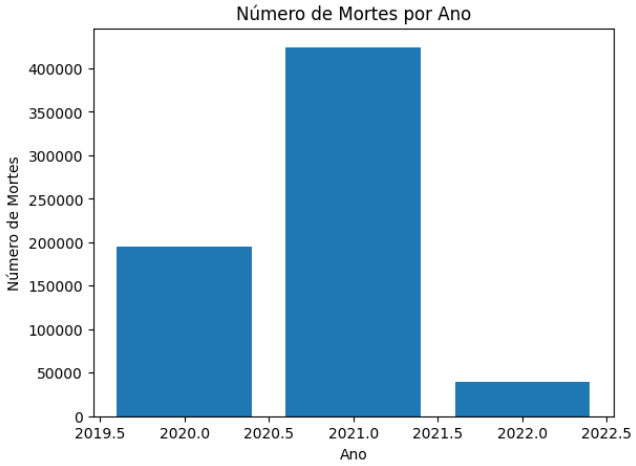
```
primeira_data = df['last_available_date'].min()
ultima_data = df['last_available_date'].max()

print("Primeira data:", primeira_data)
print("Última data:", ultima_data)
```

✓ 0.3s

```
Primeira data: 2020-02-25 00:00:00
Última data: 2022-03-27 00:00:00
```

Disponível no notebook trabalho-final.ipynb

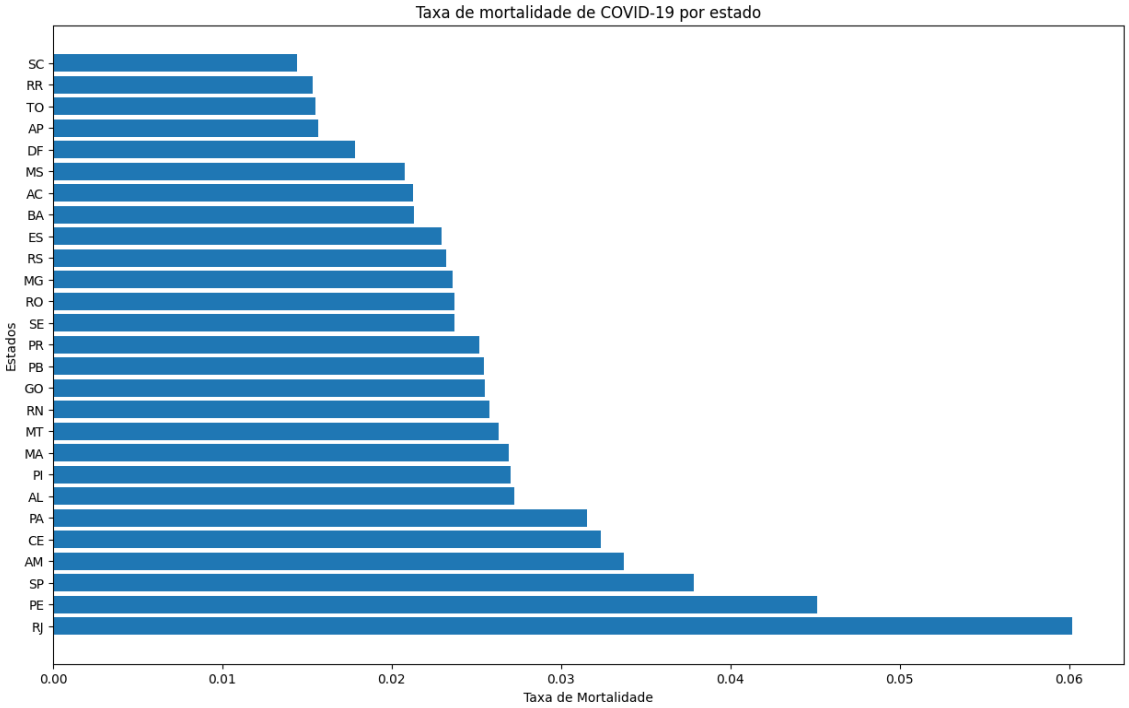
O gráfico abaixo mostra a quantidades de mortes registradas nos anos de 2020, 2021 e 2022, dentro do intervalo entre a primeira e a última data registrada.



Disponível no notebook trabalho-final.ipynb

Taxa de mortalidade por estado

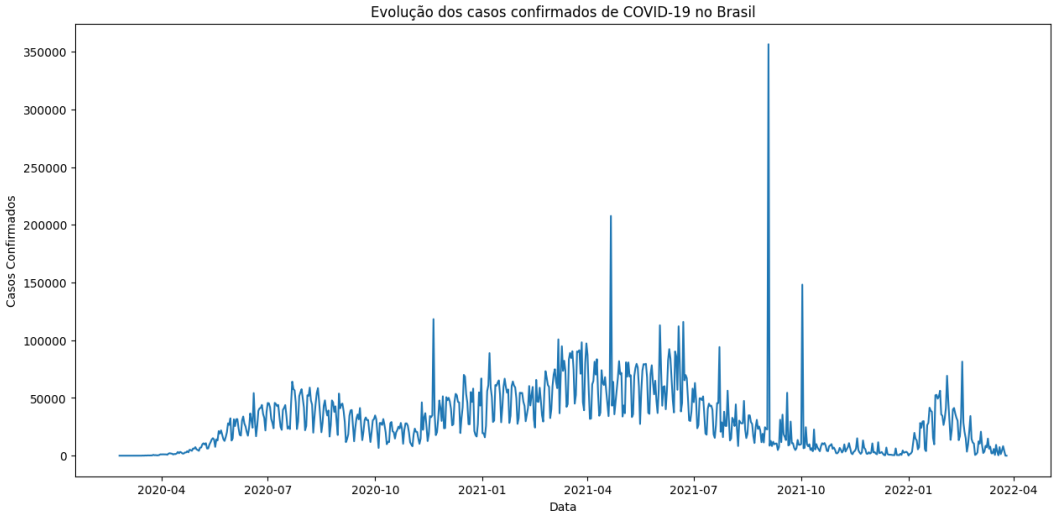
A taxa de mortalidade por estado está exibida no gráfico abaixo. De acordo com a população do estado, o Rio de Janeiro foi o estado com a maior Taxa de mortalidade e em segundo lugar ficou o estado de Pernambuco.



Disponível no notebook trabalho-final.ipynb

Evolução de casos de COVID-19 por data

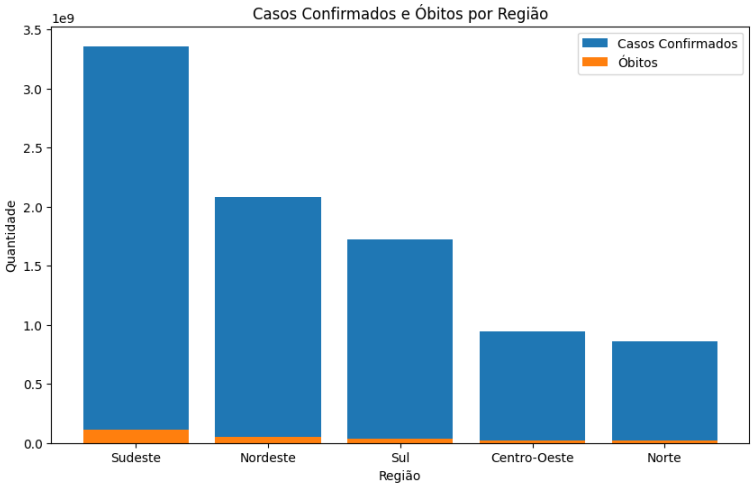
O dia 03/09/2021 é considerado o pico de casos confirmados no Brasil. Foi registrado 356.442 casos. O segundo pico é do dia 21/04/2021 e o terceiro no dia 02/10/2021.



Disponível no notebook trabalho-final.ipynb

Análise regional do Brasil

A região com o maior número de casos confirmados foi a Sudeste, seguido de Nordeste, Sul, Centro-Oeste e Norte. Percebe-se também que a COVI-19 afetou as regiões de maneiras distintas, podem principalmente ter sido influenciados pelos fatores de densidade populacional da região, infraestrutura de saúde e das medidas de controle adotadas pelos governos.



Disponível no notebook trabalho-final.ipynb

Conclusão

A conclusão dessa análise forneceu uma visão geral dos principais aspectos da pandemia da COVID-19 no Brasil.

Análises como a de taxa de crescimentos dos casos por data nos permite identificas os períodos em que houve aumento e diminuição da propagação da COVI-19. Essa análise acaba sendo primordial para monitorar a evolução da pandemia no país ao longo do tempo e, consequentemente, auxiliar no controle da doença. Ao analisar os casos por região, percebe-se uma disparidade. Tal disparidade pode ter sido ocasionada pelas políticas públicas adotadas, pela infraestrutura da região e pela densidade populacional.

Em resumo, pode-se concluir que a COVID-19 teve impactos bem heterogêneos no Brasil, variando entre cidades, estados e regiões.