UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO CIENTÍFICA
DEPARTAMENTO DE MATEMÁTICA APLICADA

DAVI ALVES MONTEIRO CARVALHO

Análise automática dos casos de COVID-19: Arquivo HTML

Campinas
11/01/2021

DAVI ALVES MONTEIRO CARVALHO

Análise automática dos casos de COVID-19: Arquivo HTML

Monografia apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos para obtenção de créditos na disciplina Projeto Supervisionado, sob a orientação do Prof. Alberto Saa.

Resumo

Esse trabalho foi realizado em meio à crise da pandemia da COVID-19 e teve como objetivo auxiliar o Professor Alberto Saa no desenvolvimento de um website que performasse uma análise dos dados da pandemia no Brasil. Os dados utilizados foram os disponibilizados pelo Ministério da Saúde [1] e a análise estatística foi feita com base no modelo SIR. Esta monografia trata do código usado para gerar a página HTML do *website*. Durante o projeto, foram elaborados os gráficos de mortes totais, mortes diárias, e razão entre mortes diárias e casos diários. Além disso, foram criados gráficos em formato de *gifs* e introduzida a computação em paralelo, que reduziu em cerca de 6 vezes o tempo de compilação do código.

Abstract

This project was developed during the COVID-19 crisis, and its objective was to assist Professor Alberto Saa in creating a website that performed an analysis of the data regarding the pandemic. The data used was provided by the Brazilian Ministry of Health [1], and the statistical analysis was performed based on the SIR model. This monograph will discuss the code used to generate the website's HTML page. Along the project, graphs of the death toll, daily deaths and the ratio between daily deaths and daily cases were elaborated. Beyond that, graphs in the gif format were created, and parallel computing was introduced in the code, which reduced the compilation time by approximately 6 times.

Conteúdo

1 Introdução

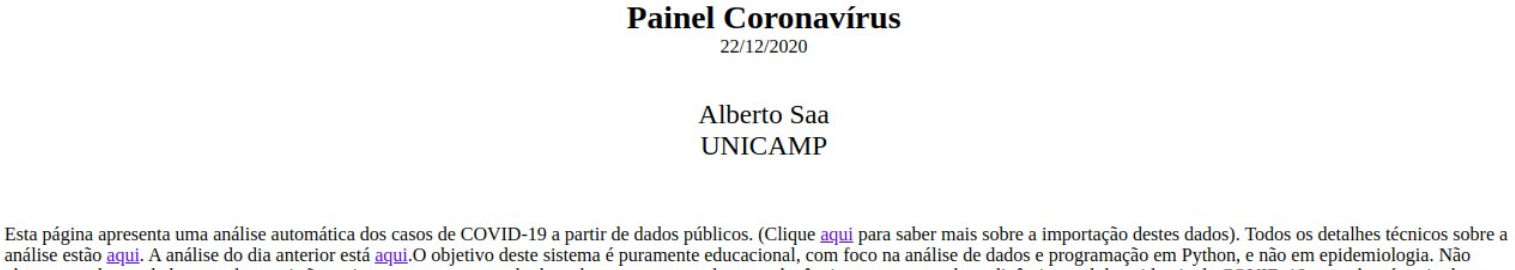
Atualmente o mundo se encontra em uma das piores crises de saúde da história da humanidade: a pandemia do coronavírus SARS-CoV-2, causador da COVID-19 . Mediante a esse problema na saúde pública, faz-se necessário a elaboração de ferramentas eficientes de análise de dados para combater a doença. Esse projeto se dedica ao uso do modelo estatístico SIR para o estudo de casos de COVID-19 no Brasil, a partir de dados disponibilizados pelo Ministério da Saúde. As análises são inseridas em um documento HTML e disponibilizadas via *webpage* [5]. A autoria do projeto pertence ao Professor Alberto Saa, e coube aos alunos Davi Carvalho, Sabrina Zani e Jefferson da Silva o auxílio no desenvolvimento da página web. Essa monografia tratará somente da elaboração do algoritmo responsável pelo arquivo HTML. Para conhecer o modelo estatístico e o tratamento de dados usados, vide [6] e [2].

2 O código

O código deste projeto foi escrito totalmente em Python 3, fazendo uso das bibliotecas *imageio, numpy, codecs, hashlib, scipy.stats, os, matplotlib.pyplot, csv, datetime, io, time, zipfile, requests, joblib.Parallel, joblib.delayed*, disponíveis na plataforma Anaconda. Ele está disposto majoritariamente em dois arquivos [4], um contendo as funções utilizadas (painelCOVID.py) e o outro contendo o código principal (COVID.py). Ao rodar o código, é gerada uma página HTML [5] que contém a análise diária dos dados disponibilizados pelo Ministério da Saúde e textos introdutórios e explicativos sobre elas. A seguir será feito um resumo do código, e os detalhes das funções se encontrarão nas seções seguintes.

O programa começa iniciando as variáveis *alpha*, *gamma1*, *gamma2*, *date* e *date1*. Os parâmetros *gamma1* e *gamma2* são os limites inferior e superior de γ, *date* é a data dos dados da análise atual e *date1* é a data dos dados da última análise. É necessário ter os dados da última análise porque o Ministério da Saúde disponibiliza apenas o número total de casos e de mortos, então, para encontrar mortes diárias e casos diários, é preciso subtrair os dados atuais dos dados da última análise. As informações são baixadas do site do Ministério [1] para o repositório [4]. A partir desse ponto, elas são tratados pela função *read_github*.

Após a obtenção e tratamento dos dados pela função *read_github*, é aberto um documento HMTL e escreve-se uma mensagem inicial através da função *write_opening*. Em seguida, são criados os diretórios *graf* e *gifs*, que armazenam os gráficos e gifs. Finalmente, a análise dos dados para o Brasil é feita com a função *graf_brasil*.


Painel Coronavírus
22/12/2020

Alberto Saa
UNICAMP

Esta página apresenta uma análise automática dos casos de COVID-19 a partir de dados públicos. (Clique aqui para saber mais sobre a importação destes dados). Todos os detalhes técnicos sobre a análise estão aqui. A análise do dia anterior está aqui.O objetivo deste sistema é puramente educacional, com foco na análise de dados e programação em Python, e não em epidemiologia. Não obstante, todos os dados tratados aqui são reais e, portanto, os resultados talvez possam ter alguma relevância para se entender a dinâmica real da epidemia de COVID-19, a qual está muito bem analisada, por exemplo, aqui. Os dados e códigos necessários para gerar esta página estão aqui, sinta-se à vontade para utilizá-los como quiser.

Análise realizada a partir dos dados do Ministério da Saúde, clique aqui para mais detalhes.

Figura 1: Mensagem inicial impressa pela função *write_opening*.

Quando a análise de dados para o Brasil é concluída, o programa passa para a análise de casos e mortes por milhão em cada estado, realizada pela função *write_js*. Depois são feitas as demais análises para as cidades e estados pela função *graf*. Ao longo dessa monografia, os casos e mortes diárias serão chamados de novos casos e novas mortes, enquanto que o número total de mortes e casos serão chamados de mortes acumuladas e casos acumulados.

2.1 Funções *read_github* e *write_js*

A função *read_github* é responsável por extrair as informações do arquivos salvo no repositório [4], correspondente à data salva na variável *date*. Ela acessa o arquivo no formato *raw*, o salva em uma pasta *zip* e armazena o conteúdo na lista *linecsv*. Depois, ela cria uma um dicionário chamado *dict_estados*, que possui os estados como chaves e os valores são vazios. No final, a função retorna *linecsv* e *dict_estados*.

A função *write_js* é responsável por gerar os dados necessários para os cálculos de r_0, e de mortes e casos por milhão para cada estado, além de graficar essas informações. As informações são dispostas em dois gráficos, um de casos por milhão e outro de mortes por milhão, com cada curva representando um estado. Ao colocar o *mouse* sobre a curva de um estado, visualiza-se o r_0 efetivo médio.

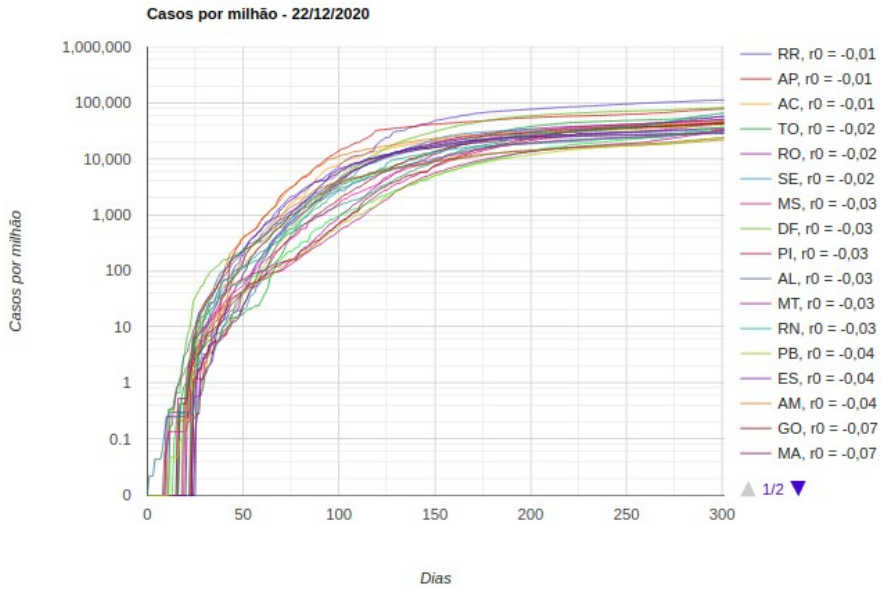


Figura 2: Gráfico de casos por milhão criado pela função *write_js*.

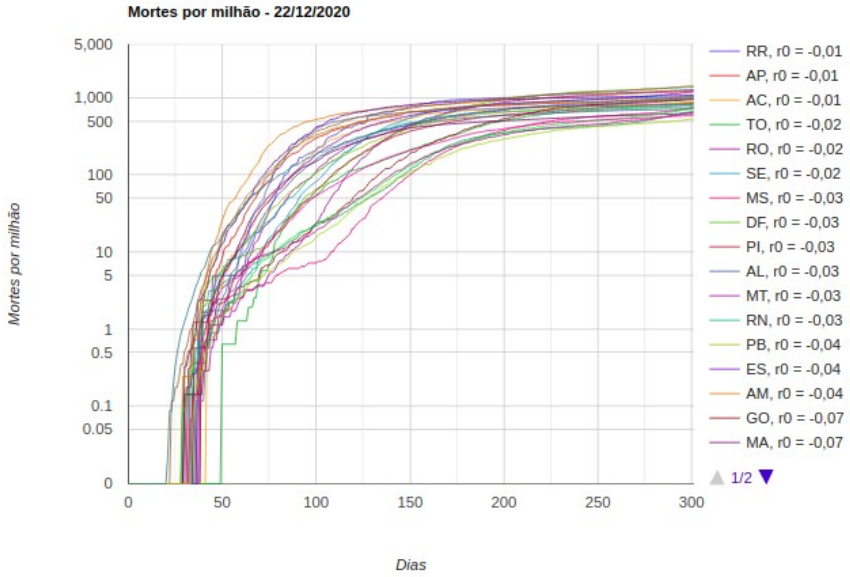


Figura 3: Gráfico de mortes por milhão criado pela função *write_js*

2.2 Funções *graf* e *graf_brasil*

Será feita primeiramente a explicação da função *graf*. O objetivo dessa função é fazer a análise dos casos para os estados e as cidades. As informações inclusas na análise são: número de casos e mortes (totais e por milhão de habitantes), cálculos para o r_0, previsão do número total de casos para os próximos 5 dias e gráficos. Detalhes sobre os gráficos encontram-se na subseção 2.3.

A função inicia chamando a função *read_csv_data*, que lê os elementos na lista *linecsv* e retorna as séries temporais para os casos acumulados e mortes acumuladas, o número de elementos na série, as datas do primeiro e últimos casos e a população. Os dados são salvos no dicionário *res* e posteriormente têm seus valores armazenados em variáveis locais dentro de *graf*.

Após o tratamento feito pela função *read_csv_data*, são calculadas as quantidades de novos casos a cada dia da série, subtraindo a quantidade de casos acumulados no dia pela quantidade de casos acumulados do dia anterior. O mesmo é feito para a quantidade de novos óbitos no dia. Logo após, são feitas as suavizações das séries de casos e óbitos usando janelas de uma semana, através da função *smooth*. Para a quantidade de casos, dois tipos de suavização são realizadas: com duas iterações e com quatro interações.

Finalmente, é feito a quantidade de novos casos e novos óbitos para os dados suavizados.

Depois das suavizações, são realizadas as previsões de casos para os próximos cinco dias. A estratégia adotada para realizar as previsões é fazer uma regressão linear dos casos para os últimos dez dias, e usar o resultado obtido para prever a quantidade de casos para os próximos cinco dias. A regressão linear é realizada através da função *linregress*, pertencente à biblioteca *scipy.stats*, e as previsões são armazenadas na lista *R_prev*.

Quando o cálculo das previsões é concluído, a função passa a calcular os valores de r_0. A função R_0dif encontra, para cada dia, os valores de r_0 para *gamma1* e *gamma2* pelo método diferencial. Semelhantemente, a função R_0int encontra os valores de r_0 para *gamma1* e *gamma2* pelo método integral. O r_0 exibido na análise será o calculado pelo método integral, então encontra-se a média (r_0 efetivo médio) e o desvio padrão desses coeficientes para as duas últimas semanas, e os valores máximo e mínimo do r_0 para o último dia analisado. Através do r_0 efetivo médio, calcula-se o limiar imunidade de grupo (nR).

Após os cálculos do r_0, é definida um hashing e armazenado na variável *regfile*, e uma variável *regstr*, que identifica o estado ou a cidade. A variável *regfile* é usada para dar nome aos gráficos gerados. Em seguida, os gráficos e gifs são criados, e uma página HMTL secundária (*html_file_local*) é gerada. Finalmente, cria-se um dicionário armazenado na variável *write_dict* com todas as informações obtidas até agora e, através da função *write_analise*, essas informações são colocadas no *html_file_local*. A função não retorna nada.

A função *graf_brasil* é bem semelhante à função *graf*. A diferença entre ambas está na forma que elas disponibilizam as informações nos documentos HMTL. Para o Brasil, a análise é disponibilizada na página HMTL principal, enquanto para os estados e cidades, as informações são disponibilizadas em páginas HMTL secundárias. Além disso, as variáveis *R_smooth2* e *dR_smooth2* calculadas pela função *graf_brasil* são usadas na função *write_js*, logo a função retorna esses valores. No restante, ambas as funções são iguais. O modelo da análise para o Brasil, estados e cidades encontra-se no Anexo.

2.3 Gráficos

Todos os gráficos, com exceção dos produzidos pela função *write_js*, são produzidos pelas funções *graf* e *graf_brasil*. São onze gráficos ao todo: casos acumulados, mortes acumuladas, novos casos, novas mortes, casos por semana, mortes por semana, casos e mortes por milhão, razão entre novas mortes e novos casos, r_0 efetivo diferencial, r_0 efetivo integral, razão μ.

Os gráficos de casos acumulados, mortes acumuladas, novos casos e novas mortes são plotados na forma de um gráfico de barras, que mostra os dados da forma que são disponibilizados pelo ministério da saúde, e um gráfico de linhas, que mostra os dados suavizados para uma janela de uma semana com 4 iterações do filtro de média movel. Os dados do ministério da saúde mostram apenas os casos totais até determinado dia, então, para encontrar novas mortes e novos casos, subtrai-se os valores do dia atual pelos valores do dia anterior. Eles foram graficados no formato de gif, de forma que o gif roda com apenas 1 loop quando a página é aberta. A figura 4 mostra um exemplo.

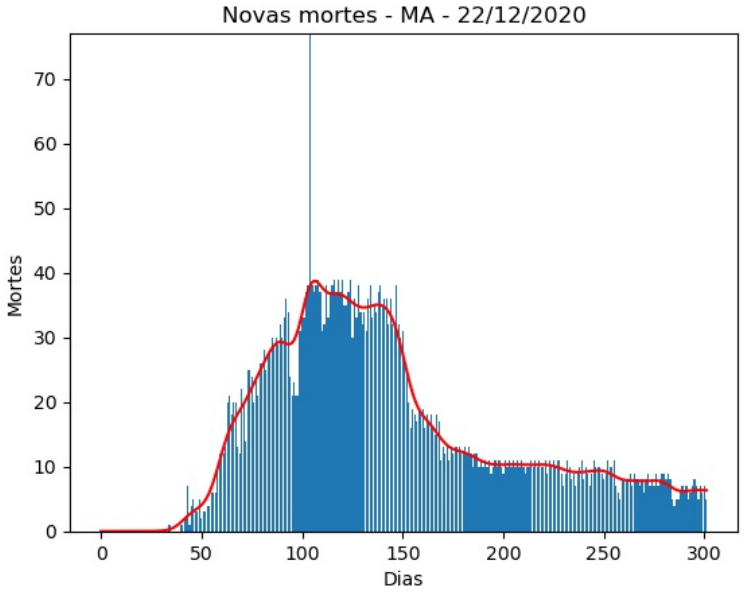


Figura 4: Gráfico de novas mortes para o estado do Maranhão.

Os gráficos de casos por semana e mortes por semana são plotados em forma de gráficos de barras, mostrando o total de casos em determinada semana da pandemia. Caso a semana atual da análise ainda não tenha se encerrado, a última barra do gráfico é gerada em uma cor diferente, mostrando que ainda é uma semana em andamento. Esses

gráficos foram plotados no formato de gif, de forma que o gif roda com apenas 1 loop quando a página é aberta. A figura 5 mostra um exemplo.

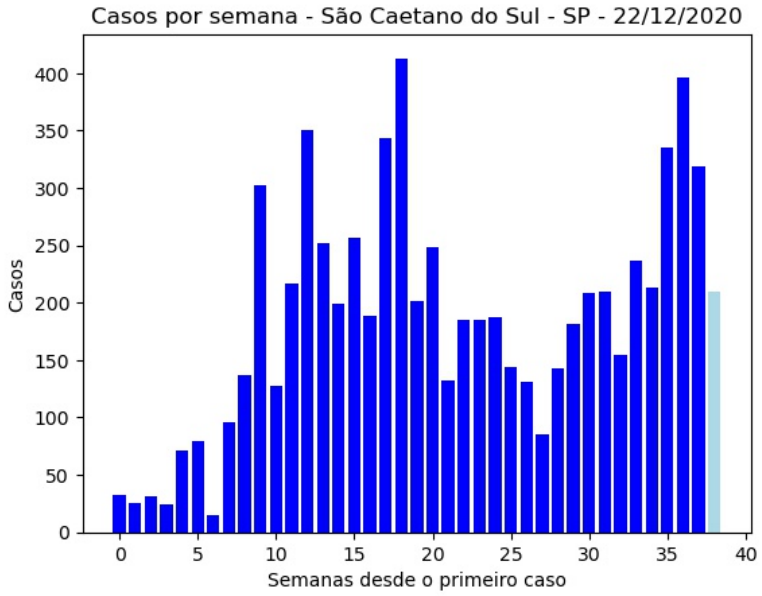


Figura 5: Gráfico de casos por semana para a cidade de São Caetano do Sul - SP.

Os dados de casos e mortes por milhão de habitantes servem como uma base comparativa entre estados e cidades, colocando todos em um mesmo referencial populacional. Esses dados são gerados pela seguinte relação

$$dados_{pm} = 10^6 \cdot \frac{dados_{MS}}{Popul} \tag{1}$$

onde $dados_{pm}$ são os dados (casos ou mortes) por milhão de habitantes, $dados_{MS}$ são os dados disponibilizados pelo Ministério da Saúde e $Popul$ é a população total analisada. Os dados de casos e mortes são plotados no mesmo gráfico, com uma curva para cada, em escala logarítmica, de forma que o último ponto de ambos coincidem. Dessa forma, caso as curvas se afastem, há uma mudança na relação entre mortos e contaminados a cada 1 milhão de habitantes. A figura 6 mostra um exemplo.

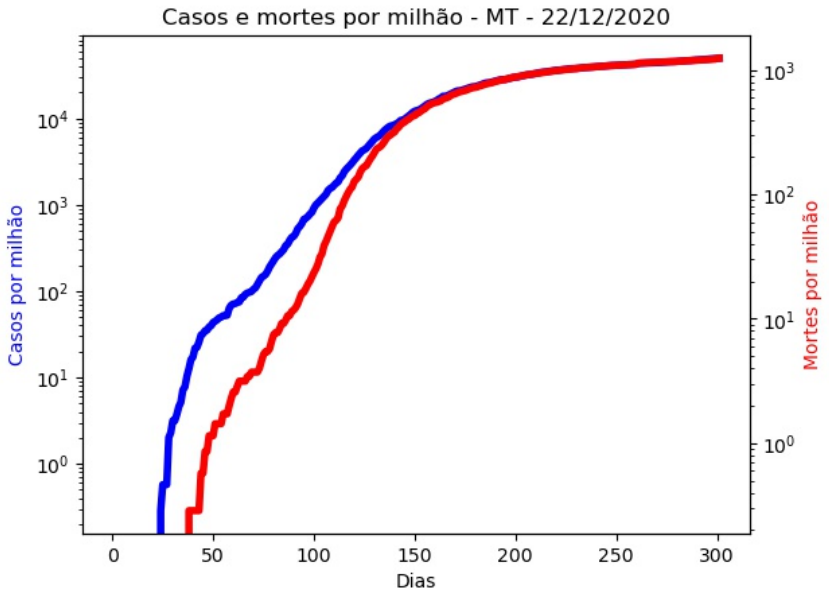


Figura 6: Gráfico de casos por semana para a cidade de São Caetano do Sul - SP.

O gráfico da razão entre novas mortes e novos casos tem como propósito analisar o progresso da letalidade da COVID-19: o aumento da razão indica que os vírus está matando mais em relação ao número de casos. Os valores dessas razões foram obtidos pela seguinte relação

$$razao_{NM/NC} = \frac{novas\_mortes}{novos\_casos} \tag{2}$$

onde *novas_mortes* e *novos_casos* são os mesmos dados plotados pelos gráficos *novas mortes* e *novos caso*. Esses valores foram graficados de forma discreta, e a suavização (janela de uma semana e duas iterações da média móvel) foi representada por um gráfico de linha. A figura 7 mostra um exemplo.

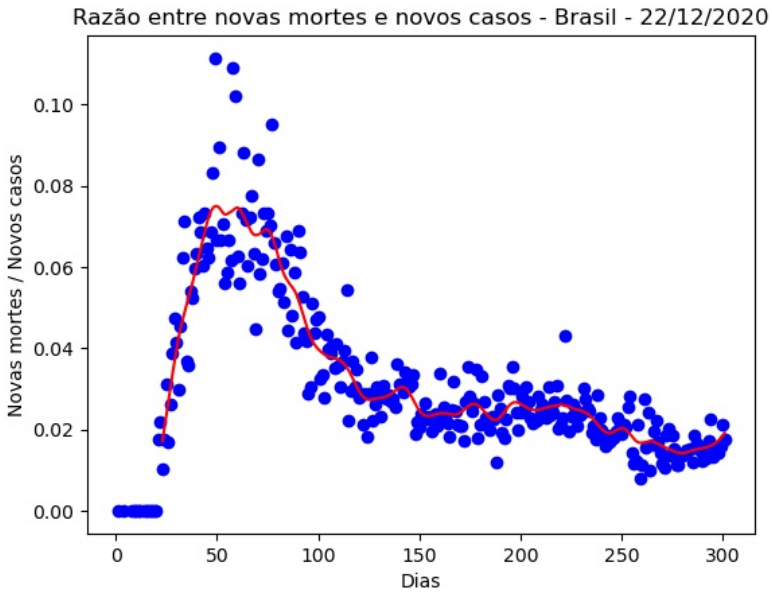


Figura 7: Gráfico da razão entre novas mortes e novos casos para o Brasil.

Os gráficos de r_0 efetivo diferencial e r_0 efetivo integral mostra uma região delimitada por duas curvas, cada uma calculada em um extremo de γ. Logo, a região dentro da curva mostra todos os valores possíveis de r_0. O gráfico mostra uma reta em azul plotada no valor de $r_0 = 1$. Esse valor é significativo porque é o limite que determina se a pandemia está em expansão ou em retração. A figura 7 mostra um exemplo.

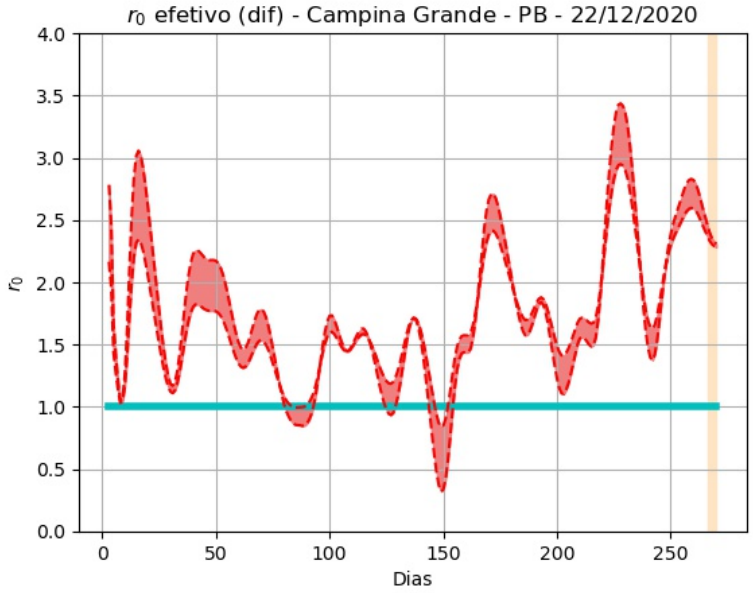


Figura 8: Gráfico do r_0 efetivo diferencial para Campina Grande - PB.

Da mesma forma que os gráficos de r_0, o gráfico de μ exibe uma região delimitada por duas curvas, que são os valores de μ calculados nos valores extremos de γ. Dessa forma, a região exibe todos os valores possíveis de μ. A figura 7 mostra um exemplo.

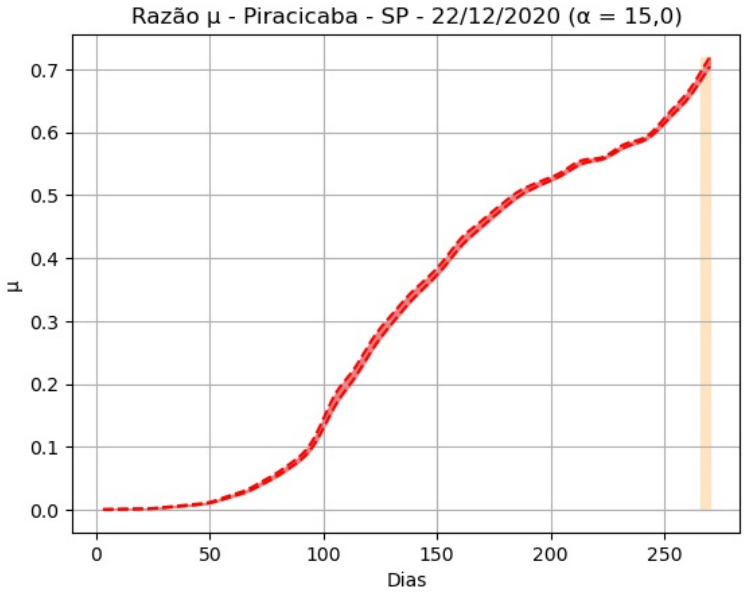


Figura 9: Gráfico do μ para Piracicaba - SP.

2.4 Computação paralela

A execução dos códigos deste trabalho foi realizada na máquina Lugo do Instituto de Matemática e Computação Científica da Unicamp, que possui 48 núcleos de processamento. Foi necessário usar essa máquina porque a criação de gifs tornou o algoritmo muito custoso para ser rodado no computador pessoal. Para acelerar o processamento do código, foi usado o processo de computação paralela [3].

Tradicionalmente, os softwares são projetados para funcionar de forma serial, o que significa que, para uma determinada tarefa, apenas um núcleo de processamento é usado, e as instruções do algoritmo são executadas uma de cada vez. Na computação paralela, o algoritmo é dividido em tarefas menores (*jobs*) que são executadas simultaneamente por vários núcleos de processamento. Essas tarefas precisam ser independentes, ou seja, o resultado da execução de uma das partes não pode depender das outras.

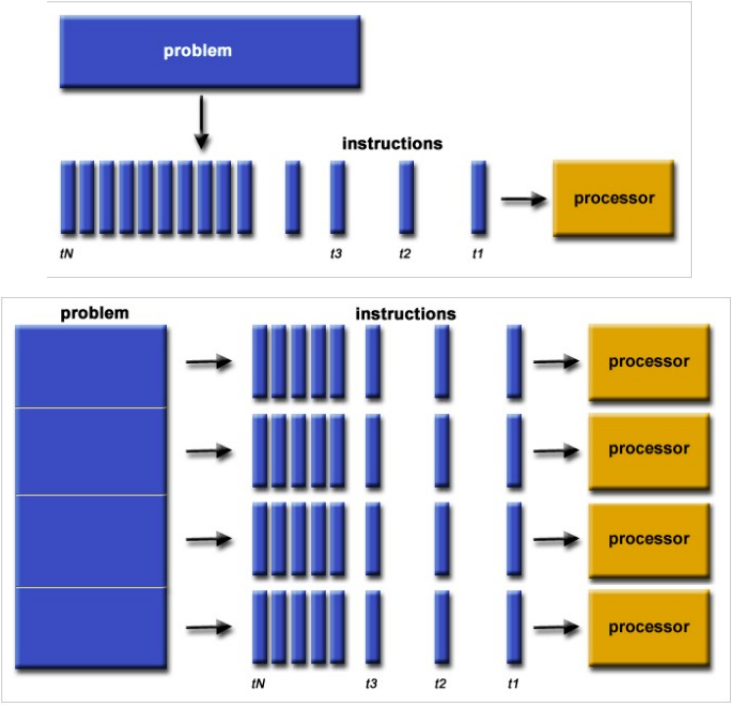


Figura 10: Acima, ilustração do processo de computação sequencial. Abaixo, ilustração do processo de computação paralela. Figura retirada de [3].

No algoritmo usado, a paralelização do processamento foi usado ao chamar a função *graf*, tanto na análise dos estados quanto na análise das cidades. Foram analisados 27 estados (26 estados e o Distrito Federal) e 41 cidades, então, para os estados, o processamento foi dividido em 27 *jobs* e, para as cidades, 41 *jobs*. A paralelização foi possível porque as análises dos estados e das cidades são independentes entre si. Para realizar a paralelização, foram usadas as funções *Parallel* e *delayed*, da biblioteca *joblib*, conforme figura 11.

```
if __name__ == "__main__":
    Parallel(n_jobs=41)(delayed(painel.graf)(reg,linecsv,gamma1,gamma2,alpha,
    reglist,date,graf,gifs,dict_estados) for reg in reglist)
```

Figura 11: Funções *Parallel* e *delayed* usadas na análise das cidades.

A função *Parallel* inicia uma instância com paralelização em n núcleos, onde n é dado pela variável *n_jobs*. No parênteses ao lado, cria-se uma lista cujos elementos são a função *graf* avaliada em cada estado (ou em cada cidade), e essa lista é passada para a função *Parallel*, a fim de que os elementos sejam organizados em tarefas. A função

delayed indica que, apesar de a função *graf* estar sendo chamada, ela ainda não deve ser executada, para que a função não seja executada antes de passar pelo processo de paralelização.

3 Resultados

Como dito anteriormente, o objetivo desse trabalho foi auxiliar o Professor Alberto Saa no desenvolvimento do código gerador da página HTML. Em relação à primeira versão do código, foram acrescentados gráficos de mortes acumuladas, novas mortes, e razão entre novas mortes e novos casos. Além disso, foi incorporado no código principal o algoritmo de geração de gifs, elaborado pela aluna Sabrina Zani, e implementada a computação em paralelo, que trouxe uma grande economia de tempo à compilação do código. Em sua fase de teste, o código foi implementado para gerar apenas o gif do gráfico de novos casos. Antes da paralelização, o algoritmo levava cerca de 1h30min para ser compilado. Após a paralelização, ele levava apenas 15 minutos, ou seja, o tempo de compilação foi reduzido em cerca de 6 vezes.

4 Conclusão

Ao longo deste projeto, foi aprimorado o código de autoria do Professor Alberto Saa, com o objetivo de realizar análises dos casos de COVID-19 no Brasil. Foi possível ver a utilização prática do modelo epidemiológico SIR para a análise do comportamento de uma patologia, aprender o processo de criação de imagens animadas (*gifs*) e também verificar a eficiência da computação paralela na otimização do tempo de compilação de códigos.

Referências

[1] Ministério da Saúde, 2020. url: https://covid.saude.gov.br/.

[2] Jefferson da Silva. Análise automática dos casos de covid-19: Análise estatística. 01 2021.

[3] Lawrence Livermore National Laboratory. *Introduction to Parallel Computing Tutorial*. URL: https://hpc.llnl.gov/training/tutorials/introduction-parallel-computing-tutorialWhatis.

[4] A. Saa. 2020. url: https://github.com/albertosaa/COVID.

[5] A. Saa. Análise automática do painel coronavírus, 2020. url: http://vigo.ime.unicamp.br/COVID/.

[6] Sabrina Zani. Análise automática dos casos de covid-19: Modelo sir. 01 2021.

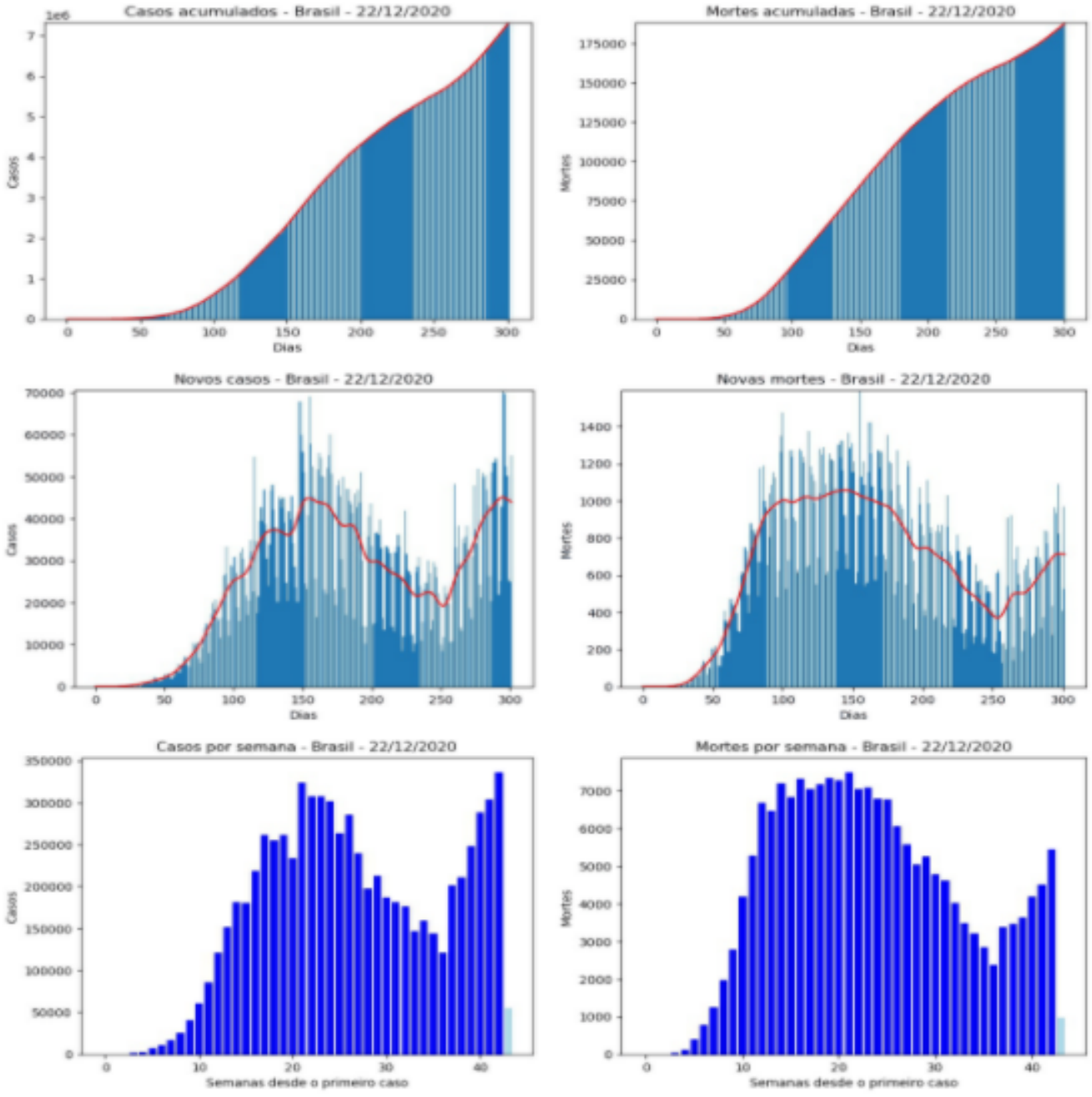
Anexo: análise gerada pelas funções *graf* e *graf_brasil*

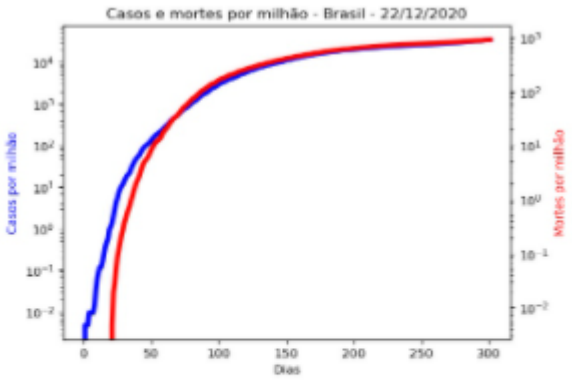
O mesmo modelo de análise exibido aqui para o Brasil é feito para os estados e cidades.

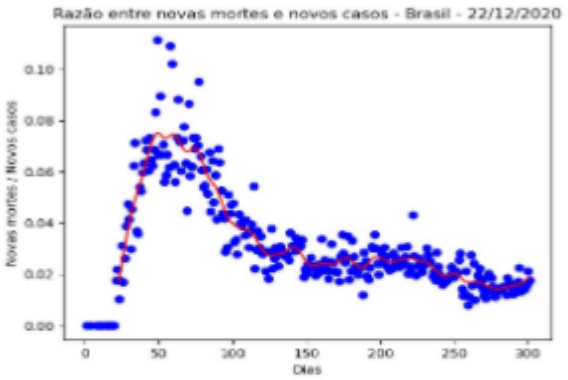
Brasil - 22/12/2020.

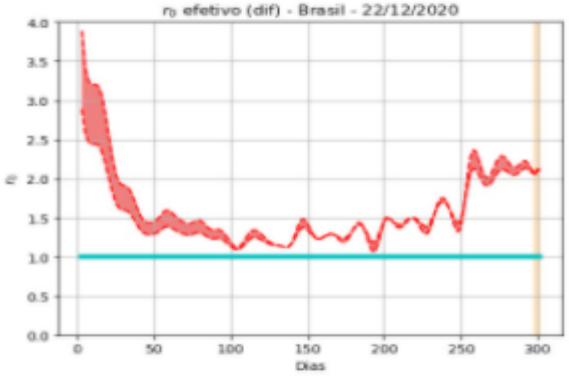
Detalhes técnicos, aqui. Clique aqui para uma versão em PDF desta análise.

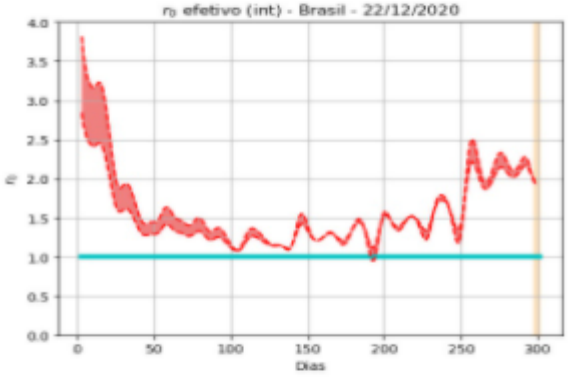
População: 210.147.125. Início e fim da série: 2020-02-25 e 2020-12-22. (302 elementos - 43 semanas e 1 dia).
Número de casos totais e mortes: 7.318.821 e 188.259. (34.827 e 896 por milhão de habitantes, respectivamente.)
r_0 (integral) efetivo médio (duas últimas semanas - três dias de atraso): 2,11 (std = 0,09). Ultimo intervalo para r_0 (três dias de atraso): (1,93 : 1,97).
Limiar imunidade de grupo n_R (baseado no valor de r_0 (integral) efetivo médio) = 0,53.
Previsão do número total de casos para os próximos 5 dias: 7.367.427, 7.416.034, 7.464.641, 7.513.248, 7.561.855.




Casos e mortes por milhão - Brasil - 22/12/2020
Casos por milhão
Mortes por milhão
Dias


Razão entre novas mortes e novos casos - Brasil - 22/12/2020
Novas mortes / Novos casos
Dias


r_0 efetivo (dif) - Brasil - 22/12/2020
r_0
Dias


r_0 efetivo (int) - Brasil - 22/12/2020
r_0
Dias


Razão μ - Brasil - 22/12/2020 (α = 15,0)
μ
Dias