Projeto Métodos Numéricos: Estimimar a Progressão da Ômicron no Estado de PE

Equipe:

João Pedro Ribeiro da Silva Dias - jprsd

Rodrigo Santos Batista - rsb6

Vinícius Sales Oliveira - vso2

Yasmin Maria Wanderley Soares - ymws

Resumo

(1) A variante Ômicron do SARS-CoV-2 pode já ser o vírus de mais rápida propagação de toda a história. A informação foi dada pelo médico infectologista norte-americano Roby Bhattacharyya do Hospital Geral de Massachusetts. A nova cepa é dominante em várias nações do mundo e está levando à explosão do número de casos de covid-19.

O médico e pesquisador fez um cálculo entre a Ômicron e o sarampo, um dos vírus mais contagiosos. Ele concluiu que, num cenário de ausência de vacinação, um caso de sarampo daria origem a mais 15 casos em apenas 12 dias. Já um caso de Ômicron daria origem a 216 casos no mesmo período. A estimativa significa que, em 35 dias, a Ômicron poderia atingir 280 mil pessoas, enquanto o sarampo afetaria 2.700.

No entanto, num cenário em que a maioria da população está vacinada ou já teve covid-19, o especialista estima que um caso de Ômicron dê origem a apenas mais três casos, número semelhante ao do vírus original, ausente de mutações.

Nesse sentido, tornou-se pertinente analisar nesse trabalho um cenário onde há persistencia viral numa dada população, tendo em vista suas variantes. Para efeito de simplicidade, trabalharemos, apenas, com a variante mais contagiosa e atual (ômicron), levando em conta os dados locais do estado de Pernambuco.

Partiremos de tres premissas básicas para essa análise: O primeiro caso leva em conta a evolução da doença na taxa atual ao que ela se encontra com a presença da vacinação. Já o segundo caso, é uma análise da doença, caso não houvessem vacinas, e o terceiro caso idealiza um sistema de vacinação mais eficiente e desconsidera a resistência à vacina como elemento profilático e mitigador, fazendo com que mais indivíduos consigam ser vacinados, diminuindo a taxa de infectados e por conseguinte, de óbitos.

Para realizar essas análises, utilizaremos um modelo variante do SIR, o SIRV com acréscimo de um "E" que será a taxa de expostos ao vírus, no qual será analisado o número de indivíduos suscetíveis, infectados, removidos, vacinados e expostos, para estimar a evolução da doença mesmo na presença das vacinas. Assim como a progressão da taxa de vacinados da população no território de Pernambuco. Dessa maneira, o modelo tentará evidenciar que as vacinas, mesmo ainda não tendo 100% de eficácia, são extremamente importantes para achatar a curva de contaminados e mortos, assim como para a mitigar os transtornos causados pelo vírus.

Introdução

Coronavírus é um vírus zoonótico, um RNA vírus da ordem Nidovirales, da família Coronaviridae (2). Esta é uma família de vírus que causam infecções respiratórias, os quais foram isolados pela primeira vez em 1937 e descritos como tal em 1965, em decorrência do seu perfil na microscopia assemelhando-se a uma coroa(3). Tendo como parâmetro o ano de 2019, onde ainda não se tinha previsão da proporção da doença e quais as implicações que suas possíveis mutações; foi identificada pela primeira vez em em Wuhan, na província de Hubei, China por Dezembro de 2019. Este vírus causa síndrome respiratória aguda grave que podem se tornar potencialmente fatais.

(4) Em 26 de novembro de 2021 (OMS 2021), a OMS designou a variante SARS-CoV-2 B.1.1.529 como uma variante preocupante (VoC) com base no conselho do Grupo Consultivo Técnico da OMS sobre Evolução de Vírus (VOC). Esta variante recebeu o nome de Omicron. A África do Sul relatou o Omicron à OMS pela primeira vez em 24 de novembro de 2021. Na África do Sul, as infecções foram altas nas últimas semanas, coincidindo com a descoberta do Omicron. No Brasil, a variante Ômicron foi confirmada para o primeira vez no dia 30 de novembro em São Paulo, já em Pernambuco a variante foi detectada entre 26 de novembro de 2021 e 4 de janeiro deste ano.

Objetivos

Tendo em vista o cenário atual em que há muita resistencia à vacina por uma parcela da população e por setores expecíficos do governo, ainda que comprovada sua eficiencia e eficácia. Vamos aqui estabelecer um contraste, projetanto um cenário no qual a existência dela é quase nula ou nenhuma. Dessa manera, será observada a progressão da doênça ao longo do tempo e uma análise em cima do gráfico será efetuada para fins de comparação, caso dispuséssimos de uma maior taxa de vacinação.

Métodologia

(9) O modelo matemático SIR (suscetível, infectados e removidos) é largamente utilizado para simular uma pandemia, por ser um modelo simples. Então, esse trabalho contará com o suporte de uma variante desse modelo: O SIRV + E. Essa variante também leva em consideração a população dos vacinados + espostos. Vamos assumir aqui, que a população do estado de Pernambuco permanece constante ao longo do tempo, dessa maneira, as equações serão dadas da seguinte forma:

O modelo matemático SIR (suscetível, infectados e removidos) é o mais simples para simular uma pandemia, nosso projeto utilizará uma variante desse modelo: O SIRV. Essa variante também leva em consideração a população dos vacinados. Assumindo que a população de Pernambuco permanece constante com o tempo, temos as seguintes equações:

$$N = S(t) + I(t) + R(t) + V(t)$$

$$0 = S' + I' + R' + V'$$

$$\frac{dS(t)}{dt} = \omega r R(t) - \lambda S(t) + \omega v V(t) - \phi S(t)$$

$$\frac{dV(t)}{dt} = \phi S(t) - \omega v V(t) - (1 - \psi)\lambda V(t)$$

$$\frac{dI(t)}{dt} = \lambda S(t) + (1 - \psi)\lambda V(t) - \gamma I(t)$$

$$\frac{dR(t)}{dt} = \gamma I(t) - \omega r R(t)$$

Onde:

- N é a população total
- S(t): representa a população suscetível a ter a doença no tempo t, ou seja, é a população que não foi infectada e que não possui imunização.
- I(t): representa a população infectada com o vírus no tempo t, esses são os indivíduos que estão infectados e podem espalhar a doença para a população que está no estado suscetível.
- R(t): representa a população removida no tempo t, dizer que uma população foi removida nesse modelo significa que ou ficou recuperada da doença.
- V(t): representa a população efetivamente vacinada no tempo t. São os indivíduos que foram vacinados e ganharam imunidade total à doença.
- ωr: representa a taxa com a qual os indivíduos recuperados voltam para o grupo dos suscetíveis devido à perda de imunidade.
- ωv:representa a taxa com a qual os indivíduos vacinados voltam para o grupo dos suscetíveis devido à perda de imunidade.

- γ: Representa a taxa de recuperação dos indivíduos infectados por dia, a taxa $1/\gamma$ representa a média do período de infecção da doença.
- ψ: Representa a taxa de vacinação e a eficácia da vacina, é calculada fazendo a fração de indivíduos que ganharam imunização total com todos os que foram vacinados multiplicado pela taxa de vacinação de pernambuco.
- λ: representa a taxa de infecção por pessoa e é dada por $\lambda = \frac{\beta I}{N}$
- Φ: representa a taxa de vacinação

Coleta de dados:

Os dados foram obtidos do Monitora Covid em conjunto com a FioCruz, assim como dados dos IBGE para estimar a população de Pernambuco em Janeiro de 2022. Com isso temos N = 9674793.

Para o número de infectados iniciais coletamos do site "https://dados.seplag.pe.gov.br/apps/corona.html" , que fornece o numero de infectados no mês de Janeiro de 2022.

Como estamos analisando no período de início da vacinação temos que o V0 = 0

Para calcularmos os Removidos iniciais, estamos somando a quantidade de mortos e casos confirmados do dia 1 de janeiro em 2022 de Pernambuco.

Dessa forma, tem-se que o número de suscetíveis será o total da população subtraído com os removidos iniciais, vacinados iniciais e infectados iniciais.

Tem-se que a taxa de transmissão é aproximadamente igual a β = 2,02, para a variante ômicron, pois é mais contagiosa.

Sabendo que o período médio de infecção da Covid- 19 é 14 dias então, $\gamma = 1/14 = 0{,}07$.

Para estimar a taxa de eficácia da vacina se utiliza a média da eficácias de cada vacina (AstraZenica - 91,3%; CoronaVac - 95%; Janssem - 95%; Pfizer - 96%) (6) e portanto, resulta em uma eficacia de δ = 94% ou 0,94.

Para se calcular a taxa de vacinação do estado de Pernambuco divide-se a média de pessoas vacinadas por dia, pela população total.

Para calcular-se a porcentagem de pessoas imunizadas por dia, o γ, multiplica-se a taxa de vacinação do estado pela eficácia da vacina.

Para taxa de mortalidade, foi apenas uma coleta de dados do site oficial de monitoramento de casos (5), cujo resultado é obtido por meio de uma fração do (num de óbitos) / (num de casos confirmados) o que nos dá ξ = 0,009977 (5)

Vacinas

Para analisar cada um dos cénarios que serão modelados.Sabemos que as taxas de edicácia da vacina variam conforme sua produção (CoronaVac, AstraZeneca, Pfizer e Janssen), consideraremos aqui, em busca de uma precisão maior, uma média ponderada das eficácias dessas diferentes variações da vacina do coronavírus. Então, a partir do **acompanhamentos do recebimento das vacinas**[15] disponibilizado pelo governo do estado de Pernambuco.

Cálculo da Eficácia Média $\Longrightarrow \alpha$

coronavac $\Rightarrow$ 4.549.920 $\Rightarrow$ 25.78 %

Eficácia = 50.38 %

Astrazeneca $\Rightarrow$ 5.044.420 $\Rightarrow$ 28.60 %

Eficácia = 70.00 %

Jassen $\Rightarrow$ 305.510 $\Rightarrow$ 1.73 %

Eficácia = 66.9 %

Pfizer $\Rightarrow$ 7.743.720 $\Rightarrow$ 43.89 %

Eficácia = 91.30 %

α = 74.24%

Modelos e bibliotecas usados para a simulação

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.graph_objects as go
from scipy.integrate import odeint
```

Realidade

Para comparar os resultados do modelo com a realidade, é necessário primeiramente analisar e projetar dados sobre a Covid-19 em um gráfico. Para isso, utilizamos as bases de dados "COVID-19 em Dados" e "vacinados-PE" para exibir o número de casos acumulados e número de vacinados em Pernambuco respectivamente.

```
dataset = pd.read_csv('COVID-19 em Dados.csv', sep=',')

pe = dataset.fillna('0') # preenche celulas vazias com zero

# convertendo valores para inteiro
pe["confirmados"].astype(str).astype(int)

# selecionando colunas do dataset
df_graph = pd.DataFrame({'Casos acumulados':
```

```
pe['confirmados'].to_list(), 'Data': pe['dt_referencia'].to_list()})


df_graph['Data'] = pd.to_datetime(df_graph['Data'])
datas = (df_graph['Data'] >= '2021-02-01') & (df_graph['Data'] <=
'2022-01-31')
dt_filtrado = df_graph[datas]

# convertendo valores para inteiro
pe['obitos'].astype(str).astype(int)

# selecionando a coluna de obitos
df_graph_2 = pd.DataFrame({'Mortes acumulados':
pe['obitos'].to_list(), 'Data': pe['dt_referencia'].to_list()})


df_graph_2['Data'] = pd.to_datetime(df_graph_2['Data'])
datas = (df_graph_2['Data'] >= '2021-02-01') & (df_graph_2['Data'] <=
'2022-01-31')
dt_filtrado_2 = df_graph_2[datas]

trace1 = go.Scatter(x=dt_filtrado['Data'], y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados',)
trace2 = go.Scatter(x=dt_filtrado_2['Data'], y=dt_filtrado_2['Mortes
acumulados'],mode='lines',name='Mortes acumulados',)
data = [trace1,trace2]

layout = go.Layout( title = 'Acumulado de casos reais de covid em
Pernambuco', #define o título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                        )

fig= go.Figure(data, layout = layout)

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

##Variantes da Covid

```
dataset = pd.read_csv('owid-covid-data.csv', sep = ',')
nova_base = dataset[dataset['location'] == 'Brazil']

# primeira onda Brasil (variante alfa)
df_graph_3 = pd.DataFrame({'Casos acumulados':
nova_base['total_cases'].to_list(), 'Data':
nova_base['date'].to_list()})

df_graph_3['Data'] = pd.to_datetime(df_graph_3['Data'])
datas = (df_graph_3['Data'] >= '2020-04-05') & (df_graph_3['Data'] <=
'2020-11-07')
df_data = df_graph_3[datas]

# segunda onda Brasil (variante Zeta)
df_graph_4 = pd.DataFrame({'Casos acumulados':
nova_base['total_cases'].to_list(), 'Data':
nova_base['date'].to_list()})

df_graph_4['Data'] = pd.to_datetime(df_graph_4['Data'])
datas = (df_graph_4['Data'] >= '2020-11-07') & (df_graph_4['Data'] <=
'2021-02-27')
df_data_2 = df_graph_4[datas]

# terceira onda Brasil (variante Gama)
df_graph_5 = pd.DataFrame({'Casos acumulados':
nova_base['total_cases'].to_list(), 'Data':
nova_base['date'].to_list()})

df_graph_5['Data'] = pd.to_datetime(df_graph_5['Data'])
datas = (df_graph_5['Data'] >= '2021-02-27') & (df_graph_5['Data'] <=
'2021-08-15')
df_data_3 = df_graph_5[datas]

# quarta onda Brasil (variante Delta)
df_graph_6 = pd.DataFrame({'Casos acumulados':
nova_base['total_cases'].to_list(), 'Data':
nova_base['date'].to_list()})

df_graph_6['Data'] = pd.to_datetime(df_graph_6['Data'])
datas = (df_graph_6['Data'] >= '2021-08-15') & (df_graph_6['Data'] <=
'2021-12-15')
df_data_4 = df_graph_6[datas]

# quinta onda Brasil (variante Ômicron)
df_graph_7 = pd.DataFrame({'Casos acumulados':
nova_base['total_cases'].to_list(), 'Data':
nova_base['date'].to_list()})

df_graph_7['Data'] = pd.to_datetime(df_graph_7['Data'])
```

```
datas = (df_graph_7['Data'] >= '2021-12-15') & (df_graph_7['Data'] <=
'2022-05-11')
df_data_5 = df_graph_7[datas]



#########################################################################
#########################################################################
#############################################
trace1 = go.Scatter(x=df_data['Data'], y=df_data['Casos
acumulados'],mode='lines',name='Casos acumulados Alfa',)
trace2 = go.Scatter(x=df_data_2['Data'], y=df_data_2['Casos
acumulados'],mode='lines',name='Casos acumulados Zeta',)
trace3 = go.Scatter(x=df_data_3['Data'], y=df_data_3['Casos
acumulados'],mode='lines',name='Casos acumulados Gama',)
trace4 = go.Scatter(x=df_data_4['Data'], y=df_data_4['Casos
acumulados'],mode='lines',name='Casos acumulados Delta',)
trace5 = go.Scatter(x=df_data_5['Data'], y=df_data_5['Casos
acumulados'],mode='lines',name='Casos acumulados Ômicron',)
data = [trace1,trace2,trace3,trace4,trace5]

layout = go.Layout( title = 'Acumulado de casos reais de covid no
Brasil', #define o título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Data'}, #título do eixo x
                   yaxis = {'title': 'Casos acumulados'},  #título do
eixo y
                        )

fig= go.Figure(data, layout = layout)

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura

dataset = pd.read_csv('owid-covid-data.csv', sep = ',')
nova_base = dataset[dataset['location'] == 'Brazil']

# primeira onda Brasil (variante alfa)
df_graph_8 = pd.DataFrame({'Casos acumulados':
nova_base['new_cases'].to_list(), 'Data':
nova_base['date'].to_list()})
```

```
df_graph_8['Data'] = pd.to_datetime(df_graph_8['Data'])
datas = (df_graph_8['Data'] >= '2020-04-05') & (df_graph_8['Data'] <=
'2020-11-07')
df_data_6 = df_graph_8[datas]

# segunda onda Brasil (variante Zeta)
df_graph_9 = pd.DataFrame({'Casos acumulados':
nova_base['new_cases'].to_list(), 'Data':
nova_base['date'].to_list()})

df_graph_9['Data'] = pd.to_datetime(df_graph_9['Data'])
datas = (df_graph_9['Data'] >= '2020-11-07') & (df_graph_9['Data'] <=
'2021-02-27')
df_data_7 = df_graph_9[datas]

# terceira onda Brasil (variante Zeta)
df_graph_10 = pd.DataFrame({'Casos acumulados':
nova_base['new_cases'].to_list(), 'Data':
nova_base['date'].to_list()})

df_graph_10['Data'] = pd.to_datetime(df_graph_10['Data'])
datas = (df_graph_10['Data'] >= '2021-02-27') & (df_graph_10['Data']
<= '2021-08-15')
df_data_8 = df_graph_10[datas]

# quarta onda Brasil (variante Delta)
df_graph_11 = pd.DataFrame({'Casos acumulados':
nova_base['new_cases'].to_list(), 'Data':
nova_base['date'].to_list()})

df_graph_11['Data'] = pd.to_datetime(df_graph_11['Data'])
datas = (df_graph_11['Data'] >= '2021-08-15') & (df_graph_11['Data']
<= '2021-12-15')
df_data_9 = df_graph_11[datas]

# quinta onda Brasil (variante Ômicron)
df_graph_12 = pd.DataFrame({'Casos acumulados':
nova_base['new_cases'].to_list(), 'Data':
nova_base['date'].to_list()})

df_graph_12['Data'] = pd.to_datetime(df_graph_12['Data'])
datas = (df_graph_12['Data'] >= '2021-12-15') & (df_graph_12['Data']
<= '2022-05-11')
df_data_10 = df_graph_12[datas]


#########################################################################
#########################################################################
################################################
```

```
trace1 = go.Scatter(x=df_data_6['Data'], y=df_data_6['Casos
acumulados'],mode='lines+markers',name='Casos acumulados Alfa',)
trace2 = go.Scatter(x=df_data_7['Data'], y=df_data_7['Casos
acumulados'],mode='lines+markers',name='Casos acumulados Zeta',)
trace3 = go.Scatter(x=df_data_8['Data'], y=df_data_8['Casos
acumulados'],mode='lines+markers',name='Casos acumulados Gama',)
trace4 = go.Scatter(x=df_data_9['Data'], y=df_data_9['Casos
acumulados'],mode='lines+markers',name='Casos acumulados Delta',)
trace5 = go.Scatter(x=df_data_10['Data'], y=df_data_10['Casos
acumulados'],mode='lines+markers',name='Casos acumulados Ômicron',)

data = [trace1,trace2,trace3,trace4,trace5]

layout = go.Layout( title = 'Casos diários reais de covid no Brasil',
#define o título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Data'}, #título do eixo x
                   yaxis = {'title': 'Casos diários'},  #título do eixo
y
                        )

fig= go.Figure(data, layout = layout)

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

Utilizando o Modelo SIR

Como uma forma de demonstrarmos a evolução de nosso projeto primeiro iremos utilizar o modelo SIR para que com ele tentemos fazer nossas aproximações de casos acumulados

```
def SIR_sem_vacina(y,t,N,beta,gama):

  if t < 70: # 11 abril
      r = 0.025
  elif t < 160: # 12 de abril até 10 de julho
    r = 0.088
  elif t < 170:
    r = 0.055
  elif t < 380: # 21 de julho ate 20 fevereiro
    r = 0.049
  else:
```

```
    r = 0.095

  S,I,R = y
  beta = r/S

  dSdt = -beta * S * I
  dIdt = beta * S * I - gama * I
  dRdt = gama * I

  return dSdt,dIdt,dRdt

N = 9674793
v0 = 0
i0 = 132000
r0 = 131000 #dados obtidos na secretaria de saude de permabuco
s0 = N - i0 - r0 - v0

D = 15 # infecção dura 15 dias
gama = 1./D

beta = 0.069

y0 = s0, i0, r0

t = np.linspace(0,len(dt_filtrado['Data']),len(dt_filtrado['Data']))
sol = odeint(SIR_sem_vacina, y0,t,args=(N, beta, gama))
S,I,R = sol.T

trace1 = go.Scatter(x=dt_filtrado['Data'], y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados',)
trace2 = go.Scatter(x=dt_filtrado_2['Data'],
y=(I+R),mode='lines',name='Casos Modelo',)
data = [trace1,trace2]

layout = go.Layout( title = 'Acumulado de casos reais de covid em
Pernambuco', #define o título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                        )

fig= go.Figure(data, layout = layout)

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
```

```
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

##Utilizando o SIR para prever casos ate o dia 31/07/2022

```
N = 9674793
v0 = 0
i0 = 132000
r0 = 131000 #dados obtidos na secretaria de saude de permabuco
s0 = N - i0 - r0 - v0

D = 15 # infecção dura 15 dias
gama = 1./D

beta = 0.069

y0 = s0, i0, r0

t = np.linspace(0,len(dt_filtrado['Data'])
+181,len(dt_filtrado['Data'])+181)
sol = odeint(SIR_sem_vacina, y0,t,args=(N, beta, gama))
S,I,R = sol.T

trace1 = go.Scatter(x=t, y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados',)
trace2 = go.Scatter(x=t, y=(I+R),mode='lines',name='Casos Modelo',)
data = [trace1,trace2]

layout = go.Layout( title = 'Acumulado de casos reais de covid em
Pernambuco', #define o título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                        )

fig= go.Figure(data, layout = layout)

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

Análise das condições atuais no estado de Pernambuco com a variante ômicron

```
def SVIR(y, t, gama, psi, phi, omega_v, omega_r, N):

  if t < 4 :  # até dia 05/02/2021 não havia vacinação - 2ª dose
    phi,V,omega_v,psi = 0,0,0,0
    r = 0.025
  if t < 70: # 11 abril
    r = 0.00025
  elif t < 160: # 12 de abril até 10 de julho
    r = 0.090
  elif t < 170:
    r = 0.05
  elif t < 380: # 21 de julho ate 20 fevereiro
    r = 0.042
  else:
    r = 0.105

  S,V,I,R = y
  beta = r/S
  Lambda=beta

  dSdt = omega_r * R - Lambda * S * I + omega_v * V - phi * S
  dVdt = (psi-1) * Lambda*V*I - omega_v*V + phi * S
  dIdt = (psi-1) * Lambda * V * I - omega_v * V + phi * S
  dRdt = gama*I - omega_r * R

  return dSdt,dVdt,dIdt,dRdt

#Parâmetros para gerar a solução:

gama = 1/15        # Taxa de recuperação(Tempo internado (em dias))
psi = 0.7424       # Eficácia da vacina
phi = 0.008466     # Taxa de vacinação
omega_v = 0.001170 # Imunidade da vacina cai 50% ao ano
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793     # População total
i0 = 132000     # número inicial de infectados total acumulado
r0 = 131000     # número inicial de recuperados
v0 = 0          # número inicial de vacinados
d0 = 20659      # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
+181,len(dt_filtrado['Data'])+181)

s0 = N - i0 - r0 - v0 - d0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema

solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
```

```
N))

S, V, I, R = solucao.T

#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)

trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)

trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')

layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                         )

data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

Análise no estado de Pernambuco sem vacinação

Aqui iremos aplicar um cénario hipotético em que não há aplicação e concepção de vacinas para mostrar o impacto dessa medida na duração de casos e consequentemente em sua propagação

```
#Parâmetros para gerar a solução:

gama = 1/15         # Taxa de recuperação(Tempo internado (em dias))
psi = 0             # Eficácia da vacina
phi = 0             # Taxa de vacinação
omega_v = 0         # Imunidade da vacina cai 50% ao ano
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793     # População total
i0 = 132000     # número inicial de infectados total acumulado
r0 = 131000     # número inicial de recuperados
v0 = 0          # número inicial de vacinados
d0 = 20659      # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
+181,len(dt_filtrado['Data'])+181) #181 pois representa os dias
restantes entre 31/01 até 31/07

s0 = N - i0 - r0 - v0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema

solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
N))

S, V, I, R = solucao.T

#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)

trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)

trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')
```

```
layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                        )

data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
          y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

Análise em PE levando em conta uma taxa de vacinação mais eficaz

```
#Parâmetros para gerar a solução:

gama = 1/15         # Taxa de recuperação(Tempo internado (em dias))
psi = 1             # Eficácia da vacina
phi = 0.008466      # Taxa de vacinação
omega_v = 0.001170 # Imunidade da vacina cai 50% ao ano
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793     # População total
i0 = 132000     # número inicial de infectados total acumulado
r0 = 131000     # número inicial de recuperados
v0 = 0          # número inicial de vacinados
d0 = 20659      # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
+181,len(dt_filtrado['Data'])+181)

s0 = N - i0 - r0 - v0 - d0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema

solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
N))

S, V, I, R = solucao.T
```

```
#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)

trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)

trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')

layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                        )

data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

Análise dos diferentes tipos de vacinas

ASTRAZENECA

A vacina britânica Oxford-Astrazeneca utiliza uma tecnologia biomolecular baseada no chamado “vetor viral”, que consiste na utilização de um vírus modificado para estimular o sistema imunológico na produção de anticorpos contra o novo coronavírus. Na fabricação

da vacina, uma espécie de vírus enfraquecido (adenovírus ChAdOx1), conhecido por causar gripe comum em chimpanzés, após ser modificado para não se multiplicar, carrega parte do material genético do SARS-CoV-2 responsável pela produção de uma proteína (“Spike”) que auxilia o vírus da COVID-19 a invadir as células humanas. Assim, após a vacinação, o adenovírus começa a produzir essa proteína Spike, ensinando o sistema imunológico humano que toda partícula com essa proteína deve ser destruída.

Variante alfa

```
#Parâmetros para gerar a solução:
gama = 1/15        # Taxa de recuperação(Tempo internado (em dias))
psi = 0.745       # Eficácia da vacina
phi = 0.008466     # Taxa de vacinação
omega_v = 0.001170 # Imunidade da vacina cai 50% ao ano
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793     # População total
i0 = 132000     # número inicial de infectados total acumulado
r0 = 131000     # número inicial de recuperados
v0 = 0          # número inicial de vacinados
d0 = 20659      # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
+181,len(dt_filtrado['Data'])+181)
#gera um vetor (0,1,2,3..,151) que representa o valores de t usados
para gerar a solução.

s0 = N - i0 - r0 - v0 - d0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema

solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
N))

S, V, I, R = solucao.T

#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)

trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)

trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
```

```
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')

layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                        )

data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

#####**Variante Gama**

```
#Parâmetros para gerar a solução:

gama = 1/15        # Taxa de recuperação(Tempo internado (em dias))
psi = 0.779        # Eficácia da vacina
phi = 0.008466     # Taxa de vacinação
omega_v = 0.001170 # Imunidade da vacina cai 50% ao ano
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793     # População total
i0 = 132000     # número inicial de infectados total acumulado
r0 = 131000     # número inicial de recuperados
v0 = 0          # número inicial de vacinados
d0 = 20659      # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
+181,len(dt_filtrado['Data'])+181)

s0 = N - i0 - r0 - v0 - d0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema

solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
```

```
N))

S, V, I, R = solucao.T

#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)

trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)

trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')

layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                         )

data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

#####**Variante Delta**

```
#Parâmetros para gerar a solução:
gama = 1/15        # Taxa de recuperação(Tempo internado (em dias))
```

```
psi = 0.67          # Eficácia da vacina
phi = 0.008466      # Taxa de vacinação
omega_v = 0.001170 # Imunidade da vacina cai 50% ao ano
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793     # População total
i0 = 132000     # número inicial de infectados total acumulado
r0 = 131000     # número inicial de recuperados
v0 = 0          # número inicial de vacinados
d0 = 20659      # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
+181,len(dt_filtrado['Data'])+181)
#gera um vetor (0,1,2,3..,151) que representa o valores de t usados
para gerar a solução.

s0 = N - i0 - r0 - v0 - d0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema

solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
N))

S, V, I, R = solucao.T

#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)

trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)

trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')

layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
```

```
                        )

data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

variante ômicron

```
#Parâmetros para gerar a solução:

gama = 1/15        # Taxa de recuperação(Tempo internado (em dias))
psi = 0.655        # Eficácia da vacina
phi = 0.008466     # Taxa de vacinação
omega_v = 0.155342 # Imunidade da vacina cai 50% ao ano
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793     # População total
i0 = 132000     # número inicial de infectados total acumulado
r0 = 131000     # número inicial de recuperados
v0 = 0          # número inicial de vacinados
d0 = 20659      # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
+181,len(dt_filtrado['Data'])+181)

s0 = N - i0 - r0 - v0 - d0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema

solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
N))

S, V, I, R = solucao.T

#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)
```

```
trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)

trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')

layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                        )

data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

CORONAVAC (SINOVAC BIOTECH)

A CoronaVac foi criada por meio de uma tecnologia molecular já muito utilizada em outros imunizantes. Assim como nas vacinas da gripe, poliomielite, hepatite e da meningite, ela é composta por vírus inativado, ou popularmente como “vírus morto”. As partes do novo coronavírus presentes na vacina são apenas aquelas que permitem o reconhecimento do vírus pelo nosso sistema imune e não pela sua parte responsável por causar a doença. Sendo assim, a produção do imunizante consiste em inativar o coronavírus, de maneira que fique incapaz de se multiplicar e transmitir a doença, pois torna-se incapaz de infectar as células humanas.

variante alfa

```
#Parâmetros para gerar a solução:
gama = 1/15        # Taxa de recuperação(Tempo internado (em dias))
psi = 0.507         # Eficácia da vacina
```

```
phi = 0.008466     # Taxa de vacinação
omega_v = 0.001170 # Imunidade da vacina cai 50% ao ano
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793     # População total
i0 = 132000     # número inicial de infectados total acumulado
r0 = 131000     # número inicial de recuperados
v0 = 0          # número inicial de vacinados
d0 = 20659      # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
+181,len(dt_filtrado['Data'])+181)
#gera um vetor (0,1,2,3..,151) que representa o valores de t usados
para gerar a solução.

s0 = N - i0 - r0 - v0 - d0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema

solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
N))

S, V, I, R = solucao.T

#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)

trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)

trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')

layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                        )
```

```
data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

variante Zeta

```
#Parâmetros para gerar a solução:
gama = 1/15        # Taxa de recuperação(Tempo internado (em dias))
psi = 0.50         # Eficácia da vacina
phi = 0.008466     # Taxa de vacinação
omega_v = 0.001170 # Imunidade da vacina cai 50% ao ano
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793     # População total
i0 = 132000     # número inicial de infectados total acumulado
r0 = 131000     # número inicial de recuperados
v0 = 0          # número inicial de vacinados
d0 = 20659      # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
+181,len(dt_filtrado['Data'])+181)
#gera um vetor (0,1,2,3..,151) que representa o valores de t usados
para gerar a solução.

s0 = N - i0 - r0 - v0 - d0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema

solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
N))

S, V, I, R = solucao.T

#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)
```

```
trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)

trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')

layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                        )

data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

variante Gama

```
#Parâmetros para gerar a solução:

gama = 1/15        # Taxa de recuperação(Tempo internado (em dias))
psi = 0.59       # Eficácia da vacina
phi = 0.008466     # Taxa de vacinação
omega_v = 0.001170 # Imunidade da vacina cai 50% ao ano
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793     # População total
i0 = 132000     # número inicial de infectados total acumulado
r0 = 131000     # número inicial de recuperados
v0 = 0          # número inicial de vacinados
d0 = 20659      # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
```

```
+181,len(dt_filtrado['Data'])+181)

s0 = N - i0 - r0 - v0 - d0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema

solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
N))

S, V, I, R = solucao.T

#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)

trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)

trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')

layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                         )

data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))
```

```
fig.show() #mostra a figura
```

#####**Variante Delta**

```
#Parâmetros para gerar a solução:

gama = 1/15        # Taxa de recuperação(Tempo internado (em dias))
psi = 0.40         # Eficácia da vacina
phi = 0.008466     # Taxa de vacinação
omega_v = 0.001170 # Imunidade da vacina cai 50% ao ano
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793     # População total
i0 = 132000     # número inicial de infectados total acumulado
r0 = 131000     # número inicial de recuperados
v0 = 0          # número inicial de vacinados
d0 = 20659      # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
+181,len(dt_filtrado['Data'])+181)

s0 = N - i0 - r0 - v0 - d0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema

solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
N))

S, V, I, R = solucao.T

#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)

trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)

trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')

layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
```

```
título do gráfico
                  titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                  xaxis = {'title': 'Dias'}, #título do eixo x
                  yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                       )

data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

#####**variante Ômicron**

```
#Parâmetros para gerar a solução:
gama = 1/15        # Taxa de recuperação(Tempo internado (em dias))
psi = 0.12         # Eficácia da vacina
phi = 0.008466     # Taxa de vacinação
omega_v = 0.001170 # Imunidade da vacina cai 50% ao ano
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793     # População total
i0 = 132000     # número inicial de infectados total acumulado
r0 = 131000     # número inicial de recuperados
v0 = 0          # número inicial de vacinados
d0 = 20659      # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
+181,len(dt_filtrado['Data'])+181)
#gera um vetor (0,1,2,3..,151) que representa o valores de t usados
para gerar a solução.

s0 = N - i0 - r0 - v0 - d0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema

solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
N))

S, V, I, R = solucao.T
```

```
#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)

trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)

trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')

layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                        )

data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

JANSSEN

A vacina da Jannsen utiliza a tecnologia de vetor viral não replicante, semelhante à utilizada na vacina de Oxford/AstraZeneca e também no imunizante russo Sputnik V. Essa tecnologia utiliza um adenovírus modificado geneticamente para estimular a produção de anticorpos no organismo.

variante alfa

```
#Parâmetros para gerar a solução:
gama = 1/15        # Taxa de recuperação(Tempo internado (em dias))
psi = 0.66         # Eficácia da vacina
phi = 0.008466     # Taxa de vacinação
omega_v = 0.001170 # Imunidade da vacina cai 50% ao ano
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793     # População total
i0 = 132000     # número inicial de infectados total acumulado
r0 = 131000     # número inicial de recuperados
v0 = 0          # número inicial de vacinados
d0 = 20659      # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
+181,len(dt_filtrado['Data'])+181)
#gera um vetor (0,1,2,3..,151) que representa o valores de t usados
para gerar a solução.

s0 = N - i0 - r0 - v0 - d0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema

solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
N))

S, V, I, R = solucao.T

#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)

trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)

trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')

layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
```

```
                  xaxis = {'title': 'Dias'}, #título do eixo x
                  yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                       )

data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

variante Beta

```
#Parâmetros para gerar a solução:
gama = 1/15         # Taxa de recuperação(Tempo internado (em dias))
psi = 0.64          # Eficácia da vacina
phi = 0.008466      # Taxa de vacinação
omega_v = 0.001170 # Imunidade da vacina cai 50% ao ano
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793     # População total
i0 = 132000     # número inicial de infectados total acumulado
r0 = 131000     # número inicial de recuperados
v0 = 0          # número inicial de vacinados
d0 = 20659      # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
+181,len(dt_filtrado['Data'])+181)
#gera um vetor (0,1,2,3..,151) que representa o valores de t usados
para gerar a solução.

s0 = N - i0 - r0 - v0 - d0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema

solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
N))

S, V, I, R = solucao.T

#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
```

```
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)

trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)

trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')

layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                        )

data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

variante Gama

```
#Parâmetros para gerar a solução:
gama = 1/15        # Taxa de recuperação(Tempo internado (em dias))
psi = 0.669        # Eficácia da vacina
phi = 0.008466     # Taxa de vacinação
omega_v = 0.001170 # Imunidade da vacina cai 50% ao ano
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793     # População total
i0 = 132000     # número inicial de infectados total acumulado
r0 = 131000     # número inicial de recuperados
```

```
v0 = 0          # número inicial de vacinados
d0 = 20659      # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
+181,len(dt_filtrado['Data'])+181)
#gera um vetor (0,1,2,3..,151) que representa o valores de t usados
para gerar a solução.

s0 = N - i0 - r0 - v0 - d0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema

solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
N))

S, V, I, R = solucao.T

#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)

trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)

trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')

layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                         )

data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
```

```
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

variante Delta

```
#Parâmetros para gerar a solução:
gama = 1/15         # Taxa de recuperação(Tempo internado (em dias))
psi = 0.76          # Eficácia da vacina
phi = 0.008466      # Taxa de vacinação
omega_v = 0.001170 # Imunidade da vacina cai 50% ao ano
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793      # População total
i0 = 132000      # número inicial de infectados total acumulado
r0 = 131000      # número inicial de recuperados
v0 = 0           # número inicial de vacinados
d0 = 20659       # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
+181,len(dt_filtrado['Data'])+181)
#gera um vetor (0,1,2,3..,151) que representa o valores de t usados
para gerar a solução.

s0 = N - i0 - r0 - v0 - d0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema

solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
N))

S, V, I, R = solucao.T

#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)

trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)

trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
```

```
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')

layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                        )

data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

#####**variante Ômicron**

```
#Parâmetros para gerar a solução:
gama = 1/15         # Taxa de recuperação(Tempo internado (em dias))
psi = 0.85          # Eficácia da vacina
phi = 0.008466      # Taxa de vacinação
omega_v = 0.001170 # Imunidade da vacina cai 50% ao ano
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793     # População total
i0 = 132000     # número inicial de infectados total acumulado
r0 = 131000     # número inicial de recuperados
v0 = 0          # número inicial de vacinados
d0 = 20659      # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
+181,len(dt_filtrado['Data'])+181)
#gera um vetor (0,1,2,3..,151) que representa o valores de t usados
para gerar a solução.

s0 = N - i0 - r0 - v0 - d0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema
```

```
solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
N))

S, V, I, R = solucao.T

#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)

trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)

trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')

layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                        )

data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

PFIZER

A vacina Pfizer/BioNTech utiliza a tecnologia de RNA mensageiro (mRNA). O material genético sintético, que carrega o código genético do SARS-CoV-2, estimula o organismo a gerar anticorpos contra o vírus.

Em outras palavras, o objetivo é que o mRNA sintético dê as instruções ao corpo humano para a produção de proteínas encontradas na superfície do vírus. São essas proteínas que levam à resposta do sistema imunológico e trazem a proteção para o indivíduo.

variante Alfa

```
#Parâmetros para gerar a solução:
gama = 1/15         # Taxa de recuperação(Tempo internado (em dias))
psi = 0.93          # Eficácia da vacina
phi = 0.008466      # Taxa de vacinação
omega_v = 0.001170 # Imunidade da vacina cai 50% ao ano
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793      # População total
i0 = 132000      # número inicial de infectados total acumulado
r0 = 131000      # número inicial de recuperados
v0 = 0           # número inicial de vacinados
d0 = 20659       # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
+181,len(dt_filtrado['Data'])+181)
#gera um vetor (0,1,2,3..,151) que representa o valores de t usados
para gerar a solução.

s0 = N - i0 - r0 - v0 - d0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema

solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
N))

S, V, I, R = solucao.T

#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)

trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)
```

```
trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')

layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                         )

data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

```
/usr/local/lib/python3.7/dist-packages/scipy/integrate/odepack.py:248:
ODEintWarning:

Excess work done on this call (perhaps wrong Dfun type). Run with
full_output = 1 to get quantitative information.
```

variante Beta

```
#Parâmetros para gerar a solução:
gama = 1/15        # Taxa de recuperação(Tempo internado (em dias))
psi = 0.88         # Eficácia da vacina
phi = 0.008466     # Taxa de vacinação
omega_v = 0.001170 # Imunidade da vacina cai 50% ao ano
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793     # População total
i0 = 132000     # número inicial de infectados total acumulado
r0 = 131000     # número inicial de recuperados
v0 = 0          # número inicial de vacinados
d0 = 20659      # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
```

```
+181,len(dt_filtrado['Data'])+181)
#gera um vetor (0,1,2,3..,151) que representa o valores de t usados
para gerar a solução.

s0 = N - i0 - r0 - v0 - d0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema

solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
N))

S, V, I, R = solucao.T

#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)

trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)

trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')

layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                        )

data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
```

```
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

variante Gama

```
#Parâmetros para gerar a solução:
gama = 1/15        # Taxa de recuperação(Tempo internado (em dias))
psi = 0.80         # Eficácia da vacina
phi = 0.008466     # Taxa de vacinação
omega_v = 0.001170 # Imunidade da vacina cai 50% ao ano
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793     # População total
i0 = 132000     # número inicial de infectados total acumulado
r0 = 131000     # número inicial de recuperados
v0 = 0          # número inicial de vacinados
d0 = 20659      # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
+181,len(dt_filtrado['Data'])+181)
#gera um vetor (0,1,2,3..,151) que representa o valores de t usados
para gerar a solução.

s0 = N - i0 - r0 - v0 - d0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema

solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
N))

S, V, I, R = solucao.T

#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)

trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)

trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')
```

```
layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                        )

data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

Variante Delta

```
#Parâmetros para gerar a solução:
gama = 1/15        # Taxa de recuperação(Tempo internado (em dias))
psi = 0.80         # Eficácia da vacina
phi = 0.008466     # Taxa de vacinação
omega_v = 0.001170 # Imunidade da vacina cai 50% ao ano
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793     # População total
i0 = 132000     # número inicial de infectados total acumulado
r0 = 131000     # número inicial de recuperados
v0 = 0          # número inicial de vacinados
d0 = 20659      # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
+181,len(dt_filtrado['Data'])+181)
#gera um vetor (0,1,2,3..,151) que representa o valores de t usados
para gerar a solução.

s0 = N - i0 - r0 - v0 - d0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema

solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
N))
```

```
S, V, I, R = solucao.T

#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)

trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)

trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')

layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                        )

data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

variante Ômicron

```
#Parâmetros para gerar a solução:
gama = 1/15        # Taxa de recuperação(Tempo internado (em dias))
psi = 0.70         # Eficácia da vacina
phi = 0.008466     # Taxa de vacinação
omega_v = 0.001170 # Imunidade da vacina cai 50% ao ano
```

```
omega_r = 0.001897 # Imunidade de recuperação cai 10% a cada 90 dias

N = 9674793     # População total
i0 = 132000     # número inicial de infectados total acumulado
r0 = 131000     # número inicial de recuperados
v0 = 0          # número inicial de vacinados
d0 = 20659      # número inicial de pessoas mortas
t = np.linspace(0,len(dt_filtrado['Data'])
+181,len(dt_filtrado['Data'])+181)
#gera um vetor (0,1,2,3..,151) que representa o valores de t usados
para gerar a solução.

s0 = N - i0 - r0 - v0 - d0
y0 = s0, i0, r0, v0 #vetor que armazena as condições iniciais do
sistema

solucao = odeint(SVIR, y0, t, args=(gama, psi, phi, omega_v, omega_r,
N))

S, V, I, R = solucao.T

#cada uma das variáveis trace abaixo armazena o conjunto de pontos
gerada pela função SIR que queremos plotar.
#Veja que como queremos plotar 4 conjuntos de dados diferentes S,I,R,C
armazenamos cada um deles em uma varíavel.
trace1 = go.Scatter(x=t, y=S,mode='lines',name='Suscetíveis',)

trace2 = go.Scatter(x=t, y=I, mode='lines',name='Infectados',)

trace3 = go.Scatter(x=t, y=R, mode='lines', name='Recuperados',)

trace4 = go.Scatter(x=t, y=(I+R), mode='lines',name='Casos
acumulados',)

trace5 = go.Scatter(x=t, y=dt_filtrado['Casos
acumulados'],mode='lines',name='Casos acumulados Reais',)

trace6 = go.Scatter(x=t, y=V,mode = 'lines',name = 'Vacinados')

layout = go.Layout( title = 'Modelo SIRV com exposição', #define o
título do gráfico
                   titlefont = {'family': 'Arial', 'size': 22,'color':
'#7f7f7f'}, #define a fonte e cor e fonte do título
                   xaxis = {'title': 'Dias'}, #título do eixo x
                   yaxis = {'title': 'População em 1º de Janeiro de
2022'},  #título do eixo y
                        )

data=[trace1,trace2,trace3,trace4,trace5,trace6] #vetor que armazena
```

```
os pontos que queremos plotar

fig = go.Figure(data, layout = layout) #função que gera a figura

fig.update_layout(
legend_orientation="h", #define a localização da legenda
legend=dict(
         y=-0.15,       #coloca um recuo na legenda para que ela fique
mais longe do título do eixo x.
        ))

fig.show() #mostra a figura
```

Resultados e discussões

Pode-se observar, como esperado, que o pico dos infectados no caso em que há uma vacina é menor e levemente deslocado para a esquerda em relação ao cenário sem vacina, o que ratifica a eficiência desse modelo, pois através dela, consiguiu-se - nesse cenário - postergar e, principalmente, controlar o surto de casos, fazendo com que menos pessoas sofram com os efeitos da contaminação. É possível observar também que, caso a adesão à vacinação fosse mais efetiva, tanto pela população quanto pelos orgãos governamentais, o pico dos infectados seria menor em realação ao apresentado e, consequentemente, o número de mortos também, comprovando que ela, a vacina, tem um impacto direto na progressão ou regressão da doença e que, por isso, deve-se estimular e ampliar sua adesão. Outrossim, vale ressaltar que essa simulação, leva em conta dados de 1 de janeiro de 2022, é compativel com os dados atuais sobre a progressão da doença e ,conseguiu prever com certa precisão, o cenário atual de acordo com o Monitora Covid em associação com a Fiocruz (5).

Conclusão

Apesar de alguns estudos e vários especialistas apontarem que, do ponto de vista individual, a variante Ômicron é menos letal, do ponto de vista da saúde pública e do atendimento, esse não parece ser o caso. O volume de casos provocado por essa variante é extremamente alto e, mesmo que o percentual de pessoas que necessitem de atendimento especializado seja pequeno frente ao volume extremamente alto de casos, as redes de atendimento acabam sendo ocupadas e, em última análise, a desassistência à saúde ocorre, elevando o número de óbitos. Esse aumento não se assemelha ao de variantes observadas anteriormente, como a Delta na Europa e Gamma no Brasil. No entanto, os números são expressivos, sobretudo quando consideramos que a Covid-19 é, hoje, uma doença previnível do ponto de vista de complicações clínicas, desde que exista uma alta taxa de vacinação na população.

A onda de casos da variante Ômicron aparentemente parece ter um comportamento diferente das outras variantes, apresentando um pico acelerado e posteriormente queda acentuada, conferindo alguma imunidade à população que eventualmente tenha tido a infecção. Em pessoas vacinadas e sem complicações prévias, a doença parece seguir um curso mais brando. No entanto, em pessoas não vacinadas e mesmo sem comorbidades ou

fatores de risco, essa variante apresenta um risco para internação e óbitos. De forma indireta, a onda de Ômicron valida os efeitos esperados para a vacinação da população.

É de extrema importância que se busque maior fomento à vacinação das populações nas diferentes regiões do mundo e dentro do Brasil, para evitar a circulação de forma acelerada do vírus, dificultar o surgimento de novas variantes e aliviar a demanda por atendimento especializado em saúde, sobretudo nos não vacinados, nos que possuem resistência à vacinação por desinformação ou fanatismo seja em ele em qualquer esfera.

Referências

(1) https://agenciabrasil.ebc.com.br/internacional/noticia/2022-01/omicron-pode-ser-o-virus-de-mais-rapida-propagacao-da-historia

(2) https://portalarquivos2.saude.gov.br/images/pdf/2020/fevereiro/11/protocolo-manejo-coronavirus.pdf

(3) https://saude.gov.br/saude-de-a-z/coronavírus

(4) Belfin RV, Bródka P, Radhakrishnan BL, Rejula V. 2020. COVID-19 peak estimation and effect of nationwide lockdown in India. medRxiv. 2020.05.09.20095919.

(6) https://www.cnnbrasil.com.br/saude/taxa-de-transmissao-da-covid-19-cai-no-brasil-apos-um-mes-e-meio/

(5) https://bigdata-covid19.icict.fiocruz.br/

(7) https://noticias.r7.com/saude/periodo-medio-de-incubacao-do-coronavirus-e-de-51-dias-10032020

(8) https://dados.seplag.pe.gov.br/apps/corona.html

(9) https://www.medrxiv.org/content/10.1101/2021.02.05.21250572v1.full

https://www.tre-pe.jus.br/imprensa/noticias-tre-pe/2022/Marco/cas-divulga-informacoes-sobre-covid-e-variante-omicron

https://especiais.g1.globo.com/bemestar/vacina/2021/mapa-brasil-vacina-covid/

(12) https://coronavirus.saude.mg.gov.br/blog/229-vacinacao-coronavac-astrazeneca-oxford

(13) https://www.who.int/news-room/feature-stories/detail/the-j-j-covid-19-vaccine-what-you-need-to-know

(14) https://www.medrxiv.org/content/10.1101/2021.02.05.21250572v1.full

(15) no link abaixo poderão ser vizualizados dados referesntes a quantidade de doses entregues por fabricante ao estado de pernambuco:
http://portal.saude.pe.gov.br/sites/portal.saude.pe.gov.br/files/novo_modelo_acompanhamento_recebimento_vacinas_covid-19_com_destinacao.pdf