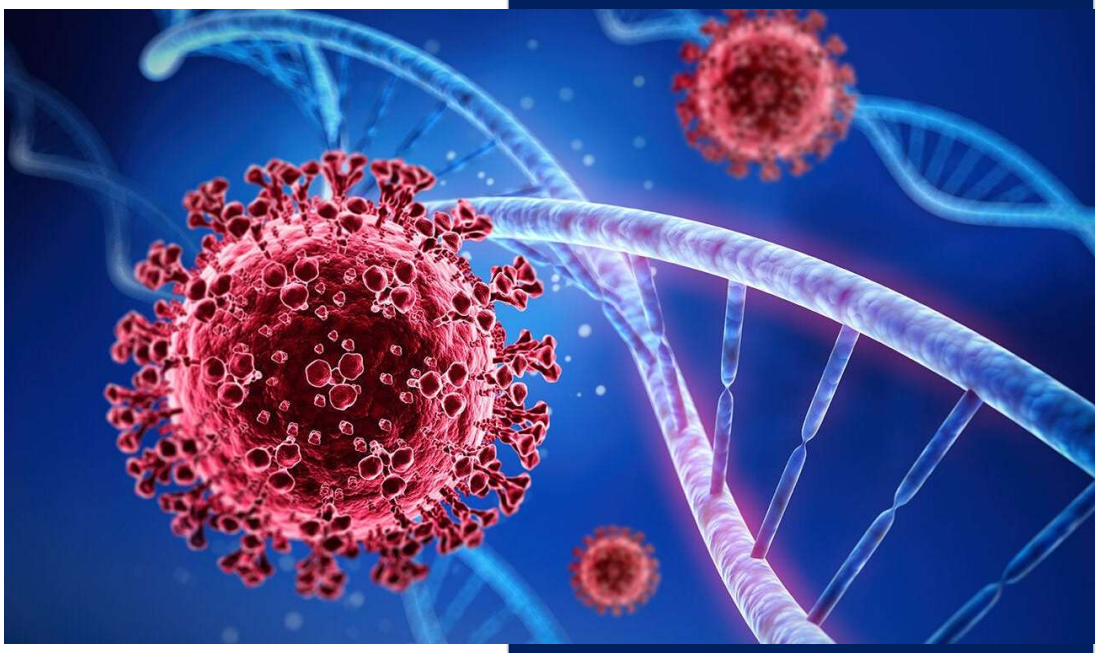
DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Relatório do Projeto de LAPR1



15/01/2022

Turma A/B Equipa 02, JRaid Technologies

1211343, Afonso Costa

1200967, Diogo Araújo

1210814, Inês Costa

1211359, José Barbosa

1211345, Rodrigo Peireso

Carlos Ferreira (cgf)
Ana Barata (abt)
Cláudia Lima (cal)
Sandra Luna (slu)
Jorge Duarte (fjd)
Stella Abreu (sau)

isep Instituto Superior de Engenharia do Porto

ÍNDICE

ÍNDICE DE IMAGENS

EQUAÇÕES

FIGURAS

TABELAS

1. Introdução

Este projeto, desenvolvido no contexto da Unidade Curricular de LAPR1, foi concebido e desenvolvido de acordo com os requisitos apresentados: desenvolvimento de uma aplicação em linguagem Java (Horstmann,2015) para a instituição Monitorização da Saúde em Portugal (MSP) [1] que permita estudar e prever a evolução da pandemia de COVID-19.

O presente relatório tem como objetivo descrever a aplicação concebida, assim como o processo de desenvolvimento e os resultados obtidos.

Neste documento serão apresentados a metodologia de trabalho utilizada, a análise das cadeias de Markov e decomposição LU pelo método de Crout, o desenvolvimento e implementação da aplicação e o caso de estudo.

2. Metodologia de Trabalho

Neste ponto será descrita de forma pertinente e detalhada a metodologia de trabalho utilizada no desenvolvimento do projeto proposto, passando por pontos essenciais para a compreensão da mesma.

2.1 Scrum no desenvolvimento do Projeto

A metodologia Scrum acompanhou o desenvolvimento do projeto a nível organizacional e de implementação, sendo isto atingido utilizando *frameworks* simples, intercalando ciclos curtos de atividade e os respetivos planeamentos prévios (responsáveis pela ponderação do trabalho a realizar no próximo Scrum Sprint). Esta metodologia ágil permitiu a integração do grupo, num ambiente onde existia espaço para incerteza e criatividade o que potenciou a aprendizagem da equipa. Assim sendo, esta metodologia pautada pelo seu grau de flexibilidade e adaptabilidade, possibilitou à equipa desenvolver um produto com qualidade num espaço de tempo reduzido.

Parte da aplicação da metodologia descrita, consiste no conceito de Scrum Master. Esta seria uma posição estática ou rotativa adotada por um elemento da equipa de desenvolvimento que atua como facilitador e garante a qualidade do produto, assim como o cumprimento dos prazos de entrega definidos. No desenvolvimento deste projeto o Scrum Master contribuiu para realizar uma revisão atenta das tarefas previamente atribuídas, tomando em especial atenção as datas-limite patentes nos cartões do Trello, estimadas pelo próprio desenvolvedor da tarefa tornando as mesmas mais justas e realistas. Este certificou-se também da conclusão das tarefas, tentando manter a qualidade do projeto como um todo, manter um registo no "Diário de Bordo" consistente e gradual acompanhando o trabalho desenvolvido pelos vários elementos e mediar discussões intraequipa sobre qualquer temática relevante ao projeto.

2.2 Planeamento no desenvolvimento do projeto

No âmbito do planeamento utilizou-se a ferramenta Trello (referida anteriormente), uma aplicação de gestão de projetos que de forma conveniente e eficiente integrou as tomadas de decisão ao longo do projeto, nos vários momentos de implementação e reuniões de planeamento. Relativamente às reuniões de planeamento, foram parte integral do projeto, ajudando no alinhamento dos processos simultâneos de desenvolvimento, reduzindo possíveis impedimentos e aumentando a produtividade da equipa nos vários ciclos de implementação. Na seguinte tabela encontram-se discriminadas todas as ferramentas de desenvolvimento que foram utilizadas:

Ferramenta	Objetivo	Utilização	Vantagens	Desvantagens
Bitbucket & Sourcetree	Repositório descentralizado, que permite acesso e alteração de ficheiros por vários utilizadores em simultâneo associado à *GUI (Sourcetree)*	Partilha de informação e atualizações do desenvolvimento do website	Facilita o desenvolvimento do projeto em simultâneo pelos vários membros	Interface confusa numa primeira utilização
Microsoft Teams	Comunicação e partilha de informação entre os membros da equipa	Partilha de documentos e associação de outras plataformas (*Bitbucket, Trello*)	Partilha rápida e eficaz de informação e acesso facilitado a outras plataformas	Interface confusa para iniciantes
Moodle	Sistema de Gestão de Aprendizagem	Consulta de documentos e informações sobre o projeto	Interface simples, secções organizadas e fácil acesso a ficheiros e ferramentas	Ausência de notificações sobre atualizações de conteúdo
Discord	Plataforma de comunicação	Reuniões, partilha de informação e distribuição de tarefas	Fácil acesso e utilização Permite a partilha de ecrã e de informação	Falhas constantes de ligação com os servers do *Discord*
IntelliJ	Editor de código-fonte	Edição do código necessário para a elaboração do website	Permite escrever e editar o código de uma forma mais acessível	Avisos de erros pouco explícitos para principiantes
Notepad++	Editor de texto e de código fonte	Editar segmentos curtos de código	Mais expandidas relativamente ao *Notepad*	Apenas útil em pequenas alterações no código devido à sua simplicidade
Mendeley	Gestão de referências	Automatização da criação de referências	Facilita o uso, criação e organização de referências	Quantidade de passos para configurar
Trello	Gestão e Organização de Projetos	Gestão e Organização das tarefas do projeto	Facilita o uso, criação e organização das tarefas da equipa	Ausência de modelos predefinidos

Tabela 1 - Ferramentas Utilizadas

2.3 Reflexão crítica sobre a dinâmica da equipa

A equipa revelou-se preparada ao longo das semanas de iteração, melhorando consideravelmente a comunicação e fazendo uma gestão e organização do trabalho a realizar bem estruturadas e eficientes. Todos os elementos apresentaram níveis de empenho, esforço e assiduidade consideráveis, mantendo a responsabilidade e o trabalho como pontos de importância integral, alcançando assim, um cumprimento constante de todas as datas-limite. O grupo adotou uma estratégia de cooperação entre os vários constituintes do mesmo, colmatando os possíveis problemas de divisões de trabalho desiguais e problemas comuns no desenvolvimento de projetos como a diferença de ritmos de trabalho derivada pela diferença de competências técnicas e a existência de dependências entre as várias funcionalidades do programa.

Esta estratégia consistiu numa mobilização dos elementos para outras tarefas no momento de conclusão das tarefas inicialmente atribuídas aos mesmos, passando estes a ter a função de reforço e suporte a uma outra parte do projeto. Assim sendo, a equipa apoiando-se numa metodologia simples e dinâmica conseguiu desenvolver de forma competente um produto robusto e interativo.

3. Cadeias de Markov e decomposição LU pelo método de Crout

3.1. Cadeias de Markov

Uma cadeia de Markov é um processo estocástico (*Stochastic Processes | Brilliant Math & Science Wiki*, n.d.) (processo no qual valores mudam aleatoriamente ao longo do tempo) que descreve transições entre vários estados de acordo com certas regras probabilísticas (*Markov Chains | Brilliant Math & Science Wiki*, n.d.). O processo de Markov diferencia-se dos demais processos estocásticos por obedecer à propriedade de Markov (*Markov Property - Wikipedia*, n.d.), que enuncia que a probabilidade de um sistema estar num estado i num passo t+1, apenas depende do estado em que estava no passo t:

$$P(X_{t+1} = x|X_0, X_1, X_3, \dots, X_n) = P(X_{t+1} = x|X_n) \text{ em que } x_i \geq 0, \Sigma_i x_i = 1$$

Equação 1

As cadeias de Markov (*Markov Property - Wikipedia*, n.d.) são utilizadas para modelar e descrever fenómenos que aparentam variar de forma aleatória. Exemplos disto são a movimentação de partículas, jogos de tabuleiro que utilizam dados, evolução de uma população de microrganismos e evolução de pandemias. Este último exemplo coincide com o objetivo deste projeto que é estudar e fazer previsões sobre a pandemia COVID-19 em Portugal.

Um processo de Markov está associado a uma matriz de transição de probabilidades (ou matriz de Markov) e pode ser representado por um grafo denominado diagrama de estados.

3.2. Matriz de Transição

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & a_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{p_{n1}} & \mathrm{p_{n2}} & \cdots & \mathrm{p_{nn}} \end{bmatrix}$$

Equação 2

A matriz de transição P é a representação matricial das probabilidades de transição. É uma matriz quadrada de dimensões $n \times n$ em que n é o número de estados do sistema.

Cada entrada p_{ij} da matriz corresponde á probabilidade de um membro da população transitar do estado j para o estado i e tem-se que $0 \leq p_{ij} \leq 1$. Caso a probabilidade seja $p_{ij} = 0$, o membro da população é incapaz de voltar ao estado i partindo do estado j. Uma probabilidade de $p_{ij} = 1$ significa que o membro passar do estado j para o estado i no próximo passo é um acontecimento certo.

A cada passo, o membro tem de permanecer no mesmo estado ou transitar para outro estado. Isto implica que a soma das entradas em cada coluna da matriz é igual a 1. Por exemplo, para a coluna n tem-se:

$$p_{1n} + p_{2n} + \cdots + p_{nn} = 1$$

Equação 3

3.3. Aplicações e exemplo

Como exposto em cima, os processos de Markov podem modelar diversos fenómenos que aparentam variar de forma aleatória. A título explicativo, apresenta-se de seguida um exemplo de um fenómeno modelado por uma cadeia de Markov.

Modelo de Previsão Meteorológica:

Considere-se um modelo para a previsão do tempo numa cidade. Neste modelo considera-se que num dia pode estar sol, chover ou nevar.

Se num dia faz sol, no dia seguinte chove 60% das vezes, neva 20% das vezes e faz sol novamente 30% das vezes.

Se num dia chover, no dia seguinte neva 15% das vezes, faz sol 30% das vezes e chove novamente 20% das vezes.

Se num dia neva, no dia seguinte chove 20% das vezes, faz sol 40% das vezes e neva novamente 65% das vezes.

Na figura Figura 1 apresenta-se o Diagrama de transição de estados correspondente a esta cadeia de Markov.

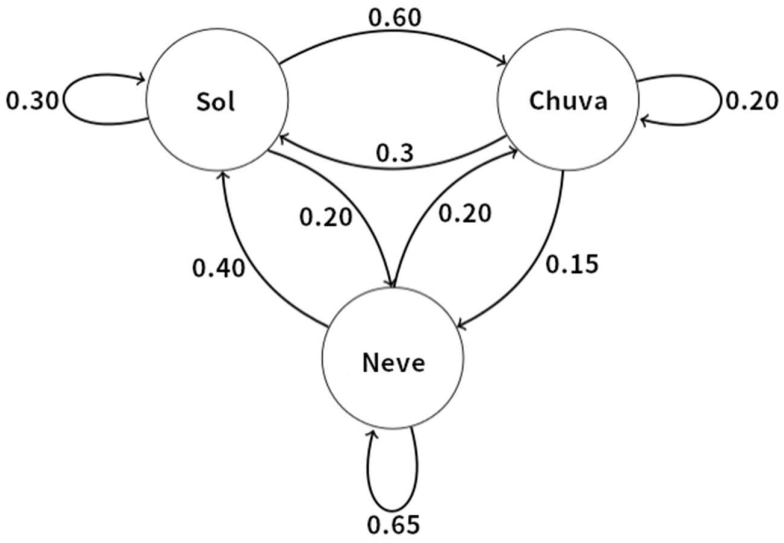


Figura 1 - Diagrama de Transição

A este modelo está associada a matriz de transições P representada na Equação 4.

$$P = \begin{bmatrix} 0.30 & 0.60 & 0.20 \\ 0.30 & 0.20 & 0.15 \\ 0.40 & 0.20 & 0.65 \end{bmatrix}$$

Equação 4

O estado de um sistema pode ser representado por um vetor coluna X_{n1} em que n é o número de estados do sistema em questão. Para este exemplo, a matriz de estado encontra-se na Equação 5.

$$X = \begin{bmatrix} x_{Sol} \\ x_{Chuva} \\ x_{Neve} \end{bmatrix}$$

Equação 5

Sabendo a matriz de estado X_t no dia t, obtém-se a matriz de estado no dia $t+1$, multiplicando-se a matriz de transição pela matriz de estado em t (Equação 6).

$$X_{t+1} = PX_t$$

Equação 6

Para $t+k$, basta elevar a matriz de transições P a k e multiplicá-la pela matriz de estado X_t.

$$X_{t+k} = P^k X_t$$

Equação 7

Isto ocorre como resultado da propriedade de Markov, pois X_{t+k} apenas está dependente de X_t (estado inicial) e k (número de passos).

Considerando agora o exemplo em questão, assuma-se que no dia t neva:

$$X_t = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Equação 8

Para $t+3$, a matriz de estado será:

$$X_{t+3} = \begin{bmatrix} 0.30 & 0.60 & 0.20 \\ 0.30 & 0.20 & 0.15 \\ 0.40 & 0.20 & 0.65 \end{bmatrix} \begin{bmatrix} 0.30 & 0.60 & 0.20 \\ 0.30 & 0.20 & 0.15 \\ 0.40 & 0.20 & 0.65 \end{bmatrix} \begin{bmatrix} 0.30 & 0.60 & 0.20 \\ 0.30 & 0.20 & 0.15 \\ 0.40 & 0.20 & 0.65 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.303 \\ 0.201 \\ 0.496 \end{bmatrix}$$

Equação 9

Cada uma das entradas da matriz resultante irá corresponder à probabilidade de no dia $t+3$: fazer sol (x_{11}), chover (x_{21}) e nevar (x_{31}).

3.4. Decomposição LU

No âmbito deste projeto realizam-se previsões sobre a pandemia COVID-19 em Portugal. Este processo é descrito por uma cadeia de Markov absorvente.

A cadeia de Markov que modela este processo contém 4 estados transientes (não infetado / infetado / hospitalizado / internado em UCI) e 1 estado absorvente (óbito). Um estado transiente é um estado do qual é possível transitar para outros estados, sendo um estado absorvente um estado do qual não é possível regressar/transitar para qualquer outro estado.

Numa cadeia de Markov absorvente (*Absorbing Markov Chain - Wikipedia*, n.d.; Yashinski, 2021), o número esperado de visitas a um estado transiente j começando no estado i (antes de ser absorvido) é dado pela entrada ij da matriz N (Equação 10)

$$N = \sum_{k=0}^{\infty} Q^k = (I - Q)^{-1}$$

Equação 10

Em que a probabilidade de transitar do estado i para o estado j em exatamente k passos é dada pela entrada ij de Q^k, sendo Q a matriz de transição e I a matriz identidade com as mesmas dimensões que Q.

Neste projeto é de especial relevância determinar o número de dias até chegar ao estado óbito. Com o intuito de facilitar este cálculo eliminaram-se a coluna e a linha relativas a este estado da matriz de transição fornecida, obtendo-se uma matriz de transições sem estado absorvente com dimensões 4×4, passando esta a ser a nova matriz Q.

Aplicando a fórmula na Equação 11, calcula-se o número esperado de dias até ao estado óbito começando num estado transiente i utilizando a fórmula:

$$M = 1 \times N$$

Equação 11

onde 1 é um vetor linha de comprimento igual ao número de estados transientes e cujas entradas tomam todas o valor 1. O número esperado de dias até ao óbito começando no estado i é dado pela entrada do i vetor M.

3.4.1. Método de Crout vs. método de Gauss-Jordan

Com o objetivo de facilitar o cálculo da matriz inversa de $(I - Q)$, implementou-se a decomposição LU recorrendo ao método de Crout. A decomposição $A = LU$ foi escolhida como alternativa á eliminação de Gauss-Jordan devido à sua menor complexidade computacional.

3.4.2. Implementação do método de Crout

Para calcular a inversa da matriz quadrada $A = (I - Q)$ utilizando o método de Crout (*MATRIX INVERSION AND EIGEN VALUE PROBLEMS*, n.d.), começa por se fazer a decomposição LU de A, ou seja, $A = LU$ onde L é uma matriz triangular inferior e U é uma matriz triangular superior cujos elementos na diagonal tomam o valor de 1.

Com isto, uma vez que a multiplicação da inversa de duas matrizes é igual à inversa da multiplicação dessas matrizes, tem-se que:

$$A^{-1} = (LU)^{-1} = U^{-1} L^{-1}$$

Equação 12

Esta propriedade permite achar a inversa de A de forma simples, uma vez que o cálculo de L^{-1} e de U^{-1} são de baixa complexidade devido a ambas as matrizes serem triangulares. Utilizando a igualdade $LL^{-1} = I$, pode encontrar-se a matriz L^{-1} que, tal como L, é triangular inferior. Da mesma forma, pode achar-se U^{-1} recorrendo à igualdade $UU^{-1} = I$. A matriz U^{-1} é também uma matriz triangular superior com diagonal 1 como U.

Após o cálculo de U^{-1} e L^{-1}, apenas resta multiplicar as duas matrizes para obter A^{-1}.

3.4.3. Decomposição de A em A=LU

3.4.3.1. Exploração teórica

Considere-se:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}, L = \begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \text{ e } U = \begin{bmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Então $A = LU$:

$$\begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

$$\begin{bmatrix} l_{11} & l_{11}u_{12} & l_{11}u_{13} & l_{11}u_{14} \\ l_{21} & l_{21}u_{12} + l_{22} & l_{21}u_{13} + l_{22}u_{23} & l_{21}u_{14} + l_{22}u_{24} \\ l_{31} & l_{31}u_{12} + l_{32} & l_{31}u_{13} + l_{32}u_{23} + l_{33} & l_{31}u_{14} + l_{32}u_{24} + l_{33}u_{34} \\ l_{41} & l_{41}u_{12} + l_{42} & l_{41}u_{13} + l_{42}u_{23} + l_{43} & l_{41}u_{14} + l_{42}u_{24} + l_{43}u_{34} + l_{44} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

Igualando coeficientes e simplificando tem-se:

linha 1: $l_{11} = a_{11}, u_{12} = \frac{a_{12}}{l_{11}}, u_{13} = \frac{a_{13}}{l_{11}}, u_{14} = \frac{a_{14}}{l_{11}}$

linha 2: $l_{21} = a_{21}, l_{22} = a_{22} - l_{21}u_{12}, u_{23} = \frac{a_{23} - (l_{21}u_{13})}{l_{22}}, u_{24} = \frac{a_{24} - (l_{21}u_{14})}{l_{22}}$

linha 3: $l_{31} = a_{41}, l_{32} = a_{32} - l_{31}u_{12}, l_{33} = a_{33} - (l_{31}u_{13} + l_{32}u_{23}), u_{34} = \frac{a_{34} - (l_{31}u_{14} + l_{32}u_{24})}{l_{33}}$

linha 4: $l_{41} = a_{41}, l_{42} = a_{42} - l_{41}u_{12}, l_{43} = a_{43} - (l_{41}u_{13} + l_{42}u_{23}),$

$$l_{44} = a_{44} - (l_{41}u_{14} + l_{42}u_{24} + l_{43}u_{34})$$

Igualando e simplificando os coeficientes constata-se que, seguindo a ordem de **cima para baixo (linha a linha)**, e da **esquerda para a direita (coluna a coluna)** para todas as entradas da matriz A, é sempre possível determinar o valor da entrada l_{ij} ou u_{ij} utilizando valores calculados anteriormente.

Com a análise das igualdades obtidas, deduz-se que as entradas de L são dadas por:

$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} (l_{ik} u_{kj})$$

Equação 13

e as entradas de U são dadas por:

$$u_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} (l_{ik} u_{kj})}{l_{ii}}$$

Equação 14

3.4.3.2. Implementação em Java

Uma vez que seguindo a ordem acima apontada é possível fazer a computação simultânea de L e U, criou-se o seguinte método (Figura 2 - Método de decomposição LU) que recebe a matriz A (*matrix*) por parâmetro, calcula as matrizes U (*upper*) e L (*lower*) paralelamente e retorna ambas numa matriz tridimensional cuja primeira entrada é ocupada pela matriz U e a segunda entrada é ocupada pela matriz L.

```
1911  @     public static double[][][] luDecomposition(double[][] matrix) {
1912            int size = matrix.length;
1913            double[][] upper = new double[size][size];
1914            double[][] lower = new double[size][size];
1915            for (int order = 0; order < size; order++) {
1916                //lower
1917                for (int lineLower = order; lineLower < size; lineLower++) {
1918                    double sum = 0;
1919                    for (int columnLower = 0; columnLower < order; columnLower++)
1920                        sum += lower[lineLower][columnLower] * upper[columnLower][order];
1921                    lower[lineLower][order] = matrix[lineLower][order] - sum;
1922                }
1923                //upper
1924                for (int columnUpper = order; columnUpper < size; columnUpper++) {
1925                    if (order == columnUpper)
1926                        upper[order][order] = 1;
1927                    else {
1928                        double sum = 0;
1929                        for (int lineUpper = 0; lineUpper < columnUpper; lineUpper++) {
1930                            sum += lower[order][lineUpper] * upper[lineUpper][columnUpper];
1931                        }
1932                        upper[order][columnUpper] = (matrix[order][columnUpper] - sum) / lower[order][order];
1933                    }
1934                }
1935            }
1936            return new double[][][]{upper, lower};
1937        }
```

Figura 2 - Método de decomposição LU

O ciclo na linha do código l. 1915 percorre cada linha da matriz inicial (de cima para baixo) pois os cálculos serão feitos linha a linha para evitar substituições posteriores.

São calculadas primeiro as entradas da matriz L pertencentes a essa linha da matriz utilizando o ciclo na l. 1917. Na linha selecionada pelo ciclo da l. 1917, o ciclo da l. 1919 seleciona a coluna da entrada a ser calculada e é calculado o somatório, apresentado na fórmula da Equação 13.

Após o cálculo do somatório, é efetuada a subtração do mesmo à entrada da matriz A na posição selecionada pelos ciclos das ll. 1915 e 1917, (Equação 13), obtendo-se assim o valor da matriz L na posição calculada.

No ciclo da l. 1924, são calculadas as entradas da matriz U pertencentes a essa linha. Caso a entrada se encontre na diagonal, é preenchida com o valor 1 (ll. 1925-1926).

Caso contrário, é calculado o somatório na Equação 14 relativo a essa entrada. Posteriormente a entrada de U é calculada subtraindo o somatório à entrada da mesma posição da matriz A e dividindo tudo pela entrada da matriz L situada na diagonal nessa linha.

Este processo é repetido para cada linha da matriz A até que todas as entradas de L e U se encontram preenchidas uma vez que são do mesmo tamanho que a matriz inicial.

O método retorna uma matriz de 3 dimensões que contém a matriz U na posição [0][][] e a matriz L na [1][][] do $array$ de retorno.

3.4.4. Cálculo da inversa de L

3.4.4.1. Exploração teórica

Como referido em 3.4.1, utilizando a igualdade $LL^{-1} = I$, pode encontrar-se a matriz L^{-1} que, tal como L, é triangular inferior.

Considere-se:

$$L = \begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix}, L^{-1} = \begin{bmatrix} x_{11} & 0 & 0 & 0 \\ x_{21} & x_{22} & 0 & 0 \\ x_{31} & x_{32} & x_{33} & 0 \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} \text{ e } I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Então $LL^{-1} = I$:

$$\begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} x_{11} & 0 & 0 & 0 \\ x_{21} & x_{22} & 0 & 0 \\ x_{31} & x_{32} & x_{33} & 0 \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

linha 1: $x_{11} = \frac{1}{l_{11}}$

linha 2:$x_{21} = \frac{-l_{21}x_{11}}{l_{22}}, x_{22} = \frac{1}{l_{22}}$

linha 3:$x_{31} = \frac{-(l_{31}x_{11}+l_{32}x_{21})}{l_{33}}, x_{32} = \frac{-(l_{31}x_{12}+l_{32}x_{22})}{l_{33}}, x_{33} = \frac{1}{l_{33}}$

linha 4:$x_{41} = \frac{-(l_{41}x_{11}+l_{42}x_{21}+l_{43}x_{31})}{l_{44}}$, $x_{42} = \frac{-(l_{41}x_{12}+l_{42}x_{22}+l_{43}x_{32})}{l_{44}}$, $x_{43} = \frac{-(l_{41}x_{13}+l_{42}x_{23}+l_{43}x_{33})}{l_{44}}$, $x_{44} = \frac{1}{l_{44}}$

Igualando e simplificando os coeficientes constata-se que, seguindo a ordem de **cima para baixo (linha a linha)**, e da **esquerda para a direita (coluna a coluna)** para todas as entradas da matriz L^{-1}, é sempre possível determinar o valor da entrada x_{ij} utilizando os valores já determinados.

Com a análise das igualdades obtidas, deduz-se que as entradas de L^{-1} são dadas por:

$$x_{ij} = \frac{-\sum_{k=1}^{i-1}(l_{ik}u_{kj})}{l_{ii}}$$

Equação 15

Caso a entrada se encontre na diagonal, as entradas são dadas por:

$$x_{ii} = \frac{1}{l_{ii}}$$

Equação 16

3.4.4.2. Implementação em Java

Utilizando o que foi descoberto no capítulo 3.4.3.1, criou-se o seguinte método (Figura 3 - Método de inversão de matriz L) que recebe a matriz L (*matrixLower*) por parâmetro e retorna L^{-1} (*invertedMatrix*).

```
1939  @    public static double[][] invertLower(double[][] matrixLower) {
1940           int size = matrixLower.length;
1941           double[][] invertedMatrix = new double[size][size];
1942
1943           for (int line = 0; line < size; line++) {
1944               for (int column = 0; column <= line; column++) {
1945                   if (line == column)
1946                       invertedMatrix[line][column] = 1 / matrixLower[line][column];
1947                   else {
1948                       double sum = 0;
1949                       for (int nextColumn = column; nextColumn < line; nextColumn++) {
1950                           sum += matrixLower[line][nextColumn] * invertedMatrix[nextColumn][column];
1951                       }
1952                       invertedMatrix[line][column] = -sum / matrixLower[line][line];
1953                   }
1954               }
1955           }
1956           return invertedMatrix;
1957       }
```

Figura 3 - Método de inversão de matriz L

O ciclo da l. 1943 percorre cada linha da matriz triangular inferior (cima para baixo) pois os cálculos serão feitos linha a linha para evitar utilizar valores ainda não calculados.

O ciclo da l. 1944 percorre cada entrada na linha selecionada no ciclo da l. 1943 da esquerda para a direita até à diagonal da matriz, uma vez que o resultado é uma matriz diagonal inferior.

A condição nas ll. 1945-1946 assegura que a diagonal da matriz a ser retornada é preenchida com o valor de $\frac{1}{l_{ii}}$, de acordo com a fórmula na Equação 16.

Em todos os outros casos (ll. 1947-1953), o ciclo da l. 1949 calcula o somatório necessário para o cálculo da entrada selecionada, como apresentado na fórmula da Equação 15.

Após o cálculo do somatório, o valor da entrada selecionada da matriz a ser retornada é calculado dividindo o simétrico do somatório pela entrada l_{ii} (l. 1952).

Após serem percorridas todos as entradas a ser calculadas, a matriz L^{-1} é retornada.

3.4.5. Cálculo da inversa de U

3.4.5.1. Exploração teórica

Como referido em 3.4.1, utilizando a igualdade $UU^{-1} = I$, pode encontrar-se a matriz U^{-1} que, tal como U, é triangular inferior e tem a diagonal com valor 1.

Considere-se:

$$U = \begin{bmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}, U^{-1} = \begin{bmatrix} 1 & y_{12} & y_{13} & y_{14} \\ 0 & 1 & y_{23} & y_{24} \\ 0 & 0 & 1 & y_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ e } I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Então $UU^{-1} = I$:

$$\begin{bmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & y_{12} & y_{13} & y_{14} \\ 0 & 1 & y_{23} & y_{24} \\ 0 & 0 & 1 & y_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

linha 4: Sem cálculos a efetuar

linha 3: $y_{34} = -u_{34}$

linha 2: $y_{24} = -(u_{23}y_{34} + u_{24}), y_{23} = -u_{23}$

linha 1: $y_{14} = -(u_{12}y_{24} + u_{13}y_{34} + u_{14}), y_{13} = -(u_{12}y_{23} + u_{13}), y_{12} = -u_{12}$

Igualando e simplificando os coeficientes constata-se que, seguindo a ordem de **baixo para cima (linha a linha)** e da **direita para a esquerda (coluna a coluna)** para todas as entradas da matriz U^{-1} é possível determinar o valor da entrada y_{ij} utilizando os valores já conhecidos.

Com a análise das igualdades obtidas, conclui-se que as entradas de U^{-1} são dadas por:

$$y_{ij} = -\sum_{k=1}^{j} (u_{ik}y_{kj})$$

Equação 17

3.4.5.2. Implementação em Java

Utilizando o que foi descoberto no capítulo 3.4.4.1, criou-se o seguinte método (Figura 4 - Método de inversão da matriz U) que recebe a matriz U (*matrixUpper*) por parâmetro e retorna U^{-1} (*invertedMatrix*).

```
1959  @    public static double[][] invertUpper(double[][] matrixUpper) {
1960           int size = matrixUpper.length;
1961           double[][] invertedMatrix = new double[size][size];
1962
1963           for (int column = size - 1; column >= 0; column--) {
1964               for (int line = column; line >= 0; line--) {
1965                   if (line == column)
1966                       invertedMatrix[line][column] = 1;
1967                   else {
1968                       double sum = 0;
1969                       for (int nextLine = line; nextLine <= column; nextLine++) {
1970                           sum += matrixUpper[line][nextLine] * invertedMatrix[nextLine][column];
1971                       }
1972                       invertedMatrix[line][column] = -sum;
1973                   }
1974               }
1975           }
1976           return invertedMatrix;
1977       }
```

Figura 4 - Método de inversão da matriz U

O ciclo da l. 1963 percorre cada linha da matriz triangular inferior (baixo para cima) pois os cálculos serão feitos linha a linha para evitar substituições posteriores.

O ciclo da l. 1964 assegura que as entradas são calculadas da direita para a esquerda na linha selecionada até a diagonal, uma vez que o resultado é uma matriz diagonal superior.

A condição da l. 1965 assegura que as entradas da diagonal da matriz a ser retornada tomam o valor 1.

Caso isso não se verifique (ll. 1967-1973), o ciclo da l. 1969 calcula o somatório (Equação 17) que será utilizado no cálculo da entrada selecionada pelos dois ciclos anteriores.

Após o cálculo do somatório, o simétrico do resultado do somatório é guardado na entrada de U^{-1} que estava a ser calculada (l. 1972).

Após serem percorridas todos as entradas a ser calculadas, a matriz U^{-1} é retornada.

3.5. Conclusão - Utilização do Resultado da Decomposição LU

Como foi exposto em 3.3.1, a decomposição LU foi feita com o propósito de facilitar o cálculo da inversa de $(I - Q)$.

Uma vez que se conseguiu algoritmizar o cálculo de L^{-1} e U^{-1}, basta agora efetuar a subtração de matrizes $I - Q$, e calcular a inversa do resultado utilizando os 3 métodos apresentados anteriormente para a aplicação do método de Crout.

Para calcular a matriz M (Equação 11), utiliza-se o método *multiplyMatrices* para realizar a multiplicação e consegue-se finalmente obter o número previsto de dias até o indivíduo atingir o estado de óbito.

A realização de previsões será explorada mais a fundo no capítulo 5.

4. Desenvolvimento e Implementação da Aplicação

A linguagem de programação utilizada no desenvolvimento deste projeto foi java, com o intuito de aplicar os conhecimentos consolidados na unidade curricular de APROG.

4.1 Estrutura da Aplicação

No decorrer do nosso projeto, recorremos à modularização, tendo criado assim 83 métodos todos interligados entre si. Foram criados métodos cuja função é calcular resultados e outros imprimir esses mesmos resultados. Para além disso, recorremos também à modularização para ler dados e validar os mesmos.

O esquema do anexo A apresenta de forma organizada as dependências de cada método. Encontra-se subdividido em 2 partes: os métodos de cálculos e os métodos referentes à parte do menu (leitura, validação e impressão). Por conseguinte, cada método dentro de cada categoria encontra-se ordenado, juntamente com os métodos seus dependentes, por ordem de aparecimento no programa.

O programa apresenta uma vertente interativa e uma não interativa.

4.1.1 Modo Interativo

O modo interativo da aplicação é executado quando não é inserido nenhum argumento na linha de comandos. Neste modo todos os parâmetros necessários são solicitados em tempo de execução ao utilizador, existindo um menu de seleção de ficheiros e um menu com todas as funcionalidades da aplicação.

```
boolean returnToMainMenu = false;
while (!returnToMainMenu) {

    System.out.println("\n-----O que deseja visualizar?-----");
    System.out.println("\nOpção 1: Variações");
    System.out.println("Opção 2: Comparações");
    System.out.println("Opção 3: Total");
    System.out.println("Opção 4: Previsões");
    System.out.println("Opção 0: Retornar ao Menu Principal");
    System.out.println("Selecione o número da opção que deseja:");

    String visualizationOption = in.next();


    switch (visualizationOption) {
        case "1":
            if (!checkIfNull(dateArrayCumulative)) {...} else {...}

            break;
        case "2"://medio e desvio
            if (!checkIfNull(dateArrayCumulative) || !checkIfNull(dateArrayTotal)) {...} else {...}
            break;
        case "3":
            if (!checkIfNull(dateArrayTotal)) {...} else {...}
            break;
        case "4":
            if (!checkIfNull(dateArrayTotal) && !checkIfNull(matrixData)) {...}else{...}
            break;
        case "0":
            returnToMainMenu = true;
            System.out.println("-----Retornando ao menu principal-----");
            break;

        default:
            System.out.println("Ocorreu um erro. A opção selecionada é inválida!");
            break;
    }
}
```

Figura 5 – Estrutura dos Menus e Submenus

4.1.2 Modo Não Interativo

No modo não interativo, o utilizador, como o nome indica, não tem qualquer interação com o programa. Desta forma, pretende-se que todas as funcionalidades da aplicação sejam executadas por um único comando (desde que sejam introduzidos todos os parâmetros requeridos), que possua a seguinte estrutura: java -jar ProjetoLAPR1.jar -r X -di DD-MM-AAAA -df DD-MM-AAAA -di1 DD-MM-AAAA -df1 DD-MM-AAAA -di2 DD-MM-AAAA -df2 DD-MM-AAAA -T DD-MM-AAAA registoNumeroTotalCovid19.csv registoNumerosAcumuladosCovid19.txt matrizTransicao.txt nomeFicheiroSaida.txt.

Caso o utilizador não introduza os parâmetros corretamente, ser-lhe-á apresentada uma mensagem de erro de forma a guiá-lo à introdução correta dos dados.

```
if (args.length > 0) {
    notInteractiveMode(args, dateArrayCumulative, dataArrayCumulative, dateArrayTotal,dataArrayTotal, matrixData);
} else { //Menu Interativo
    while (!closeApp) {...}
}
```

Figura 6 - Condição de Execução dos Modos

4.2 Estrutura da Chamada dos Métodos de Cálculo

Em ambos os modos, depois de recebido o input do utilizador, os dados serão validados e tratados pelos métodos de cálculo. Após isto, serão chamados os respetivos métodos de impressão.

4.2.1 Validações dos Dados

Após a leitura dos dados introduzidos pelo utilizador, a aplicação verifica se as datas introduzidas se encontram no formato certo através do método *isValidDate*. De seguida, é verificado pelo método *dateRangeIsInArray* se as datas se encontram no ficheiro de dados fornecido.

```
public static boolean dateRangeIsInArray(Date[] dateArray, Date initialDate, Date finalDate, int arrayLength) {
    boolean isInitialDate = false, isFinalDate = false;

    for (int index = 0; index < arrayLength; index++) {
        if (initialDate.equals(dateArray[index]))
            isInitialDate = true;
        if (finalDate.equals(dateArray[index]))
            isFinalDate = true;
    }
    return isInitialDate && isFinalDate;
}
```

Figura 7 - Método dateRangeIsInArray

Dependendo do tipo de cálculo a ser efetuado, podem ser efetuadas verificações como por exemplo:

Se o cálculo é possível com as datas fornecidas, no caso de os dados acumulados se a data introduzida for a primeira do ficheiro, no caso das comparações, se os intervalos têm o mesmo tamanho e se não efetuada a sua correção, entre outras.

4.2.2 Impressão dos Resultados

Após realização dos cálculos, os valores relevantes são retornados em estruturas de dados (*arrays)* pelos métodos de cálculos pertinentes à operação solicitada pelo utilizador. Este retorno é utilizado por métodos de impressão de dados que variam de acordo com a funcionalidade selecionada pelo utilizador, a resolução selecionada e o ficheiro de dados sobre o qual o utilizador está a realizar operações. Também existe a hipótese de guardar os dados apresentados na consola num ficheiro .csv. No caso do modo não interativo, os dados que seriam apresentados na consola seriam armazenados num ficheiro .txt.

```
public static void printResult(int[] variationDataArrayInitial, int[] variationDataArrayFinal, int[] differenceVector, double average1, double average2, double sd1, double sd2, String[] daysInString, String[] daysInString2, boolean dayBefore) throws IOException {

    System.out.printf("\nComparação diária entre %s / %s e %s / %s\n\n", daysInString[0], daysInString[variationDataArrayInitial.length - 1], daysInString2[0], daysInString2[variationDataArrayFinal.length - 1]);
    System.out.printf("%14s |%14s |%10s |%10s |%10s\n", "Dia Período 1", "Dia Período 2", "Período 1", "Período 2", "Diferença");
    if (!dayBefore)
        System.out.printf("%14s |%14s | Erro: Dados insuficientes para calcular a variacao a %s.\n", daysInString[0], daysInString2[0], daysInString[0]);
    else
        System.out.printf("%14s |%14s |%10d |%10d |%10d\n", daysInString[0], daysInString2[0], variationDataArrayInitial[0], variationDataArrayFinal[0], differenceVector[0]);
    for (int i = 1; i < variationDataArrayInitial.length; i++) {
        System.out.printf("%14s |%14s |%10d |%10d |%10d\n", daysInString[i], daysInString2[i], variationDataArrayInitial[i], variationDataArrayFinal[i], differenceVector[i]);
    }
    System.out.printf("\nPeriodo %s a %s: Media: %.4f | Desvio Padrão: %.4f", daysInString[0], daysInString[variationDataArrayInitial.length - 1], average1, sd1);
    System.out.printf("\nPeriodo %s a %s: Media: %.4f | Desvio Padrão: %.4f\n", daysInString2[0], daysInString2[variationDataArrayFinal.length - 1], average2, sd2);
}
```

Figura 8 - Método printResult

4.3 Métodos de Cálculo

O nosso programa permite realizar diversas operações com os dados carregados.

4.3.1 Totais

A apresentação dos dados totais não requer o tratamento de dados através de métodos de cálculo.

```java
public static int getTotalDataArrayRequiredValue(int option, Date date, Date[] dateArray, int[][] dataArray, int arrayLength) {
    int result = 0;
    for (int i = 0; i < arrayLength; i++) {
        if (date.equals(dateArray[i])) {
            result = dataArray[i][option];
            break;
        }
    }
    return result;
}
```

Figura 9 - Método getTotalDataArrayRequiredValue

Este método procura a informação do tipo de dados pedido no dia introduzido pelo utilizador.

4.3.2 Variações

Existem vários métodos que permitem calcular variações utilizando os dados acumulados para o tipo de resolução pedida pelo utilizador (diária, semanal ou mensal).

```java
public static int[] weeklyFluctuation(int[][] dataArray, Date[] dateArray, Date initialDate, Date finalDate, int dataType, int arrayLength, boolean dayBefore) {
    Date[] rangeDate = findCompleteWeeks(initialDate, finalDate);
    int[] indexRange = calculateIndexRange(dateArray, rangeDate[0], rangeDate[1], arrayLength);
    int numberOfWeeks = numberOfWeeks(rangeDate);
    int[] weeklyFluctuation = new int[numberOfWeeks];
    if (!dayBefore)
        weeklyFluctuation[0] = 0;
    else
        weeklyFluctuation[0] = variation(dataArray, finalIndex: indexRange[0] + 6, initialIndex: indexRange[0] - 1, dataType);
    for (int week = 1; week < numberOfWeeks; week++)
        weeklyFluctuation[week] = variation(dataArray, finalIndex: indexRange[0] + 6 + (week) * 7, initialIndex: indexRange[0] - 1 + (week) * 7, dataType);
    return weeklyFluctuation;
}
```

Figura 10 - Método weeklyFluctuation

O método de variação semanal começa por procurar as semanas completas no intervalo introduzido pelo utilizador, sendo de seguida calculado o tipo de dados pretendido no novo intervalo de tempo. O mesmo se aplica ao método de variação diária e mensal.

4.3.3 Comparações

Estes métodos utilizam os valores das variações e dos totais para comparar 2 intervalos fornecidos pelo utilizador sendo também calculados a média e o desvio padrão.

4.3.4 Previsões

O nosso programa tem a capacidade de estimar o número total de membros da população num determinado estado, num determinado dia, assim como de calcular o número esperado de dias que um indivíduo num determinado estado demora até atingir o estado de óbito.

Isto é abordado de forma mais pormenorizada no capítulo 3.

5. Caso de Estudo

No estudo em causa, é pretendido analisar a evolução da pandemia de COVID-19 nos primeiros sete dias de janeiro de 2022, prevendo com o auxílio das funcionalidades da aplicação desenvolvida o número total de não infetados, o número total de infetados, o número total de hospitalizados, o número total de internados em Unidades de Cuidados Intensivos (UCI) e o número total de óbitos registados em cada dia. É pretendido também comparar os valores previstos com os totais reais que se verificaram nos respetivos dias, para assim ser possível avaliar o desempenho dos dois modelos de previsão utilizados (matrizes de transição A e B).

Para o cálculo dos valores de previsão de ambos os modelos, foi considerada a população inicial do último dia de dezembro de 2021, sendo que para cada modelo foi avaliado a sua qualidade de previsão, através do cálculo da soma do valor absoluto da diferença entre as previsões realizadas e os valores observados para cada tipo de dados em cada dia do intervalo a considerar, como se apresenta na seguinte fórmula:

$$Erro = \sum_{t=1}^{7} \left| x^t_{previsao} - x^t_{observados} \right|$$

Equação 18

De seguida, encontram-se as duas matrizes de transição utilizadas por cada modelo de previsão, assim como duas tabelas para cada modelo com os números obtidos em cada previsão, os números totais reais nessa data, a diferença entre o valor previsto pelo valor observado ($x^t_{previsao} - x^t_{observados}$), e o Erro a considerar para cada tipo de dados em cada dia, segundo a fórmula apresentada anteriormente.

$$A = \begin{bmatrix} 0.9995 & 0.03 & 0.002 & 0.001 & 0 \\ 0.0005 & 0.96 & 0.004 & 0.015 & 0 \\ 0 & 0.007 & 0.98 & 0.02 & 0 \\ 0 & 0.003 & 0.01 & 0.95 & 0 \\ 0 & 0 & 0.004 & 0.014 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 0.9995 & 0.049 & 0.005 & 0.001 & 0 \\ 0.0005 & 0.95 & 0.005 & 0.018 & 0 \\ 0 & 0.0009 & 0.98 & 0.02 & 0 \\ 0 & 0.0001 & 0.0085 & 0.947 & 0 \\ 0 & 0 & 0.0015 & 0.014 & 1 \end{bmatrix}$$

Equação 19 - Matrizes de transição A e B

Modelo A

	NÃO INFETADOS			INFETADOS		
Dia	Total	Previsão	Diferença	Total	Previsão	Diferença
01/01/2022	9951559	9969466	17907	196223	176554,3	-19668,7
02/01/2022	9944382	9969783,1	25401,1	203322	174496,2	-28825,8
03/01/2022	9939750	9970041,2	30291,2	207859	172533,1	-35325,9
04/01/2022	9933809	9970243,2	36434,2	213749	170660,9	-43088,1
05/01/2022	9908402	9970391,6	61989,6	239098	168875,4	-70222,6
06/01/2022	9899960	9970489,2	70529,2	247440	167172,8	-80267,2
07/01/2022	9893098	9970538,2	77440,2	254240	165549,4	-88690,6
Erro			**319992,5**			**366088,9**

Tabela 2

Modelo A

	HOSPITALIZADOS			INTERNADOS			ÓBITOS		
Dia	Total	Previsão	Diferença	Total	Previsão	Diferença	Total	Previsão	Diferença
01/01/2022	1023	2257,4	1234,4	142	684,1	542,1	21	24,1	3,1
02/01/2022	1081	3461,8	2380,8	148	1202,2	1054,2	14	42,7	28,7
03/01/2022	1167	4638,1	3471,1	147	1700,2	1553,2	10	73,4	63,4
04/01/2022	1203	5787,1	4584,1	147	2179,1	2032,1	15	115,8	100,8
05/01/2022	1251	6909,5	5658,5	143	2640	2497	14	169,4	155,4
06/01/2022	1311	8006,3	6695,3	158	3083,7	2925,7	25	234	209
07/01/2022	1353	9078	7725	161	3511,1	3350,1	17	309,2	292,2
Erro			**31749,2**			**13954,4**			**852,6**

Tabela 3

Modelo B

	NÃO INFETADOS			INFETADOS		
Dia	Total	Previsão	Diferença	Total	Previsão	Diferença
01/01/2022	9951559	9972864,6	21305,6	196223	174768,7	-21454,3
02/01/2022	9944382	9976447,8	32065,8	203322	171025,5	-32296,5
03/01/2022	9939750	9979846,6	40096,6	207859	167472,2	-40386,8
04/01/2022	9933809	9983070,2	49261,2	213749	164099,3	-49649,7
05/01/2022	9908402	9986127,5	77725,5	239098	160897,7	-78200,3
06/01/2022	9899960	9989027,1	89067,1	247440	157858,5	-89581,5
07/01/2022	9893098	9991776,9	98678,9	254240	154973,7	-99266,3
Erro			**408200,7**			**410835,4**

Tabela 4

Modelo B

	HOSPITALIZADOS			INTERNADOS			ÓBITOS		
Dia	Total	Previsão	Diferença	Total	Previsão	Diferença	Total	Previsão	Diferença
01/01/2022	1023	1167,3	144,3	142	163,9	21,9	21	21,6	0,6
02/01/2022	1081	1304,5	223,5	148	182,6	34,6	14	25,6	11,6
03/01/2022	1167	1436	269	147	201,1	54,1	10	30,1	20,1
04/01/2022	1203	1562	359	147	219,4	72,4	15	35,1	20,1
05/01/2022	1251	1682,8	431,8	143	237,5	94,5	14	40,5	26,5
06/01/2022	1311	1798,7	487,7	158	255,3	97,3	25	46,4	21,4
07/01/2022	1353	1909,9	556,9	161	272,8	111,8	17	52,6	35,6
Erro			**2472,2**			**486,6**			**135,9**

Tabela 5

5.1. Análise e discussão dos resultados

Depois de calculadas as previsões e quando comparadas com os valores reais do dia, é possível verificar que para qualquer tipo de dados (não infetados, infetados, hospitalizados, etc.) e para qualquer dos modelos (A e B), a diferença entre o número total real e o previsto vai sendo cada vez maior à medida que que o dia é mais afastado do dia inicial. É possível também verificar que as previsões de infetados são sempre inferiores aos valores reais observados, assim como os valores das previsões de não infetados, de hospitalizados, de internados em UCI e de óbitos são sempre superiores aos valores reais observados.

Em relação ao modelo A, verifica-se que o erro nos não infetados e nos infetados é cerca de 21,6% e 10,9% menor que os erros do modelo B, respetivamente. Em relação ao modelo B, o erro nos hospitalizados, internados e óbitos é consideravelmente menor que os do modelo A, sendo cerca de 11.8, 27.6 e 5.3 vezes menor, respetivamente.

Podemos assim concluir que ambos os modelos de previsão, embora nos primeiros dias os dados das previsões não sejam muito diferentes da realidade, quanto mais afastados do primeiro dia de consideração para os cálculos, mais afastados são os resultados das previsões dos valores reais.

É também possível verificar que o modelo A é mais eficaz nas previsões de infetados e não infetados, e o modelo B é mais nas restantes previsões.

6. Conclusão

A realização deste relatório teve como objetivo documentar o desenvolvimento da aplicação assim como todas as suas funcionalidades e como o utilizador interage com a mesma. Ao longo do projeto, a equipa mostrou empenho e assiduidade o que resultou no cumprimento de todos os prazos de submissão definidos pela mesma. Houve comunicação constante e entreajuda entre a equipa, o que facilitou o desenvolvimento do projeto e a aprendizagem dos seus elementos.

A aplicação desenvolvida tem em vista o estudo e a previsão da evolução da pandemia de COVID-19 de uma forma simples e intuitiva. Através da leitura de dados referentes à evolução pandémica, o programa executa diversas operações que permitem ao utilizador visualizar de uma forma clara as informações relativas a cada parâmetro que pretende analisar e permite fazer uma previsão do estado da pandemia, com base nas Cadeias de Markov e no Método de Crout.

A utilização de modularização e da metodologia Scrum foram fundamentais na elaboração do projeto.

Relativamente ao caso de estudo, verificaram-se discrepâncias entre os resultados previstos e obtidos, podendo concluir-se que ambos os modelos fornecidos não permitem efetuar previsões exatas sobre a evolução da pandemia.

Referências

Absorbing Markov chain - Wikipedia. (n.d.). Acedido em janeiro 21, 2022, de https://en.wikipedia.org/wiki/Absorbing_Markov_chain

Markov Chains | Brilliant Math & Science Wiki. (n.d.). Acedido em janeiro 21, 2022, de https://brilliant.org/wiki/markov-chains/

Markov property - Wikipedia. (n.d.). Acedido em janeiro 21, 2022, de https://en.wikipedia.org/wiki/Markov_property#Definition

MATRIX INVERSION AND EIGEN VALUE PROBLEMS. (n.d.).

Stochastic Processes | Brilliant Math & Science Wiki. (n.d.). Acedido em janeiro 21, 2022, de https://brilliant.org/wiki/stochastic-processes/

Yashinski, A. (2021). *Absorbing Markov Chains*.

ANEXOS

ANEXO A


Projeto02.java
Main
Menu
chooseNumberType
askDates
printResult
printResult
printResult
subMenu
readDatesMenu
saveCSVMenuVariations
saveCSVMenuVariations
saveCSVMenuComparison
saveCSVbyResolution
validateDates
isValidDate
readDayNIM
notInteractiveMode
printVariationsInTXT
printTotalsInTXT
printDaysTillDeathTXT
printPredictionsTXT
calcAndPrintComparisonCumulativeTXT
calcAndPrintComparisonTotalTXT
printComparisonTXT
validateInputFile
validateResolution
printErrorNIM
saveComparisonCSV
saveDailyCSV
saveWeeklyCSV
saveMonthlyCSV
printAllInCSV
fileVerification
fileVerification
checkIfNull
checkIfNull
readDay
readPredictionDay
printPredictions
printTotal
Cálculo
addDays
addMonth
copyArrayContent
sameSizeIntervals
fillArray
dailyFluctuation
calculateIndexRange
variation
weeklyFluctuation
findCompleteWeeks
weekIsComplete
numberOfWeeks
monthlyFluctuation
monthIsComplete
findCompleteMonth
numberOfMonths
differenceVector
calculateAverage
calculateStandardDeviation
dateRangeIsInArray
variationIsPossibleInDayBefore
variationIsPossible
calculateAllVariations
numberOfDays
WeekIntervalsInString
daysInString
monthsInString
fillMatrix
powerMatrix
multiplyMatrices
duplicateMatrix
dateIsInArray
predictDataInDay
calculateIndex
historicData
removeDeaths
subtractMatrices
getTotalDataArrayRequiredValue
getAllTotalArrayRequiredValues
luDecomposition
invertLower
invertUpper
daysToDeathMatrixN
daysToDeathMatrixM
invertMatrix