Modelo epidemiológico com compartimentos SEIQHRF em redes: implementação e inferência via MCMC

Brasil

Modelo epidemiológico com compartimentos SEIQHRF em redes: implementação e inferência via MCMC

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

55 p., il. (algumas color.) ; 30 cm.
Modelo epidemiológico com compartimentos SEIQHRF em redes: implementação e inferência via MCMC/ . – 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: implementação e inferência via MCMC

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, pelas conversas sobre ciência;
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.

2 Metodologia

Modelos epidemiológicos são baseados na compartimentalização de indivíduos de uma população. (KEELING; EAMES, 2005) Normalmente, os compartimentos representam estados da doença. Esses modelos descrevem a proporção de indivíduos de uma população em cada estado ao longo do tempo.

2.1 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 **??** ilustra a dinâmica entre os compartimentos SIR.

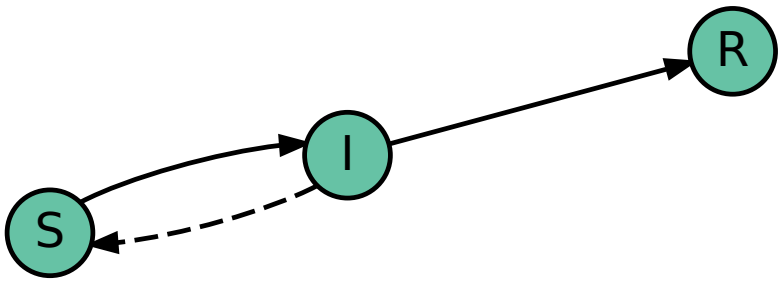


Figura 1 – SIR. Linhas pontilhadas são infecções e linhas sólidas são progressões.

2.2 Modelos determinísticos e estocásticos

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 pode ser expressado 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 e γ é a taxa de recuperação da doença, e $S + I + R = 1$. Esses modelos podem não possuem soluções analíticas exatas, e podem ser ajustados de forma determinística (aproximada) ou estocástica (**??**).

O pacote `EpiModel` (JENNESS; GOODREAU; MORRIS, 2018) da linguagem de programação R (JENNESS; GOODREAU; MORRIS, 2018) possui uma API para ajustar modelos epidemiológicos. No `EpiModel`, modelos determinísticos são chamados de *Deterministic Contact Models* (DCMs), e modelos estocásticos são chamados de *Invidual Contact Models* (ICMs).

Usando o `EpiModel`, fizemos o ajuste de dois modelos SIR. Um dos ajustes é DCM, e o outro ICM, com parâmetros idênticos. O tamanho da população é $N = 1000$, $beta = 0.16$ e $\gamma = 0.02$. No tempo 0, $S = 0.9$ e $I = 0.1$. A Figura 2 ilustra as progressões estimadas pelos ajustes em questão.

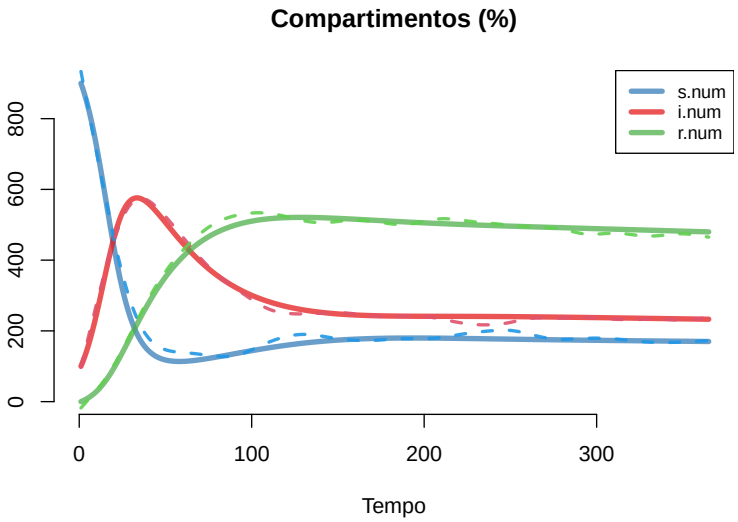


Figura 2 – Comparação entre ajustes determinístico (DCM, linhas sólidas) e estocástico (ICM, linhas tracejadas).

2.3 Modelos em redes

Quando o pressuposto de ... beta pode ser um vetor... subpopulação espacial, classe, etc.

Muitas vezes, temos conhecimento de relações entre indivíduos, que condicionam as probabilidades de contato entre os pares. Modelos em redes...

Não há soluções analíticas para essa classe de modelos, que são ajustados numericamente por meio de algoritmos de Markov Chain Monte Carlo (**??**)...

... (ANDERSSON; BRITTON, 2000)

2.4 SEIQHRF

Muitos conjuntos de estados foram propostos SIDARTHE (GIORDANO et al., 2020)...

O SEIQHRF (CHURCHES, 2020), ilustrado na Figura 3, adiciona quatro estados ao SIR:

- **Exposed**: infectado e assintomático. Pode infectar o suscetível;
- **Quarantined**: infectado e sintomático. Pode infectar o suscetível, com menor probabilidade de encontro que os demais estados infectados;
- **Hospitalized**: infectado e sintomático. Não encontra suscetíveis e tem maior probabilidade de morte;
- **Fatality**: morte.

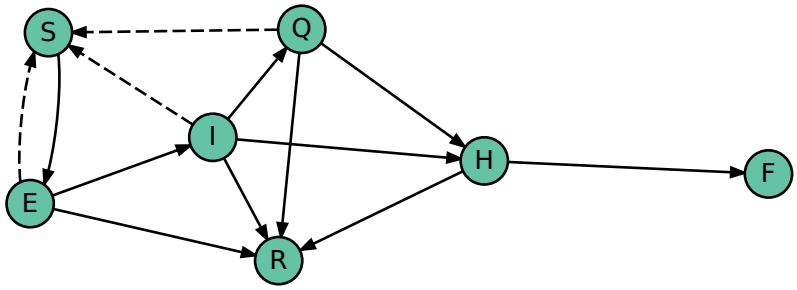


Figura 3 – SEIQHRF. Linhas pontilhadas são infecções e linhas sólidas são progressões.

2.5 Exponential random graph models

Formalmente, um grafo aleatório $Y \in \mathcal{Y}$ consiste em um conjunto de n vértices e m arestas $\{Y_{ij} : i = 1, \ldots, n; j = 1, \ldots, n\}$, no qual $Y_{ij} = 1$ se os vértices (i, j) são conectados e $Y_{ij} = 0$ caso contrário.

Exponential random graph models (ERGMs) são modelos da família exponencial que descrevem grafos aleatórios. Um ERGM é definido como:

$$P(Y = y|\theta) = \frac{\exp(\theta^T s(y))}{c(\theta)}, \quad \forall y \in \mathcal{Y}$$

onde θ é um vetor de parâmetros, $s(y)$ é um vetor de estatísticas suficientes e $c(\theta)$ é uma constante normalizadora. As estatísticas suficientes podem ser funções da rede, ou de atributos dos vértices.

ERGMs são utilizados na modelagem de redes sociais (EXPONENTIAL..., 2012).

3 Implementação

O objetivo principal deste trabalho foi implementar um modelo epidemiológico em redes, com os compartimentos $SEIQHRF$. Esse objetivo foi alcançado por meio de três ferramentas: a linguagem de programação R (R Core Team, 2021); o pacote `ergm` (HANDCOCK et al., 2020) para o ajuste dos ERGMs e suas respectivas simulações; e o pacote `EpiModel` (JENNESS; GOODREAU; MORRIS, 2018) para o ajuste dos modelos epidemiológicos em redes e suas respectivas simulações.

O uso do `ergm` para o ajuste dos ERGMs é trivial, e não será detalhado nesta seção. Contudo, a implementação do SEQIHRF no `EpiModel` envolveu a programação de duas funções customizadas, descrevendo o processo gerador dos dados do $SEIQHRF$ em redes. Essa implementação pode ser encontrada no Github... (**??**)

3.1 netsim

A *Application Programming Interface* (API) para modelos em redes do `EpiModel` tem uma funnção principal, a `netsim`. Simplificada para o contexto deste trabalho, a função é descrita pelo pseudocódigo abaixo:

3.2 Parâmetros

3.3 Módulos

3.3.1 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)
}
```

3.3.2 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

...

4.1 Ajuste do ERGM

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. Os vértices representam alunos, e as arestas as amizades entre alunos. Existem 1461 alunos, e cada aluno tem três atributos: `Grade` (turma), `Race` (raça) e `Sex` (sexo).

A figura 4 ilustra o grafo em questão.

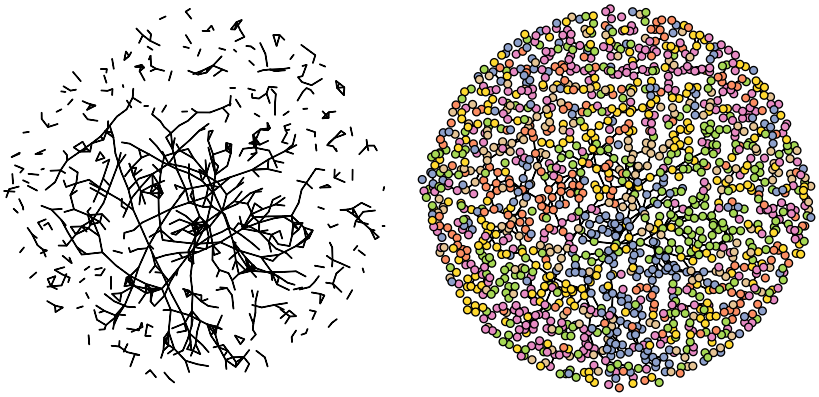


Figura 4 – Faux Magnolia High. À esquerda, apenas as arestas do grafo. À direita, os vértices coloridos de acordo com a turma do aluno.

A figura 5 ilustra a proporção dos níveis de cada atributo.

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

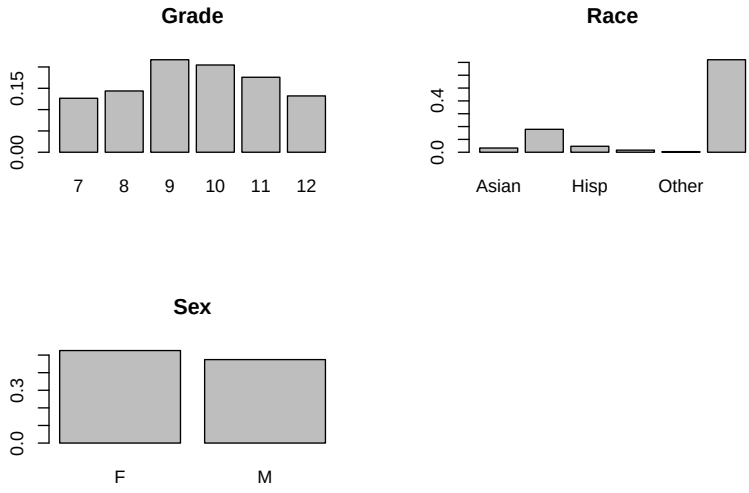


Figura 5 – Proporções dos níveis de cada atributo do Faux Mangolia High.

> 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)

Para simplificarmos a notação, discutiremos a especificação do modelo na sintaxe de `formula` do R. Nessa sintaxe, `dados ~ s_1 + s_2 + ... + s_n`, onde `dados` é o grafo Faux Magnolia High, e cada `s_i` é uma ou mais estatísticas suficientes de $s(y)$. Na sintaxe de fórmula, 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)
```

As estatísticas calculadas pelas funções são:

- `edges`: numero 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)`: ...

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 6 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.

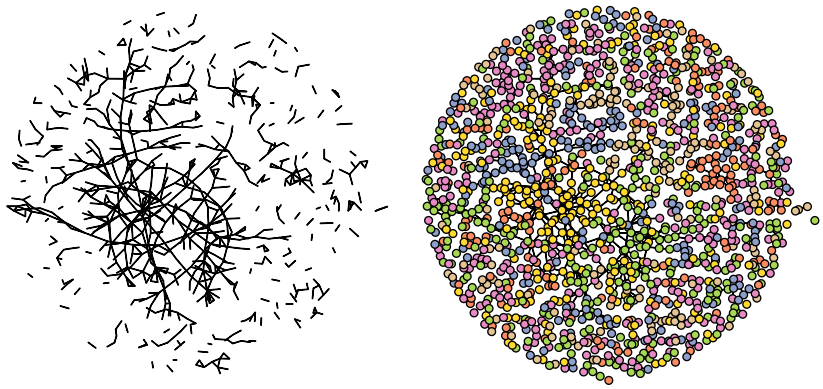


Figura 6 – Simulação gerada com nosso ajuste do Faux Magnolia High. À esquerda, apenas as arestas do grafo. À direita, os vértices coloridos de acordo com a turma do aluno.

A Figura 7 mostra uma comparação entre a distribuição de graus dos dois grafos.

Tabela 2 – Parâmetros de infecção das simulações.

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

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

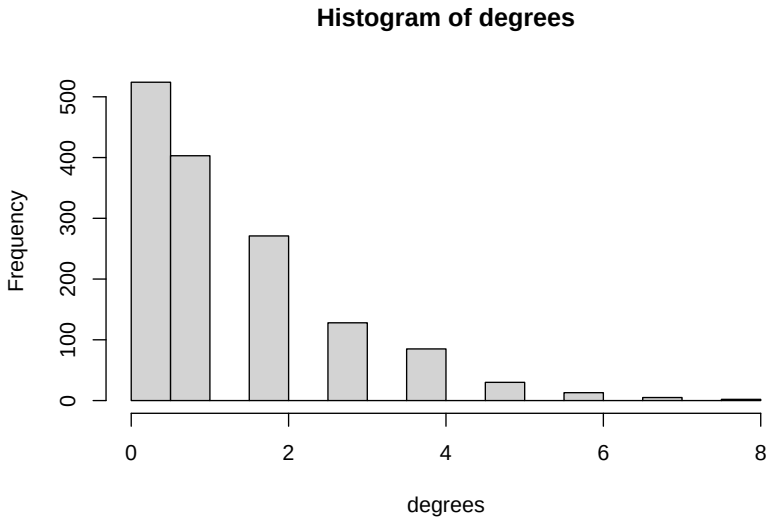


Figura 7 – Comparação entre as distribuições de grau do grafo original e da simulação.

4.2 Simulações epidemiológicas

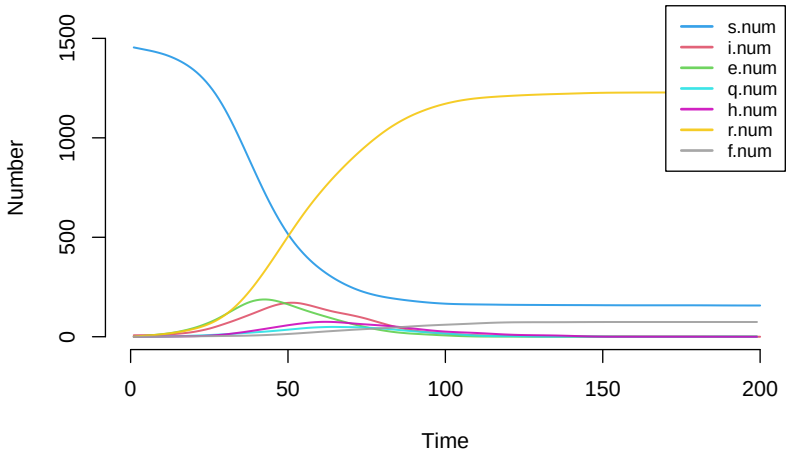
Antes de simular os modelos epidemiológicos, fixamos alguns parâmetros para as simulações. A simulação relizada pelo `EpiModel` funciona com unidades arbitrárias de tempo. No contexto do nosso exemplo, o tempo será contado em dias, e as simulações são limitadas a 200 dias. Essa escolha foi feita porque 200 é o número de dias letivos no Brasil. Por consequência, os parâmetros de taxa de encontro terão uma frequência diária.

Os parâmetros fixos de infecção estão na tabela **??**.

4.2.1 Simulação exploratória

Ver se os parâmetros fazem sentido. Ver que o código está correto. Explorar possibilidades de visualização.

O pacote `EpiModel` (JENNESS; GOODREAU; MORRIS, 2018) da ferramentas



4.2.2 Teste de hipoteses

Teste de hipoteses com quarentena vs. sem quarentena. Teste de hipoteses muitos hospitais vs. poucos hospitais

5 Conclusão

TODO...

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.

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 2 vezes nas páginas 23 e 27.

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 27.

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 27.

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 3 vezes nas páginas 29, 33 e 34.

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 3 vezes nas páginas 26, 29 e 36.

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 na página 25.

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.

R Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria, 2021. Disponível em: <https://www.R-project.org/>. Citado na página 29.

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 33.

Apêndices

APÊNDICE A – Código

```
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)
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)

# # ICM
# param <- param.dcm(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)
# init <- init.dcm(s.num = 90, i.num = 10, r.num = 0)
# control <- control.dcm(type = "SIR", nsteps = 365)
# det <- dcm(param, init, control)
# param <- param.icm(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)
# init <- init.icm(s.num = 90, i.num = 10, r.num = 0)
# control <- control.icm(type = "SIR", nsteps = 365, nsims = 10)
# sim <- icm(param, init, control)
```

```
plot(
  dcm_fit,
  alpha = 0.75, lwd = 4,
  main = "Compartimentos (%)",
  ylab = "", xlab = "Tempo"
)
plot(
  icm_fit,
  qnts = FALSE, sim.lines = FALSE, add = TRUE,
  mean.lty = 2, legend = FALSE
)
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)
netsim <- function(ergm_fit, parameters, initial_state, control) {
  simulations <- 1:control$n_simulations

  for (s in simulations) {

  }
  steps <- 1:control$n_steps

}
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
))
# TODO add edge histogram?
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"
)
```

```r
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)
mean(degrees)
hist(degrees)
nsteps <- 200
# Model parameters
# 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 das simulações.",
  booktabs = TRUE
)
# 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)

# Possible progressions in the SEIQHRF model
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)
)
# 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
)
# Initial conditions
init <- init.net(e.num = 10, i.num = 10)
source("R/module-fx.R")

# Control settings
control <- control.net(
  type = NULL,
  nsteps = nsteps,
  nsims = 1,
```

```
  ncores = 1,
  infection.FUN = infect,
  progress.FUN = progress,
  verbose = FALSE
)
# Run the network model simulation with netsim
sim <- netsim(magnolia, param, init, control)
plot(sim)
```