PREVISÕES INFECTADOS COVID ALAGOAS VIA MÉTODO DE EULER

Feito por:
João Pedro Apolonio de Sousa Matos 12558360
Bruno Croso Cunha da Silva 5524390

Professor: André Salles de Carvalho

Monitora: Milena Heidecher de Oliveira

1. Introdução:

Esse trabalho foi feito para, através da análise de dados do coronavírus no Brasil descobrir as variáveis número de infectados que um suscetível encontra diariamente em média (b) e quanto tempo a doença dura em um infectado em média em dezenas de dias (k) ótimos para a melhor previsão do futuro número de infectados(i) da covid 19 dentro de um prazo de duas semanas (quatorze dias). Para isso será usado o método de Euler para resolução de Equações Diferenciais Ordinárias que, nesse caso, serve para fazer a previsão de i nos próximos dias e as médias móveis para suavização da subnotificação e demais problemas nos dados. Além disso, diversas iterações do somatório da diferença ao quadrado entre o valor observado e o valor previsto com nossas técnicas supondo diferentes valores de b e k foram utilizados como testes, conforme descrito mais detalhadamente no desenvolvimento.

2. Desenvolvimento:

2.1 Tratamento dos dados:

Nessa análise usamos as ferramentas: pandas para importação, compreensão e tratamento inicial dos dados e matplotlib para fazer os gráficos. Para importar e os dados e filtrar para apenas os pertinentes a Alagoas, usamos as funções da biblioteca Pandas:

```
#Seção importação do arquivo csv via pandas, convertendo em um DataFrame.

df = pd.read_csv("https://raw.githubusercontent.com/wcota/covid19br/master/cases-brazil-states.csv")
```

```
#Seção para filtragem de dados, selecionando somente os do estado de Alagoas.

intermediario = df['state'] == 'AL'         #Aqui, seleciona-se todas as linhas em que a coluna 'state' seja igual a 'AL'
al = df[intermediario]                      #Cria-se um novo DataFrame 'al'que contem somente os dados relativos à Alagoas
```

Após isso, foram criadas uma estimativa para o total de pessoas e as listas ainda vazias dos valores por dia das variáveis: r (recuperados), s (taxa de suscetíveis), i (proporção de de infectados). Posteriormente os dados foram carregados para listas para facilitação do trabalho através de um *for loop* e com isso preenchemos as listas s, i, r citadas previamente no parágrafo, conforme demonstrado a seguir.

```
'''Para descobrir a população total de alagoas durante o periodo de pesquisa,
será feito uma "regra de três" utilizando os casos totais e os casos totais
por 100.000 habitantes, de modo que:'''

N = (1*100000)/(0.02996)                        #N representa a população total

print(f'A população total no início do periodo analisado é de aproximadamente é de {int(N)} habitantes')
```

```
A população total no início do periodo analisado é de aproximadamente é de 3337783 habitantes
```

```
#Listas de proporções para a análise posterior:
s = []    #Suscetíveis/N
i = []    #Infectados/N
r = []    #Recuperados/N
```

```
for cont in al.index:                       #Aqui serão criadas as listas definidas acima

  novas_mortes.append(al['newDeaths'][cont])
  mortes.append(al['deaths'][cont])
  novos_casos.append(al['newCases'][cont])
  casos_totais.append(al['totalCases'][cont])
  recuperados.append(al['recovered'][cont])

  p_suscetiveis = (N - al['totalCases'][cont]) / N
  s.append(p_suscetiveis)

  rec = al['recovered'][cont]
  p_infectados = (al['totalCases'][cont] - rec) / N
  i.append(p_infectados)

  p_recuperados = (rec) / N
  r.append(p_recuperados)

  dias.append(contador)
  contador += 1
```

2.2 Análise dos dados:

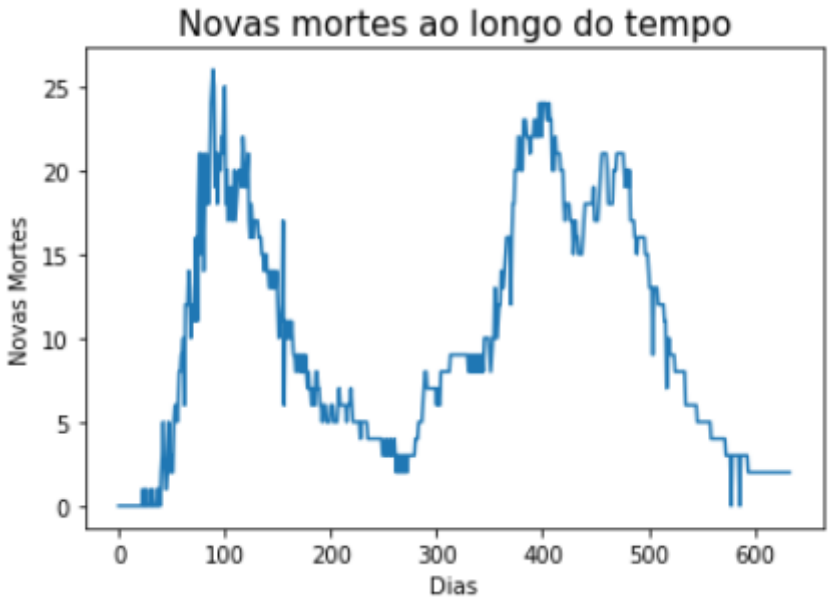
Para analisarmos os dados colocamos em gráficos as informações das listas referidas no final da última sessão com o código abaixo:

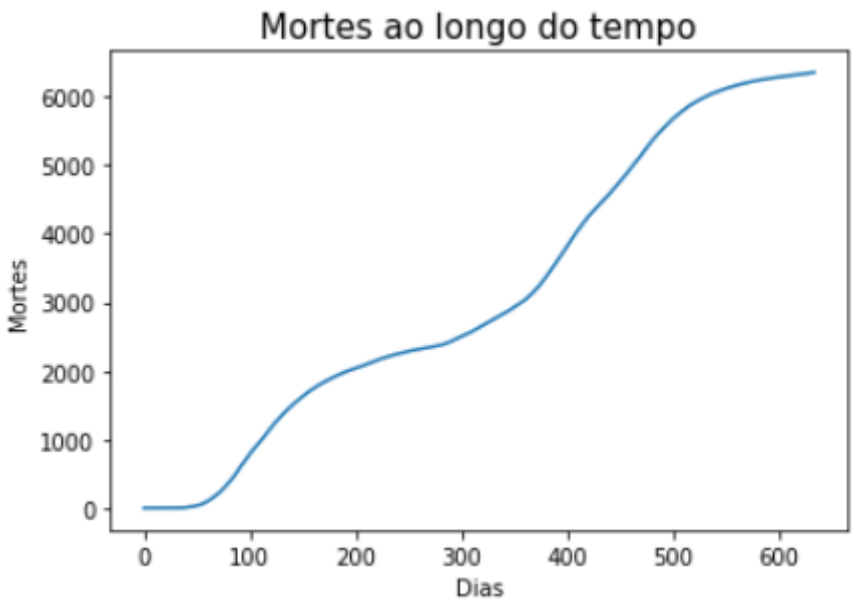
```
plt.plot(dias,novas_mortes)                                  #Utiliza a biblioteca matplotlib para criar o gráfico
plt.title('Novas mortes ao longo do tempo', fontsize=15)     #Define um título para o gráfico e o tamanho de sua fonte
plt.xlabel('Dias', fontsize=10)                              #Define o nome do eixo horizontal e o tamanho de sua fonte
plt.ylabel('Novas Mortes', fontsize=10)                      #Define o nome do eixo vertical e o tamanho de sua fonte
plt.show()                                                   #Mostra o gráfico
print()

plt.plot(dias,mortes)
plt.title('Mortes ao longo do tempo', fontsize=15)
plt.xlabel('Dias', fontsize=10)
plt.ylabel('Mortes', fontsize=10)
plt.show()
print()

plt.plot(dias,novos_casos)
plt.title('Novos casos ao longo do tempo', fontsize=15)
plt.xlabel('Dias', fontsize=10)
plt.ylabel('Novos casos', fontsize=10)
plt.show()
print()
```

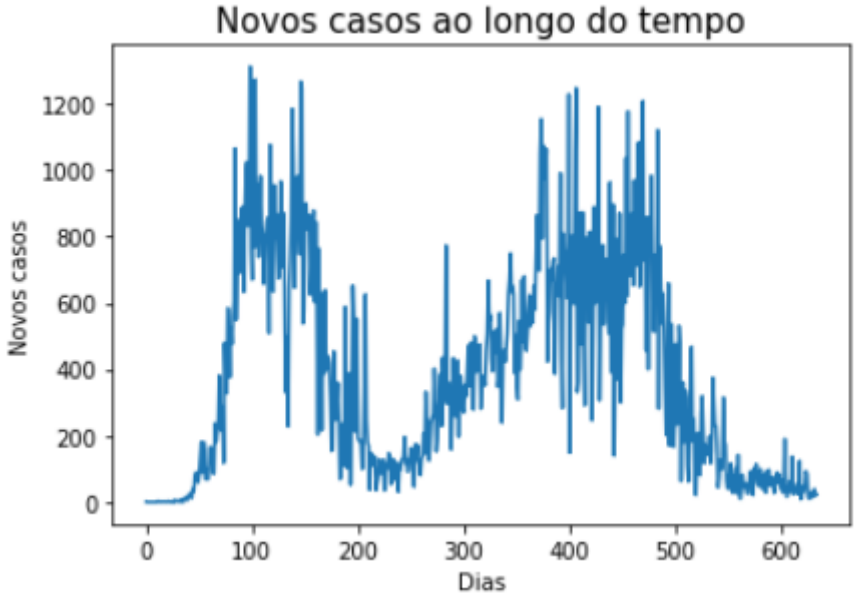
```
plt.plot(dias,casos_totais)
plt.title('Casos totais ao longo do tempo', fontsize=15)
plt.xlabel('Dias', fontsize=10)
plt.ylabel('Casos totais', fontsize=10)
plt.show()
print()

plt.plot(dias,recuperados)
plt.title('Recuperados ao longo do tempo', fontsize=15)
plt.xlabel('Dias', fontsize=10)
plt.ylabel('Recuperados', fontsize=10)
plt.show()
print()

plt.plot(dias,s)
plt.title('Proporção de suscetíveis ao longo do tempo', fontsize=15)
plt.xlabel('Dias', fontsize=10)
plt.ylabel('Proporção de suscetíveis', fontsize=10)
plt.show()
print()
```

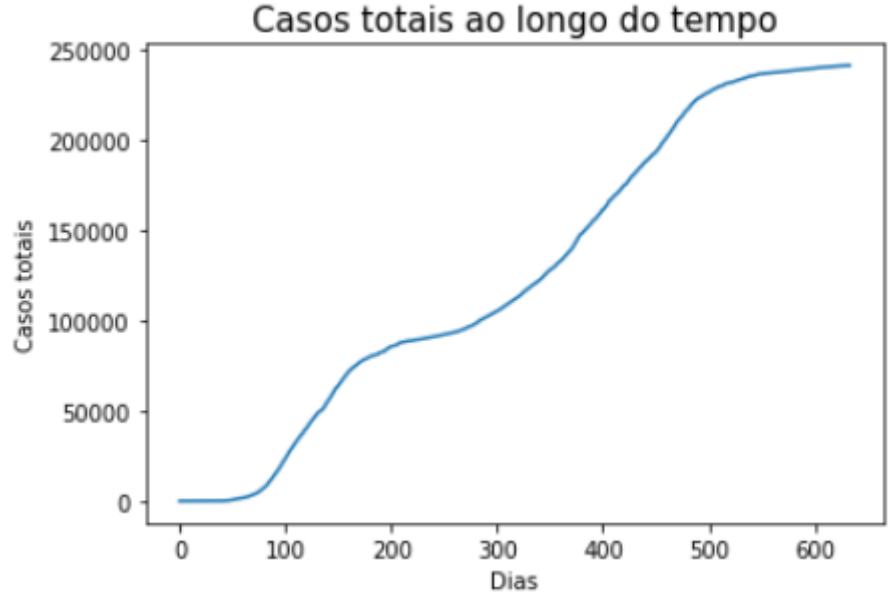
```
plt.plot(dias,i)
plt.title('Proporção infectados ao longo do tempo', fontsize=15)
plt.xlabel('Dias', fontsize=10)
plt.ylabel('Proporção de infectados', fontsize=10)
plt.show()
print()

plt.plot(dias,r)
plt.title('Proporção recuperados ao longo do tempo', fontsize=15)
plt.xlabel('Dias', fontsize=10)
plt.ylabel('Proporção de recuperados', fontsize=10)
plt.show()
print()
```

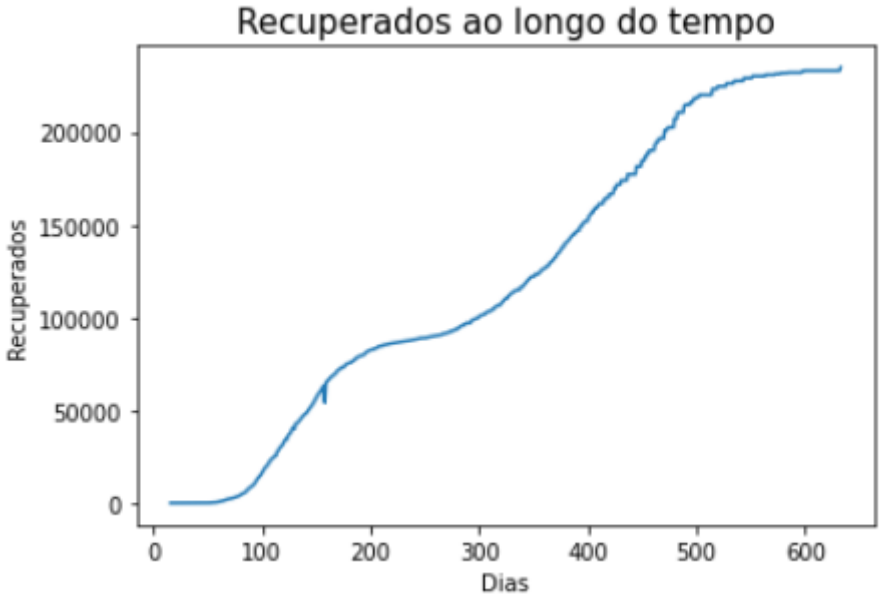
Ao rodarmos este trecho de código, obtemos os seguntes graficos:

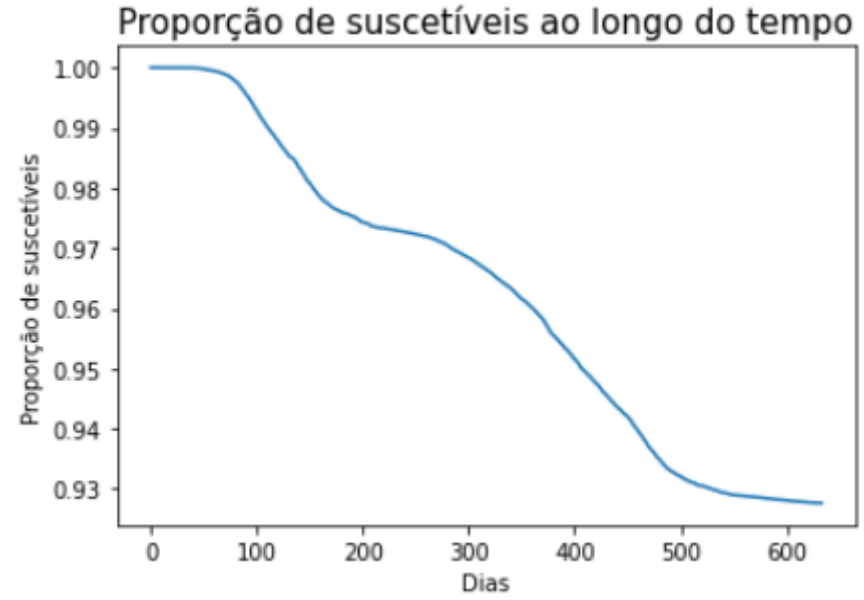



Mortes ao longo do tempo
Mortes
Dias
0
1000
2000
3000
4000
5000
6000
0
100
200
300
400
500
600


Novos casos ao longo do tempo
Novos casos
Dias
0
200
400
600
800
1000
1200
0
100
200
300
400
500
600


Casos totais ao longo do tempo
Casos totais
Dias
0
50000
100000
150000
200000
250000
0
100
200
300
400
500
600


Recuperados ao longo do tempo
Recuperados
Dias
0
50000
100000
150000
200000
0
100
200
300
400
500
600

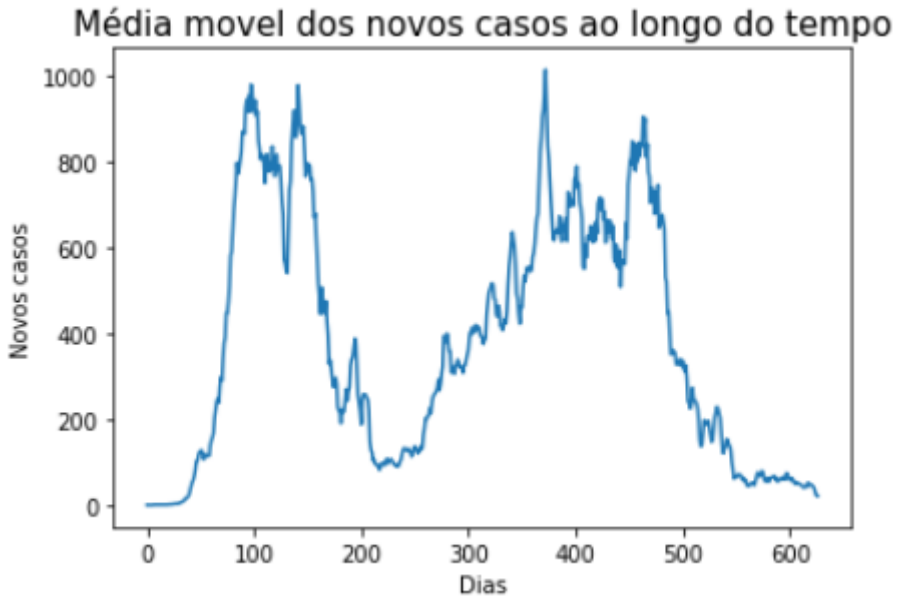

Proporção de suscetíveis ao longo do tempo
Proporção de suscetíveis
1.00
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0
100
200
300
400
500
600
Dias


Proporção infectados ao longo do tempo
Proporção de infectados
0.005
0.004
0.003
0.002
0.001
0.000
0
100
200
300
400
500
600
Dias

Com a finalidade de suavizar a base de dados relativa aos novos casos, criou-se uma segunda lista com médias móveis e, ao seguir, plotou-se seu gráfico, conforme o seguinte trecho de código:

```
#Nesta sessão, será identificado o primeiro pico de casos utilizando médias moveis

MV_novos_casos = []
dias = []
contador = 0
for c in range(len(novos_casos)-7):
  MV = (novos_casos[c] + novos_casos[c+1] + novos_casos[c+2] + novos_casos[c+3] +
        novos_casos[c+4] + novos_casos[c+5] + novos_casos[c+6])/7   #Tira-se a média de 7 dias a partir do dia 'c'.
  MV_novos_casos.append(MV)
  dias.append(contador)
  contador += 1

plt.plot(dias,MV_novos_casos)
plt.title('Média movel dos novos casos ao longo do tempo', fontsize=15)
plt.xlabel('Dias', fontsize=10)
plt.ylabel('Novos casos', fontsize=10)
plt.show()
print()
```

Ao executar esta célula, obtivem-se o seguinte gráfico:



Através deste, é possível observar que o pico está entre os dias 75 e 125. Deste modo, para identificar o primeiro pico exato, usamos o código:

```
possiveis_picos = MV_novos_casos[75:125]
pico = MV_novos_casos.index(max(possiveis_picos))
print(f'\nO primeiro pico ocorre no dia {pico}.')
```

Assim, constata-se que o pico ocorreu no dia 97. Após isso, os dados que usaremos foram apenas dos dias posteriores ao primeiro pico para comparação com nosso modelo de previsão do método de Euler. Por isso filtramos os dados para após o dia de pico. Perceba que agora s, i, r se referem ao valor fixo da taxa de infectados, taxa de suscetíveis e taxa de recuperados no primeiro dia, não mais a lista desses valores ao longo do tempo.

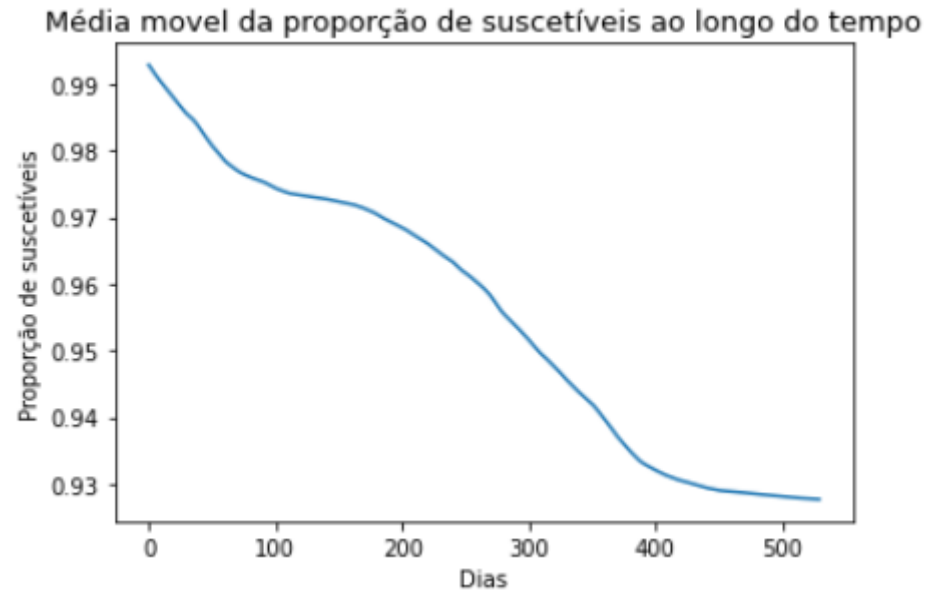
```
#Nesta sessão, iremos fatiar as listas de modo a analisar somente o período desejado

novas_mortes = novas_mortes[:97]
mortes = mortes[:97]
novos_casos = novos_casos[:97]
casos_totais = casos_totais[:97]
recuperados = recuperados[:97]
MV_novos_casos = MV_novos_casos[:97]

lst_s_futuro=s[97:]
lst_i_futuro=i[97:]
lst_r_futuro=r[97:]

s = s[97]
i = i[97]
r = r[97]
```

Após isso, criou-se listas referentes às médias móveis da taxa de infectados e de suscetíveis a partir do primeiro pico e, após isso, foi feita a criação dos respectivos gráficos a partir do seguinte código

```
#Nesta sessão, serão calculadas as médias móveis da proporção de suscetíveis e da proporção de infectados após o dia 97 (primeiro pico).

lst_MV_s = []
lst_MV_i = []
dias = []
contador = 0

for c in range(537-7):          #634 (dias total) -97 (dias até o primeiro pico)=537
  MV_s = (lst_s_futuro[c] + lst_s_futuro[c+1] + lst_s_futuro[c+2] +
          lst_s_futuro[c+3] + lst_s_futuro[c+4] + lst_s_futuro[c+5] + lst_s_futuro[c+6])/7
  lst_MV_s.append(MV_s)

  MV_i = (lst_i_futuro[c] + lst_i_futuro[c+1] + lst_i_futuro[c+2] +
          lst_i_futuro[c+3] + lst_i_futuro[c+4] + lst_i_futuro[c+5] + lst_i_futuro[c+6])/7
  lst_MV_i.append(MV_i)

  dias.append(contador)
  contador += 1
```

```
plt.plot(dias,lst_MV_s)
plt.title('Média movel da proporção de suscetíveis ao longo do tempo', fontsize=13)
plt.xlabel('Dias', fontsize=10)
plt.ylabel('Proporção de suscetíveis', fontsize=10)
plt.show()
print()

plt.plot(dias,lst_MV_i)
plt.title('Média movel da proporção de infectados ao longo do tempo', fontsize=13)
plt.xlabel('Dias', fontsize=10)
plt.ylabel('Proporção de infectados', fontsize=10)
plt.show()
```

Com a execução desta célula, obtem-se o seguinte:



2.3 Método de Euler para resolução de Equações Diferenciais Ordinárias e previsões:

Para aplicar o Método de Euler para resolução de Equações Diferenciais Ordinárias, primeiro foi criado uma função que retorna a taxa de variação da taxa de infectados (i) e da taxa de suscetíveis (s), conforme o código a seguir:

```
Após isso, são plotados os campos vetoriais correspondentes para cada valor de b e k
utilizados nos testes, para servir como exemplos.'''

def campo_vetores(s, i, b, k):
  '''(int, int, int, int) ---> (int, int)

  Recebe os valores iniciais s (proporção de sucetíveis no dia do primeiro pico),
  i (proporção de infectados no dia do primeiro pico), b (número médio de infectados
  que um suscetível encontra por dia) e k (relativo à duração da doença em um infectado)

  Retorna o par ordenado (x,y) das componentes horizontais e verticais do vetor correspondente
  à variação de s e i para os respectivos valores de b e k.
  '''

  mudanca_s = (-1)*(b*s*i)          #Componente horizontal de F(s,i)
  mudanca_i = ((b*s)-k)*i           #Componente vertical de F(s,i)

  return mudanca_s, mudanca_i
```

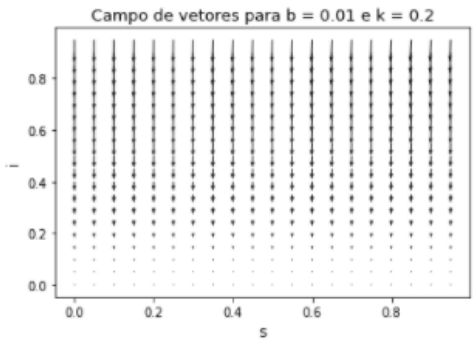
Deve-se frisar que, para tal, foi feito uso da fórmula demonstrada em aula:

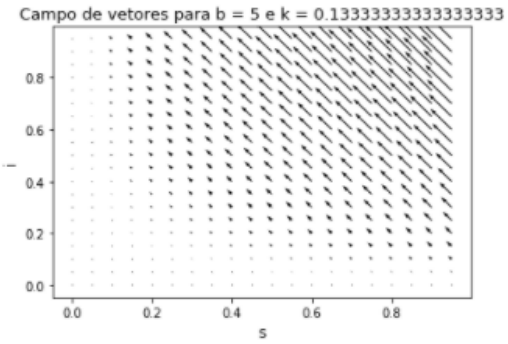
```
mudanca_s = (-1)*(b*s*i)
mudanca_i = ((b*s)-k)*i
```

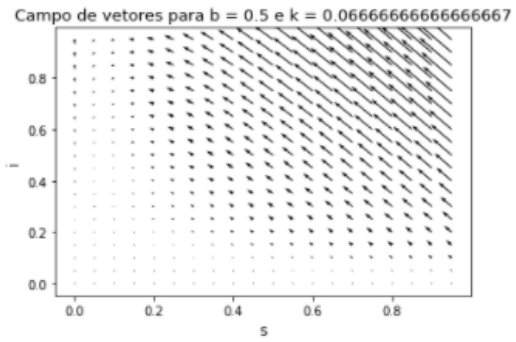
Usando essa função, foram criados vários campos de vetores usando diferentes constantes para b e k, aplicando o conhecimento adquirido de EDO's na aula e de criação de gráficos diversas vezes, com o intúito de demonstrar possíveis campos de vetores dependendo dos valores atribuidos às constantes b e k. Para tal usamos o seguinte código:

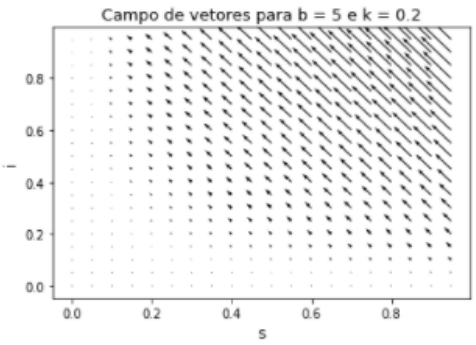
```
lista_k = [(1/15), (2/15), (3/15)]  #lista com alguns valores para serem testados para k no intervalo [(1/15),(3/15)]
lista_b = [0.01,0.5,1,1.5,2,3,4,5]  #lista com alguns valores para serem testados para b no intervalo [0,5]

for b in lista_b:
  for k in lista_k:
    #varições são os ritimios de mudança das variáveis x se refere a s e y se refere a i
    variacoes_x = []
    variacoes_y = []

    #eixo é uma forma de indicar os vetores iniciais(canto direito do gráfico)
    eixo_x = []
    eixo_y = []

    #faremos os gráficos acompanhando o desenrolar de 20 vetores no extremo canto direito
    for contador_1 in range(20):
      x = contador_1/20
      for contador_2 in range(20):
        y = contador_2/20

        eixo_x.append(x)
        eixo_y.append(y)

        (variacao_x, variacao_y) = campo_vetores(x, y, b, k)
        variacoes_x.append(variacao_x)
        variacoes_y.append(variacao_y)

    plt.quiver(eixo_x, eixo_y, variacoes_x, variacoes_y)
    plt.title(f'Campo de vetores para b = {b} e k = {k}', fontsize=13)
    plt.xlabel('s', fontsize=12)
    plt.ylabel('i', fontsize=12)
    plt.show()
    print()
```

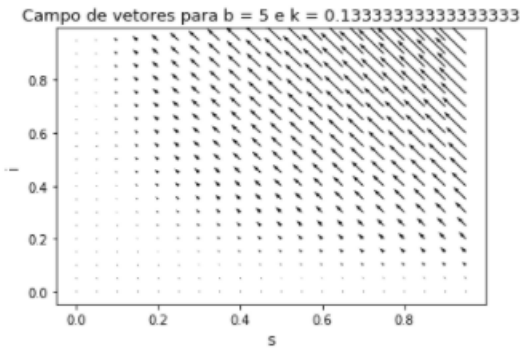
Alguns dos gráficos ficaram assim:

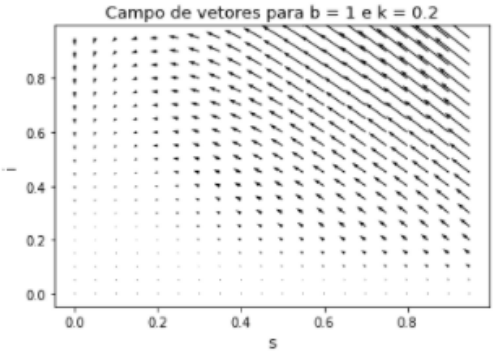


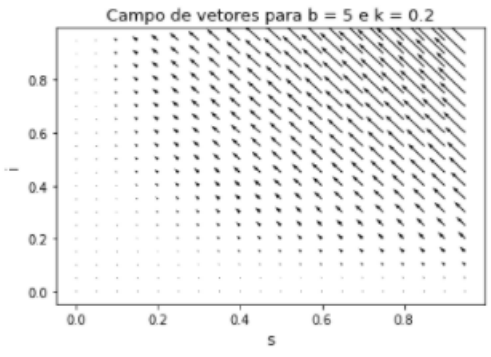


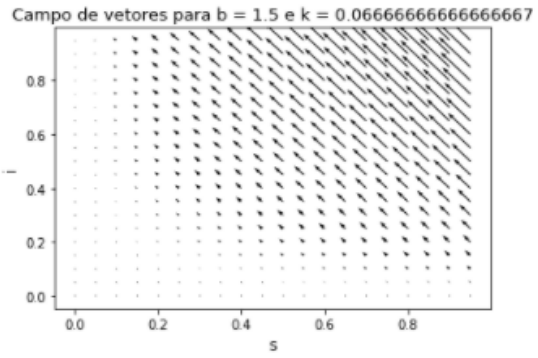












Em seguida criou-se uma função Euler() que retorna uma lista de previsões para i e s para os próximos 14 dias com base na teoria do método de Euler para resolução de Equações Diferenciais Ordinárias, como demonstrado a seguir:

```python
#Nesta sessão, será desenvolvida a função que calcula, segundo o método de Euler, previsões futuras.
#para a proporção de infectados e de proporção de suscetíveis em uma lista.

def euler(s,i,b,k):
  '''(int, int ,int ,int) ---> (list, list)
  Recebe a proporção de suscetíveis inicial (s), a proporção de infectados inicial (i), à quantos infectados um
  suscetível tem contato por dia (em média) e um valor (k) relativo ao tempo que a doença fica em um infectado.
  Chama a função campo_vetores(s, i, b, k) para estabelecer as variações em s e i.

  Retorna uma lista contendo as previsões para a proporção de suscetíveis e a proporção de infectados para os próximos 14 dias.
  '''
  s_anterior = s
  i_anterior = i

  previsoes_s = [s]
  previsoes_i = [i]

  (mudanca_s, mudanca_i) = campo_vetores(s, i, b, k)

  for count in range(14):

    s_proximo = s_anterior+(1/14) * mudanca_s
    previsoes_s.append(s_proximo)
    s_anterior = s_proximo

    i_proximo = i_anterior+(1/14) * mudanca_i
    previsoes_i.append(i_proximo)
    i_anterior = i_proximo

    (mudanca_s, mudanca_i) = campo_vetores(s_anterior, i_anterior, b, k)

  return previsoes_s, previsoes_i
```

Por fim foram criadas listas de valores para as constantes b e k que foram iteradas no código que faz várias previsões com o método de Euler em questão e utiliza da seguinte fórmula para calcular o erro entre as estimativas entre a taxa de infectados prevista e a observada é:

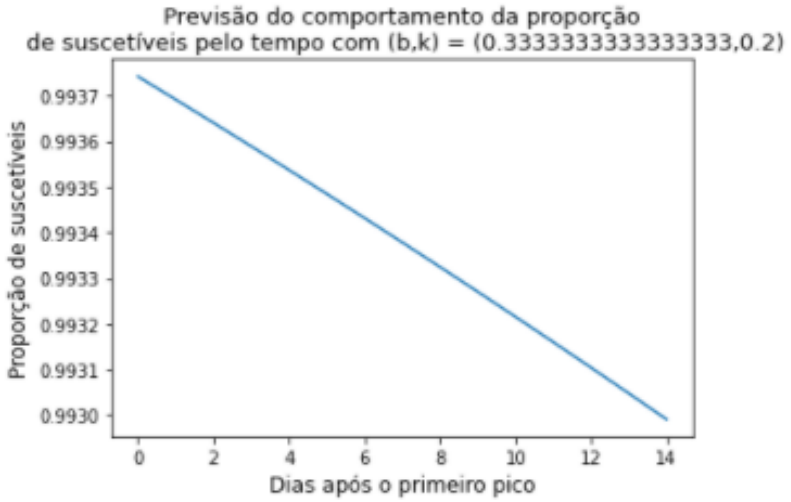
$$\sum_t (i(t) - \mathrm{dsi}(t))^2,$$

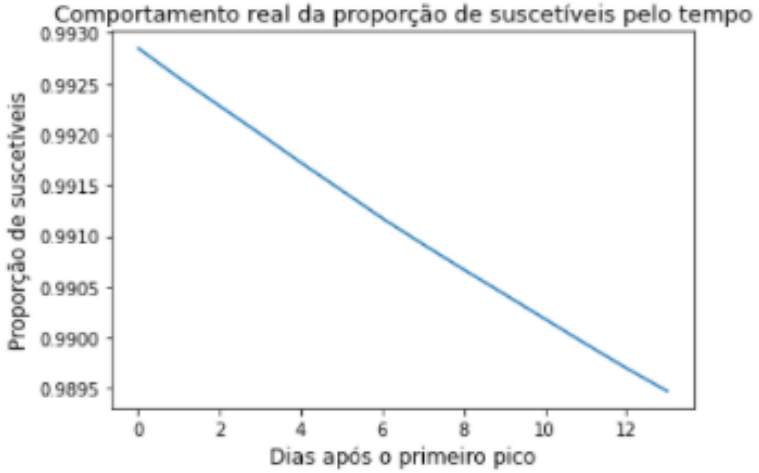
O código em questão que acaba por selecionar a melhor combinação para b e k dentre as possíveis combinações é:

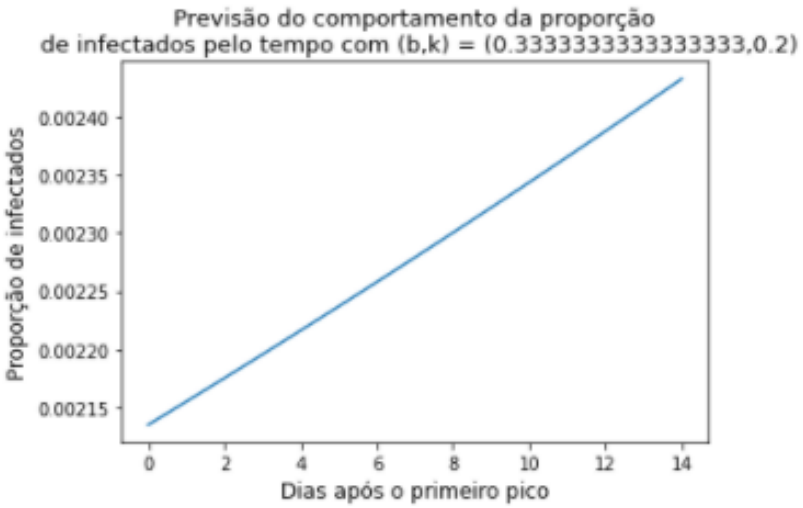
```
#Aqui testaremos valores para b e k de forma a trazer previsões mais próximas do que de fato ocorreu 14 dias após o primeiro pico.

#lista_b contem alguns valores de b no intervalo [0,5]
lista_b = [0.3333333333333333, 0.6666666666666666, 1.0, 1.3333333333333333, 1.6666666666666667, 2.0, 2.3333333333333335,
           2.6666666666666665, 3.0, 3.3333333333333335, 3.6666666666666665, 4.0, 4.333333333333333, 4.666666666666667]

#Cria valores para k no intervalo [(1/15),(1/5)]
lista_k = []
for cont in range(10):
  lista_k.append((1/15)+cont*(2/135))

soma_s = 10000000000000000      #Atribui valores muito elevados para soma_s
soma_i = 10000000000000000      #Atribui valores muito elevados para soma_i
menor_i = soma_i

#Otimizando b e k focando na proporção de infectadis (i):
for b in lista_b:
  for k in lista_k:
      (previsoes_s,previsoes_i)=euler(s,i,b,k)
      for count in range(len(previsoes_i)):
        soma_i=(previsoes_i[count]-lst_MV_i[count])**2      #Verifica o erro quadratico associado
  if soma_i < menor_i:                                     #"Seleciona" o melhor par (b, k) baseado no par que resulta em um menor erro
    menor_i = soma_i
    melhorbk = (b,k)
```

Com isso descobre-se que b = 0.3333 e k = 0.2 são os valores que minimizam o erro em questão. Por fim, foram feitos os gráficos das melhores previsões para os 14 dias após o primeiro pico e os valores de fato observados nesse momento.

```
plt.plot(previsoes_s)
plt.title(f'Previsão do comportamento da proporção\n de suscetíveis pelo tempo com (b,k) = ({melhorbk[0]},{melhorbk[1]})', fontsize=13)
plt.xlabel('Dias após o primeiro pico', fontsize=12)
plt.ylabel('Proporção de suscetíveis', fontsize=12)
plt.show()
print()

plt.plot(lst_MV_s[:14])
plt.title('Comportamento real da proporção de suscetíveis pelo tempo', fontsize=13)
plt.xlabel('Dias após o primeiro pico', fontsize=12)
plt.ylabel('Proporção de suscetíveis', fontsize=12)
plt.show()
print()

plt.plot(previsoes_i)
plt.title(f'Previsão do comportamento da proporção\n de infectados pelo tempo com (b,k) = ({melhorbk[0]},{melhorbk[1]})', fontsize=13)
plt.xlabel('Dias após o primeiro pico', fontsize=12)
plt.ylabel('Proporção de infectados', fontsize=12)
plt.show()
print()

plt.plot(lst_MV_i[:14])
plt.title('Comportamento real da proporção de infectados pelo tempo', fontsize=13)
plt.xlabel('Dias após o primeiro pico', fontsize=12)
plt.ylabel('Proporção de infectados', fontsize=12)
plt.show()
print()
```


Previsão do comportamento da proporção
de suscetíveis pelo tempo com (b,k) = (0.3333333333333333,0.2)
Proporção de suscetíveis
0.9937
0.9936
0.9935
0.9934
0.9933
0.9932
0.9931
0.9930
0
2
4
6
8
10
12
14
Dias após o primeiro pico


Comportamento real da proporção de suscetíveis pelo tempo
Proporção de suscetíveis
0.9930
0.9925
0.9920
0.9915
0.9910
0.9905
0.9900
0.9895
0
2
4
6
8
10
12
Dias após o primeiro pico



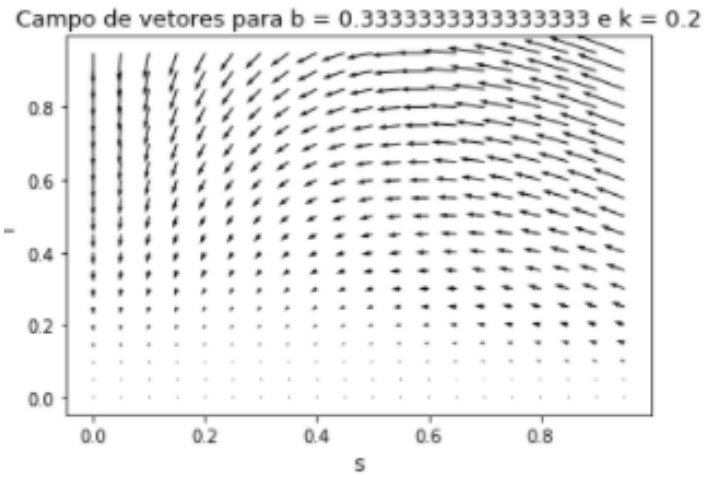
:

As observações seguem os valores previstos de forma relativamente satisfatória, com erro de 2.6902227523778352e-8.

Para finalizar foi feito o campo de vetores com b e k ótimos:

```
#Nesta sessão, será plotado o campo de vetores com b e k ideais (valores obtidos acima)

b,k = melhorbk

variacoes_x = []
variacoes_y = []

eixo_x = []
eixo_y = []

for contador_1 in range(20):
  x = contador_1/20
  for contador_2 in range(20):
    y = contador_2/20

    eixo_x.append(x)
    eixo_y.append(y)

    (variacao_x, variacao_y) = campo_vetores(x, y, b, k)
    variacoes_x.append(variacao_x)
    variacoes_y.append(variacao_y)


plt.quiver(eixo_x, eixo_y, variacoes_x, variacoes_y)
plt.title(f'Campo de vetores para b = {b} e k = {k}', fontsize=13)
plt.xlabel('s', fontsize=12)
plt.ylabel('i', fontsize=12)
plt.show()
```

Ao executarmos este trecho de código, obtemos o seguinte:



3. Conclusão.

O resultado final desse trabalho é que a estimativa de que o valor de b = 0.3333 e k = 0.2 trazem previsões para os 14 dias após o primeiro pico de covid 19 muito próximos aos valores de fato observados, com erro de apenas 2.6902227523778352e-8. Portanto é possível observar que o método de Euler para resolução de Equações Diferenciais Ordinárias ainda é aplicável no mundo moderno e pode trazer previsões muito precisas dentro do contexto de uma pandemia.