# Modeling Demographic Stochasticity and Thermal Mismatch in Host-Pathogen Wildlife Systems

Modeling Code

Andy Carlino

2025-01-24

## Setup:

Working Directory and Libraries:

```
# Working Directory
setwd("C:\\Users\\andre\\OneDrive\\Documents\\GitHub\\DemoStoch_ThermMismatch_EpiModels")

# Libraries
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.4.4     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(tidyr)
library(ggplot2)
library(stats)
library(reshape2)
```

```
Attaching package: 'reshape2'

The following object is masked from 'package:tidyr':

    smiths
```

```
library(grid)
library(gridExtra)
```

```
Attaching package: 'gridExtra'

The following object is masked from 'package:dplyr':

    combine
```

```
library(dplyr)
```

Source Functions:

```
# Tau leap function
source("tau_leap_function.R")

# Simulation of the within-host model function
source("model_simulation_function.R")
```

## Quadratic Functions for Parameter Values along Thermal Gradient:

Thermal Performance Curve using a Quadratic function:

$$\text{Parameter} = -q * [\text{temp} - t_{min}][\text{temp} - t_{max}]$$

Min-Max Normalization to get all parameters on 0-1 scale for easier visualization

$$\text{Normalized Parameter} = \frac{value - min}{max - min}$$

```
# Generate temperature range
temperatures <- seq(9, 16, length.out = 50)
```

Estimating the Range of Parameter Phi (Viral Replication Rate)

```
estimate_phi <- function(temperature, q_phi, tmin_phi, tmax_phi) {
  # Quadratic equation
  phi <- pmax(-q_phi * (temperature - tmin_phi) * (temperature - tmax_phi), 0)
  return(phi)
}

# Set q for phi
q_phi <- 0.015
# Set temperature range
tmin_phi <- 1
tmax_phi <- 30

# Calculate corresponding values of phi
phi_values <- estimate_phi(temperatures, q_phi, tmin_phi, tmax_phi)

# Min-Max normalization
norm_phi <- (phi_values - min(phi_values)) / (max(phi_values) - min(phi_values))
```

```r
# Peak value of the phi Parameter
temp_peak_phi <- (tmin_phi + tmax_phi) / 2
peak_phi <- estimate_phi(temp_peak_phi, q_phi, tmin_phi, tmax_phi)
```

Estimating the Range of Parameter Alpha (Mass-action Attack Rate)

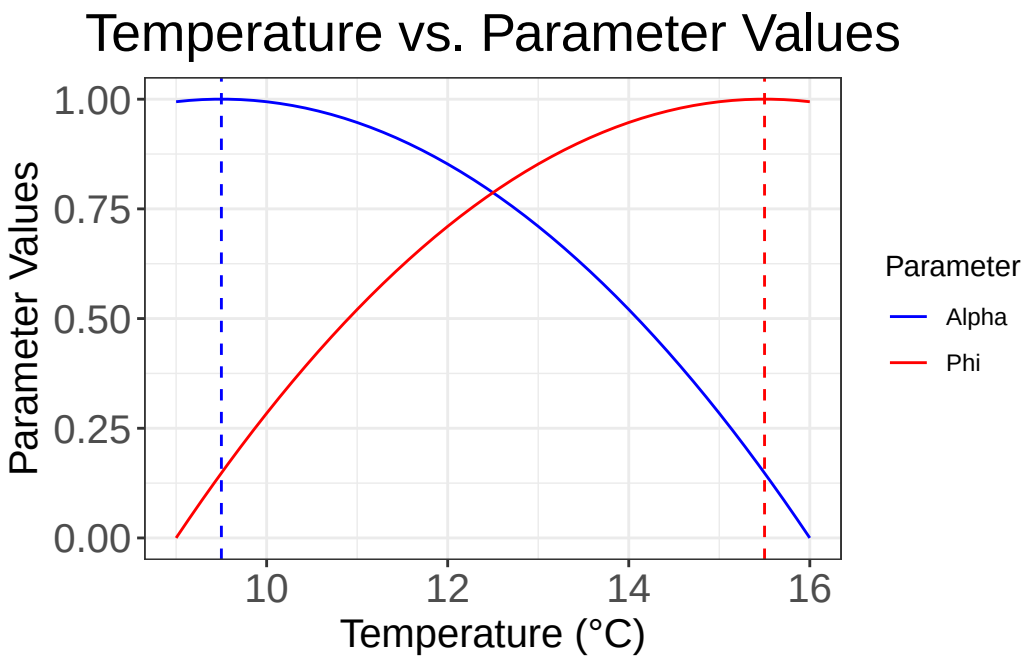
```r
estimate_alpha <- function(temperature, q_alpha, tmin_alpha, tmax_alpha){
  # Quadratic Equation
  alpha <- pmax(-q_alpha * (temperature - tmin_alpha) * (temperature - tmax_alpha), 0)
  return(alpha)
}

# Set q value for alpha
q_alpha <- 0.0085
# Set temperature range
tmin_alpha <- -5
tmax_alpha <- 24

# Calculate corresponding values of alpha
alpha_values <- estimate_alpha(temperatures, q_alpha, tmin_alpha, tmax_alpha)

# Min-Max Normalization
norm_alpha <- (alpha_values - min(alpha_values)) / (max(alpha_values) - min(alpha_values))

# Peak value of the alpha Parameter
temp_peak_alpha <- (tmin_alpha + tmax_alpha) / 2
peak_alpha <- estimate_alpha(temp_peak_alpha, q_alpha, tmin_alpha, tmax_alpha)
```

Estimating the Range of Parameter Psi (Immune Growth Rate in Response to Virus)

```r
estimate_psi <- function(temperature, q_psi, tmin_psi, tmax_psi){
  # Quadratic Equation
  psi <- pmax(-q_psi * (temperature - tmin_psi) * (temperature - tmax_psi), 0)
  return(psi)
}

# Set q value for psi
q_psi <- 0.0041
# Set temperature range
tmin_psi <- 1
tmax_psi <- 28

# Calculate corresponding values of alpha
psi_values <- estimate_psi(temperatures, q_psi, tmin_psi, tmax_psi)

# Min-Max normalization
norm_psi <- (psi_values - min(psi_values)) / (max(psi_values) - min(psi_values))

# Peak value of the alpha Parameter
temp_peak_psi <- (tmin_psi + tmax_psi) / 2
peak_psi <- estimate_psi(temp_peak_psi, q_psi, tmin_psi, tmax_psi)
```

Plot the functions:

```r
plot_data <- data.frame(Temperature = temperatures,
                        Phi = norm_phi,
                        Alpha = norm_alpha)
                        #Psi = psi_values)

# Plot Phi Values
phi_plot <- ggplot(plot_data, aes(x = Temperature, y = Phi)) +
  geom_line(color = "red") +
  labs(x = "Temperature (°C)", y = "Phi (Pathogen) Values") +
  geom_vline(xintercept = temp_peak_phi, color = "red", linetype = "dashed") +
  theme_bw() +
  ggtitle("Temperature vs. Parameter Values") +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(plot.title = element_text(size = 20)) +
  theme(axis.text = element_text(size = 15)) +
  theme(axis.title = element_text(size = 15))

# Plot Alpha Values
alpha_plot <- ggplot(plot_data, aes(x = Temperature, y = Alpha)) +
  geom_line(color = "blue") +
  labs(x = "Temperature (°C)", y = "Alpha (Host) Values") +
  geom_vline(xintercept = temp_peak_alpha, color = "blue", linetype = "dashed") +
  theme_bw() +
  theme(axis.text = element_text(size = 15)) +
  theme(axis.title = element_text(size = 15))

# Plot Psi Values (Commented out, but code can easily be adopted to look at this parameter if needed...)
#psi_plot <- ggplot(plot_data, aes(x = Temperature, y = Psi)) +
#  geom_line(color = "green") +
#  labs(x = "Temperature (C)", y = "Psi (Immune Response to Virus) Values") +
#  geom_vline(xintercept = temp_peak_psi, color = "green", linetype = "dashed") +
#  theme_bw() +
#  theme(axis.text = element_text(size = 15)) +
#  theme(axis.title = element_text(size = 15))


# To plot both parameters together with normalized values:
plot_data_long <- plot_data %>% pivot_longer(cols = c(Phi, Alpha), names_to = "Parameter", values_to = "Val

combined_plot <- ggplot(plot_data_long, aes(x = Temperature, y = Value, color = Parameter)) +
  geom_line() +
  geom_vline(xintercept = temp_peak_phi, color = "red", linetype = "dashed") + # For Phi peak
  geom_vline(xintercept = temp_peak_alpha, color = "blue", linetype = "dashed") + # For Alpha peak
  labs(x = "Temperature (°C)", y = "Parameter Values") +
  theme_bw() +
  ggtitle("Temperature vs. Parameter Values") +
  theme(plot.title = element_text(hjust = 0.5, size = 20),
        axis.text = element_text(size = 15),
        axis.title = element_text(size = 15)) +
  scale_color_manual(values = c("Phi" = "red", "Alpha" = "blue"))

combined_plot
```




### Testing Multiple Scenarios (High, Mid [multiple], Null)

Using this format, we're going to create and test different scenarios, incrementally closing the gap between thermal optima (by bringing tmin/tmax closer for each step), ultimately testing a 'null' scenario where thermal optima completely overlap (i.e., no thermal mismatch).

#### High Mismatch

```
# FIRST: Phi value for High Mismatch
# Set temperature range (High)
tmin_phi_high <- 1
tmax_phi_high <- 30

# Calculate corresponding values of phi
phi_values_high <- estimate_phi(temperatures, q_phi, tmin_phi_high, tmax_phi_high)

# Min-Max normalization
norm_phi_high <- (phi_values_high - min(phi_values_high)) / (max(phi_values_high) - min(phi_values_high))

# Peak value of the phi Parameter
temp_peak_phi_high <- (tmin_phi_high + tmax_phi_high) / 2
peak_phi_high <- estimate_phi(temp_peak_phi_high, q_phi, tmin_phi_high, tmax_phi_high)

# ------------------------------------------------
# NEXT: Alpha value for High Mismatch
# Set temperature range
tmin_alpha_high <- -5
tmax_alpha_high <- 24

# Calculate corresponding values of alpha
```

```r
alpha_values_high <- estimate_alpha(temperatures, q_alpha, tmin_alpha_high, tmax_alpha_high)

# Min-Max Normalization
norm_alpha_high <- (alpha_values_high - min(alpha_values_high)) / (max(alpha_values_high) - min(alpha_value

# Peak value of the alpha Parameter
temp_peak_alpha_high <- (tmin_alpha_high + tmax_alpha_high) / 2
peak_alpha_high <- estimate_alpha(temp_peak_alpha_high, q_alpha, tmin_alpha_high, tmax_alpha_high)
```

**Mid 1 Mismatch**

```r
# FIRST: Phi value for Mid 1 Mismatch
# Set temperature range
tmin_phi_mid1 <- 0.25
tmax_phi_mid1 <- 29.25

# Calculate corresponding values of phi
phi_values_mid1 <- estimate_phi(temperatures, q_phi, tmin_phi_mid1, tmax_phi_mid1)

# Min-Max normalization
norm_phi_mid1 <- (phi_values_mid1 - min(phi_values_mid1)) / (max(phi_values_mid1) - min(phi_values_mid1))

# Peak value of the phi Parameter
temp_peak_phi_mid1 <- (tmin_phi_mid1 + tmax_phi_mid1) / 2
peak_phi_mid1 <- estimate_phi(temp_peak_phi_mid1, q_phi, tmin_phi_mid1, tmax_phi_mid1)

# ------------------------------------------------
# NEXT: Alpha value for Mid 1 Mismatch
# Set temperature range
tmin_alpha_mid1 <- -4.25
tmax_alpha_mid1 <- 24.75

# Calculate corresponding values of alpha
alpha_values_mid1 <- estimate_alpha(temperatures, q_alpha, tmin_alpha_mid1, tmax_alpha_mid1)

# Min-Max Normalization
norm_alpha_mid1 <- (alpha_values_mid1 - min(alpha_values_mid1)) / (max(alpha_values_mid1) - min(alpha_value

# Peak value of the alpha Parameter
temp_peak_alpha_mid1 <- (tmin_alpha_mid1 + tmax_alpha_mid1) / 2
peak_alpha_mid1 <- estimate_alpha(temp_peak_alpha_mid1, q_alpha, tmin_alpha_mid1, tmax_alpha_mid1)
```

**Mid 2 Mismatch**

```r
# FIRST: Phi value for Mid 2 Mismatch
# Set temperature range
tmin_phi_mid2 <- -0.5
tmax_phi_mid2 <- 28.5

# Calculate corresponding values of phi
phi_values_mid2 <- estimate_phi(temperatures, q_phi, tmin_phi_mid2, tmax_phi_mid2)
```

```r
# Min-Max normalization
norm_phi_mid2 <- (phi_values_mid2 - min(phi_values_mid2)) / (max(phi_values_mid2) - min(phi_values_mid2))

# Peak value of the phi Parameter
temp_peak_phi_mid2 <- (tmin_phi_mid2 + tmax_phi_mid2) / 2
peak_phi_mid2 <- estimate_phi(temp_peak_phi_mid2, q_phi, tmin_phi_mid2, tmax_phi_mid2)

# ------------------------------------------------
# NEXT: Alpha value for Mid 2 Mismatch
# Set temperature range
tmin_alpha_mid2 <- -3.5
tmax_alpha_mid2 <- 25.5

# Calculate corresponding values of alpha
alpha_values_mid2 <- estimate_alpha(temperatures, q_alpha, tmin_alpha_mid2, tmax_alpha_mid2)

# Min-Max Normalization
norm_alpha_mid2 <- (alpha_values_mid2 - min(alpha_values_mid2)) / (max(alpha_values_mid2) - min(alpha_value

# Peak value of the alpha Parameter
temp_peak_alpha_mid2 <- (tmin_alpha_mid2 + tmax_alpha_mid2) / 2
peak_alpha_mid2 <- estimate_alpha(temp_peak_alpha_mid2, q_alpha, tmin_alpha_mid2, tmax_alpha_mid2)
```

**Mid 3 Mismatch**

```r
# FIRST: Phi value for Mid 3 Mismatch
# Set temperature range
tmin_phi_mid3 <- -1.25
tmax_phi_mid3 <- 27.75

# Calculate corresponding values of phi
phi_values_mid3 <- estimate_phi(temperatures, q_phi, tmin_phi_mid3, tmax_phi_mid3)

# Min-Max normalization
norm_phi_mid3 <- (phi_values_mid3 - min(phi_values_mid3)) / (max(phi_values_mid3) - min(phi_values_mid3))

# Peak value of the phi Parameter
temp_peak_phi_mid3 <- (tmin_phi_mid3 + tmax_phi_mid3) / 2
peak_phi_mid3 <- estimate_phi(temp_peak_phi_mid3, q_phi, tmin_phi_mid3, tmax_phi_mid3)

# ------------------------------------------------
# NEXT: Alpha value for Mid 3 Mismatch
# Set temperature range
tmin_alpha_mid3 <- -2.75
tmax_alpha_mid3 <- 26.25

# Calculate corresponding values of alpha
alpha_values_mid3 <- estimate_alpha(temperatures, q_alpha, tmin_alpha_mid3, tmax_alpha_mid3)

# Min-Max Normalization
norm_alpha_mid3 <- (alpha_values_mid3 - min(alpha_values_mid3)) / (max(alpha_values_mid3) - min(alpha_value
```

```r
# Peak value of the alpha Parameter
temp_peak_alpha_mid3 <- (tmin_alpha_mid3 + tmax_alpha_mid3) / 2
peak_alpha_mid3 <- estimate_alpha(temp_peak_alpha_mid3, q_alpha, tmin_alpha_mid3, tmax_alpha_mid3)
```

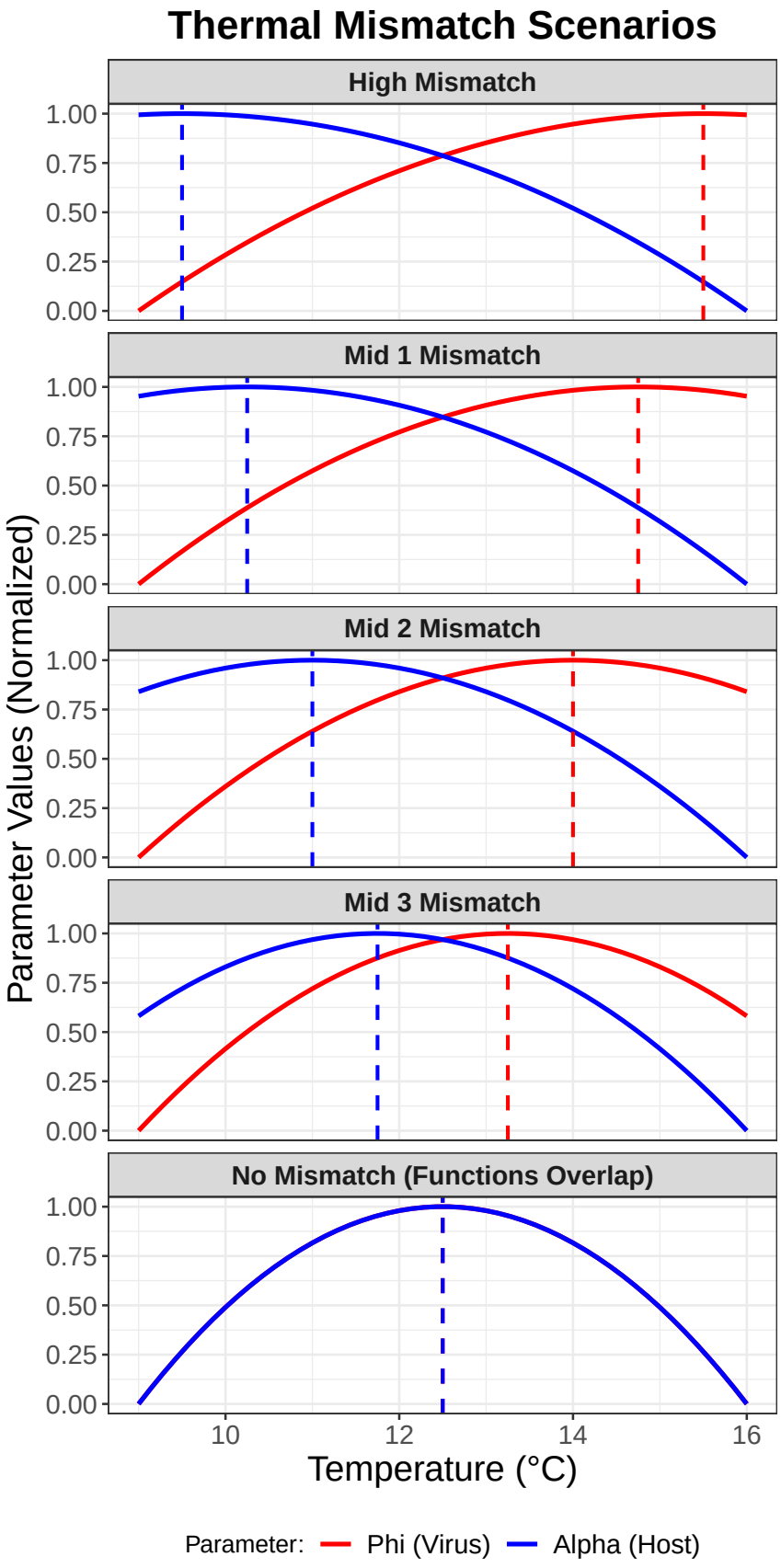
**No Mismatch (i.e., Null Scenario)**

```r
# FIRST: Phi value for No Mismatch
# Set temperature range
tmin_phi_null <- -2
tmax_phi_null <- 27

# Calculate corresponding values of phi
phi_values_null <- estimate_phi(temperatures, q_phi, tmin_phi_null, tmax_phi_null)

# Min-Max normalization
norm_phi_null <- (phi_values_null - min(phi_values_null)) / (max(phi_values_null) - min(phi_values_null))

# Peak value of the phi Parameter
temp_peak_phi_null <- (tmin_phi_null + tmax_phi_null) / 2
peak_phi_null <- estimate_phi(temp_peak_phi_null, q_phi, tmin_phi_null, tmax_phi_null)

# ------------------------------------------------
# NEXT: Alpha value for No Mismatch
# Set temperature range
tmin_alpha_null <- -2
tmax_alpha_null <- 27

# Calculate corresponding values of alpha
alpha_values_null <- estimate_alpha(temperatures, q_alpha, tmin_alpha_null, tmax_alpha_null)

# Min-Max Normalization
norm_alpha_null <- (alpha_values_null - min(alpha_values_null)) / (max(alpha_values_null) - min(alpha_value

# Peak value of the alpha Parameter
temp_peak_alpha_null <- (tmin_alpha_null + tmax_alpha_null) / 2
peak_alpha_null <- estimate_alpha(temp_peak_alpha_null, q_alpha, tmin_alpha_null, tmax_alpha_null)
```

**Now we can plot the scenarios:**

```r
param_function_plot_data <- data.frame(
  Temperature = rep(temperatures, 5),
  Normalized_Phi = c(norm_phi_high, norm_phi_mid1, norm_phi_mid2, norm_phi_mid3, norm_phi_null),
  Normalized_Alpha = c(norm_alpha_high, norm_alpha_mid1, norm_alpha_mid2, norm_alpha_mid3, norm_alpha_null)
  Scenario = factor(rep(c("High Mismatch", "Mid 1 Mismatch", "Mid 2 Mismatch", "Mid 3 Mismatch", "No Mismat
)

optimal_temps <- data.frame(
  Scenario = factor(c("High Mismatch", "Mid 1 Mismatch", "Mid 2 Mismatch", "Mid 3 Mismatch", "No Mismatch")
  Temp_Peak_Phi_opt = c(temp_peak_phi_high, temp_peak_phi_mid1, temp_peak_phi_mid2, temp_peak_phi_mid3, tem
  Temp_Peak_Alpha_opt = c(temp_peak_alpha_high, temp_peak_alpha_mid1, temp_peak_alpha_mid2, temp_peak_alpha
)
```

```r
# pivot for ggplot
param_function_plot_data_long <- param_function_plot_data %>%
  pivot_longer(cols = c(Normalized_Phi, Normalized_Alpha),
               names_to = "Parameter", values_to = "Normalized_Value")

quad_param_plot_data <- merge(param_function_plot_data_long, optimal_temps, by = "Scenario")
quad_param_plot_data$Parameter <- factor(quad_param_plot_data$Parameter,
                                         levels = c("Normalized_Phi", "Normalized_Alpha"))

# Create the plot
quad_param_plot <- ggplot(quad_param_plot_data,
                          aes(x = Temperature, y = Normalized_Value, color = Parameter)) +
  geom_line(linewidth = 1) +
  facet_wrap(~ Scenario, ncol = 1,
             labeller = labeller(Scenario = c("No Mismatch" = "No Mismatch (Functions Overlap)",
                                              "High Mismatch" = "High Mismatch",
                                              "Mid 1 Mismatch" = "Mid 1 Mismatch",
                                              "Mid 2 Mismatch" = "Mid 2 Mismatch",
                                              "Mid 3 Mismatch" = "Mid 3 Mismatch"))) +
  scale_color_manual(values = c("Normalized_Phi" = "red", "Normalized_Alpha" = "blue"),
                     labels = c("Phi (Virus)", "Alpha (Host)")) +
  geom_vline(aes(xintercept = Temp_Peak_Phi_opt), linetype = "dashed", col = "red", linewidth = 0.8) +
  geom_vline(aes(xintercept = Temp_Peak_Alpha_opt), linetype = "dashed", col = "blue", linewidth = 0.8) +
  theme_bw() +
  labs(x = "Temperature (°C)", y = "Parameter Values (Normalized)",
       color = "Parameter:", title = "Thermal Mismatch Scenarios") +
  theme(strip.text = element_text(size = 12, face = "bold"),
        legend.position = "bottom",
        plot.title = element_text(hjust = 0.5, size = 18, face = "bold"),
        axis.title = element_text(size = 16),
        axis.text = element_text(size = 12),
        legend.text = element_text(size = 12))

quad_param_plot
```


Thermal Mismatch Scenarios
High Mismatch
Mid 1 Mismatch
Mid 2 Mismatch
Mid 3 Mismatch
No Mismatch (Functions Overlap)
Parameter Values (Normalized)
1.00
0.75
0.50
0.25
0.00
10
12
14
16
Temperature (°C)
Parameter: Phi (Virus) Alpha (Host)

```
#ggsave("Figures/quad_param_plot.pdf", plot = quad_param_plot,
#       device = "pdf", width = 5, height = 10)
```

## Running the Model (Multiple Temperatures and Multiple Scenarios)

First, setup the code to run model simulations. Then use the parameter values (for phi and alpha) calculated above to run the model at the different scenarios and along the thermal gradient.

Initial setup of universal variables/parameters:

```
# Set time/tau:
tau <- 1 # leap size
t_max <- 40 # days
n_times <- floor(t_max/tau)
t_vec <- seq(1, t_max, length.out = n_times)

# Vector to store Viral and Immune Load
V_vec <- vector(mode = "numeric", length = n_times)
Z_vec <- vector(mode = "numeric", length = n_times)

z_scalar <- 1000

death_threshold <- 3000
clearance_threshold <- 0

# Set initial values
V_vec[1] <- 250                      # Initial viral density
Z_vec[1] <- rpois(1, 0.35*z_scalar)  # Initial immune component density

# Set number of iterations per scenario per temperature:
num_sims <- 100

# Set temperatures to check:
temp_checks <- c(9.5, 12.5, 15.5)
```

### High mismatch

```
# Store results for high mismatch:
results_high <- list()

# Loop through each set temperature for high mismatch scenario:
for (temp in temp_checks) {

  # Temp-specific parameters:
  phi_high <- estimate_phi(temp, q_phi, tmin_phi_high, tmax_phi_high)
  alpha_high <- estimate_alpha(temp, q_phi, tmin_alpha_high, tmax_alpha_high)

  # All Parms:
  params_high <- list(
    phi = phi_high,
    alpha = alpha_high,
```

```
    delta = 1.29,
    psi = 0.91,
    gamma = 0.13,
    N_z = 1.29 * 0.35,
    z_scalar = z_scalar
  )

  # Initialize a list to store simulation results for the current temperature
  temp_results_high <- vector("list", num_sims)

  # Run model for all simulations:
  for (sim in 1:num_sims) {
    sim_results <- model_simulation(
      V_vec = V_vec,
      Z_vec = Z_vec,
      params = params_high,
      death_threshold = death_threshold,
      clearance_threshold = clearance_threshold
    )

    # Store the simulation results as a list
    temp_results_high[[sim]] <- list(Simulation = sim,
                                     V_vec = sim_results$V_vec,
                                     Z_vec = sim_results$Z_vec)
  }

  # Convert the list of simulation results to a data frame
  temp_results_high_df <- do.call(rbind, lapply(temp_results_high, as.data.frame))

  # Add results for this temperature to the overall list
  results_high[[paste0("Temp", temp)]] <- temp_results_high_df
}
```

**Mid 1 Mismatch**

```
# Store results for mid 1 mismatch:
results_mid1 <- list()

# Loop through each set temperature for mid 1 mismatch scenario:
for (temp in temp_checks) {

  # Temp-specific parameters:
  phi_mid1 <- estimate_phi(temp, q_phi, tmin_phi_mid1, tmax_phi_mid1)
  alpha_mid1 <- estimate_alpha(temp, q_phi, tmin_alpha_mid1, tmax_alpha_mid1)

  # All Parms:
  params_mid1 <- list(
    phi = phi_mid1,
    alpha = alpha_mid1,
    delta = 1.29,
    psi = 0.91,
    gamma = 0.13,
    N_z = 1.29 * 0.35,
```

```
    z_scalar = z_scalar
  )

  # Initialize a list to store simulation results for the current temperature
  temp_results_mid1 <- vector("list", num_sims)

  # Run model for all simulations:
  for (sim in 1:num_sims) {
    sim_results <- model_simulation(
      V_vec = V_vec,
      Z_vec = Z_vec,
      params = params_mid1,
      death_threshold = death_threshold,
      clearance_threshold = clearance_threshold
    )

    # Store the simulation results as a list
    temp_results_mid1[[sim]] <- list(Simulation = sim,
                                     V_vec = sim_results$V_vec,
                                     Z_vec = sim_results$Z_vec)
  }

  # Convert the list of simulation results to a data frame
  temp_results_mid1_df <- do.call(rbind, lapply(temp_results_mid1, as.data.frame))

  # Add results for this temperature to the overall list
  results_mid1[[paste0("Temp", temp)]] <- temp_results_mid1_df
}
```

**Mid 2 Mismatch**

```
# Store results for mid 2 mismatch:
results_mid2 <- list()

# Loop through each set temperature for mid 2 mismatch scenario:
for (temp in temp_checks) {

  # Temp-specific parameters:
  phi_mid2 <- estimate_phi(temp, q_phi, tmin_phi_mid2, tmax_phi_mid2)
  alpha_mid2 <- estimate_alpha(temp, q_phi, tmin_alpha_mid2, tmax_alpha_mid2)

  # All Parms:
  params_mid2 <- list(
    phi = phi_mid2,
    alpha = alpha_mid2,
    delta = 1.29,
    psi = 0.91,
    gamma = 0.13,
    N_z = 1.29 * 0.35,
    z_scalar = z_scalar
  )

  # Initialize a list to store simulation results for the current temperature
```

```
  temp_results_mid2 <- vector("list", num_sims)

  # Run model for all simulations:
  for (sim in 1:num_sims) {
    sim_results <- model_simulation(
      V_vec = V_vec,
      Z_vec = Z_vec,
      params = params_mid2,
      death_threshold = death_threshold,
      clearance_threshold = clearance_threshold
    )

    # Store the simulation results as a list
    temp_results_mid2[[sim]] <- list(Simulation = sim,
                                     V_vec = sim_results$V_vec,
                                     Z_vec = sim_results$Z_vec)
  }

  # Convert the list of simulation results to a data frame
  temp_results_mid2_df <- do.call(rbind, lapply(temp_results_mid2, as.data.frame))

  # Add results for this temperature to the overall list
  results_mid2[[paste0("Temp", temp)]] <- temp_results_mid2_df
}
```

**Mid 3 Mismatch**

```
# Store results for mid 3 mismatch:
results_mid3 <- list()

# Loop through each set temperature for mid 3 mismatch scenario:
for (temp in temp_checks) {

  # Temp-specific parameters:
  phi_mid3 <- estimate_phi(temp, q_phi, tmin_phi_mid3, tmax_phi_mid3)
  alpha_mid3 <- estimate_alpha(temp, q_phi, tmin_alpha_mid3, tmax_alpha_mid3)

  # All Parms:
  params_mid3 <- list(
    phi = phi_mid3,
    alpha = alpha_mid3,
    delta = 1.29,
    psi = 0.91,
    gamma = 0.13,
    N_z = 1.29 * 0.35,
    z_scalar = z_scalar
  )

  # Initialize a list to store simulation results for the current temperature
  temp_results_mid3 <- vector("list", num_sims)

  # Run model for all simulations:
  for (sim in 1:num_sims) {
```

```
    sim_results <- model_simulation(
      V_vec = V_vec,
      Z_vec = Z_vec,
      params = params_mid3,
      death_threshold = death_threshold,
      clearance_threshold = clearance_threshold
    )

    # Store the simulation results as a list
    temp_results_mid3[[sim]] <- list(Simulation = sim,
                                     V_vec = sim_results$V_vec,
                                     Z_vec = sim_results$Z_vec)
  }

  # Convert the list of simulation results to a data frame
  temp_results_mid3_df <- do.call(rbind, lapply(temp_results_mid3, as.data.frame))

  # Add results for this temperature to the overall list
  results_mid3[[paste0("Temp", temp)]] <- temp_results_mid3_df
}
```

**No Mismatch (i.e., Null hypothesis)**

```
# Store results for mid 2 mismatch:
results_null <- list()

# Loop through each set temperature for mid 2 mismatch scenario:
for (temp in temp_checks) {
  # Temp-specific parameters:
  phi_null <- estimate_phi(temp, q_phi, tmin_phi_null, tmax_phi_null)
  alpha_null <- estimate_alpha(temp, q_phi, tmin_alpha_null, tmax_alpha_null)

  # All Parms:
  params_null <- list(
    phi = phi_null,
    alpha = alpha_null,
    delta = 1.29,
    psi = 0.91,
    gamma = 0.13,
    N_z = 1.29 * 0.35,
    z_scalar = z_scalar
  )

  # Initialize a list to store simulation results for the current temperature
  temp_results_null <- vector("list", num_sims)

  # Run model for all simulations:
  for (sim in 1:num_sims) {
    sim_results <- model_simulation(
      V_vec = V_vec,
      Z_vec = Z_vec,
      params = params_null,
      death_threshold = death_threshold,
```

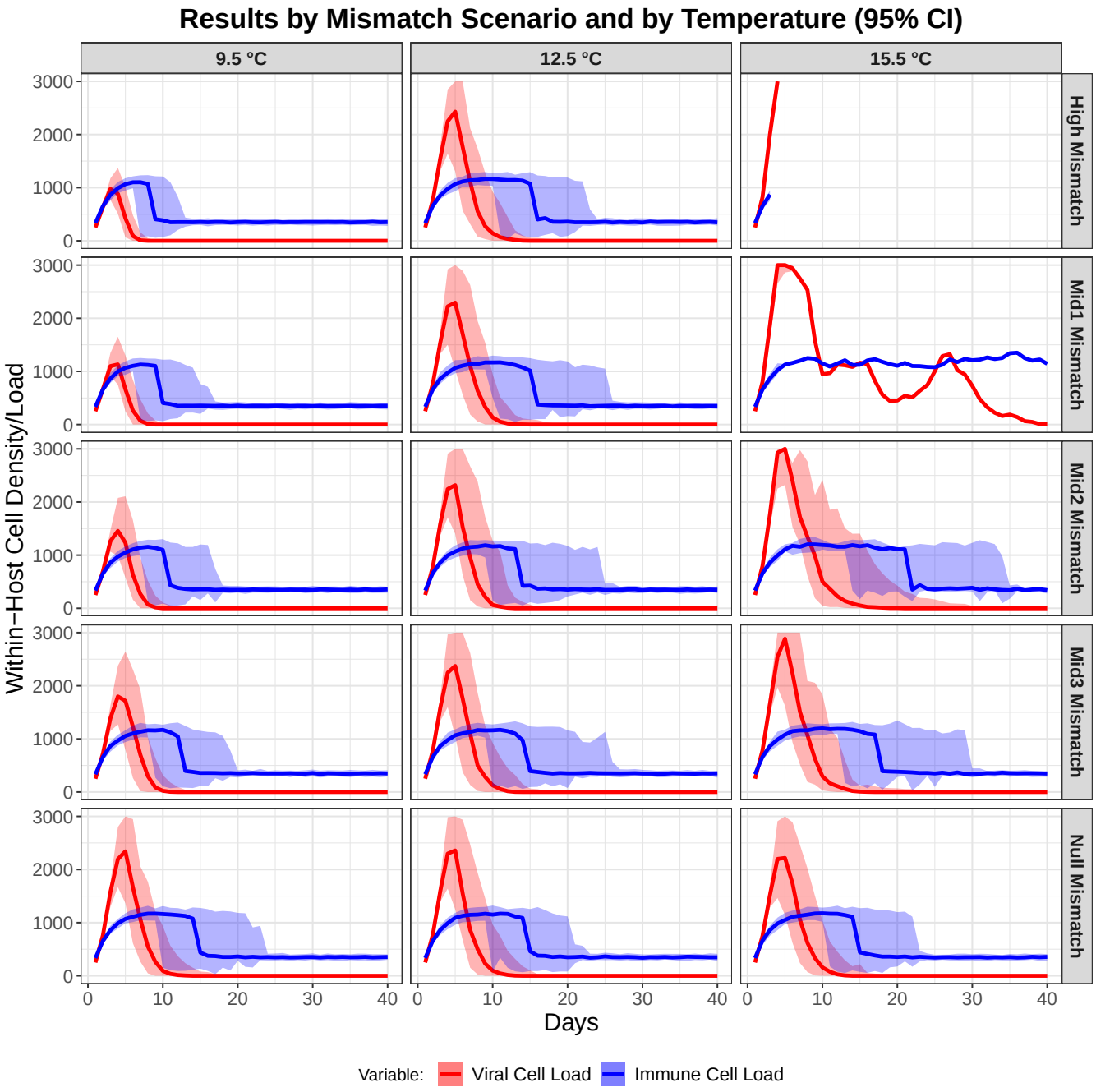
```
      clearance_threshold = clearance_threshold
    )

    # Store the simulation results as a list
    temp_results_null[[sim]] <- list(Simulation = sim,
                                     V_vec = sim_results$V_vec,
                                     Z_vec = sim_results$Z_vec)
  }

  # Convert the list of simulation results to a data frame
  temp_results_null_df <- do.call(rbind, lapply(temp_results_null, as.data.frame))

  # Add results for this temperature to the overall list
  results_null[[paste0("Temp", temp)]] <- temp_results_null_df
}
```

**Create plot with all results (4 Scenarios, 3 Temps each)**

```
# Function to combine the results
combine_results <- function(results_list, scenario_name){
  do.call(rbind, lapply(names(results_list), function(temp_name){
    temp_data <- results_list[[temp_name]]
    temp_data$Scenario <- scenario_name
    temp_data$Temperature <- as.numeric(gsub("Temp", "", temp_name))
    return(temp_data)
  }))
}

# All scenarios to one data frame
all_combined_results <- rbind(
  combine_results(results_high, "High"),
  combine_results(results_mid1, "Mid1"),
  combine_results(results_mid2, "Mid2"),
  combine_results(results_mid3, "Mid3"),
  combine_results(results_null, "Null")
)

# Summary stats for Confidence intervals and medians for plotting
summary_results <- all_combined_results %>%
  tidyr::unnest(cols = c(V_vec, Z_vec)) %>%
  group_by(Scenario, Temperature) %>%
  mutate(Day = rep(1:40, times = n() / 40)) %>%
  group_by(Scenario, Temperature, Day) %>%
  summarise(
    V_median = median(V_vec, na.rm = TRUE),
    V_low = quantile(V_vec, 0.025, na.rm = TRUE),
    V_high = quantile(V_vec, 0.975, na.rm = TRUE),
    Z_median = median(Z_vec, na.rm = TRUE),
    Z_low = quantile(Z_vec, 0.025, na.rm = TRUE),
    Z_high = quantile(Z_vec, 0.975, na.rm = TRUE),
    .groups = "drop"
  )
```

Geom_ribbon() to show 95% CI and variability for each scenario/temp plot

```
# Create the figure (facet_wrap, rows are scenarios, columns are temperatures)
scenario_by_temp_plot_variability <- ggplot(summary_results, aes(x = Day)) +
  # V_vec
  geom_ribbon(aes(ymin = V_low, ymax = V_high, fill = "V_vec"), alpha = 0.3) +
  geom_line(aes(y = V_median, color = "V_vec"), size = 1.2) +
  # Z_vec
  geom_ribbon(aes(ymin = Z_low, ymax = Z_high, fill = "Z_vec"), alpha = 0.3) +
  geom_line(aes(y = Z_median, color = "Z_vec"), size = 1.2) +
  # facet by scenario and temperature
  facet_grid(Scenario ~ Temperature,
             labeller = labeller(Temperature = function(x) paste0(x, " °C"),
                                 Scenario = function(x) paste0(x, " Mismatch"),
                                 )) +
  # aesthetics:
  scale_fill_manual(
    name = "Variable: ",
    values = c("V_vec" = "red", "Z_vec" = "blue"),
    labels = c("Viral Cell Load", "Immune Cell Load")
  ) +
  scale_color_manual(
    name = "Variable: ",
    values = c("V_vec" = "red", "Z_vec" = "blue"),
    labels = c("Viral Cell Load", "Immune Cell Load")
  ) +
  labs(title = "Results by Mismatch Scenario and by Temperature (95% CI)",
       x = "Days",
       y = "Within-Host Cell Density/Load") +
  theme_bw() +
  theme(legend.position = "bottom",
        legend.box = "horizontal",
        panel.grid = element_line(color = "gray90"),
        plot.title = element_text(hjust = 0.5, size = 18, face = "bold"),
        axis.title = element_text(size = 16),
        axis.text = element_text(size = 12),
        strip.text = element_text(size = 12, face = "bold"),
        legend.text = element_text(size = 12))

scenario_by_temp_plot_variability
```

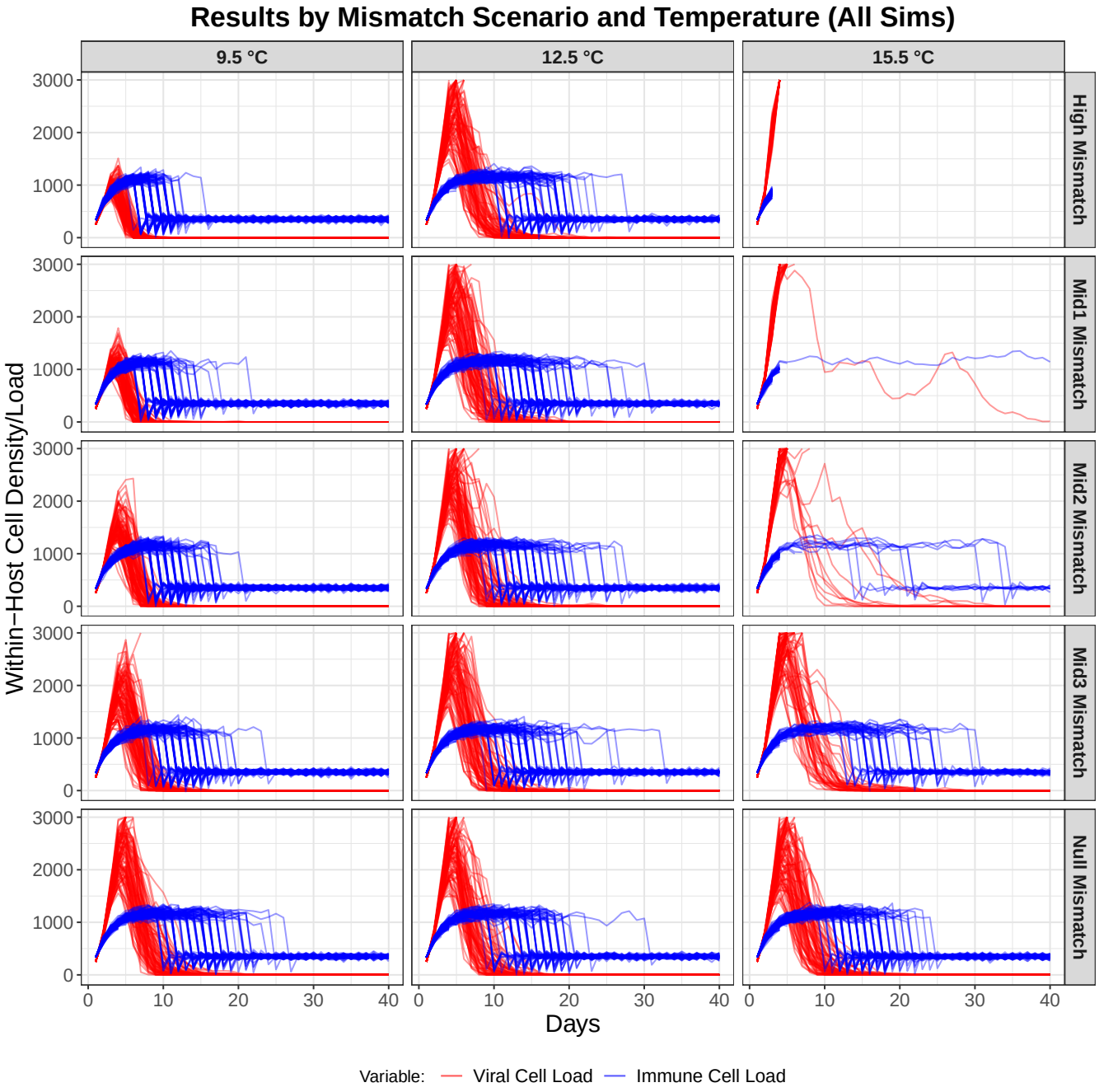



```
#ggsave("Figures/scenario_by_temp_plot_variability.pdf", plot = scenario_by_temp_plot_variability,
#       device = "pdf", width = 10, height = 10)
```

Now a plot to show all individual simulations:

```
individual_simulations <- all_combined_results %>%
  tidyr::unnest(cols = c(V_vec, Z_vec)) %>%
  mutate(Day = rep(1:40, times = n() / 40),
             Simulation_ID = rep(1:100, each = 40, times = length(unique(Scenario)) * length(unique(Tempera
)
```

```r
# Create the plot
scenario_by_temp_plot_individualsims <- ggplot(individual_simulations, aes(x = Day)) +
  # V_vec
  geom_line(aes(y = V_vec, group = interaction(Scenario, Temperature, Simulation_ID), color = "V_vec"),
            alpha = 0.4, linewidth = 0.5) +
  # Z_vec
  geom_line(aes(y = Z_vec, group = interaction(Scenario, Temperature, Simulation_ID), color = "Z_vec"),
            alpha = 0.4, linewidth = 0.5) +
  # Facet by scenario and temperature
  facet_grid(Scenario ~ Temperature,
             labeller = labeller(Temperature = function(x) paste0(x, " °C"),
                                 Scenario = function(x) paste0(x, " Mismatch"))) +
  # Aesthetics:
  scale_color_manual(
    name = "Variable: ",
    values = c("V_vec" = "red", "Z_vec" = "blue"),
    labels = c("Viral Cell Load", "Immune Cell Load")
  ) +
  labs(title = "Results by Mismatch Scenario and Temperature (All Sims)",
       x = "Days",
       y = "Within-Host Cell Density/Load") +
  theme_bw() +
  theme(legend.position = "bottom",
        legend.box = "horizontal",
        panel.grid = element_line(color = "gray90"),
        plot.title = element_text(hjust = 0.5, size = 18, face = "bold"),
        axis.title = element_text(size = 16),
        axis.text = element_text(size = 12),
        strip.text = element_text(size = 12, face = "bold"),
        legend.text = element_text(size = 12))

scenario_by_temp_plot_individualsims
```




```
#ggsave("Figures/scenario_by_temp_plot_individualsims.pdf", plot = scenario_by_temp_plot_individualsims,
#       device = "pdf", width = 10, height = 10)
```

## Track the Proportion of deaths across thermal gradient

**High Mismatch:**

```r
# Data frame to store death proportion for high mismatch
death_prop_high <- data.frame(Temperature = temperatures, Proportion_Deaths = NA)

# Now loop through each temperature and run model simulations
for(temp_idx in seq_along(temperatures)) {
  temp <- temperatures[temp_idx]

  # Temp-specific parameters:
  phi_high <- estimate_phi(temp, q_phi, tmin_phi_high, tmax_phi_high)
  alpha_high <- estimate_alpha(temp, q_phi, tmin_alpha_high, tmax_alpha_high)

  # All Parms:
  params_high <- list(
    phi = phi_high,
    alpha = alpha_high,
    delta = 1.29,
    psi = 0.91,
    gamma = 0.13,
    N_z = 1.29 * 0.35,
    z_scalar = z_scalar
  )

  # Simulations and track deaths
  death_count <- 0
  for (sim in 1:num_sims){
    sim_results <- model_simulation(
      V_vec = V_vec,
      Z_vec = Z_vec,
      params = params_high,
      death_threshold = death_threshold,
      clearance_threshold = clearance_threshold
    )

    # Was death_threshold reached?
    if(max(sim_results$V_vec, na.rm = TRUE) >= death_threshold){
      death_count <- death_count + 1
    }
  }

  # Proportion of death for this temperature
  death_prop_high$Proportion_Deaths[temp_idx] <- death_count/num_sims
}
```

**Mid 1 Mismatch:**

```r
# Data frame to store death proportion for mid1 mismatch
death_prop_mid1 <- data.frame(Temperature = temperatures, Proportion_Deaths = NA)

# Now loop through each temperature and run model simulations
for(temp_idx in seq_along(temperatures)) {
  temp <- temperatures[temp_idx]

  # Temp-specific parameters:
```

```
  phi_mid1 <- estimate_phi(temp, q_phi, tmin_phi_mid1, tmax_phi_mid1)
  alpha_mid1 <- estimate_alpha(temp, q_phi, tmin_alpha_mid1, tmax_alpha_mid1)

  # All Parms:
  params_mid1 <- list(
    phi = phi_mid1,
    alpha = alpha_mid1,
    delta = 1.29,
    psi = 0.91,
    gamma = 0.13,
    N_z = 1.29 * 0.35,
    z_scalar = z_scalar
  )

  # Simulations and track deaths
  death_count <- 0
  for (sim in 1:num_sims){
    sim_results <- model_simulation(
      V_vec = V_vec,
      Z_vec = Z_vec,
      params = params_mid1,
      death_threshold = death_threshold,
      clearance_threshold = clearance_threshold
    )

    # Was death_threshold reached?
    if(max(sim_results$V_vec, na.rm = TRUE) >= death_threshold){
      death_count <- death_count + 1
    }
  }

  # Proportion of death for this temperature
  death_prop_mid1$Proportion_Deaths[temp_idx] <- death_count/num_sims
}
```

**Mid 2 Mismatch:**

```
# Data frame to store death proportion for mid2 mismatch
death_prop_mid2 <- data.frame(Temperature = temperatures, Proportion_Deaths = NA)

# Now loop through each temperature and run model simulations
for(temp_idx in seq_along(temperatures)) {
  temp <- temperatures[temp_idx]

  # Temp-specific parameters:
  phi_mid2 <- estimate_phi(temp, q_phi, tmin_phi_mid2, tmax_phi_mid2)
  alpha_mid2 <- estimate_alpha(temp, q_phi, tmin_alpha_mid2, tmax_alpha_mid2)

  # All Parms:
  params_mid2 <- list(
    phi = phi_mid2,
    alpha = alpha_mid2,
    delta = 1.29,
```

```
    psi = 0.91,
    gamma = 0.13,
    N_z = 1.29 * 0.35,
    z_scalar = z_scalar
  )

  # Simulations and track deaths
  death_count <- 0
  for (sim in 1:num_sims){
    sim_results <- model_simulation(
      V_vec = V_vec,
      Z_vec = Z_vec,
      params = params_mid2,
      death_threshold = death_threshold,
      clearance_threshold = clearance_threshold
    )

    # Was death_threshold reached?
    if(max(sim_results$V_vec, na.rm = TRUE) >= death_threshold){
      death_count <- death_count + 1
    }
  }

  # Proportion of death for this temperature
  death_prop_mid2$Proportion_Deaths[temp_idx] <- death_count/num_sims
}
```

**Mid 3 Mismatch:**

```
# Data frame to store death proportion for mid3 mismatch
death_prop_mid3 <- data.frame(Temperature = temperatures, Proportion_Deaths = NA)

# Now loop through each temperature and run model simulations
for(temp_idx in seq_along(temperatures)) {
  temp <- temperatures[temp_idx]

  # Temp-specific parameters:
  phi_mid3 <- estimate_phi(temp, q_phi, tmin_phi_mid3, tmax_phi_mid3)
  alpha_mid3 <- estimate_alpha(temp, q_phi, tmin_alpha_mid3, tmax_alpha_mid3)

  # All Parms:
  params_mid3 <- list(
    phi = phi_mid3,
    alpha = alpha_mid3,
    delta = 1.29,
    psi = 0.91,
    gamma = 0.13,
    N_z = 1.29 * 0.35,
    z_scalar = z_scalar
  )

  # Simulations and track deaths
  death_count <- 0
```

```
  for (sim in 1:num_sims){
    sim_results <- model_simulation(
      V_vec = V_vec,
      Z_vec = Z_vec,
      params = params_mid3,
      death_threshold = death_threshold,
      clearance_threshold = clearance_threshold
    )

    # Was death_threshold reached?
    if(max(sim_results$V_vec, na.rm = TRUE) >= death_threshold){
      death_count <- death_count + 1
    }
  }

  # Proportion of death for this temperature
  death_prop_mid3$Proportion_Deaths[temp_idx] <- death_count/num_sims
}
```

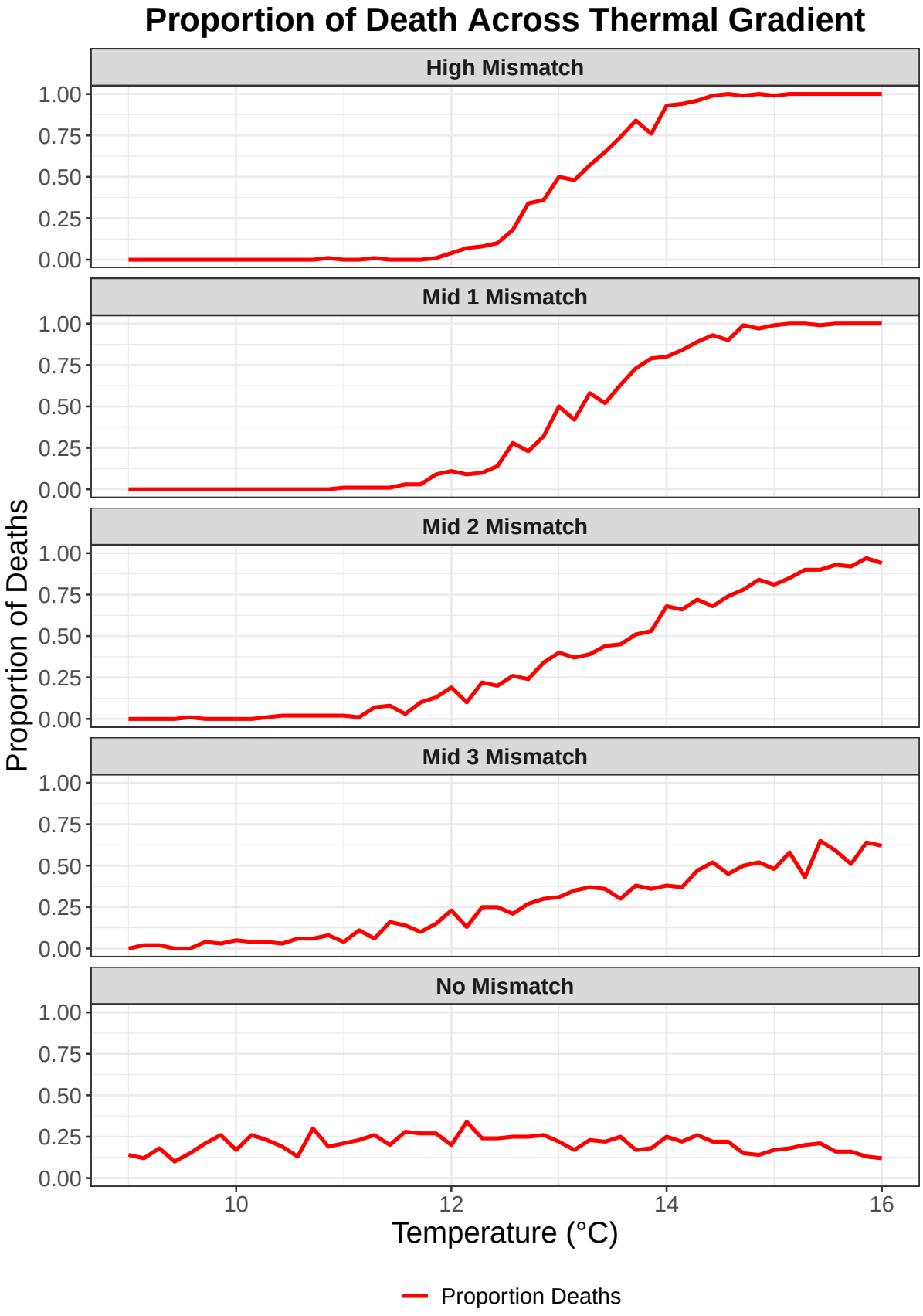
**Null Mismatch:**

```
# Data frame to store death proportion for null mismatch
death_prop_null <- data.frame(Temperature = temperatures, Proportion_Deaths = NA)

# Now loop through each temperature and run model simulations
for(temp_idx in seq_along(temperatures)) {
  temp <- temperatures[temp_idx]

  # Temp-specific parameters:
  phi_null <- estimate_phi(temp, q_phi, tmin_phi_null, tmax_phi_null)
  alpha_null <- estimate_alpha(temp, q_phi, tmin_alpha_null, tmax_alpha_null)

  # All Parms:
  params_null <- list(
    phi = phi_null,
    alpha = alpha_null,
    delta = 1.29,
    psi = 0.91,
    gamma = 0.13,
    N_z = 1.29 * 0.35,
    z_scalar = z_scalar
  )

  # Simulations and track deaths
  death_count <- 0
  for (sim in 1:num_sims){
    sim_results <- model_simulation(
      V_vec = V_vec,
      Z_vec = Z_vec,
      params = params_null,
      death_threshold = death_threshold,
      clearance_threshold = clearance_threshold
    )
```

```
    # Was death_threshold reached?
    if(max(sim_results$V_vec, na.rm = TRUE) >= death_threshold){
      death_count <- death_count + 1
    }
  }

  # Proportion of death for this temperature
  death_prop_null$Proportion_Deaths[temp_idx] <- death_count/num_sims
}
```

**Plot the proportion of death at each scenario**

```
# Combine results to one data.frame
death_data <- data.frame(
  Temperature = rep(temperatures, times = 5),
  Proportion_Deaths = c(death_prop_high$Proportion_Deaths,
                        death_prop_mid1$Proportion_Deaths,
                        death_prop_mid2$Proportion_Deaths,
                        death_prop_mid3$Proportion_Deaths,
                        death_prop_null$Proportion_Deaths),
  Scenario = rep(c("High Mismatch", "Mid 1 Mismatch", "Mid 2 Mismatch", "Mid 3 Mismatch", "No Mismatch"), e
)

death_data$Legend_Label <- "Proportion Deaths"

# Create the plot
death_plot <- ggplot(death_data, aes(x = Temperature, y = Proportion_Deaths, color = Legend_Label)) +
  geom_line(linewidth = 1) +
  facet_wrap(~ Scenario, ncol = 1) +
  scale_color_manual(values = c("Proportion Deaths" = "red")) +
  theme_bw() +
  labs(x = "Temperature (°C)", y = "Proportion of Deaths",
       color = NULL, title = "Proportion of Death Across Thermal Gradient") +
  theme(strip.text = element_text(size = 12, face = "bold"),
        legend.position = "bottom",
        plot.title = element_text(hjust = 0.5, size = 18, face = "bold"),
        axis.title = element_text(size = 16),
        axis.text = element_text(size = 12),
        legend.text = element_text(size = 12))

death_plot
```


Proportion of Death Across Thermal Gradient
High Mismatch
Mid 1 Mismatch
Mid 2 Mismatch
Mid 3 Mismatch
No Mismatch
1.00
0.75
0.50
0.25
0.00
Proportion of Deaths
10
12
14
16
Temperature (°C)
Proportion Deaths

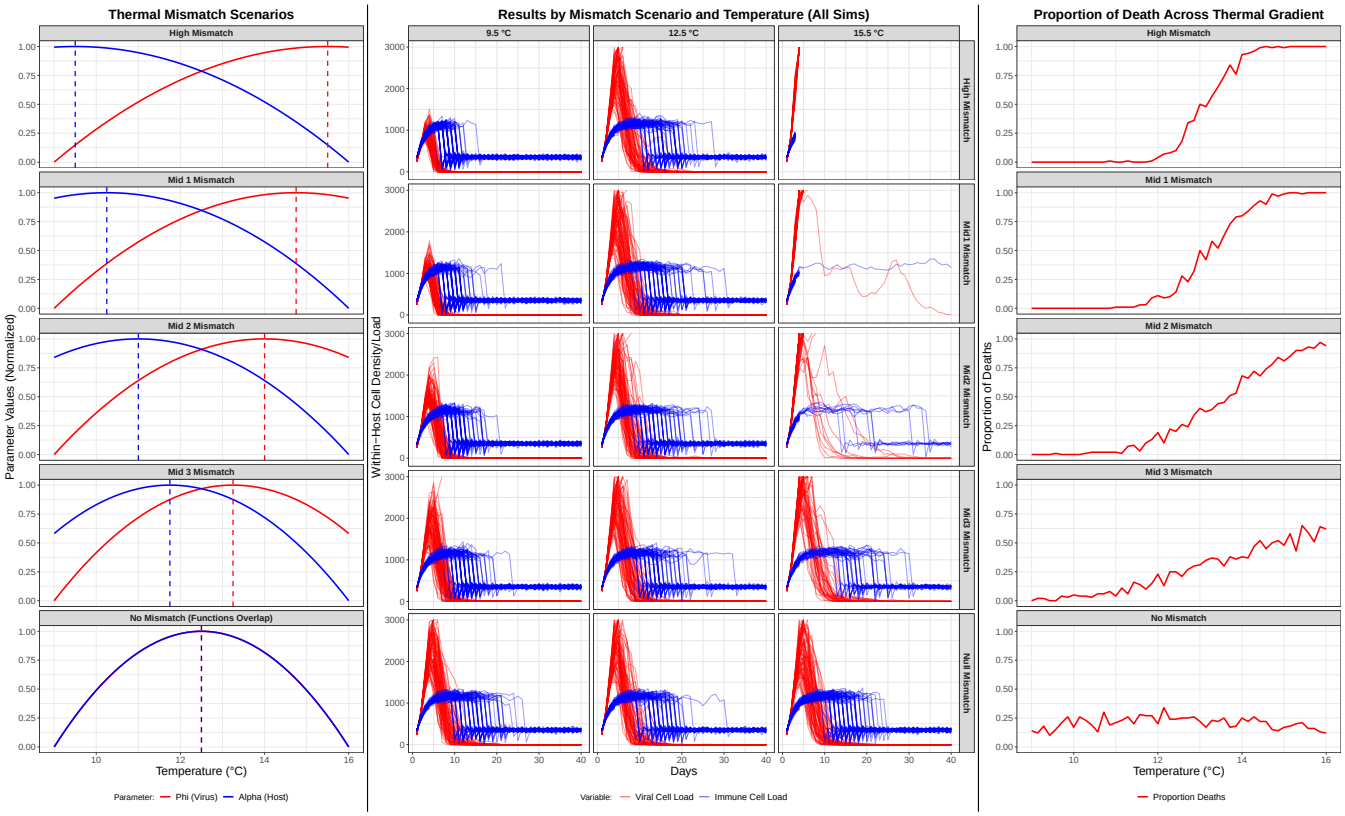
```
#ggsave("Figures/death_plot.pdf", plot = death_plot,
#       device = "pdf", width = 7, height = 10)
```

## Display all plots side by side:

```
p1 <- quad_param_plot
p2_a <- scenario_by_temp_plot_variability
p2_b <- scenario_by_temp_plot_individualsims
p3 <- death_plot

# setting widths
widths <- c(0.75, 1.25, 0.75)
total_width <- sum(widths)
prop1 <- widths[1] / total_width
prop2 <- (widths[1] + widths[2]) / total_width
```

**Plot with all individual simulations**

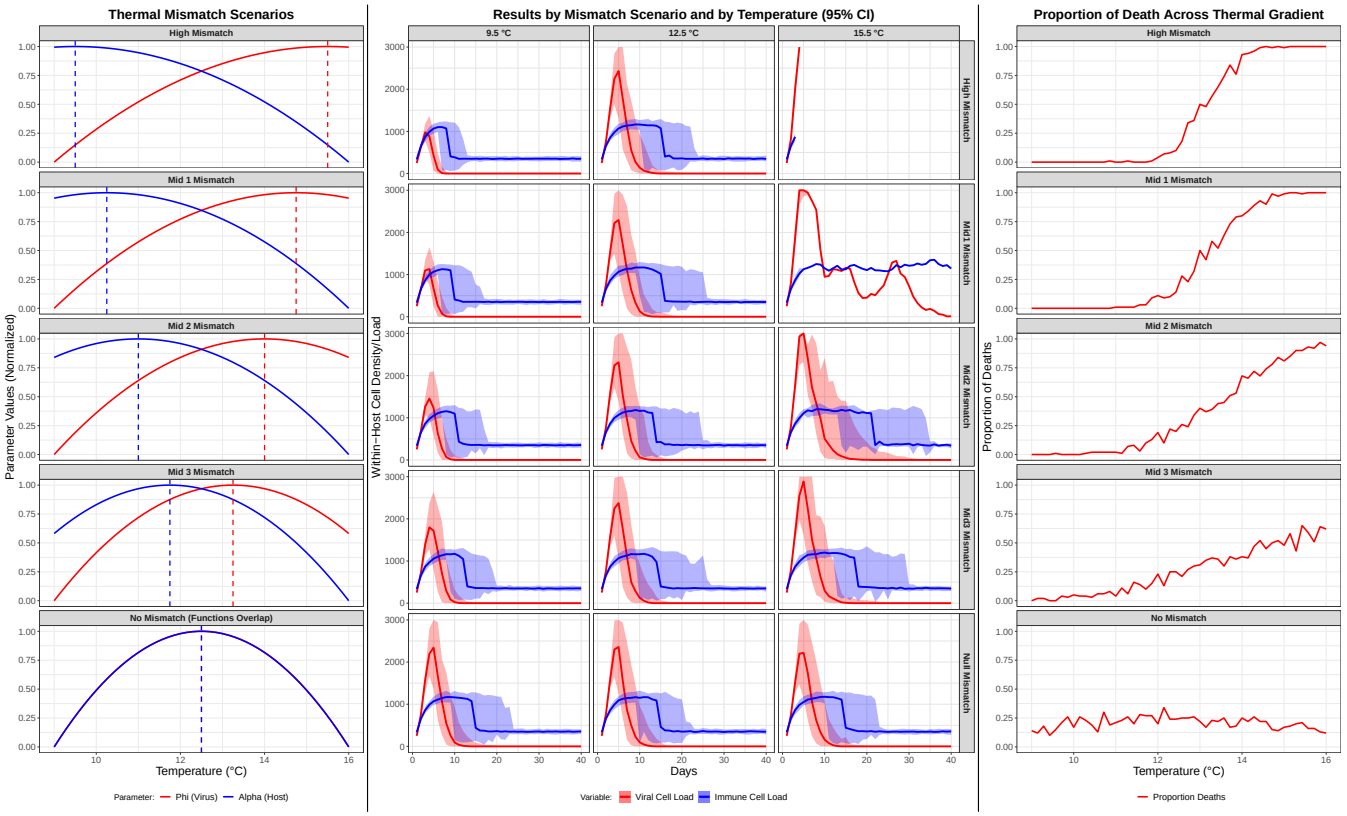
```
all_plots_individual_sims <- gridExtra::grid.arrange(
  p1, p2_b, p3,
  ncol = 3,
  widths = widths,
  padding = unit(1.5, "cm")
)

grid.lines(x = c(prop1, prop1), y = c(0, 1),
           gp = gpar(col = "black", lwd = 2))
grid.lines(x = c(prop2, prop2), y = c(0, 1),
           gp = gpar(col = "black", lwd = 2))
```




```
invisible(all_plots_individual_sims)

#ggsave("Figures/all_plots_individual_sims.pdf", plot = all_plots_individual_sims,
#       device = "pdf", width = 25, height = 15)
```

### Plot with variability

```
all_plots_variability <- gridExtra::grid.arrange(p1, p2_a, p3,
                                                 ncol = 3,
                                                 widths = widths,
                                                 padding = unit(1.5, "cm"))
grid.lines(x = c(prop1, prop1), y = c(0, 1),
           gp = gpar(col = "black", lwd = 2))
grid.lines(x = c(prop2, prop2), y = c(0, 1),
           gp = gpar(col = "black", lwd = 2))
```




```
invisible(all_plots_variability)

#ggsave("Figures/all_plots_variability.pdf", plot = all_plots_variability,
#       device = "pdf", width = 25, height = 15)
```