A Simple Model of Between-Hive Transmission of Nosemosis

Keimpe Dijkstra, Wouter Zeevat

2022-06-19

Contents

1 Introduction

Nosemosis is a big problem especially in North America. It is causing a lot of bees to die which is leading to a honeybee crisis in North America, It is even starting to become a problem in Europe now aswell. Nosemosis is a disease caused by two different species of microsporidian parasites, That are a type of fungus that form spores. Whenever a bee becomes infected, it has problems digesting food, which results to a shorter lifespan. Whenever a colony's queen gets infected, they will stop laying eggs. All leading to the whole colony dying [1].

Something what makes the disease especially dangerous is the spreading. One bee could be infected, leading to the whole hive to be infected. Whenever one of those bees is close to another hive, it could infect one of their bees resulting in the whole process happening again and again. Until a whole apiary is infected.

the spread of the infection follows this reproductive cycle.

SPRING

- Queen lays more eggs than usual.
- The amount of adult bees increases.
- Spore transmission as older bees feed younger and young will clean the comb cells contaminated with spores.
- New emerged bees get spores from older bees.
- Nosemosis has a spike when it is spring.

Summer

- Queen still lays eggs.
- Young bees continue to arrive become infected.
- The infection rate stays high until mid-summer

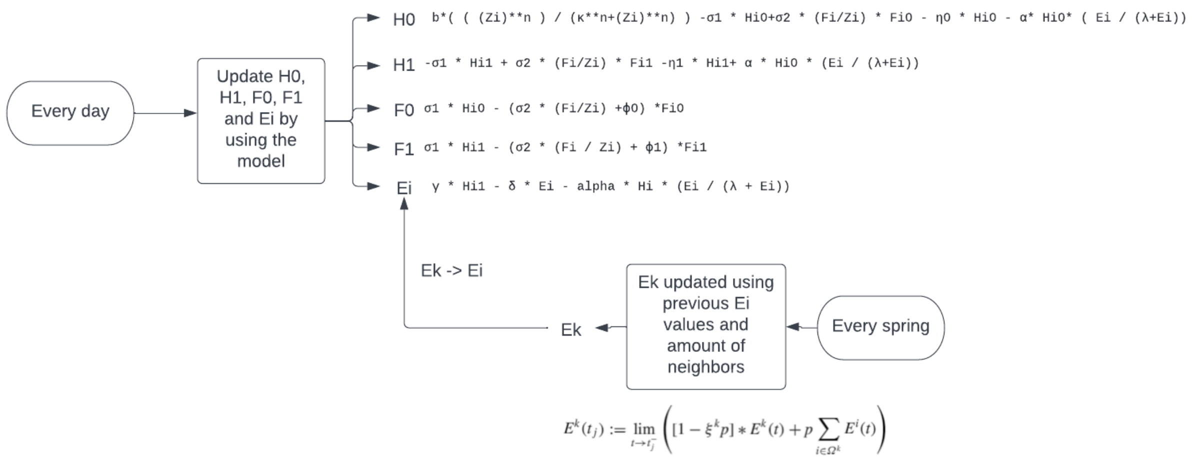
Autumn and winter

- The queen reduces laying eggs
- N. ceranae spore transmission decreases
- There is a Winter die-back which is not caused by N. ceranae infection [2]

As mentioned, the spread of Nosemosis is a big problem and needs to be addressed. Which is why there is a mathematical model that follows the spread of Nosemosis [3]. This mathematical model is used to create a simulation that will be used to understand the spread of Nosemosis throughout different hives in an apiary. The simulation could be used by beekeepers that want to keep track of the health of their hives. The goal of this project is to give a visual output of what the spread of Nosemosis looks like. And what the big factors that change the outcome are.

2 The model

The model, models the spread of Nosemosis throughout different hives, each day the 5 main values get updated, but every year when spring arrives, the last value (which is the disease level), gets changed according to different values due to hive maintenance in early spring.


Every day
Update H0, H1, F0, F1 and Ei by using the model
H0
H1
F0
F1
Ei
Ek -> Ei
Ek
Ek updated using previous Ei values and amount of neighbors
Every spring

```
H0  b*( ( (Zi)**n ) / (κ**n+(Zi)**n) ) -σ1 * Hi0+σ2 * (Fi/Zi) * Fi0 - η0 * Hi0 - α* Hi0* ( Ei / (λ+Ei))
H1  -σ1 * Hi1 + σ2 * (Fi/Zi) * Fi1 -η1 * Hi1+ α * Hi0 * (Ei / (λ+Ei))
F0  σ1 * Hi0 - (σ2 * (Fi/Zi) +ϕ0) *Fi0
F1  σ1 * Hi1 - (σ2 * (Fi / Zi) + ϕ1) *Fi1
Ei  γ * Hi1 - δ * Ei - alpha * Hi * (Ei / (λ + Ei))
```

$$E^k(t_j) := \lim_{t \to t_j^-} \left([1 - \xi^k p] * E^k(t) + p \sum_{i \in \Omega^k} E^i(t) \right)$$

Figure 1: image

2.1 The grid system

Before explaining the formula, it is important to understand how the hives are positioned. This is necessary because the hives interact with eachother. A neighbourhood is a bunch of hives next to eachother on a grid. Each hive has its neighbours. for example: on a 3x3 grid, hive 1 has 3 neighbours (2, 4 and 5), and hive 5 has 8 neighbours (1, 2, 3, 4, 6, 7, 8, 9). This also means that whenever hive 5 is infected, It spreads faster than if hive 1 was infected.

2.2 The equations

The susceptible hive population (Hi0) is calculated within the first equation. The Infected hive population (Hi1) is calculated in the second formula, which uses the environmental disease potential (Ei) to calulate the sub-population. The third and fourth formulas calculate the same as Hi0 and Hi1, Except that it is for the foragers. The last and most important formula calculates the environmental disease potential (Ei). This also gets affected by another formula that only runs every first day of spring. The environmental disease potential for a curtain hive (Ek), is that value (which overwrites the Ei value). Resulting to the Ei value getting overwritten by the Ek value every year. The Ek formula is a little complicated. At first, it uses The sum of all its neighbous Ek values. Then it calculates the lim of the result of the formula, together with all its previous results. This happens in order to calculate where the Ek value is going to.

The parameters differ each season. The reason for this is that There's different factors every season. The values of all the parameters are shown in the table below. The initial values of the states are 10ˆ4 for Hi0 and Fi0, and 0 for the other three. The reasoning behind the initial values are that there needs to be a population to begin with. Which is 10ˆ4 bees in this case. The spore deposition rate parameter (y), is different than the other parameters. The values are xyW, meaning it needs to multiply by yw. The article says yw is 0.1. Meaning all the spore deposition rate values will be substituted by 0.1.

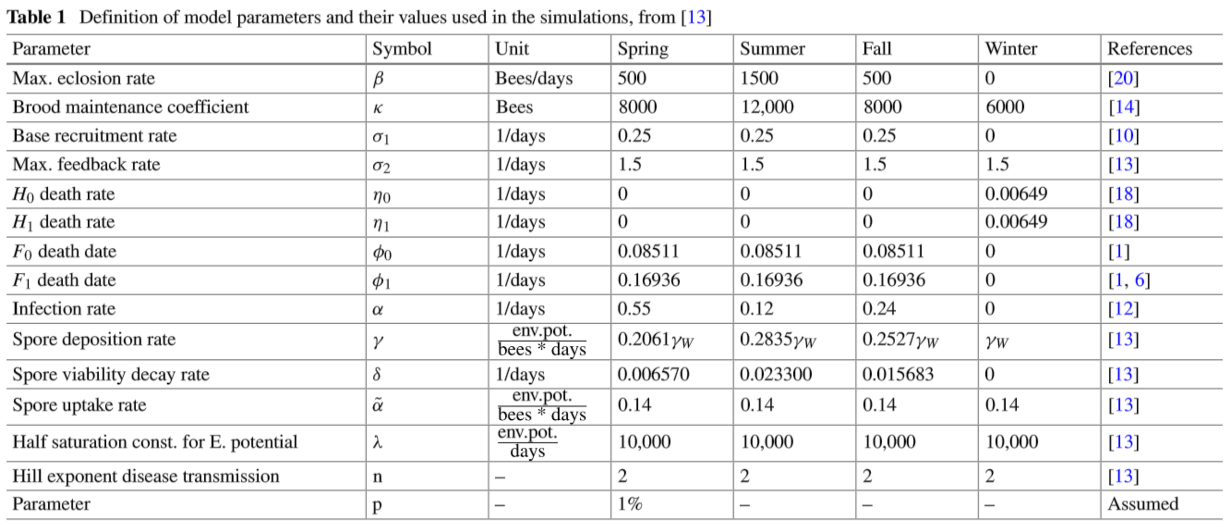


Table 1 Definition of model parameters and their values used in the simulations, from [13]

Parameter	Symbol	Unit	Spring	Summer	Fall	Winter	References
Max. eclosion rate	β	Bees/days	500	1500	500	0	[20]
Brood maintenance coefficient	κ	Bees	8000	12,000	8000	6000	[14]
Base recruitment rate	σ_1	1/days	0.25	0.25	0.25	0	[10]
Max. feedback rate	σ_2	1/days	1.5	1.5	1.5	1.5	[13]
H_0 death rate	η_0	1/days	0	0	0	0.00649	[18]
H_1 death rate	η_1	1/days	0	0	0	0.00649	[18]
F_0 death date	ϕ_0	1/days	0.08511	0.08511	0.08511	0	[1]
F_1 death date	ϕ_1	1/days	0.16936	0.16936	0.16936	0	[1, 6]
Infection rate	α	1/days	0.55	0.12	0.24	0	[12]
Spore deposition rate	γ	$\frac{\text{env.pot.}}{\text{bees * days}}$	$0.2061\gamma_W$	$0.2835\gamma_W$	$0.2527\gamma_W$	γ_W	[13]
Spore viability decay rate	δ	1/days	0.006570	0.023300	0.015683	0	[13]
Spore uptake rate	$\tilde{\alpha}$	$\frac{\text{env.pot.}}{\text{bees * days}}$	0.14	0.14	0.14	0.14	[13]
Half saturation const. for E. potential	λ	$\frac{\text{env.pot.}}{\text{days}}$	10,000	10,000	10,000	10,000	[13]
Hill exponent disease transmission	n	–	2	2	2	2	[13]
Parameter	p	–	1%	–	–	–	Assumed

Figure 2: image

3 Results

3.1 Apiary model

The way in which the apiary is set up is quite simple, the hives are placed within a rectangular grid. It is assumed that only neighboring beehives can transmit the pathogen, with an exception of the beehive maintenance in early spring. So in this model a hive in the corner would be able to contaminate its three neighbors and a hive in the middle of the grid would be able to affect eight neighboring hives. We constructed a function called “apiary_creation “ which builds a model of any grid size of n1×n2, stored in a dictionary-like object with the first hive being in the bottom left corner.

3.2 base scenario

Firstly we simulate a scenario in which no pathogen is present, the beehives are thus identical. Fig1 show the state variables, the hive quickly reaches a periodic solution, in which the population peaks at the end of summer and reaches its minimum at the end of winter. Fig2 shows the environmental potential value, the measure of contamination with the pathogen, this stays zero because no nosemosis is introduced.

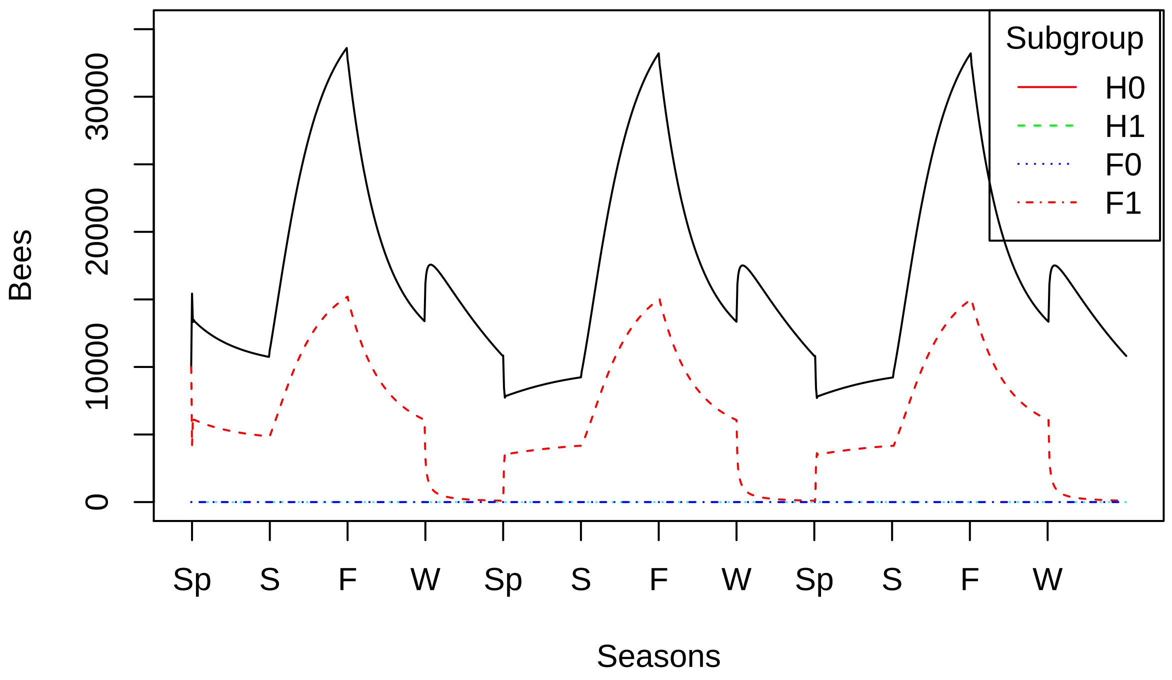


Fig1

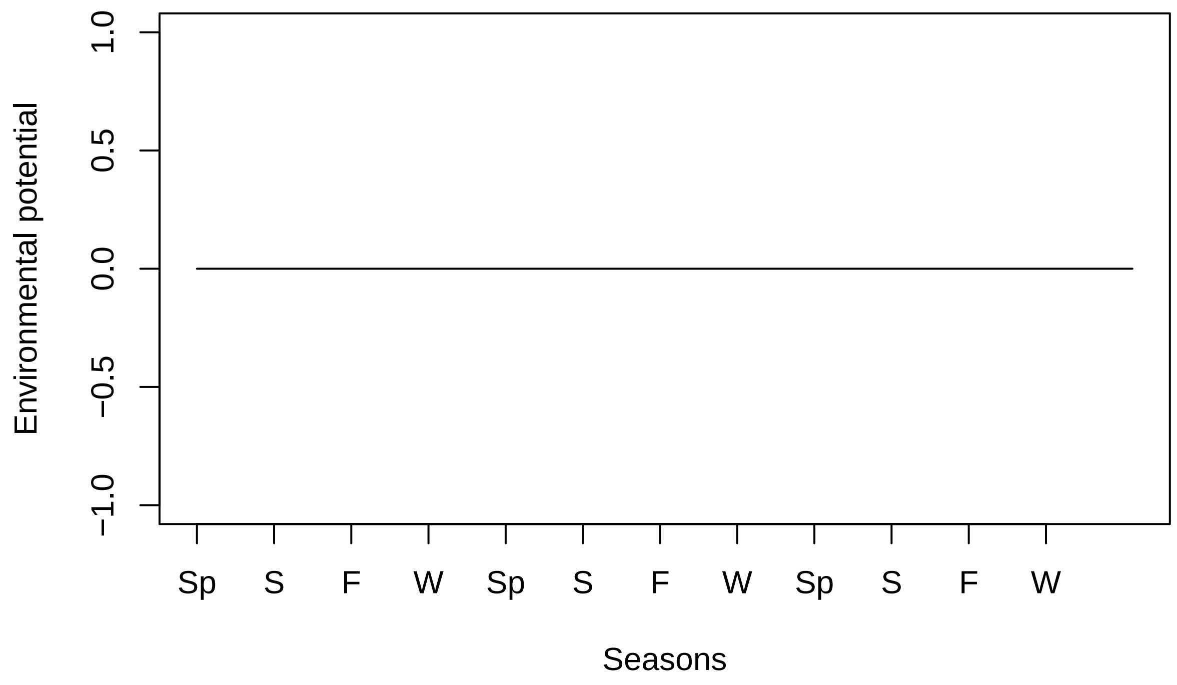


Fig2

3.3 Disease propagation between hives

Two simulations were conducted in order to study the effect of the hives, one in which the first hive infected was hive one, a corner hive. And in the second simulation center hive number 13 was infected. Introducing the pathogen is done by setting Hi1 to ten, the infected hive bee sub-population, to 10 on the first day of the first winter. In all the three seasons before that the state variables were kept at their starting values. For this simulation we assumed the spore uptake ($\tilde{a}$) is 14 and a between hive transmission (p) of 1%. For the spore deposition (y) we assumed 0.1 for the first three seasons and 0.0088 for the winter. The simulation ran for 30 years.

Fig3 describes the environmental potential of the first hive, we observe an unexplainable sudden rise and fall of the Ei where we would theoretically expect a gradual increase of Ei. Zooming in on the graph after the peak in fig4, we see a periodic solution with the maximum value being approximately 400. This suggest that the bee hives do not suffer so much from the fungus that the whole hive collapses, but rather that the disease is season bound. Looking at the same plots but for the 30th hive, we see that that the pathogen reached the hive after about five years. From there it shows the same pattern as the first hive.

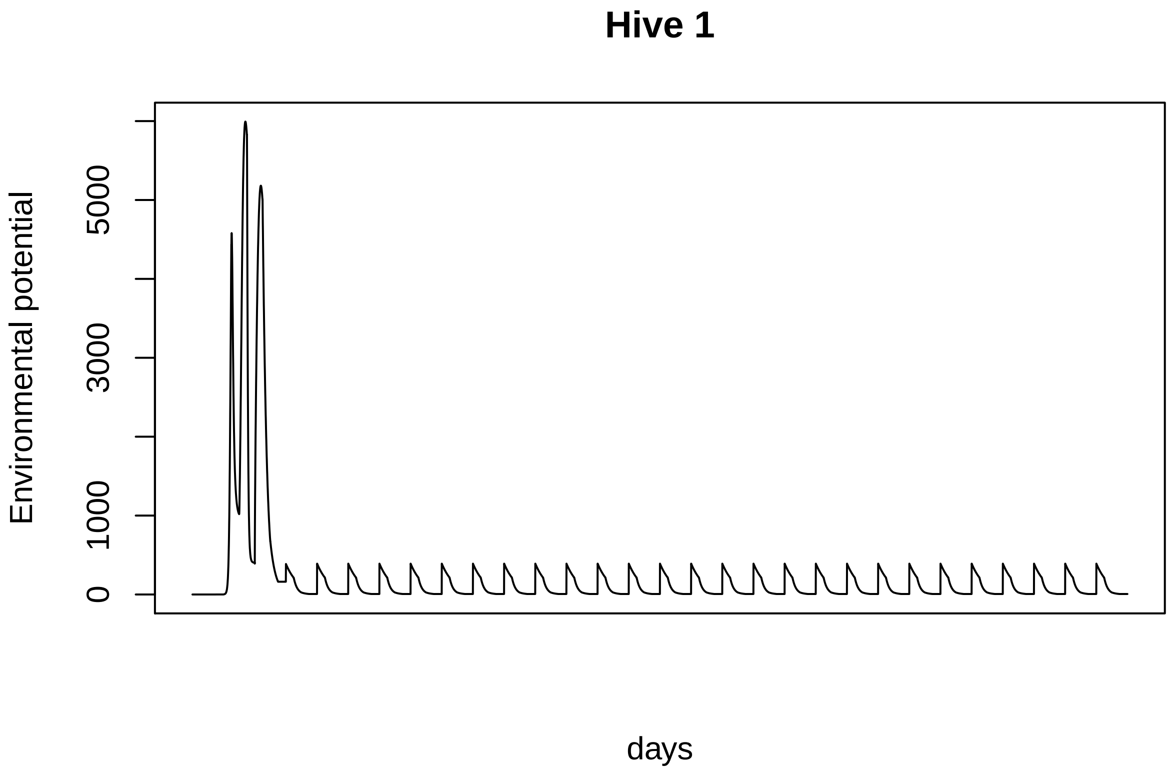


Fig3

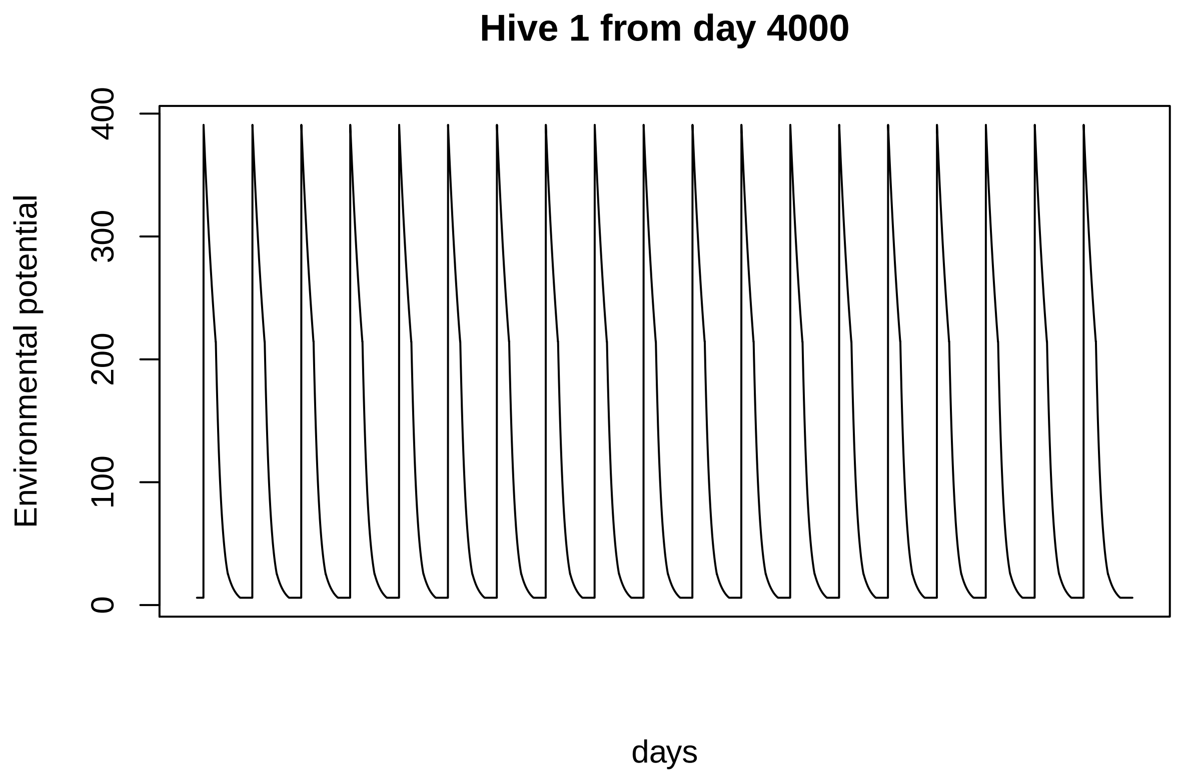

Hive 1 from day 4000
Environmental potential
0
100
200
300
400
days

Fig4

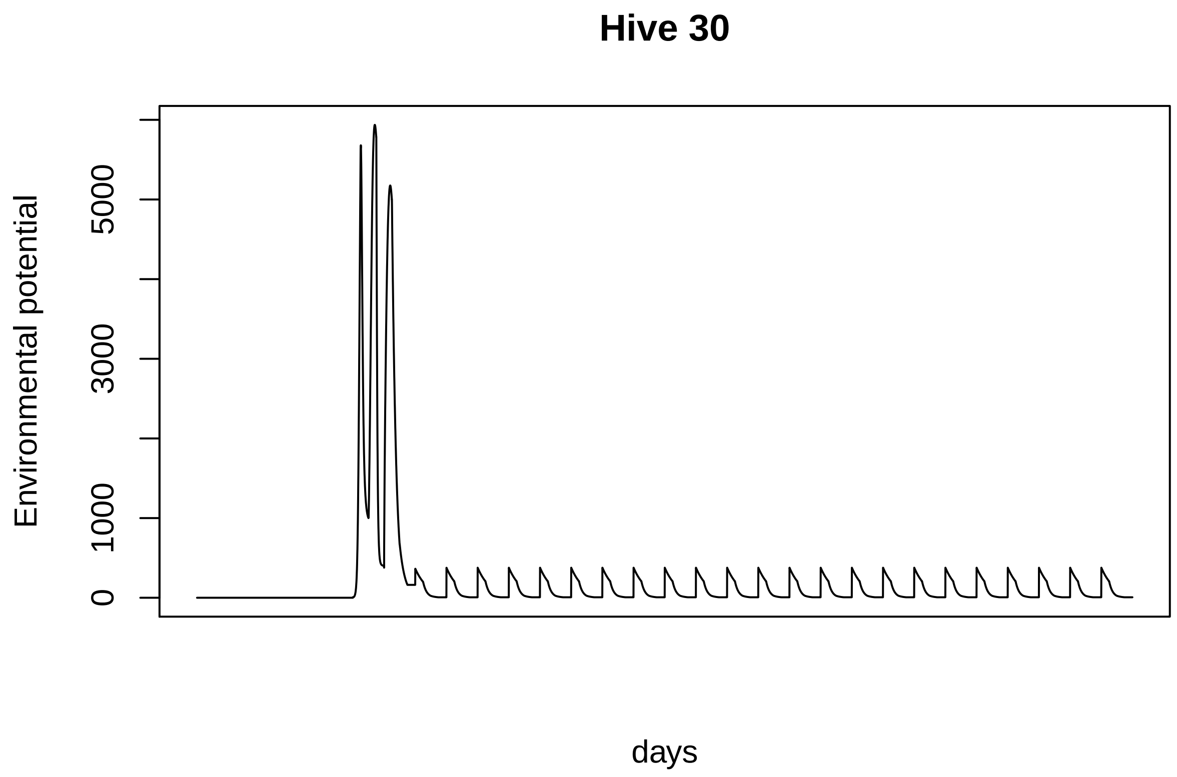

Hive 30
Environmental potential
0
1000
3000
5000
days

Fig3

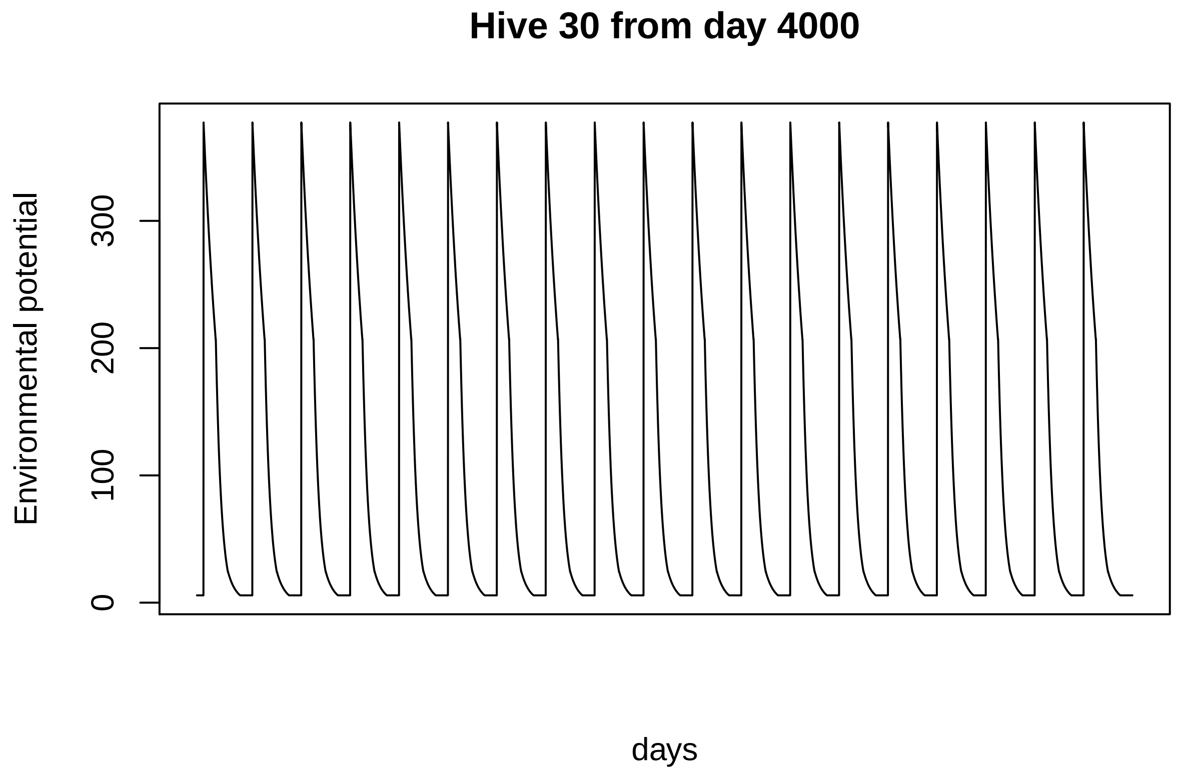

Hive 30 from day 4000
Environmental potential
0
100
200
300
days

Fig4

In the second simulation we look at hives number thirteen, the initially infected hive, and its neighbor hive fifteen. We again see the same patterns arise in both hives reaching a maximum value of about 400, but a key distinction being that the fifteenth hive is infected way sooner than was the case in the previous simulation with hive thirty.

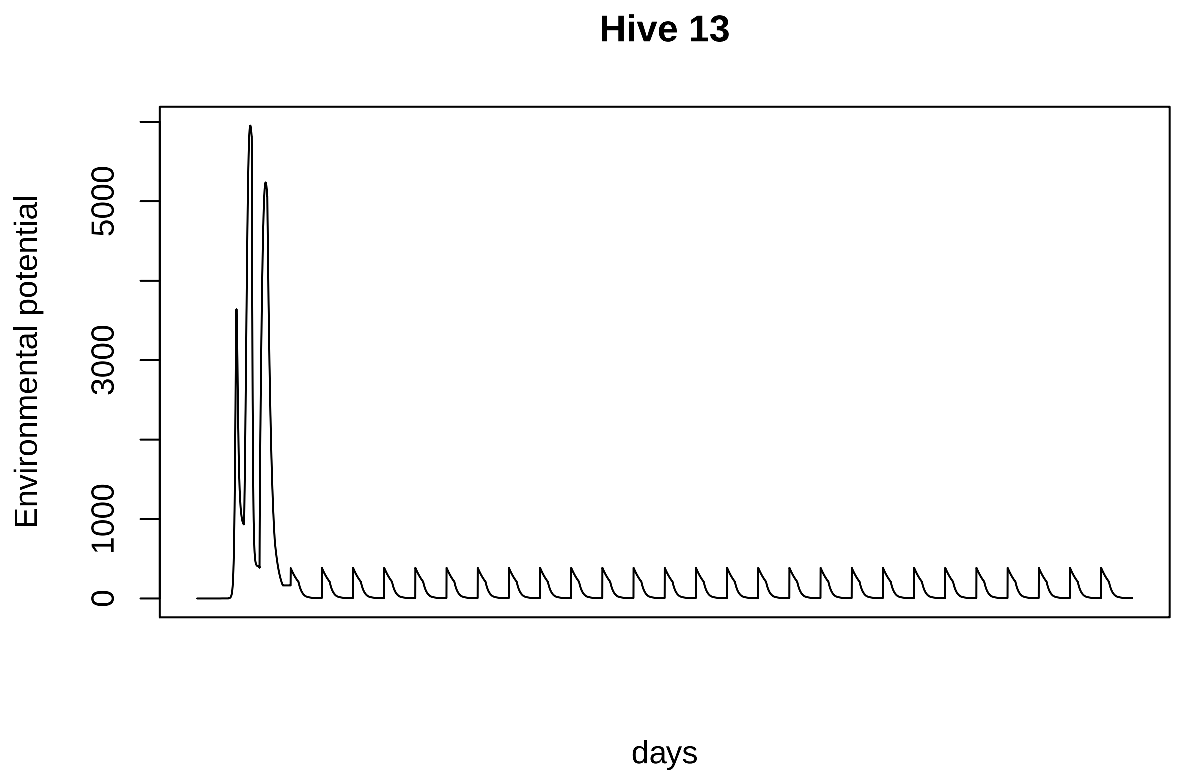


Fig5

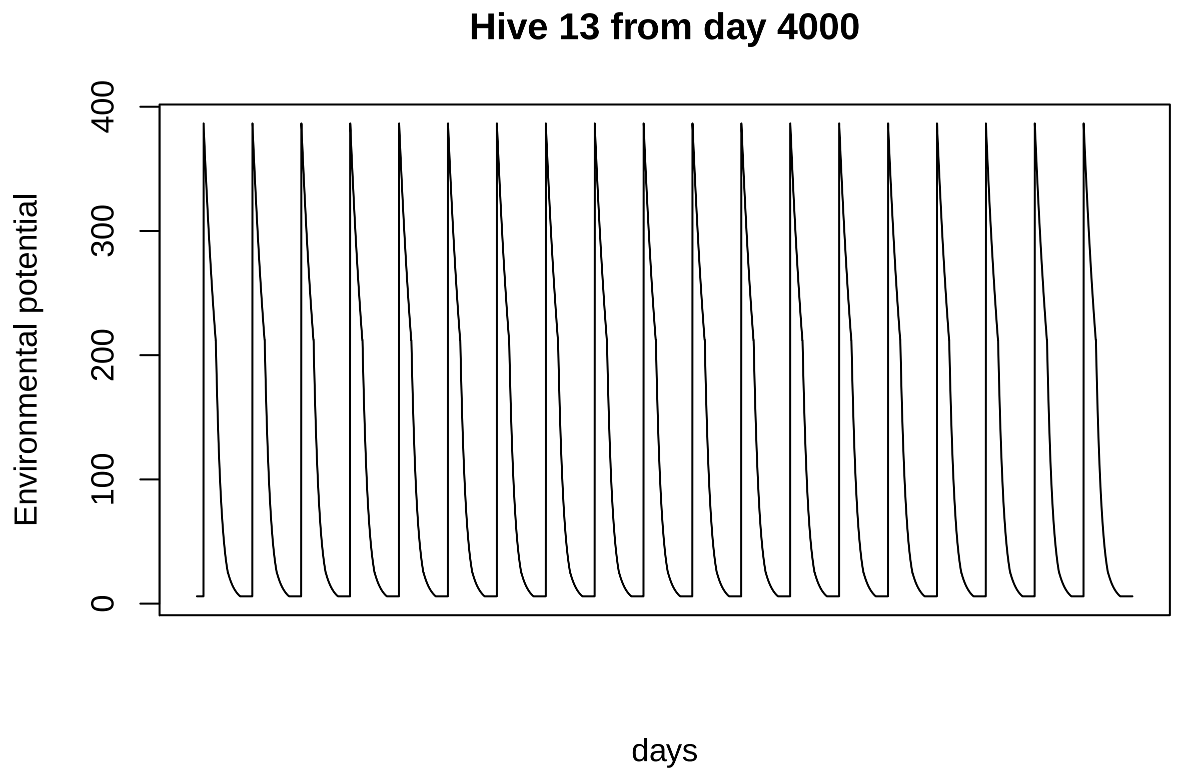
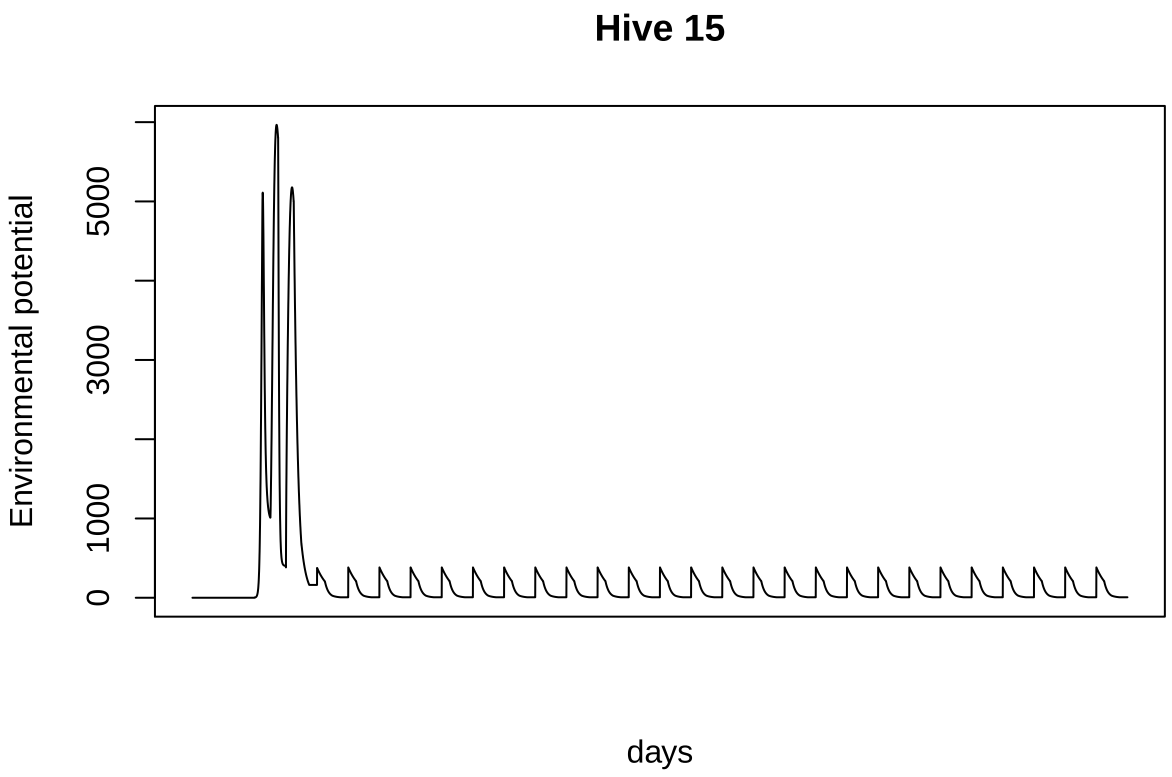
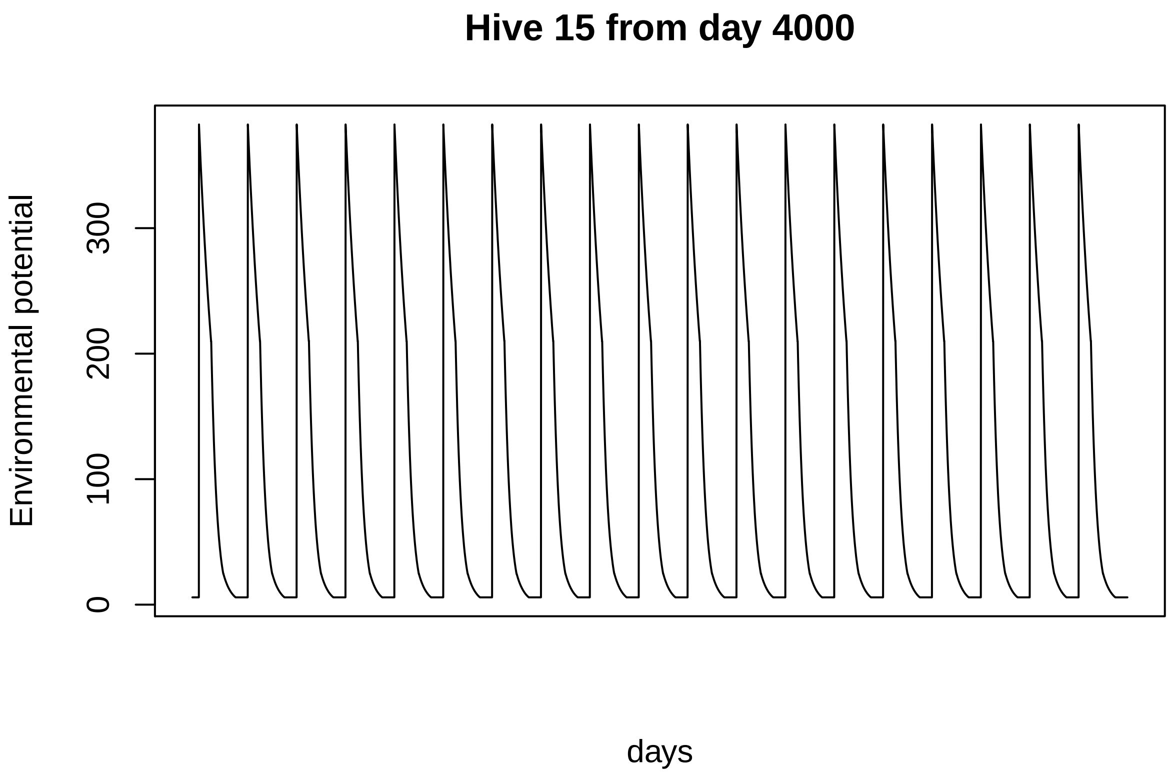


Fig6



Hive 15
Environmental potential
0
1000
3000
5000
days

Fig5



Hive 15 from day 4000
Environmental potential
0
100
200
300
days

Fig6

4 conclusion

Looking at the way the simulations come to a periodic solution in which the pathogen arises in the spring and nearly disappears in the winter, this suggests that the infestation does not trigger the hives to fail completely but they just get weaker. Much like the flu does in the human population.

The second conclusion we can draw from the model is that the propagation of the fungus when a center hive is infected versus when a corner or a hive at the edge is infected is significantly faster. This can be validated by the fact that only neighboring hives infect each other, so a hive in the middle having five more neighbors than a corner hive has a more than two fold range causing the pathogen to expand more rampant across the apiary.

The final conclusion the model comes to is that having neighboring hives also infected with the pathogen does not cause a higher plateau value in the eventual periodic solution. This tells us that there are no long term effects of which hive was infected first, the hives all go through the same pattern when infected, this can also be said for the size of an apiary.

5 discussion

The main critique on the original paper from our side is the limit function in the formula calculating Ek. It is very unclear how this function should be implemented in the differential equation solver. We would have loved to contact the authors of the paper about this, but due to time constraints this was not an option anymore. We in turn found a work around that was satisfyïng enough using a max value with a moderate cutoff.

The main assumption that the model holds is that the infection of neighboring hives only occurs once a year with direct neighbors. It seems weird that the hives would only contaminate other hives during hive maintenance, and then only their neighboring hives. Because when performing hive maintenance, the action of the beekeeper cleaning and emptying the hives, the actual physical location of the hive would not have an impact on contamination but rather the order in which hives are maintained. Also there is no scenario in which bees infect other hives on their own, not a far-fetched idea.

There is also a critique to be had on our implementation of the model, mainly the unexplainable sudden rise and fall of the Ei value in our plots. This is most certainly due to a programmatic error on our side. Also because of time constraints there is no possibility for us to assess this problem, but we presume the rest of the model is correctly implemented because it shows similar results as the original paper.

6 rcode

6.1 Base model

The implementation of the base model in the desolve ode function, characters had to be changes in order for R studio to handle them.

```r
formula <- function(t, y, parms) {
    with(as.list(c(y, parms)), {

        Hi <- Hi0 + Hi1
        Fi <- Fi0 + Fi1
        Zi <- Hi + Fi

        dHi0 = b * (((Zi)^n)/(k^n + Zi^n)) - rho1 * Hi0 + rho2 * (Fi/Zi) * Fi0 -
            nn0 * Hi0 - alpha1 * Hi0 * (Ei/(lambda + Ei))

        dHi1 = -rho1 * Hi1 + rho2 * (Fi/Zi) * Fi1 - nn1 * Hi1 + alpha1 * Hi0 * (Ei/(lambda +
            Ei))

        dFi0 = rho1 * Hi0 - (rho2 * (Fi/Zi) + eps0) * Fi0

        dFi1 = rho1 * Hi1 - (rho2 * (Fi/Zi) + eps1) * Fi1

        dEi <- y * Hi1 - sumy * Ei - alphat * Hi * (Ei/(lambda + Ei))

        return(list(c(dHi0, dHi1, dFi0, dFi1, dEi)))
    })
}
```

6.2 Apiary setup

This function creates the grid in which the neigtbors of each hive are determined.

```r
n1 <- 6  #length dimension of apiary grid
n2 <- 5  #width dimension of apiary grid

total <- n1 * n2  #total number of hives
hives <- seq(1, total)  #create a sequence of the totaal amount of hives

apiary_creation <- function(n1, n2, total, hives) {
    hive <- 1
    # initiate new dictionairy object
    apiary <- Dict$new(temp = 0)

    while (hive <= length(hives)) {
        # determine the neigbors of the current hive
        neighbors <- c(hive - 1, hive + 1, hive - n1, hive + n1, hive - n1 - 1, hive -
            n1 + 1, hive + n1 - 1, hive + n1 + 1)
        neighbors <- neighbors[neighbors > 0]
        neighbors <- neighbors[neighbors <= total]
        # remove when when hive is on the left edge
        if (hive%%n1 == 1) {
            neighbors <- neighbors[neighbors != hive + n1 - 1]
            neighbors <- neighbors[neighbors != hive - 1]
            neighbors <- neighbors[neighbors != hive - n1 - 1]
        }
        # remove when when hive is on the right edge
        if (hive%%n1 == 0) {
            neighbors <- neighbors[neighbors != hive + 1]
            neighbors <- neighbors[neighbors != hive - n1 + 1]
            neighbors <- neighbors[neighbors != hive + n1 + 1]
        }
        apiary[as.character(hive)] <- neighbors
        hive <- hive + 1
    }
    apiary$remove("temp")
    return(apiary)
}
# store the dictionairy in the apiary object
apiary <- apiary_creation(n1, n2, total, hives)
```

6.3 Yearly hive maintenance

Implementation of the formula describing the yearly hive maintenance.

```
p <- 0.01  #The p parameter value

ev_simulation <- function(p, apiary, Ek, year) {
    k <- 1
    # New vector to hold all the generated Ek values
    Ek_updated <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0)

    while (k <= apiary$length) {
        neighbour.ek <- Ek[apiary$get(k)]  #get the neigbors of the k'th hive
        x <- (1 - length(apiary$get(k)) * p) * Ek[k] + p * sum(neighbour.ek)  #run the formula
        x <- as.numeric(x)  #convert string to number
        if (year > 1) {
            # When already run, retake the previous from the hives.ek dataframe
            prev.dEk <- as.vector(hives.ek[1:year, k])
            EKs <- append(prev.dEk, x)
        } else {
            EKs <- x
        }
        EKs <- EKs[!is.na(EKs)]  #remove NAs
        dEk <- quantile(Re(EKs), 0.99, na.rm = TRUE)  #alternative implementation of lim function
        Ek_updated[k] <- dEk  #Put the new Ek value in the updated vector

        k <- k + 1
    }
    return(Ek_updated)
}
```

6.4 general formula running the base model for a year

This function runs the base model for a year with differing parameters. We made two versions of this function, this one was used to simulate the infection of hive one. The function in which hive thirteen was infected differs slightly.

```r
gen_formula_infected <- function(state, first_winter, hive) {

    if (first_winter == 1) {
        # in the first three seasons we want to use the base states
        state <- c(Hi0 = 10^4, Hi1 = 0, Fi0 = 10^4, Fi1 = 0, Ei = 0)
    }
    # run the simulation for the spring
    out.spring <- as.data.frame(ode(times = times, y = state, parms = parameters.spring,
        func = formula, method = "euler"))

    if (first_winter == 1) {
        state <- c(Hi0 = 10^4, Hi1 = 0, Fi0 = 10^4, Fi1 = 0, Ei = 0)
    } else {
        state <- c(Hi0 = out.spring[nrow(out.spring), 2], Hi1 = out.spring[nrow(out.spring),
            3], Fi0 = out.spring[nrow(out.spring), 4], Fi1 = out.spring[nrow(out.spring),
            5], Ei = out.spring[nrow(out.spring), 6])
    }
    # run the simulation for the summer
    out.summer <- as.data.frame(ode(times = times, y = state, parms = parameters.summer,
        func = formula, method = "euler"))

    if (first_winter == 1) {
        state <- c(Hi0 = 10^4, Hi1 = 0, Fi0 = 10^4, Fi1 = 0, Ei = 0)
    } else {

        state <- c(Hi0 = out.summer[nrow(out.summer), 2], Hi1 = out.summer[nrow(out.summer),
            3], Fi0 = out.summer[nrow(out.summer), 4], Fi1 = out.summer[nrow(out.summer),
            5], Ei = out.summer[nrow(out.summer), 6])
    }
    # run the simulation for the fall
    out.fall <- as.data.frame(ode(times = times, y = state, parms = parameters.fall,
        func = formula, method = "euler"))

    if (first_winter + hive == 2) {

        state <- c(Hi0 = 10^4, Hi1 = 10, Fi0 = 10^4, Fi1 = 0, Ei = 0)
    } else {
        state <- c(Hi0 = out.fall[nrow(out.fall), 2], Hi1 = out.fall[nrow(out.fall),
            3], Fi0 = out.fall[nrow(out.fall), 4], Fi1 = out.fall[nrow(out.fall),
            5], Ei = out.fall[nrow(out.fall), 6])
    }
    # run the simulation for the winter
    out.winter <- as.data.frame(ode(times = times, y = state, parms = parameters.winter,
        func = formula, method = "euler"))

    # add a column with time
    out.summer$time <- out.summer$time + (364/4)
    out.fall$time <- out.fall$time + 2 * (364/4)
    out.winter$time <- out.winter$time + 3 * (364/4)

    out.year <- rbind(out.spring, out.summer, out.fall, out.winter)  #combine all the results of the se
    return(list(out.year, out.winter))
}
```

6.5 Running the simulation

Here we also have multiple versions, in this one the Ei values for the first hive are stored for plotting.

```
# first initialize the parameters for ech season
parameters.spring <- c(b = 500, k = 8000, rho1 = 0.25, rho2 = 1.5, nn0 = 0, nn1 = 0,
    eps0 = 0.08511, eps1 = 0.16936, alpha1 = 0.55, y = 0.1, sumy = 0.00657, alphat = 0.14,
    lambda = 10000, n = 2, p = 1)
parameters.summer <- c(b = 1500, k = 12000, rho1 = 0.25, rho2 = 1.5, nn0 = 0, nn1 = 0,
    eps0 = 0.08511, eps1 = 0.16936, alpha1 = 0.12, y = 0.1, sumy = 0.0233, alphat = 0.14,
    lambda = 10000, n = 2, p = 0)
parameters.fall <- c(b = 500, k = 8000, rho1 = 0.25, rho2 = 1.5, nn0 = 0, nn1 = 0,
    eps0 = 0.08511, eps1 = 0.16936, alpha1 = 0.24, y = 0.1, sumy = 0.015683, alphat = 0.14,
    lambda = 10000, n = 2, p = 0)
parameters.winter <- c(b = 0, k = 6000, rho1 = 0, rho2 = 1.5, nn0 = 0.00649, nn1 = 0.00649,
    eps0 = 0, eps1 = 0, alpha1 = 0, y = 0.0088, sumy = 0, alphat = 0.14, lambda = 10000,
    n = 2, p = 0)

out.years <- data.frame()  #create the out.years dataframe

index <- 0

Ek <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0)  #initial values for Ek

# create multiple dataframes for storage of the data, named accordingly
hives.state <- data.frame(matrix(ncol = 30, nrow = 30 * 5))
hives.ek <- data.frame(matrix(ncol = 30, nrow = 30))
hives.ei <- data.frame(matrix(ncol = length(hives)))
names(hives.ei)[1] <- "Ei"

# loop for thirty years
while (year < 30) {
    # update vavlues
    year <- year + 1
    hive <- 1
    # loop for all the hives
    while (hive <= length(Ek)) {
        # for the first year define the state
        if (year == 1) {
            winter.state <- c(Hi0 = 10^4, Hi1 = 0, Fi0 = 10^4, Fi1 = 0, Ei = Ek[hive])
        } else {
            # if not the first year, fetch the last used state
            winter.state <- c(Hi0 = hives.state[(year - 2) * 5 + 1, hive], Hi1 = hives.state[(year -
                2) * 5 + 2, hive], Fi0 = hives.state[(year - 2) * 5 + 3, hive], Fi1 = hives.state[(year
                2) * 5 + 4, hive], Ei = hives.ek[year - 1, hive])
        }
```

```r
        # Run the base model and store the wanted data in a datframe
        out.year <- as.data.frame(gen_formula_infected(winter.state, year, hive)[1])

        # if discussing the first hive, hold the Ei values for plotting
        if (hive == 1) {
            temp.ei <- as.data.frame(out.year$Ei)
            names(temp.ei)[1] <- "Ei"
            hives.ei <- rbind(hives.ei, temp.ei)
        }

        # Fetch the data from out.year to create a new state for the ode solver
        x <- 1 + ((year - 1) * 5)
        y <- 5 + ((year - 1) * 5)
        hives.state[x:y, hive] <- c(out.year[nrow(out.year), 2], out.year[nrow(out.year),
            3], out.year[nrow(out.year), 4], out.year[nrow(out.year), 5], out.year[nrow(out.year),
            6])

        hive <- hive + 1
    }
    # at the end of the year, run the yearly hive maintenance and store the
    # values appropriately
    hives.ek[year, ] <- ev_simulation(0.01, apiary, hives.state[(year) * 5, ], year)
}
```

7 References

https://beeaware.org.au/archive-pest/nosema/#ad-image-0 [1]
https://www.sciencedirect.com/topics/medicine-and-dentistry/
nosemosis#:~:text=Nosemosis%20symptoms%20include%20lifespan%20reduction%2C%20lethargic%20behavior%2C%20and
[2] https://link.springer.com/chapter/10.1007/978-3-319-99719-3_35 [3]