Grupo Responsável pelo código

Thiago Costa, Fábio Antero, Félix Garcia

Caracterização do problema de contagio por vírus (matéria do G1)

Santana do Jacaré (-20.903345, -45.128259 ; -20.896826, -45.131666)

Não tem espaços verticais e o tranporte coletivo tem pouca relevancia

Metodologia (comparar 3 políticas: isolamento, minimização de interação social ou combinação) Superfície de risco Movimentação de forasteiros As pessoas se deslocam majoritariamente em direção ao centro da cidade

http://rmarkdown.rstudio.com.

Código de simulação de contágio

```
## Carregando os pacotes necessários
options(java.parameters = "-Xmx6g")
options(max.print=1000000)

source('/Users/fgarcia/Programas/R_Simulation/T3/Parametros.R', encoding = 'UTF-8', echo=TRUE)
```

```
##
## > get.p.forasteiros.entrar <- function() {
## +     return(0.05)
## + }
##
## > get.p.forasteiros.permenecer <- function() {
## +     return(0.2)
## + }
##
## > get.p.forasteiros.sair <- function() {
## +     return(0.05)
## + }
```

```
source('/Users/fgarcia/Programas/R_Simulation/T3/Funcoes_R.R', encoding = 'UTF-8', echo=TRUE)
```

```
##
## > if (!require("pacman")) install.packages("pacman")
```

```
## Loading required package: pacman
```

```
##
## > pacman::p_load("tidyverse", "MASS", "plyr", "data.table",
## +     "gganimate", "Rfast", "gifski", "readr", "tibble", "plotrix")
##
## > atualizar.direcao <- function(dados) {
## +     ind.casa <- which(dados$casa == TRUE)
```

```
## +     ind.sair <- ind.casa[sample(c(TRUE, FALSE), length(ind.casa .... [TRUNCATED]
##
## > atualizar.posicao <- function(dados) {
## +     delta <- tamanho.passo * cbind(sin(dados$direc), cos(dados$direc))
## +     delta[dados$direc == 0, ] <- 0 .... [TRUNCATED]
##
## > atualizar.casa <- function(dados) {
## +     tam.dados <- length(dados[, "ind"])
## +     tmp <- dados[dados$resid == 1, c("lat", "long")]
## +     d <- (tmp .... [TRUNCATED]
##
## > atualizar.risco <- function(dados, tempo, superficie) {
## +     tmp <- dados %>% filter(status == 1)
## +     if (empty(tmp)) {
## +         risco <- superf .... [TRUNCATED]
##
## > atualizar.status <- function(dados, tempo, superficie,
## +     linha.grid) {
## +     tam.dados <- length(dados[, "ind"])
## +     risco <- superficie[, te .... [TRUNCATED]
##
## > adicionar.forasteiros <- function(dados, tempo, superficie) {
## +     dados.interno <- dados
## +     insereForasteiros <- sample(c(TRUE, FALSE), prob =  .... [TRUNCATED]
##
## > remover.forasteiros <- function(dados) {
## +     dados.interno <- dados
## +     if (max.pop.forasteiros > 0) {
## +         excluirForasteiros <- sample(c( .... [TRUNCATED]
##
## > executar_simulacao <- function(n.pop, TT) {
## +     for (tempo in 2:TT) {
## +         dados[[tempo]] <- dados[[tempo - 1]]
## +         dados[[tempo]]$dire .... [TRUNCATED]
```

```r
set.seed(12345) # Definindo a semente

### Definindo os parâmetros do modelo
TT <- 24*7*4*5 # Número de passos (tempo 1 hora)
n.pop <- 4600 # Tamanho da população (divisível por 8)

### Criando a lista que irá receber os dados simulados
dados <- vector(TT, mode="list")
names(dados) <- 1:(TT)

### Definindo os locais de residência

santana <- read.csv("/Users/fgarcia/Programas/R_Simulation/T3/santana.csv", sep = ";") #residencias e p

### Definindo locais de deslocamento
locais <- santana %>%
  filter(CLAS_SUB != "RE1" & CLAS_SUB != "RE2") %>%
  mutate(ind= seq.int(1,249),
         lat = POINT_Y,
```

```
        long = POINT_X)
locais <- locais[,c(9:11)]

# separando unidades residenciais
residencias <- santana %>%
  filter(CLAS_SUB == "RE1" | CLAS_SUB == "RE2") %>% #filtra unidade residenciais
  mutate(ind = seq.int(1,1890), #cria novas variáveis
         lat = POINT_Y,
         long = POINT_X)
residencias <- residencias[,c(9:11)] #separa colunas que serão utilizadas


### Parametros da cidade
max.pop.forasteiros <- 10 # Tamanho máximo da população de forasterios que entrará ou sairá da cidade n
#-20.901358
#-45.1292318
#centro <- c(lat=-20.902805, long=-45.127847)
centro <- c(lat=20.901358, long=-45.129231)
lat.extremos <- c(lat.min=-20.909100, lat.max=-20.897918)
long.extremos <- c(long.min=-45.134000, long.max=-45.122818)

### Parametros de deslocamento
p.sair.casa <- .05 # Probabilsidade de uma pessoa em casa sair no minuto em questão
p.voltar.casa <- .05 # Probabilidade de uma pessoa parada fora de casa entrar em movimento de retorno
# A probabilidade abaixo induz uma distribuição geométrica para o "número de movimentos até parar"
p.parar.fora <- .2 # Probabilidade de uma pessoa em movimento fora de casa parar
# Desvio padrão do ruído que desloca a direção de movimentação em relação ao centro da cidade
angulo.sigma <- 1  # Quanto maior o valor menor é a chance dos moradores irem para o centro
# .0001 é aproximadamente 30 metros
tamanho.passo <- .0002 # Velocidade do deslocamento por unidade de tempo

### Parametros de disseminação do vírus
taxa.decaimento <- 1-(1/60) #Taxa .99 com que a contaminação do ambiente é preservada no tempo seguinte
raio.risco <- .05 # .05 (o que equivale a aproximadamente 9 metros) parâmetro que define o risco de con
# n" - (n/60                                        )*1.852 = distância em metros
# n' - n*1.852 = distância em metros
# n° - n*60*1.852 = distância em metros

escala.risco <- 1E-6  # valor que relaciona o risco (nível de contaminação) à probabilidade de infecção
n.grid <- 100  # Número de pontos no grid de estimação da superficie de risco
p.recuperar <- ((1+.0001)^12)-1 # Probabilidade de uma pessoa infectada se recuperar em um instante de
p.hospt <- ((1+.0019)^(1/(24)))-1 # Probabilidade de uma pessoa infectada ser hospitalizada.
p.uti <- ((1+.0295)^(1/(24)))-1 # Probabilidade de uma pessoa hospitalizada necessitar de UTI.
p.obito <- ((1+.0013)^(1/(24)))-1 # Probabilidade de uma pessoa em UTI vir a óbito.

# Induzir pessoas a morar juntas
casas <- data.frame(
  ind=sample(dim(residencias)[[1]], n.pop, replace=TRUE)) %>%
  merge(residencias, casas, by.x = "ind", by.y = "ind")

### Criando a população no tempo 1
desloc <- sample(dim(locais)[[1]],n.pop/2, replace = TRUE)
```

```r
dados[[1]] <- data.frame(
  ind=1:n.pop,
  lat=c(casas$lat[1:(n.pop/2)],
        locais[desloc,2]),
  long=c(casas$long[1:(n.pop/2)],
         locais[desloc,3]),
  casa=c(rep(1L, n.pop/2), rep(0L, n.pop/2)),
  status=c(rep(1L, 1), rep(0L, n.pop-1)), # 0=saudável, 1=doente, 2=recuperado, 3=hospitalizado
  # 4=uti, 5=óbito
  # trocar a linha abaixo para mover as pessoas como na função atualizar.direcao
  direc=rep(0, n.pop),
  resid=rep(1, n.pop)
)

indices <- (n.pop/8*7):n.pop
angulo.aux <- atan2(centro[1] - dados[[1]]$lat[indices],
                    centro[2] - dados[[1]]$long[indices])
dados[[1]]$direc[indices] <- rnorm(length(indices), angulo.aux, angulo.sigma)


### Criando a superficie de risco
superficie <- matrix(nr=n.grid^2, nc=2+TT) # Superficie de risco
colnames(superficie) <- c("long", "lat", paste0("densidade_", 1:TT))
tmp <- dados[[1]] %>%
  filter(status==1)
aux <- kde2d(tmp$long, tmp$lat, h=rep(raio.risco, 2), n=n.grid,
             lims=c(range(long.extremos), range(lat.extremos)))
superficie[, 1:2] <- unlist(with(aux, expand.grid(x, y)))
superficie[, 3] <- as.vector(aux$z)
grid.lat <- unique(superficie[, 1])
grid.long <- unique(superficie[, 2])

### Definindo o índice da linha correspondente ao ponto do grid mais próximo à posição corrente
linha.grid <- numeric(n.pop)
for(ind in 1:n.pop) {
  indices.lat.grid <- which.min(abs(dados[[1]][ind, "lat"] - grid.lat))
  indices.lat <- (0:(n.grid-1))*n.grid + indices.lat.grid
  indice.long <- which.min(abs(dados[[1]][ind, "long"] - superficie[indices.lat, 2]))
  linha.grid[ind] <- indices.lat[indice.long]
}

dados <- executar_simulacao(n.pop, TT)
```

```
## Tempo:  3360  - Fim !
```

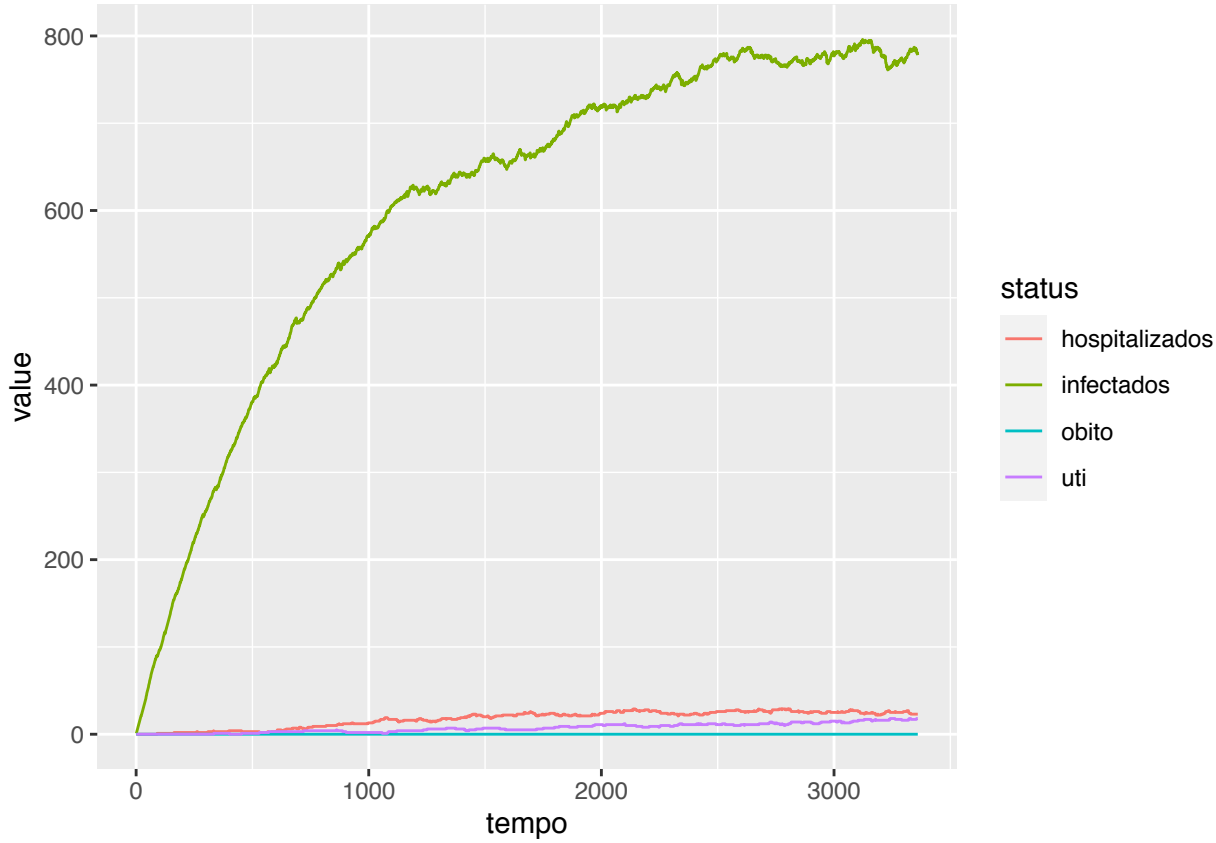
```r
### Sumarizando os resultados
#### Tempo da simulação ####
####     24*7*4*5       ####


dados.plot <- as.data.frame(data.table::rbindlist(dados, idcol="tempo"))
class(dados.plot$tempo) <- "integer"
dados.plot %>%
  group_by(tempo) %>%
```

```
  dplyr::summarise(#saudaveis=sum(status==0),
                   #infectados=sum(status==1),
                   #recuperados=sum(status==2)) %>%
                   hospitalizados=sum(status==3),
                   uti=sum(status==4),
                   obito=sum(status==5)) %>%
  gather(., status, value, -tempo) %>%
  ggplot(aes(tempo, value, color=status)) +
  geom_line()
```


```
# Exibindo os dados sumarizados
dados.plot %>%
  group_by(tempo>6000) %>%
  dplyr::summarise(saudaveis=sum(status==0),
                   infectados=sum(status==1),
                   recuperados=sum(status==2),
                   hospitalizados=sum(status==3),
                   uti=sum(status==4),
                   obito=sum(status==5))
```

```
## # A tibble: 1 x 7
##   `tempo > 6000` saudaveis infectados recuperados hospitalizados   uti obito
##   <lgl>              <int>      <int>       <int>          <int> <int> <int>
## 1 FALSE            9816301    2037949     3536659          59111 24482     0
```

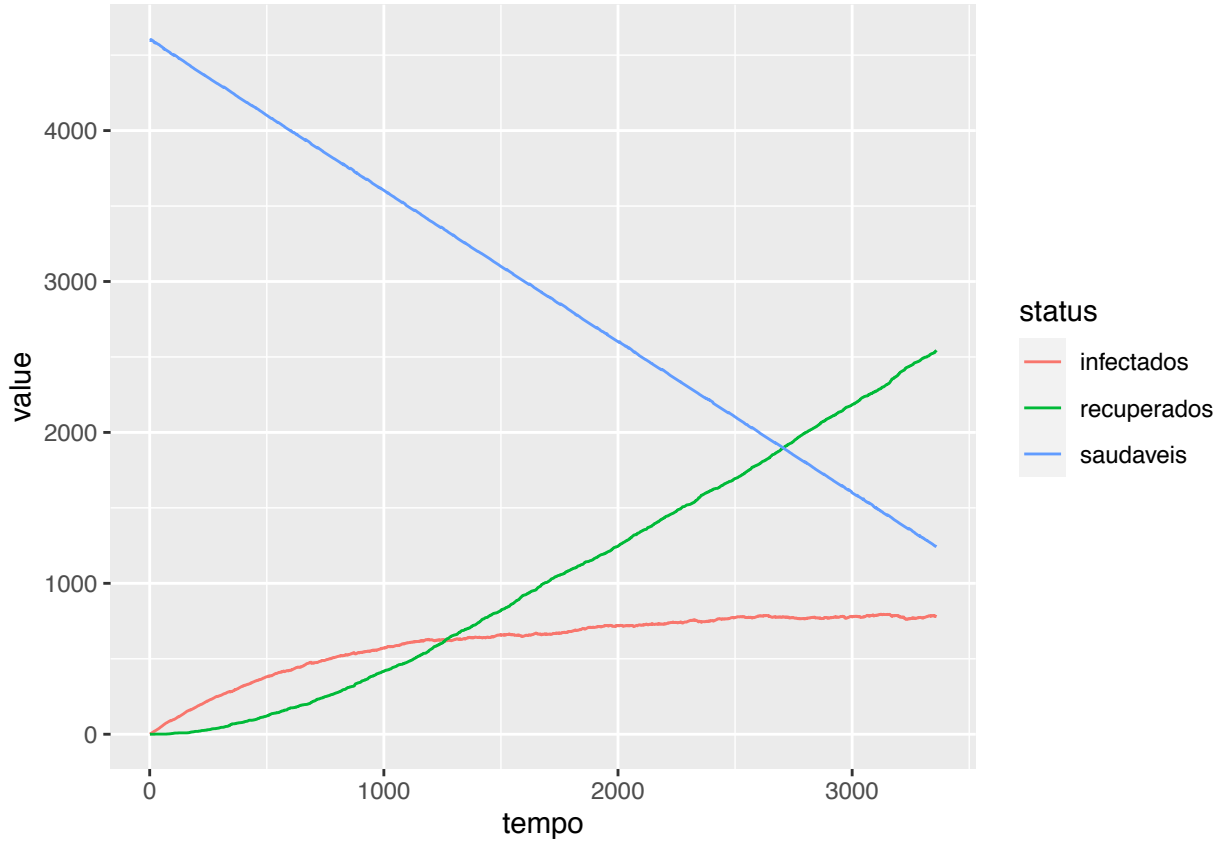
```
  dplyr::summarise(#saudaveis=sum(status==0),
                   infectados=sum(status==1),
                   #recuperados=sum(status==2),
                   hospitalizados=sum(status==3),
                   uti=sum(status==4),
                   obito=sum(status==5)) %>%
  gather(., status, value, -tempo) %>%
  ggplot(aes(tempo, value, color=status)) +
  geom_line()
```




```
# Exibindo os dados sumarizados
dados.plot %>%
  group_by(tempo>6000) %>%
  dplyr::summarise(saudaveis=sum(status==0),
                   infectados=sum(status==1),
                   recuperados=sum(status==2),
                   hospitalizados=sum(status==3),
                   uti=sum(status==4),
                   obito=sum(status==5))
```

```
## # A tibble: 1 x 7
##   `tempo > 6000` saudaveis infectados recuperados hospitalizados   uti obito
##   <lgl>              <int>      <int>       <int>          <int> <int> <int>
## 1 FALSE            9816301    2037949     3536659          59111 24482     0
```

```
  dplyr::summarise(saudaveis=sum(status==0),
                   infectados=sum(status==1),
                   recuperados=sum(status==2)) %>%
                   #hospitalizados=sum(status==3),
                   #uti=sum(status==4),
                   #obito=sum(status==5)) %>%
  gather(., status, value, -tempo) %>%
  ggplot(aes(tempo, value, color=status)) +
  geom_line()
```




```
# Exibindo os dados sumarizados
dados.plot %>%
  group_by(tempo>6000) %>%
  dplyr::summarise(saudaveis=sum(status==0),
                   infectados=sum(status==1),
                   recuperados=sum(status==2),
                   hospitalizados=sum(status==3),
                   uti=sum(status==4),
                   obito=sum(status==5))
```

```
## # A tibble: 1 x 7
##   `tempo > 6000` saudaveis infectados recuperados hospitalizados   uti obito
##   <lgl>              <int>      <int>       <int>          <int> <int> <int>
## 1 FALSE            9816301    2037949     3536659          59111 24482     0
```