Machine Learning para prever a evolução do COVID-19 no Brasil

Manipulando dados com blibioteca Pandas e Python

Digital Innovation One

In [1]:

```
# Importar biblioteca
import pandas as pd
import numpy as np
from datetime import datetime
import plotly.express as px
import plotly.graph_objects as go
import re
```

In [2]:

```
# Importar os dados (upload do arquivo)
df = pd.read_csv('E:\REPOSITÓRIO\PUBLICO\Machine-Learning-Evolucao-COVID-19-Brasil\covid_19_data.csv')
df
```

Out[2]:

	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
0	1	01/22/2020	Anhui	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
1	2	01/22/2020	Beijing	Mainland China	1/22/2020 17:00	14.0	0.0	0.0
2	3	01/22/2020	Chongqing	Mainland China	1/22/2020 17:00	6.0	0.0	0.0
3	4	01/22/2020	Fujian	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
4	5	01/22/2020	Gansu	Mainland China	1/22/2020 17:00	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
26708	26709	05/19/2020	Wyoming	US	2020-05-20 02:32:19	776.0	10.0	0.0
26709	26710	05/19/2020	Xinjiang	Mainland China	2020-05-20 02:32:19	76.0	3.0	73.0
26710	26711	05/19/2020	Yukon	Canada	2020-05-20 02:32:19	11.0	0.0	11.0
26711	26712	05/19/2020	Yunnan	Mainland China	2020-05-20 02:32:19	185.0	2.0	183.0
26712	26713	05/19/2020	Zhejiang	Mainland China	2020-05-20 02:32:19	1268.0	1.0	1267.0

26713 rows × 8 columns

In [3]:
```
# Verificando os tipos de dados
df.dtypes
```

Out[3]:
```
SNo                   int64
ObservationDate      object
Province/State       object
Country/Region       object
Last Update          object
Confirmed           float64
Deaths              float64
Recovered           float64
dtype: object
```

In [4]:
```
# Alterar tipo de dado das tabelas ('ObservationDate', 'Last Update') de "object" para tipo "datetime"
df = pd.read_csv('E:\REPOSITÓRIO\PUBLICO\Machine-Learning-Evolucao-COVID-19-Brasil\covid_19_data.csv', parse_dates=['ObservationDa
df
```

Out[4]:

	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
0	1	2020-01-22	Anhui	Mainland China	2020-01-22 17:00:00	1.0	0.0	0.0
1	2	2020-01-22	Beijing	Mainland China	2020-01-22 17:00:00	14.0	0.0	0.0
2	3	2020-01-22	Chongqing	Mainland China	2020-01-22 17:00:00	6.0	0.0	0.0
3	4	2020-01-22	Fujian	Mainland China	2020-01-22 17:00:00	1.0	0.0	0.0
4	5	2020-01-22	Gansu	Mainland China	2020-01-22 17:00:00	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
26708	26709	2020-05-19	Wyoming	US	2020-05-20 02:32:19	776.0	10.0	0.0
26709	26710	2020-05-19	Xinjiang	Mainland China	2020-05-20 02:32:19	76.0	3.0	73.0
26710	26711	2020-05-19	Yukon	Canada	2020-05-20 02:32:19	11.0	0.0	11.0
26711	26712	2020-05-19	Yunnan	Mainland China	2020-05-20 02:32:19	185.0	2.0	183.0
26712	26713	2020-05-19	Zhejiang	Mainland China	2020-05-20 02:32:19	1268.0	1.0	1267.0

26713 rows × 8 columns

In [5]:

```python
# Verificando os tipos de dados
df.dtypes
```

Out[5]:

```
SNo                         int64
ObservationDate    datetime64[ns]
Province/State             object
Country/Region             object
Last Update        datetime64[ns]
Confirmed                 float64
Deaths                    float64
Recovered                 float64
dtype: object
```

In [6]:

```python
# Função "corrige_colunas" elimina caracter especiais, espaço e transfoma em letras minúsculas
def corrige_colunas(col_name):
    return re.sub(r"[/| ]", "", col_name).lower()
```

In [7]:

```python
# Testando a função "corrige_colunas"
corrige_colunas("ADGe/P ou")
```

Out[7]:

```
'adgepou'
```

In [8]:

```python
# Exibir nome das colunas em uma lista
df.columns
```

Out[8]:

```
Index(['SNo', 'ObservationDate', 'Province/State', 'Country/Region',
       'Last Update', 'Confirmed', 'Deaths', 'Recovered'],
      dtype='object')
```

In [9]:

```python
# Alterar os nomes de todas as colunas do "df"
df.columns = [corrige_colunas(col) for col in df.columns]
df.columns
```

Out[9]:

```
Index(['sno', 'observationdate', 'provincestate', 'countryregion',
       'lastupdate', 'confirmed', 'deaths', 'recovered'],
      dtype='object')
```

In [10]:

```
# Visualizando as 10 primeiras linhas
df.head(10)
```

Out[10]:

	sno	observationdate	provincestate	countryregion	lastupdate	confirmed	deaths	recovered
0	1	2020-01-22	Anhui	Mainland China	2020-01-22 17:00:00	1.0	0.0	0.0
1	2	2020-01-22	Beijing	Mainland China	2020-01-22 17:00:00	14.0	0.0	0.0
2	3	2020-01-22	Chongqing	Mainland China	2020-01-22 17:00:00	6.0	0.0	0.0
3	4	2020-01-22	Fujian	Mainland China	2020-01-22 17:00:00	1.0	0.0	0.0
4	5	2020-01-22	Gansu	Mainland China	2020-01-22 17:00:00	0.0	0.0	0.0
5	6	2020-01-22	Guangdong	Mainland China	2020-01-22 17:00:00	26.0	0.0	0.0
6	7	2020-01-22	Guangxi	Mainland China	2020-01-22 17:00:00	2.0	0.0	0.0
7	8	2020-01-22	Guizhou	Mainland China	2020-01-22 17:00:00	1.0	0.0	0.0
8	9	2020-01-22	Hainan	Mainland China	2020-01-22 17:00:00	4.0	0.0	0.0
9	10	2020-01-22	Hebei	Mainland China	2020-01-22 17:00:00	1.0	0.0	0.0

In [11]:

```
# Quantidade de linhas e colunas
df.shape
```

Out[11]:

```
(26713, 8)
```

In [12]:

```
# Verificando se temos dados faltantes na tabela (nulos)
df.isnull().sum()
```

Out[12]:

```
sno                    0
observationdate        0
provincestate      13831
countryregion          0
lastupdate             0
confirmed              0
deaths                 0
recovered              0
dtype: int64
```

In [13]:

```
# Deletando coluna indesejada 'provincestate'
df.drop(['provincestate'], axis=1, inplace=True)
```

```
In [14]: # Verificando se temos dados faltantes na tabela (nulos)
         df.isnull().sum()
```

```
Out[14]: sno                0
         observationdate    0
         countryregion      0
         lastupdate         0
         confirmed          0
         deaths             0
         recovered          0
         dtype: int64
```

```
In [15]: # Quantidade de linhas e colunas
         df.shape
```

```
Out[15]: (26713, 7)
```

Brasil - COVID-19

Vamos selecionar apenas os dados do Brasil para investigar

```
In [16]: # Exibir palavra 'chave' e quantificar sua aparição
         df.countryregion.value_counts()
```

```
Out[16]: US                     4990
         Mainland China         3687
         Canada                 1093
         Australia               788
         France                  752
                                ... 
         North Ireland             1
         Channel Islands           1
         Cape Verde                1
         Republic of Ireland       1
         East Timor                1
         Name: countryregion, Length: 223, dtype: int64
```

In [17]:

```
# Exibir lista de palavra 'chaves' existente dentro da coluna "countryregion"
df["countryregion"].unique()
```

Out[17]:

```
array(['Mainland China', 'Hong Kong', 'Macau', 'Taiwan', 'US', 'Japan',
       'Thailand', 'South Korea', 'Singapore', 'Philippines', 'Malaysia',
       'Vietnam', 'Australia', 'Mexico', 'Brazil', 'Colombia', 'France',
       'Nepal', 'Canada', 'Cambodia', 'Sri Lanka', 'Ivory Coast',
       'Germany', 'Finland', 'United Arab Emirates', 'India', 'Italy',
       'UK', 'Russia', 'Sweden', 'Spain', 'Belgium', 'Others', 'Egypt',
       'Iran', 'Israel', 'Lebanon', 'Iraq', 'Oman', 'Afghanistan',
       'Bahrain', 'Kuwait', 'Austria', 'Algeria', 'Croatia',
       'Switzerland', 'Pakistan', 'Georgia', 'Greece', 'North Macedonia',
       'Norway', 'Romania', 'Denmark', 'Estonia', 'Netherlands',
       'San Marino', ' Azerbaijan', 'Belarus', 'Iceland', 'Lithuania',
       'New Zealand', 'Nigeria', 'North Ireland', 'Ireland', 'Luxembourg',
       'Monaco', 'Qatar', 'Ecuador', 'Azerbaijan', 'Czech Republic',
       'Armenia', 'Dominican Republic', 'Indonesia', 'Portugal',
       'Andorra', 'Latvia', 'Morocco', 'Saudi Arabia', 'Senegal',
       'Argentina', 'Chile', 'Jordan', 'Ukraine', 'Saint Barthelemy',
       'Hungary', 'Faroe Islands', 'Gibraltar', 'Liechtenstein', 'Poland',
       'Tunisia', 'Palestine', 'Bosnia and Herzegovina', 'Slovenia',
       'South Africa', 'Bhutan', 'Cameroon', 'Costa Rica', 'Peru',
       'Serbia', 'Slovakia', 'Togo', 'Vatican City', 'French Guiana',
       'Malta', 'Martinique', 'Republic of Ireland', 'Bulgaria',
       'Maldives', 'Bangladesh', 'Moldova', 'Paraguay', 'Albania',
       'Cyprus', 'St. Martin', 'Brunei', 'occupied Palestinian territory',
       "('St. Martin',)", 'Burkina Faso', 'Channel Islands', 'Holy See',
       'Mongolia', 'Panama', 'Bolivia', 'Honduras', 'Congo (Kinshasa)',
       'Jamaica', 'Reunion', 'Turkey', 'Cuba', 'Guyana', 'Kazakhstan',
       'Cayman Islands', 'Guadeloupe', 'Ethiopia', 'Sudan', 'Guinea',
       'Antigua and Barbuda', 'Aruba', 'Kenya', 'Uruguay', 'Ghana',
       'Jersey', 'Namibia', 'Seychelles', 'Trinidad and Tobago',
       'Venezuela', 'Curacao', 'Eswatini', 'Gabon', 'Guatemala',
       'Guernsey', 'Mauritania', 'Rwanda', 'Saint Lucia',
       'Saint Vincent and the Grenadines', 'Suriname', 'Kosovo',
       'Central African Republic', 'Congo (Brazzaville)',
       'Equatorial Guinea', 'Uzbekistan', 'Guam', 'Puerto Rico', 'Benin',
       'Greenland', 'Liberia', 'Mayotte', 'Republic of the Congo',
       'Somalia', 'Tanzania', 'The Bahamas', 'Barbados', 'Montenegro',
       'The Gambia', 'Kyrgyzstan', 'Mauritius', 'Zambia', 'Djibouti',
       'Gambia, The', 'Bahamas, The', 'Chad', 'El Salvador', 'Fiji',
       'Nicaragua', 'Madagascar', 'Haiti', 'Angola', 'Cabo Verde',
       'Niger', 'Papua New Guinea', 'Zimbabwe', 'Cape Verde',
       'East Timor', 'Eritrea', 'Uganda', 'Bahamas', 'Dominica', 'Gambia',
       'Grenada', 'Mozambique', 'Syria', 'Timor-Leste', 'Belize', 'Laos',
       'Libya', 'Diamond Princess', 'Guinea-Bissau', 'Mali',
       'Saint Kitts and Nevis', 'West Bank and Gaza', 'Burma',
       'MS Zaandam', 'Botswana', 'Burundi', 'Sierra Leone', 'Malawi',
       'South Sudan', 'Western Sahara', 'Sao Tome and Principe', 'Yemen',
       'Comoros', 'Tajikistan', 'Lesotho'], dtype=object)
```

In [18]:
```
# Filtro: exibir tabela apenas onde na coluna "countryregion" possuir a palavra "Brazil"
df.loc[df.countryregion == 'Brazil']
```

Out[18]:

	sno	observationdate	countryregion	lastupdate	confirmed	deaths	recovered
82	83	2020-01-23	Brazil	2020-01-23 17:00:00	0.0	0.0	0.0
2455	2456	2020-02-26	Brazil	2020-02-26 23:53:02	1.0	0.0	0.0
2559	2560	2020-02-27	Brazil	2020-02-26 23:53:02	1.0	0.0	0.0
2668	2669	2020-02-28	Brazil	2020-02-26 23:53:02	1.0	0.0	0.0
2776	2777	2020-02-29	Brazil	2020-02-29 21:03:05	2.0	0.0	0.0
...	...	...	...	...	...	...	...
24850	24851	2020-05-15	Brazil	2020-05-16 02:32:19	220291.0	14962.0	84970.0
25227	25228	2020-05-16	Brazil	2020-05-17 02:32:32	233511.0	15662.0	89672.0
25604	25605	2020-05-17	Brazil	2020-05-18 02:32:21	241080.0	16118.0	94122.0
25981	25982	2020-05-18	Brazil	2020-05-19 02:32:18	255368.0	16853.0	100459.0
26358	26359	2020-05-19	Brazil	2020-05-20 02:32:19	271885.0	17983.0	106794.0

85 rows × 7 columns

In [19]:

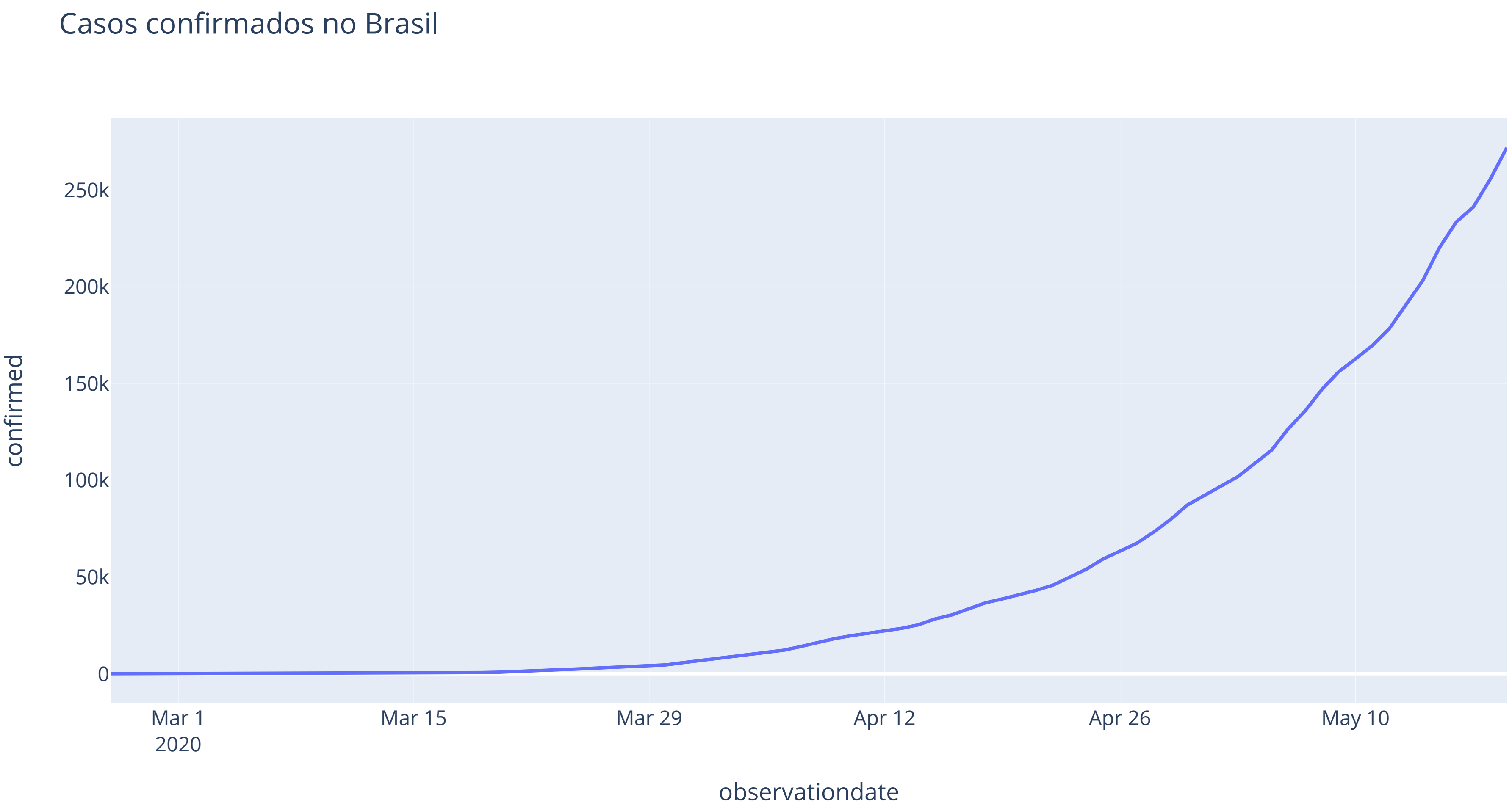
```python
# Criar nova tabela "brasil" utilizando filtro com duas condições:
# 1ª: coluna "countryregion" possuir apenas a palavra chave "Brazil"
# 2ª: coluna "confirmed" deve ter valor maior que zero
brasil = df.loc[ (df.countryregion == 'Brazil') & (df.confirmed > 0) ]
#Visualizando as 5 primeiras linhas
brasil.head()
```

Out[19]:

	sno	observationdate	countryregion	lastupdate	confirmed	deaths	recovered
2455	2456	2020-02-26	Brazil	2020-02-26 23:53:02	1.0	0.0	0.0
2559	2560	2020-02-27	Brazil	2020-02-26 23:53:02	1.0	0.0	0.0
2668	2669	2020-02-28	Brazil	2020-02-26 23:53:02	1.0	0.0	0.0
2776	2777	2020-02-29	Brazil	2020-02-29 21:03:05	2.0	0.0	0.0
2903	2904	2020-03-01	Brazil	2020-02-29 21:03:05	2.0	0.0	0.0

Gráfico de casos COVID-19 confirmados no Brasil

```
In [20]: # Gráfico da evolução de casos confirmados
         px.line(brasil, 'observationdate', 'confirmed', title='Casos confirmados no Brasil')
```



Novos casos de COVID-19 no Brasil por dia

In [21]:
```
#Visualizando as 10 primeiras linhas
brasil.head(10)
```

Out[21]:

	sno	observationdate	countryregion	lastupdate	confirmed	deaths	recovered
2455	2456	2020-02-26	Brazil	2020-02-26 23:53:02	1.0	0.0	0.0
2559	2560	2020-02-27	Brazil	2020-02-26 23:53:02	1.0	0.0	0.0
2668	2669	2020-02-28	Brazil	2020-02-26 23:53:02	1.0	0.0	0.0
2776	2777	2020-02-29	Brazil	2020-02-29 21:03:05	2.0	0.0	0.0
2903	2904	2020-03-01	Brazil	2020-02-29 21:03:05	2.0	0.0	0.0
3032	3033	2020-03-02	Brazil	2020-02-29 21:03:05	2.0	0.0	0.0
3173	3174	2020-03-03	Brazil	2020-02-29 21:03:05	2.0	0.0	0.0
3322	3323	2020-03-04	Brazil	2020-03-04 20:33:02	4.0	0.0	0.0
3486	3487	2020-03-05	Brazil	2020-03-04 20:33:02	4.0	0.0	0.0
3647	3648	2020-03-06	Brazil	2020-03-06 20:33:03	13.0	0.0	0.0

In [22]:
```
# Quantidade de linhas e colunas
brasil.shape
```

Out[22]:
```
(84, 7)
```

In [23]:
```
# Técnica de programação funcional
# Esta função cria uma nova coluna que em resumo => (casos_confirmado_hoje - casos_confirmado_ontem = aumento_casos)
brasil['novoscasos'] = list(map( lambda x: 0 if (x==0) else brasil ['confirmed'].iloc[x] - brasil['confirmed'].iloc[x-1], np.arang
```

```
C:\Users\Décio\AppData\Local\Temp\ipykernel_14552\1565154260.py:3: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-vers
us-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
```

In [24]:
```
# Exibir tabela
brasil
```

Out[24]:

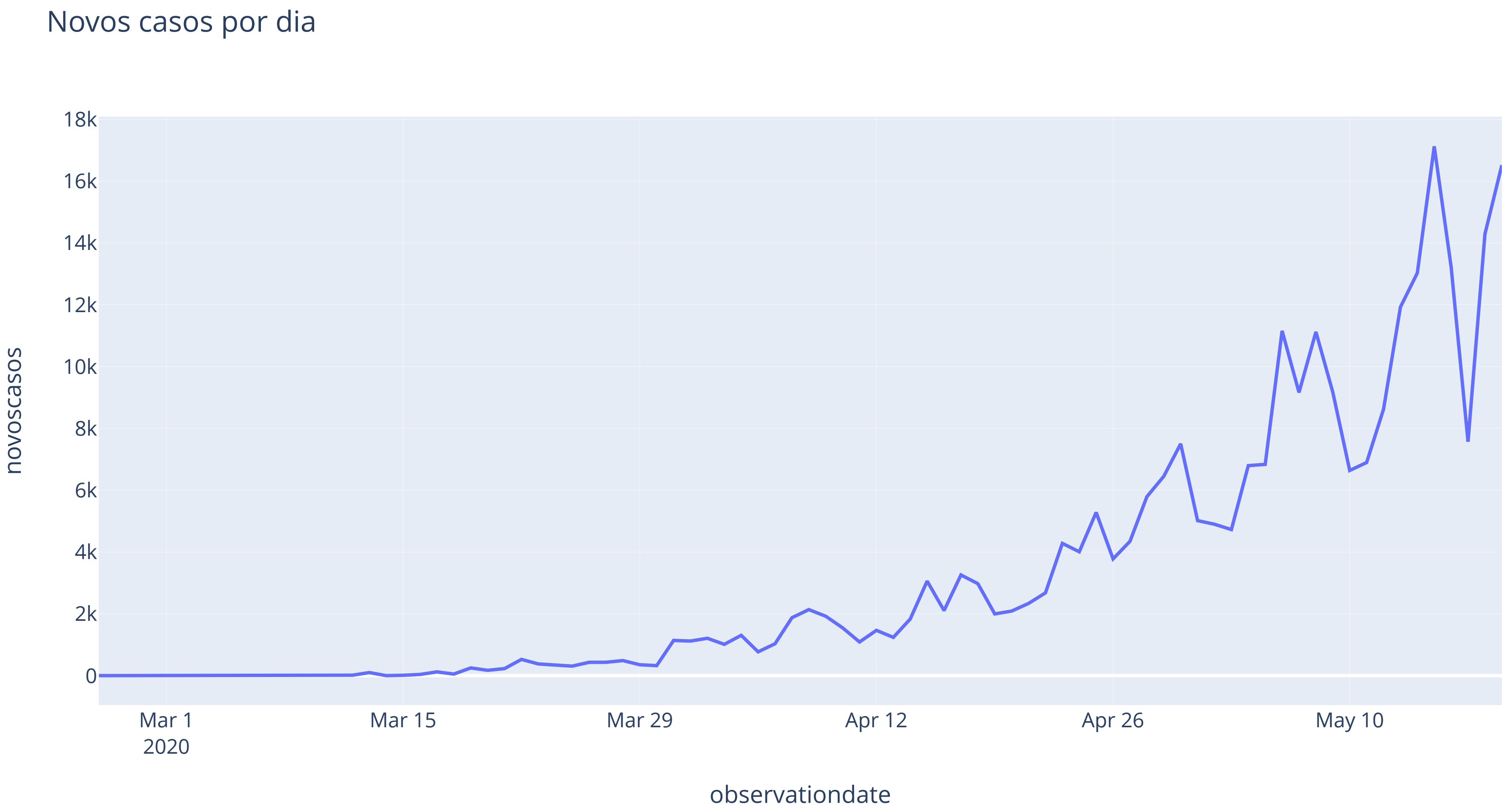
	sno	observationdate	countryregion	lastupdate	confirmed	deaths	recovered	novoscasos
2455	2456	2020-02-26	Brazil	2020-02-26 23:53:02	1.0	0.0	0.0	0.0
2559	2560	2020-02-27	Brazil	2020-02-26 23:53:02	1.0	0.0	0.0	0.0
2668	2669	2020-02-28	Brazil	2020-02-26 23:53:02	1.0	0.0	0.0	0.0
2776	2777	2020-02-29	Brazil	2020-02-29 21:03:05	2.0	0.0	0.0	1.0
2903	2904	2020-03-01	Brazil	2020-02-29 21:03:05	2.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
24850	24851	2020-05-15	Brazil	2020-05-16 02:32:19	220291.0	14962.0	84970.0	17126.0
25227	25228	2020-05-16	Brazil	2020-05-17 02:32:32	233511.0	15662.0	89672.0	13220.0
25604	25605	2020-05-17	Brazil	2020-05-18 02:32:21	241080.0	16118.0	94122.0	7569.0
25981	25982	2020-05-18	Brazil	2020-05-19 02:32:18	255368.0	16853.0	100459.0	14288.0
26358	26359	2020-05-19	Brazil	2020-05-20 02:32:19	271885.0	17983.0	106794.0	16517.0

84 rows × 8 columns

In [25]:
```
# Quantidade de linhas e colunas
brasil.shape
```

Out[25]:
```
(84, 8)
```

```
In [26]: # Visualizando gráfico da nova coluna criada "novoscasos"
         px.line(brasil, x='observationdate', y='novoscasos', title='Novos casos por dia')
```

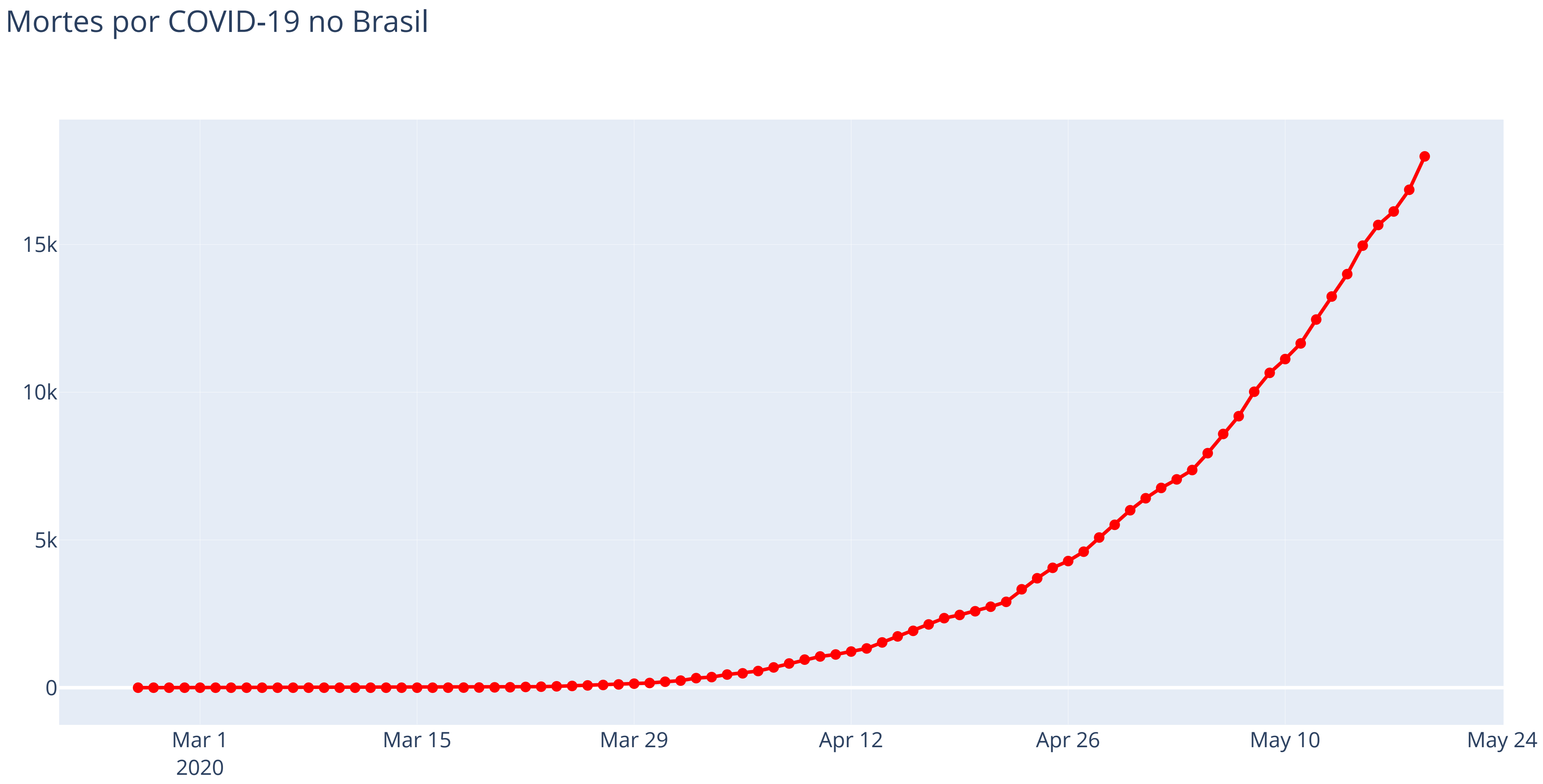


Progressão de mortes no Brasil

```
In [27]: # Criar gráfico de "Mortes"
         fig = go.Figure()
         fig.add_trace( go.Scatter(x=brasil.observationdate, y=brasil.deaths, name='Mortes', mode='lines+markers', line={'color':'red'}))

         # Layout de aprasentação
         fig.update_layout(title='Mortes por COVID-19 no Brasil')

         # Exibir gráfico
         fig.show()
```



Taxa de crescimento COVID-19 no Brasil

Taxa média de crescimento de COVID-19

In [28]:

```
# taxa_crescimento = (presente/passado)**(1/n)-1
def taxa_crescimento(data, variable, data_inicio=None, data_fim=None):
    # Se data inicio for None, define como a primeira data disponível
    if data_inicio == None:
        data_inicio = data.observationdate.loc[data[variable] > 0].min()
    else:
        data_inicio = pd.to_datetime(data_inicio)
    # Se data fim for None, define como a ultima data disponível
    if data_fim == None:
        data_fim = data.observationdate.iloc[-1]
    else:
        data_fim = pd.to_datetime(data_fim)

    # Define os valores do presente e passado
    passado = data.loc[data.observationdate == data_inicio, variable].values[0]
    presente = data.loc[data.observationdate == data_fim, variable].values[0]

    # Define o número de pontos no tempo que vamos avaliar
    n = (data_fim - data_inicio).days

    # Calcular a taxa
    taxa = (presente/passado)**(1/n) - 1
    return taxa*100
```

In [29]:

```
# Taxa de crescimento médio do COVID-19 no Brasil em todo o período
taxa_crescimento(brasil, 'confirmed')
```

Out[29]:

```
16.27183353112116
```

Taxa de crescimento diário de COVID-19

In [30]:

```
# taxa_crescimento_diaria = (hoje-ontem)/ontem, (inicio do segundo dia até o ultimo)
def taxa_cresciemnto_diaria (data, variable, data_inicio=None):
    # Se data inicio for None, define como a primeira data disponível
    if data_inicio == None:
        data_inicio = data.observationdate.loc[data[variable] > 0].min()
    else:
        data_inicio = pd.to_datetime(data_inicio)

    data_fim = data.observationdate.max()

    # Define o número de pontos no tempo que vamos avaliar
    n = (data_fim - data_inicio).days

    # Taxa calculada de um dia para o outro
    taxas = list(map( lambda x: (data[variable].iloc[x] - data[variable].iloc[x-1]) / data[variable].iloc[x-1], range(1,n+1)))
    return np.array(taxas) * 100
```

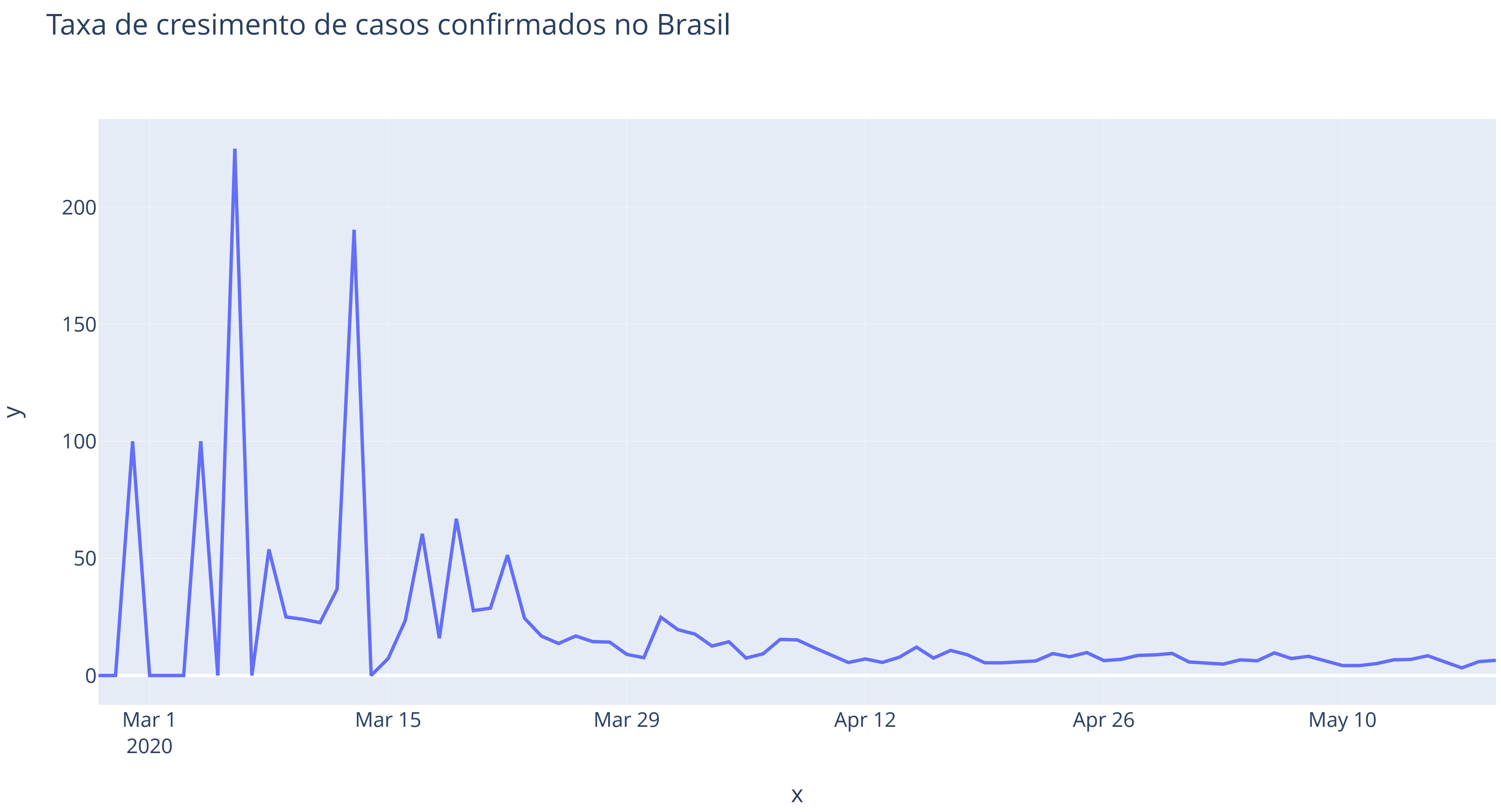
In [31]:

```
# Taxa de crescimento diaria do COVID-19 no Brasil
tx_dia = taxa_cresciemnto_diaria(brasil, 'confirmed')
# Exibir
tx_dia
```

Out[31]:

```
array([  0.        ,   0.        , 100.        ,   0.        ,
         0.        ,   0.        , 100.        ,   0.        ,
       225.        ,   0.        ,  53.84615385,  25.        ,
        24.        ,  22.58064516,  36.84210526, 190.38461538,
         0.        ,   7.28476821,  23.45679012,  60.5       ,
        15.88785047,  66.93548387,  27.69726248,  28.75157629,
        51.4201763 ,  24.45019405,  16.78794179,  13.66266133,
        16.87548943,  14.47236181,  14.25226807,   9.01639344,
         7.58928571,  24.8525879 ,  19.57320273,  17.67115272,
        12.58080557,  14.39929329,   7.43243243,   9.26325247,
        15.40169394,  15.22017956,  11.88620903,   8.54521335,
         5.54537122,   7.06807546,   5.57858688,   7.81903542,
        12.10513815,   7.4329096 ,  10.70501233,   8.83557983,
         5.44492335,   5.4043566 ,   5.73350023,   6.21648599,
         9.35157462,   8.00823407,   9.77184834,   6.36504619,
         6.88748019,   8.58316283,   8.80726429,   9.41456987,
         5.75200431,   5.31224919,   4.86714727,   6.67216624,
         6.29257964,   9.66263912,   7.23633807,   8.19087742,
         6.24055441,   4.25346499,   4.23788714,   5.08272698,
         6.69027125,   6.85190152,   8.42960156,   6.00115302,
         3.24138906,   5.92666335,   6.4679208 ])
```

In [32]:
```python
# Definir o primeiro dia
primeiro_dia = brasil.observationdate.loc[brasil.confirmed > 0].min()
# Gráfico de linha da taxa de crescimento de casos confirmados no Brasil
px.line(x=pd.date_range(primeiro_dia, brasil.observationdate.max())[1:], y=tx_dia, title='Taxa de cresimento de casos confirmados
```



Previsão para COVID-19 no Brasil

In [33]:

```
# Importar novas bibliotecas
from statsmodels.tsa.seasonal import seasonal_decompose
import matplotlib.pyplot as plt
```

In [34]:

```
# Exibir data e casos confirmados de COVID-19
confirmados = brasil.confirmed
confirmados.index = brasil.observationdate
confirmados
```

Out[34]:

```
observationdate
2020-02-26         1.0
2020-02-27         1.0
2020-02-28         1.0
2020-02-29         2.0
2020-03-01         2.0
                ...
2020-05-15    220291.0
2020-05-16    233511.0
2020-05-17    241080.0
2020-05-18    255368.0
2020-05-19    271885.0
Name: confirmed, Length: 84, dtype: float64
```

In [35]:

```
# Decompor os casos confirmados de COVID-19
res = seasonal_decompose(confirmados)
res
```

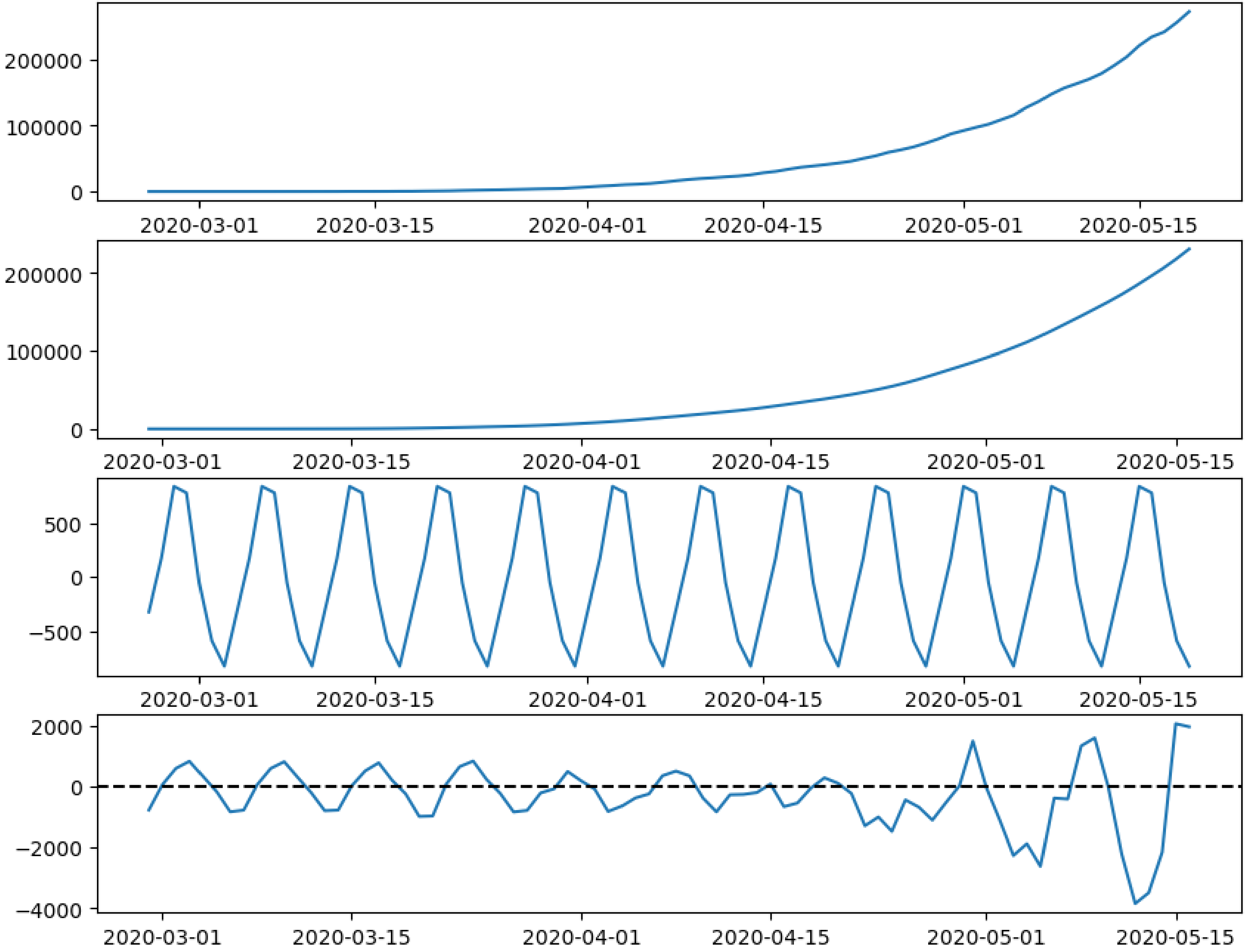
Out[35]:

```
<statsmodels.tsa.seasonal.DecomposeResult at 0x1d21c3da00>
```

In [36]:

```
# Gerar 4 gráficos (observados, tendência, sazonalidade, ruído)
fig, (ax1, ax2, ax3, ax4) = plt.subplots(4, 1, figsize=(10,8))

ax1.plot(res.observed) # observados
ax2.plot(res.trend) # tendência
ax3.plot(res.seasonal) # sazonalidade
ax4.plot(confirmados.index, res.resid) # ruÍdo (resido)
ax4.axhline(0, linestyle='dashed', c='black')
# Exibir gráfico
plt.show()
```



Modelo ARIMA (Autoregressive Integrated Moving Average)

Modelo autorregressivo integrado de média móvel

In [37]:

```
# Instalando biblioteca ARIMA
!pip install pmdarima
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pmdarima in c:\users\décio\appdata\roaming\python\python39\site-packages (2.0.1)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in c:\programdata\anaconda3\lib\site-packages (from pmdarima) (0.
29.32)
Requirement already satisfied: numpy>=1.21 in c:\programdata\anaconda3\lib\site-packages (from pmdarima) (1.21.5)
Requirement already satisfied: joblib>=0.11 in c:\programdata\anaconda3\lib\site-packages (from pmdarima) (1.1.0)
Requirement already satisfied: scikit-learn>=0.22 in c:\programdata\anaconda3\lib\site-packages (from pmdarima) (1.0.2)
Requirement already satisfied: urllib3 in c:\programdata\anaconda3\lib\site-packages (from pmdarima) (1.26.11)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in c:\programdata\anaconda3\lib\site-packages (from pmdarima) (63.4.1)
Requirement already satisfied: pandas>=0.19 in c:\programdata\anaconda3\lib\site-packages (from pmdarima) (1.4.4)
Requirement already satisfied: scipy>=1.3.2 in c:\programdata\anaconda3\lib\site-packages (from pmdarima) (1.9.1)
Requirement already satisfied: statsmodels>=0.13.2 in c:\programdata\anaconda3\lib\site-packages (from pmdarima) (0.13.2)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\programdata\anaconda3\lib\site-packages (from pandas>=0.19->pmdarima)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\anaconda3\lib\site-packages (from pandas>=0.19->pmdarima) (2022.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn>=0.22->pmdar
ima) (2.2.0)
Requirement already satisfied: patsy>=0.5.2 in c:\programdata\anaconda3\lib\site-packages (from statsmodels>=0.13.2->pmdarima)
(0.5.2)
Requirement already satisfied: packaging>=21.3 in c:\programdata\anaconda3\lib\site-packages (from statsmodels>=0.13.2->pmdarima)
(21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\programdata\anaconda3\lib\site-packages (from packaging>=21.3->stat
smodels>=0.13.2->pmdarima) (3.0.9)
Requirement already satisfied: six in c:\programdata\anaconda3\lib\site-packages (from patsy>=0.5.2->statsmodels>=0.13.2->pmdarim
a) (1.16.0)
```

In [38]:

```
# Importar nova biblioteca
from pmdarima.arima import auto_arima
# Modelo a ser criado ajusta a modelagem do pacote ARIMA automaticamente
modelo = auto_arima(confirmados)
modelo
```

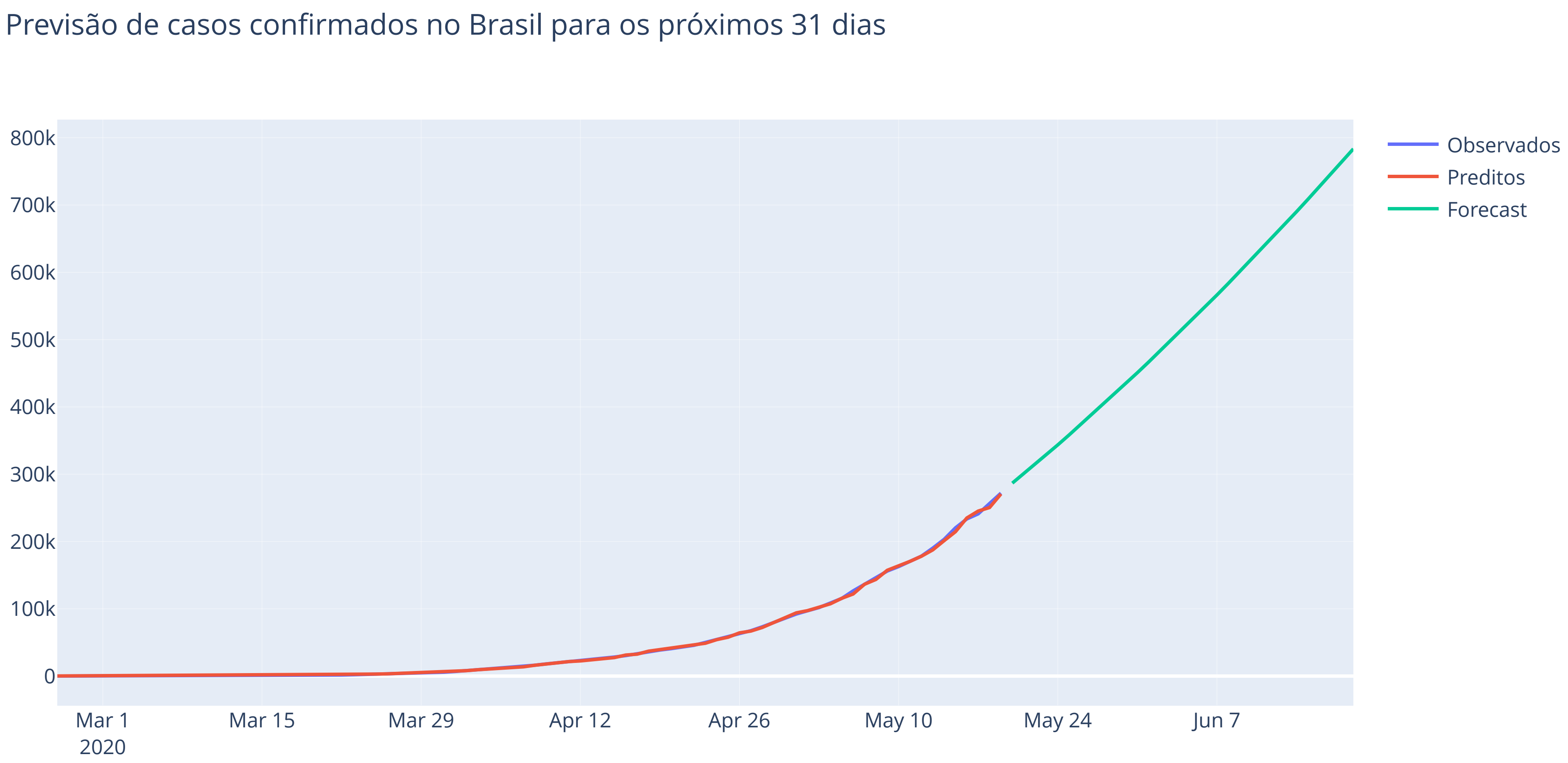
Out[38]:

```
ARIMA(order=(2, 2, 1), scoring_args={}, suppress_warnings=True)
```

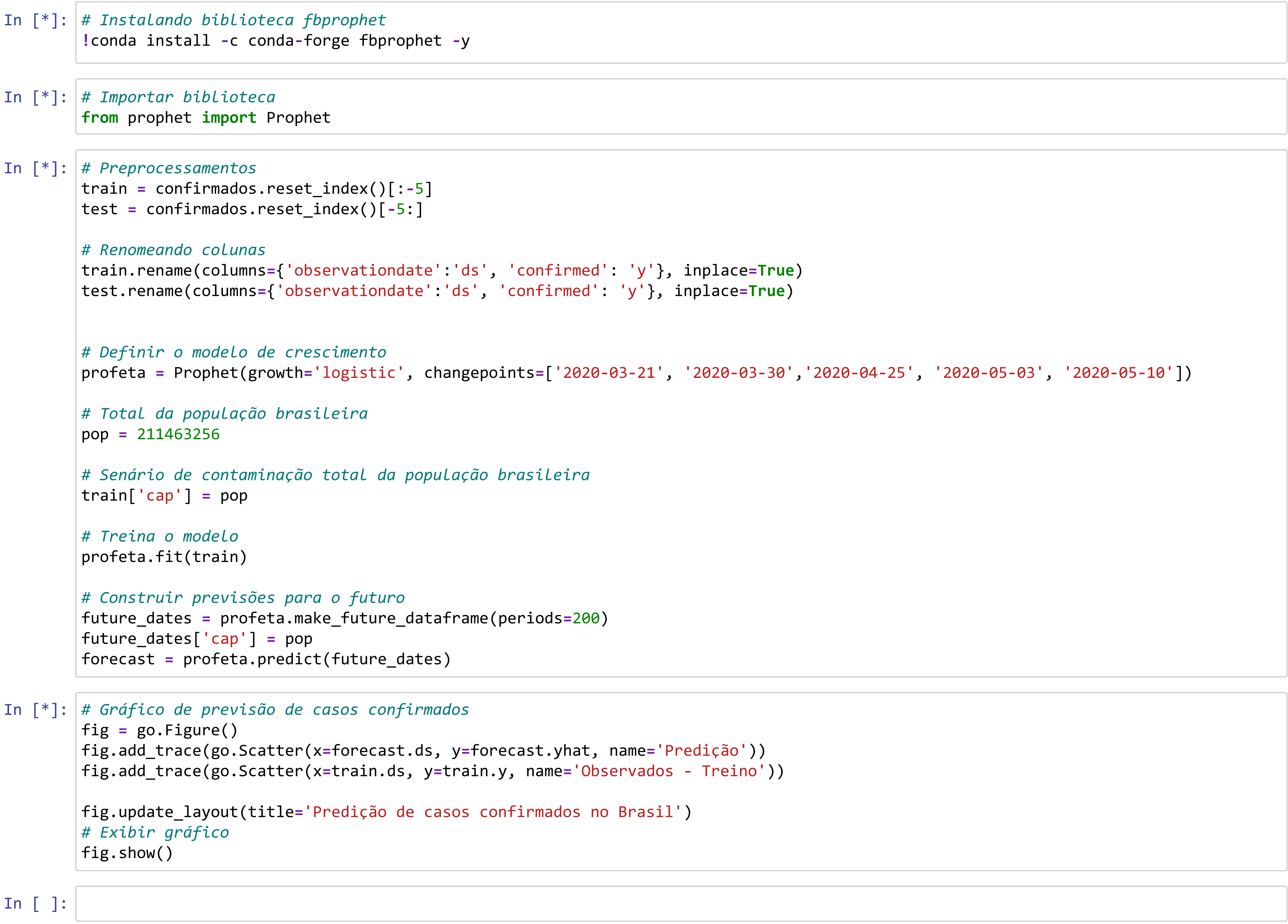
In [39]:

```python
# Gráfico de previsão para 31 dias afrente na pandemia de COVID-19
fig = go.Figure(go.Scatter(x=confirmados.index, y=confirmados, name='Observados'))
fig.add_trace(go.Scatter(x=confirmados.index, y=modelo.predict_in_sample(), name='Preditos'))
fig.add_trace(go.Scatter(x=pd.date_range('2020-05-20', '2020-06-20'), y=modelo.predict(31), name='Forecast'))

fig.update_layout(title='Previsão de casos confirmados no Brasil para os próximos 31 dias')
# Exibir gráfico
fig.show()
```


Previsão de casos confirmados no Brasil para os próximos 31 dias

Modelo de crescimento



```
In [*]: # Instalando biblioteca fbprophet
        !conda install -c conda-forge fbprophet -y
```

```
In [*]: # Importar biblioteca
        from prophet import Prophet
```

```
In [*]: # Preprocessamentos
        train = confirmados.reset_index()[:-5]
        test = confirmados.reset_index()[-5:]

        # Renomeando colunas
        train.rename(columns={'observationdate':'ds', 'confirmed': 'y'}, inplace=True)
        test.rename(columns={'observationdate':'ds', 'confirmed': 'y'}, inplace=True)


        # Definir o modelo de crescimento
        profeta = Prophet(growth='logistic', changepoints=['2020-03-21', '2020-03-30','2020-04-25', '2020-05-03', '2020-05-10'])

        # Total da população brasileira
        pop = 211463256

        # Senário de contaminação total da população brasileira
        train['cap'] = pop

        # Treina o modelo
        profeta.fit(train)

        # Construir previsões para o futuro
        future_dates = profeta.make_future_dataframe(periods=200)
        future_dates['cap'] = pop
        forecast = profeta.predict(future_dates)
```

```
In [*]: # Gráfico de previsão de casos confirmados
        fig = go.Figure()
        fig.add_trace(go.Scatter(x=forecast.ds, y=forecast.yhat, name='Predição'))
        fig.add_trace(go.Scatter(x=train.ds, y=train.y, name='Observados - Treino'))

        fig.update_layout(title='Predição de casos confirmados no Brasil')
        # Exibir gráfico
        fig.show()
```

```
In [ ]:
```