Synchrony and Persistence of Recurrent Epidemics (Supplementary Material)

Group Name: The Infective Collective

Group Members: Aurora Basinski-Ferris, Michael Chong, Daniel Park, Daniel Presta

April 8, 2018 @ 23:00

1 Introduction

This supplement contains the necessary code to reproduce the data and graphs in the main paper. We also include additional discussion and explanation of some of our methods and results.

2 The Model

2.1 Construction of the model

Here, we provide a detailed derivation of the model. First, consider the following coupled SIR model consisting of n patches:

$$\begin{aligned}
\frac{dS_k}{dt} &= \mu N - \left(\sum_{j=1}^{n} \beta(t) m_{kj} I_j + \mu\right) S_k \\
\frac{dI_k}{dt} &= \left(\sum_{j=1}^{n} \beta(t) m_{kj} I_j + \mu\right) S_k - (\mu + \gamma) I_k \\
\frac{dR_k}{dt} &= \gamma I_k - \mu R_k
\end{aligned} \tag{1}$$

where S_k, I_k and R_k are numbers of susceptible, infected and recovered individuals in patch k, respectively. N is population size in every patch and is assumed to be constant. $\beta(t)$ is the transmission rate, γ is the per capita recovery rate and μ is the per capita death rate as well as birth rate. m_{kj} is the proportion of contacts from patch j that are dispersed to patch k. The dispersal matrix is then given by $M = [m_{kj}]$ and we have $\sum_{k=1}^{n} m_{kj} = 1$.

Note that a susceptible individual in patch i leaves the susceptible compartment at rate $\sum_{j=1}^{n} \beta(t) m_{ij} I_j + \mu$. We can view this quantity as hazard and obtain the probability that a

susceptible individual survives during a time interval $(t, t+\Delta t)$:

$$\exp\left(-\int_t^{t+\Delta t}\sum_{j=1}^{n}\beta(s)m_{kj}I_j(s)+\mu ds\right), \tag{2}$$

where for sufficiently small Δt, we can write

$$\int_t^{t+\Delta t}\sum_{j=1}^{n}\beta(s)m_{kj}I_j(s)+\mu ds \approx \left(\sum_{j=1}^{n}\beta(t)m_{kj}I_j(s)+\mu\right)\Delta t. \tag{3}$$

Rearranging, number of individuals that leave the susceptible compartment after Δ time step is given by

$$S_{k,\text{leave}}(t) = \left(1-\exp\left(-\left(\sum_{j=1}^{n}\beta(t)m_{kj}I_j(s)+\mu\right)\Delta t\right)\right)S_k(t) \tag{4}$$

Assuming that the transmission rate and number of infected individuals stays constant over the interval $(t, t+\Delta)$, the probability that a susceptible individual leaves the compartment due to infection is given by

$$\frac{\sum_{j=1}^{n}\beta(t)m_{kj}I_j(s)}{\sum_{j=1}^{n}\beta(t)m_{kj}I_j(s)+\mu} \tag{5}$$

Incidence (i.e., number of suseptible individuals that become infected during $(t, t+\Delta t)$ interval) is then given by the product of above probability and $S_{i,\text{leave}}(t)$. Similarly, an infected individual and a recovered individual suffer from constant hazard of $\gamma+\mu$ and μ, respectively. Since these rates are not time-dependent, these can each be translated into transition probabilities assuming exponentially distributed life time in each compartment, yielding

$$\begin{aligned} I_{k,\text{leave}}(t) &= (1-\exp(-(\gamma+\mu)\Delta t))I_k(t) \\ R_{k,\text{leave}}(t) &= (1-\exp(-\mu\Delta t))R_k(t) \end{aligned} \tag{6}$$

Then, the full discrete time model is given by

$$\begin{aligned} S_k(t+\Delta t) &= b_k(t)+S_k(t)-S_{k,\text{leave}}(t) \\ I_k(t+\Delta t) &= i_k(t)+I_k(t)-I_{k,\text{leave}}(t) \\ R_k(t+\Delta t) &= r_k(t)+R_k(t)-R_{k,\text{leave}}(t) \end{aligned} \tag{7}$$

where $b_k(t)$, $i_k(t)$, and $r_k(t)$ represent number of new susceptible, infected, and recovered individuals that are produced between the interval $(t, t+\Delta t)$:

$$\begin{aligned} i_k(t) &= \frac{\sum_{j=1}^{n}\beta(t)m_{ij}I_j(s)}{\sum_{j=1}^{n}\beta(t)m_{ij}I_j(s)+\mu}S_{k,\text{leave}}(t) \\ r_k(t) &= \frac{\gamma}{\gamma+\mu}R_{k,\text{leave}}(t) \\ b_k(t) &= S_{k,\text{leave}}(t)-i_k(t)+I_{k,\text{leave}}(t)-r_k(t) \end{aligned} \tag{8}$$

2.2 Time-dependent transmission rate

Since many recurrent epidemics are childhood diseases, we choose to look at a time-dependent transmission rate that is dependent on an elementary student's school year. From Bauch and Earn, we define the time-dependent transmission rate as

$$\beta(t) = \begin{cases} b_0(1 + 2(1 - p_s)b_1) & \text{school days} \\ b_0(1 - 2p_s b_1) & \text{non-school days} \end{cases}, \tag{9}$$

where b_0 is the mean transmission rate, $b_1 == 0.25$ is the amplitude of the term-time forcing, and $p_s = 0.7589$ is the approximate proportion of the year that falls in the school term. We define our school days and non-school days in the following manner (all dates inclusive, leap years not considered):

- Each day (including weekends) from January 7th to March 9th is considered a school day.
- Each day from March 10th to March 17th is a non-school day, since March break approximately occurs during this time of year.
- Each day (including weekends) from March 18th to June 30th is a school day.
- Each day from July 1st to September 3rd is a non-school day, as summer vacation occurs during this time.
- Each day from September 4th to December 22nd is a school day.
- Each day from December 23rd to January 6th is a non-school day (Christmas, New Years break).

We chose these days after examining past school calendars from the Toronto District School Board and making reasonable assumptions regarding the average school opening and school closing dates, as well as the average beginning and end dates of school breaks. Furthermore, we included weekends to account for contact-heavy children's extracurricular activities that may take place on weekends during the school year, and to make the code easier to implement (greatly reduce the number of "`if`" statements).

2.3 Load packages

Before we attempt any numerical analysis, we want to load all necessary pacakges first.

```
library(Rcpp)
```

```
library(deSolve)
library(tidyr)
library(dplyr)
library(ggplot2); theme_set(theme_bw(base_size = 12,
                                     base_family = "Times"))
library(gridExtra)
library(stringr)

if (.Platform$OS.type=="windows") {
    windowsFonts(Times=windowsFont("Times"))
}
```

2.4 Implementation of the model

Implementation of the model is done using the `Rcpp` package. The source code is too long to be displayed here. Interested readers should look at the source code `SIRmodel_npatch.cpp`. Before we perform any analysis, we have to load the model file:

```
sourceCpp("SIRmodel_npatch.cpp")

## Warning in normalizePath(file, winslash = "/"):  path[1]="SIRmodel_npatch.cpp":
No such file or directory
## Error in sourceCpp("SIRmodel_npatch.cpp"):  file not found:  'SIRmodel_npatch.cpp'
```

2.5 Parameters

Base parameters are stored as a list.

```
base.params <- list(
    R0=17,
    pop=1e6,
    b1=0.25,
    gamma=365/13,
    mu=0.02,
    dt=1/365,
    nsteps=365*100
)
```

Here, we deinfe a function named `initfun` that returns states at the endemic equilibrium assuming constant transmission rate.

```
initfun <- function(param,
                    n.patch=1,
                    round=FALSE) {
    with(param,{
        epsilon <- mu/(mu+gamma)

        ll <- list(
            S=rep(pop/R0, n.patch),
            I=rep(epsilon*(1-1/R0)*pop, n.patch)
        )

        if (round) ll <- lapply(ll, round)

        ll$R <- pop - ll$S - ll$I

        ll
    })
}
```

Base initial conditions are then equal to endemic equilibrium.

```
base.init <- initfun(base.params)
```

3 Bifurcation diagram

In order to make a bifurcation diagram, we vary $\mathcal{R}_0$ from 1 to 20 every 0.2 steps. Then, for each $\mathcal{R}_0$, 20 simulations are run from random initial conditions for 4000 years to remove any transient behaviours that might be present. Here, we check whether the simulation file is present. If not, simulations are run and saved as an `.rda` file. Note that this simulation takes approximately 20 hours. Thus, we load a saved file instead.

```
if (file.exists("bifurcation.rda")) {
```

```
    load("bifurcation.rda")
} else {
    nsim <- 20
    R0vec <- seq(1, 20, by=0.2)

    set.seed(101)
    init <- data.frame(
        S=seq(from=0, to=0.1, length.out=nsim) * base.params[["pop"]],
        I=seq(from=0, to=0.0001, length.out=nsim) * base.params[["pop"]]
    )

    init[] <- apply(init, 2, sample)
    init$R <- base.params[["pop"]] - (init$S + init$I)

    blist <- vector('list', length(R0vec))

    for (R in R0vec) {

        pp <- base.params
        pp[["R0"]] <- R
        pp[["nsteps"]] <- 365 * 4000

        blist[[which(R0vec==R)]] <- lapply(1:nsim, function(x){
            ii <- init[x,]
            df <- SIRmodel_npatch(pp, ii, matrix(1), term_time)
            prev <- tail(df$I[df$time%%1==0], 100)/pp[["pop"]]

            data.frame(
                prevalence=prev,
                R0=R
            )
        })
            save("blist", file="bifurcation.rda")
        }

    save("blist", file="bifurcation.rda")

    }
```

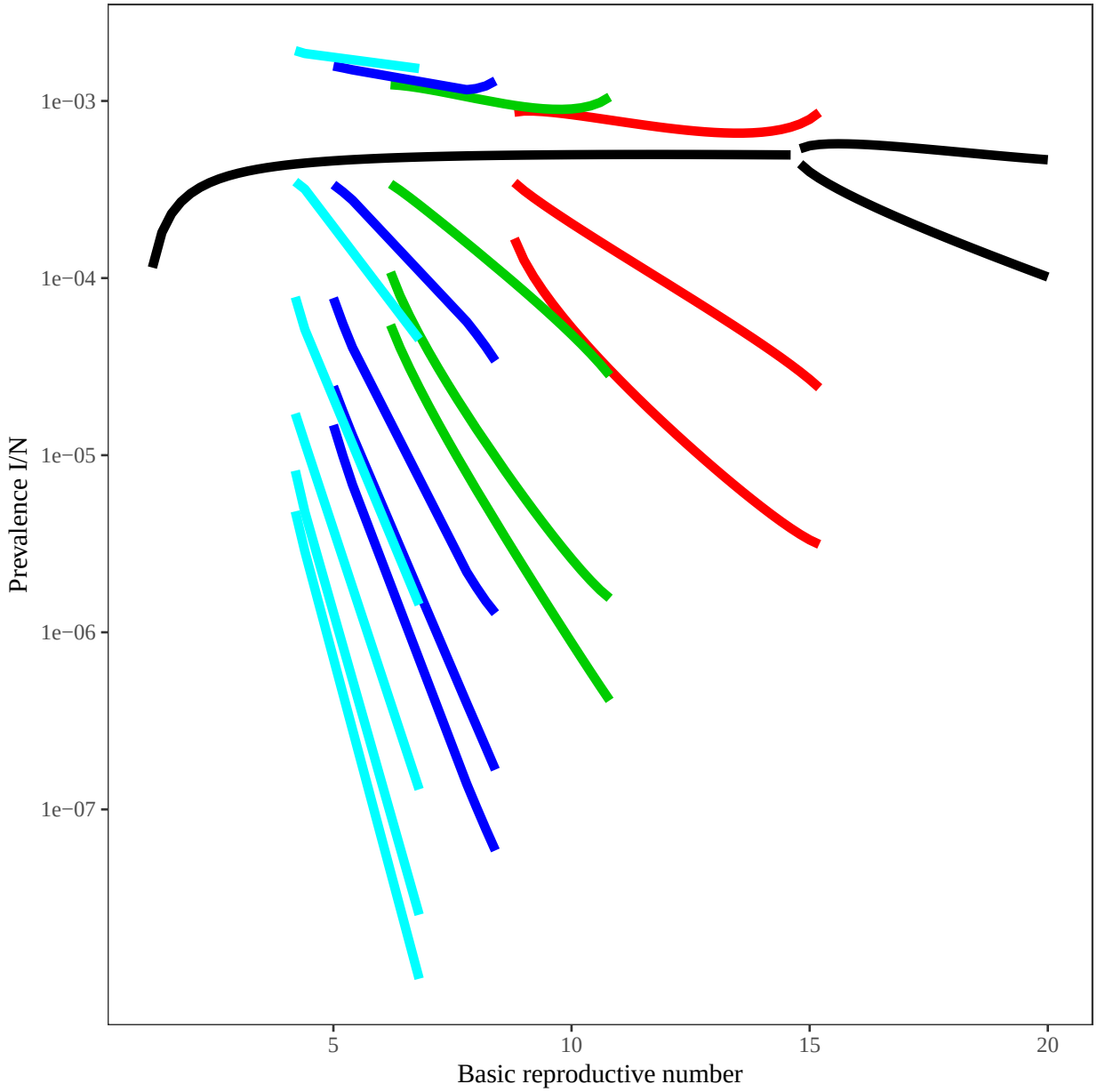
Once the simulation is done, we can clean the simulation data to generate a data frame that can be used to create a bifurcation diagram:

```
bdf <- blist %>%
    lapply(bind_rows, .id="sim") %>%
    bind_rows

bifur_df <- bdf %>%
    group_by(sim, R0) %>%
    filter(prevalence > 0) %>%
    filter(!duplicated(prevalence)) %>%
    group_by(R0) %>%
    filter(!duplicated(round(prevalence, 10))) %>%
    group_by(sim, R0) %>%
    mutate(
        i=n()
    ) %>%
    group_by(R0, i) %>%
    mutate(
        prevalence=ifelse(i==1, mean(prevalence), prevalence)
    ) %>%
    filter(!duplicated(prevalence)) %>%
    group_by() %>%
    mutate(
        sim=ifelse(i==1, 1, sim)
    ) %>%
    group_by(sim, R0, i) %>%
    mutate(
        prevalence=sort(prevalence),
        j=seq_along(i)
    ) %>%
    group_by %>%
    mutate(
        sim=ifelse(i==5 & round(R0, 1)==6.8, 2, 1)
    ) %>%
    group_by(R0, i, j, sim) %>%
    summarize(
        prevalence=mean(prevalence)
    ) %>%
    as.data.frame
```

We can now pass this data to `ggplot` objects in order to overlay a different plot on top of the bifurcation diagram. The following code illustrates how bifurcation diagrams are created

```
gbifur <- ggplot(bifur_df) +
```

```
    geom_path(aes(R0, prevalence, group=interaction(i, j, sim),
                  col=factor(i)), lwd=2) +
    scale_y_log10("Prevalence I/N",
                  breaks=c(1e-3, 1e-4, 1e-5, 1e-6, 1e-7)) +
    scale_x_continuous("Basic reproductive number") +
    scale_color_manual(values=c(1, 1, 2, 3, 4, 5, 6)) +
    theme(
        legend.position = "none",
        panel.grid = element_blank()
    )
plot(gbifur)
```



4 Probability of coherence in the deterministic model

Before going into the bulk of the deterministic model section, we load a few useful functions which will be used throughout. The first is a function to calculate the incoherence measure defined in the main paper: Let $\vec{I}(t) = (I_1(t), \ldots, I_n(t))$ denote the number of individuals infected at time t in patches $k = 1, \ldots, n$. We then define incoherence δ at time t as

$$\delta(t) = ||\vec{I}(t) - \langle \vec{I}(t) \rangle \vec{e}||_2,$$

where $\langle \vec{I}(t) \rangle = \sum_{k=1}^{n} I_k(t)/n$ denotes the mean prevalence among the patches at time t, and $\vec{e} = (1, \ldots, 1)$.

```
# coherence metric calculation
coherence_calc <- function(x) {
    norm(x - rep(mean(x), length(x)), type = "2")
}
```

Below is a function that generates random initial conditions with $0 < S_k < 0.1N$, and $0 < I_k < 0.0001N$ for each patch as described Earn et al. (2000): A Simple Model for Complex Dynamical Transitions in Epidemics.

```
r.init.science <- function(m) {
    initial <- list(S = runif(m, 0, 0.1 * base.params$pop))
    initial$I <- runif(m, 0, 1e-04 * base.params$pop)
    initial$R <- rep(base.params$pop, m) - initial$S - initial$I
    initial
}
```

Lastly, we have a function that returns the results of the model simulation (from the `.cpp` file) in a convenient format.

```
run_SIR <- function(p, init, m, seas) {
```

```
    # get number of patches
    n <- nrow(m)

    # Run the model
    raw.result <- SIRmodel_npatch(p, init, m, seas)

    # Initialize result data frame
    result <- data_frame(t = raw.result$time)

    # Pull results
    S <- raw.result$S
    I <- raw.result$I
    R <- raw.result$R

    # Change column names
    colnames(S) <- str_c("S", 1:n)
    colnames(I) <- str_c("I", 1:n)
    colnames(R) <- str_c("R", 1:n)

    # Combine data frames
    cbind(result, S, I, R)
}
```

Here we check if the data file corresponding to $m = 0.0001$ is present. If it is, then we load that file, and if it isn't, then we run the simulation that creates that file. Note that loading in the file will save a lot of time and if it is missing, we strongly recommend you get the file from us. Here we only show the code that loads data corresponding to $m = 0.0001$, but the code for the remaining m values (0.001, 0.01, 0.1, 0.5) are similar and therefore hidden.

You may be asking why we run all of these separately. This was to split up the computational load among our measly laptops. The m values were split up as follows (by assigning the `mvec` vector accordingly:

- `mvec = 0.0001` (assigned to `df1`)
- `mvec = c(0.001, 0.01)` (assigned to `df2`)
- `mvec = c(0.1, 0.5)` (assigned to `df3`)

For each m value we test 100 random initial conditions and run the model for 100 years. Then we record the average incoherence value in the last 2 years and the last 5 years of the simulation for each initial condition.

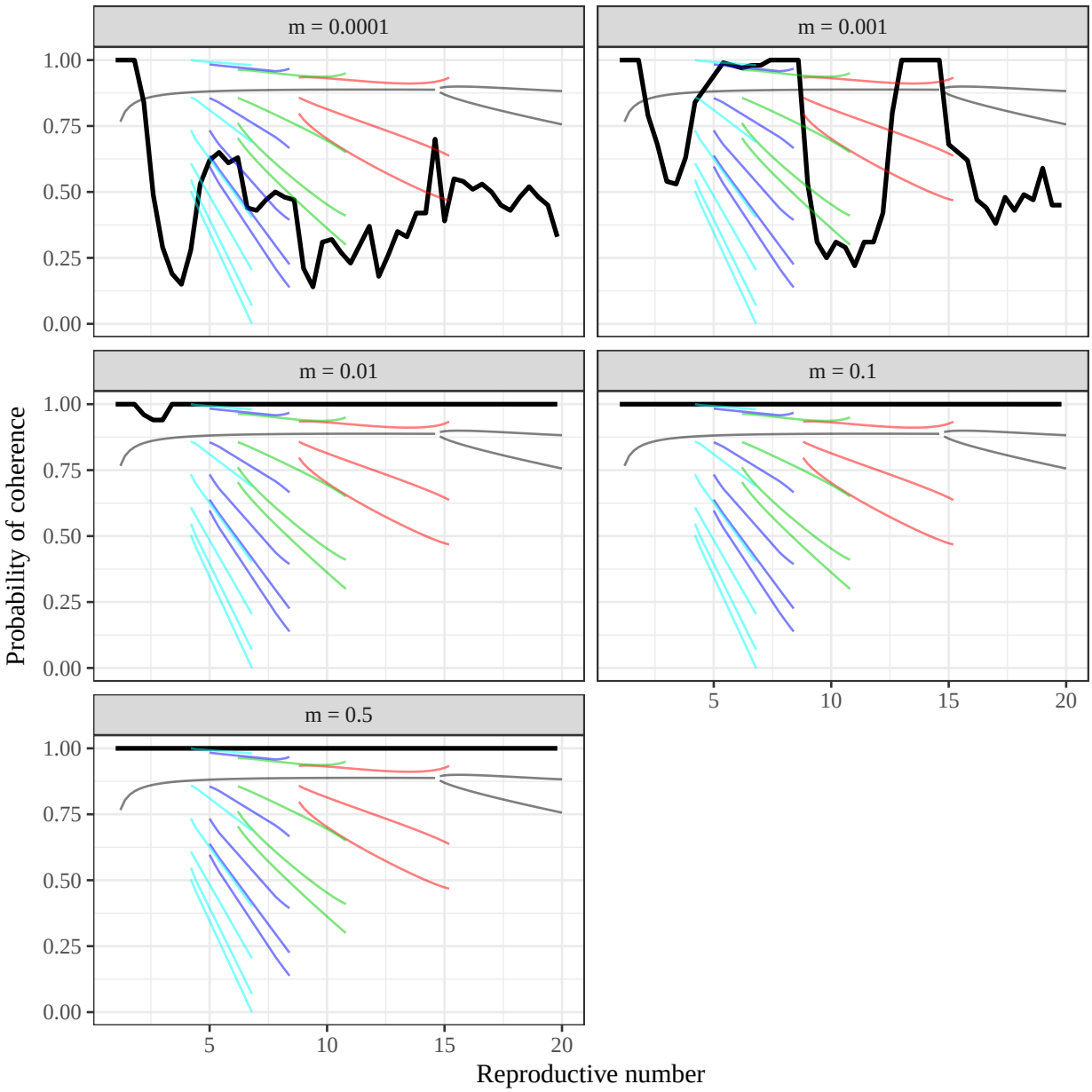
```
if (file.exists("coherence_m0.0001.rda")) {
```

```
    load("coherence_m0.0001.rda")
    df1 <- reslist
} else {
    nsim <- 100
    R0vec <- seq(1, 20, by = 0.4)
    mvec <- c(1e-04)
    reslist <- vector("list", length(mvec))
    set.seed(101)
    for (m in mvec) {
        M <- matrix(c(1 - m, m, m, 1 - m), 2, 2)
        subreslist <- vector("list", length(R0vec))
        for (R in R0vec) {
            print(paste(m, R, sep = ","))

            pp <- base.params
            pp[["R0"]] <- R

            subsubreslist <- vector("list", nsim)

            for (i in 1:nsim) {

                init <- r.init.science(2)
                df <- run_SIR(pp, init, M, term_time)

                incoherence2 <- apply(df[(nrow(df) - 2 * 365):nrow(df), c("I1",
                  "I2")], 1, coherence_calc) %>% mean()
                incoherence5 <- apply(df[(nrow(df) - 5 * 365):nrow(df), c("I1",
                  "I2")], 1, coherence_calc) %>% mean()

                subsubreslist[[i]] <- data_frame(S01 = init$S[1], S02 = init$S[2],
                  I01 = init$I[1], I02 = init$I[2], R0 = R, incoherence2 = incoherence2,
                  incoherence5 = incoherence5)
            }
            subreslist[[which(R0vec == R)]] <- do.call("rbind", subsubreslist)
        }

        ss <- do.call("rbind", subreslist)
        ss$m <- m
        reslist[[which(mvec == m)]] <- ss
    }
}
```

Here we combine the three data frames that consist of the data for different m values, and then we construct the plot showing the probability of coherence in the main paper. This

particular figure uses a threshold of average incoherence < 100 in the last 2 years.

```
R0df <- bind_rows(df1, df2, df3) %>% group_by(R0, m) %>% summarize(prob.coherence = leng
    100)))/100) %>% mutate(m.factor = as.factor(m))

bifur_df_norm <- bifur_df %>% mutate(lp = log(prevalence)) %>% mutate(nlp = lp -
    min(lp)) %>% mutate(nlp = nlp/max(nlp))

levels(R0df$m.factor) <- c("m = 0.0001", "m = 0.001", "m = 0.01", "m = 0.1",
    "m = 0.5")

gprob.main <- ggplot(R0df) + geom_line(aes(R0, prob.coherence), size = 1) +
    facet_wrap(~m.factor, ncol = 2) + geom_path(data = bifur_df_norm, aes(R0,
    nlp, group = interaction(i, j, sim), col = factor(i)), alpha = 0.5) + labs(x = "Repr
    y = "Probability of coherence") + scale_color_manual(values = c(1, 1, 2,
    3, 4, 5, 6)) + theme(legend.position = "none")

gprob.main
```



Next, we calculate the mean probabilities of coherence for mixing values of 0.01% ($m = 0.0001$), 0.1% ($m = 0.001$), and 1% ($m = 0.01$) across all basic reproductive number values.

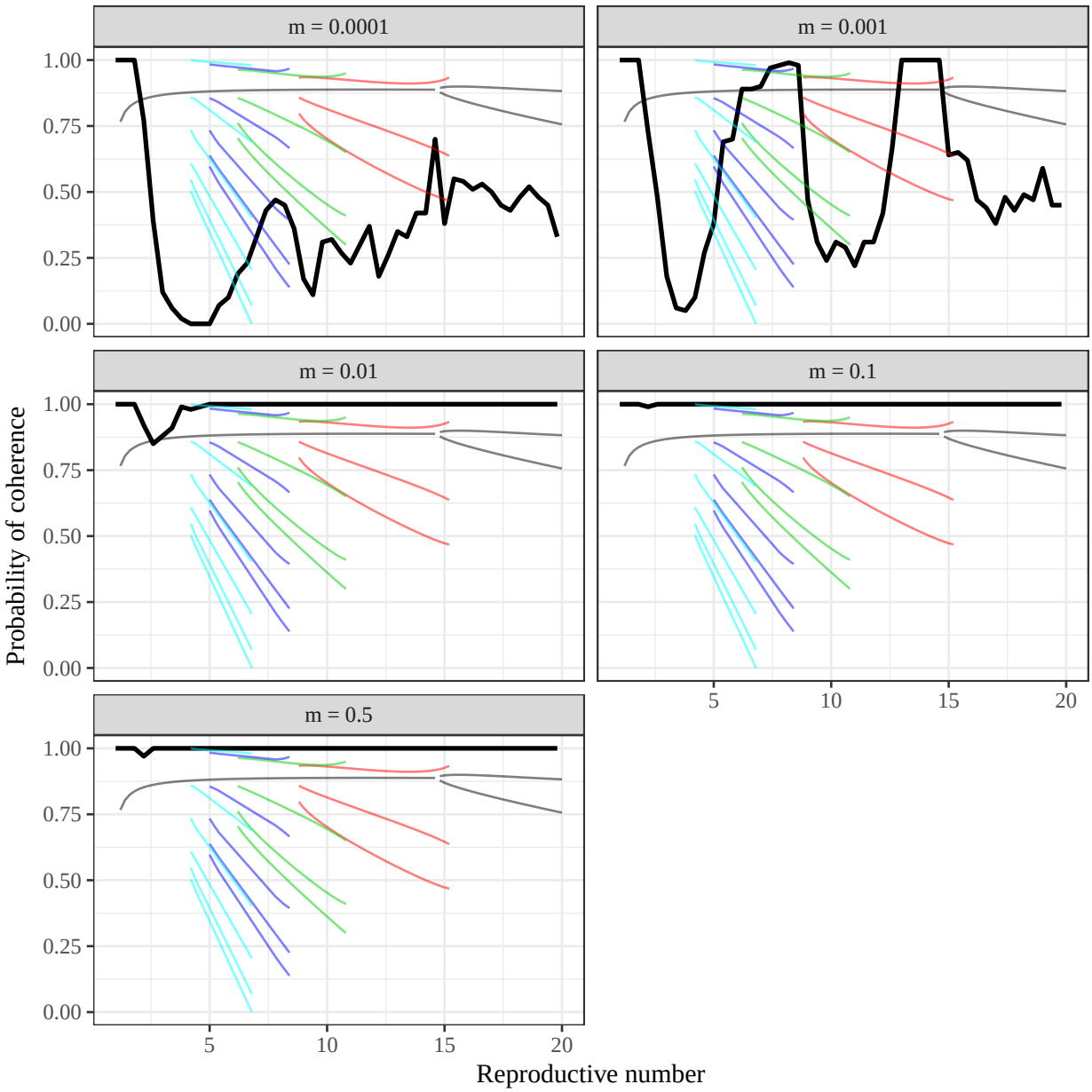
```
# for 0.0001
```

```r
R0df_0.0001 <- bind_rows(df1[[1]]) %>% group_by(R0, m) %>% summarize(prob.coherence = le
    100)))/100)
str_c("The mean for m=0.0001 is ", mean(R0df_0.0001$prob.coherence))

## [1] "The mean for m=0.0001 is 0.459166666666667"

# for 0.001
R0df_0.001 <- R0df <- bind_rows(df2[[1]]) %>% group_by(R0, m) %>% summarize(prob.coheren
    100)))/100)
str_c("The mean for m=0.001 is ", mean(R0df_0.001$prob.coherence))

## [1] "The mean for m=0.001 is 0.689375"

# for 0.01
R0df_0.01 <- R0df <- bind_rows(df2[[2]]) %>% group_by(R0, m) %>% summarize(prob.coherenc
    100)))/100)
str_c("The mean for m=0.01 is ", mean(R0df_0.01$prob.coherence))

## [1] "The mean for m=0.01 is 0.996666666666667"
```

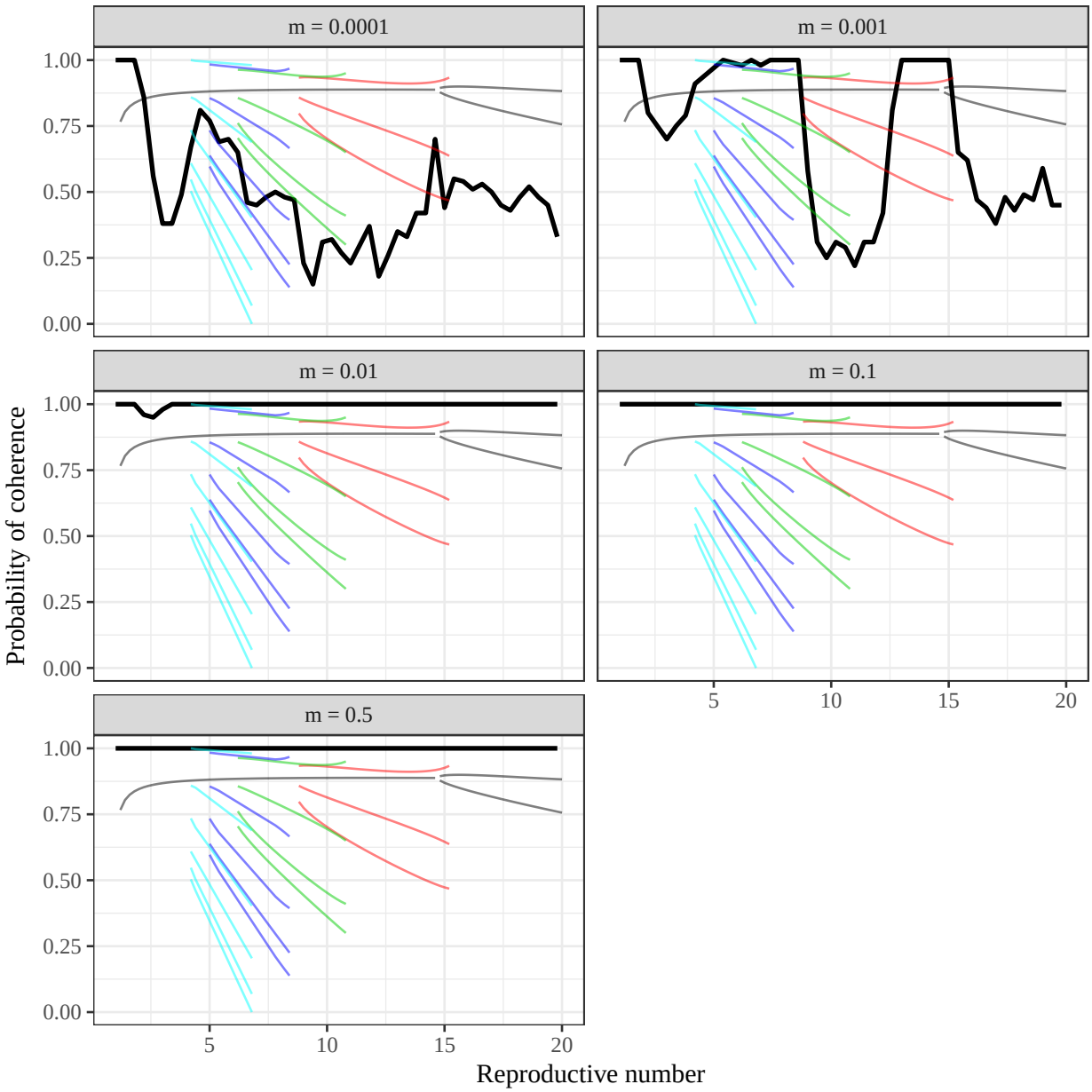
4.1 Choice of incoherence threshold

Here we demonstrate that we obtain similar coherence probabilities even with different choices of threshold. This demonstrates that our observed results and inferences aren't sensitive to our choice of coherence threshold.

Threshold of 10:

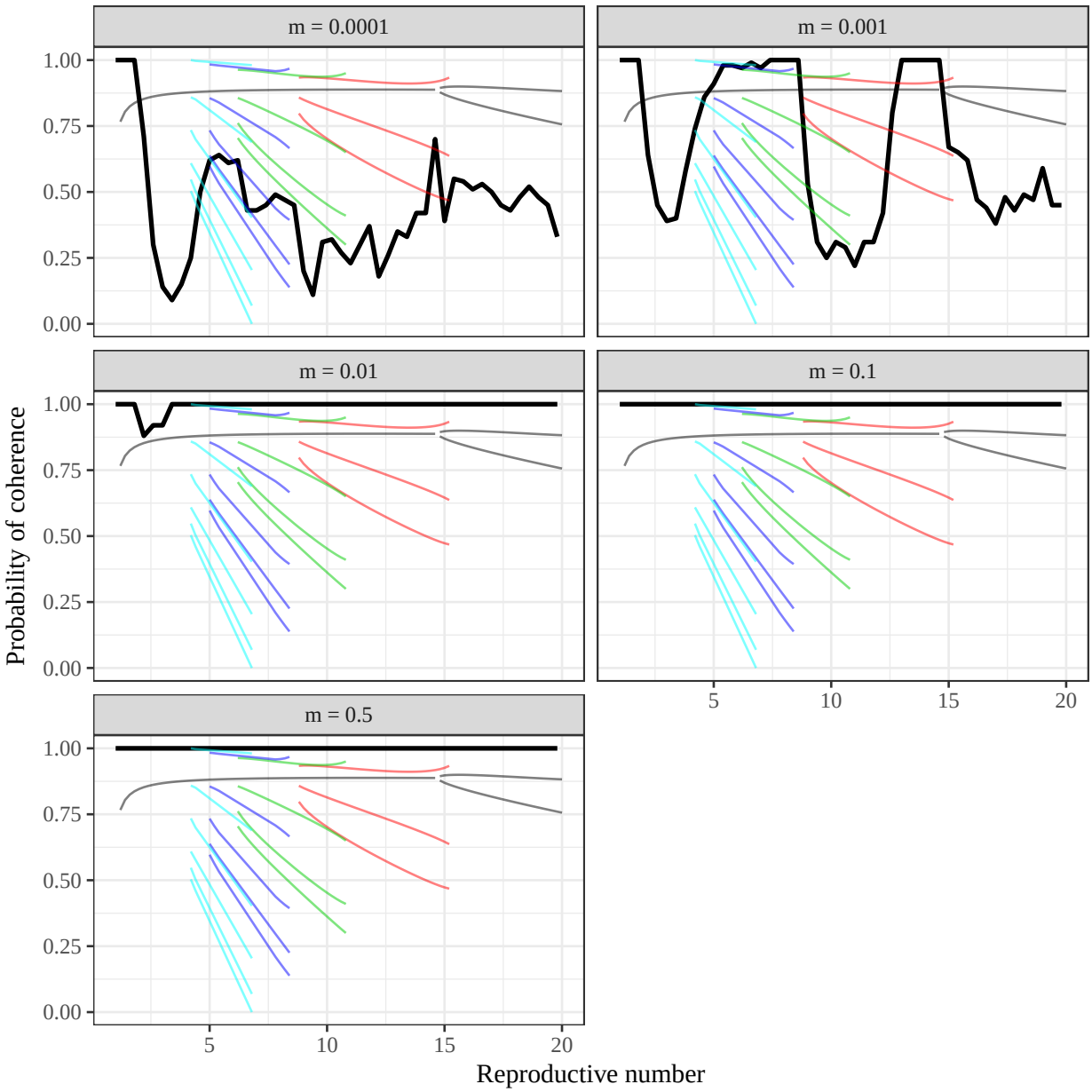


Threshold of 200:



4.2 Coherence in the last 5 years

Here we show that our inferences are also not sensitive to whether we measure the average incoherence over the last 2 or 5 years. The plot in the main paper uses a last-2-year measure of average incoherence, but we see below that changing this to the last 5 years (with the same incoherence < 100 threshold) results in similar behaviour.



5 Synchrony and asynchrony in stochastic model

We observe stretches of both synchrony and asynchrony in simulated trajectory for the stochastic model. We let $\mathcal{R}_0 = 17$ and let the mixing proportion be given by 0.1% ($m = 0.001$). Our initial conditions are given by the endemic equilibrium, and the rest of our parameters are taken from the base parameters list. We observe the change from synchronous behaviour to asynchronous behaviour in Figure 2, which can be reproduced using the following code:

```r
pp3 <- base.params
pp3[["R0"]] <- 17.0

init <- initfun(pp3, 2, T)
m <- 0.001
M <- matrix(c(1-m, m, m, 1-m), 2, 2)

set.seed(10)
df4 <- SIRmodel_npatch_stochastic(pp3, init, M, term_time)

## Error in SIRmodel_npatch_stochastic(pp3, init, M, term_time):  could not find
function "SIRmodel_npatch_stochastic"

simdf3 <- list(
    patch1=data.frame(
        time=df4$time,
        I=df4$I[,1]
    ),
    patch2=data.frame(
        time=df4$time,
        I=df4$I[,2]
    )
) %>%
    bind_rows(.id="patch") %>%
    gather(key, value, -time, -patch) %>%
    filter(time > 60, time < 85)

## Error in data.frame(time = df4$time, I = df4$I[, 1]):  object 'df4' not found

gex <- ggplot(simdf3) +
    geom_line(aes(time, value, col=patch), lwd=1.1) +
    scale_x_continuous("Time (years)", expand=c(0, 0)) +
    scale_y_continuous("Prevalence") +
    scale_color_manual(labels=c("Patch 1", "Patch 2"), values=c("#D55E00", "#0072B2")) +
    theme(
        panel.grid = element_blank(),
        legend.title = element_blank(),
        legend.position = c(0.1, 0.84),
        strip.text = element_text(size=25))

## Error in ggplot(simdf3):  object 'simdf3' not found

plot(gex)

## Error in plot(gex):  object 'gex' not found
```

6 Probability of extinction

6.1 Factorial simulation

Here, we vary $\mathcal{R}_0$ from 1 to 20 by 0.2 steps and simulate the stochastic model 100 times for each combination of $\mathcal{R}_0$ and mixing proportion m. Here, we do not use random initial conditions as it may lead to exaggeration of global extinction probability (some initial conditions may lead to immediate extinction without a start of an epidemic). Instead, we start from the endemic equilibrium. Once again, as this simulation takes approximately 11 hours, we load a saved file instead.

```
if (file.exists("stochastic.rda")) {
```

```
    load("stochastic.rda")
} else {
    nsim <- 100
    R0vec <- seq(1, 20, by=0.2)
    mvec <- c(0.001, 0.01, 0.1, 0.5)

    reslist <- vector('list', length(mvec))
    set.seed(101)
    for (m in mvec) {

        M <- matrix(c(1-m, m, m, 1-m), 2, 2)
        subreslist <- vector('list', length(R0vec))

        for (R in R0vec) {
            pp <- base.params
            pp[["R0"]] <- R

            subsubreslist <- vector('list', nsim)
            for (i in 1:nsim) {
                init <- initfun(pp, 2, T)
                df <- SIRmodel_npatch_stochastic(base.params, init, M, term_time)
                zero1 <- which(df$I[,1]==0)
                zero2 <- which(df$I[,2]==0)
                subsubreslist[[i]] <- data.frame(
                    sim=i,
                    R0=R,
                    local=(any(df$I[,1]==0) || any(df$I[,2]==0)),
                    global=(any(df$I[,1]==0 & df$I[,2]==0)),
                    rescue1=length(which(diff(zero1) != 1)),
                    rescue2=length(which(diff(zero2) != 1))
                )
            }
            subreslist[[which(R0vec==R)]] <- do.call("rbind", subsubreslist)
        }
        ss <- do.call("rbind", subreslist)
        ss$m <- m
        reslist[[which(mvec==m)]] <- ss
    }
    save("reslist", file="stochastic.rda")
}
```

6.2 Plot

Using the simulated file, we can plot the results of the simulation. First, we clean up the data using `dplyr` and `tidyr` pacakge so that it can be used with a `ggplot` function.

```
sdf <- reslist %>%
    bind_rows %>%
    group_by(R0, m) %>%
    summarize(
        local=mean(local),
        global=mean(global)
    ) %>%
    gather(key, value, -R0, -m) %>%
    mutate(m=paste0("m = ", m),
           key=factor(key, levels=c("global", "local"),
                      labels=c("Global extinction", "Local extinction")))
```

We are interested in comparing global and local extinction probabilities for measles parameters. The following code demonstrates how to achieve appropriate statistics.

```
sdf %>%
    filter(R0 > 12.5, R0 <=18, m=="m = 0.001") %>%
    group_by(key) %>%
    summarize(
        lwr=range(value)[1],
        upr=range(value)[2]
    )

## # A tibble: 2 x 3
##   key                lwr   upr
##   <fct>            <dbl> <dbl>
## 1 Global extinction 0.    0.130
## 2 Local extinction  0.230 0.650
```

We are interested in comparing global extinction probabilities across different mixing proportions. The following code demonstrates how to achieve appropriate statistics.

```
sdf %>%
```

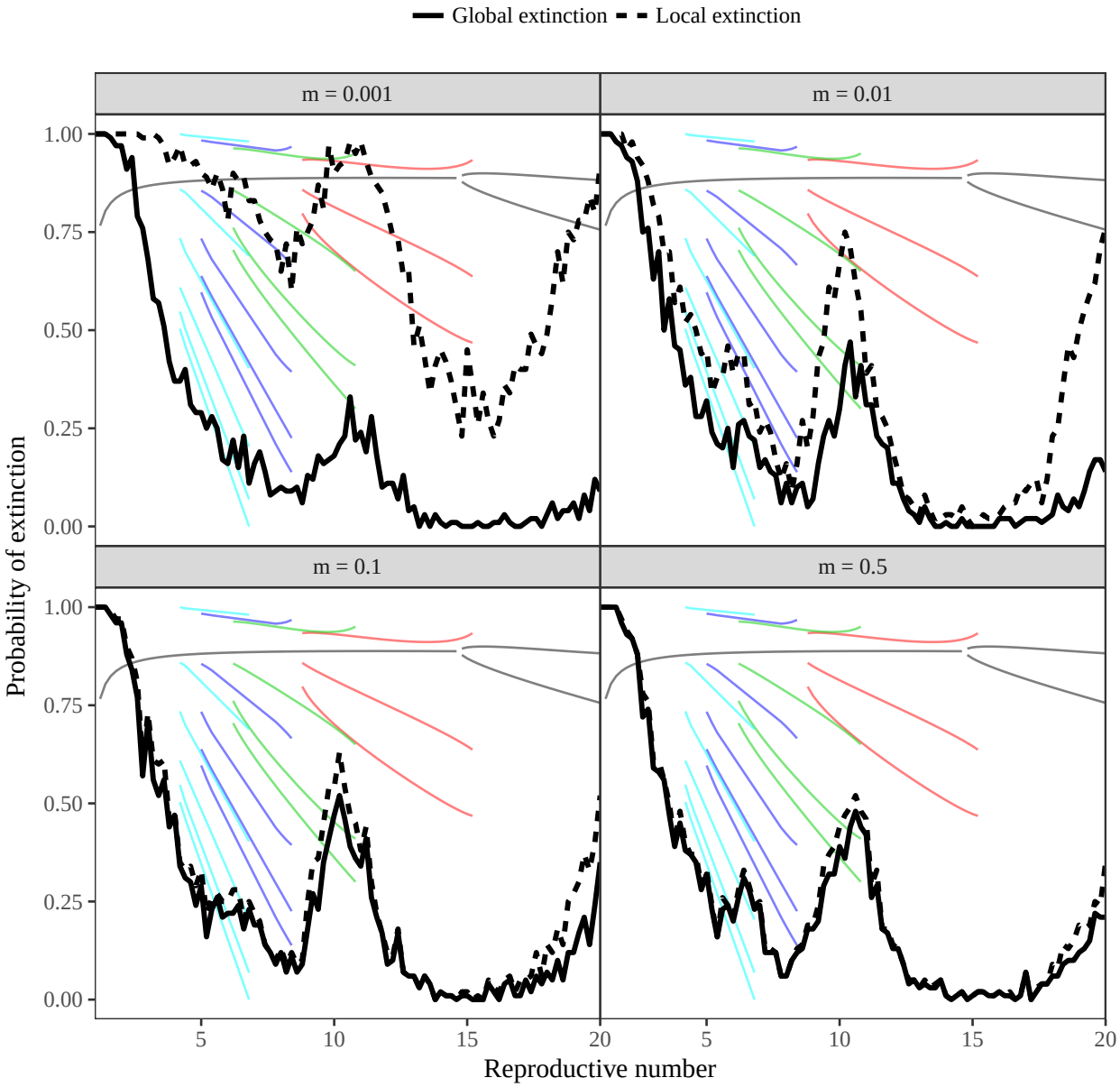
```
    filter(R0 > 10, key=="Global extinction") %>%
    group_by(m) %>%
    summarize(
        lwr=range(value)[1],
        upr=range(value)[2]
    )

## # A tibble: 4 x 3
##   m           lwr   upr
##   <chr>     <dbl> <dbl>
## 1 m = 0.001    0. 0.330
## 2 m = 0.01     0. 0.470
## 3 m = 0.1      0. 0.520
## 4 m = 0.5      0. 0.480
```

Before we plot the result of our stochastic simulation, we have to normalize the bifurcation diagram so that it can be layed below the probability diagram.

```
bifur_df_norm <- bifur_df %>%
    mutate(lp=log(prevalence)) %>%
    mutate(nlp=lp-min(lp)) %>%
    mutate(nlp=nlp/max(nlp))
```

Finally, we generate the diagram of interest:

```
gstoch <- ggplot(bifur_df_norm) +
    geom_point(data=filter(bifur_df_norm, i==5, round(R0, 1)==6.8),
               aes(R0, nlp, group=interaction(i, j), col=factor(i)), size=0.5, alpha=0.5
    geom_path(aes(R0, nlp, group=interaction(i, j, sim), col=factor(i)), alpha=0.5) +
    geom_line(data=sdf, aes(R0, value, lty=key), lwd=1.1) +
    scale_y_continuous("Probability of extinction") +
    scale_x_continuous("Reproductive number", expand=c(0,0)) +
    facet_wrap(~m) +
    scale_color_manual(values=c(1, 1, 2, 3, 4, 5), guide=FALSE) +
    theme(
        legend.position = "top",
        legend.title = element_blank(),
        panel.spacing = grid::unit(0, "cm"),
        panel.grid = element_blank()
    )

plot(gstoch)
```



7 Save results for inclusion in main paper

We now save our figures for use in the final paper.

The plot for probability of coherence:

```
ggsave("probabilitycoherence.pdf", device= "pdf", gprob.main, width=7, height = 5)
```

8 Time to generate this document

CPU time to generate this document: 7.435S seconds