Modelo epidemiológico com compartimentos SEIQHRF em redes

Brasil

Modelo epidemiológico com compartimentos SEIQHRF em redes

Modelo canônico de trabalho monográfico acadêmico em conformidade com as normas ABNT apresentado à comunidade de usuários LaTeX.

Universidade Federal de Minas Gerais

Instituto de Ciências Exatas

Departamento de Estatística

Orientador: Adrian Luna

Brasil

65p. : il. (algumas color.) ; 30 cm.
Modelo epidemiológico com compartimentos SEIQHRF em redes/ . – Brasil, -
Orientador: Adrian Luna

Monografia – Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Estatística, -
1. Palavra-chave1 2. Palavra-chave2. 2. Palavra-chave3. I. Orientador. II. Universidade xxx. III. Faculdade de xxx. IV. Título

Modelo epidemiológico com compartimentos SEIQHRF em redes

Modelo canônico de trabalho monográfico acadêmico em conformidade com as normas ABNT apresentado à comunidade de usuários LaTeX.

Trabalho aprovado. Brasil, 1 de novembro de 2021:

Adrian Luna
Adrian Luna

Professor
Ilka Reis

Professor
Glaura Franco

Brasil

Dedicada às pessoas cientistas.

Agradecimentos

Agradeço
ao Adrian, pela orientação;
à Mari, por coisas infinitas;
aos meus amigos, pelo tempo ocioso;
e à minha família, pela oportunidade.

When we can't think for ourselves, we can always quote.
(Ludwig Wittgenstein)

Resumo

A pandemia causada pela COVID-19 motivou o desenvolvimento de novos modelos epidemiológicos, com o objetivo de orientar a tomada de decisão sobre novas políticas públicas no combate à doença. Dado que os impactos da pandemia existem não apenas para os indivíduos infectados pela doença, mas também para a sociedade como um todo, muitos desses modelos descrevem consequênciais sócio-estruturais da pandemia, tais quais a quarentena e a superlotação de hospitais. Por isso, modelos epidemiológicos compartimentais dividem a população em compartimentos, que descrevem os estados dos indivíduos. Normalmente, os estados são relativos apenas à doença, mas, no caso de modelos para a pandemia, é útil generalizar estados para representar características sócio-estruturais dos indivíduos. Este trabalho foi desenvolvido a partir de um desses novos modelos, o SEIQHRF (Susceptible, Exposed, Infected, Quarantined, Hospitalized, Recovered, Fatality). O objetivo foi implementar simulações numéricas do SEIQHRF em grafos, representando redes sociais. Além disso, também exemplificamos alguns casos de uso para as simulações. Até a data de publicação deste trabalho, só existiam implementações do SEIQHRF considerando populações de misturas homogêneas. Com a nossa implementação, agora é possível fazer simulações do modelo em questão para pequenas populações, considerando estruturas mais complexas de mistura. O conteúdo deste trabalho tem a seguinte ordem: uma introdução aos conceitos básicos de modelos epidemiológicos e grafos aleatórios; detalhamentos sobre a implementação do SEIQHRF; ajuste e simulação de um ERGM (exponential random graph model) representando uma escola de ensino médio; exemplos de simulações e testes de hipóteses utilizando o SEIQHRF no grafo em questão.

Palavras-chave: COVID-19. epidemiologia. ERGM. SEIQHRF.

Abstract

This is the english abstract.

Keywords: latex. abntex. text editoration.

Lista de ilustrações

Lista de tabelas

Sumário

Sumário

1 Introdução

O avanço de epidemias em populações fechadas é um problema que pode ser modelado como um processo Markoviano temporal com um número finito de estados, também chamados de compartimentos (KENDALL, 1956). As estimativas fornecidas por esses modelos sob o efeito de diferentes intervenções informam a construção de políticas públicas eficientes no combate às doenças estudadas.

No último ano, muitos modelos foram propostos para estimar a propagação e os impactos da doença COVID-19, causada pelo vírus Sars-CoV-2. Alguns conjuntos de estados foram criados com o propósito de estimar os efeitos sociais e estruturais da pandemia (**??**). Um desses conjuntos é o $SEIQHRF$ (Susceptible-Exposed-Quarantined-Hospitalized-Recovered-Fatality) (CHURCHES, 2020).

Na UFMG, o projeto *COVID-19: proposta de um modelo epidemiológico que incorpora estruturas sociais de contágio* tem mapeado o grafo de relações pessoais do Aglomerado da Serra em Belo Horizonte, com o objetivo de modelar a progressão da COVID-19 em contextos de aglomerações urbanas. Esse projeto motivou a implementação de modelos de redes utilizando os estados $SEIQHRF$ nesta monografia.

Para isso, implementamos os algoritmos necessários para as simulações numéricas. Além disso, também exemplificamos alguns casos de uso para as simulações, utilizando dados de estrutura similar aos dados do Aglomerado da Serra. Até a data de publicação deste trabalho, só existiam implementações do SEIQHRF considerando populações de misturas homogêneas. Agora, com a nossa implementação, é possível fazer simulações do modelo em questão para pequenas populações, considerando estruturas mais complexas de mistura.

O conteúdo deste trabalho tem a seguinte ordem: uma introdução aos conceitos básicos de modelos epidemiológicos e grafos aleatórios; detalhamentos sobre a implementação do SEIQHRF; ajuste e simulação de um ERGM (exponential random graph model) representando uma escola de ensino médio; exemplos de simulações e testes de hipóteses utilizando o SEIQHRF no grafo em questão.

2 Metodologia

...

2.1 Conjuntos de estados

Modelos epidemiológicos são baseados na compartimentalização de indivíduos de uma população. (KEELING; EAMES, 2005) Normalmente, os compartimentos representam estados da doença, mas podem também representar outras propriedades dos indivíduos, conforme discutimos nas subseções a seguir.

2.1.1 SIR

Um conjunto de compartimentos comumente utilizado para doenças infecciosas que conferem imunidade vitalícia (caxumba, por exemplo) é o SIR. Esse conjunto tem três estados:

- **Susceptible**: suscetível. Indivíduos suscetíveis podem entrar em contato com indivíduos infectados. Dado o contato, há probabilidade de que suscetíveis se tornem infectados;
- **Infected**: infectado. Indivíduos infectados podem progredir e se tornarem recuperados;
- **Recovered**: recuperado.

A Figura 1 ilustra as dinâmicas de infecção e progressão entre os compartimentos SIR.

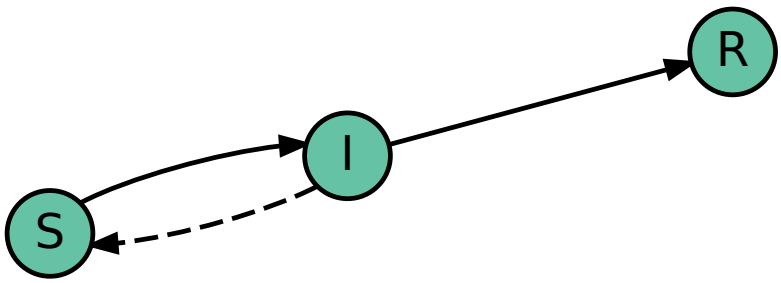

Figura 1 – SIR. Linhas pontilhadas são infecções e linhas sólidas são progressões.

2.1.2 SEIQHRF

Alguns conjuntos de estados foram propostos para modelar as consequências sociais e estruturais da pandemia do COVID-19. Por exemplo, o conjunto $SIDARTHE$ (GIORDANO et al., 2020) (susceptible, infected, diagnosed, ailing, recognized, threatened, healed, extinct) adiciona alguns estados para descrever o estado de identificação e gravidade da doença nos indivíduos.

O $SEIQHRF$ (susceptible, exposed, infected, quarantined, hospitalized, fatality) (CHURCHES, 2020), modelo que implementamos neste trabalho, adiciona quatro estados ao SIR:

- **Exposed**: infectado, assintomático;
- **Quarantined**: infectado, sintomático, pratica isolamento social. Tem uma taxa de encontros reduzida;
- **Hospitalized**: infectado, sintomático, requer hospitalização. Tem uma taxa de encontros reduzida e maior probabilidade de transição para fatalidade. Esse estado tem um limite de vagas. Quando o limite é excedido, a probabilidade de transição para fatalidade é multiplicada por uma constante;
- **Fatality**: morte.

A Figura 2 ilustra as dinâmicas de infecção e progressão do $SEIQHRF$.

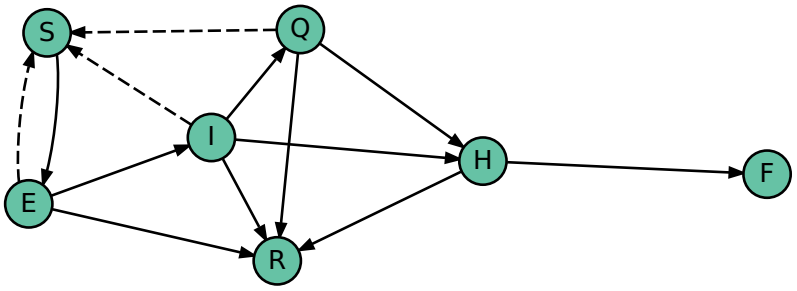

Figura 2 – SEIQHRF. Linhas pontilhadas são infecções e linhas sólidas são progressões.

2.2 Modelos epidemiológicos

Esses modelos descrevem a proporção de indivíduos de uma população em cada estado ao longo do tempo... (**??**)

2.2.1 Determinístico

Sem dinâmicas vitais (processos naturais de nascimento e morte) e com uma população com mistura homogênea (na qual os indivíduos têm probabilidades idênticas de se encontrarem), o SIR é definido pelo seguinte conjunto de equações diferenciais (KEELING; EAMES, 2005):

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -\beta SI$$
$$\frac{\mathrm{d}I}{\mathrm{d}t} = \beta SI - \gamma I$$
$$\frac{\mathrm{d}R}{\mathrm{d}t} = \gamma I$$

Onde β é a taxa de transmissão, γ é a taxa de recuperação da doença, e $S + I + R = 1. \ldots$ R_0

Dados os mesmos pressupostos, o $SEIQHRF$ é definido por um conjunto similar:

$$\begin{aligned}
\frac{dN_S(t)}{dt} &= \nu N - \beta N_S(t)\frac{(N_I(t)+N_E(t)+N_Q(t))}{N} - \mu N_S(t),\\
\frac{dN_I(t)}{dt} &= \beta N_I(t)(N_S(t)+N_E(t)) - \delta_1 N_I(t) - \delta_2 N_I(t) - \gamma N_I(t) - \mu N_I(t),\\
\frac{dN_E(t)}{dt} &= \beta N_E(t)(N_S(t)-N_I(t)) - \mu N_S(t),\\
\frac{dN_Q(t)}{dt} &= \beta N_Q(t) N_S(t) + \delta_1 N_I(t) - \delta_2 N_Q(t) - \gamma N_Q(t) - \mu N_Q(t),\\
\frac{dN_H(t)}{dt} &= \delta_1 N_I(t) + \delta_2 N_Q(t) - \gamma N_H(t) - \gamma N_H(t) - \mu_F N_H(t)(-\mu N_H(t)),\\
\frac{dN_R(t)}{dt} &= \gamma(N_I(t)+N_Q(t)+N_H(t)) - \mu N_R(t),\\
\frac{dN_F(t)}{dt} &= \mu_F N_H(t),
\end{aligned}$$

...

2.2.2 Estocástico

... (ANDERSSON; BRITTON, 2000)

2.2.3 Em redes

Quando o pressuposto de mistura homogênea não é razoável, modelos determinísticos e estocásticos podem ser adaptados, redefinindo os parâmetros do modelo como vetores, representando o comportamento de subpopulações. Contudo, para expressar estruturas sociais mais complexas, podemos utilizar modelos em redes (KEELING; EAMES, 2005).

Seja o grafo $Y = (E, V), Y \in \mathcal{Y}$, no qual os vértices V representam indivíduos e as arestas E representam as relações entre os indivíduos. Seja o processo $X_i^t = 0, 1, 2$, com $i \in V$, onde 0 representa o indivíduo suscetível, 1 infectado e 2 recuperado. O processo $X_i(t)$ obedece:

$$\forall i \in V(g), \quad \mathbb{P}(X_i(t+\delta t) = x_i' | X_i(t) = x_i) = \begin{cases} \beta M_i(x)\delta t & \text{se } x_i = 0 \text{ e } x_i' = 1 \\ \gamma \delta t & \text{se } x_i = 1 \text{ e } x_i' = 2 \\ 1 - [\beta M_i(x) + \gamma]\delta t & \text{c.c,} \end{cases}$$

onde $M_i(x)$ é o número de vizinhos infectados do vértice i. Finalmente $N_S(t), N_I(t)$ e $N_R(t)$ são as somas dos vértices com valores 0, 1 ou 2.

2.3 Grafos aleatórios

Um modelo aleatório para grafos é definido pela coleção

$$\{\Pr_\theta(Y), Y \in \mathcal{Y}_n, \theta \in \Theta\},$$

Onde $\mathcal{Y}_n = \{Y = (E, V), |V| = n, Y \text{ simples}\}$ é uma coleção de grafos simples de ordem n que é chamada de espaço amostral, e

$$\forall \theta \in \Theta, \quad \mathbb{P}_\theta : \mathcal{Y}_n \to [0,1]$$

é uma distribuição de probabilidades em $\mathcal{Y}_n$.

2.3.1 Exponential random graph models

Seja $g : \mathcal{Y} \to \mathbb{R}^p$ o mapa de um grafo às suas estatísticas suficientes, $\theta \in \mathbb{R}^q$ um vetor de q parâmetros do modelo, e $\eta(\theta) : \mathbb{R}^q \to \mathbb{R}^p$ um mapa de θ a parâmetros naturais $\theta \in \mathbb{R}^{\text{l}}$. O modelo exponencial para grafos aleatórios é definido como (HUNTER; HANDCOCK, 2006):

$$\Pr_{\eta,g}(Y = y|\theta) = \frac{\exp(\eta(\theta) \cdot g(y))}{c_{\eta,g}(\theta)}, \quad y \in \mathcal{Y}$$

No caso mais comum, tratado neste trabalho, $\eta(\theta) = \theta$. As estatísticas suficientes $g(y)$ podem ser funções da rede, ou de características dos vértices. Por exemplo, o número de arestas é uma estatística função da rede, enquanto o número de vértices compartilhando um mesmo atributo é uma função dos vértices.

Na literatura em inglês, essa classe de modelos é chamada de *exponential random graph models* (ERGM). Ao longo deste trabalho, vamos nos referir à classe como ERGM. Detalhes sobre estimação de parâmetros foram discutidos na próxima subseção, que contém uma extensão do ERGM.

A flexibilidade do ERGM faz com que essa classe de modelos tenha inúmeras aplicações. ERGMs são comumente utilizados na modelagem de redes sociais (EXPONENTIAL..., 2012). Nesse contexto, os vértices são indivíduos e as arestas são relacionamentos entre os indivíduos. Relacionamentos podem significar uma de várias coisas: amizades, interações, compartilhamentos em plataformas online de redes sociais, etc.

2.3.2 Temporal ERGM

Redes dinâmicas - que evoluem ao longo do tempo - têm inúmeras aplicações. A classe de modelos *Temporal ERGM* (TERGM) é uma extensão da ERGM, que permite a modelagem da evolução de uma rede social por meio da dissolução e formação de arestas ao longo do tempo discreto (KRIVITSKY; HANDCOCK, 2013).

Seja $g : \mathcal{Y}^2 \to \mathbb{R}^p$ a função que mapeia dois grafos a suas estatísticas suficientes para a transição entre as redes y^{t-1} no tempo $t-1$ e y^t no tempo t. A probabilidade de transição entre y^{t-1} e y^t é definida como:

$$\Pr_{\eta,g}(Y^t = y^t|Y^{t-1} = y^{t-1}; \theta) = \frac{\exp(\eta(\theta) \cdot g(y^t, y^{t-1}))}{c_{\eta,g}(\theta, y^{t-1})}, \quad y^t, y^{t-1} \in \mathcal{Y}$$

Generalizando para uma cadeia de ordem k, seja $g : \mathcal{Y}^{k+1} \to \mathbb{R}^p$,

$$\Pr_{\eta,g}(Y^t = y^t|Y^{t-1} = y^{t-1}; \ldots; Y^{t-k} = y^{t-k}; \theta) = \frac{\exp(\eta(\theta) \cdot g(y^{t-1}, \ldots, y^{t-k}))}{c_{\eta,g}(\theta, y^{t-1}, \ldots, y^{t-k})}, \quad y^t, y^{t-1}, \ldots, y^{t-k} \in \mathcal{Y},$$

onde

$$c_{\eta,g}(\theta, y^{t-1}, \ldots, y^{t-k}) = \sum_{y' \in \mathcal{Y}} \exp(\eta(\theta) \cdot g(\theta, y', y^{t-1}, \ldots, y^{t-k})).$$

TERGMs são uma extensão natural de ERGMs. TERGMs são basicamente ERGMs que evoluem ao longo do tempo. Sem perda de generalidade, contanto que a ordem k seja finita, a máxima log-verossimilhaça condicional desse modelo pode ser obtida por meio da maximização de

$$l(\theta) = \eta(\theta) \cdot \big(\sum_{t=1}^{T} g(y^t, y^{t-1}) \big) - \log \big(\prod_{t=1}^{T} c_{\eta,g}(\theta, y^{t-1}) \big)$$

A maximização dessa função envolve o uso de um algoritmo computacional do tipo *Markov Chain Monte Carlo Maximum Likelihood Estimate* (MCMCMLE) (GEYER; THOMPSON, 1992).

Por fim, descrevemos brevemente como gerar uma cadeia de Markov cuja distribuição estacionária é $\Pr_{\eta,g}(Y^t = y^t | Y^{t-1} = y^{t-1}; \ldots; Y^{t-k} = y^{t-k}; \theta)$. Uma cadeia simples que tem essa característica é construída da seguinte forma: a cada passo da cadeia, selecionamos (estocástica ou deterministicamente) uma aresta (i, j), e então decidimos se $Y_{ij} = 0$ ou $Y_{ij} = 1$. Um jeito de fazer isso é utilizando Gibbs sampling, onde o novo valor de Y_{ij} é amostrado da distribuição de $Y_{ij} | Y_{ij}^C = y_{ij}^C$, onde Y_{ij}^C é o resto do grafo. Daí, $Y_{ij} | Y_{ij}^C = y_{ij}^C$ tem uma distribuição de Bernoulli, com chances dadas por:

$$\frac{\Pr(Y_{ij} = 1 | Y_{ij}^C = y_{ij}^C)}{\Pr(Y_{ij} = 0 | Y_{ij}^C = y_{ij}^C)} = \exp(\eta(\theta) \Delta g(y)_{ij}),$$

onde $\Delta g(y)_{ij} = (g(y) | y_{ij} = 1) - (g(y) | y_{ij} = 0)$. Implementações de variações dessa ideia costumam ser feitas por meio de algoritmos de Metropolis-Hastings (HUNTER; HANDCOCK, 2006).

2.3.3 Separable TERGMs

...

3 Implementação

Os objetivos principais deste trabalho foram implementar e demonstrar o modelo epidemiológico $SEIQHRF$ em redes. Nesta seção, detalhamos as partes mais importantes da implementação, incluindo bibliotecas, funções e algoritmos utilizados ou desenvolvidos. As três ferramentas principais desta seção são a linguagem de programação R (R Core Team, 2021); o pacote `ergm` (HANDCOCK et al., 2020) para o ajuste via MCMC dos ERGMs e suas respectivas simulações; e o pacote `EpiModel` (JENNESS; GOODREAU; MORRIS, 2018) para as simulações dos modelos epidemiológicos em redes.

A implementação do $SEIQHRF$ no `EpiModel` envolveu a programação de algumas funções customizadas, descrevendo o processo de simulação do modelo em redes. Essas funções e todo o código para reproduzir esta monografia estão disponíveis no apêndice A deste trabalho, e também no Github. (**??**)

3.1 ergm

O pacote `ergm` foi utilizado para ajustar um ERGM. Para os fins deste trabalho, o uso deste pacote foi trivial, então não detalharemos os algoritmos a fundo. Aqui, apenas destacaremos as funções que utilizamos, junto à documentação relevante dessas funções. As subseções a seguir são apresentadas no formato:

`função(parâmetros)`

Descrição da função.

3.1.1 Ajuste

`ergm(formula)`

Ajusta um ERGM do tipo $\exp(\hat{\theta} \cdot g(y))$ a partir de uma `formula`. Uma `formula` tem o formato `network ~ s_1 + ... + s_n`, na qual `network` é o grafo observado y e `s_1 + ... + s_n` são as funções que calculam as estatísticas de $g(y)$. O ajuste é feito por um algoritmo MCMCMLE de Metropolis-Hastings.

3.1.2 Estatísticas

`edges`

Número de arestas - tamanho do conjunto $\{(i, j)\}$.

`nodematch(attr, diff = FALSE)`

Homofilia uniforme - tamanho do conjunto $\{(i, j) | \text{atributo}_i = \text{atributo}_j\}$.

`nodematch(attr, diff = TRUE)`

homofilia diferencial. p estatísticas da rede, onde p é o tamanho do conjunto {atributos}. $\{(i, j)\ | \text{atributo}_i = \text{atributo}_j = k\}$, onde o valor de k é o menor valor único do atributo em questão.

`absdiff(attr)`

diferenca absoluta - $|\text{atributo}_i - \text{atributo}_j|$.

```
gwesp(decay, fixed)

...
```

3.2 EpiModel

O pacote `EpiModel` (JENNESS; GOODREAU; MORRIS, 2018) possui uma *Application Programming Interface* (API) para ajustar modelos epidemiológicos. No `EpiModel`, modelos determinísticos são chamados de *Deterministic Contact Models* (DCMs), modelos estocásticos de *Invidual Contact Models* (ICMs), e modelos em redes de *Network Models*.

Para demonstrar o uso do `EpiModel`, fizemos o ajuste de dois modelos SIR, um DCM e um ICM, com parâmetros iguais. O tamanho da população é $N = 1000$, com $\beta = 0.16$ e $\gamma = 0.02$. No tempo 0, $S = 90$ e $I = 10$. A Figura 3 ilustra os cenários estimados pelos modelos em questão.

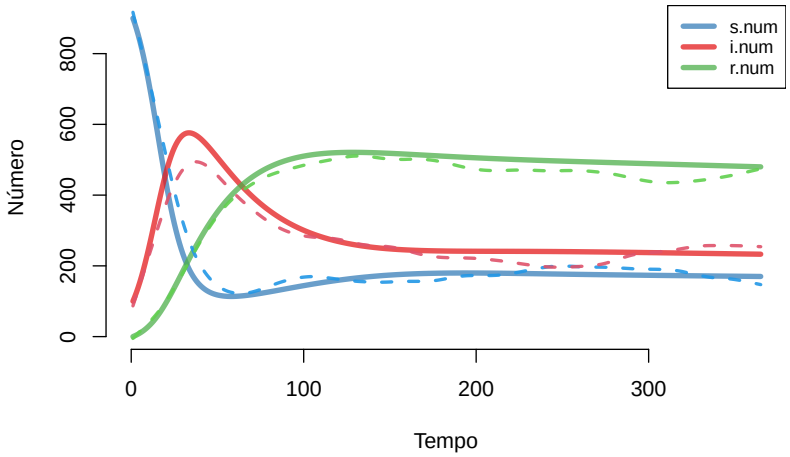

Figura 3 – Comparação entre ajustes DCM (linhas sólidas) e ICM (linhas tracejadas).

Apesar do uso do `EpiModel` ser trivial para modelos comuns, tais quais o SIR e o $SEIR$, modelos mais complexos dependem da implementação de funções customizadas, chamas de módulos. No caso de modelos em redes, primeiro é necessário ajustar um STERGM por meio da função `netest`. Depois, esse ajuste é utilizado pela função `netsim` para gerar simulações do modelo epidemiológico. Essas simulações dependem dos módulos em questão. No caso do $SEIQHRF$, programamos dois módulos: o `infect` e o `progress`.

O fluxo descrito acima está ilustrado no diagrama da Figura 4, e as funções relevantes detalhadas nas subseções a seguir.

3.2.1 netest

Ajusta um ERGM e utiliza um método de aproximação (CARNEGIE et al., 2014) para transformá-lo em um STERGM, transformando um modelo de rede estática em um modelo de rede dinâmica. Essa função é basicamente uma função auxiliar que usa as funções `ergm` e `stergm` do pacote `ergm` em conjunto com a lógica necessária para a aproximação.

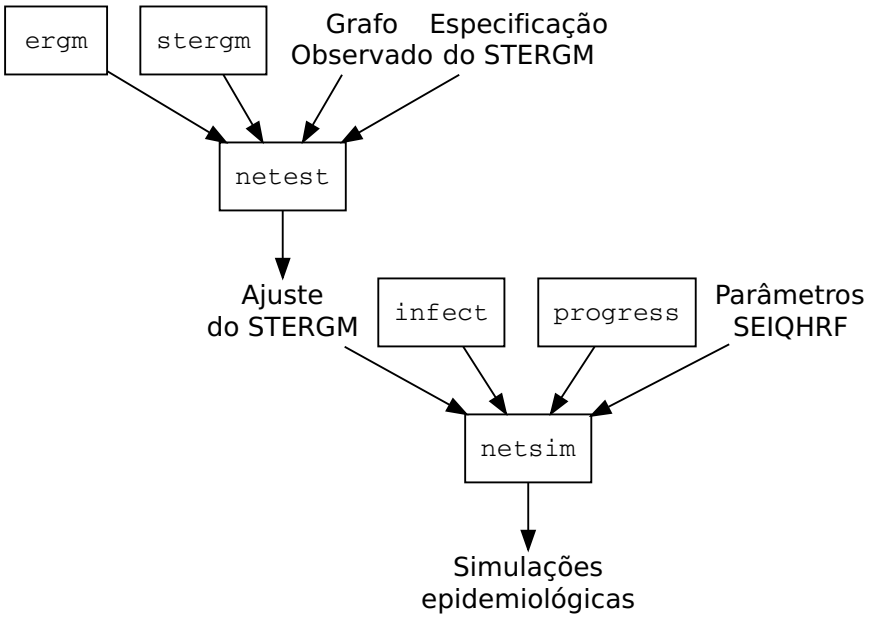

Figura 4 – Fluxograma da implementação.

A aproximação depende de alguns pré-requisitos, que não exploramos a fundo neste trabalho. O mais importante é que as estatísticas usadas para os coeficientes de dissolução devem ser um subconjunto das estatísticas usadas para os coeficientes do modelo estático. Dado isso, grosso modo, um ERGM é estimado com coeficientes de formação baseados nos coeficientes da rede estática. Desses coeficientes, subtrai-se os coeficientes de dissolução.

3.2.2 netsim

Essa função simula os modelos epidemiológicos de acordo com os parâmetros de entrada. Simplificada para o contexto deste trabalho, a `netsim` é descrita pelo pseudocódigo abaixo:

Algorithm 1: netsim

Result: Simulações epidemiológicas

Input : Ajuste do STERGM, Número de simulações, Tempo das simulações, Função de infecção (infect), Função de progressão (progress), Parâmetros das funções de infecção e progressão

Output: Lista de matrizes, cada uma representando uma simulação, com o número de indivíduos em cada estado em cada tempo.

```
for simulacão = 1 até simulação = número de simulações do
    for simulacão = 1 até simulação = número de simulações do
        simular um STERGM a partir do ajuste
        for tempo = 1 até tempo = tempo da simulação do
            formar ou dissolver arestas do STERGM
            rodar o processo de infecção
            rodar o processo de progressão
            salvar os resultados
        end
    end
end
```

3.2.3 infect

...

```
infect <- function(dat, at) {

  ## Attributes ##
  active <- get_attr(dat, "active")
  status <- get_attr(dat, "status")

  ## TODO ##
  if (at == 2) {
    infTime <- rep(NA, length(active))
    infTime[which(status == "i")] <- 1
    dat <- set_attr(dat, "infTime", infTime)
  } else {
    infTime <- get_attr(dat, "infTime")
  }

  ## Parameters ##
  inf.pars <- get_param(dat, "inf.pars")
  prog.pars <- get_param(dat, "prog.pars")
```

```
sum.pars <- get_param(dat, "sum.pars")

## Find infected nodes ##
infectedStatus <- sum.pars$infected.status
idsInf <- which(active == 1 & status %in% infectedStatus)
nActive <- sum(active == 1)
nElig <- length(idsInf)

## Initialize default incidence at 0 ##
nInf <- 0

## If any infected nodes, proceed with transmission ##
if (nElig > 0 && nElig < nActive) {

  ## Look up discordant edgelist ##
  infectiveStatus <- sum.pars$infective.status
  del <- discord_edgelist(dat, at, infectiveStatus)

  ## If any discordant pairs, proceed ##
  if (!is.null(del)) {

    # Set parameters on discordant edgelist data frame
    del <- merge(del, inf.pars, by = "from")

    # Stochastic transmission process
    transmit <- rbinom(nrow(del), 1, del$final.prob)

    # Keep rows where transmission occurred
    del <- del[transmit == 1, ]

    # Look up new ids if any transmissions occurred
    idsNewInf <- unique(del$sus)
    nInf <- length(idsNewInf)

    # Set new attributes for those newly infected
    if (nInf > 0) {
      status[idsNewInf] <- "e"
      infTime[idsNewInf] <- at
      dat <- set_attr(dat, "status", status)
      dat <- set_attr(dat, "infTime", infTime)
    }
  }
}
```

```
  ## Save summary statistic for S->E flow
  dat <- set_epi(dat, "se.flow", at, nInf)

  return(dat)
}
```

3.2.4 progress

...

```
progress <- function(dat, at) {

  ## Attributes ##
  active <- get_attr(dat, "active")
  status <- get_attr(dat, "status")
  previousStatus <- status

  ## Parameters ##
  progPars <- get_param(dat, "prog.pars")
  sum.pars <- get_param(dat, "sum.pars")

  ## X to Y progression process ##
  progRates <- sum.pars$prog.rates
  probProg <- progRates[status]
  probProg[is.na(probProg)] <- 0
  isProg <- active & rbinom(length(status), 1, probProg)
  noProg <- all(isProg == FALSE)

  if (noProg) {
    status <- status
  } else {
    nextStates <- sum.pars$next.states
    nextProbs <- sum.pars$next.probs
    statusProg <- status[isProg == 1]
    status[isProg == 1] <- mapply(
      sample,
      x = nextStates[statusProg],
      size = 1,
      prob = nextProbs[statusProg]
    )
  }

  ## Write out updated status attribute ##
```

```
  dat <- set_attr(dat, "status", status)

  ## Save flows ##
  uniqueFlowNames <- sum.pars$flow.names
  vecFlows <- paste0(previousStatus, status, ".flow")[isProg]
  progFlows <- table(vecFlows)
  progFlowNames <- names(progFlows)
  for (flowName in uniqueFlowNames) {
    value <- if (flowName %in% progFlowNames) progFlows[flowName] else 0
    dat <- set_epi(dat, flowName, at, value)
  }

  ## Save nums ##
  uniqueNumNames <- sum.pars$num.names
  progNums <- table(status[active == 1])
  names(progNums) <- paste0(names(progNums), ".num")
  for (numName in uniqueNumNames) {
    value <- if (numName %in% names(progNums)) progNums[numName] else 0
    dat <- set_epi(dat, numName, at, value)
  }

  return(dat)
}
```

4 Simulação e resultados

O objetivo dessas simulações é demonstrar o modelo proposto nesta monografia, que futuramente será utilizado na modelagem da COVID-19 no Aglomerado da Serra. Os dados que utilizamos são um *proxy* para os dados reais, que são uma rede social representando uma pequena comunidade fechada, na qual as relações entre indivíduos têm padrões por atributos de faixa etária.

4.1 Simulação dos dados

4.1.1 Faux Magnolia High

O pacote `ergm` (HANDCOCK et al., 2020) contém diversos exemplos de grafo, entre eles o `faux.magnolia.high`. Este grafo é uma simulação de uma rede de amigos de uma escola do sul dos Estados Unidos, onde os vértices representam alunos e as arestas amizades entre alunos.

A documentação do `ergm` contém detalhes sobre o modelo utilizado na simulação do `faux.magnolia.high`.

> The data set is based upon a model fit to data from two school communities from the AddHealth Study, Wave I (RESNICK et al., 1997). It was constructed as follows:
>
> The two schools in question (a junior and senior high school in the same community) were combined into a single network dataset. Students who did not take the AddHealth survey or who were not listed on the schools' student rosters were eliminated, then an undirected link was established between any two individuals who both named each other as a friend. All missing race, grade, and sex values were replaced by a random draw with weights determined by the size of the attribute classes in the school. (HANDCOCK et al., 2020)

Existem 1461 alunos, e cada aluno tem três atributos: `Grade` (turma), `Race` (raça) e `Sex` (sexo).

A figura 5 ilustra o grafo em questão.

A figura 6 ilustra a proporção dos níveis de cada atributo.

4.1.2 Ajuste do ERGM

Para simplificarmos a notação, discutiremos a especificação do modelo na sintaxe de `formula` do `ergm`, discutida na seção de implementação. A especificação do modelo é:

```
magnolia ~
  edges +
  nodematch("Grade", diff = TRUE) +
  nodematch("Race", diff = TRUE) +
  nodematch("Sex", diff = FALSE) +
  absdiff("Grade") +
  gwesp(0.25, fixed = TRUE)
```

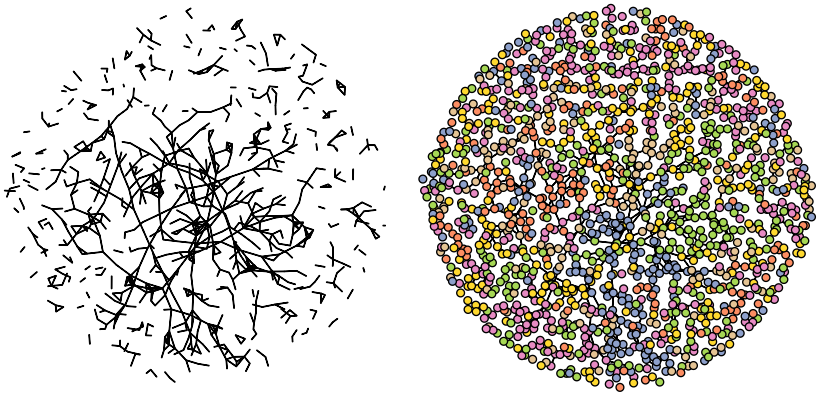

Figura 5 – Faux Magnolia High. À esquerda, apenas as arestas do grafo. À direita, os vértices coloridos de acordo com a turma do aluno.

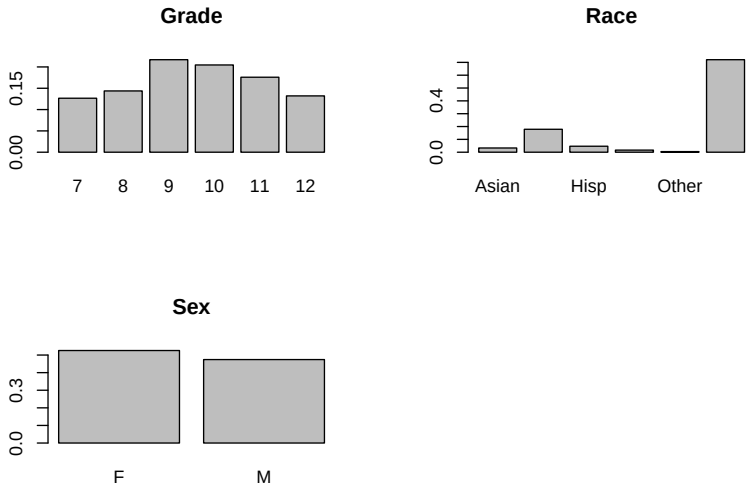

Figura 6 – Proporções dos níveis de cada atributo do Faux Mangolia High.

Tabela 1 – Coeficientes do ajuste Magnolia.

Estatística	Fator	Nível	Coeficiente	OR
edges	NA	NA	-8.27	0.00
nodematch	Grade	7	1.50	4.50
nodematch	Grade	8	1.42	4.13
nodematch	Grade	9	0.96	2.60
nodematch	Grade	10	1.06	2.88
nodematch	Grade	11	1.20	3.32
nodematch	Grade	12	1.29	3.63
nodematch	Race	Asian	1.96	7.07
nodematch	Race	Black	1.36	3.89
nodematch	Race	Hisp	0.36	1.43
nodematch	Race	NatAm	3.46	31.86
nodematch	Race	Other	-Inf	0.00
nodematch	Race	White	0.88	2.42
nodematch	Sex	NA	0.73	2.07
absdiff	Grade	NA	-0.91	0.40
gwesp	NA	NA	1.86	6.43

Usamos a especificação documentada para ajustar um ERGM. As estimativas dos coeficientes deste ajuste estão na tabela 1. Como nossa especificação do ERGM descreve perfeitamente o processo gerador dos dados, não é necessário que verifiquemos o ajuste do modelo criteriosamente. A Figura 7 mostra um exemplo de grafo simulado a partir do nosso ajuste. É visualmente notável a similaridade estrutural das arestas. Nessa simulação, os vértices são idênticos, tanto em quantidade quanto em atributos.

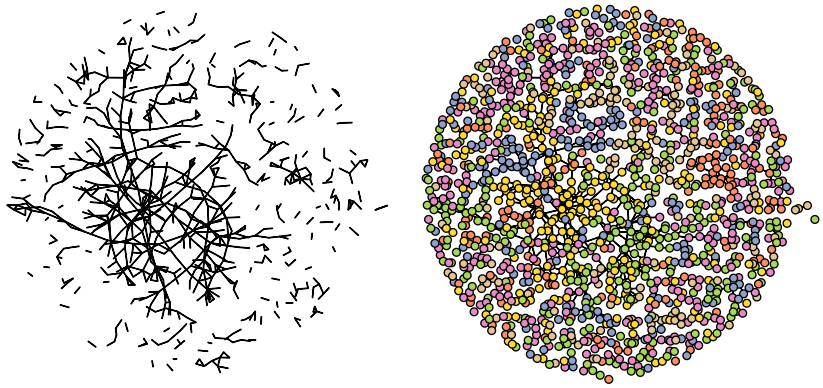

Figura 7 – Simulação gerada com nosso ajuste do Faux Magnolia High. À esquerda, apenas as arestas do grafo. À direita, os vértices coloridos de acordo com a turma do aluno.

A Figura **??** mostra uma comparação entre a distribuição de graus dos dois grafos.

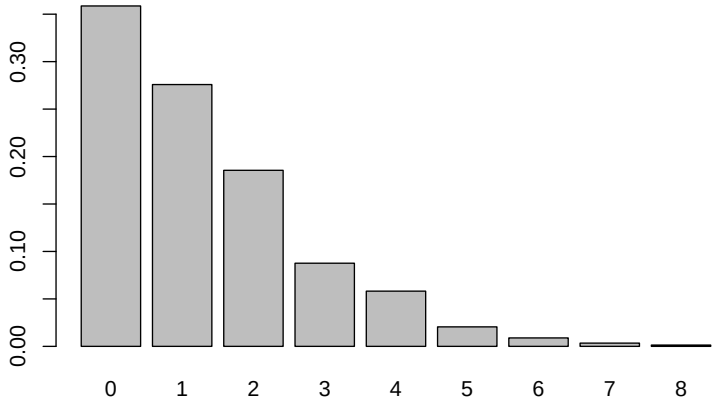

Figura 8 – Comparação entre as distribuições de grau do grafo original e da simulação.

```
## [1] 1.333333
```

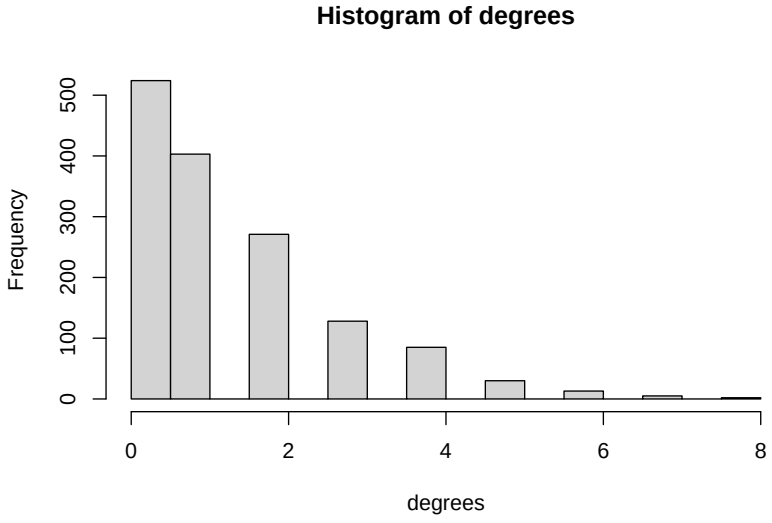

Figura 9 – Comparação entre as distribuições de grau do grafo original e da simulação.

4.2 Simulações epidemiológicas

Nesta seção, demonstramos o modelo $SEIQHRF$ em redes. Para o modelo da rede, ajustamos e simulamos ERGMs baseados em uma *high school* estadunidense. Escolhemos essa rede por representar uma

Tabela 2 – Parâmetros de infecção do Cenário A.

De	Para	Encontros/dia	P(Infecção)	P(Encontro e infecção)
E	S	10.0	0.02	0.18
I	S	10.0	0.05	0.40
Q	S	2.5	0.02	0.05

pequena população com relações determinadas por atributos dos indivíduos, particularmente por suas faixas etárias (representadas pelas turmas dos alunos).

O projeto que originou este trabalho, *COVID-19: proposta de um modelo epidemiológico que incorpora estruturas sociais de contágio*, tem como foco a rede social do Aglomerado da Serra, que tem uma estrutura similar de subpopulações de diferentes faixas.

Antes de simular os modelos epidemiológicos, fixamos alguns parâmetros para as simulações. A simulação relizada pelo `EpiModel` funciona com unidades arbitrárias de tempo. No contexto do nosso exemplo, o tempo é contado em dias, e as simulações rodam por 170 dias. Essa escolha foi feita porque 170 é o número médio de dias letivos nos Estados Unidos. Por consequência, os parâmetros de taxa de encontro têm frequência diária. Ademais, fixamos o número de 30 simulações por cenário. Esse número foi escolhido para obtermos um tempo computacional reduzido e um tamanho amostral razoável para estimarmos a variabilidade nas simulações.

Além disso, fixamos um número inicial de 20 indivíduos expostos (assintomáticos) e 10 indivíduos infectados (sintomáticos). Finalmente, fixamos também os parâmetros relativos aos hospitais. Para a população de alunos da escola, estabelecemos 5 vagas no estado de hospitalização. Quando há mais de 5 alunos hospitalizados, significa que os hospitais estão superlotados. Por consequência, a probabilidade de transição para fatalidade é multiplicada por 1.5.

4.2.1 Simulação exploratória

Fizemos uma primeira simulação exploratória, a qual chamamos de Cenário A. Essa simulação teve o objetivo de demonstrar o comportamento do modelo com alguns parâmetros simples. Os parâmetros de taxa de progressão entre os estados da doença e de probabilidade de infecção por encontro foram sugeridos no artigo que deu origem ao $SEIQHRF$ (CHURCHES, 2020), e são baseados em estudos preliminares da COVID-19. Os parâmetros de infecção estão na tabela 2, e os parâmetros de progressão estão na tabela 3.

Rodamos 30 simulações nesse cenário, e salvamos os resultados. Uma visualização interessante para modelos de grafos está na Figura 10, que ilustra os estados de cada vértice da rede ao longo do tempo. A Figura 11 mostra as prevalências médias ao longo do tempo das 30 simulações do Cenário A.

Podemos ver a distribuição por turma...

4.2.2 Comparação de cenários

No $SEIQHRF$, o estado de quarentena representa apenas uma situação de auto-quarentena para os indivíduos infectados. Contudo, podemos simular um cenário com taxas de encontro reduzidas para entendermos os possíveis efeitos de uma situação de isolamento social sobre o modelo.

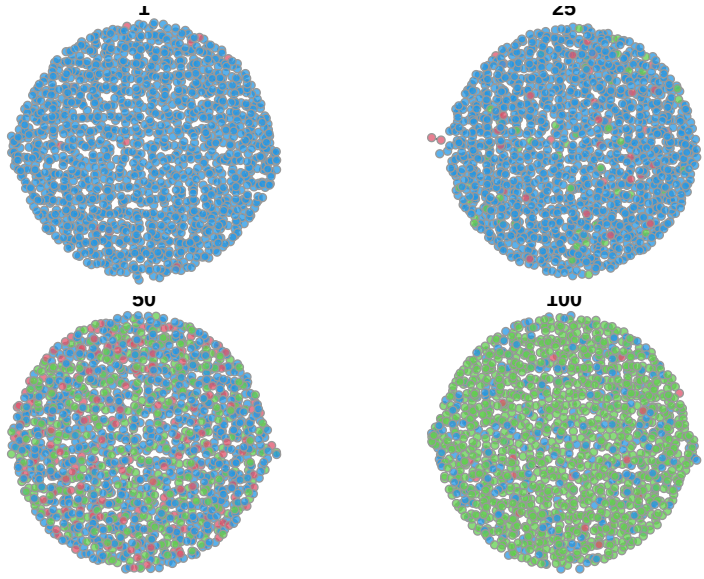

Figura 10 – Rede ao longo de uma das simulações do Cenário A. O número acima da rede é o número de dias percorridos, e as cores indicam vértices suscetíveis (azul), em estados de infecção (vermelho) e recuperados (verde).

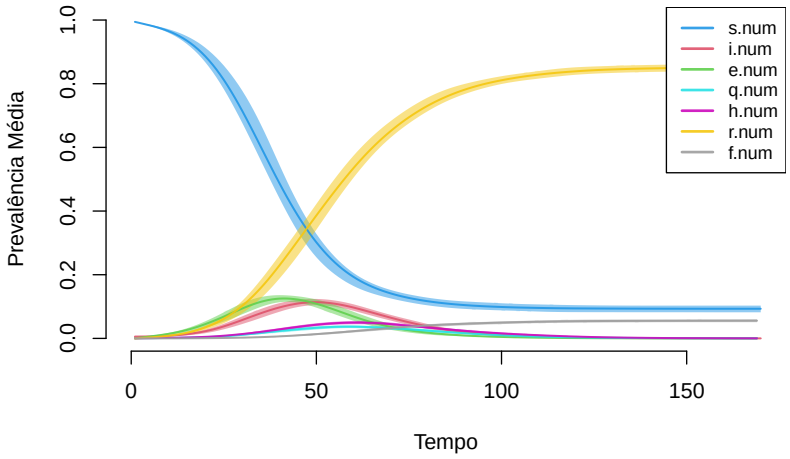

Figura 11 – Prevalência ao longo do tempo no Cenário A. Linhas são prevalências médias, e áreas são intervalos de prevalência.

Tabela 3 – Parâmetros de infecção do Cenário A.

De	Para	Transições/dia
E	I	0.10
E	R	0.05
I	Q	0.03
I	H	0.03
I	R	0.05
Q	H	0.03
Q	R	0.05
H	R	0.07
H	F	0.02

Tabela 4 – Parâmetros de infecção do Cenário B.

De	Para	Encontros/dia	P(Infecção)	P(Encontro e infecção)
E	S	3.00	0.02	0.06
I	S	3.00	0.05	0.14
Q	S	0.75	0.02	0.02

Dado isso, demonstramos uma comparação de cenários com o nosso modelo, reduzindo as taxas de encontro em 70%, produzindo os parâmetros de infecção da Tabela 4. Chamamos esse cenário de Cenário B em contraposição ao Cenário A, proposto na subseção anterior (ver tabela Tabela 2).

Notamos que, no Cenário B, ilustrado pela Figura 12, a curva de prevalência dos estados infectados é completamente achatada, em contraste ao Cenário A.

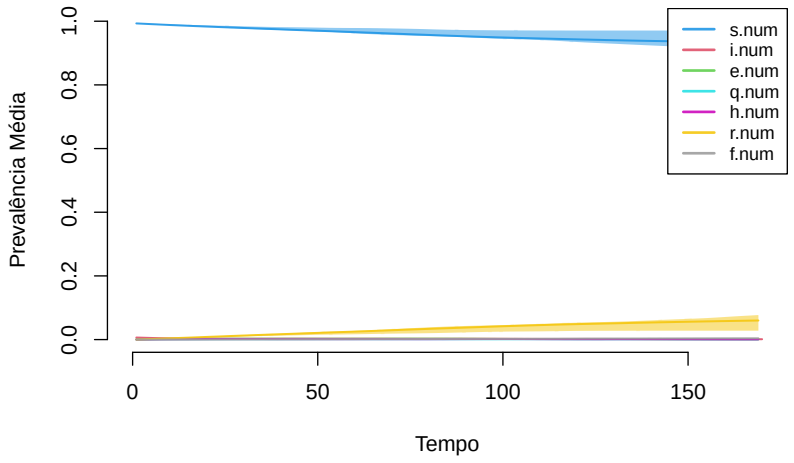

Figura 12 – Prevalência ao longo do tempo no Cenário A. Linhas são prevalências médias, e áreas são intervalos de prevalência.

5 Conclusões e trabalhos futuros

...

Referências

ANDERSSON, H.; BRITTON, T. Stochastic epidemic models and their statistical analysis, volume 151 of lecture notes in statistics. In: _____. [S.l.: s.n.], 2000. v. 151. Citado na página 27.

CARNEGIE, N. et al. An approximation method for improving dynamic network model fitting. *Journal of Computational and Graphical Statistics*, v. 24, p. 00–00, 05 2014. Citado na página 32.

CHURCHES, T. *Tim Churches Health Data Science Blog: Modelling the effects of public health interventions on COVID-19 transmission using R - part 2*. 2020. Disponível em: <https://timchurches.github.io/blog/posts/2020-03-18-modelling-the-effects-of-public-health-interventions-on-covid-19-transmission-part-2/>. Citado 3 vezes nas páginas 23, 26 e 43.

EXPONENTIAL Random Graph Models for Social Networks: Theory, Methods, and Applications. [S.l.]: Cambridge University Press, 2012. (Structural Analysis in the Social Sciences). Citado na página 28.

GEYER, C. J.; THOMPSON, E. A. Constrained monte carlo maximum likelihood for dependent data. *Journal of the Royal Statistical Society. Series B (Methodological)*, [Royal Statistical Society, Wiley], v. 54, n. 3, p. 657–699, 1992. ISSN 00359246. Disponível em: <http://www.jstor.org/stable/2345852>. Citado na página 29.

GIORDANO, G. et al. Modelling the covid-19 epidemic and implementation of population-wide interventions in italy. *Nature Medicine*, Springer Science and Business Media LLC, v. 26, n. 6, p. 855–860, Apr 2020. ISSN 1546-170X. Disponível em: <http://dx.doi.org/10.1038/s41591-020-0883-7>. Citado na página 25.

HANDCOCK, M. S. et al. *ergm: Fit, Simulate and Diagnose Exponential-Family Models for Networks*. [S.l.], 2020. R package version 3.11.0. Disponível em: <https://CRAN.R-project.org/package=ergm>. Citado 2 vezes nas páginas 31 e 39.

HUNTER, D.; HANDCOCK, M. Inference in curved exponential family models for networks. *Journal of Computational and Graphical Statistics*, American Statistical Association, v. 15, n. 3, p. 565–583, set. 2006. ISSN 1061-8600. Funding Information: The authors are grateful to Steven Goodreau and Tom Snijders for very helpful comments and discussions. This research is supported by Grant DA012831 from NIDA and Grant HD041877 from NICHD. Citado 2 vezes nas páginas 28 e 29.

JENNESS, S. M.; GOODREAU, S. M.; MORRIS, M. Epimodel: An r package for mathematical modeling of infectious disease over networks. *Journal of Statistical Software, Articles*, v. 84, n. 8, p. 1–47, 2018. ISSN 1548-7660. Disponível em: <https://www.jstatsoft.org/v084/i08>. Citado 2 vezes nas páginas 31 e 32.

KEELING, M. J.; EAMES, K. T. Networks and epidemic models. *Journal of The Royal Society Interface*, v. 2, n. 4, p. 295–307, 2005. Disponível em: <https://royalsocietypublishing.org/doi/abs/10.1098/rsif.2005.0051>. Citado 3 vezes nas páginas 25, 26 e 27.

KENDALL, D. G. Deterministic and stochastic epidemics in closed populations. In: _____. *Volume 4 Contributions to Biology and Problems of Health*. University of California Press, 1956. p. 149–166. Disponível em: <https://doi.org/10.1525/9780520350717-011>. Citado na página 23.

KRIVITSKY, P. N.; HANDCOCK, M. S. A separable model for dynamic networks. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Wiley, v. 76, n. 1, p. 29–46, Mar 2013. ISSN 1369-7412. Disponível em: <http://dx.doi.org/10.1111/rssb.12014>. Citado na página 28.

R Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria, 2021. Disponível em: <https://www.R-project.org/>. Citado na página 31.

RESNICK, M. et al. Protecting adolescents from harmfindings from the national longitudinal study on adolescent health. *JAMA : the journal of the American Medical Association*, v. 278, p. 823–32, 09 1997. Citado na página 39.

Apêndices

APÊNDICE A – Código e versionamento

```
knitr::opts_chunk$set(
  echo = FALSE,
  fig.align = "center",
  out.width = "67%",
  out.height = "67%",
  message = FALSE,
  warning = FALSE,
  cache = TRUE
)
COLORS <- RColorBrewer::brewer.pal(8, "Set2")
USE_CACHE <- TRUE
library(DiagrammeR)

plot_states <- function(states, edge_df) {
  set.seed(0)
  set <- paste(states, collapse = "")
  path <- file.path("images", paste0(set, ".png"))
  create_graph() |>
    add_node(
      label = states,
      node_aes = node_aes(
        color = "black",
        fillcolor = COLORS[1],
        fontcolor = "black",
        height = 0.25,
        width = 0.25
      )
    ) |>
    add_edge_df(edge_df) |>
    render_graph(layout = "fr")
}

states <- c("S", "I", "R")
ids <- seq_along(states)
names(ids) <- states
edge_df <- create_edge_df(
  from = ids[c("S", "I", "I")],
  to = ids[c("I", "R", "S")],
```

```
  style = c("solid", "solid", "dashed"),
  color = "black"
)
plot_states(states, edge_df)
states <- c("S", "E", "I", "Q", "H", "R", "F")
ids <- seq_along(states)
names(ids) <- states
from <- c('E', 'E', 'I', 'I', 'I', 'Q', 'Q', 'H', 'H', 'S', 'E', 'I', 'Q')
to <- c('I', 'R', 'Q', 'H', 'R', 'H', 'R', 'R', 'F', 'E', 'S', 'S', 'S')
edge_df <- create_edge_df(
  from = ids[from],
  to = ids[to],
  style = rep(c("solid", "dashed"), c(10, 3)),
  color = "black"
)
plot_states(states, edge_df)
library(EpiModel)
# set.seed(1995)

# Parameters
params <- list(
  inf.prob = 0.2, act.rate = 0.8, rec.rate = 1/50,
  a.rate = 1/100, ds.rate = 1/100, di.rate = 1/90,
  dr.rate = 1/100
)
N <- 1000
inits <- list(s.num = N * 0.9, i.num = N * 0.1, r.num = 0)
controls <- list(type = "SIR", nsteps = 365)

# DCM
param <- do.call(param.dcm, params)
init <- do.call(init.dcm, inits)
control <- do.call(control.dcm, controls)
dcm_fit <- dcm(param, init, control)

# ICM
param <- do.call(param.icm, params)
init <- do.call(init.icm, inits)
control <- do.call(control.icm, controls)
icm_fit <- icm(param, init, control)

plot(
  dcm_fit,
```

```
  alpha = 0.75, lwd = 4,
  ylab = "Número", xlab = "Tempo"
)
plot(
  icm_fit,
  qnts = FALSE, sim.lines = FALSE, add = TRUE,
  mean.lty = 2, legend = FALSE
)
fluxogram <- grViz("
digraph boxes_and_circles {
  # a 'graph' statement
  graph [overlap = true, fontsize = 10]

  # several 'node' statements
  node [shape = box, fontname = Courier]
  ergm; stergm; netest; netsim; infect; progress

  node [shape = none, fontname = Helvetica, fixedsize = true, width = 0.9]

  ergm -> netest
  stergm -> netest
  'Grafo\nObservado' -> netest
  'Especificação\ndo STERGM' -> netest
  netest -> 'Ajuste\ndo STERGM'
  'Ajuste\ndo STERGM' -> netsim
  infect -> netsim
  progress -> netsim
  'Parâmetros\nSEIQHRF' -> netsim
  netsim -> 'Simulações\nepidemiológicas'
}
")
fluxogram
infect <- function(dat, at) {

  ## Attributes ##
  active <- get_attr(dat, "active")
  status <- get_attr(dat, "status")

  ## TODO ##
  if (at == 2) {
    infTime <- rep(NA, length(active))
    infTime[which(status == "i")] <- 1
    dat <- set_attr(dat, "infTime", infTime)
```

```
} else {
  infTime <- get_attr(dat, "infTime")
}

## Parameters ##
inf.pars <- get_param(dat, "inf.pars")
prog.pars <- get_param(dat, "prog.pars")
sum.pars <- get_param(dat, "sum.pars")

## Find infected nodes ##
infectedStatus <- sum.pars$infected.status
idsInf <- which(active == 1 & status %in% infectedStatus)
nActive <- sum(active == 1)
nElig <- length(idsInf)

## Initialize default incidence at 0 ##
nInf <- 0

## If any infected nodes, proceed with transmission ##
if (nElig > 0 && nElig < nActive) {

  ## Look up discordant edgelist ##
  infectiveStatus <- sum.pars$infective.status
  del <- discord_edgelist(dat, at, infectiveStatus)

  ## If any discordant pairs, proceed ##
  if (!is.null(del)) {

    # Set parameters on discordant edgelist data frame
    del <- merge(del, inf.pars, by = "from")

    # Stochastic transmission process
    transmit <- rbinom(nrow(del), 1, del$final.prob)

    # Keep rows where transmission occurred
    del <- del[transmit == 1, ]

    # Look up new ids if any transmissions occurred
    idsNewInf <- unique(del$sus)
    nInf <- length(idsNewInf)

    # Set new attributes for those newly infected
    if (nInf > 0) {
```

```
        status[idsNewInf] <- "e"
        infTime[idsNewInf] <- at
        dat <- set_attr(dat, "status", status)
        dat <- set_attr(dat, "infTime", infTime)
      }
    }
  }

  ## Save summary statistic for S->E flow
  dat <- set_epi(dat, "se.flow", at, nInf)

  return(dat)
}
progress <- function(dat, at) {

  ## Attributes ##
  active <- get_attr(dat, "active")
  status <- get_attr(dat, "status")
  previousStatus <- status

  ## Parameters ##
  progPars <- get_param(dat, "prog.pars")
  sum.pars <- get_param(dat, "sum.pars")

  ## X to Y progression process ##
  progRates <- sum.pars$prog.rates
  probProg <- progRates[status]
  probProg[is.na(probProg)] <- 0
  isProg <- active & rbinom(length(status), 1, probProg)
  noProg <- all(isProg == FALSE)

  if (noProg) {
    status <- status
  } else {
    nextStates <- sum.pars$next.states
    nextProbs <- sum.pars$next.probs
    statusProg <- status[isProg == 1]
    status[isProg == 1] <- mapply(
      sample,
      x = nextStates[statusProg],
      size = 1,
      prob = nextProbs[statusProg]
    )
```

```
  }

  ## Write out updated status attribute ##
  dat <- set_attr(dat, "status", status)

  ## Save flows ##
  uniqueFlowNames <- sum.pars$flow.names
  vecFlows <- paste0(previousStatus, status, ".flow")[isProg]
  progFlows <- table(vecFlows)
  progFlowNames <- names(progFlows)
  for (flowName in uniqueFlowNames) {
    value <- if (flowName %in% progFlowNames) progFlows[flowName] else 0
    dat <- set_epi(dat, flowName, at, value)
  }

  ## Save nums ##
  uniqueNumNames <- sum.pars$num.names
  progNums <- table(status[active == 1])
  names(progNums) <- paste0(names(progNums), ".num")
  for (numName in uniqueNumNames) {
    value <- if (numName %in% names(progNums)) progNums[numName] else 0
    dat <- set_epi(dat, numName, at, value)
  }

  return(dat)
}
data(faux.magnolia.high)
summary_magnolia <- summary(faux.magnolia.high)
library(ergm)
plot_magnolia <- function(network, vertex_size = 0) {
  set.seed(2)
  plot(
    network,
    vertex.cex = vertex_size,
    vertex.col = COLORS[get.node.attr(faux.magnolia.high, "Grade") - 5]
  )
}

data(faux.magnolia.high)
par(mar = rep(0, 4), mfrow = c(1, 2))
plot_magnolia(faux.magnolia.high, 0)
plot_magnolia(faux.magnolia.high, 1)
plot_attribute_distribution <- function(attribute) {
```

```
  tab <- table(get.node.attr(faux.magnolia.high, attribute))
  barplot(prop.table(tab), names.arg = names(tab), main = attribute)
}

par(mfrow = c(2, 2))
invisible(lapply(
  c("Grade", "Race", "Sex"),
  plot_attribute_distribution
))
magnolia ~
  edges +
  nodematch("Grade", diff = TRUE) +
  nodematch("Race", diff = TRUE) +
  nodematch("Sex", diff = FALSE) +
  absdiff("Grade") +
  gwesp(0.25, fixed = TRUE)
library(EpiModel)

magnolia_path <- file.path("models", "magnolia.rds")
if (USE_CACHE && file.exists(magnolia_path)) {
  magnolia <- readRDS(magnolia_path)
} else {
  formation <-
    ~
    edges +
    nodematch("Grade", diff = T) +
    nodematch("Race", diff = T) +
    nodematch("Sex", diff = F) +
    absdiff("Grade") +
    gwesp(0.25, fixed = T)
  coef_diss <- dissolution_coefs(
    dissolution = ~ offset(edges),
    duration = 50
  )
  coef_diss$coef.crude <- -Inf
  magnolia_fit <- ergm(
    faux.magnolia.high ~
      edges + nodematch("Grade", diff=T) + nodematch("Race", diff=T) +
      nodematch("Sex",diff=F) + absdiff("Grade") + gwesp(0.25,fixed=T),
    burnin = 10000, interval = 1000, MCMCsamplesize = 2500, maxit = 25,
    control = control.ergm(steplength = 0.25, MCMLE.maxit = 100)
  )
  magnolia <- netest(
```

```
    faux.magnolia.high,
    formation = formation,
    target.stats = NULL,
    coef.diss = coef_diss,
    set.control.ergm = control.ergm(
      MCMC.burnin = 10000,
      MCMC.interval = 1000,
      MCMC.samplesize = 2500,
      MCMLE.steplength = 1,
      MCMLE.maxit = 1
    )
  )
  magnolia$fit <- magnolia_fit

  saveRDS(magnolia, magnolia_path)
}
library(data.table)
library(kableExtra)

coefs <- coef(magnolia$fit)
coeftab <- data.table(coef = coefs)
cols <- c("Estatística", "Fator", "Nível", "V4")
coeftab[, (cols) := tstrsplit(names(coefs), "[.]")]
coeftab[
  Estatística %in% c("edges", "gwesp"),
  `:=`(Fator = NA, Nível = NA)
]
coeftab[, Coeficiente := round(coef, 2)]
coeftab[, OR := round(exp(coef), 2)]

kbl(
  coeftab[, .(Estatística, Fator, Nível, Coeficiente, OR)],
  caption = "Coeficientes do ajuste Magnolia.",
  booktabs = TRUE
)
magnolia_sim <- simulate(
  magnolia$fit,
  nsim = 1,
  burnin = 100000000,
  constraint = "ConstantEdges"
)

par(mar = rep(0, 4), mfrow = c(1, 2))
```

```
plot_magnolia(magnolia_sim, 0)
plot_magnolia(magnolia_sim, 1)
data(faux.magnolia.high)
A <- as.matrix(faux.magnolia.high)
degrees <- rowSums(A)
barplot(prop.table(table(degrees)))
mean(degrees)
hist(degrees)
# library(parallel)
#
# n_cores <- detectCores()
# eff_cores <- if (n_cores > 4) 4 else n_cores
# dx <- netdx(
#   magnolia,
#   nsims = eff_cores,
#   nsteps = 365,
#   # set.control.ergm = control.simulate.ergm(
#   #   MCMC.burnin = 1e5,
#   #   MCMC.interval = 1000,
#   #   MCMLE.maxit = 1000,
#   #   MCMLE.steplength = 1
#   # ),
#   ncores = eff_cores
# )
# plot(dx, plots.joined = FALSE, qnts.alpha = 0.8)
# Number of steps
nsteps <- 170
# Number of simulations
nsims <- 30
# Initial conditions
init <- init.net(e.num = 20, i.num = 10)
# Possible infections in the SEIQHRF model
inf.pars <- data.frame(
  from = c("e", "i", "q"),
  to = c("s", "s", "s"),
  act.rate = c(10, 10, 2.5),
  inf.prob = c(0.02, 0.05, 0.02)
)
inf.pars$final.prob <- 1 - (1 - inf.pars$inf.prob)^inf.pars$act.rate

inf.tab <- inf.pars
inf.tab$from <- toupper(inf.tab$from)
inf.tab$to <- toupper(inf.tab$to)
```

```
inf.tab$final.prob <- round(inf.tab$final.prob, 2)
names(inf.tab) <- c(
  "De", "Para", "Encontros/dia",
  "P(Infecção)", "P(Encontro e infecção)"
)
kbl(
  inf.tab,
  caption = "Parâmetros de infecção do Cenário A.",
  booktabs = TRUE
)
# Progression parameters
prog.pars <- data.frame(
  from = c('e', 'e', 'i', 'i', 'i', 'q', 'q', 'h', 'h'),
  to =   c('i', 'r', 'q', 'h', 'r', 'h', 'r', 'r', 'f'),
  rate = c(1/10, 1/20, 1/30, 1/30, 1/20, 1/30, 1/20, 1/15, 1/50)
)

prog.tab <- prog.pars
prog.tab$from <- toupper(prog.tab$from)
prog.tab$to <- toupper(prog.tab$to)
prog.tab$rate <- round(prog.tab$rate, 2)
names(prog.tab) <- c("De", "Para", "Transições/dia")
kbl(
  prog.tab,
  caption = "Parâmetros de infecção do Cenário A.",
  booktabs = TRUE
)
source("R/module-fx.R")

# Summary parameters
states <- c(inf.pars$from, inf.pars$to, prog.pars$from, prog.pars$to)
unique.states <- unique(states)
sum.pars <- with(
  prog.pars,
  list(
    num.names = paste(unique.states, "num", sep = "."),
    flow.names = paste0(from, to, ".flow"),
    prog.rates = tapply(rate, from, function(x) 1 - prod(1 - x)),
    next.states = tapply(to, from, list),
    next.probs = tapply(rate, from, function(x) list(x / sum(x))),
    infective.status = unique(inf.pars$from),
    infected.status = unique(from)
  )
```

```
)
# Param list
param <- param.net(
  prog.pars = prog.pars,
  inf.pars = inf.pars,
  sum.pars = sum.pars
)
# Control settings
control <- control.net(
  type = NULL,
  nsteps = nsteps,
  nsims = nsims,
  ncores = 1,
  infection.FUN = infect,
  progress.FUN = progress,
  verbose = FALSE
)
# Run the network model simulation with netsim
sim_path <- "models/epi-sim-A.rds"
if (file.exists(sim_path)) {
  sim_A <- readRDS(sim_path)
} else {
  sim_A <- netsim(magnolia, param, init, control)
  saveRDS(sim_A, sim_path)
}
set.seed(1995)
par(mar = rep(0.1, 4), mfrow = c(2, 2))
ats <- c(1, 25, 50, 100)
invisible(lapply(
  ats,
  function(at) plot(
    sim_A,
    type = "network",
    at = at,
    col.status = T,
    main = at,
    vertex.cex = 1.5
  )
))
plot(
  sim_A,
  popfrac = TRUE,
  xlab = "Tempo",
```

```
  ylab = "Prevalência Média",
  targ.col = COLORS
)
inf.pars$act.rate <- inf.pars$act.rate * 0.3
inf.pars$final.prob <- 1 - (1 - inf.pars$inf.prob)^inf.pars$act.rate

inf.tab <- inf.pars
inf.tab$from <- toupper(inf.tab$from)
inf.tab$to <- toupper(inf.tab$to)
inf.tab$final.prob <- round(inf.tab$final.prob, 2)
names(inf.tab) <- c(
  "De", "Para", "Encontros/dia",
  "P(Infecção)", "P(Encontro e infecção)"
)
kbl(
  inf.tab,
  caption = "Parâmetros de infecção do Cenário B.",
  booktabs = TRUE
)
param$inf.pars <- inf.pars
# Run the network model simulation with netsim
sim_path <- "models/epi-sim-B.rds"
if (file.exists(sim_path)) {
  sim_B <- readRDS(sim_path)
} else {
  sim_B <- netsim(magnolia, param, init, control)
  saveRDS(sim_B, sim_path)
}
plot(
  sim_B,
  popfrac = TRUE,
  xlab = "Tempo",
  ylab = "Prevalência Média",
  targ.col = COLORS
)
Sys.info()
```

```
##                                                 sysname
##                                                 "Linux"
##                                                 release
##                                    "5.11.0-27-generic"
##                                                 version
## "#29~20.04.1-Ubuntu SMP Wed Aug 11 15:58:17 UTC 2021"
```

```
##                                 nodename
##                                     "pc"
##                                  machine
##                                "x86_64"
##                                    login
##                                  "luis"
##                                     user
##                                  "luis"
##                               effective_user
##                                  "luis"
```