UNCERTAINTY QUANTIFICATION OF PLANT DISEASE DYNAMICS USING SIR MODEL-V

AUTHOR
Group GACM41000 45

Introduction

We analyze plant disease dynamics using an **SIR model with host response** (Model V). The system is defined by:

Disease Dynamics Equations

$$\frac{dS}{dt} = b(\kappa - N) - \lambda(t)S - f(I,S) \qquad \text{(Change in susceptible stems)}$$
$$\frac{dI}{dt} = \lambda(t)S - dI \qquad \text{(Change in infected stems)}$$
$$\frac{dR}{dt} = dI \qquad \text{(Change in removed stems)}$$

Component Functions

$$f(I,S) = \frac{\alpha I}{\gamma + I + S} \qquad \text{(Host response function)}$$
$$\lambda(t) = \lambda_0 e^{-\mu t} \qquad \text{(Time-varying infection rate)}$$
$$N = S + I + R \qquad \text{(Total population)}$$

Model Parameters

Symbol	Description
b	Stem production rate
κ	Carrying capacity
λ_0	Initial infection rate
μ	Infection rate decay parameter
α	Host response parameter
γ	Saturation constant
d	Removal rate

1. Define the Model V ODE System

In this section we define the ODE model based on the paper and fit the model based on the equations

$$f(I,S) = \frac{\alpha I}{\gamma + I + S} \qquad \text{(Host response function)}$$

$$\lambda(t) = \lambda_0 e^{-\mu t} \qquad \text{(Time-varying infection rate)}$$

$$N = S + I + R \qquad \text{(Total population)}$$

This section defines the mathematical model used to simulate the disease dynamics in Model V. The function sir_model_v calculates how the number of susceptible (S), infected (I), and removed (R) individuals change over time. Here's how each part of the function works:

- ***N <- S + I + R:*** *Computes the total population at any time.*
- *l**ambda <- lambda0 * exp(-mu * time)**: Defines the infection rate that decreases over time.*
- ***f <- (alpha * I) / (gamma + I + S):*** *Represents how the infection pressure increases based on current infected and susceptible individuals.*
- ***dS <- b * (kappa - N) - lambda * S - f**: Calculates how the number of susceptible individuals changes.*
- ***dI <- lambda * S - d * I:*** *Calculates how the number of infected individuals changes.*
- ***dR <- d * I:*** *Calculates how the number of removed individuals change*

```
# Define the model (Model V)
sir_model_v <- function(time, state, parameters) {
  with(as.list(c(state, parameters)), {
    N <- S + I + R
    lambda <- lambda0 * exp(-mu * time)
    f <- (alpha * I) / (gamma + I + S)
    dS <- b * (kappa - N) - lambda * S - f
    dI <- lambda * S - d * I
    dR <- d * I
    return(list(c(dS, dI, dR)))
  })
}
```

2.Defining the Parameters from Table and Setting the Initial conditions:

- We define the **initial values of S, I, and R as zero,** meaning no individuals are infected or removed at the beginning.

- We then **Set the parameter values for Model V under high density**, using values given in **Table 2 of the article**.
- We create a **time sequence from 0 to 14 (in steps of 0.1) representing thermal time.**
- *Using the ode function, we solve the system of differential equations (Model V) with these values to simulate how S, I, and R change over time.*
- The result is saved as a data frame.

```
# Initial conditions
state <- c(S = 0, I = 0, R = 0)

# Parameter values from Table 2 for Model V (high density)
parameters <- c(
  b = 1.467,
  kappa = 5.743,
  lambda0 = 0.499,
  mu = 0.247,
  alpha = 2.106,
  gamma = 0.026,
  d = 0.058
)

# Time points (thermal time in degree days)
times <- seq(0, 14, by = 0.1)

# Solve the ODE
out <- as.data.frame(ode(y = state, times = times, func = sir_model_v, parms = parameters))

head(out)
```

3.Plot the Simulated Results

- This plot shows the simulation results for Model V under high-density conditions.
- The **blue line represents the number of susceptible stems (S)** over time, while the red line shows the number of infected stems (I).
- The **x-axis represents thermal time in degree-days, and the y-axis shows stem count**.
- This visual helps us understand how the infection spreads through the population over time.

```
# Plot the results
plot(out$time, out$S, type = "l", col = "blue", ylim = c(0,6), ylab = "Stem Count", xlab = "Therma
lines(out$time, out$I, col = "red")
legend("topright", legend = c("Susceptible (S)", "Infected (I)"), col = c("blue", "red"), lty = 1
```

4.Fitting the Observed Data:

- *In this section, we create a data frame with estimated values for susceptible (S) and infected (I) stem counts at specific time points.*
- *These values were visually approximated from Figure 2 of the article .*
- *The time is recorded in thermal time (degree-days).*
- *This observed data will later be used to compare against our model's predictions and to fit the model parameters*.

```
# Your estimated data (thermal time and observed S, I)
data_obs <- data.frame(
  time = c(0, 2.3505, 3.6495, 5.0103, 6.1237, 7.2371, 11.6907),
  I_obs = c(0, 3.2836, 1.4030, 1.0149, 1.0149, 1.0149, 1.1045),
  S_obs = c(0, 1.5958, 2.8119, 3.2122, 3.1145, 2.8975, 1.9895)
)
```

5.Defining the objective Function and Optimizing the Parameters with optim:

The function **rss_function(par)** is defined to take the **model parameters (par) as input**, where these parameters represent the key variables in Model V, such as the **infection rate (lambda0), transmission rate (b), and otherparameters.**

Within The function;

- Compute the predicted values for susceptible (S) and infected (I) over time using the ode solver with the current parameters.
- Calculate the difference (error) between observed and predicted values for both susceptible and infected individuals at each time point:
  - ***rss_S: The squared differences for the susceptible data.***
  - ***rss_I: The squared differences for the infected data.***
- Sum the squared errors for both susceptible and infected groups to get the total RSS

Optimizing the Parameters with optim:

- The optim function is used to minimize the RSS.
  - **Initial Guess (init_guess)**: We provide an initial set of parameter estimates, which are based on values from the literature (Table 2).

- After Which it uses **L-BFGS-B method,** which is an optimization technique suitable when parameters can't be negative.
- Then we **set the lower bounds and the max iterations** for the optimization Process.

The best-fit parameters (those that minimize the RSS) are extracted from the fit$ par object. The final RSS value (fit$ value) gives us a measure of how well the model fits the data: a lower RSS means a better model fit

```
# Objective function: sum of squared errors
rss_function <- function(par) {
  names(par) <- c("b", "kappa", "lambda0", "mu", "alpha", "gamma", "d")
  state <- c(S = 0, I = 0, R = 0)
  times <- data_obs$time
  out <- as.data.frame(ode(y = state, times = times, func = sir_model_v, parms = par))
  pred <- out[, c("S", "I")]
  rss_S <- sum((data_obs$S_obs - out$S)^2)
  rss_I <- sum((data_obs$I_obs - out$I)^2)
  rss <- sum((data_obs$S - pred$S)^2 + (data_obs$I - pred$I)^2)
  return(rss)
}
# Initial guess for the parameters
init_guess <- c(
  b = 1.467, kappa = 5.743, lambda0 = 0.499,
  mu = 0.247, alpha = 2.106, gamma = 0.026, d = 0.058
)

# Fit the model
fit <- optim(init_guess, fn = rss_function, method = "L-BFGS-B",
             lower = rep(0.0001, 7), control = list(maxit = 1000))

# Show fitted parameters
best_fit_params <- fit$par
print(best_fit_params, 5)

# Get RSS value
print(paste("Best RSS:", fit$value))
```

6.Analyzing the Model Fit:

- *This section calculates the* ***Residual Sum of Squares (RSS) separately for the susceptible (S) and infected (I) data,*** *using the* ***best-fit parameters obtained from the previous step.***
- ***The squared differences between the observed values of susceptible individuals (S_obs) and the predicted values from the model (output$S)*** *are summed up to calculate the RSS for* ***the susceptible group (rss_S).***

- *Similarly, the **RSS for infected individuals is computed by summing the squared differences between the observed infected data (I_obs) and the predicted values (output$**I). the cat function is used to print out the RSS values for both susceptible and infected groups, rounded to five decimal places for better clarity.*

```
# Cost function: returns RSS for S and I separately
state <- c(S = 0, I = 0, R = 0)
times <- data_obs$time
output <- as.data.frame(ode(y = state, times = times,
                            func = sir_model_v,
                            parms = best_fit_params))

rss_S <- sum((data_obs$S_obs - output$S)^2)
rss_I <- sum((data_obs$I_obs - output$I)^2)
cat("RSS for S:", round(rss_S, 5), "\nRSS for I:", round(rss_I, 5))
```

7.Comparision Between Model fit and the Observed Data:

In this section, we use ggplot2 to visualize the comparison between the model predictions and observed data:

- **Susceptible Stems (S):** *The blue line represents the model's predicted susceptible stems over time, while the black points show the observed values. The plot helps us assess the model's fit for susceptible stems.*
- **Infected Stems (I):** *Similarly, the red line shows the predicted infected stems, and the black points represent the observed data. This plot allows us to compare the model's prediction for infected stems with the actual data.*

```
library(ggplot2)

# Plot for Susceptible
p_s <- ggplot() +
  geom_line(data = output, aes(x = time, y = S), color = "blue", size = 1) +
  geom_point(data = data_obs, aes(x = time, y = S_obs), color = "black", shape = 17, size = 3) +
  labs(title = "Susceptible Stems: Model V vs Observed",
       x = "Time", y = "Susceptible (S)") +
  theme_minimal()
p_s
```

```
# Plot for Infected
p_i <- ggplot() +
  geom_line(data = output, aes(x = time, y = I), color = "red", size = 1) +
  geom_point(data = data_obs, aes(x = time, y = I_obs), color = "black", shape = 17, size = 3) +
  labs(title = "Infected Stems: Model V vs Observed",
       x = "Time", y = "Infected (I)") +
  theme_minimal()
p_i
```

8.Residual Comparision

We calculate and plot the residuals (the differences between observed and predicted values) for both susceptible (S) and infected (I) stems:

- **Residuals for Susceptible Stems (S):** *The blue plot shows the residuals for susceptible stems over time. This is calculated as the difference between observed values (S_obs) and predicted values (S). A smaller residual indicates a better model fit.*
- **Residuals for Infected Stems (I):** *The red plot shows the residuals for infected stems, calculated similarly. These residuals help us evaluate how well the model predicts the infected population at each time point.*

```
residuals_S <- data_obs$S_obs - output$S
residuals_I <- data_obs$I_obs - output$I
plot(data_obs$time, residuals_S, type="b", col="blue", main="Residuals: S", ylab="Residual", xlab=
```

```
plot(data_obs$time, residuals_I, type="b", col="red", main="Residuals: I", ylab="Residual", xlab="
```

9.Sensitivity Analysis

- In this section, we perform a local sensitivity analysis to measure how changes in each model parameter affect the model outputs:
  - We use the finite difference method, slightly **increasing one parameter at a time** (by a small value ε = 0.001).
  - We **compare the output with the original model to compute how much S and I change** due t**o each parameter.**

```
# Sensitivity analysis for each parameter using finite differences
sensitivity_analysis <- function(base_params, param_names, epsilon = 1e-3) {
  base_state <- c(S = 0, I = 0, R = 0)
  times <- seq(0, 14, by = 0.1)
  base_out <- as.data.frame(ode(y = base_state, times = times, func = sir_model_v, parms = base_pa

  sensitivities <- list()

  for (pname in param_names) {
    perturbed <- base_params
    perturbed[pname] <- perturbed[pname] + epsilon
    perturbed_out <- as.data.frame(ode(y = base_state, times = times, func = sir_model_v, parms =

    dS <- (perturbed_out$S - base_out$S) / epsilon
```

```
    dI <- (perturbed_out$I - base_out$I) / epsilon

    sensitivities[[pname]] <- data.frame(time = times, dS = dS, dI = dI)
  }

  return(sensitivities)
}

# Run sensitivity analysis
param_names <- names(best_fit_params)
sensitivity_results <- sensitivity_analysis(best_fit_params, param_names)
```

10.Plotting Sensitivity Results

- For each parameter, we show how small changes affect:
  - ***Susceptible stems (S) – in blue***
  - ***Infected stems (I) – in red***
- These plots show the rate of change of S and I with respect to each parameter over time (∂S/∂θ and ∂I/∂θ).

```
# Plot sensitivity for Susceptible stems
library(tidyr)

for (pname in names(sensitivity_results)) {
  df <- sensitivity_results[[pname]]
  plot(df$time, df$dS, type = "l", col = "blue", lwd = 2,
       main = paste("Sensitivity of S to", pname),
       ylab = "∂S/∂θ", xlab = "Time")
}
```

```
# Plot sensitivity for Infected stems
for (pname in names(sensitivity_results)) {
  df <- sensitivity_results[[pname]]
  plot(df$time, df$dI, type = "l", col = "red", lwd = 2,
       main = paste("Sensitivity of I to", pname),
       ylab = "∂I/∂θ", xlab = "Time")
}
```

11.Maximum Sensitivity Summary Plot

- This bar plot summarizes the maximum sensitivity of each model output (Susceptible (S) and Infected (I)) to every model parameter:

- ***For each parameter, we show the highest absolute sensitivity value across all time points.***
- ***The taller the bar, the more impact that parameter has on the model output.***

```
# Extract max absolute sensitivities for each parameter
library(tibble)

summary_df <- tibble(
  Parameter = names(sensitivity_results),
  Max_dS = sapply(sensitivity_results, function(df) max(abs(df$dS))),
  Max_dI = sapply(sensitivity_results, function(df) max(abs(df$dI)))
)

# Melt for ggplot
library(tidyr)
summary_long <- pivot_longer(summary_df, cols = c(Max_dS, Max_dI), names_to = "Output", values_to

# Rename for nicer plot
summary_long$Output <- recode(summary_long$Output, Max_dS = "Susceptible (S)", Max_dI = "Infected

# Plot
library(ggplot2)
ggplot(summary_long, aes(x = Parameter, y = MaxSensitivity, fill = Output)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Maximum Sensitivity of Model Outputs to Parameters",
       y = "Max |∂Output/∂Parameter|",
       x = "Model Parameter") +
  theme_minimal() +
  scale_fill_manual(values = c("steelblue", "tomato"))
```

12.Residual Analysis of Model Fit

- **This plot shows the residuals, which are the differences between the observed data points and the model predictions:**
  - **Residuals for Susceptible (S) and Infected (I)** are shown as **separate colored points.**
  - A dashed horizontal line at 0 represents **perfect model fit.**
  - Points above or below this line indicate **overestimation or underestimation** by the model.

```
# Create residuals in tidy format
library(tidyr)

residuals_df <- data_obs %>%
  left_join(output, by = "time") %>%
  mutate(
    S_residual = S_obs - S,
```

```
    I_residual = I_obs - I
  ) %>%
  select(time, S_residual, I_residual) %>%
  pivot_longer(cols = c(S_residual, I_residual),
               names_to = "Compartment", values_to = "Residual") %>%
  mutate(Compartment = recode(Compartment,
                              S_residual = "Susceptible (S)",
                              I_residual = "Infected (I)"))

# Plot residuals
ggplot(residuals_df, aes(x = time, y = Residual, color = Compartment)) +
  geom_point(size = 3) +
  geom_hline(yintercept = 0, linetype = "dashed") +
  labs(title = "Residuals – Model V Fit",
       x = "Time",
       y = "Observed - Model") +
  theme_minimal()
```

13.Sensitivity to Parameter b – Stem Production Rate

- we tested how changing the parameter b (which controls how fast new stems are made) affects the results of our model. We used three different values of b — a smaller one, the original best-fit one, and a larger one — and looked at how the number of Susceptible (S) and Infected (I) stems changed over time.
    - When **b is small, fewer new stems are create**d, so the values of S and I stay lower.
    - When **b is large, more stems are created**, so both S and I grow faster and reach higher numbers.
    - This shows that **b is a sensitive parameter** — *small changes to its value can make a big difference in the results*

```
# Values of 'b' to test around the base value (Model V high density value: 1.467)
b_values <- c(1.0, 1.467, 1.9)
colors <- c("red", "green", "blue")

# Store results
results_b <- list()

# Time points
times <- seq(0, 14, by = 0.1)

# Loop through b values
for (b in b_values) {
  params_b <- best_fit_params
  params_b["b"] <- b

  out_b <- as.data.frame(ode(y = c(S = 0, I = 0, R = 0),
```

```
                              times = times,
                              func = sir_model_v,
                              parms = params_b))
  out_b$b_value <- b
  results_b[[as.character(b)]] <- out_b
}

# Combine into one data frame
sensitivity_b_df <- bind_rows(results_b)

# Plot for Susceptible (S)
ggplot(sensitivity_b_df, aes(x = time, y = S, color = factor(b_value))) +
  geom_line(size = 1) +
  labs(title = "Sensitivity to b (Stem Production Rate)",
       subtitle = "Effect of b on Susceptible (S)",
       x = "Time", y = "Susceptible",
       color = "b value") +
  theme_minimal()
```

```
# Plot for Infected (I)
ggplot(sensitivity_b_df, aes(x = time, y = I, color = factor(b_value))) +
  geom_line(size = 1) +
  labs(title = "Sensitivity to b (Stem Production Rate)",
       subtitle = "Effect of b on Infected (I)",
       x = "Time", y = "Infected",
       color = "b value") +
  theme_minimal()
```