Dengue Free Feira: Sistema de Controle da Dengue

Pedro Lucas F. de Souza

[1]UEFS – Universidade Estadual de Feira de Santana, Av. Transnordestina, s/n, Novo Horizonte, Feira de Santana – BA, Brasil – 44036-900

pedrofernandeseng.comp@gmail.com

Abstract. *This article describes and reports the development and operation of a system in Python for dengue case control statistics in Feira de Santana, Bahia. It's accountable for computing statistics relevant to disease control, such as suspected, negative and positive cases and their percentage in relation to the general population or specific ones in smaller regions, such as neighborhoods. It can be configured to insert new data, consult existing ones, and calculate percentages of total data by neighborhoods and data, in addition to present progression rates of positive and negative cases in a time interval. The development of this type of software demands that data structures be changeable and compatible to comply with the necessities and mathematical models for monitoring and statistics pertinent to disease control. This article also aims to define and show the applicability of these structures and their relevance in the development of this disease control system, such as by describing the tests and errors made and presenting the book of instructions for this system.*

Resumo. *Este artigo descreve e relata o desenvolvimento e funcionamento de um sistema em linguagem Python de estatísticas de controle de casos de dengue na cidade de Feira de Santana, Bahia. Ele computa estatísticas pertinentes ao controle de doenças, como casos suspeitos, negativados, positivados e sua percentualidade em relação à população geral ou específica em regiões menores, como bairros. Ele pode ser configurado para incrementar novos dados, consultá-los e calcular percentuais de dados totais especificados por bairros e datas, além de apresentar taxas de progressão de casos positivados e negativados em um intervalo de tempo. O desenvolvimento desse tipo sistema demanda de estruturas de dados alteráveis e computáveis para atender às necessidades e modelos matemáticos para o monitoramento e estatísticas pertinentes ao controle de doenças. Este artigo visa também definir e mostrar a aplicabilidade dessas estruturas e sua relevância no desenvolvimento do sistema de controle de doenças tal como descrever os testes e erros feitos e apresentar o manual de uso desse sistema.*

1. Introdução

Durante a história da humanidade, as civilizações foram expostas à doenças transmissíveis, enfermidades transmitidas por objetos infectados ou entre indivíduos. Ao longo da história, houve inúmeros exemplos históricos de doenças desse tipo, como a famigerada peste negra, ocorrida em meados do século XIV na Europa, causada pela bactéria *Yersinia pestis*, transmitida por pulgas presentes nos ratos em barcos comerciais, o que facilitou a difusão da doença através do comércio marítimo feito entre os portos da Europa

[Silva 2024]. Tal enfermidade é um tipo específico de doença transmissível conhecida como *zoonose*, caracterizada pela transmissão do agente causador da doença feita dos animais para os humanos [Secretária de Saúde do Paraná 2024].

Há vários tipos de zoonoses ao redor do mundo, pois os transmissores podem ser endêmicos ou não de regiões específicas do planeta. Nesse sentido, no contexto brasileiro, a dengue é uma das mais conhecidas zoonoses em território nacional. A doença faz parte do grupo dos arboviroses, isto é, doenças causadas por vírus transmitidos por artrópodes. O vetor de transmissão[1] é o mosquito *Aedes aegypti*, oriundo da África, trazido pelos navios negreiros. Os fatores ambientais, tal como clima tropical, saneamento básico deficiente, urbanização crescente e aumento populacional favorecem a reprodução desses vetores, favorecendo o aumento de ocorrência da doença. A dengue é causadora da doença febril aguda, caracteriza por febres de 39ºC a 40ºC, dores musculares e articulares, cefaleias e dores atrás dos olhos, podendo trazer a vítima à óbito em alguns casos. [Brasil 2024]

Nesse sentido, para além do uso da medicina para aplicar terapias e tratamentos a fim de combater a doença, a computação faz-se necessária para realizar o monitoramento da doença, como sua taxa de progressão ou regressão ao longo do tempo em regiões gerais ou específicas. Essa ação é fundamental para auxiliar os órgãos públicos e seus servidores na elaboração de estratégias para combater os vetores de transmissão e diminuir a incidência da doença nas localidades, como melhora da infraestrutura de saneamento básico e aplicação de vacinas [IPM 2024]. A computação, em específico, monitora a doença através de dados e modelos matemáticos que lidam com registros de novos casos, computação de dados estatísticos e até taxa de progressão ou regressão da doença. Ademais, este artigo apresenta e descreve um sistema de controle capaz de monitorar a dengue ao longo do tempo mediante registros de novos dados, que são computados a fim de obter a progressão ou regressão da doença na cidade de Feira de Santana, tal como seus bairros, e a taxa de incidência nessas localidades.

2. Metodologia

2.1. Referencial teórico

O desenvolvimento do sistema de controle de dengue que envolve análise de dados depende de estruturas computacionais mais complexas, capazes de armazenar enormes quantidades de dados. Tais estruturas devem ser ordenáveis, manipuláveis e computáveis, além de serem organizáveis em conjunto. Em linguagem interpretativa Python, elas são conhecidas como listas e matrizes. As listas são responsáveis por armazenar elementos, sejam *inteiros*, *pontos flutuantes* ou *strings*, ordenados por índices relativos a sua posição ao longo da estrutura. Elas são manipuláveis, incrementando, permutando ou removendo dados; e são também iteráveis[2] e computáveis, buscando elementos por índice, tipo e valor e armazenando-os em variáveis. Elas podem armazenar outras listas, e esse aninhamento é chamado de matriz e possui as mesmas propriedades [Menezes 2016]. Além dessas estruturas, há também os dicionários, responsáveis por armazenar encadeamentos de elementos em pares *chave-valor*. Nessas estruturas, os pares podem ser adicionados ou retirados, e, diferentemente das listas, os valores estão associados às chaves, não a

[1] Vetor de transmissão é o organismo responsável por transmitir o patógeno.

[2] Iteráveis são objetos que podem ser percorridos elemento por elemento, geralmente em *loop*.

sua posição ordenada. Isso auxilia em mecanismos de busca, pois os valores podem ser chamados a partir de suas chaves [Menezes 2016].

O dado de entrada é uma matriz de dados relativos às estatísticas da dengue, como datas, bairros, população, casos suspeitos, negativos e positivos. Esses dados estão armazenados em um arquivo do tipo *csv*, que armazena dados estruturados em linhas que possuem elementos separados por vírgula. Em Python, o arquivo é lido e retorna uma matriz com todas as listas e seus elementos formatados em *string*, que podem ser convertidos para *inteiro* ou *ponto flutuante* a depender da necessidade. Essa operação é feita a partir do uso da biblioteca *csv*, que oferece o conjunto de ferramentas para realizar a leitura de arquivos desse tipo e reescrevê-los com novos dados, armazenando as informações mesmo após o encerramento do sistema [Python Software Foundation 2024]. Além da biblioteca *csv*, o desenvolvimento do sistema necessita de validação e implementação de novas datas para os novos dados, então é necessário o uso da biblioteca *datetime*, nativa do Python e utilizada para fazer alteração e validação de datas [Python Software Foundation 2024].

O código-fonte do sistema é ordenado por estruturas de comando responsáveis por executar tarefas, como implementação de novos dados, busca e visualização de informações, além da computação de estatísticas. Essas estruturas são chamadas de funções e são responsáveis por executar comandos a partir de parâmetros de entrada, computá-los e retornar valores. Elas são úteis para isolar instruções específicas e estruturar melhor o código-fonte [Menezes 2016].

2.2. Desenvolvimento do sistema

O código-fonte é estruturado em blocos gerais de funções responsáveis por cada parte do sistema, como leitura do arquivo *csv*, layout, visualização, mecanismos de busca, implementação de dados e computação de estatísticas; além do programa principal, responsável por iniciar e finalizar o sistema. Nesse sentido, o bloco referente ao arquivo *csv* contém duas funções desenvolvidas responsáveis por sua leitura e reescrita. A primeira função ler o arquivo e retorna todos os dados em formato de listas, que são armazenadas em uma matriz. Em caso de implementação de novos dados, estes são acrescidos na matriz e reescritos no arquivo pela segunda função, mantendo-o atualizado.

O programa principal é a estrutura do código-fonte que executa todas as funções conforme as necessidades do usuário. Os comandos são dados por meio das entradas numéricas e verificados para atender aos condicionais e executar determinada tarefa. O programa principal fundamenta-se em uma variável de inicialização de valor *booleano*[3] verdadeiro, responsável por manter um *loop* que assegura a constante execução do programa principal. Somente com a entrada específica do usuário, a variável é falseada, e o programa é encerrado.

O bloco de funções responsável pelo layout, que interage constantemente com o programa principal, imprime as informações e os menus que orientam o usuário, formando a interface interativa do sistema. Elas imprimem os menus principal, de relatório e de busca e são constituídas por comandos de *print*, que exibem os menus e informações relativas à dengue, e *input* de números inteiros, que retornam os valores de entrada do usuário conforme a opção desejada. O bloco responsável pela visualização contém ape-

[3]Em computação, é um dado que pode ter dois valores lógicos: verdadeiro ou falso.

nas a função que itera a matriz dos dados solicitados, o parâmetro da função, e imprime-os em formato tabular a fim facilitar a leitura do usuário.

Os mecanismos de busca são divididos em três tipos: para data e bairro específicos; para data específica; e para intervalo de datas. Essas funções recebem bairros e datas como parâmetros, fornecidos pelo usuário. A partir disso, as funções iteram toda matriz de dados e retornam dados correspondentes aos parâmetros através de condicionais. Esses são utilizados como parâmetros de funções de visualização e tratamento de dados. Para garantir a entrada correta dos parâmetros e evitar erros de digitação, foram necessárias duas funções: uma para a verificação de datas; outra, para retornar corretamente as *strings* referentes aos bairros. Por meio da biblioteca *datetime*, a entrada de data é validada mediante comparação com as datas já inseridas na matriz, evitando a inserção de datas fora do intervalo de tempo, o que retornaria valores vazios. Para evitar erro de digitação dos bairros, é criado um dicionário para formar pares chave-valor, em que as chaves são números inteiros positivos; e os valores, bairros. Isso permite que o usuário digite o número, que passa pelo crivo do tratamento de erros, e retorne o valor correspondente, garantindo a entrada dos parâmetros corretamente.

O bloco responsável pela implementação de dados é constituído por uma única função que recebe a matriz de dados como parâmetro. A inserção de dados novos é dependente dos dados da data anterior, sendo assim, os novos dados são computados em função dos anteriores. Contudo, existem dois dados que são inalteráveis: o bairro e sua população, que são iguais ao longo do conjunto de dados. Os dados dependentes são a data, que depende da data anterior; os casos suspeitos, que são sempre novos; e os casos positivos e negativos, que dependem dos casos suspeitos do dia anterior, pois os casos suspeitos podem ser positivados ou negativados na data seguinte. Sendo a matriz uma estrutura ordenada por índice, a chamada pode ser feita por índices numéricos inteiros, sejam positivos ou negativos. Enquanto os positivos retornam os dados primeiros até os últimos; os negativos fazem o contrário. Então, para retornar os últimos dados da última data, a função utiliza índices negativos. A função cria uma matriz auxiliar que armazena os novos dados, chama o valor da última data, muda-a para a data do dia seguinte, através da biblioteca *datetime*, e a função itera os dados a partir do último elemento até o primeiro da última data e solicita do usuário os novos dados, validados e computados, e são armazenados juntamente com os dados fixos em uma lista, adicionada à matriz auxiliar, e o processo se repete para todos os bairros. A matriz auxiliar, para manter a boa ordenação dos bairros, é revertida através da função *reverse* [Python Software Foundation 2024], e seus itens são iterados e adicionados à matriz de dados original.

O cálculo das estatísticas é feita por um bloco de funções que computam os dados referentes aos tipos de buscas. Para a busca conforme data e bairro específico; a função responsável recebe a lista com esse dado e chama seus valores através de índices, que são convertidos em inteiros e têm seus valores percentuais computados e impressos na tela. Nesse caso, os dados calculam os percentuais de casos suspeitos e positivos em relação à população do bairro. Para a busca em data específica, a função recebe como parâmetro a matriz com esses dados e, analogamente ao processo anterior, computa a percentualidade dos dados e imprime o resultado. Nesse caso, os percentuais são referentes à quantidade de casos suspeitos, negativos e positivos em relação ao total de casos notificados, que é a soma de todos os tipos de casos na data especificada. Para a busca em intervalo de datas,

os valores retornados das datas inicial e final e, analogamente aos outros tratamentos de dados, têm seus dados validados, computados e impressos. A percentualidade dessa busca calcula a diferença de casos positivos e negativos entre a data inicial e final tal como sua diferença percentual, isto é, a variação percentual entre as datas. As porcentagens foram padronizadas e arrendondas para até duas casas decimais após vírgulas com o uso da função *round* [Python Software Foundation 2024].

3. Resultados e discussões

3.1. Manual de uso

Para executar o programa, é necessário que o arquivo *csv* de entrada esteja no mesmo diretório que o código-fonte, além de um interpretador Python instalado na máquina.

Segue adiante um passo-a-passo para executar o sistema.

1. Inicie o interpretador
2. Navegue o interpretador até diretório em que se encontra o código-fonte
3. Execute o interpretador

A interface é intuitiva; os menus são compostos de opções vinculadas a números, bastando digitar o valor numérico correspondente à opção que se deseja. Em caso de implementação de dados, digite os dados conforme o bairro informado. Caso a operação de implementação de dados não envolva todos os bairros, os novos dados são perdidos, e a operação precisa ser reiniciada. Em caso de busca, é preciso sempre se assegurar que sejam digitados os dados corretos, como datas e o código do bairro.

3.2. Testes, entrada e saída

A verificação das operações do sistema é fundamental para certificar seu bom funcionamento. Para realizar tal validação, toma-se dados amostrais iniciais que serão computados e validados.

Considera-se os elementos da tabela 1 como parte da matriz de dados. Para verificar a implementação de novos números, toma-se a tabela 2 como referência para a entrada de novos dados.

Data	Bairros	Habitantes	Casos suspeitos	Casos negativos	Casos positivos
23/03/2024	Tomba	55007	90	25	505
23/03/2024	Campo Limpo	47060	50	40	130
23/03/2024	Muchila	22496	43	12	70
23/03/2024	Conceição	21694	20	30	55

Tabela 1: Dados amostrais iniciais

Bairros	Novos casos suspeitos	Novos casos negativos	Novos casos positivos
Tomba	10	0	0
Campo Limpo	0	10	10
Muchila	5	3	0
Conceição	0	10	0

Tabela 2: Dados para teste de implementação

Ao executar o sistema e implementar os dados mostrados na tabela 2, o sistema deve ter como saída os dados da tabela 3.

Data	Bairros	Habitantes	Casos suspeitos	Casos negativos	Casos positivos
24/03/2024	Tomba	55007	100	25	505
24/03/2024	Campo Limpo	47060	30	50	140
24/03/2024	Muchila	22496	45	15	70
24/03/2024	Conceição	21694	10	40	55

Tabela 3: Dados finais após o processamento

Observa-se nos dados da tabela 3 a dependência dos novos dados em relação aos do dia anterior. Os números referentes aos bairros e sua população permanecem inalteráveis, pois considera-se uma população fixa, e as datas variam em um dia. Matematicamente, os dados para a nova data dependem pelas seguintes equações:

$$Cp_{t+1} = Cp_t + NCp \tag{1}$$

$$Cn_{t+1} = Cn_t + NCn \tag{2}$$

$$Cs_{t+1} = Cs_t + NCs - (NCp + NCn) \tag{3}$$

$$Cs_{t+1} + Cn_{t+1} + Cp_{t+1} \leq H \tag{4}$$

$$NCp + NCn \leq Cs_t \tag{5}$$

$t \in \mathbb{N}^*$ é o índice do dia[4],
H é a quantidade de habitantes,
Cp_t é a quantidade de casos positivos no dia t,
Cn_t é a quantidade de casos negativos no dia t,
Cs_t é a quantidade de casos suspeitos no dia t,
NCp é a quantidade de novos casos positivos,
NCn é a quantidade de novos casos casos negativos,
NCs é a quantidade de novos casos casos suspeitos,
$Cp_t, Cn_t, Cs_t, NCp, NCn, NCs, H \in \mathbb{N}$

As fórmulas são generalizadas para qualquer amostra de população, seja por bairros ou pela cidade. Neste caso, como cada bairro é associado a um conjunto de dados de casos e habitantes, utiliza-se as equações para a população e tipos de caso por bairro.

Nota-se que, ao aplicar as equações 1, 2 e 3 para os elementos das tabelas 1 e 2, os resultados são congruentes com os valores da tabela 3. Observa-se também que todos os valores são naturais, portanto não admite valores negativos como resultado. As equações 4 e 5 garantem que os resultados sejam sempre números naturais.

[4]Note que $t + 1$ indica o dia seguinte ao dia t.

3.2.1. Percentualidade: entrada e saída

Os cálculos referentes às porcentagens são impressos na tela e são determinados por relações matemáticas percentuais. A seguir, essas equações são melhor descritas. No sistema, o usuário tem a opção de buscar dados e, a depender do tipo de busca, há um retorno percentual diferente. Como mencionado anteriormente, há três tipos de busca, e os resultados de seus cálculos são descritos a seguir.

Considera-se, primeiramente, a busca por bairro e data. Este tipo de busca calcula a percentualidade de casos suspeitos em relação ao total de habitantes.

$$PCs = \frac{Cs_t}{H} \times 100 \tag{6}$$

$$PCp = \frac{Cp_t}{H} \times 100 \tag{7}$$

PCs é o percentual de casos suspeitos,
PCp é o percentual de casos positivos

Para exemplo, supõe-se os dados do bairro Tomba descritos na tabela 1. Ao aplicar os dados nas equações 6 e 7, obtém-se os resultados aproximados mostrados na tabela 4 como saídas dadas pelo sistema.

Data	Bairros	Habitantes	Casos suspeitos (%)	Casos positivos (%)
23/03/2024	Tomba	55007	0.16%	0.92%

Tabela 4: Dados percentuais do bairro Tomba em 23 de Março de 2024

Para a busca por data específica. Os percentuais são relativos às parcelas de cada tipo de caso e o total de casos notificados. A equação 8 descreve matematicamente o cálculo desse tipo de percentual.

$$PC_t = \frac{C_t}{Cs_t + Cn_t + Cp_t} \times 100 \tag{8}$$

PC é o percentual do tipo de caso no dia t,
$C_t \in \{Cs_t, Cn_t, Cp_t\}$

Para ilustrar melhor, considere os dados da tabela 5.

Data	Habitantes	Casos suspeitos	Casos negativos	Casos positivos
22/03/2024	419233	1256	908	1524

Tabela 5: Dados totais referentes à data 22 de Março de 2024

Os valores mostrados na tabela 6 são os valores a serem retornados pelo sistema e são resultados aproximados dos valores resultantes da equação 8.

Data	Habitantes	Casos suspeitos (%)	Casos negativos (%)	Casos positivos (%)
22/03/2024	419233	34.06%	24.62%	41.32%

Tabela 6: Dados percentuais referentes à data 22 de Março de 2024

Para buscas para intervalo de tempo, a percentualidade é dada pela variação dos casos positivos e negativos entre duas datas. Suponho duas datas t_i e t_f, tal que são inicial e final, respectivamente, considera-se os dados das tabelas 5 e 7.

Data	Habitantes	Casos suspeitos	Casos negativos	Casos positivos
23/03/2024	419233	1384	1128	2772

Tabela 7: Dados totais referentes à data 23 de Março de 2024

$$\Delta P_p = \frac{Cp_{t_f} - Cp_{t_i}}{Cp_{t_i}} \times 100 \tag{9}$$

$$\Delta P_n = \frac{Cn_{t_f} - Cn_{t_i}}{Cn_{t_i}} \times 100 \tag{10}$$

ΔP_p indica a variação percentual de casos positivos entre as datas,
ΔP_n indica a variação percentual de casos negativos entre as datas

Variação de casos negativos (%)	Variação de positivos (%)
24.23%	81.89%

Tabela 8: Percentualidade de variação de casos positivos e negativos entre as datas 22 e 23 de Março de 2024

As equações 9 e 10 descrevem variações de casos positivos e negativos totais, respectivamente, entre as datas final e inicial. Como os números desses tipos de caso são sempre acrescidos ao longo do tempo, não se tem variação negativa. Observa-se na tabela 8 os percentuais retornados pelo sistema.

Por fim, todas as equações aqui apresentadas são aplicadas pelo sistema e representam todo modelo matemático utilizado para computação de estatísticas pertinentes ao problema, além de recurso aritmético para garantir a correta entrada de dados.

3.3. Erros

Os erros encontrados envolvem o arquivo *csv* e a formatação das datas. Caso o arquivo de entrada não tiver, no mínimo, 25 bairros após o cabeçalho; a operação de modificação pode atestar dois tipos de erro: *IndexError* que indica a presença de índices fora do intervalo válido, e o *ValueError* que indica a tentativa de converter *strings*, que não representa nesse caso algum número em base decimal, em um dado do tipo *inteiro*. Além disso, pode ocorrer o *FileNotFoundError* que indica que o arquivo *csv* está fora do mesmo diretório que o código-fonte ou tem nome diferente. Por isso é necessário padronizar o arquivo em termos de nome e estrutura de seus dados. Em relação às datas, o erro frequente é o *ValueError* que indica a digitação incorreta da data, não seguindo o padrão *dd/mm/aaaa*.

4. Conclusão

Este relatório apresentou a solução computacional de um sistema de controle de dengue em linguagem interpretativa Python. A solução resolve o problema proposto, fazendo uso de várias estruturas de dados, como matrizes, listas e dicionários, manipulando-as a fim de computar os dados pertinentes ao problema; o uso de funções para executar tarefas específicas e deixar o código-fonte melhor estruturado; e, por fim, ferramentas de manipulação de arquivos do tipo *csv*. A explicação do desenvolvimento do sistema neste relatório centrou-se em explicar o funcionamento e a implementação das estruturas de dados e funções a fim de resolver o problema. A solução é fundamentada em generalizar ou especificar tarefas de modo estruturado e organizado, integrando-as em partes harmônicas entre si.

O sistema apresenta erros, já mencionados, que derivam da falta de padronização do arquivo *csv*, portanto, servindo somente para o formato específico desse arquivo, que deve estar contido no mesmo diretório que o código-fonte. Além disso, outro problema é o não cumprimento do uso correto da formatação de data, que pode levar a erros durante a execução do sistema. Versões futuras podem tratar melhor esses erros e tornar o código mais generalista a fim de torná-lo mais flexível e aplicável em situações relacionadas ao controle de doenças.

Referências

Brasil (2024). Dengue. `https://www.gov.br/saude/pt-br/assuntos/saude-de-a-a-z/d/dengue`. Acesso em: 24 de Maio de 2024.

IPM (2024). Vigilância em saúde. `https://www.ipm.com.br/blog/vigilancia/vigilancia-em-saude-veja-como-a-tecnologia-pode-facilitar-a-gestao-no-seu-municipio/`. Acesso em: 24 de Maio de 2024.

Menezes, N. N. C. (2016). *Introdução à programação com Python*. Novatec.

Python Software Foundation (2024). Python documentation. `https://docs.python.org/3.12/`. Acesso em: 31 de Maio de 2024.

Secretária de Saúde do Paraná (2024). Zoonoses. `https://www.saude.pr.gov.br/Pagina/Zoonoses`. Acesso em: 24 de Maio de 2024.

Silva, D. N. (2024). Peste negra. `https://brasilescola.uol.com.br/historiag/pandemia-de-peste-negra-seculo-xiv.htm`. Acesso em: 24 de Maio de 2024.