UnB

CIC0203 - Computação Experimental - TA - 2022.2 - Tarefa T6 - Aprimoramento de uma Simulação

URL Read-only Overleaf: https://www.overleaf.com/read/hmggjqxrmdjc

Repositório Githhub: hhttps://github.com/mutenesss/Fulmen-Mollis

Erick Rodrigues Fraga (mutenesss)

Brasília, 2023-01-06 18:50:51Z

Lista de tarefas pendentes

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Este documento contém o produto da tarefa especificada no título deste documento, conforme as orientações em https://www.overleaf.com/read/cytswcjsxxqh.

Parte I

Simulação Computacional

Capítulo 1

T6 - Aprimoramento de uma Simulação: Laboratório e Experimento Fulmen Mollis, por Erick Rodrigues Fraga (mutenesss)

1.1 Introdução

Este capítulo apresenta a construção e uso do laboratório de simulações *Fulmen Mollis* para a realização de experimentos que tem por objetivo investigar a hipótese causal *Quanto maior a taxa de mortalidade de um vírus, menor a quantidade de seres infectados?* que relaciona variáveis independentes e variáveis dependentes, supostamente presente nos estudos bibliométricos por mim realizados e disponíveis em **??**.

É composto por mais cinco seções:

1. Descrição do fenômeno real;
2. Apresentação do laboratório de simulações;
3. Apresentação de análises exploratórias dos dados de experimentos realizados com o uso do laboratório;
4. Discussão sobre *insights* obtidos após os experimentos; e
5. Conclusões.

1.2 O Fenômeno do Mundo Real

O fenômeno modelado pelo laboratório é a transmissão de um vírus em um grupo de pessoas, sendo assim, um fenômeno social e biológico. Não será definido um ambiente em que essas

pessoas se conectam, será levado em consideração apenas que elas tem conexões entre si, e que essas conexões podem levar a transmissão de um vírus na comunidade.

O artigo (MANOUT; CIARI, 2021) apresenta uma visualização de como o ambiente afeta a transmissão de um vírus, utilizando a COVID-19 como base de estudo.

O livro (AXHAUSEN; ETH ZÜRICH, 2016) apresenta um simulador de comportamentos humanos no transporte, cuja complexidade pode ir aumentando conforme necessidade.

Ambas as referências citadas acima são importantes para este laboratório, pois ambos se complementam na questão do ambiente afetar como um vírus trafega na sociedade baseado no comportamento humano.

1.2.1 Descrição de um Vírus

Biologicamente falando, um vírus é um organismo acelulado, constituído principalmente por proteínas e DNA ou RNA. Seu objetivo básico de vida é propagação e reprodução. Como vírus necessitam de outros seres para executar seu objetivo de vida, eles se movimentam entre conjuntos de seres. Chamaremos os seres que estão infectados de hospedeiros.

Para nós, o importante em relação aos vírus é a forma em que eles se propagam. Nessa simulação, vamos considerar um vírus com transmissão aérea. Essa forma de transmissão pode ser vista em 1.1

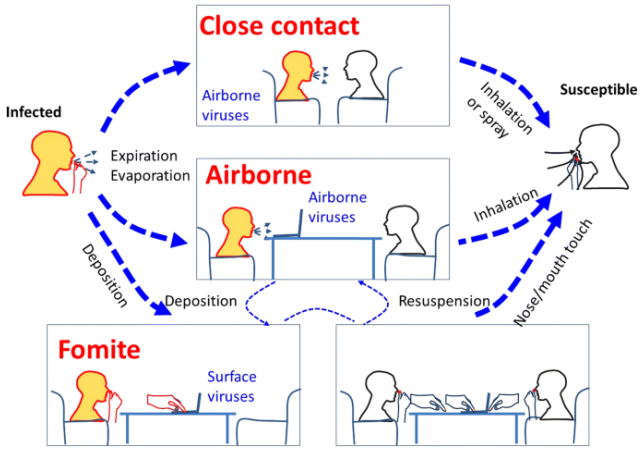

Figura 1.1: Exemplo do modo de transmissão aérea de um vírus.

1.2.2 Definindo um grupo

Para a definição de um grupo, estaremos considerando apenas seres humanos, os quais se conectam por *redes sociais*, as quais podem ser representadas por grafos, como o grafo na 1.2

No grafo, nossos agentes estão interconectados entre si, com todos interagindo com seus vizinhos, e são essas interações que tornam interessante o estudo do fenômeno de transmissão para nós, pois elas definem se um vírus será passado para uma próxima pessoa, assim

expandindo a sua rede de possíveis infecções, ou ele permanece no seu hospedeiro atual, possivelmente sendo contido.

Figura 1.2: Grafo de conexões em uma rede social

1.3 O Laboratório Fulmen Mollis

O laboratório Fulmen Mollis trabalha com um simulador que simula o fenômeno da transmissão de um vírus em uma comunidade. Esse fenômeno é interessante e importante de ser estudado, pois entender como ele funciona nos permite tomar medidas de proteção contra vírus que encontramos no mundo real.

1.3.1 O Conceito da Simulação

O código base dessa simulação foi retirado do exemplo *Virus on Network*, que pode ser encontrado no framework MESA, que é um framework que permite a execução de simulações multi-agente utilizando a linguagem Python.

O código base pode ser encontrando neste repositório.

Considerando que os exemplos do simulador não levam em consideração todas as possíveis variáveis de cada modelo, estaremos observando uma versão simplificada do fenômeno real.

Ainda assim, mesmo de forma simplificada, é possível entender a importância do contato entre hospedeiros e possíveis hospedeiros para a transmissão de um vírus de forma eficaz.

Contextualizando a simulação deste laboratório, estaremos definindo como ambiente um local de trabalho, onde as pessoas entram em contato com seus conhecidos, e esse contato possibilita a transmissão de um vírus dentro desse grupo. Cada pessoa nesse local de trabalho será apresentada na simulação como um nodo, ou uma bolinha, e suas conexões com outras pessoas do local será representada por linhas que ligam os nodos.

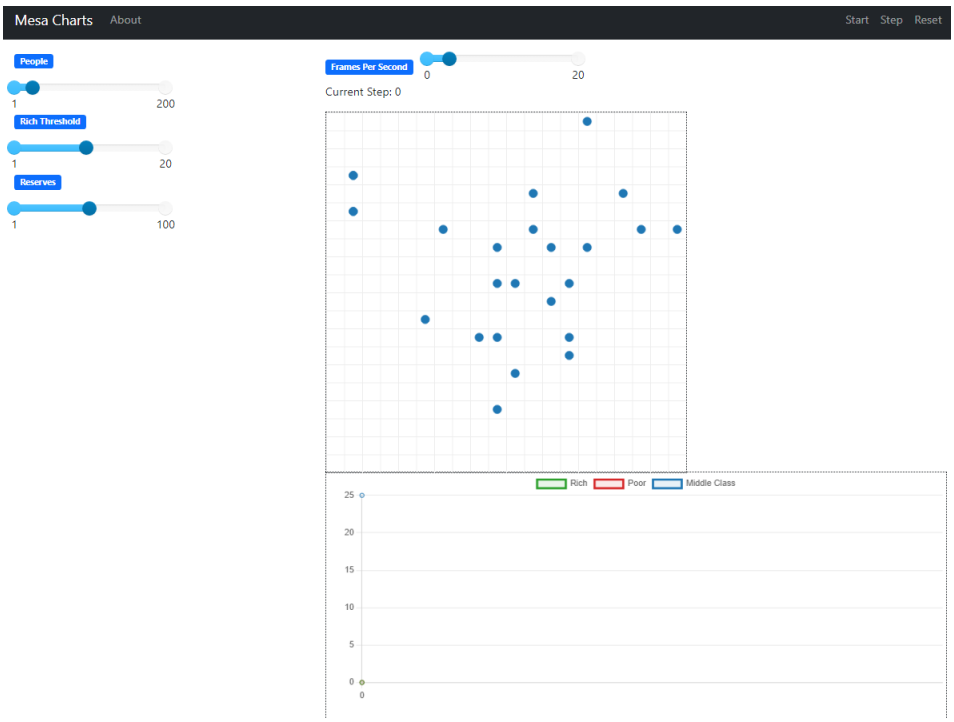

Figura 1.3: Exemplo do Simulador Mesa

1.3.2 O Simulador

Anterior a quaisquer adições feitas neste laboratório, a simulação *Virus on Network* é formada por sete principais variáveis independentes e três variáveis dependentes, as quais serão apresentadas a seguir. A simulação não tem um método de parada por padrão, sendo necessário parar a simulação manualmente ou implementar um método de parada.

Figura 1.4: Exemplo da Simulação *Virus on Network* presente no simulador MESA

1.3.2.1 Variáveis Independentes ou de Controle

São as seguintes as variáveis Independentes ou de Controle, manipuláveis na interface gráfica do simulador:

Number of Agents - Define a quantidade de agentes presentes na simulação. Pode ter valores de 10 a 100 e os valores são incrementados de 1 em 1.

Avg Node Degree - É o grau médio de cada agente, define, em média, quantas conexões cada agente tem com outros agentes. Pode ter valores entre 3 e 8 e os valores são incrementados de 1 em 1.

Initial Outbreak Size - Determina o tamanho inicial de quantas pessoas estarão infectadas com o vírus na simulação. Pode ter valores entre 1 e 10 e os valores são incrementados de 1 em 1.

Virus Spread Chance - É a probabilidade do vírus se transmitir de um portador para outro. Pode der valores entre 0 e 1 e os valores são incrementados de 0.1 em 0.1.

Virus Check Frequency - É a probabilidade dos portadores do vírus testarem se estão infectados ou não. Pode ter valores entre 0 e 1 e os valores são incrementados de 0.1 em 0.1.

Recovery Chance - É a probabilidade de uma pessoa infectada se recuperar da infecção. Pode ter valores entre 0 e 1 e os valores são incrementados de 0.1 em 0.1.

Gain Resistance Chance - É a probabilidade de uma pessoa já recuperada obter resistência ao vírus no futuro. Pode ter valores entre 0 e 1 e os valores são incrementados de 0.1 em 0.1.

1.3.2.2 Variáveis Dependentes

São as seguintes as variáveis Dependentes, cujos valores são coletados e apresentados na interface gráfica do simulador:

Infected - É o número de agentes infectados com o vírus no momento.

Susceptible - É o número de agentes não infectados com o vírus e que não são resistentes ao vírus no momento.

Resistant - É o número de agentes que se recuperaram da infecção e obtiveram resistência ao vírus.

Resistant/Susceptible Ratio - É a porcentagem de agentes que se tornaram resistentes ao vírus comparados aos agentes ainda suscetíveis a infecção.

1.3.3 A Hipótese Causal

Neste laboratório, a hipótese causal a ser observada é *Quanto maior a taxa de mortalidade de um vírus, menor será a sua transmissibilidade em um grupo*. Para isso, será levado em consideração que o vírus se propaga apenas por agentes capazes de interação com outros.

Analisando o dataset obtido anteriormente, não foi possível encontrar artigos que corroborem a hipótese causal deste laboratório, mas foi possível encontrar artigos que indicam que a diminuição da transmissibilidade de um vírus diminuem a quantidade de mortes causadas por ele. A diminuição dessa transmissibilidade pode ser efetuada por diversos processos, os artigos (HINCH et al., 2022) e (ANGELOPOULOU; MYKONIATIS, 2022) apresentam modelos que estimam o efeito que essa variável tem em uma sociedade, e o artigo (SILVA; DAS; IZURIETA, 2017) apresentam um modelo onde são tomadas algumas formas não farmacêuticas para mitigar efeitos de uma pandemia.

1.3.4 O Código do Simulador

Buscando argumentos que se alinhem com a hipótese causal, o simulador *Virus on Network* foi alterado, adicionando um novo estado que representa mortes no modelo, em conjunto com variáveis dependentes desse novo estado.

A figura 1.5 apresenta todas as variáveis independentes do modelo. A figura 1.7 apresenta as variáveis dependentes do modelo, em conjunto com os gráficos criados pela simulação. O resultado da simulação pode ser visto na figura 1.6

Foi adicionado também uma forma de parada do simulador, determinando o extermínio de uma infecção ou o extermínio da população presente no modelo.

A seguir, serão apresentados com detalhes todas as mudanças adicionadas ao simulador.

1.3.4.1 Model.py

Nesta sessão, serão apresentadas as mudanças feitas no arquivo *model.py*, que são as mudanças que alteram a lógica de funcionamento da simulação.

Listagem de Código 1.1: Código da definição de estados.

```
8  class State(Enum):
9      SUSCEPTIBLE = 0
10     INFECTED = 1
11     RESISTANT = 2
12     DEAD = 3
13
14
15 def number_state(model, state):
16     return sum(1 for a in model.grid.get_all_cell_contents() if a.state is state)
17
18
19 def number_infected(model):
20     return number_state(model, State.INFECTED)
21
22
23 def number_susceptible(model):
24     return number_state(model, State.SUSCEPTIBLE)
25
26
27 def number_resistant(model):
28     return number_state(model, State.RESISTANT)
29
30 def number_dead(model):
31     return number_state(model, State.DEAD)
32
33 def number_alive(model):
34     return model.num_nodes - number_dead(model)
```

Na linha 12 é possível visualizar a adição do estado DEAD no simulador, em conjunto com mais dois métodos de retorno de dados, presentes nas linhas 30-34.

Listagem de Código 1.2: Código da definição de variáveis iniciais do modelo

```
39
40     def __init__(
41         self,
42         num_nodes=10,
43         avg_node_degree=3,
44         initial_outbreak_size=1,
45         virus_spread_chance=0.4,
46         virus_check_frequency=0.4,
47         recovery_chance=0.3,
48         gain_resistance_chance=0.5,
49         mortality_chance=0.1,
50     ):
```

Aqui, na função que define o modelo, *VirusOnNetwork*, é possível ver a adição da variável *mortality_chance* na linha 49, que adiciona uma probabilidade de ocorrer o estado de DEAD no modelo.

Listagem de Código 1.3: Código para coleta de dados

```
60         self.virus_spread_chance = virus_spread_chance
```

```
61          self.virus_check_frequency = virus_check_frequency
62          self.recovery_chance = recovery_chance
63          self.gain_resistance_chance = gain_resistance_chance
64          self.mortality_chance = mortality_chance
65
66          self.datacollector = mesa.DataCollector(
67              {
68                  "Infected": number_infected,
69                  "Susceptible": number_susceptible,
70                  "Resistant": number_resistant,
71                  "Dead": number_dead,
72                  "Alive": number_alive,
73              }
74          )
```

Na linha 64 é adicionada aos dados de criação de cada agente, a variável *mortality_ chance* sendo utilizada.

Nas linhas 68-72 estão presentes as definições dos dados que devem ser coletados pelo DataCollector, onde é coletado o número de agentes sobreviventes do modelo, em conjunto ao número de mortos.

Após isso, cada agente do modelo é inicializado.

Listagem de Código 1.4: Código de coleta de agentes vivos no modelo

```
108     def live_agents(self):
109         try:
110             return (number_state(self, State.RESISTANT) + number_state(
111                 self, State.SUSCEPTIBLE)) / self.num_nodes
112         except ZeroDivisionError:
113             return math.inf
```

A função *live_ agents* calcula a quantidade de agentes vivos durante cada passo da simulação. Os dados obtidos por essa função são vistos em um gráfico que será apresentado posteriormente.

Listagem de Código 1.5: Inicialização de cada agente do modelo

```
125 class VirusAgent(mesa.Agent):
126     def __init__(
127         self,
128         unique_id,
129         model,
130         initial_state,
131         virus_spread_chance,
132         virus_check_frequency,
133         recovery_chance,
134         gain_resistance_chance,
135         mortality_chance,
136     ):
137         super().__init__(unique_id, model)
138
139         self.state = initial_state
140
141         self.virus_spread_chance = virus_spread_chance
142         self.virus_check_frequency = virus_check_frequency
143         self.recovery_chance = recovery_chance
144         self.gain_resistance_chance = gain_resistance_chance
145         self.mortality_chance = mortality_chance
146
147     def try_die(self):
148         if self.random.random() < self.mortality_chance:
149             self.state = State.DEAD
150         else:
151             self.try_to_infect_neighbors()
```

A classe *VirusAgent* determina como cada agente deve se portar na simulação. Nas linhas 126-136 é possível visualizar a sua inicialização, em conjunto com todas as variáveis necessárias para tal.

Nas linhas 147-152 está definida a função *try_ die*, que é a função que determina caso um agente infectado deve ser dado como morto ou não.

Listagem de Código 1.6: Definição de cada passo.

```
184     def step(self):
185         if number_infected(self.model) == 0:
186             self.model.running = False
187         else:
188             if self.state is not State.DEAD:
189                 if self.state is State.INFECTED:
190                     self.try_die()
191                 self.try_check_situation()
```

Nas linhas 184-192 estão definidas os passos a serem tomados durante a simulação.

Nas linhas 185-186 está definida a condição de parada do modelo, que é determinada pela ausência de agentes infectados no modelo.

Nas linhas 187-192 está definida a condição a ser avaliada durante cada passo da simulação, onde o estado de um agente só pode mudar caso o agente esteja vivo, limitando assim a capacidade de transmissão do vírus apenas para agentes vivos com vizinhos que também estejam vivos.

1.3.4.2 Server.py

Nesta sessão, serão apresentadas as mudanças executadas no arquivo *server.py*, em conjunto as figuras que apresentam o efeito de todas as mudanças no modelo.

Listagem de Código 1.7: Definição das cores do modelo

```
11      def node_color(agent):
12          return {State.INFECTED: "#FF0000", State.SUSCEPTIBLE: "#008000", State.DEAD: "000000"}.get(
13              agent.state, "#808080"
14          )
```

Nesta parte, é definida a cor de cada estado dos agentes, com agentes infectados representados pela cor vermelha, agentes suscetíveis representados pela cor verde, agentes resistentes ao vírus representados pela cor cinza e agentes mortos representados pela cor preta.

Listagem de Código 1.8: Definição dos gráficos

```
53  chart = mesa.visualization.ChartModule(
54      [
55          {"Label": "Infected", "Color": "#FF0000"},
56          {"Label": "Susceptible", "Color": "#008000"},
57          {"Label": "Resistant", "Color": "#808080"},
58      ]
59  )
60  chart2 = mesa.visualization.ChartModule(
61      [
62          {"Label": "Dead", "Color": "#000000"},
63          {"Label": "Alive", "Color": "#008000"},
64      ]
65  )
66
67  def get_resistant_susceptible_ratio(model):
68      ratio = model.resistant_susceptible_ratio()
69      ratio_text = "&infin;" if ratio is math.inf else f"{ratio:.3f}"
70      infected_text = str(number_infected(model))
71
72      return "Resistant/Susceptible Ratio: {}<br>Infected Remaining: {}".format(
73          ratio_text, infected_text
74      )
75
76  def get_dead_alive_ratio(model):
77      ratio = model.live_agents()
78      ratio_text = "&infin;" if ratio is math.inf else f"{ratio:.3f}"
79      dead_text = str(number_dead(model))
80
81      return "Live/Dead Ratio: {}<br>Number of Dead People: {}".format(
82          ratio_text, dead_text
83      )
```

Nesta parte, são definidos os gráficos que apresentam os dados de cada passo da simulação.

Nas linhas 53-65 é vista a definição de cor do que cada nodo representa em cada gráfico. Ambos os gráficos podem ser vistos em 1.7 sendo as linhas 53-59 o gráfico superior e as linhas 60-65 o gráfico inferior.

Nas linhas 76-83 é definida a função que calcula a porcentagem de agentes vivos no modelo. Essa função obtém os dados da função *live_ agents* presente no arquivo *model.py* e apresenta seu resultado acima do gráfico inferior presente em 1.7.

Listagem de Código 1.9: Definição do Slider de Mortalidade

```
138     "mortality_chance": mesa.visualization.Slider(
139         "Virus Mortality Rate",
140         0.01,
141         0.0,
142         1.0,
143         0.01,
144         description="Probability that a agent will die from the virus",
145     ),
146 }
```

Nesta parte é adicionada um Slider que altera a taxa de mortalidade do modelo, podendo ela ser nula ou 100%. Esse Slider pode ser aumentado por incrementos de 1%, possibilitando assim uma maior variabilidade quando comparada a de incrementos de 10%.

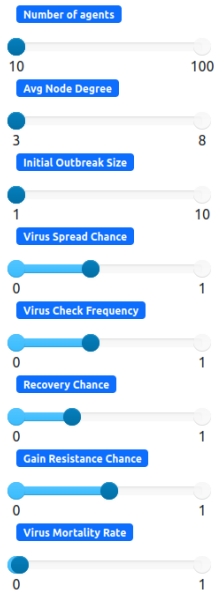

Figura 1.5: Variáveis Independentes da Simulação

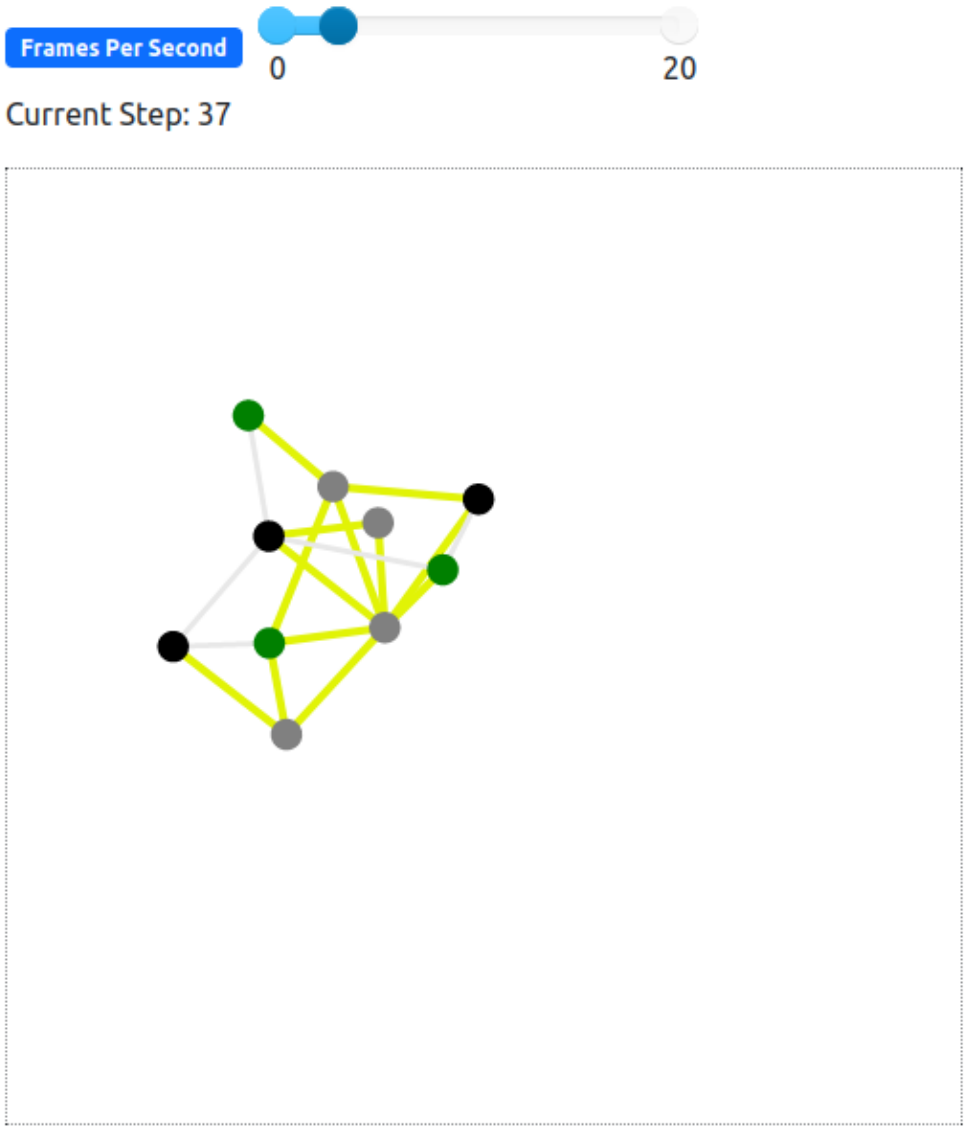

Figura 1.6: Representação visual da simulação

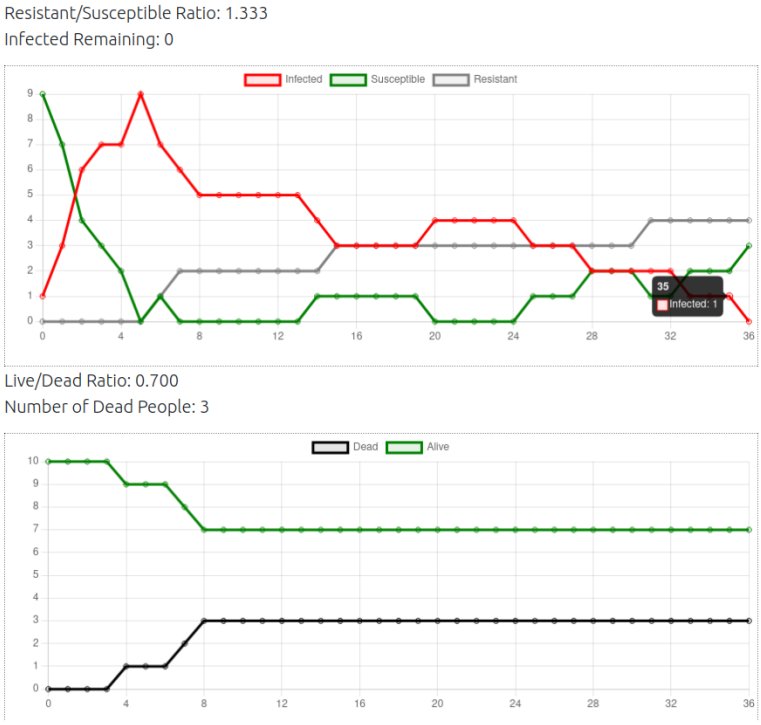

Figura 1.7: Variáveis Dependentes da Simulação

1.4 Os Experimentos Realizados

Foram executados 20 experimentos, divididos em 4 conjuntos de 5 experimentos, alterando apenas uma variável entre cada grupo. Buscando visualizar bem o comportamento da simulação, quaisquer simulações onde acontecia um fim imediato ou o número de mortes fosse exatamente igual ao número de infectados inicialmente, foram desconsiderados para os resultados.

Os parâmetros utilizados na simulação foram os seguintes:

- Number of Agents - 30
- Average Node Degree - 4
- Initial Outbreak Size - 5
- Virus Spread Chance - 0.4
- Virus Check Frequency - 0.4
- Recovery Chance - 0.3
- Gain Resistance Chance - 0.5

- Virus Mortality Rate - 0.1, 0.9, 0.5, 0.34

A lógica para a escolha dos valores para a variável *Virus Mortality* foi:

0.1 ou 10% - Testar a simulação com um valor baixo, mas não o mínimo total do modelo.

0.9 ou 90% - Testar a simulação com um valor alto, mas que tornasse possível visualizar alterações no modelo sem necessitar de muitas execuções.

0.5 ou 50% - Valor intermediário da Simulação.

0.34 ou 34% - Embora sejam diferentes dados, valor inspirado por CFM(Case Fatality Rate) do vírus MERS-CoV apresentado em (MATHIEU et al., 2020).

A tabela com os dados obtidos dos experimentos pode ser observada em 1.1

Virus Mortality	Dead	Resistant	Susceptible	Steps
0.1	12	11	7	28
0.1	18	8	4	27
0.1	20	4	6	20
0.1	22	5	3	12
0.1	18	10	2	21
0.9	8	0	22	3
0.9	7	0	23	4
0.9	9	0	21	3
0.9	7	0	23	4
0.9	4	1	25	2
0.5	11	0	19	5
0.5	10	0	20	5
0.5	19	2	9	9
0.5	12	1	17	5
0.5	6	0	24	5
0.34	13	1	16	10
0.34	7	0	23	4
0.34	8	1	21	4
0.34	17	2	11	11
0.34	21	4	5	14

Tabela 1.1: Dados do Experimeto

1.5 Discussão e *insights* preliminares sobre as hipóteses

Embora os resultados apresentem uma visão favorável a hipótese causal, onde quanto maior a taxa de mortalidade do vírus menor a quantidade de pessoas infectadas por ele, não é possível concluir que a hipótese tenha sido comprovada.

Em principal, podemos observar a ausência de um contador de infectados entre os resultados, podendo obter esse valor apenas inferindo do número de mortos.

Analisando apenas pelo número de mortos, é possível dizer que a hipótese foi comprovada, devido a diminuição do número de mortos quanto maior a variável *Virus Mortality Rate*, ainda assim, é possível observar também grandes flutuações nos resultados obtidos, indicando necessidade de mais testes para confirmação da hipótese.

1.6 Conclusão

Realizar este laboratório foi uma experiência interessante. Ter as primeiras impressões com a simulação multiagente e analisar uma simulação do tipo foi algo que se provou com uma dificuldade razoável, mas ainda muito possível.

Entender diferentes tipos de dados e estatísticas foi interessante, pois tornou possível visualizar diversos erros em inferências feitas anteriormente as pesquisas, como possível ver na variável adicionada modelo, onde a mesma não seria uma porcentagem de mortalidade mas apenas uma probabilidade de causar a morte em um agente.

Acredito que os testes futuros, onde haverão mais execuções do modelo, trarão ainda mais curiosidades sobre o tema estudado.

Bibliografia

ANGELOPOULOU, Anastasia; MYKONIATIS, Konstantinos. Hybrid modelling and simulation of the effect of vaccination on the COVID-19 transmission. en. *Journal of Simulation*, p. 1–12, abr. 2022. ISSN 1747-7778, 1747-7786. DOI: 10.1080/17477778.2022.2062260. Disponível em: <https://www.tandfonline.com/doi/full/10.1080/17477778.2022.2062260>. Acesso em: 6 jan. 2023. Citado na p. 10.

AXHAUSEN, Kay W.; ETH ZÜRICH. *The Multi-Agent Transport Simulation MATSim*. Edição: ETH Zürich. Ubiquity Press, ago. 2016. ISBN 978-1-909188-75-4. DOI: 10.5334/baw. Disponível em: <http://www.ubiquitypress.com/site/books/10.5334/baw/>. Acesso em: 16 dez. 2022. Citado na p. 6.

HINCH, Robert et al. Estimating SARS-CoV-2 variant fitness and the impact of interventions in England using statistical and geo-spatial agent-based models. en. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, v. 380, n. 2233, p. 20210304, out. 2022. ISSN 1364-503X, 1471-2962. DOI: 10.1098/rsta.2021.0304. Disponível em: <https://royalsocietypublishing.org/doi/10.1098/rsta.2021.0304>. Acesso em: 6 jan. 2023. Citado na p. 10.

MANOUT, Ouassim; CIARI, Francesco. Assessing the Role of Daily Activities and Mobility in the Spread of COVID-19 in Montreal With an Agent-Based Approach. *Frontiers in Built Environment*, v. 7, p. 654279, jul. 2021. ISSN 2297-3362. DOI: 10.3389/fbuil.2021.654279. Disponível em: <https://www.frontiersin.org/articles/10.3389/fbuil.2021.654279/full>. Acesso em: 16 dez. 2022. Citado na p. 6.

MATHIEU, Edouard et al. Coronavirus Pandemic (COVID-19). *Our World in Data*, mar. 2020. Disponível em: <https://ourworldindata.org/mortality-risk-covid>. Acesso em: 6 jan. 2023. Citado na p. 17.

SILVA, Walter; DAS, Tapas K.; IZURIETA, Ricardo. Estimating disease burden of a potential A(H7N9) pandemic influenza outbreak in the United States. en. *BMC Public Health*, v. 17, n. 1, p. 898, dez. 2017. ISSN 1471-2458. DOI: 10.1186/s12889-017-4884-5. Disponível em: <https://bmcpublichealth.biomedcentral.com/articles/10.1186/s12889-017-4884-5>. Acesso em: 6 jan. 2023. Citado na p. 10.