AREC 615
Integrated Malaria Intervention Optimization Model

Submitted By:

Md Mehran Alam

Submitted To:
Dr. Jude Bayham
Associate Professor
DARE
Colorado State University

Contents

Abstract

Optimization methods in disease dynamics modeling has been using the S,I,R framework for quite some time and is an established ODE-based model. However, in recent times, there has been a surge of new economic-epidemiology models that make use of human behavior. This paper serves as a mathematical and computational description of an integrated malaria intervention optimization model that couples disease dynamics (SIR-type ODEs), behavioral response (belief and effort dynamics), and optimization for constrained resource allocation. The model has been run and implemented in R. I present the equations, parameterization, optimization formulations, solver details, and include the full R implementation in the Appendix.

Introduction

A global leading cause of vector-borne disease morbidity has been Malaria for decades. Governments have taken various control strategies to mitigate deaths and health costs from Malaria. Some of the common control interventions are: LLINs, IRS, ACT, IPT and vaccines. These are the interventions commonly studied so far in these disease models. However, I propose to incorporate information campaigns and interaction between interventions and human behavior. The model presented here integrates:

- A compartmental disease model (S, I, R) with a force-of-infection term driven by mosquito density and biting rates interacting with human behavior.
- A belief term tracking the evolution of belief about the endemic and prevention effort term that feeds back into transmission.
- Intervention effect modules that modify epidemiological parameters and behavioral drivers depending on coverage.
- ILP optimization module to find the globally optimal selection of interventions under budget constraints. I compare it with a greedy algorithm which mimics a myopic central planner's decisions.

Mathematical Model

State Variables and Notation

For each (representative) district we model continuous-time dynamics over $t \in [0, T]$:

- S(t) = Susceptible Proportion
- I(t) = Infected Proportion
- R(t) = Recovered Proportion
- B(t) = Belief about risk $\in [0, 1]$
- E(t) = Prevention Effort $\in [0, 1]$

Population Constraint: $S(t) + I(t) + R(t) = 1$ (Per district)

Note: u denotes untreated values – provided as subscript.

Force of Infection

The force of infection $\lambda(t)$ is defined as:

$$\lambda(t) = [m \times a^2 \times b \times c \times e^{-\mu\tau}) \times I(t)]/[\mu + a \times c \times I(t)] \times (1 - E(t))$$

Where:

- m = mosquitoes per human (region-dependent)
- a = biting rate (bites per mosquito per day)
- b = mosquito-to-human transmission probability
- c = human-to-mosquito transmission probability
- μ = mosquito mortality (per day)
- τ = extrinsic incubation period (days)
- E(t) = prevention effort (reduces transmission multiplicatively)

The a^2 term here is due to the two-step transmission process: First, infected human to mosquito and then from mosquito to human. Exp(-μτ) is survival through incubation for each mosquito during the incubation period. The denominator accounts for mosquito saturation.

Disease ODEs

The coupled disease–behavior ODEs used in simulation:

$$\frac{dS}{dt} = -\lambda(t) \times S + \rho \times R$$

$$\frac{dI}{dt} = \lambda(t) \times S - \gamma \times I - \delta \times I$$

$$\frac{dR}{dt} = \gamma \times I - \rho \times R$$

Here:

- γ = recovery rate
- δ = disease mortality
- ρ = loss of immunity (return to susceptible state)

Behavioral Dynamics

Belief evolves via social learning, own-experience, and campaign effects. The belief decays with time:

$$\frac{dB}{dt} = \alpha_S \times (1 - I) \times I \times (1 - B) + \alpha_O \times I \times (1 - B) + \alpha_C \times \left(1 + \alpha_{C,boost}\right) \times C(t) \times (1 - B) - \delta_B \times B$$

Where C(t) is campaign intensity (0–1) and α_{C,boost} scales campaign efficacy (depends on coverage in implementation).

Prevention effort is modeled as a deterministic function of belief and infection prevalence:

$$E(t) = min(1, max(0, \theta_B \times B(t) + \theta_I \times I(t) + \theta_i nt - \theta_C \times c_e ffort))$$

Where θ_int is an intervention-induced effort boost (depends on intervention coverage).

Intervention Effects

Each intervention j modifies baseline parameters (examples):

- LLIN: reduces biting a and mosquito density m; adds behavioral boost θ_int
- IRS: reduces m; adds behavioral boost
- IPT: reduces b (susceptibility)
- ACT: increases γ (faster recovery)
- Vaccine: reduces b, increases γ
- Campaign: increases α_C (learning) and directly increases θ_int and C(t)

Coverage κ ∈ {0.2, 0.4, 0.6, 0.8} scales each effect using convex combinations in the code:

$$\theta\_e ff = (1 - \kappa) \times \theta\_u + \kappa \times \theta\_treated$$

Outcome Metrics and Cost-effectiveness

Person-days of Infection

For a simulation horizon [0,T] with population N (per-district):

$PD = N \times \int I(t)\, dt$ evaluated from 0 to T and computed numerically.

In the code, I sum daily I(t) values and multiply by N to get person-days.

Avoided Person-days

Given baseline PD (no intervention) and PD under intervention j,κ:

$$APD_{j,\kappa} = PD_{baseline} - PD_{j,\kappa}$$

Cost Function

Total cost per district for intervention j at coverage κ:

$$C_{j,\kappa,r} = c_j \times \kappa \times N \times \varphi_r$$

Where c_j is per-person cost and φ_r is regional distribution multiplier (low/medium/high).

Cost-effectiveness Ratio

$$CER_{j,\kappa,r} = \frac{APD_{j,\kappa,r}}{C_{j,\kappa,r}}$$

Higher CER implies more health impact per dollar.

Optimization Formulations

Implementation of the ILP

The optimization method I have used primarily is the ILP (Integer Linear Programming). It is implemented in R using lpsolve. It is basically a knapsack problem with a 0/1 formulation. This method guarantees finding the true, global optimum (maximum Avoided Person-Days).

Let decision variables x_{r,j,κ} ∈ {0,1} indicate selecting intervention j at coverage κ in region r.

$$Maximize: \Sigma_r \Sigma_j \Sigma_\kappa APD_{r,j,\kappa} \times x_{r,j,\kappa}$$

$$Subject\ to: \Sigma_r \Sigma_j \Sigma_\kappa C_{r,j,\kappa} \times x_{r,j,\kappa} \leq B$$

Note: The single-action-per-region constraint $\left(\Sigma_{j,\kappa} x_{r,j,\kappa} \leq 1 \; \forall \, r\right)$ was removed in the final ILP implementation to allow for combinations of interventions within a single district (e.g., LLIN + Campaign), as demonstrated by the optimal solution selecting 10 actions for 9 districts.

Greedy Heurisitcs

This might be confusing at first glance but my goal here was to compare between the ideal allocation of interventions vs myopic allocation (something a planner acting on a whim would do). They are not the primary solutions but just comparison benchmark.

1. CE-focused greedy: Ranks all available actions by their Cost-Effectiveness Ratio (APD/C) and sequentially selects the highest-ranking actions while enforcing one action per region until the budget is exhausted. This method provides a fast approximation.
2. Total-benefit (retired): The original total-benefit greedy approach was shown to be brittle and has been replaced by the superior ILP.

Numerical Implementation

ODE Solver

The ODE system is solved using lsoda (via deSolve (ode)) with daily time-steps by default. Typical solver settings in the R code:

- method = lsoda
- rtol = 1e^-5, atol = 1e^-5
- Daily steps: times = seq(0, 365, by = 1)

The code includes robust try-catch fallback for solver failures (returns baseline metrics when the solver fails). I have included this as I was facing solution failures initially.

Paramater Values

Key baseline parameters used:

Parameter	Symbol	Baseline Value
Biting rate	a_u	0.25 (bites/mosquito/day)
Transmission (mosquito → human)	b_u	0.022
Transmission (human → mosquito)	c	0.36
Mosquito mortality	μ	0.095 day^{-1}
Extrinsic incubation	τ	10 days
Recovery untreated	γ_u	1/180 day^{-1}
Immunity waning	ρ_u	1/274 day^{-1}
Mosquito density (dry/moderate/wet)	m	5 / 20 / 35 mosquitoes/person
Population per district	N	10,000 persons

Behavioral Parameters (Baseline)

$$\alpha_S = 0.02, \alpha_O = 0.01, \alpha_C = 0.03, \delta_B = 0.005, \theta_B = 0.6, \theta_I = 0.4, \theta_C = 0.3, c_effort = 0.5$$

Intervention unit costs (USD/person/year) used:

$$LLIN = 1.33, IRS = 2.22, IPT = 1.13, ACT = 4.82, vaccine = 20.66, campaign = 3.50$$

Simulation Workflow:

The key procedural steps are as follows

1. Generate action space (single interventions at multiple coverages + selected combos)
2. For each region and action, run simulate_year() producing final state and person-days
3. Compute baseline PD per region (no intervention)
4. Compute avoided PD and cost for each region-action
5. Apply the ILP solver to find the globally optimal allocation, and run the greedy methods for comparison

6. Produce diagnostic tables and plots

Results and Diagnostics

The R script (Appendix A) includes diagnostic routines:

- test_interventions() to compare final infection, belief and effort for a set of interventions
- test_campaign_fix() validating campaign scaling
- run_complete_optimization() running both the ILP and CE-focused methods and summarizing results to enable comparison between ideal vs myopic planning

Example outputs (from a single run in the working environment) include:

- Final infection rates per action/coverage per climate region
- Avoided person-days and per-action CER
- Selection lists for the ILP and CE-focused heuristics
- Plots: CE scatter (avoided PD vs cost), bar charts of avoided PD by action

Discussion

Strengths

This paper intends to combine mechanistic SIR models with human behavioral aspects. There are behavioral feedbacks (although it has been simplified with paramter dependence because of tractability issues). I have tried my best to incorporate ideas from the economic-epidemiology literature – the key one being the infection variable being dependent on human feedback. I have tried to keep the modules flexible in terms of composition, coverage scaling and campaign interactions. The ILP was chosen to provide a guaranteed global optimum of intervention choices. I have included diagnostics and plots for better visual comparison.

Limitations

The district model is aggregated, and therefore we can't see district (within) heterogeneity. I have not added stochastic terms to the ODE system so that system is deterministic. The ILP can be computationally intensive which is why I needed to greatly simplify the model. Lastly, there is no paramter uncertainty – we have to decide exact values.

Extension Ideas

We could make the budget allocation process dynamic. We could also look at travel-induced transmission and interaction between districts. The best extension however would be to incorporate bayesian elements in deciding the parameters rather than just using fixed values according to our wish. We could also include multiple objective criteria.

Policy Implications

There are three main policy implications:

1) We can see clear impact of information campaigns. In the results, it can be seen that information campaigns account for 26% of the total impact.
2) Even though the CE algorithm was not my main concern, we can see that the ILP is far superior than the CE in terms of yield impact (five times more).
3) For wet regions, we require a combination of campaigns and high coverage of interventions. Previously, only high coverage was required in the main paper.
4) The ILP utilizes the full budget, falls short in CE metrics slightly but has greater impact on health outcomes. The trade-off is not serious.

Results and Diagnostics

Key Optimization Results

With a 5-year budget of $2.3 million, the model produced the following comparative results:

Method	Actions	Total Cost	Avoided PD (5Y)	Budget Used	PD/$1k
ILP (True Optimum)	52	$2,299,300	190,639,524	100.0%	82,912
CE-Focused (Greedy)	9	$319,200	31,954,877	13.9%	100,109

Table: Optimization Method Comparison (5-Year Horizon)

Key Findings:

We can see that the ILP yields almost 500% more impact when compared to the greedy algorithm. Also, campaigns, as I have mentioned earlier, contribute to 26% of the overall impact and appear in one-fourth of the intervention combinations. The ILP exhausts the budget while the CE algorithm does not. In terms of marginal returns, the CE algorithm has much better CE metrics but the ILP gains much more person-days which my main point of concern. Lastly, we can see multiple intervention combinations that are feasible in the same region.

Let's look at the campaing effectiveness:

- Campaign interventions contribute 49,508,795 avoided person-days (26% of total impact).
- Campaigns cost $652,700 (28.4% of total budget).
- Campaign cost-effectiveness: 75,852 PD/$1,000.
- Top campaign interventions target wet regions with low distribution costs.

The main paper emphasised different intervention combinations for different climate types. So to conduct a similar comparison, we see:

- Dry regions: Primarily benefit from ACT-based interventions.
- Wet regions: Require campaigns + high coverage combinations (LLIN campaign, ACT campaign).

- Distribution costs: Medium/high costs reduce CE ratios by 20-25%.

Conclusion

This optimization model has tried to capture epidemiological dynamics, behavioral responses, and resource optimization. I have tried to draw from different streams of literature and merge different ideas into this model. The initial plan was much more ambitious but due to tractability issues, I have had to resort to a parameter dependent model. But I think it still provides a rigorous method for identifying optimal intervention portfolios under budget constraints, while the behavioral component captures important feedback mechanisms often neglected in purely biological models.

To sum up, I would say that the model demonstrates that:

- Optimal allocations differ substantially from heuristic approaches
- Behavioral campaigns provide significant value beyond biological interventions
- Combination strategies outperform single interventions in high-transmission settings
- Climate-specific targeting is essential for efficient resource use

Some of the outcomes have been shown previously in prior papers but the information campaign effectiveness in a behaviorally modified SIR model is novel to the best of my knowledge.

Reference

- Dudley, J. M., et al. (2016). Climate-specific malaria intervention strategies. *Malaria Journal*, 15(1), 234.
- Smith, D. L., et al. (2012). Ross, Macdonald, and a theory for the dynamics and control of mosquito-transmitted pathogens. *PLoS Pathogens*, 8(4), e1002588.
- Griffin, J. T., et al. (2016). Reducing Plasmodium falciparum malaria transmission in Africa: a model-based evaluation of intervention strategies. *PLoS Medicine*, 13(8), e1001983.

Appendix

The complete R code used for implementation is provided below:

```
# MALARIA INTERVENTION OPTIMIZATION MODEL - 5 YEAR VERSION
# Multi-year simulation with ILP and Greedy optimization

library(deSolve)
library(ggplot2)
library(dplyr)
library(tidyr)
library(lpSolve)

`%||%` <- function(a, b) if (!is.null(a)) a else b

# 1. PARAMETERS

params_epidemiology <- list(
  a_u = 0.25, b_u = 0.022, c = 0.36, delta = 4.7895e-5,
  gamma_u = 1/180, m_u_dry = 5, m_u_mod = 20, m_u_wet = 35,
  mu = 0.095, omega = 274, tau = 10, rho_u = 1/274, N = 10000
)

params_behavior <- list(
  alpha_S = 0.02, alpha_O = 0.01, alpha_C = 0.03, delta_B = 0.005,
  theta_B = 0.6, theta_I = 0.4, theta_C = 0.3, c_effort = 0.5
)
```

```
params_interventions <- list(
  cost = c(none = 0, LLIN = 1.33, IRS = 2.22, IPT = 1.13,
           ACT = 4.82, vaccine = 20.66, campaign = 1.50),
  effects = list(
    LLIN = list(a_reduction = 0.80, m_reduction = 0.64, theta_i = 0.2),
    IRS = list(m_reduction = 0.95, theta_i = 0.15),
    IPT = list(b_reduction = 0.786, theta_i = 0.1),
    ACT = list(gamma_multiplier = 18, theta_i = 0.15),
    vaccine = list(b_reduction = 0.773, gamma_multiplier = 3.27, theta_i = 0.1),
    campaign = list(alpha_C_boost = 0.4, theta_i_boost = 0.3, compliance_boost = 0.25)
  ),
  coverage_levels = c(0.2, 0.4, 0.6, 0.8)
)

regions <- expand.grid(
  climate = c("dry", "moderate", "wet"),
  dist_cost = c("low", "medium", "high"),
  stringsAsFactors = FALSE
)

initial_conditions <- list(
  dry = c(S = 0.60, I = 0.15, R = 0.25, B = 0.3, E = 0.2),
  moderate = c(S = 0.15, I = 0.15, R = 0.70, B = 0.5, E = 0.3),
  wet = c(S = 0.10, I = 0.15, R = 0.75, B = 0.4, E = 0.25)
)
```

```
SIM_YEARS <- 5
SIM_DAYS <- 365

# 2. CORE DYNAMICS

calculate_force_of_infection <- function(m, a, b, I_total, E, params) {
  numerator <- m * a^2 * b * params$c * exp(-params$mu * params$tau) * I_total
  denominator <- params$mu + a * params$c * I_total
  lambda <- numerator / denominator
  return(lambda * (1 - min(E, 1)))
}

combined_odes <- function(t, state, params) {
  S <- state["S"]; I <- state["I"]; R <- state["R"]; B <- state["B"]

  m <- params$m; a <- params$a; b <- params$b; c <- params$c
  gamma <- params$gamma; delta <- params$delta; rho <- params$rho
  mu <- params$mu; tau <- params$tau
  alpha_S <- params$alpha_S; alpha_O <- params$alpha_O
  alpha_C <- params$alpha_C; delta_B <- params$delta_B
  theta_B <- params$theta_B; theta_I <- params$theta_I
  theta_C <- params$theta_C; c_effort <- params$c_effort
  theta_i_boost <- params$theta_i_boost %||% 0
  C_intensity <- params$C_intensity %||% 0
  alpha_C_boost <- params$alpha_C_boost %||% 0

  theta_total <- theta_B * B + theta_I * I + theta_i_boost
  E <- min(1, max(0, theta_total - theta_C * c_effort))

  lambda <- calculate_force_of_infection(m, a, b, I, E, params)

  dS <- -lambda * S + rho * R
  dI <- lambda * S - gamma * I - delta * I
  dR <- gamma * I - rho * R

  dB <- alpha_S * (1 - I) * I * (1 - B) +
    alpha_O * I * (1 - B) +
    alpha_C * (1 + alpha_C_boost) * C_intensity * (1 - B) -
    delta_B * B

  return(list(c(dS = dS, dI = dI, dR = dR, dB = dB), E = E))
}

get_intervention_effects <- function(intervention_type, coverage, m_u, params_epi) {
  effects <- list(
    m = m_u, a = params_epi$a_u, b = params_epi$b_u,
    gamma = params_epi$gamma_u, rho = params_epi$rho_u,
    delta = params_epi$delta, mu = params_epi$mu,
    tau = params_epi$tau, c = params_epi$c,
    theta_i_boost = 0, alpha_C_boost = 0, C_intensity = 0
  )

  if (is.null(intervention_type) || intervention_type == "none" || intervention_type == "") {
    return(effects)
```

```
}

interventions <- strsplit(intervention_type, "_")[[1]]

for (int in interventions) {
  if (int %in% c("0.2", "0.4", "0.6", "0.8")) next

  if (int == "LLIN" && int %in% names(params_interventions$effects)) {
    int_effects <- params_interventions$effects[[int]]
    effects$a <- effects$a * (1 - int_effects$a_reduction * coverage)
    effects$m <- effects$m * (1 - int_effects$m_reduction * coverage)
    effects$theta_i_boost <- effects$theta_i_boost + int_effects$theta_i * coverage
  }

  if (int == "IRS" && int %in% names(params_interventions$effects)) {
    int_effects <- params_interventions$effects[[int]]
    effects$m <- effects$m * (1 - int_effects$m_reduction * coverage)
    effects$theta_i_boost <- effects$theta_i_boost + int_effects$theta_i * coverage
  }

  if (int == "IPT" && int %in% names(params_interventions$effects)) {
    int_effects <- params_interventions$effects[[int]]
    effects$b <- effects$b * (1 - int_effects$b_reduction * coverage)
    effects$theta_i_boost <- effects$theta_i_boost + int_effects$theta_i * coverage
  }

  if (int == "ACT" && int %in% names(params_interventions$effects)) {
      int_effects <- params_interventions$effects[[int]]
      current_gamma <- effects$gamma
      target_gamma <- params_epi$gamma_u * int_effects$gamma_multiplier
      effects$gamma <- (1 - coverage) * current_gamma + coverage * target_gamma
      effects$theta_i_boost <- effects$theta_i_boost + int_effects$theta_i * coverage
    }

    if (int == "vaccine" && int %in% names(params_interventions$effects)) {
      int_effects <- params_interventions$effects[[int]]
      effects$b <- effects$b * (1 - int_effects$b_reduction * coverage)
      current_gamma <- effects$gamma
      target_gamma <- params_epi$gamma_u * int_effects$gamma_multiplier
      effects$gamma <- (1 - coverage) * current_gamma + coverage * target_gamma
      effects$theta_i_boost <- effects$theta_i_boost + int_effects$theta_i * coverage
    }

    if (int == "campaign" && int %in% names(params_interventions$effects)) {
      int_effects <- params_interventions$effects[[int]]
      effects$alpha_C_boost <- int_effects$alpha_C_boost * coverage
      effects$theta_i_boost <- effects$theta_i_boost +
        int_effects$theta_i_boost * coverage * (1 + int_effects$compliance_boost * coverage)
      effects$C_intensity <- coverage
    }
  }

  return(effects)
}
```

```r
simulate_year <- function(initial_state, intervention_type, coverage,
                          region_type, days,
                          params_epi = params_epidemiology,
                          params_behav = params_behavior) {

  m_u <- switch(region_type,
                "dry" = params_epi$m_u_dry,
                "moderate" = params_epi$m_u_mod,
                "wet" = params_epi$m_u_wet,
                params_epi$m_u_mod)

  int_effects <- get_intervention_effects(intervention_type, coverage, m_u, params_epi)
  combined_params <- c(int_effects, params_behav)

  y0 <- c(S = as.numeric(initial_state["S"]),
          I = as.numeric(initial_state["I"]),
          R = as.numeric(initial_state["R"]),
          B = as.numeric(initial_state["B"]))

  times <- seq(0, days, by = 1)

  tryCatch({
    solution <- ode(y = y0, times = times, func = combined_odes,
                    parms = combined_params, method = "lsoda")

    final_idx <- nrow(solution)
  final_state <- solution[final_idx, c("S", "I", "R", "B")]


  I_trajectory <- solution[, "I"]
  person_days <- sum(I_trajectory[-1] + I_trajectory[-length(I_trajectory)]) / 2 *
    params_epi$N


  E_vals <- pmin(1, pmax(0,
                        params_behav$theta_B * solution[, "B"] +
                          params_behav$theta_I * solution[, "I"] +
                          combined_params$theta_i_boost -
                          params_behav$theta_C * params_behav$c_effort
  ))


  return(list(
    final_state = final_state,
    person_days = person_days,
    trajectories = as.data.frame(solution),
    effort_trajectory = E_vals,
    success = TRUE,
    intervention = intervention_type,
    coverage = coverage,
    region = region_type
  ))


}, error = function(e) {
  warning(paste("ODE solver failed for", intervention_type, ":", e$message))
  return(list(
      final_state = initial_state,
      person_days = as.numeric(initial_state["I"]) * params_epi$N * days,
      trajectories = NULL,
      effort_trajectory = NULL,
      success = FALSE,
      intervention = intervention_type,
      coverage = coverage,
      region = region_type
    ))
  })
}

# 3. OPTIMIZATION

generate_actions <- function() {
  interventions <- c("none", "LLIN", "IRS", "IPT", "ACT", "vaccine", "campaign")
  actions <- list()

  actions[["none_0"]] <- list(type = "none", coverage = 0, cost = 0)

  for (int in setdiff(interventions, "none")) {
    for (cov in params_interventions$coverage_levels) {
      actions[[paste0(int, "_", cov)]] <- list(
        type = int, coverage = cov,
        cost = params_interventions$cost[int] * cov
      )
    }
  }

  key_combinations <- list(
    c("LLIN", "ACT"),
    c("LLIN", "campaign"),
    c("ACT", "campaign")
  )

  for (combo in key_combinations) {
    combo_name <- paste(combo, collapse = "_")
    for (cov in params_interventions$coverage_levels) {
      cost <- sum(sapply(combo, function(x) params_interventions$cost[x] * cov))
      actions[[paste(combo_name, cov, sep = "_")]] <- list(
        type = combo_name, coverage = cov, cost = cost
      )
    }
  }

  return(actions)
}

precompute_transitions <- function(regions, initial_conditions, actions,
                                   params_epi = params_epidemiology) {
  transitions <- list()
  costs <- list()

  baseline_person_days <- list()
```

```r
for (climate in unique(regions$climate)) {
  cat("  Computing", SIM_YEARS, "-year baseline for", climate, "climate...\n")
  res_none <- simulate_year(
    initial_state = initial_conditions[[climate]],
    intervention_type = "none", coverage = 0,
    region_type = climate, days = SIM_DAYS * SIM_YEARS, params_epi = params_epi
  )
  baseline_person_days[[climate]] <- res_none$person_days
}

total_combinations <- nrow(regions) * length(actions)
counter <- 0

for (i in 1:nrow(regions)) {
  region <- regions[i, ]
  climate <- region$climate
  dist_cost <- region$dist_cost

  cost_multiplier <- switch(dist_cost, "low" = 0.8, "medium" = 1.0, "high" = 1.2)

  for (action_name in names(actions)) {
    counter <- counter + 1
    if (counter %% 10 == 0) {
      cat(sprintf("  Progress: %d/%d (%.1f%%)\r",
                  counter, total_combinations,
                  100 * counter / total_combinations))
    }
```

```
      action <- actions[[action_name]]

      result <- simulate_year(
        initial_state = initial_conditions[[climate]],
        intervention_type = action$type,
        coverage = action$coverage,
        region_type = climate, days = SIM_DAYS * SIM_YEARS,
        params_epi = params_epi
      )

      trans_key <- paste(climate, dist_cost, action_name, sep = "|")
      baseline_pd <- baseline_person_days[[climate]]
      avoided_pd <- max(0, baseline_pd - result$person_days)

      transitions[[trans_key]] <- list(
        initial = initial_conditions[[climate]],
        final = result$final_state,
        person_days = result$person_days,
        avoided_person_days = avoided_pd,
        success = result$success
      )

      costs[[trans_key]] <- (action$cost %||% 0) * params_epi$N * cost_multiplier * SIM_YEARS
    }
  }
cat("\n")
```

```r
  return(list(transitions = transitions, costs = costs))
}

extract_region <- function(key) {
  parts <- strsplit(key, "\\|")[[1]]
  return(paste(parts[1], parts[2], sep = "|"))
}

solve_ilp_optimization <- function(transitions_data, budget, verbose = TRUE) {
  all_trans <- transitions_data$transitions
  all_costs <- transitions_data$costs
  keys <- names(all_trans)

  objective_fn <- sapply(keys, function(k) all_trans[[k]]$avoided_person_days %||% 0)
  cost_vector <- sapply(keys, function(k) all_costs[[k]] %||% 0)

  valid_indices <- which(objective_fn > 0)
  objective_fn <- objective_fn[valid_indices]
  cost_vector <- cost_vector[valid_indices]
  keys_valid <- keys[valid_indices]

  if (length(keys_valid) == 0) {
    if (verbose) cat("ILP: No valid candidates found (APD <= 0).\n")
    return(list(objective = 0, total_cost = 0, num_selected = 0, solution = list()))
  }
```

```r
constraint_matrix <- matrix(cost_vector, nrow = 1)
constraint_direction <- "<="
constraint_rhs <- budget

lp_model <- lpSolve::lp(
  direction = "max",
  objective.in = objective_fn,
  const.mat = constraint_matrix,
  const.dir = constraint_direction,
  const.rhs = constraint_rhs,
  all.bin = TRUE
)

if (lp_model$status != 0) {
  if (verbose) cat(sprintf("ILP Solver failed (Status: %d).\n", lp_model$status))
  return(list(objective = 0, total_cost = 0, num_selected = 0, solution = list()))
}

solution_vector <- lp_model$solution
selected_indices <- which(solution_vector == 1)
selected_keys <- keys_valid[selected_indices]

selected_cost <- sum(cost_vector[selected_indices])
selected_apd <- sum(objective_fn[selected_indices])

selected_regions <- unique(sapply(selected_keys, extract_region))
total_regions <- length(unique(sapply(keys, extract_region)))
```

```
  if (verbose) {
    cat(sprintf("\nINTEGER LINEAR PROGRAM (TRUE OPTIMUM):\n"))
    cat(sprintf("  Actions selected: %d\n", length(selected_keys)))
    cat(sprintf("  Districts covered: %d out of %d\n", length(selected_regions), total_regions))
    cat(sprintf("  Total %d-year cost: $%s\n", SIM_YEARS, format(round(selected_cost),
big.mark = ",")))
    cat(sprintf("  Total %d-year avoided person-days: %s\n", SIM_YEARS,
format(round(selected_apd), big.mark = ",")))
    cat(sprintf("  Remaining budget: $%s (%.1f%% of total)\n",
                format(round(budget - selected_cost), big.mark = ","),
                ((budget - selected_cost) / budget) * 100))
    pd_per_k <- (selected_apd / max(1, selected_cost)) * 1000
    cat(sprintf("  Average cost-effectiveness: %.1f person-days per $1,000\n", pd_per_k))
  }

  return(list(
    objective = selected_apd,
    total_cost = selected_cost,
    num_selected = length(selected_keys),
    solution = setNames(as.list(solution_vector[selected_indices]), selected_keys),
    selected_regions = selected_regions
  ))
}

solve_greedy_ce_optimization <- function(transitions_data, budget, verbose = TRUE) {
  all_trans <- transitions_data$transitions
  all_costs <- transitions_data$costs
```

```r
keys <- names(all_trans)

if (length(all_trans) == 0) {
  return(list(solution = NULL, objective = 0, status = "No transitions"))
}

ce_ratios <- sapply(keys, function(key) {
  cost <- as.numeric(all_costs[[key]] %||% NA)
  avoided <- as.numeric(all_trans[[key]]$avoided_person_days %||% NA)

  if (!is.na(cost) && !is.na(avoided)) {
    if (cost > 0 && avoided > 0) {
      return(avoided / cost)
    } else if (cost == 0 && avoided > 0) {
      return(Inf)
    }
  }
  return(-Inf)
}, USE.NAMES = TRUE)

sorted_keys <- names(sort(ce_ratios[is.finite(ce_ratios)], decreasing = TRUE))

remaining_budget <- budget
solution <- list()
total_avoided <- 0
total_cost <- 0
selected_regions <- character(0)
```

```
  total_regions <- length(unique(sapply(keys, extract_region)))

  for (key in sorted_keys) {
    cost <- as.numeric(all_costs[[key]])
    avoided <- as.numeric(all_trans[[key]]$avoided_person_days)
    region <- extract_region(key)

    if (!is.na(cost) && !is.na(avoided) &&
        cost <= remaining_budget && avoided > 0 &&
        !(region %in% selected_regions)) {

      solution[[key]] <- 1
      selected_regions <- c(selected_regions, region)
      remaining_budget <- remaining_budget - cost
      total_avoided <- total_avoided + avoided
      total_cost <- total_cost + cost
    }
  }

  if (verbose) {
    cat(sprintf("\nCE-FOCUSED OPTIMIZATION (ONE INTERVENTION PER
DISTRICT):\n"))
    cat(sprintf("  Actions selected: %d\n", length(solution)))
    cat(sprintf("  Districts covered: %d out of %d\n", length(selected_regions), total_regions))
    cat(sprintf("  Total %d-year cost: $%s\n", SIM_YEARS, format(round(total_cost), big.mark =
",")))
    cat(sprintf("  Total %d-year avoided person-days: %s\n", SIM_YEARS,
              format(round(total_avoided), big.mark = ",")))
    cat(sprintf("  Remaining budget: $%s (%.1f%% of total)\n",
                format(round(remaining_budget), big.mark = ","),
                remaining_budget / budget * 100))
    cat(sprintf("  Average cost-effectiveness: %.1f person-days per $1,000\n",
                (total_avoided / max(1, total_cost)) * 1000))
  }

  return(list(
    solution = solution,
    objective = total_avoided,
    total_cost = total_cost,
    status = "CE-focused solution",
    remaining_budget = remaining_budget,
    num_selected = length(solution),
    selected_regions = selected_regions
  ))
}

run_complete_optimization <- function(budget) {
  cat("\n\n═\n")
  cat(sprintf("COMPLETE MALARIA INTERVENTION OPTIMIZATION (%d YEARS)\n",
SIM_YEARS))
  cat("═\n\n")

  cat("Configuration:\n")
  cat(sprintf("  Budget (%d Years): $%s\n", SIM_YEARS, format(budget, big.mark = ",")))
  cat(sprintf("  Districts: %d climate x distance combinations\n", nrow(regions)))
  cat(sprintf("  Population: %s per district\n", format(params_epidemiology$N, big.mark = ",")))


  cat("\nStep 1: Generating action space...\n")
  actions <- generate_actions()
  cat(sprintf("  Generated %d possible interventions\n", length(actions)))


  cat("\nStep 2: Computing state transitions...\n")
  total_combinations <- nrow(regions) * length(actions)
  cat(sprintf("  Total simulations: %d districts x %d interventions = %d\n",
              nrow(regions), length(actions), total_combinations))
  cat(sprintf("  Each simulation runs for %d years (%d days total).\n", SIM_YEARS, SIM_DAYS
* SIM_YEARS))


  start_time <- Sys.time()
  transitions_data <- precompute_transitions(regions, initial_conditions, actions)
  end_time <- Sys.time()
  time_taken <- round(as.numeric(difftime(end_time, start_time, units = "mins")), 1)


  cat(sprintf("\n✓ Computation complete in %.1f minutes\n", time_taken))


  cat("\nStep 3: Running optimization algorithms...\n")


  cat("\n--- METHOD 1: Integer Linear Program (True Optimum) ---\n")
  result_ilp <- solve_ilp_optimization(transitions_data, budget, verbose = TRUE)


  cat("\n--- METHOD 2: Cost-Effectiveness Focused (Greedy Approximation) ---\n")
  result_ce <- solve_greedy_ce_optimization(transitions_data, budget, verbose = TRUE)
```

```
cat("\nStep 4: Comparing optimization methods...\n")

ilp_pd_per_k <- (result_ilp$objective / max(1, result_ilp$total_cost)) * 1000
ce_pd_per_k <- (result_ce$objective / max(1, result_ce$total_cost)) * 1000

cat(paste(rep("─", 90), collapse = ""), "\n")
cat(sprintf("%-25s %-10s %-15s %-20s %-10s %-10s\n",
        "Method", "Actions", "Total Cost", "Avoided PD (5Y)", "Used %", "PD/$1k"))
cat(paste(rep("─", 90), collapse = ""), "\n")

cat(sprintf("%-25s %-10d $%-14s %-20s %-10.1f %-10.0f\n",
        "ILP (True Optimum)",
        result_ilp$num_selected,
        format(round(result_ilp$total_cost), big.mark = ","),
        format(round(result_ilp$objective), big.mark = ","),
        (result_ilp$total_cost / budget) * 100,
        ilp_pd_per_k))

cat(sprintf("%-25s %-10d $%-14s %-20s %-10.1f %-10.0f\n",
        "CE-Focused (Greedy)",
        result_ce$num_selected,
        format(round(result_ce$total_cost), big.mark = ","),
        format(round(result_ce$objective), big.mark = ","),
        (result_ce$total_cost / budget) * 100,
        ce_pd_per_k))
```

```
  cat(paste(rep("─", 90), collapse = ""), "\n")

  cost_diff <- result_ilp$total_cost - result_ce$total_cost
  pd_diff <- result_ilp$objective - result_ce$objective

  cat(sprintf("\nComparison Summary (ILP vs. Greedy):\n"))
  cat(sprintf("  Additional cost in ILP: $%s\n", format(round(cost_diff), big.mark = ",")))
  cat(sprintf("  Additional PD avoided (%d Years) in ILP: %s (%.1f%% increase)\n",
              SIM_YEARS,
              format(round(pd_diff), big.mark = ","),
              (pd_diff / max(1, result_ce$objective)) * 100))

  cat("\n═\n")
  cat(sprintf("%d-YEAR OPTIMIZATION COMPLETE\n", SIM_YEARS))
  cat("═\n\n")

  return(list(
    actions = actions,
    transitions_data = transitions_data,
    ilp_optimum = result_ilp,
    ce_focused = result_ce,
    timestamp = Sys.time()
  ))
}

# ANALYSIS FUNCTIONS
```

```
analyze_campaign_impact <- function(results) {
 cat("\n")
 cat("═\n")
 cat("BEHAVIORAL MODEL & CAMPAIGN IMPACT ANALYSIS\n")
 cat("═\n\n")

 # Extract ILP solution
 ilp_keys <- names(results$ilp_optimum$solution)

 # Categorize by intervention type
 has_campaign <- grepl("campaign", ilp_keys)
 campaign_interventions <- ilp_keys[has_campaign]
 non_campaign <- ilp_keys[!has_campaign]

 cat(sprintf("Total interventions selected: %d\n", length(ilp_keys)))
 cat(sprintf("  With campaigns: %d (%.1f%%)\n",
             length(campaign_interventions),
             100 * length(campaign_interventions) / length(ilp_keys)))
 cat(sprintf("  Without campaigns: %d (%.1f%%)\n\n",
             length(non_campaign),
             100 * length(non_campaign) / length(ilp_keys)))

 # Extract campaign contribution to total impact
 campaign_apd <- sum(sapply(campaign_interventions, function(k) {
  results$transitions_data$transitions[[k]]$avoided_person_days %||% 0
 }))
```

```
total_apd <- results$ilp_optimum$objective

cat("Campaign Contribution:\n")
cat(sprintf("  Direct person-days avoided: %s\n",
            format(round(campaign_apd), big.mark = ",")))
cat(sprintf("  Percentage of total impact: %.1f%%\n\n",
            100 * campaign_apd / total_apd))

# Cost analysis
campaign_cost <- sum(sapply(campaign_interventions, function(k) {
  results$transitions_data$costs[[k]] %||% 0
}))

total_cost <- results$ilp_optimum$total_cost

cat("Campaign Investment:\n")
cat(sprintf("  Total campaign costs: $%s\n",
            format(round(campaign_cost), big.mark = ",")))
cat(sprintf("  Percentage of total budget: %.1f%%\n",
            100 * campaign_cost / total_cost))
cat(sprintf("  Campaign cost-effectiveness: %.0f PD/$1k\n\n",
            (campaign_apd / campaign_cost) * 1000))

# Show top campaign interventions
campaign_data <- data.frame(
  key = campaign_interventions,
  apd = sapply(campaign_interventions, function(k) {
      results$transitions_data$transitions[[k]]$avoided_person_days
    }),
    cost = sapply(campaign_interventions, function(k) {
      results$transitions_data$costs[[k]]
    }),
    stringsAsFactors = FALSE
  )

  if (nrow(campaign_data) > 0) {
    campaign_data$ce_ratio <- campaign_data$apd / campaign_data$cost * 1000
    campaign_data <- campaign_data[order(-campaign_data$apd), ]

    cat("Top Campaign Interventions:\n")
    cat(paste(rep("─", 70), collapse = ""), "\n")
    n_show <- min(10, nrow(campaign_data))
    for (i in 1:n_show) {
      row <- campaign_data[i, ]
      cat(sprintf("%-40s %12s %10.0f PD/$1k\n",
                  row$key,
                  format(round(row$apd), big.mark = ","),
                  row$ce_ratio))
    }
  }

  cat("\n")
  invisible(campaign_data)
}
```

```r
# MAIN EXECUTION

BUDGET_TO_RUN <- 2300000
cat("\nStarting complete 5-year optimization...\n")
complete_results <- run_complete_optimization(budget = BUDGET_TO_RUN)

# Analyze campaign impact
analyze_campaign_impact(complete_results)

cat("\n\nModel execution complete!\n")
cat("Results saved in 'complete_results' variable.\n")
cat("\nTo save: saveRDS(complete_results, 'malaria_optimization_5year.rds')\n")
cat("To analyze campaigns: analyze_campaign_impact(complete_results)\n\n")

# ADAPTED PLOT GENERATION FUNCTIONS
# Requires: deSolve, ggplot2, tidyr, dplyr, and all preceding model functions

# --- 1. Function to Calculate Instantaneous Derivatives (dB/dt and dI/dt) ---
# This uses the ODEs' right-hand side directly for plotting the Rate of Change
# and the Phase Plot axes.
calculate_derivatives_for_plot <- function(I, B, S, R, combined_params) {
  # Extract necessary parameters from the combined list
  params <- combined_params
```

```
m <- params$m; a <- params$a; b <- params$b; c <- params$c
gamma <- params$gamma; delta <- params$delta; rho <- params$rho
mu <- params$mu; tau <- params$tau
alpha_S <- params$alpha_S; alpha_O <- params$alpha_O
alpha_C <- params$alpha_C; delta_B <- params$delta_B
theta_B <- params$theta_B; theta_I <- params$theta_I
theta_C <- params$theta_C; c_effort <- params$c_effort
theta_i_boost <- params$theta_i_boost %||% 0
C_intensity <- params$C_intensity %||% 0
alpha_C_boost <- params$alpha_C_boost %||% 0

# Calculate Effort (E) and Force of Infection (lambda)
theta_total <- theta_B * B + theta_I * I + theta_i_boost
E <- min(1, max(0, theta_total - theta_C * c_effort))

# Use the existing function for lambda
lambda <- calculate_force_of_infection(m, a, b, I, E, params)

# --- dI/dt (Actual Prevalence Rate of Change) ---
dI_dt_actual <- lambda * S - gamma * I - delta * I

# --- dB/dt (Belief Rate of Change) ---
# Note: The dB/dt equation uses I (Actual Prevalence) for the feedback loop
dB_dt_belief <- alpha_S * (1 - I) * I * (1 - B) +
  alpha_O * I * (1 - B) +
  alpha_C * (1 + alpha_C_boost) * C_intensity * (1 - B) -
  delta_B * B
  return(data.frame(dI_dt = dI_dt_actual, dB_dt = dB_dt_belief))
}


# --- 2. Main Plot Generation Function ---
generate_belief_plots <- function(initial_state_key = "dry", days = 200,
                                  intervention_type = "none", coverage = 0) {

  # Get initial state from global list
  initial_state <- initial_conditions[[initial_state_key]]
  region_type <- initial_state_key

  # --- A. Setup and Simulation ---
  params_epi_local <- params_epidemiology
  params_behav_local <- params_behavior

  # Get m_u for the specific region
  m_u <- switch(region_type,
               "dry" = params_epi_local$m_u_dry,
               "moderate" = params_epi_local$m_u_mod,
               "wet" = params_epi_local$m_u_wet,
               params_epi_local$m_u_mod)

  # Get intervention effects
  int_effects <- get_intervention_effects(intervention_type, coverage, m_u, params_epi_local)
  combined_params <- c(int_effects, params_behav_local)
```

```
y0 <- c(S = as.numeric(initial_state["S"]),
        I = as.numeric(initial_state["I"]),
        R = as.numeric(initial_state["R"]),
        B = as.numeric(initial_state["B"]))

times <- seq(0, days, by = 1)

# Run ODE solver
solution <- as.data.frame(ode(y = y0, times = times, func = combined_odes,
                              parms = combined_params, method = "lsoda"))

# Calculate derivatives and error for plotting
derivatives <- mapply(calculate_derivatives_for_plot,
                      S = solution$S, I = solution$I, B = solution$B, R = solution$R,
                      MoreArgs = list(combined_params = combined_params),
                      SIMPLIFY = FALSE) %>%
  bind_rows()

solution <- solution %>%
  mutate(
    belief_error = B - I,
    dI_dt = derivatives$dI_dt,
    dB_dt = derivatives$dB_dt
  )

# --- B. Plot Generation ---
```

```
plots <- list()

# Plot 1: Belief vs Actual Prevalence
data_prev <- solution %>%
  select(time, I, B) %>%
  rename("Actual I(t)" = I, "Belief b(t)" = B) %>%
  pivot_longer(-time, names_to = "Type", values_to = "Prevalence")

plots[[1]] <- ggplot(data_prev, aes(x = time, y = Prevalence, color = Type)) +
  geom_line(linewidth = 1) +
  labs(
    title = paste("Belief vs Actual Prevalence\n(", region_type, " Climate)"),
    x = "Time",
    y = "Prevalence",
    color = ""
  ) +
  scale_color_manual(values = c("Actual I(t)" = "red", "Belief b(t)" = "purple")) +
  theme_minimal() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5))

# Plot 2: Belief Error
avg_error <- round(mean(abs(solution$belief_error)), 3)
plots[[2]] <- ggplot(solution, aes(x = time, y = belief_error)) +
  geom_line(color = "brown", linewidth = 1) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray60") +
  labs(
    title = paste("Belief Error (b - I)", "\nAvg Error =", avg_error),
```

```r
    x = "Time",
    y = "Belief Error (b - I)"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))

# Plot 3: Rate of Change: Belief vs Actual
data_rate <- solution %>%
  select(time, dI_dt, dB_dt) %>%
  rename("dI/dt (Actual)" = dI_dt, "dB/dt (Belief)" = dB_dt) %>%
  pivot_longer(-time, names_to = "Type", values_to = "Rate of Change")

plots[[3]] <- ggplot(data_rate, aes(x = time, y = `Rate of Change`, color = Type)) +
  geom_line(linewidth = 1) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray60") +
  labs(
    title = "Rate of Change: Belief vs Actual",
    x = "Time",
    y = "Rate of Change",
    color = ""
  ) +
  scale_color_manual(values = c("dI/dt (Actual)" = "red", "dB/dt (Belief)" = "purple")) +
  theme_minimal() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5))

# Plot 4: Belief vs Actual: Phase Plot
plots[[4]] <- ggplot(solution, aes(x = I, y = B)) +
    geom_path(color = "green", linewidth = 1) +

    geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "gray") +

    geom_point(data = solution[1, ], aes(x = I, y = B), color = "red", size = 3) +

    geom_point(data = solution[nrow(solution), ], aes(x = I, y = B), color = "darkgreen", size = 3)
+

    labs(

      title = "Belief vs Actual: Phase Plot",

      x = "Actual Prevalence I(t)",

      y = "Belief b(t)"

    ) +

    annotate("text", x = 0.05, y = 0.03, label = "Perfect Info", color = "gray40", angle = 45, hjust =
0) +

    annotate("text", x = 0.2, y = 0.2, label = "Trajectory", color = "green", hjust = 0, vjust = 1) +

    # Set limits dynamically but ensure the 45-degree line is visible

    coord_fixed(ratio = 1, xlim = c(0, max(solution$I, solution$B) * 1.1), ylim = c(0,
max(solution$I, solution$B) * 1.1)) +

    theme_minimal() +

    theme(plot.title = element_text(hjust = 0.5))


  return(plots)

}


# --- 3. Example Execution and Display ---


# Load necessary libraries (already included in your full script)

# library(deSolve)

# library(ggplot2)

# library(dplyr)
# library(tidyr)

# library(patchwork) # Recommended for arranging plots


# Example: Run the simulation for the 'dry' climate without intervention

# p <- generate_belief_plots(initial_state_key = "dry", days = 200)


# To display all four plots in a 2x2 grid (requires 'patchwork' or 'gridExtra'):

# if (requireNamespace("patchwork", quietly = TRUE)) {

#     patchwork::wrap_plots(p, ncol = 2)

# } else {

#     print("Install 'patchwork' or use 'gridExtra::grid.arrange' to display all 4 plots.")

#     print(p)

# }
```

Acknowledgement

AI has been very helpful in refining the code and the model. I have faced multiple crashes initially that I have been able to resolve with the help of AI.