**Análise COVID-19**

**Bibliotecas**

**import** re

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** plotly.express **as** px

**import** plotly.graph\_objects **as** go

**from** datetime **import** datetime

**DataFrame**

|  | **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

df**.**info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 26713 entries, 0 to 26712

Data columns (total 8 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 SNo 26713 non-null int64

1 ObservationDate 26713 non-null datetime64[ns]

2 Province/State 12882 non-null object

3 Country/Region 26713 non-null object

4 Last Update 26713 non-null datetime64[ns]

5 Confirmed 26713 non-null float64

6 Deaths 26713 non-null float64

7 Recovered 26713 non-null float64

dtypes: datetime64[ns](2), float64(3), int64(1), object(2)

memory usage: 1.6+ MB

df**.**describe()**.**drop(columns**=**'SNo')**.**T

|  | **count** | **mean** | **std** | **min** | **25%** | **50%** | **75%** | **max** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Confirmed** | 26713.0 | 5690.647550 | 23417.369124 | 0.0 | 18.0 | 192.0 | 1350.0 | 352845.0 |
| **Deaths** | 26713.0 | 373.353236 | 2214.074387 | 0.0 | 0.0 | 3.0 | 28.0 | 35341.0 |
| **Recovered** | 26713.0 | 1736.979224 | 10864.727709 | 0.0 | 0.0 | 8.0 | 187.0 | 289392.0 |

df**.**describe(include**=**'O')**.**T

|  | **count** | **unique** | **top** | **freq** |
| --- | --- | --- | --- | --- |
| **Province/State** | 12882 | 354 | Diamond Princess cruise ship | 175 |
| **Country/Region** | 26713 | 223 | US | 4990 |

*#Alterar nomes das colunas:*

df**.**columns **=** [re**.**sub(r'[/ ]', '\_', col)**.**lower() **for** col **in** df**.**columns]

df**.**sample(4)

|  | **sno** | **observationdate** | **province\_state** | **country\_region** | **last\_update** | **confirmed** | **deaths** | **recovered** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **826** | 827 | 2020-02-05 | NaN | Nepal | 2020-01-31 08:15:53 | 1.0 | 0.0 | 0.0 |
| **17356** | 17357 | 2020-04-21 | Vermont | US | 2020-04-21 23:39:09 | 818.0 | 40.0 | 0.0 |
| **1904** | 1905 | 2020-02-20 | NaN | Germany | 2020-02-18 17:03:03 | 16.0 | 0.0 | 12.0 |
| **1874** | 1875 | 2020-02-20 | Jiangsu | Mainland China | 2020-02-20 12:53:02 | 631.0 | 0.0 | 356.0 |

**Análise dos casos de Covid-19 no Brasil**

*#df.loc[df.country\_region == 'Brazil', 'province\_state'].notnull().sum()*

df**.**loc[df**.**country\_region **==** 'Brazil']**.**notnull()**.**sum()

sno 85

observationdate 85

province\_state 0

country\_region 85

last\_update 85

confirmed 85

deaths 85

recovered 85

dtype: int64

df\_brasil **=** df**.**loc[df**.**country\_region **==** 'Brazil']**.**drop(columns**=**['province\_state','sno'])

df\_brasil

|  | **observationdate** | **country\_region** | **last\_update** | **confirmed** | **deaths** | **recovered** |
| --- | --- | --- | --- | --- | --- | --- |
| **82** | 2020-01-23 | Brazil | 2020-01-23 17:00:00 | 0.0 | 0.0 | 0.0 |
| **2455** | 2020-02-26 | Brazil | 2020-02-26 23:53:02 | 1.0 | 0.0 | 0.0 |
| **2559** | 2020-02-27 | Brazil | 2020-02-26 23:53:02 | 1.0 | 0.0 | 0.0 |
| **2668** | 2020-02-28 | Brazil | 2020-02-26 23:53:02 | 1.0 | 0.0 | 0.0 |
| **2776** | 2020-02-29 | Brazil | 2020-02-29 21:03:05 | 2.0 | 0.0 | 0.0 |
| **...** | ... | ... | ... | ... | ... | ... |
| **24850** | 2020-05-15 | Brazil | 2020-05-16 02:32:19 | 220291.0 | 14962.0 | 84970.0 |
| **25227** | 2020-05-16 | Brazil | 2020-05-17 02:32:32 | 233511.0 | 15662.0 | 89672.0 |
| **25604** | 2020-05-17 | Brazil | 2020-05-18 02:32:21 | 241080.0 | 16118.0 | 94122.0 |
| **25981** | 2020-05-18 | Brazil | 2020-05-19 02:32:18 | 255368.0 | 16853.0 | 100459.0 |
| **26358** | 2020-05-19 | Brazil | 2020-05-20 02:32:19 | 271885.0 | 17983.0 | 106794.0 |

85 rows × 6 columns

**Número de casos confirmados**

df\_brasil **=** df\_brasil[df\_brasil**.**confirmed **>** 0]

df\_brasil**.**shape

(84, 6)

fig **=** px**.**line(df\_brasil, x**=**'observationdate', y**=**'confirmed',

labels**=**{'observationdate':'Data', 'confirmed':'Número de casos confirmados'},

title**=**'Casos confirmados no Brasil', width**=**1000, height**=**400)

fig**.**update\_layout(

margin**=**dict(l**=**30, r**=**20, t**=**60, b**=**5),

font**=**dict(size**=**15, color**=**'black')

)

fig**.**show()

**Número de novos casos por dia**

*#Função para fazer a contagem de novos casos:*

*#[Subtração entre o número de casos de um dia e o dia anterior]*

**def** dif(v):

J**=**[v[i**+**1]**-**v[i] **for** i **in** range(len(v)**-**1)]

J**.**insert(0, v[0])

**return** J

**def** dif2(v):

J**=**[v[0]]

**for** i **in** range(len(v)**-**1):

J**.**append(v[i**+**1]**-**v[i])

**return** np**.**array(J)

df\_brasil **=** df\_brasil**.**assign( novoscasos**=**dif(df\_brasil['confirmed']**.**values) )

df\_brasil

Out[14]:

|  | **observationdate** | **country\_region** | **last\_update** | **confirmed** | **deaths** | **recovered** | **novoscasos** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **2455** | 2020-02-26 | Brazil | 2020-02-26 23:53:02 | 1.0 | 0.0 | 0.0 | 1.0 |
| **2559** | 2020-02-27 | Brazil | 2020-02-26 23:53:02 | 1.0 | 0.0 | 0.0 | 0.0 |
| **2668** | 2020-02-28 | Brazil | 2020-02-26 23:53:02 | 1.0 | 0.0 | 0.0 | 0.0 |
| **2776** | 2020-02-29 | Brazil | 2020-02-29 21:03:05 | 2.0 | 0.0 | 0.0 | 1.0 |
| **2903** | 2020-03-01 | Brazil | 2020-02-29 21:03:05 | 2.0 | 0.0 | 0.0 | 0.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **24850** | 2020-05-15 | Brazil | 2020-05-16 02:32:19 | 220291.0 | 14962.0 | 84970.0 | 17126.0 |
| **25227** | 2020-05-16 | Brazil | 2020-05-17 02:32:32 | 233511.0 | 15662.0 | 89672.0 | 13220.0 |
| **25604** | 2020-05-17 | Brazil | 2020-05-18 02:32:21 | 241080.0 | 16118.0 | 94122.0 | 7569.0 |
| **25981** | 2020-05-18 | Brazil | 2020-05-19 02:32:18 | 255368.0 | 16853.0 | 100459.0 | 14288.0 |
| **26358** | 2020-05-19 | Brazil | 2020-05-20 02:32:19 | 271885.0 | 17983.0 | 106794.0 | 16517.0 |

84 rows × 7 columns

In [15]:

*#Visualizando*

fig **=** px**.**line(df\_brasil, x**=**'observationdate', y**=**'novoscasos', title**=**'Novos casos por dia',

labels**=**{'observationdate':'Data', 'novoscasos':'Novos casos'}, width**=**1000, height**=**400)

fig**.**update\_layout(

margin**=**dict(l**=**20, r**=**20, t**=**60, b**=**0),

*#font\_family="Times New Roman",*

font**=**dict(

family**=**"Times New Roman", size**=**18

)

*#paper\_bgcolor="LightSteelBlue"*

)

fig**.**show()

**Número de mortes**

fig **=** go**.**Figure()

fig**.**add\_trace(

go**.**Scatter(x**=**df\_brasil**.**observationdate, y**=**df\_brasil**.**deaths,

name**=**'Mortes', mode**=**'lines+markers', line**=**dict(color**=**'red'))

)

fig**.**update\_layout(title**=**'Mortes por COVID-19 no Brasil',

xaxis\_title**=**'Data', yaxis\_title**=**'Número de mortes',

margin**=**dict(l**=**30, r**=**30, t**=**50, b**=**5),

width**=**1000, height**=**400, font**=**dict(size**=**16))

fig**.**show()

**Taxa *média* de crescimento (por dia)**

Vamos calcular a taxa de crescimento médio do COVID desde o primeiro caso.

étaxa média de crescimento=x1/x0n−1

**def** taxa\_crescimento\_medio(dados, variable, data\_inicio**=None**, data\_fim**=None**):

*#Se data\_inicio for None, define como a primeira data disponível no dataset:*

**if** data\_inicio **==** **None**:

data\_inicio **=** dados**.**observationdate**.**loc[dados[variable] **>** 0]**.**min()

**else**:

data\_inicio **=** pd**.**to\_datetime(data\_inicio)

**if** data\_fim **==** **None**:

data\_fim **=** dados**.**observationdate**.**iloc[**-**1]

**else**:

data\_fim **=** pd**.**to\_datetime(data\_fim)

*#Define os valores inicial e final:*

ini **=** dados**.**loc[dados**.**observationdate **==** data\_inicio, variable]**.**values[0]

fim **=** dados**.**loc[dados**.**observationdate **==** data\_fim, variable]**.**values[0]

*#Define o número de pontos no tempo que vamos avaliar:*

n **=** (data\_fim **-** data\_inicio)**.**days

*#Calcula a taxa:*

taxa **=** (fim**/**ini)**\*\***(1**/**n) **-** 1

print(f'Período de {n} dias.')

**return** taxa**\***100

cresc\_medio **=** taxa\_crescimento\_medio(df\_brasil, 'confirmed', '2020-03-15')

print(f'O crescimento médio dos casos confimados de COVID no Brasil no período avaliado foi de {cresc\_medio**.**round(2)}%.')

Período de 65 dias.

O crescimento médio dos casos confimados de COVID no Brasil no período avaliado foi de 12.1%.

cresc\_medio **=** taxa\_crescimento\_medio(df\_brasil, 'confirmed')

print(f'O crescimento médio dos casos confimados de COVID no Brasil no período avaliado foi de {cresc\_medio**.**round(2)}%.')

Período de 83 dias.

O crescimento médio dos casos confimados de COVID no Brasil no período avaliado foi de 16.27%.

Agora, vamos observar o comportamento da **taxa de crescimento no tempo**. Para isso, vamos definir uma função para calcular a taxa de crescimento diária.

taxa de crescimento=x1x0−1

**def** taxa\_crescimento\_diario(dados, variable, data\_inicio**=None**):

**if** data\_inicio **==** **None**:

data\_inicio **=** dados**.**observationdate**.**loc[dados[variable] **>** 0]**.**min()

**else**:

data\_inicio **=** pd**.**to\_datetime(data\_inicio)

data\_fim **=** dados**.**observationdate**.**max()

n **=** (data\_fim **-** data\_inicio)**.**days

taxas **=** list(map( **lambda** x: (dados[variable]**.**iloc[x]**/**dados[variable]**.**iloc[x**-**1]) **-** 1,

range(0**+**len(dados[variable])**-**n,len(dados[variable])) ))

**return** np**.**array(taxas)*#\*100*

**def** Grafic\_taxa\_crescimento(inicio):

taxa\_dia **=** taxa\_crescimento\_diario(df\_brasil, 'confirmed', inicio)

fig **=** go**.**Figure()

fig**.**add\_trace(

go**.**Scatter(x**=**pd**.**date\_range(inicio, df\_brasil**.**observationdate**.**max())[1:], y**=**taxa\_dia,

name**=**'Taxa', mode**=**'lines+markers', line**=**dict(color**=**'red'), marker**=**{'color':'black'})

)

fig**.**update\_layout(title**=**'Taxa de crescimento de casos confirmados no Brasil',

xaxis\_title**=**'Data', yaxis\_title**=**'Taxa de crescimento',

margin**=**dict(l**=**50, r**=**30, t**=**50, b**=**5),

width**=**1100, height**=**400, font**=**dict(size**=**16))

fig**.**show()

*#Grafic\_taxa\_crescimento(df\_brasil.observationdate.min())*

Grafic\_taxa\_crescimento('2020-03-14')

*#taxa\_dia = taxa\_crescimento\_diario(df\_brasil, 'confirmed', '2020-03-15')*

*#taxa\_dia = taxa\_crescimento\_diario(df\_brasil, 'confirmed', '2020-05-18')*

taxa\_dia **=** taxa\_crescimento\_diario(df\_brasil, 'confirmed')

taxa\_dia

*#Observe que as taxas diárias de crescimento aplicadas à população inicial*

*#fornece exatamente a população final, como esperado.*

pop\_inicial**=**1

pop\_final **=** pop\_inicial

**for** i **in** taxa\_dia:

pop\_final **\*=** (1**+**i)

print( f'{pop\_final:.1f} == {df\_brasil**.**confirmed**.**iloc[**-**1]}' )

271885.0 == 271885.0

### Predições de Séries Temporais

Vamos construir um modelo de séries temporais para prever os novos casos. Antes analisaremos a série temporal.

In [25]:

*#Instalar o pacote pmdarima antes de importar a biblioteca statsmodels, senão tem que reiniciar o notebook.*

**!**pip install pmdarima

Collecting pmdarima

Downloading pmdarima-1.8.2-cp37-cp37m-manylinux1\_x86\_64.whl (1.5 MB)

| 1.5 MB 4.3 MB/s

Collecting statsmodels!=0.12.0,>=0.11

Downloading statsmodels-0.12.2-cp37-cp37m-manylinux1\_x86\_64.whl (9.5 MB)

9.5 MB 33.1 MB/s

Requirement already satisfied: Cython!=0.29.18,>=0.29 in /usr/local/lib/python3.7/dist-packages (from pmdarima) (0.29.24)

Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.7/dist-packages (from pmdarima) (1.1.5)

Requirement already satisfied: numpy~=1.19.0 in /usr/local/lib/python3.7/dist-packages (from pmdarima) (1.19.5)

Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.7/dist-packages (from pmdarima) (0.22.2.post1)

Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.7/dist-packages (from pmdarima) (1.4.1)

Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from pmdarima) (1.24.3)

Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from pmdarima) (1.0.1)

Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /usr/local/lib/python3.7/dist-packages (from pmdarima) (57.4.0)

Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.19->pmdarima) (2.8.2)

Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.19->pmdarima) (2018.9)

Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas>=0.19->pmdarima) (1.15.0)

Requirement already satisfied: patsy>=0.5 in /usr/local/lib/python3.7/dist-packages (from statsmodels!=0.12.0,>=0.11->pmdarima) (0.5.1)

Installing collected packages: statsmodels, pmdarima

Attempting uninstall: statsmodels

Found existing installation: statsmodels 0.10.2

Uninstalling statsmodels-0.10.2:

Successfully uninstalled statsmodels-0.10.2

Successfully installed pmdarima-1.8.2 statsmodels-0.12.2

**from** statsmodels.tsa.seasonal **import** seasonal\_decompose

**def** Grafic\_decompose(serie):

res **=** seasonal\_decompose(serie)

fig, (ax1,ax2,ax3,ax4) **=** plt**.**subplots(4,1,figsize**=**(8,8))

ax1**.**plot(res**.**observed)

ax1**.**set\_title('Série original')

ax2**.**plot(res**.**trend)

ax2**.**set\_title('Tendência', pad**=**5)

ax3**.**plot(res**.**seasonal)

ax3**.**set\_title('Sazonalidade')

ax4**.**scatter(novoscasos**.**index, res**.**resid)

ax4**.**plot(novoscasos**.**index, res**.**resid)

ax4**.**axhline(0, linestyle**=**'dashed', c**=**'black')

ax4**.**set\_title('Resíduos')

fig**.**tight\_layout(pad**=**0.7)

plt**.**show()

Decompondo a série de novos casos

novoscasos **=** df\_brasil**.**novoscasos

novoscasos**.**index **=** df\_brasil**.**observationdate

Grafic\_decompose(novoscasos)

Decompondo a série de casos confirmados

confirmados **=** df\_brasil**.**confirmed

confirmados**.**index **=** df\_brasil**.**observationdate

Grafic\_decompose(confirmados)

#### Predizendo o número de casos confirmados com um AUTO-ARIMA

**from** pmdarima.arima **import** auto\_arima

fig **=** go**.**Figure(go**.**Scatter(

x**=**confirmados**.**index, y**=**confirmados, name**=**'Observed', mode**=**'lines+markers'

))

fig**.**add\_trace(go**.**Scatter(x**=**confirmados**.**index, y **=** modelo**.**predict\_in\_sample(), name**=**'Predicted'))

data\_inicial**=**pd**.**to\_datetime('2020-05-20')

data\_final **=**pd**.**to\_datetime('2020-06-04')

n **=** (data\_final **-** data\_inicial)**.**days

fig**.**add\_trace(go**.**Scatter(x**=**pd**.**date\_range(data\_inicial, data\_final), y**=**modelo**.**predict(n), name**=**'Forecast'))

fig**.**update\_layout(title**=**f'Previsão de casos confirmados para os próximos {n} dias',

yaxis\_title**=**'Casos confirmados', xaxis\_title**=**'Data',

margin**=**dict(l**=**30, r**=**30, t**=**50, b**=**5),

width**=**1100, height**=**400, font**=**dict(size**=**14))

fig**.**show()

#### Forecasting com Prophet

**!**pip3 install fbprophet

Requirement already satisfied: fbprophet in /usr/local/lib/python3.7/dist-packages (0.7.1)

Requirement already satisfied: Cython>=0.22 in /usr/local/lib/python3.7/dist-packages (from fbprophet) (0.29.24)

Requirement already satisfied: cmdstanpy==0.9.5 in /usr/local/lib/python3.7/dist-packages (from fbprophet) (0.9.5)

Requirement already satisfied: pystan>=2.14 in /usr/local/lib/python3.7/dist-packages (from fbprophet) (2.19.1.1)

Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.7/dist-packages (from fbprophet) (1.19.5)

Requirement already satisfied: pandas>=1.0.4 in /usr/local/lib/python3.7/dist-packages (from fbprophet) (1.1.5)

Requirement already satisfied: matplotlib>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from fbprophet) (3.2.2)

Requirement already satisfied: LunarCalendar>=0.0.9 in /usr/local/lib/python3.7/dist-packages (from fbprophet) (0.0.9)

Requirement already satisfied: convertdate>=2.1.2 in /usr/local/lib/python3.7/dist-packages (from fbprophet) (2.3.2)

Requirement already satisfied: holidays>=0.10.2 in /usr/local/lib/python3.7/dist-packages (from fbprophet) (0.10.5.2)

Requirement already satisfied: setuptools-git>=1.2 in /usr/local/lib/python3.7/dist-packages (from fbprophet) (1.2)

Requirement already satisfied: python-dateutil>=2.8.0 in /usr/local/lib/python3.7/dist-packages (from fbprophet) (2.8.2)

Requirement already satisfied: tqdm>=4.36.1 in /usr/local/lib/python3.7/dist-packages (from fbprophet) (4.62.0)

Requirement already satisfied: pymeeus<=1,>=0.3.13 in /usr/local/lib/python3.7/dist-packages (from convertdate>=2.1.2->fbprophet) (0.5.11)

Requirement already satisfied: pytz>=2014.10 in /usr/local/lib/python3.7/dist-packages (from convertdate>=2.1.2->fbprophet) (2018.9)

Requirement already satisfied: hijri-converter in /usr/local/lib/python3.7/dist-packages (from holidays>=0.10.2->fbprophet) (2.1.3)

Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from holidays>=0.10.2->fbprophet) (1.15.0)

Requirement already satisfied: korean-lunar-calendar in /usr/local/lib/python3.7/dist-packages (from holidays>=0.10.2->fbprophet) (0.2.1)

Requirement already satisfied: ephem>=3.7.5.3 in /usr/local/lib/python3.7/dist-packages (from LunarCalendar>=0.0.9->fbprophet) (4.0.0.2)

Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=2.0.0->fbprophet) (0.10.0)

Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=2.0.0->fbprophet) (1.3.1)

Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=2.0.0->fbprophet) (2.4.7)

I

**from** fbprophet **import** Prophet

*#preparando os dados:*

train **=** confirmados**.**reset\_index()[:**-**5]

test **=** confirmados**.**reset\_index()[**-**5:]

*#renomear colunas:*

train**.**rename(columns**=**{'observationdate':'ds','confirmed':'y'}, inplace**=True**)

test**.**rename( columns**=**{'observationdate':'ds','confirmed':'y'}, inplace**=True**)

test **=** test**.**set\_index('ds')

test **=** test['y']

*#Supondo que toda a população seja infectada:*

*#https://www.ibge.gov.br/apps/populacao/projecao/box\_popclock.php*

pop **=** 213481773

*#pop = 1000000*

train['cap'] **=** pop

train

|  | **ds** | **y** | **cap** |
| --- | --- | --- | --- |
| **0** | 2020-02-26 | 1.0 | 213481773 |
| **1** | 2020-02-27 | 1.0 | 213481773 |
| **2** | 2020-02-28 | 1.0 | 213481773 |
| **3** | 2020-02-29 | 2.0 | 213481773 |
| **4** | 2020-03-01 | 2.0 | 213481773 |
| **...** | ... | ... | ... |
| **74** | 2020-05-10 | 162699.0 | 213481773 |
| **75** | 2020-05-11 | 169594.0 | 213481773 |
| **76** | 2020-05-12 | 178214.0 | 213481773 |
| **77** | 2020-05-13 | 190137.0 | 213481773 |
| **78** | 2020-05-14 | 203165.0 | 213481773 |

79 rows × 3 columns

*#Modelo:*

profeta **=** Prophet(growth**=**"logistic", changepoints**=**['2020-03-21', '2020-03-30', '2020-04-25', '2020-05-03', '2020-05-10'])

*#Treinar o modelo:*

profeta**.**fit(train)

*#Construindo previsões para o futuro:*

future\_dates **=** profeta**.**make\_future\_dataframe(periods**=**200)

future\_dates['cap'] **=** pop

forecast **=** profeta**.**predict(future\_dates)

INFO:fbprophet:Disabling yearly seasonality. Run prophet with yearly\_seasonality=True to override this.

INFO:fbprophet:Disabling daily seasonality. Run prophet with daily\_seasonality=True to override this.

fig **=** go**.**Figure()

fig**.**add\_trace(go**.**Scatter(x**=**forecast**.**ds, y**=**forecast**.**yhat, name**=**'Predição'))

fig**.**add\_trace(go**.**Scatter(x**=**test**.**index, y**=**test, name**=**'Observados - Teste'))

fig**.**add\_trace(go**.**Scatter(x**=**train**.**ds, y**=**train**.**y, name**=**'Observados - Treino'))

fig**.**update\_layout(title**=**'Predições de casos confirmados no Brasil (supondo que toda a população seja infectada)',

yaxis\_title**=**'Casos confirmados', xaxis\_title**=**'Data',

margin**=**dict(l**=**30, r**=**30, t**=**80, b**=**5), width**=**1100, height**=**400, font**=**dict(size**=**14))

fig**.**show()