Leonardo Heidi Almeida Murakami, 11260186
Vítor Neri Roque, 12558099
Documentação do Exercício Programa

Modelo SIR - Dados de Covid-19 do Pará (PA)

Resumo

Realizamos a projeção do modelo SIR através da importação dos dados reais de casos de Covid-19 divulgados por diversos estados, no caso, foram utilizados os dados divulgados no estado do Pará. Assim, através da modelagem do rumo do cenário pandêmico pelo modelo SIR, foi possível estimar o número de infectados, suscetíveis e recuperados do vírus nas semanas seguintes da pandemia, utilizando os dados do primeiro pico da mesma para calcular tais previsões.

1 Dados Utilizados

O conjunto completo dos dados fornecidos pelo governo do Pará inclui o período dos dias 18/03/2020 até o dia 17/12/2021, onde, para podermos lidar melhor com a subnotificação dos dados, realizamos uma média móvel de 7 dias.

2 Modelo SIR

Para integrar as equações do modelo SIR utilizamos o método de Euler, que, embora não possua a maior precisão possível, demonstrou acurácia o suficiente para ser viável.

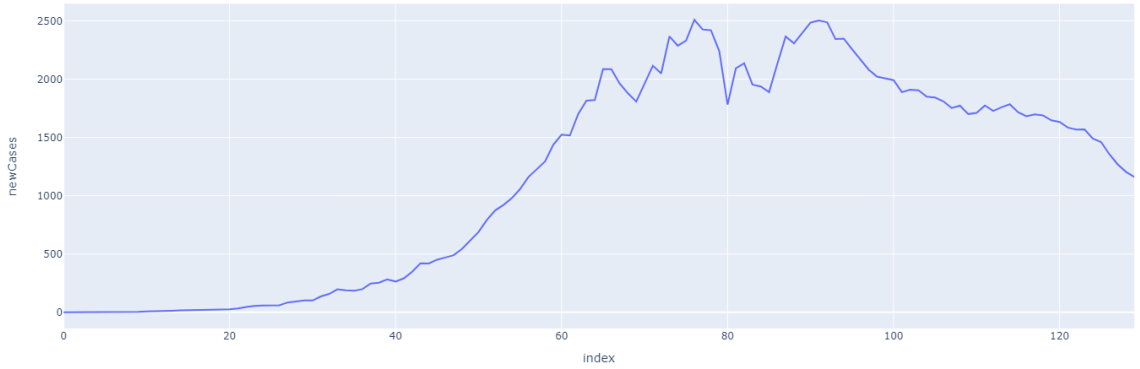
Para estimar os parâmetros k e b, foi criado um espaço de tamanho de 2 milhões contendo combinações linearmente espaçadas dos dois parâmetros, selecionamos então, aleatoriamente, 1% do espaço gerado para calcular o Erro Quadrado Médio (EQM) comparando o valor previsto com os dados reais de modo que a dupla de parâmetros com menor EQM foi escolhida para o modelo. Dessa forma, nosso modelo SIR retorna um gráfico de número de infectados, recuperados e suscetíveis utilizando os dados reais x número calculado pelo modelo SIR.

3 Desenvolvimento

Para facilitar o processo e evitar download de dados desnecessariamente, carregamos a base diretamente do Github e cortamos somente os dados do Pará. Para evitar problemas de subnotificação acabamos realizando uma média móvel de 7 dias para evitar problemas notáveis do dados, onde, periodicamente, a quantidade de casos novos registrados caia muito abaixo do esperado.

Para podermos visualizar os dados abaixo foi utilizada a biblioteca Plotly, apenas com a finalidade de possuir um gráfico mais bonito e interativo facilitando achar a data ideal para realizarmos o corte do primeiro pico, abaixo mostramos uma comparação dos dados tratados com a média móvel contar os dados “crus”.

Ao olharmos para a curva de novos casos com a média amostral, constatamos que o primeiro pico tem fim por volta do dia 130. Portanto, como o modelo SIR é projetado com os dados do primeiro pico, utilizamos somente os seguintes dados para a modelagem:



Agora com os dados devidamente tratados, podemos partir para a criação do modelo SIR, deste modo criamos uma classe denominada SIRModel onde, através do método de Euler, podemos calcular as projeções do número de infectados, recuperados e suscetíveis.

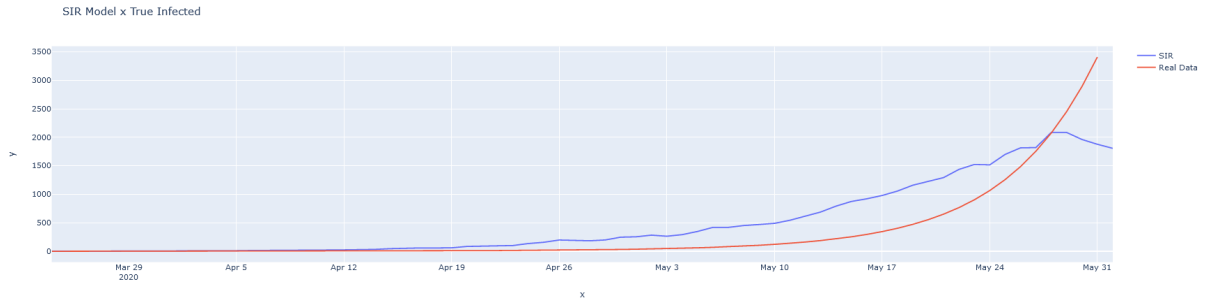
Para encontrar os melhores parâmetros, fizemos um trecho do código que testava diversas combinações de parâmetros dentro do intervalo [0, 5] para o b e do intervalo [1/15, 1/5] para o k, onde se era calculado o EQM entre o valor obtido e o valor real. Assim, quando a combinação apresentava um EQM menor do que o registrado até então, ela era salva como a melhor combinação de parâmetros encontrada. Essa forma de encontrar os parâmetros foi a escolhida por ser automática, independente do estado escolhido (possibilitando então a aplicação do mesmo código para diferentes estados) e por ser relativamente rápida, evitando testar diversas combinações desnecessárias. Vale apontar que não necessariamente é o melhor método a ser tomado, mas, devido a sua facilidade de implementação e boa capacidade de encontrar um bom parâmetro, é suficiente.

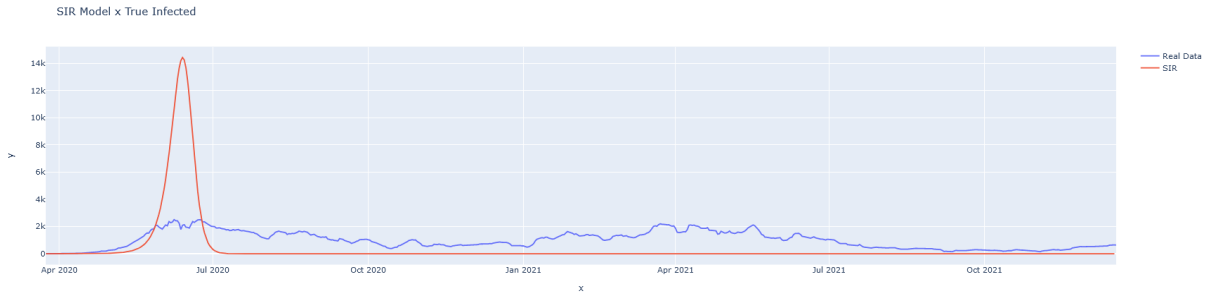
Após a escolha dos melhores parâmetros de b e k podemos então instanciar o modelo que possui a melhor chance de representar os dados reais, durante a utilização do método de "treino" do modelo, salvamos todos os dados previstos de infectados, suscetíveis, recuperados e o delta de cada dia, para podermos comparar com os valores reais

4 Resultados

4.1 Utilizando até o primeiro pico para achar b e k

Com isso, no último passo foram utilizados os dados salvos em array para plotar as seguintes previsões do modelo SIR:



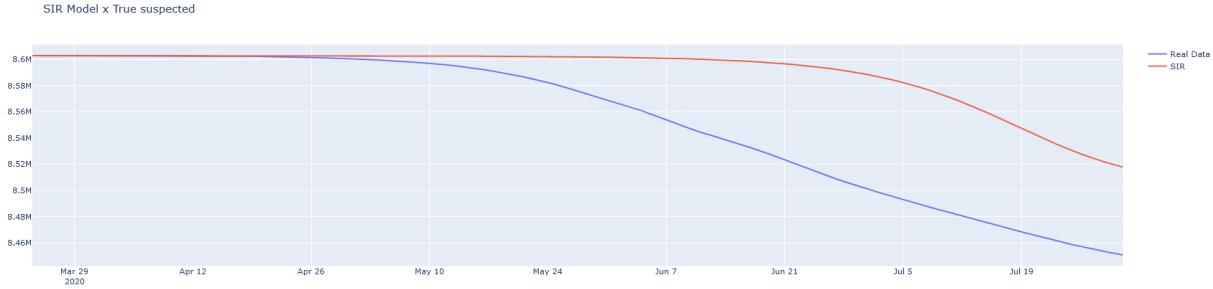


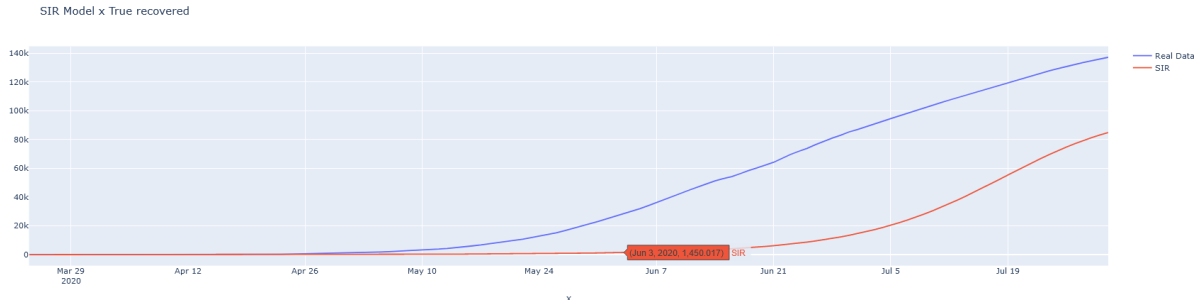
Podemos observar que, embora consigamos encontrar muito bem o período do primeiro pico através do modelo sir, não conseguimos estimar qual a quantidade máxima de infectados em um dia. Na figura acima estamos comparando a coluna newCases (representando o delta real de infectados por dia) com o delta do número de infectados (array dI) previsto pelo modelo SIR. Podemos comparar também como o modelo se comporta quando possuímos todos os dados do período, ou seja, utilizar quase todos os dados do pico

4.2 Utilizando quase todo o primeiro pico para achar b e k

Quando utilizamos quase todo o primeiro pico para achar os melhores resultados, o modelo acabou não acertando o período do primeiro pico, embora tenha sido mais preciso na quantidade de pessoas infectadas durante este.

O mesmo comportamento defasado foi visto quando comparamos os valores de pessoas suscetíveis e recuperadas durante o período, visto pelo gráfico abaixo.





5 Análise dos resultados

Em conhecimento de que o modelo SIR deve prever:

- O aumento do número de infectados seguido da queda do mesmo;
- O crescimento do número de recuperados;
- O decrescimento do número de suscetíveis;

Com base nos gráficos obtidos que expõe os resultados do modelo, podemos afirmar que o modelo SIR possui o comportamento esperado, porém não necessariamente o comportamento representativo do período da epidemia estudado apresentando algumas incoerências com os dados reais. Tais incoerências podem se dar devido a diversos fatores, como a imprecisão dos dados divulgados pelo Pará, ou a existência de duplas de parâmetros b e k melhores que estejam fora do período definido pela construção do espaço.

Mesmo assim, o modelo fornece uma ideia de como se dará o desenvolvimento da pandemia nas semanas seguintes ao primeiro pico.

Sobre os parâmetros de b e k, obtivemos que, quando não utilizamos o dado de todo o pico:

Parâmetro	Valor que minimiza EQM
b	0.1950097504875244
k	0.1922589255922589 3

Sobre os parâmetros de b e k, quando utilizamos os dados de todo o pico obtivemos que:

Parâmetro	Valor que minimiza EQM
b	0.1950097504875244
k	0.1938605271938605 4

Utilizando tais valores, nosso modelo SIR apresentou maior eficiência.

6 Parte técnica

Trecho do código que carrega, trata e processa os dados da pandemia:

```
df =
pd.read_csv("https://raw.githubusercontent.com/wcota/covid19br/master/cases-brazil-states.csv")
#CONSTANTS
STATE = 'PA' #@param {type:"string"}
FIRST_PEAK_DAY_CUT = 130 #@param {type: "integer"}
HYPEROPT_EPOCHS = 210000 #@param {type:"slider", min:10000, max:20000000, step:200000}
MOVING_AVERAGE_INTERVAL = 7 #@param {type: "integer"}
df_pa = df.query(f"state == '{STATE}'").reset_index(drop=True).fillna(0)
pop =
(df_pa['totalCases'].iat[-1]*1e5/(df_pa['totalCases_per_100k_inhabitants'].iat[-1]))
```

É importante observar que tanto o estado que será modelado quanto o dia do primeiro pico em que o dados serão retirados para a modelagem são inseridos pelo usuário.

Trecho que cria a média móvel, a compara com os dados “crus” e plota o gráfico:

```
# creating rolling average df
rolling_df_pa = df_pa.copy()
rolling_df_pa.index = rolling_df_pa['date']
rolling_df_pa['sumCases'] = rolling_df_pa.newCases.cumsum()
rolling_df_pa['suspected'] = rolling_df_pa.sumCases.apply(lambda x: pop - x)
rolling_df_pa['infected'] = rolling_df_pa['totalCases'] -
rolling_df_pa['recovered']

#creating multiple rolling average dfs
moving_average_df =
rolling_df_pa.rolling(MOVING_AVERAGE_INTERVAL).mean().dropna().reset_index()
x_col = 'index'
y_col = 'newCases'
fig = px.line(rolling_df_pa.reset_index(drop=True).reset_index(), x=x_col,
y=y_col)
fig.add_scatter(x=moving_average_df.reset_index()[x_col],
y=moving_average_df.reset_index()[y_col])

fig.data[0].name = "Original Data"
fig.data[0].showlegend = True
fig.data[1].name = f"{MOVING_AVERAGE_INTERVAL} Day Moving Average"
fig.data[1].showlegend = True

fig.show()
```

Trecho do código que seleciona apenas os dados do primeiro pico:

```
chosen_moving_average = moving_average_df
first_peak_day_cut = FIRST_PEAK_DAY_CUT

rolling_df_pa = chosen_moving_average.iloc[:first_peak_day_cut, :]
```

```
fig = px.line(rolling_df_pa.reset_index(), x='index', y='newCases')
fig.show()
```

Trecho do código que cria a classe do modelo SIR:

```
class SIRModel:
  def __init__(self, b, k, total_population):
    # SIR Params
    self.N = total_population #pop size

    self.S = [] #suspected proportion
    self.I = [] #infected proportion
    self.R = [] #recovered proportion

    self.dS = [] #delta suspected proportion
    self.dI = [] #delta infected proportion
    self.dR = [] #delta recovered proportion
    # Hyperparams
    self.b = b # in [0, 5]
    self.k = k # in [0.067, 0.2]

  def euler_method_new_tuple(self, interval):
    s, i, r = self.S[-1], self.I[-1], self.R[-1]
    # based on:
https://www.maa.org/press/periodicals/loci/joma/the-sir-model-for-spread-of-
disease-eulers-method-for-systems
    s_n = s - (self.b * s * i * interval)
    i_n = i + (((self.b * s) - self.k) * i)*interval
    r_n = r + (self.k * i)*interval

    ds_n = -((self.b * s) * i)*interval
    di_n = (((self.b * s) - self.k) * i)*interval
    dr_n = (self.k * i)*interval
    return ((s_n, i_n, r_n), (ds_n, di_n, dr_n))

  def fit(
      self,
      data,
      cases_column='newCases',
      suspected_column='suspected',
      recovered_column='recovered',
      fit_until=None,
      metricI=True,
      metricS=True,
      metricR=True,
      interval=1
    ):
    starting_point = 0
```

```
    self.I.append(data.loc[starting_point, cases_column]/self.N)
    self.S.append(1 - self.I[0])
    self.R.append(0)
    if fit_until is None:
      fit_until = len(data) - 1
    for j in range(starting_point, fit_until, interval):
      #update sir value
      ((s, i, r), (ds, di, dr)) =
self.euler_method_new_tuple(j-starting_point)
      self.S.append(s)
      self.I.append(i)
      self.R.append(r)

      self.dS.append(ds)
      self.dI.append(di)
      self.dR.append(dr)

    self.S = np.array(self.S)
    self.I = np.array(self.I)
    self.R = np.array(self.R)

    self.dS = np.array(self.dS)
    self.dI = np.array(self.dI)
    self.dR = np.array(self.dR)

  def score(self, y_true_I, y_true_S, y_true_R, metricI=True, metricS=True,
metricR=True):
    diffI = (self.I - y_true_I[:len(self.I)]/self.N)**2 if metricI else
np.array([0])
    diffS = (self.S - y_true_S[:len(self.S)]/self.N)**2 if metricS else
np.array([0])
    diffR = (self.R - y_true_R[:len(self.R)]/self.N)**2 if metricR else
np.array([0])
    return sum(diffI + diffS + diffR)
```

Nessa classe, a **função** `euler_method_new_tuple` calcula os valores de infectados, recuperados e suscetíveis do modelo SIR, enquanto a **função** `fit` salva os valores calculados para cada dia em um array e a **função** `score` calcula a métrica de erro para os três arrays salvos (de infectados, recuperados e suscetíveis).

Trecho que calcula os melhores parâmetros b e k chama a classe os utilizando:

```
import warnings
warnings.filterwarnings("ignore")

from tqdm import tqdm

b_space = np.linspace(0, 5, 20000)
k_space = np.linspace(1/15, 1/5, 1000)
```

```
params_space = np.array(np.meshgrid(b_space, k_space)).T.reshape(-1,2)

best = {
    "b": 0,
    "k": 0,
    "metric": np.inf
hyperopt_epochs = HYPEROPT_EPOCHS
interval = 1
print(f"Testing {hyperopt_epochs/len(params_space)*100}% of our params space
({hyperopt_epochs} out of {len(params_space)}) possible combinations")
for _ in tqdm(range(hyperopt_epochs)):
    idx = np.random.randint(0, len(params_space))
    b, k = params_space[idx, 0], params_space[idx, 1]
    model = SIRModel(b, k, pop)
    model.fit(rolling_df_pa, fit_until=first_peak_day_cut-1,
interval=interval)
    score = sum(((model.dI + model.dR) - rolling_df_pa['newCases'][range(0,
first_peak_day_cut-1, interval)]/model.N)**2)
    if score < best["metric"]:
      best['b'] = b
      best['k'] = k
      best['metric'] = score
print("Best parameters found!")
print(f"b:\t{best['b']}")
print(f"k:\t{best['k']}")
model = SIRModel(best['b'], best['k'], pop)
model.fit(moving_average_df)
```

É interessante observar que a variável `HYPEROPT_EPOCHS` no código é pré-configurado pelo usuário, sendo ele o responsável por indicar quantas rodadas esse trecho irá rodar para calcular a melhor dupla de parâmetros b e k. No caso do nosso código, 210000 rodadas são calculadas para isso, o que equivale a, mais ou menos, 6 minutos otimizando a parametrização. Tal valor foi escolhido para que possamos achar de fato o melhor parâmetro sem que o programa consumisse muito tempo.
Vale notar que o nosso algoritmo de otimização possui uma complexidade de O(mn) onde **m** é o número de dias contidos no pico escolhido e **n** o número de rodadas do algoritmo configuradas pela variável mencionada acima.

Trecho do código que plota o gráfico SIR Model x True infected

```
y_col = 'infected' #@param ["suspected", "infected", "recovered"]

y_col_real = {
    "suspected": "suspected",
    "infected": "newCases",
    "recovered": "recovered"
}
y_col_sir = {
    "suspected": model.S,
    "infected": (-model.dS),
```

```
    "recovered": model.R
}
fig = px.line(x=moving_average_df['date'],
y=moving_average_df[y_col_real[y_col]], title='SIR Model x True Infected')
fig.add_scatter(x=moving_average_df['date'],
y=np.array(y_col_sir[y_col])*model.N)

fig.data[0].name = "Real Data"
fig.data[0].showlegend = True
fig.data[1].name = "SIR"
fig.show()
```

Trecho do código que plota o gráfico SIR Model x True suspected

```
y_col = 'suspected' #@param ["suspected", "infected", "recovered"]

y_col_real = {
    "suspected": "suspected",
    "infected": "newCases",
    "recovered": "recovered"
}
y_col_sir = {
    "suspected": model.S,
    "infected": (-model.dS),
    "recovered": model.R
}
fig = px.line(x=moving_average_df['date'],
y=moving_average_df[y_col_real[y_col]], title='SIR Model x True Infected')
fig.add_scatter(x=moving_average_df['date'],
y=np.array(y_col_sir[y_col])*model.N)

fig.data[0].name = "Real Data"
fig.data[0].showlegend = True
fig.data[1].name = "SIR"
fig.show()
```

Trecho do código que plota o gráfico SIR Model x True {y_col}

```
y_col = 'recovered' #@param ["suspected", "infected", "recovered"]

y_col_real = {
    "suspected": "suspected",
    "infected": "newCases",
    "recovered": "recovered"
}
y_col_sir = {
    "suspected": model.S,
    "infected": (-model.dS),
    "recovered": model.R
```

```
}
fig = px.line(x=moving_average_df['date'],
y=moving_average_df[y_col_real[y_col]], title='SIR Model x True Infected')
fig.add_scatter(x=moving_average_df['date'],
y=np.array(y_col_sir[y_col])*model.N)

fig.data[0].name = "Real Data"
fig.data[0].showlegend = True
fig.data[1].name = "SIR"
fig.show()
```

7 Bibliografia

Modelo SIR em python, usado como fonte para a realização do EP:
-https://github.com/Prudhvi-Kumar-Kakani/Data-Science-CRISP-DM--Covid-19/blob/908cb345c635cf8cfa265e98faa1d796b0836fd7/src/SIR/simulation.py

Método de Euler - Modelo SIR, usado como fonte para a construção da nossa classe SIRModel:
-https://www.maa.org/press/periodicals/loci/joma/the-sir-model-for-spread-of-disease-eulers-method-for-systems