Figures for the minimal model for household-based testing and tracing in epidemics

Greg Huber   Mason Kamb   Kyle Kawagoe   Lucy M. Li
Aaron McGeever   Jonathan Miller   Boris Veytsman
Dan Zigmond

October 12, 2020

```
opts_chunk$set(
    dev='tikz',
    cache=T
)
options(tikzDefaultEngine='luatex')
library(reticulate)
library(deSolve)
library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse
1.3.0 --
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts()
--
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(ggthemes)
theme_set(theme_bw())
```

1 Introduction

Here we generate the figures for the paper.

2 Minimal testing rate

The minimal testing rate to stop the epidemics is determined from the the equation $\lambda = 0$, where λ is the leading eigenvalue of the epidemics equations (see the notation in the main paper):

$$\lambda = \frac{1}{2}\left\{\sqrt{[\alpha-\beta-(H-1)\kappa]^2+4H\alpha\beta}-\alpha+\beta-(H+1)\kappa\right\}-\gamma. \quad (1)$$

We set the parameters to sweep:

```
beta0 <- c(0.05, 0.1, 0.3, 0.6)
alpha0 <- c(0.6,1,2,5)
H0 <- 1:16
gamma0 <- c(0.06,  0.125, 0.25, 0.3)
data <- expand_grid(beta=beta0, alpha=alpha0, H=H0, gamma=gamma0)
data

## # A tibble: 1,024 x 4
##      beta alpha     H gamma
##     <dbl> <dbl> <int> <dbl>
##  1  0.05   0.6     1 0.06
##  2  0.05   0.6     1 0.125
##  3  0.05   0.6     1 0.25
##  4  0.05   0.6     1 0.3
##  5  0.05   0.6     2 0.06
##  6  0.05   0.6     2 0.125
##  7  0.05   0.6     2 0.25
##  8  0.05   0.6     2 0.3
##  9  0.05   0.6     3 0.06
## 10  0.05   0.6     3 0.125
## # ... with 1,014 more rows
```

Numerical solution:

```
kap <- function(alpha, beta, H, gamma) {
    f <- uniroot(function(kappa)
        0.5*(sqrt((alpha-beta-(H-1)*kappa)^2 +
                  4*H*alpha*beta)-alpha +
             beta - (H+1)*kappa) - gamma,
        lower=-10, upper=10)
    if(f$root>0) {
        return(f$root)
    } else {
        return(0)
    }
}
```

```
kap(0.6, 0.06, 6, 0.125)

## [1] 0.03089822

data <- data %>%
    mutate(kappa=Vectorize(kap)(alpha, beta,
        H, gamma))
data

## # A tibble: 1,024 x 5
##      beta alpha     H gamma   kappa
##     <dbl> <dbl> <int> <dbl>   <dbl>
##  1  0.05   0.6     1 0.06  0
##  2  0.05   0.6     1 0.125 0
##  3  0.05   0.6     1 0.25  0
##  4  0.05   0.6     1 0.3   0
##  5  0.05   0.6     2 0.06  0.0178
##  6  0.05   0.6     2 0.125 0
##  7  0.05   0.6     2 0.25  0
##  8  0.05   0.6     2 0.3   0
##  9  0.05   0.6     3 0.06  0.0271
## 10  0.05   0.6     3 0.125 0.00264
## # ... with 1,014 more rows
```

The plot is shown on Figure 1.

```
alpha_lab <- function(val) {
    paste0("$\\alpha=\\SI{", val, "}{days^{-1}}$")
}
beta_lab <- function(val) {
    paste0("$\\beta=\\SI{", val, "}{days^{-1}}$")
}
H_lab <- function(val) {
    paste0("$H=", val, "$")
}
ggplot(data) + geom_line(aes(H, kappa,
                             color=as_factor(gamma))) +
        facet_grid(alpha~beta,
                   labeller=labeller(.cols=beta_lab,
                                     .rows=alpha_lab)) +
        xlab("$H$") + ylab("$\\kappa$") +
    labs(color='$\\gamma, \\si{days^{-1}}$')
```

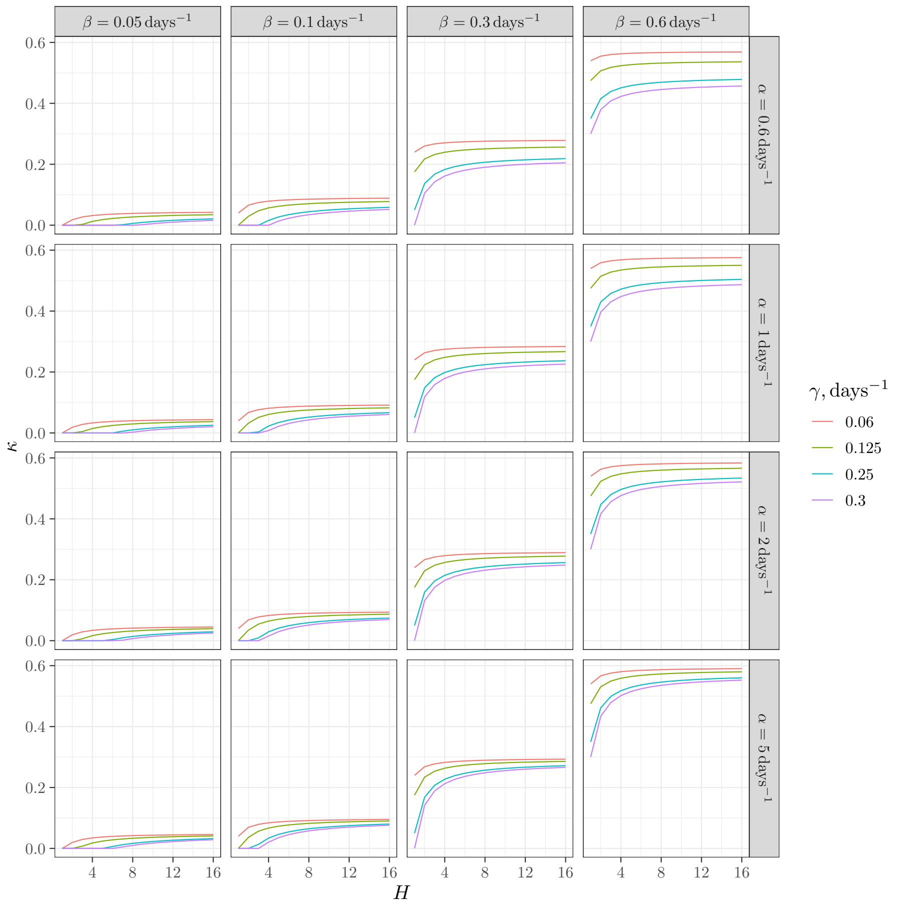

Figure 1: The minimal testing rate to stop the epidemics

3 Epidemics progression

The simulation data are in `../data/param_sweep.npy.gz`. We read it as shown below. Variables for the var column:

S: susceptible households

F: unquarantined single infection households

F_q: quarantined single infection households

G: unquarantined fully infected households

G_q: quarantined fully infected households

R: recovered

I: total unquarantined infected individuals

I_t: total infected individuals (including those in quarantine)

```r
gz <- import('gzip')
np <- import('numpy')
simdata <- np$load(gz$open("../data/param_sweep.npy.gz"))
beta0 <- c(0.05, 0.1, 0.3, 0.6)
alpha0 <- c(0.6,1,2,5)
H0 <- c(1,2,4,8,16)
rep0 <- c(1,10)
var0 <- c('S', 'F', 'Fq', 'G', 'Gq', 'R', 'I', 'It')
t0 <- seq(0, 15, by=0.1)
df <- expand_grid(beta=1:length(beta0),
                  alpha=1:length(alpha0),
                  H=1:length(H0),
                  rep=1:length(rep0),
                  var=1:length(var0),
                  t=1:length(t0))
fill <-
    Vectorize(
        function(beta, alpha, H, rep, var, t)
            simdata[beta, alpha, H, rep, var, t])
df <- df %>% mutate(value=fill(beta, alpha, H, rep, var, t)) %>%
    mutate(beta=beta0[beta],
           alpha=alpha0[alpha],
           H=H0[H],
           rep=rep0[rep],
           var=var0[var],
           t=t0[t])
df
```

```
## # A tibble: 193,280 x 7
##     beta alpha     H   rep var       t  value
##    <dbl> <dbl> <dbl> <dbl> <chr> <dbl>  <dbl>
##  1  0.05   0.6     1     1 S       0   199980
##  2  0.05   0.6     1     1 S       0.1 199980
##  3  0.05   0.6     1     1 S       0.2 199980
##  4  0.05   0.6     1     1 S       0.3 199980
##  5  0.05   0.6     1     1 S       0.4 199980
##  6  0.05   0.6     1     1 S       0.5 199980
##  7  0.05   0.6     1     1 S       0.6 199980
##  8  0.05   0.6     1     1 S       0.7 199980
##  9  0.05   0.6     1     1 S       0.8 199980
## 10  0.05   0.6     1     1 S       0.9 199980
## # ... with 193,270 more rows
```

Now mean field. We use the following equations

$$\begin{aligned}\frac{dF}{dt} &= \beta\frac{SH}{N}F + \beta\frac{SH}{N}GH - \alpha F - \gamma F - \kappa F\\ \frac{dG}{dt} &= \alpha F - \gamma G - H\kappa G\\ \frac{dS}{dt} &= -\beta\frac{SH}{N}F - \beta\frac{SH}{N}GH\end{aligned} \tag{2}$$

Numerical solution:

```r
eqs <- function(t, state, params) {
    with(as.list(c(state, params)), {
        dFdt <- beta*S*H/N*(F+G*H)-alpha*F-gamma*F - kappa*F
        dGdt <- alpha*F-gamma*G - H*kappa*G
        dSdt <- -beta*S*H/N*(F+G*H)
        list(c(dFdt, dGdt, dSdt))
    })}
prediction_params <- expand_grid(beta=beta0, alpha=alpha0,
                                 H=H0)
prediction <-
    bind_rows(lapply(1:nrow(prediction_params),
                     function(i) {
                         alpha <- prediction_params[[i, 'alpha']]
                         beta <- prediction_params[[i, 'beta']]
                         H <- prediction_params[[i, 'H']]
                         N <- 200000
                         params <-
                             list(alpha=alpha,
                                  beta=beta,
                                  H=H,
                                  gamma = 0.125,
                                  kappa = 0.06,
                                  N = N)
                         initState <- c(F=20, G=0, S=(N-20)/H)
                         times=seq(0,15, by=0.01)
                         out <- ode(initState, times, eqs,
                                    params)
                         as_tibble(as.data.frame(out)) %>%
                             mutate(alpha=alpha, beta=beta, H=H,
                                    I=F+H*G)
                     }))
prediction

## # A tibble: 120,080 x 8
##     time     F     G       S alpha  beta     H     I
##    <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>
##  1  0     20   0     199980    0.6  0.05     1  20
##  2  0.01  19.9 0.119 199980.   0.6  0.05     1  20.0
##  3  0.02  19.7 0.238 199980.   0.6  0.05     1  19.9
##  4  0.03  19.6 0.355 199980.   0.6  0.05     1  19.9
##  5  0.04  19.4 0.471 199980.   0.6  0.05     1  19.9
##  6  0.05  19.3 0.586 199980.   0.6  0.05     1  19.9
##  7  0.06  19.1 0.700 199980.   0.6  0.05     1  19.8
##  8  0.07  19.0 0.813 199980.   0.6  0.05     1  19.8
##  9  0.08  18.9 0.925 199980.   0.6  0.05     1  19.8
## 10  0.09  18.7 1.04  199980.   0.6  0.05     1  19.8
## # ... with 120,070 more rows
```

The plot is shown on Figure 2.

```
ggplot(df %>% filter(var=='I' & floor(t)==t) %>%
       group_by(beta, alpha, t, H) %>%
       summarise(I=median(value), delta=1.58*IQR(value)/sqrt(10)) %>%
       filter(I-delta>=1)) +
    geom_errorbar(aes(x=t, y=I,  ymin=I-delta,
                      ymax=I+delta, color=as_factor(H))) +
    geom_line(data=prediction %>% filter(I>=1),
              aes(time, I, color=as_factor(H))) +
    scale_y_log10() +
    facet_grid(alpha~beta,
               labeller=labeller(.cols=beta_lab,
                                 .rows=alpha_lab)) +
        xlab("Time, days") + ylab("Infected") +
    labs(color='$H$')

    ## `summarise()` regrouping output by 'beta', 'alpha', 't'
                (override with `.groups` argument)
```

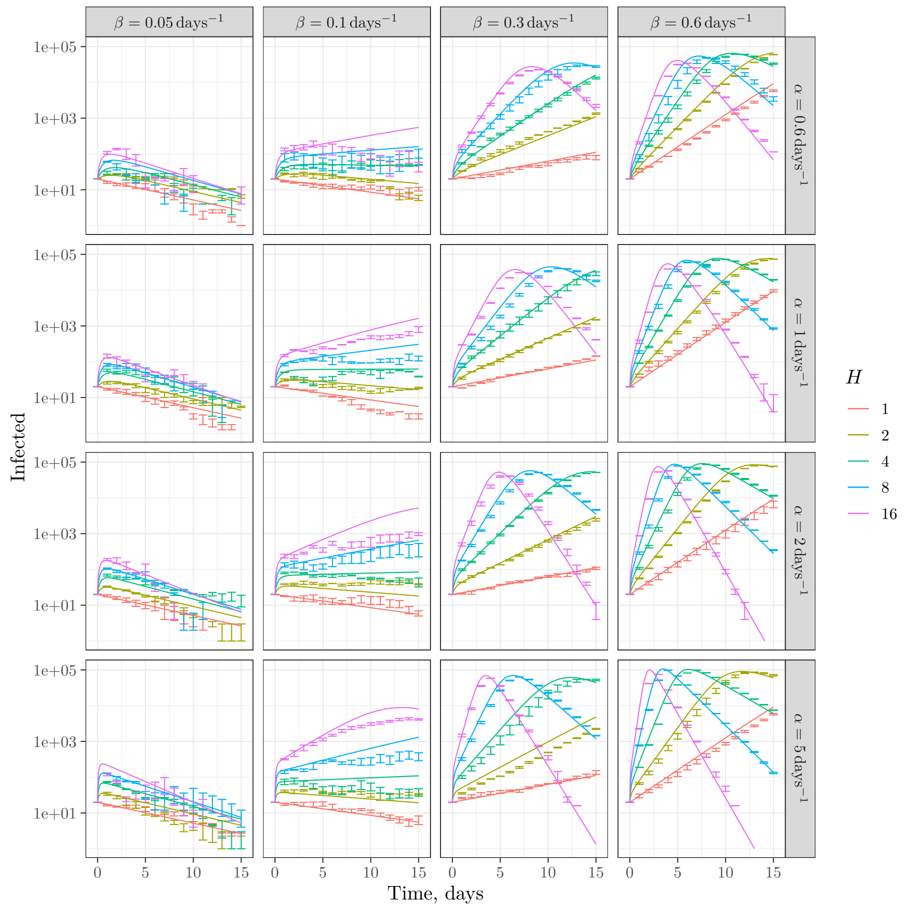

Figure 2: Predicted numbers of quarantined and non-quarantined infecteds. Error bars are simulations, lines are mean field

4 Heatmaps

The data for the heatmap is provided by the file `../data/heatmap_data.npy`.

```r
gz <- import('gzip')
np <- import('numpy')
infecteds_data <- np$load(gz$open("../data/heatmap_data.npy.gz"))
str(infecteds_data)

##  num [1:20, 1:20, 1:20, 1:5, 1:5, 1:8] 2e+05 2e+05 2e+05 2e+05 2e+05 ...

beta0 <- seq(from=0.05, to=0.6, length.out=20)
alpha0 <- seq(from=0.6, to=5, length.out=20)
kappa0 <- seq(from=0, to=0.6, length.out=20)
H0 <- c(1,2,4,8,16)
var0 <- c('S', 'F', 'Fq', 'G', 'Gq', 'R', 'I', 'It')
rep0 <- 1:5
infecteds <- expand_grid(beta_ind=1:length(beta0),
                  alpha_ind=1:length(alpha0),
                  kappa_ind=1:length(kappa0),
                  H_ind=1:length(H0),
                  rep=1:length(rep0),
                  var_ind=1:length(var0),
                  )
fill <-
    Vectorize(
        function(beta_ind, alpha_ind, kappa_ind, H_ind, rep, var_ind)
            infecteds_data[beta_ind, alpha_ind,
                           kappa_ind, H_ind, rep, var_ind])
infecteds <- infecteds %>%
    mutate(value=fill(beta_ind, alpha_ind, kappa_ind, H_ind, rep, var_ind)) %>%
    mutate(beta=beta0[beta_ind],
           alpha=alpha0[alpha_ind],
           kappa=kappa0[kappa_ind],
           H=H0[H_ind]) %>%
        mutate(var=var0[var_ind]) %>%
    filter(var=='It' | var=='R') %>%
    pivot_wider(names_from=var, values_from=value) %>%
    group_by(beta_ind, alpha_ind, kappa_ind, H_ind) %>%
    summarise(total=sum(R, na.rm=T)/length(rep0)*H +
                  sum(It, na.rm=T)/length(rep0),
              alpha=first(alpha),
              beta=first(beta),
              kappa=first(kappa),
              H=first(H))
```

```
## `summarise()` regrouping output by 'beta_ind', 'alpha_ind', 'kappa_ind',
'H_ind' (override with `.groups` argument)

infecteds

## # A tibble: 400,000 x 9
## # Groups:   beta_ind, alpha_ind, kappa_ind, H_ind [40,000]
##    beta_ind alpha_ind kappa_ind H_ind total alpha  beta kappa     H
##       <int>     <int>     <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1        1         1         1     1  29.4   0.6  0.05     0     1
##  2        1         1         1     1  29.4   0.6  0.05     0     1
##  3        1         1         1     1  29.4   0.6  0.05     0     1
##  4        1         1         1     1  29.4   0.6  0.05     0     1
##  5        1         1         1     1  29.4   0.6  0.05     0     1
##  6        1         1         1     1  29.4   0.6  0.05     0     1
##  7        1         1         1     1  29.4   0.6  0.05     0     1
##  8        1         1         1     1  29.4   0.6  0.05     0     1
##  9        1         1         1     1  29.4   0.6  0.05     0     1
## 10        1         1         1     1  29.4   0.6  0.05     0     1
## # ... with 399,990 more rows
```

The plots are on Figures 3– 7.

```r
ggplot(infecteds %>% filter(kappa_ind==1)) +
    geom_tile(aes(alpha, beta, fill=total)) +
    facet_wrap(~H, labeller=labeller(H=H_lab))  +
    scale_fill_gradient(low="green", high="red",
                        limits=c(0,200000)) +
    xlab("$\\alpha, \\si{days^{-1}}$") +
    ylab("$\\beta, \\si{days^{-1}}$") +
    labs(fill="Total cases") +
    theme(legend.position=c(0.8, 0.2))
```

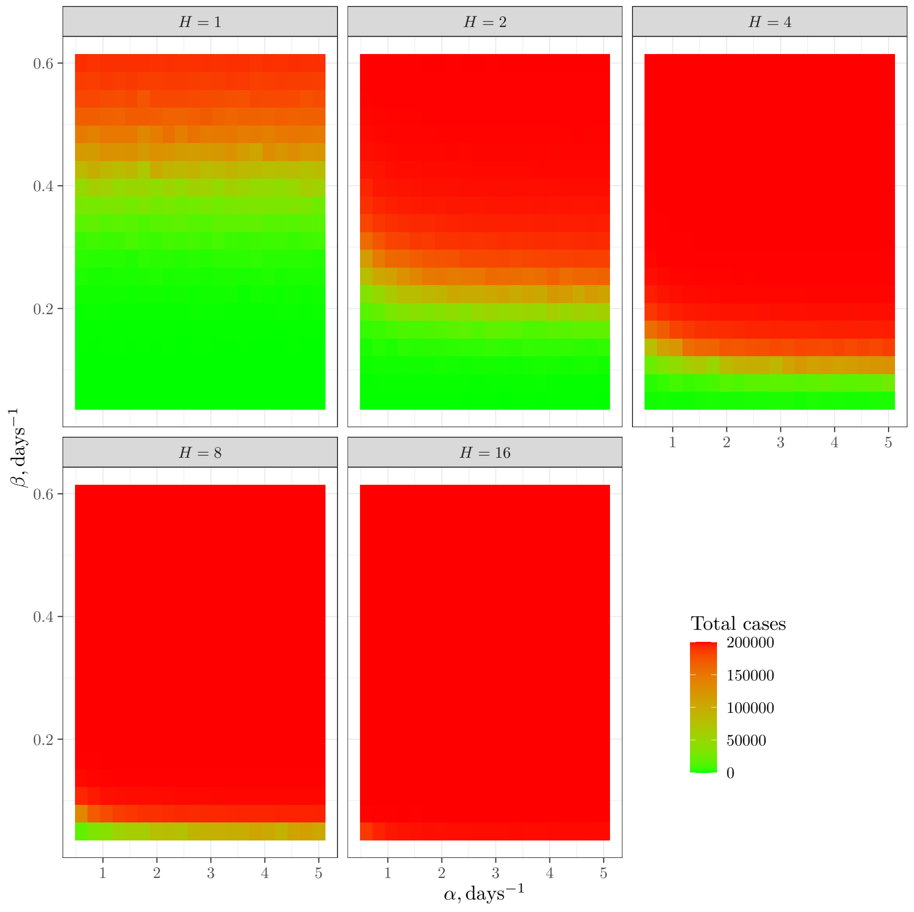

Figure 3: Total number of infections for $\kappa = 0\,\text{days}^{-1}$

```r
ggplot(infecteds %>% filter(kappa_ind==5)) +
    geom_tile(aes(alpha, beta, fill=total)) +
    facet_wrap(~H, labeller=labeller(H=H_lab))  +
    scale_fill_gradient(low="green", high="red",
                        limits=c(0,200000)) +
    xlab("$\\alpha, \\si{days^{-1}}$") +
    ylab("$\\beta, \\si{days^{-1}}$") +
    labs(fill="Total cases") +
    theme(legend.position=c(0.8, 0.2))
```

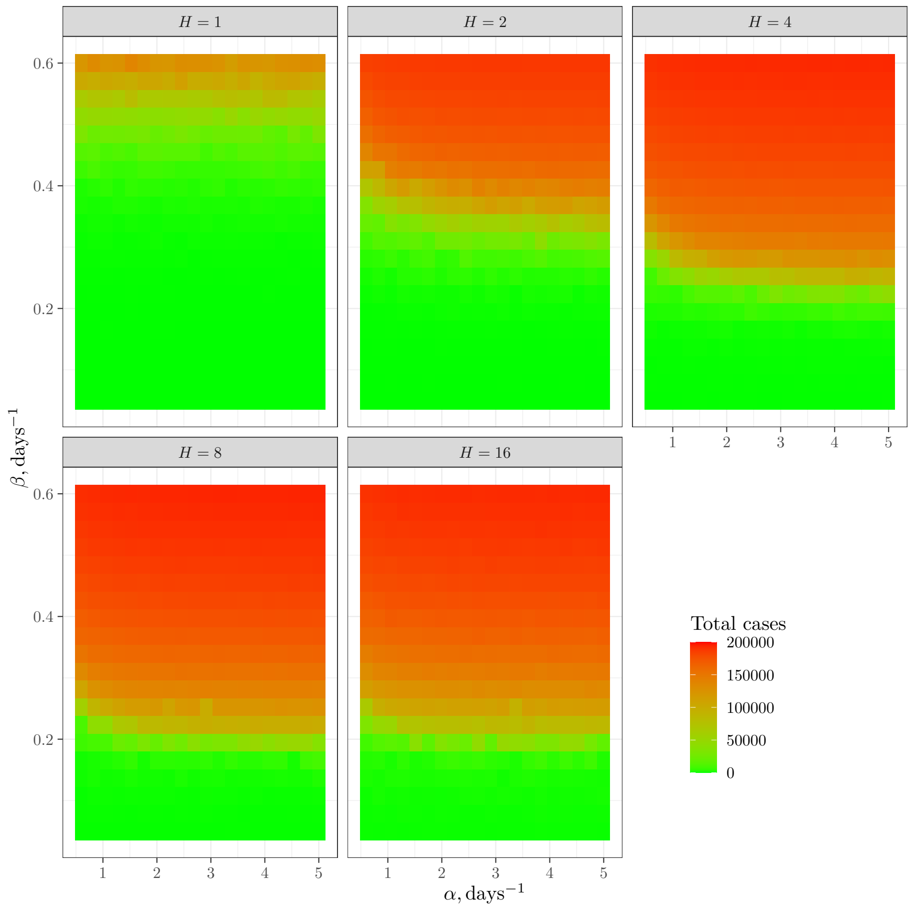

Figure 4: Total number of infections for $\kappa = 0.1263\,\text{days}^{-1}$

```r
ggplot(infecteds %>% filter(kappa_ind==8)) +
    geom_tile(aes(alpha, beta, fill=total)) +
    facet_wrap(~H, labeller=labeller(H=H_lab))  +
    scale_fill_gradient(low="green", high="red",
                        limits=c(0,200000)) +
    xlab("$\\alpha, \\si{days^{-1}}$") +
    ylab("$\\beta, \\si{days^{-1}}$") +
    labs(fill="Total cases") +
    theme(legend.position=c(0.8, 0.2))
```

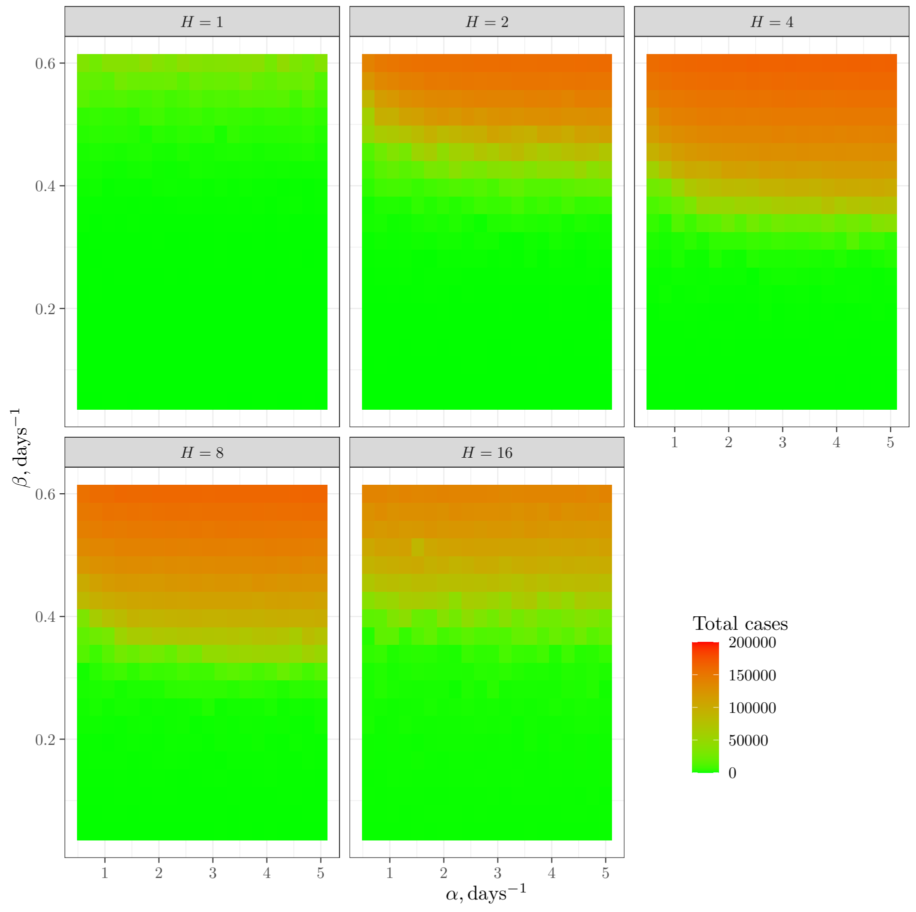

Figure 5: Total number of infections for $\kappa = 0.2211\,\mathrm{days}^{-1}$

```
ggplot(infecteds %>% filter(kappa_ind==12)) +
    geom_tile(aes(alpha, beta, fill=total)) +
    facet_wrap(~H, labeller=labeller(H=H_lab))  +
    scale_fill_gradient(low="green", high="red",
                        limits=c(0,200000)) +
    xlab("$\\alpha, \\si{days^{-1}}$") +
    ylab("$\\beta, \\si{days^{-1}}$") +
    labs(fill="Total cases") +
    theme(legend.position=c(0.8, 0.2))
```

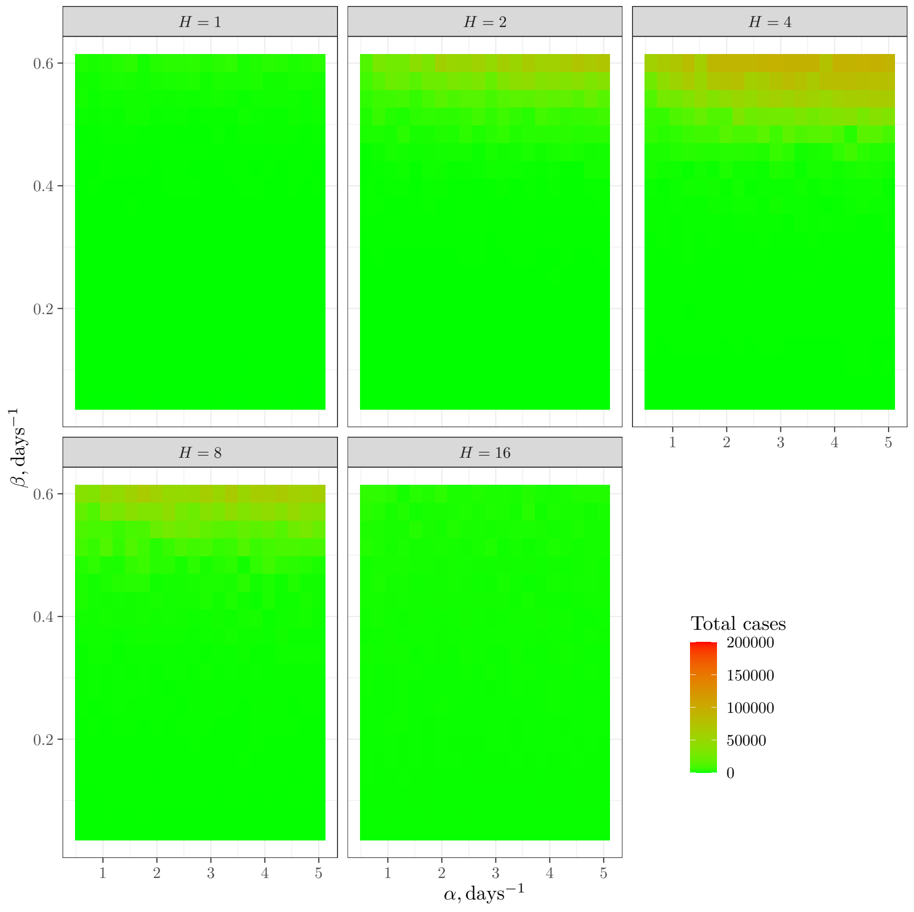

Figure 6: Total number of infections for $\kappa = 0.3474\,\text{days}^{-1}$

```
ggplot(infecteds %>% filter(kappa_ind==14)) +
    geom_tile(aes(alpha, beta, fill=total)) +
    facet_wrap(~H, labeller=labeller(H=H_lab))  +
    scale_fill_gradient(low="green", high="red",
                        limits=c(0,200000)) +
    xlab("$\\alpha, \\si{days^{-1}}$") +
    ylab("$\\beta, \\si{days^{-1}}$") +
    labs(fill="Total cases") +
    theme(legend.position=c(0.8, 0.2))
```

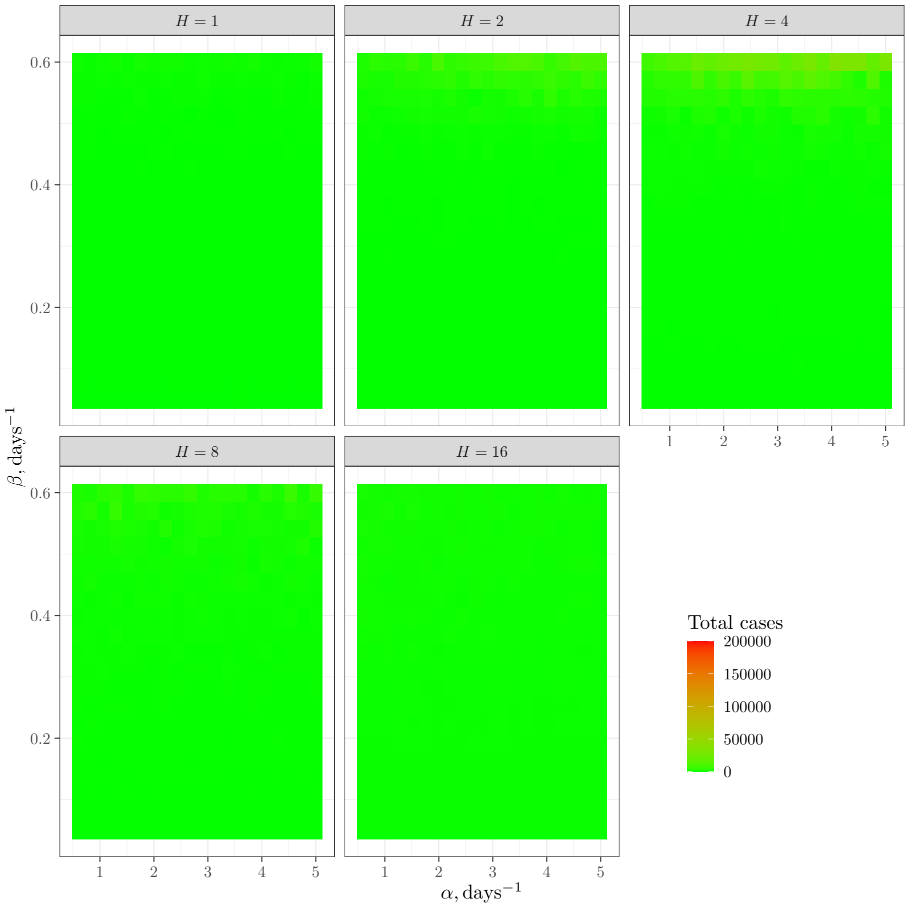

Figure 7: Total number of infections for $\kappa = 0.4105\,\text{days}^{-1}$

A Session information

```r
opts_chunk$set(
    dev='tikz',
    cache=T
)
options(tikzDefaultEngine='luatex')
library(reticulate)
library(deSolve)
library(tidyverse)
library(ggthemes)
theme_set(theme_bw())
```

```r
gsub("\\verb\\|([^\\|]*)\\|", "\\path{\\1}",
     toLatex(sessionInfo()))
```

- R version 4.0.2 (2020-06-22), `x86_64-apple-darwin19.5.0`
- Locale: `en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8`
- Running under: `macOSCatalina10.15.7`
- Matrix products: default
- BLAS/LAPACK: `/usr/local/Cellar/openblas/0.3.10_1/lib/libopenblasp-r0.3.10.dylib`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: deSolve 1.28, dplyr 1.0.2, forcats 0.5.0, Formula 1.2-3, ggplot2 3.3.2, ggthemes 4.2.0, Hmisc 4.4-1, knitr 1.30, lattice 0.20-41, lubridate 1.7.9, purrr 0.3.4, readr 1.4.0, reticulate 1.16, scales 1.1.1, sfsmisc 1.1-7, stringr 1.4.0, survival 3.2-7, tibble 3.0.3, tidyr 1.1.2, tidyverse 1.3.0
- Loaded via a namespace (and not attached): assertthat 0.2.1, backports 1.1.10, base64enc 0.1-3, blob 1.2.1, broom 0.7.1, cellranger 1.1.0, checkmate 2.0.0, cli 2.0.2, cluster 2.1.0, codetools 0.2-16, colorspace 1.4-1, compiler 4.0.2, crayon 1.3.4, data.table 1.13.0, DBI 1.1.0, dbplyr 1.4.4, digest 0.6.25, ellipsis 0.3.1, evaluate 0.14, fansi 0.4.1, farver 2.0.3, filehash 2.4-2, foreign 0.8-80, fs 1.5.0, generics 0.0.2, glue 1.4.2, grid 4.0.2, gridExtra 2.3, gtable 0.3.0, haven 2.3.1, highr 0.8, hms 0.5.3, htmlTable 2.1.0, htmltools 0.5.0, htmlwidgets 1.5.2, httr 1.4.2, jpeg 0.1-8.1, jsonlite 1.7.1, labeling 0.3, latticeExtra 0.6-29, lifecycle 0.2.0, magrittr 1.5, Matrix 1.2-18,

modelr 0.1.8, munsell 0.5.0, nnet 7.3-14, pillar 1.4.6, pkgconfig 2.0.3, png 0.1-7, R6 2.4.1, RColorBrewer 1.1-2, Rcpp 1.0.5, readxl 1.3.1, reprex 0.3.0, rlang 0.4.7, rpart 4.1-15, rstudioapi 0.11, rvest 0.3.6, splines 4.0.2, stringi 1.5.3, tidyselect 1.1.0, tikzDevice 0.12.3.1, tinytex 0.26, tools 4.0.2, vctrs 0.3.4, withr 2.3.0, xfun 0.18, xml2 1.3.2