Epidemic Model Simulation - SIRS Model

A lightweight, ready-to-use package

Wing Hei (Christy) Lai

2025-11-03

Table of contents

Abstract

To develop a ready-to-use R package for fast, reproducible SIRS (Susceptible–Infectious–Recovered–Susceptible) simulations that support rapid scenario exploration during outbreaks.

We implement discrete-time deterministic and stochastic SIRS models (single and multi-population) with constant or seasonal (β_t), simple intervention windows that scale transmission, and tidy long-format outputs for downstream analysis. Stochastic runs use binomial transitions to quantify uncertainty; results are presented with quantile ribbons and summary metrics (peak infected, peak incidence, final recovered). A lightweight R Markdown demo lets users edit inputs and reproduce all examples; an exploratory dashboard supports quick visual comparisons (not a final product). The work is conducted with The Kids Research Institute (Australia), under The Infectious Disease Ecology and Modelling(IDEM) team.

We use simulated scenarios(no external datasets) to illustrate typical outbreak dynamics and policy-style "what-if" comparisons.

Across scenarios, the package generates standardised and comparable scriptable outputs that are easy to compare and share. Seasonal β_t produces recurrent waves; simple interventions (reducing β_t lower and delay peaks; increasing waning ω shortens inter-wave intervals. Uncertainty ribbons clearly communicate plausible ranges, narrowing with stronger control or larger populations. Multi-population runs enable side-by-side comparisons under common assumptions.

The package provides a simple, fast, and reproducible baseline for outbreak scenario analysis: accessible to non-coders via the demo, extensible for analysts, and suitable for rapid, defensible communication of results and uncertainty.

Introduction

Rapid, transparent epidemic modelling is essential when an outbreak begins: assumptions change quickly, information is incomplete, and decisions cannot wait. This project develops `epi-simulation`, a ready-to-use R package that implements SIRS (Susceptible–Infected–Recovered–Susceptible) dynamics for fast, reproducible scenario analysis. The package unifies deterministic and stochastic simulations across single and multi-population settings, supports time-varying transmission(constant/seasonal β_t and simple intervention windows), and produces tidy outputs with publication-quality plots (S/I/R panels, daily incidence, and deterministic-vs-stochastic overlays with uncertainty ribbons). A lightweight dashboard is included for exploration only—the emphasis is a robust modelling core and scripted, reproducible workflows rather than a finished app.

This work is conducted in collaboration with The Kids Research Institute (Australia), within the Infectious Disease Ecology and Modelling (IDEM) team. IDEM team uses modelling and maps to measure risk for high-impact and neglected infectious diseases (including COVID-19) and provides rapid analyses to policymakers. Our contribution is a clean, extensible SIRS package that helps teams move from basic assumptions to comparable trajectories—and to communicate uncertainty—quickly and clearly.

Background & motivation

The COVID-19 pandemic showed that when an outbreak begins, time is everything. Assumptions shift by the day, data are incomplete, and decision-makers need plausible scenarios fast. In practice, modellers now use many different tools with inconsistent input names and column formats, which makes outputs inconsistent, hard to reuse, and hard to compare across regions or interventions. That fragmentation slows the exact work that must be quickest: turning basic assumptions into clear trajectories and uncertainty ranges that can inform action.

Our project responds to this with a simple, fast, reproducible approach: a unified R package that standardises inputs, simulations, and outputs so teams can generate like-for-like scenarios in minutes, not days. The emphasis is on clarity, comparability, and scripted workflows that others can repeat and extend.

Methodologically, what we do

- **Clear, separate functions + an easy demo:** deterministic vs stochastic, and single- vs multi-population, are provided as separate functions for clarity; an R Markdown demo lets users just edit inputs and run all examples end-to-end without wiring code together.
- **Time-varying scenarios:** support for simple transmission patterns (constant/seasonal) and intervention windows to reflect rapid policy changes.

- **Uncertainty first-class:** stochastic quantile ribbons and summary tables (e.g., peak timing/magnitude) so results show ranges, not just a single line.
- **Tidy, reproducible outputs:** return tables and scripted figure generation for easy comparison, version control, and downstream analysis.
- **Exploration, not a final app:** an optional lightweight dashboard to compare scenarios quickly, while the core deliverable remains code you can rerun.

Who benefits

- **Learners / non-coders:** run outbreak scenarios by editing a few parameters in the `User-Demo.Rmd` and immediately see trajectories and uncertainty.
- **Analysts & researchers:** consistent, tidy outputs enable rapid iteration, sensitivity checks, and integration into larger pipelines.
- **Policy teams & advisors:** fast, side-by-side comparisons of interventions or regions to support timely, defensible decisions.

SIRS Model

The SIRS model(STATE and (SPF) [3]) is a type of mathematical model used in epidemiology to describe the dynamics of an infectious disease in a population by dividing it into three compartments:

- **Susceptible(S)** means who have no immunity from the disease.
- **Infectious(I)** means who have the disease and can spread it to susceptibles.
- **Recovered (R)** means who have recovered from the disease and are immune.

The SIRS model extends the basic SIR model by allowing individuals in the Recovered (R) compartment to lose their immunity over time and return to the Susceptible (S) compartment. This is particularly relevant for diseases where immunity is not lifelong, such as influenza or certain strains of coronavirus.

Think of it as a loop: people move from S $\rightarrow$ I $\rightarrow$ R $\rightarrow$ S when immunity wanes. This cyclical nature captures the ongoing risk of reinfection in the population.

Three rates drive these movements:

- β is the transmission rate, governing how quickly susceptibles become infected.
- γ is the recovery rate, determining how fast infectious individuals recover and gain immunity.

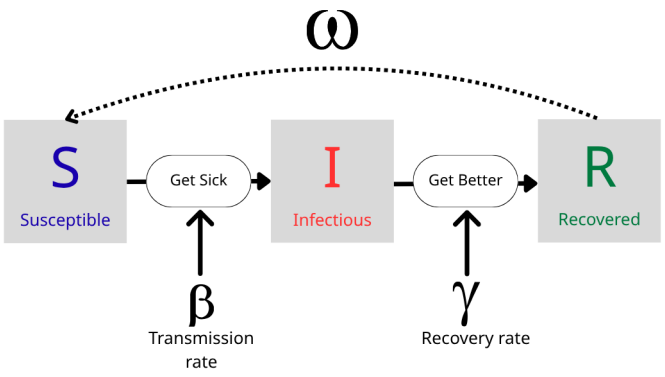

Figure 1: SIRS Model. Source: Christy Lai (2025).

- ω is the rate of loss of immunity, dictating how quickly recovered individuals become susceptible again.

Throughout, $S + I + R$ equals the total population size (we usually work in proportions so $S + I + R = 1$).

The SIRS model is typically represented by a set of ordinary differential equations (ODEs) that describe the rates of change for each compartment over time:

$$\begin{aligned}\frac{dS}{dt} &= -\beta SI + \omega R \\ \frac{dI}{dt} &= \beta SI - \gamma I \\ \frac{dR}{dt} &= \gamma I - \omega R\end{aligned}$$

The logic is straightforward:

- **Susceptibles (S)** decrease as they get infected ($-\beta SI$) but increase as recovered individuals lose immunity and return to susceptibility ($+\omega R$).
- **Infectious (I)** increase as susceptibles get infected ($+\beta SI$) but decrease as they recover ($-\gamma I$).
- **Recovered (R)** increase as infectious individuals recover ($+\gamma I$) but decrease as they lose immunity and become susceptible again ($-\omega R$).

Assumptions of this baseline SIRS model include:

- **Homogeneous mixing:** everyone has an equal chance of contacting everyone else.
- **Constant population size:** no births, deaths, or migration.

- **Instantaneous transitions:** no delays in moving between compartments.

Time-varying transmission rate (t)

In this package, β represents the transmission probability. In real outbreaks, that rate doesn't stay constant. It changes with:

- **Seasonality** (climate, humidity, indoor crowding)
- **Behaviour** (holidays, school terms, work-from-home)
- **Policy** (masking, gathering limits, vaccination campaigns)
- **Pathogen/host factors** (immune waning interacting with behaviour)

Treating β as time-varying β_t, lets us reflect these drivers and ask realistic "what-if" questions (e.g., "What if we reduce contacts by 30% for 4 weeks?").

Options we support include:

- **Constant β:** simplest baseline; good for pedagogy or short horizons.
- **Seasonal β_t:** smooth oscillation around a base level to mimic seasonal forcing (amplitude < 1, optional phase shift).
- **Intervention windows:**piecewise scaling of β_t over specified day ranges to represent policies/behaviour changes.

Mapping to the code:

- Use `make_beta()` to generate constant or seasonal β_t. base = average level, amplitude = size of fluctuation (0–1), phase = timing (days).
- Use `adjust_beta()` to scale β_t within windows (e.g., reduce by 30% during days 50–80).

For more details, example code and plots, please refer to the `make_beta()` and `adjust_beta()` sections in `Helper functions` or the `User-Demo.Rmd` file inside the repository.

Package Overview

This package focuses on fast SIRS simulation, clear visualisation, and concise summaries. Functions are grouped into three families:

Simulators – deterministic and stochastic engines for single and multi-population settings.

Plotting – S/I/R panels, daily incidence, stochastic ribbons, multi-population views, and a lightweight dashboard for quick comparisons.

Summaries & helpers – tidy coercion, cumulative incidence, headline metrics (peaks/final size), and utilities to build or adjust beta.

The dashboard is not a final product, it helps you to explore and further analysis. The main focus of this package is to provide a robust core set of simulation and plotting functions that can be scripted and reproduced.

Table 1: Epi-simulation package function catalog

Category	Function	Description
Simulators	`simulate_sirs_det()`	Deterministic single population.
	`simulate_sirs_stoch()`	Stochastic single population (many rounds).
	`simulate_sirs_multi()`	Deterministic multi-population.
	`simulate_sirs_multi_stoch()`	Stochastic multi-population.
Plotting	`plot_sirs()`	Single-pop SIR/overlay/incidence views.
	`plot_stoch()`	Single-pop Stochastic ribbons/lines & incidence.
	`plot_multi()`	Multi-pop S/I/R/incidence (combined or faceted).
	`plot_dashboard()` + `arrange_dashboard()`	Build a multi-panel dashboard.
	`plot_det_vs_stoch()`	Compare deterministic vs stochastic outputs.
Summaries & helpers	`summarize_sim()`	Headline outbreak metrics (peaks, final sizes, etc.).
	`make_beta()`	Create time-varying beta (constant, seasonal).
	`adjust_beta()`	Scale beta within specified day windows.
	`to_tidy()`	Coerce outputs to a tidy long table (time, group, sim, state, value).
	`cumulative_incidence()`	Cumulative incidence over time.
	`sanity_check()`	Verify S + I + R 1 at each time step
	`attack_rate()`	Cumulative attack rate over the simulation horizon
	`reff_from_sim()`	Calculate effective reproduction number over time.

Table 2 summarises which panels are available in each scenario, depending on the model type.

Table 2: Plotting options for different scenarios

Scenario	Options
Deterministic	SIR, Daily incidence, Overlay, Both
Stochastic	SIR Daily incidence, Overlay, Both
Multiple Population	S, I, R, Daily incidence
Dashboard	SIR, S, I, R, Daily incidence, Beta, Parameters

The functions `plot_det_vs_stoch()`, `attack_rate()`, and `reff_from_sim()` are outside the scope of this document. Readers seeking an in-depth treatment should refer to Varun's companion report available on GitHub.

Installation

The development version of epi-simulation can be installed from GitHub.

First, download the package source code from GitHub.

- Option A - Download ZIP
  - Click Code Download ZIP, then unzip locally.
- Option B - Clone via Terminal
  - Open your RStudio and paste the following into Terminal.

Then Open the `epi-simulation.Rproj` file to load the project in RStudio.

Next, load the functions in R:

We haven't released this as a formal R package yet; please source the R files directly. Sorry for the inconvenience caused.

How to use the package

We designed a Rmarkdown demo with different scenario to illustrate how to use the functions in epi-simulation package. Just 3 simple steps to run the demo:

1. Edit your inputs in the code chunk `EDIT YOUR INPUTS HERE` section.
2. Load the package functions by running the code chunk `Load package functions`.

3. Run each section to reproduce the figures and tables.

- Not familiar with coding? can directly edit the inputs in `User-Demo.Rmd` and run the code chunks.
- Advanced users? Use the demo as a starting point for custom scripts and deeper analysis.

Demo Rmarkdown

Here is the demo snapshot, for full code please check the `User-Demo.Rmd` file in the GitHub.

Hi, there!
This is a demo for all the user inputs you can edit to run the various models and simulations. Inside we drafted different sections for each scenario (A to H).

Sections

A) Deterministic (constant beta)

B) Deterministic (seasonal beta)

C) Stochastic (constant beta)

D) Stochastic (seasonal beta)

E) Multi-pop deterministic

F) Multi-pop stochastic

G) Dashboards with self arrangement

H) Summaries & helpers functions

Explore the code chunks below and edit the inputs! Have fun :)

For installation please check the `README.md` instructions.

Edit your inputs here

```
# ---- EDIT YOUR INPUTS HERE -------------------
# Common components
n_times <- 60      # Total days to simulate
pop     <- 100000  # Population size
I_init  <- 10      # Initial number of infectious individuals
beta    <- 0.75    # Transmission rate
gamma   <- 1/7     # Recovery rate
omega   <- 1/30    # Rate of loss of immunity(R -> S)
```

```r
# Seasonal beta (for B, D)
season_base  <- 0.70   # Base transmission rate
season_amp   <- 0.25   # Amplitude of seasonal variation (0   amplitude < 1)
season_phase <- 30     # Phase shift in days

# Stochastic (C,D,F)
n_sims   <- 200   # Number of independent simulation runs (columns).
epsilon  <- 1e-4  # External infection pressure.
alpha    <- 0.3   # Reporting rate
seed     <- 42    # Random seed for reproducibility

# Multi-pop (E, F)
mp_n_times <- 180                      # Total days to simulate
mp_pops    <- c(100000, 500, 200000)   # Population sizes
mp_Iinit   <- c(100, 5, 1000)          # Initial infectious individuals
mp_beta    <- 0.20                     # Transmission rate
mp_gamma   <- 1/7                      # Recovery rate
mp_omega   <- 1/30                     # Rate of loss of immunity(R -> S)

# Multi-pop stochastic (F)
mp_sims    <- 200      # Number of stochastic simulations
mp_epsilon <- 0        # Noise parameter
mp_alpha   <- NULL     # Scaling factor for stochasticity
mp_seed    <- 99       # Random seed for reproducibility

# Customise uncertainty ribbons(Default to 95%)
ribbon_probs <- c(0.025, 0.975)

# Customise colors for multiple populations
my_pop_cols <- c("#E41A1C", "#377EB8", "#4DAF4A")
```

Load package functions

Next will be A to H sections.

Please check the full code in the `User-Demo.RMD` file.

Deterministic model

Deterministic model assumes that the outcome is fully determined by the parameter values and the initial conditions. There is no randomness involved in the development of future states of the system.

Table 3 shows the basic input parameters, edit them in the `Edit your inputs here` section to see how the changes affect the simulation results.

Table 3: Deterministic Model Parameter Definitions

Parameter	Description
n_times	Total days to simulate
pop	Population size
I_init	Initial number of infectious individuals
beta	Transmission rate
gamma	Recovery rate
omega	Rate of loss of immunity (R $\rightarrow$ S)

After you run `simulate_sirs_det()` function, it will return a list containing:

- **params:** A list of input parameters used in the simulation.
- **results:** A data frame containing the simulation results with columns for time, susceptible (S), infectious (I), and recovered (R) proportions.

```
 [1] 0.0001000000 0.0001607068 0.0002582577 0.0004149998 0.0006668123
 [6] 0.0010712645 0.0017206374 0.0027626174 0.0044329544 0.0071064137
[11] 0.0113747515 0.0181621624 0.0288862409 0.0456568875 0.0714560460
[16] 0.1101199148 0.1657136922 0.2406180331 0.3318376557 0.4269078451
[21] 0.5046186031 0.5457481351 0.5465659410 0.5186673446 0.4769640431
[26] 0.4317730478 0.3883863397 0.3489988836 0.3142468171 0.2840562486
[31] 0.2580608938 0.2357990866 0.2168039038 0.2006420174 0.1869279727
[36] 0.1753268056 0.1655513340 0.1573572584 0.1505376156 0.1449173257
[41] 0.1403481618 0.1367042615 0.1338781967 0.1317775690 0.1303220791
[46] 0.1294410192 0.1290711387 0.1291548450 0.1296387104 0.1304722645
[51] 0.1316070596 0.1329960007 0.1345929321 0.1363524714 0.1382300752
[56] 0.1401823126 0.1421673126 0.1441453407 0.1460794513 0.1479361587
```

The returned list is very long so we designed a `to_tidy()` function to convert the simulation output to tidy format for easier plotting and analysis. For more details please check the `Helper functions` section.

Plotting

In deterministic model, we can use `plot_sirs()` function to visualise the simulation results. The function provides different plot options including "overlay", "sir", "incidence", and "both_side" for users to choose the best visualisation to tell their story.

"sir" option shows the S, I, R compartments over time.

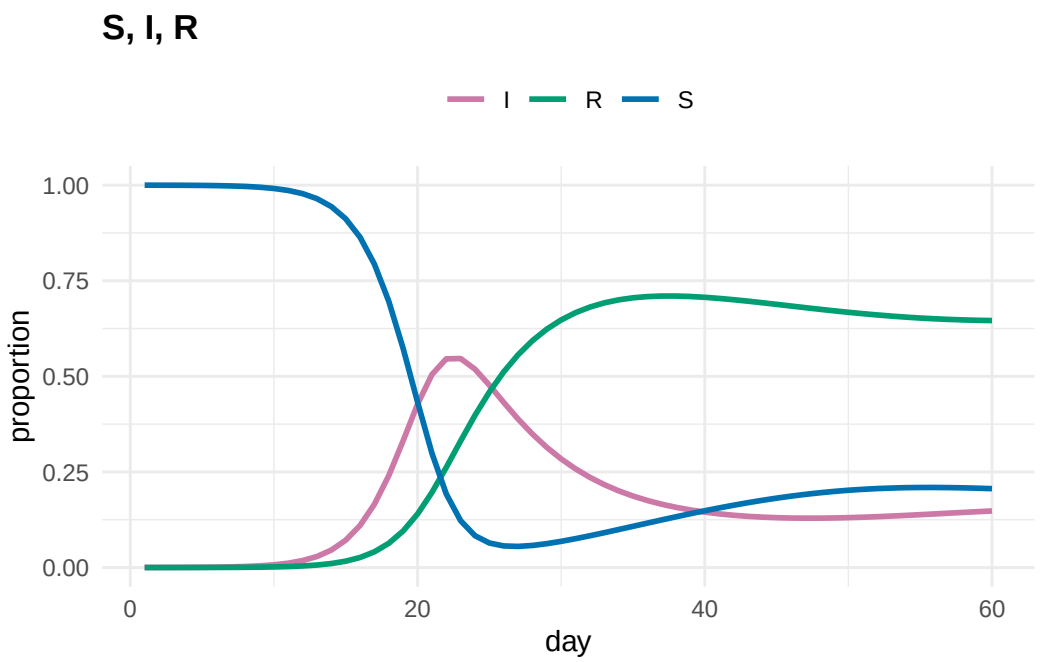

Figure 2: Deterministic SIRS plot - S, I, R compartments over time.

"incidence" option shows the daily new infections over time.

"overlay" option shows the S, I, R compartments and daily incidence in one plot.

"both_side" option shows the S, I, R compartments on the left side and daily incidence on the right side.

If you want to save the figures, you can use `ggsave()` function after plotting.

Stochastic model

Stochastic model incorporates randomness and uncertainty into the simulation process. It recognizes that real-world epidemics are influenced by random events and variations, leading to different possible outcomes even with the same initial conditions and parameters.

Table 4 shows the additional input parameters for stochastic model, edit them in the `Edit your inputs here` section to see how the changes affect the simulation results.

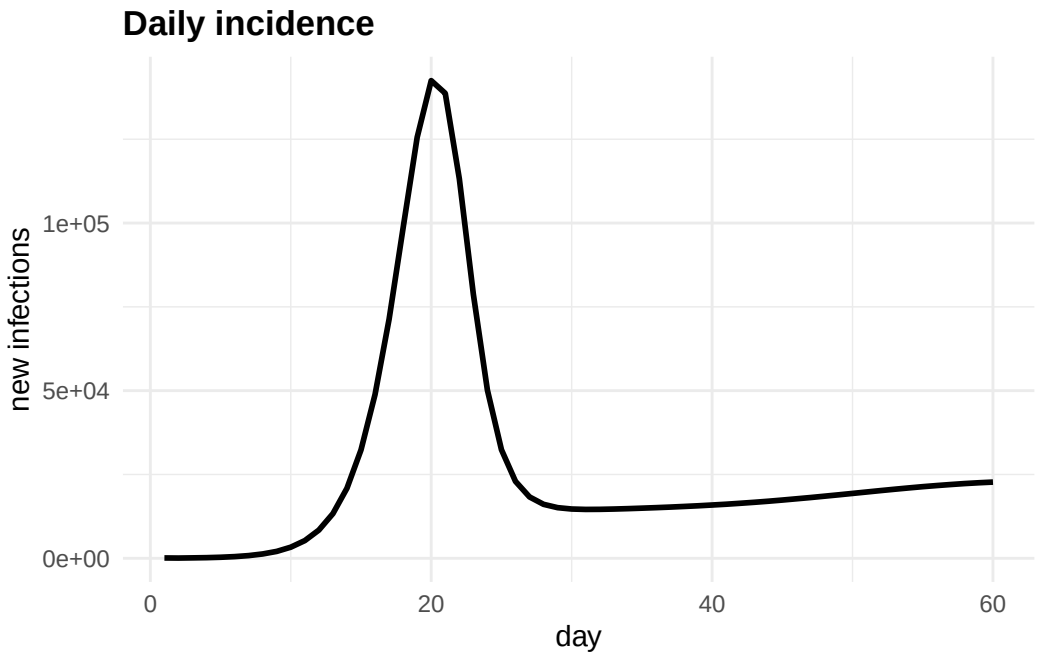

Figure 3: Deterministic SIRS plot - daily incidence over time.

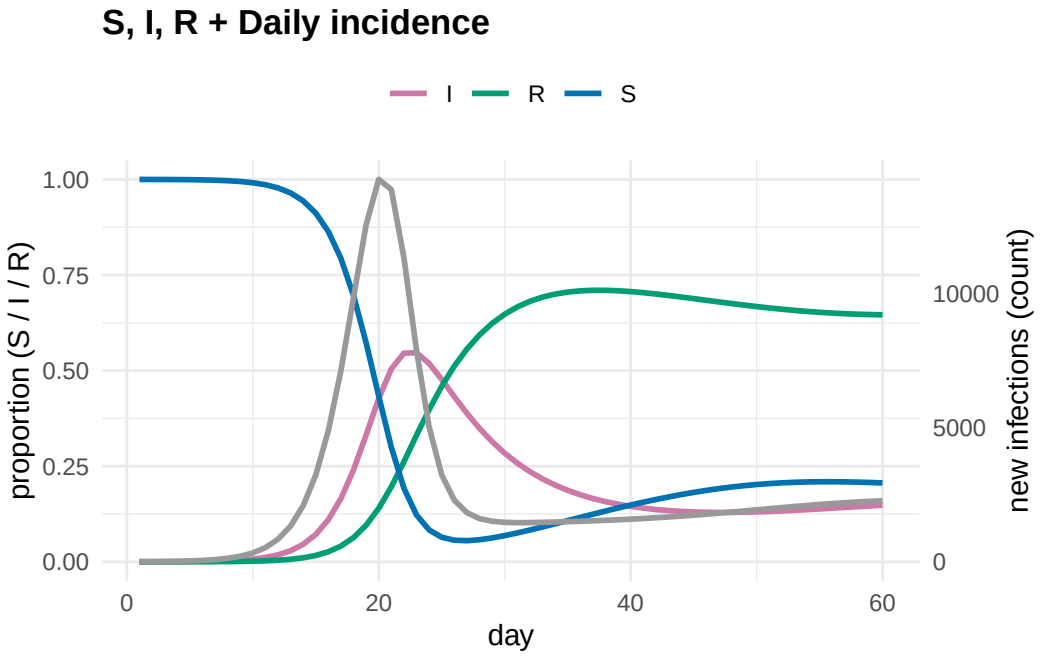

Figure 4: Deterministic SIRS plot - overlay style.

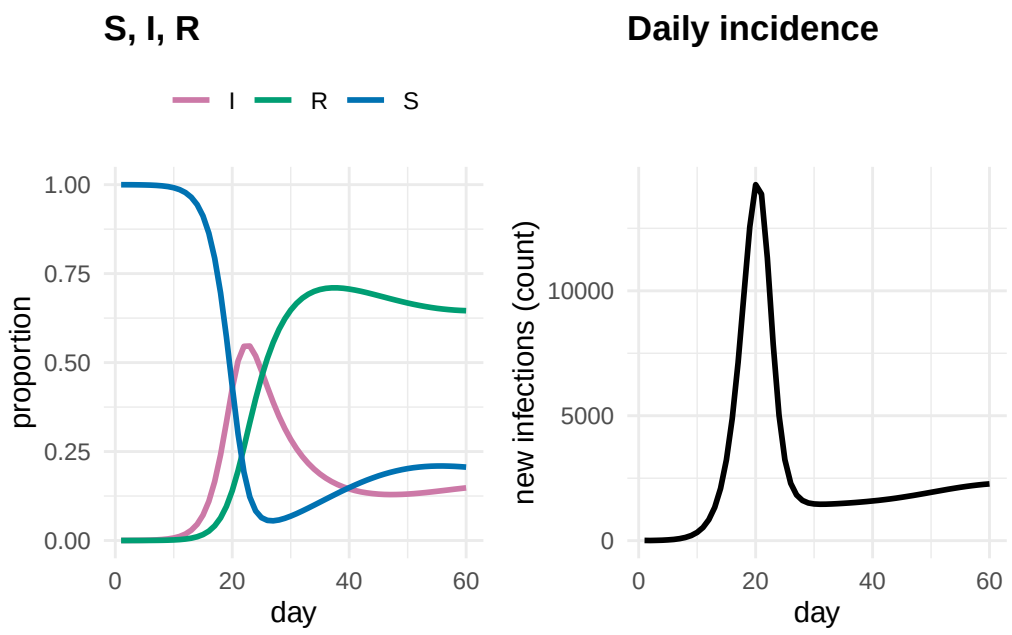

Figure 5: Deterministic SIRS plot - both side style.

Table 4: Stochastic Model Parameter Definitions

Parameter	Description
n_sims	Number of independent simulation runs
epsilon	External infection pressure
alpha	Reporting rate
seed	Random seed for reproducibility
ribbon_probs	Customise uncertainty ribbons(Default to 95%)

The `n_sims` parameter specifies the number of independent simulation runs to be performed. Each simulation run represents a separate realization of the stochastic process, allowing for the exploration of variability and uncertainty in the epidemic dynamics. By running multiple simulations, users can obtain a distribution of possible outcomes, which can be used to assess the range of potential epidemic trajectories and to quantify uncertainty in key metrics such as peak incidence, total infections, and duration of the outbreak.

If you simulate 10 days with `n_sims = 5`, the output (e.g., cases) is a 10×5 matrix: rows = days (1–10) and columns = simulation runs (1–5). Each cell is the value for that day in that run. From this matrix you can compute daily quantiles for uncertainty ribbons, plus means/medians and other summaries.

```
       [,1]  [,2]  [,3]  [,4]  [,5]
 [1,] 0.005 0.005 0.005 0.005 0.005
 [2,] 0.007 0.008 0.005 0.007 0.006
 [3,] 0.008 0.011 0.008 0.005 0.006
 [4,] 0.009 0.015 0.009 0.007 0.007
 [5,] 0.012 0.018 0.013 0.009 0.008
 [6,] 0.013 0.020 0.017 0.011 0.009
 [7,] 0.017 0.023 0.019 0.011 0.008
 [8,] 0.018 0.025 0.021 0.016 0.012
 [9,] 0.021 0.027 0.023 0.019 0.014
[10,] 0.022 0.029 0.025 0.024 0.018
```

The **epsilon(ϵ)** parameter represents the external infection pressure, which accounts for the influence of infections originating from outside the modeled population. This parameter is crucial for capturing the dynamics of disease spread in scenarios where there is a risk of new infections being introduced from external sources, such as travelers or neighboring communities. Usually the range of ϵ is within 0 to 1 and it should be relatively small compared to the transmission rate β. (which means $\epsilon << \beta$).

The **alpha(α)** parameter is the reporting rate, which reflects the proportion of actual infections that are detected and reported in the simulation. This parameter is important for modeling the observed data accurately, as not all infections may be captured due to underreporting, asymptomatic cases, or limitations in testing capacity. The value of α typically ranges from 0 to 1, where 0 indicates no reporting (all infections go undetected) and 1 indicates perfect reporting (all infections are detected and reported).

For **reproducibility**, setting a random seed using the `seed` parameter ensures that the stochastic simulations can be replicated exactly. By initializing the random number generator with a specific seed value, users can obtain consistent results across multiple runs of the simulation, which is essential for debugging, validation, and comparison of different scenarios.

Uncertainty ribbons are visual representations of the variability in the simulation outcomes. The `ribbon_probs` parameter allows users to customize the quantiles used to create these ribbons, providing insights into the range of possible epidemic trajectories. By default, the ribbons are set to represent a 95% confidence interval (2.5th to 97.5th percentiles), showing where the majority of simulation results fall. However, users can adjust these values to explore different levels of uncertainty in the results.

After you run `simulate_sirs_stoch()` function, it will return a list containing:

- **params:** A list of input parameters used in the simulation.
- **results:** A data frame containing the simulation results with columns for time, simulation number (sim), susceptible (S), infectious (I), and recovered (R) proportions.

```
PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installe
```

Table 5 we used `to_tidy()` function to convert the simulation output to tidy format for easier plotting and analysis. We suggest using `DT::datatable()` function to create an interactive table to display the simulation results.

Plotting

For visualising the stochastic simulation results, we can use `plot_stoch()` function. This function plots options are similar to `plot_sirs()` function but with uncertainty ribbons to represent the variability across multiple simulation runs.

User can choose "overlay", "SIR", "incidence", and "both" with customisable uncertainty ribbons.

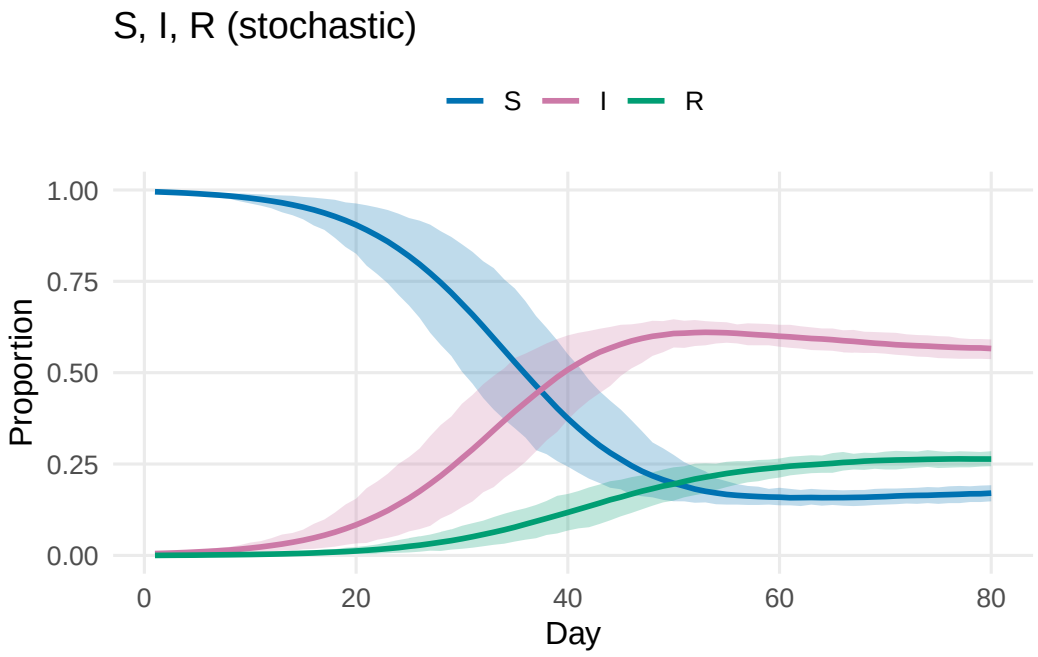

Figure 6: Stochastic SIRS plot - S, I, R compartments with uncertainty ribbons.

Figure 6 shows the S, I, R compartments over time with uncertainty ribbons.

Multi-population model

Multi-population model simulates the disease dynamics across multiple distinct populations or groups. Each population may have its own characteristics, such as population size, transmis-

Table 5: Stochastic Model Simulation Results (First Simulation Round)

Show 10 entries Search:

time	group	sim	state	value
1	1	1	S	0.995
2	1	1	S	0.993
3	1	1	S	0.992
4	1	1	S	0.992
5	1	1	S	0.991
6	1	1	S	0.99
7	1	1	S	0.987
8	1	1	S	0.984
9	1	1	S	0.979
10	1	1	S	0.976

Showing 1 to 10 of 320 entries

Previous 1 2 3 4 5 … 32 Next

sion rates, and recovery rates. The model allows for interactions between these populations, capturing the spread of the disease across different communities or regions.

Table 6: Multi-population Model Parameter Definitions

Model	Parameter	Description
Deterministic	mp_n_times	Total days to simulate
	mp_pops	Population sizes
	mp_Iinit	Initial infectious individuals
	mp_beta	Transmission rate
	mp_gamma	Recovery rate
	mp_omega	Rate of loss of immunity (R -> S)
Stochastic	mp_sims	Number of independent simulation runs
	mp_epsilon	External infection pressure
	mp_alpha	Reporting rate
	mp_seed	Random seed for reproducibility
Colors	my_pop_cols	Self-defined colors for multiple populations
Probability interval	ribbon_probs	Customise uncertainty ribbons(Default to 95%)

Table 6 shows the input parameters for multi-population model, you can edit them in the `Edit your inputs here` section to see how the changes affect the simulation results.

The parameters for multi-population model are similar to single population model. The only three inputs different are:

- `mp_n_times`, `mp_pops`:the population sizes and initial infectious, will be one value per group.

For example, if you have 3 populations with sizes 100,000, 500, and 200,000, you would set `mp_pops = c(100000, 500, 200000)`. Similarly, if the initial infectious individuals are 100, 5, and 1000 for these populations respectively, you would set `mp_Iinit = c(100, 5, 1000)`.

- `my_pop_cols`: You can define a vector of colors to represent each population in the plots.

For example, `my_pop_cols = c("#E41A1C", "#377EB8", "#4DAF4A")` assigns specific colors to three populations.

The output of `simulate_sirs_multi()` and `simulate_sirs_multi_stoch()` functions are similar to single population model but with an additional dimension for populations. The results will include the simulation outputs for each population separately, allowing for comparison of disease dynamics across different groups.

Plotting

`plot_multi()` function are used to visualise the multi-population simulation results. This function provide options to plot the S, I, R compartments and daily incidence for each population, either in a combined style or faceted style.

- `group_style = "combined"`: This option overlays the results from all populations in a single plot, using different colors to distinguish between populations.
- `group_style = "facet"`: This option creates separate subplots for each population, better visualisation for stochastic simulations with uncertainty ribbons.
- `per_million = TRUE/FALSE`: This option allows users to choose whether to display the incidence per million individuals or as absolute numbers.
- `show_bands = TRUE/FALSE`: This option allows users to choose whether to display uncertainty ribbons in the plots.

The following will use the multi-population stochastic simulation as an example.

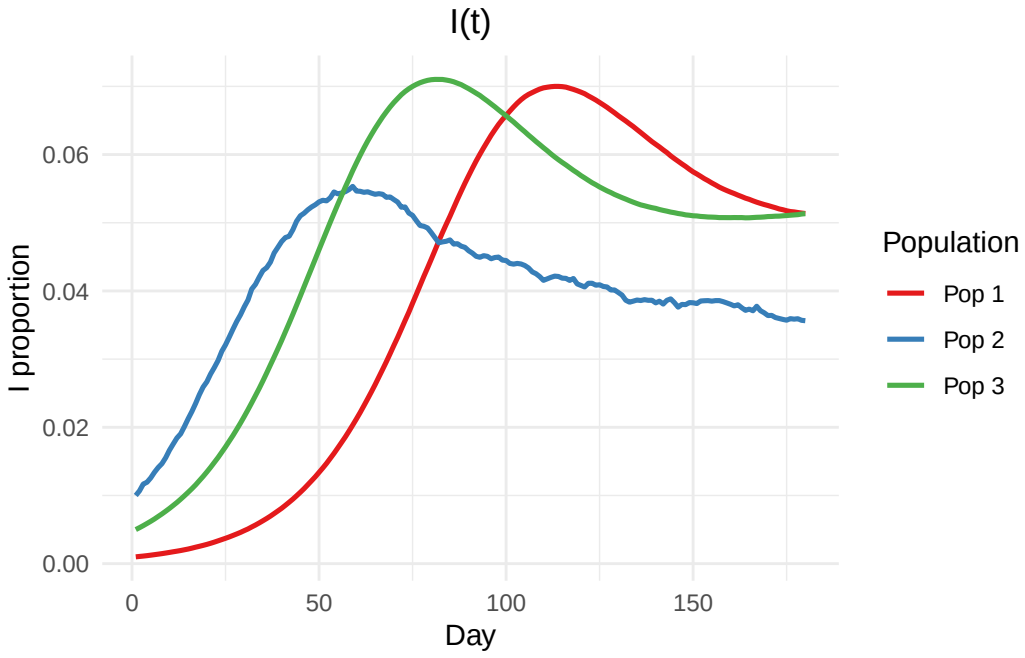

Figure 7: Multi-population stochastic plot - combined style.

Figure 7 shows the combined style plot for multi-population stochastic simulation.

Figure 8 shows the facet style plot for multi-population stochastic simulation. The uncertainty ribbons for population 2 is huge because the population size and intial infectious are very small

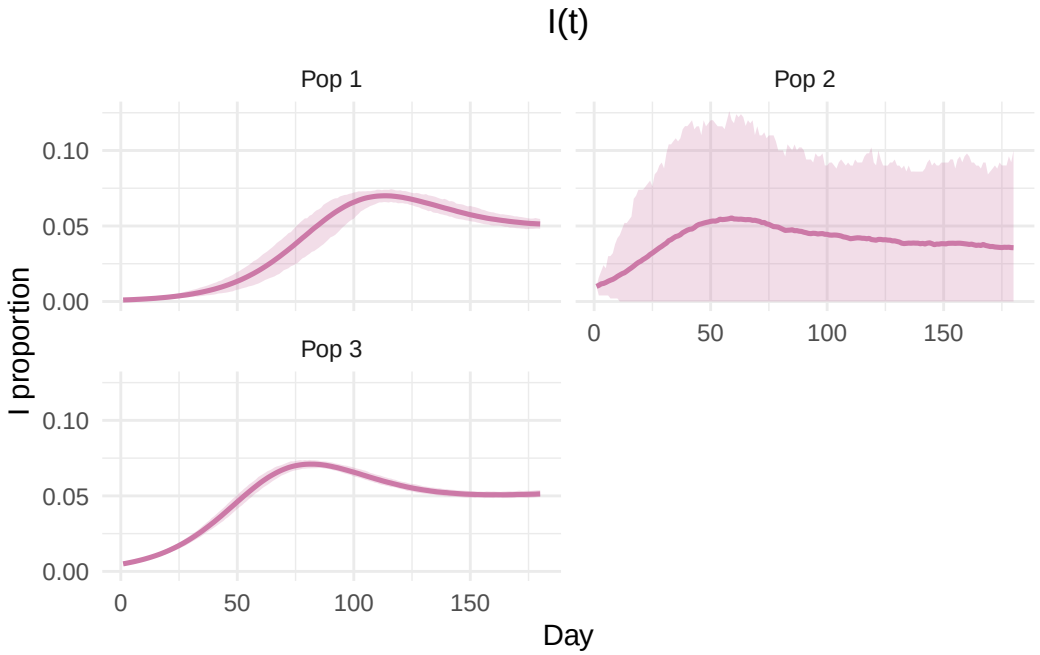

Figure 8: Multi-population stochastic plot - facet style.

(500 and 5 respectively) in this example. We suggest users to avoid simulating very small populations to get more realistic results.

Dashboard function with self arrangement(Still in development)

This function allows users to create a comprehensive dashboard that combines multiple plots into a single view. It provides an overview of the epidemic dynamics across different populations and compartments, facilitating easy comparison and analysis.

`plot_dashboard()` is still under development, current draft version support plot options including "SIR", "S", "I", "R", "incidence", "beta", and "params".

"params" panel will also show at console the simulation parameters used in the model. User can quickly check if the inputs are correct and copy them for record purpose or reproduction.

Here is the example:

```
  Simulation parameters

• n_times: 365
• P: 1
```

```
• pop_vec: 1
• I_init: 10
• beta: [vector]
• gamma: 0.1429
• 1/gamma (days): 7.00
• omega: 0.03333
• 1/omega (days): 30.00
• epsilon: -
• alpha: -
• n_sims: 1
• stochastic: FALSE
```

Created on 2025-11-02 with reprex v2.1.1

`arrange_dashboard()` function allows users to arrange the selected plots into a specified layout, such as a grid or custom arrangement. This function provides flexibility in organizing the visualizations to suit the user's preferences and analysis needs.

Inside the `arrange_dashboard()` function,

- `layout` allows users to specify the desired arrangement of the plots, such as the number of rows and columns in a grid layout. For example, `layout = c(2, 3)` would arrange the plots in a 2x3 grid.
- `collect_legend` is a logical parameter that determines whether to collect and display a single legend for all plots in the dashboard. If set to `TRUE`, the function will combine the legends from individual plots into one, which can help reduce clutter and improve the overall appearance of the dashboard. If set to `FALSE`, each plot will retain its own legend.

Figure 9 shows the deterministic dashboard 2x2 arrangement with basic SIR curve, Daily incidence, beta and parameters panel.

Figure 10 shows the multi-population stochastic dashboard 2x3 arrangement with S, I, R compartments, Daily incidence, beta and parameters panel. Beta panel show a straight line because we used constant beta in this example.

Figure 11 shows the multi-population stochastic dashboard 2x2 arrangement with S, I, R compartments and Daily incidence panel in combined style. We turned off the uncertainty ribbons`show_bands= FALSE`, so this only shows the mean trend across all simulation runs. We set `collect_legend = FALSE`, you can see each plot has its own legend which is different to Figure 9 and Figure 10.

To sum up, you can explore by changing the `layout`, `group_style`, `show_bands`, and `collect_legend` parameters to create customised dashboards that best suits your analysis

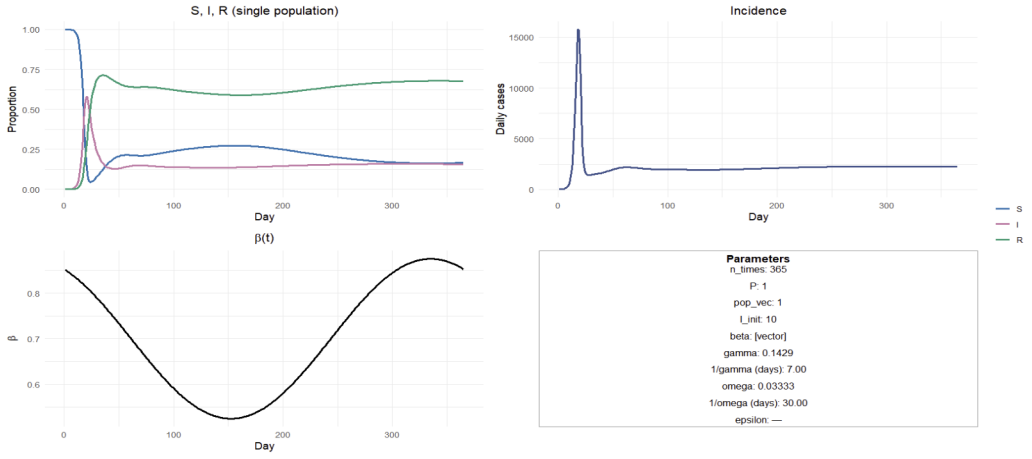

Figure 9: Deterministic dashboard 2x2 arrangement.

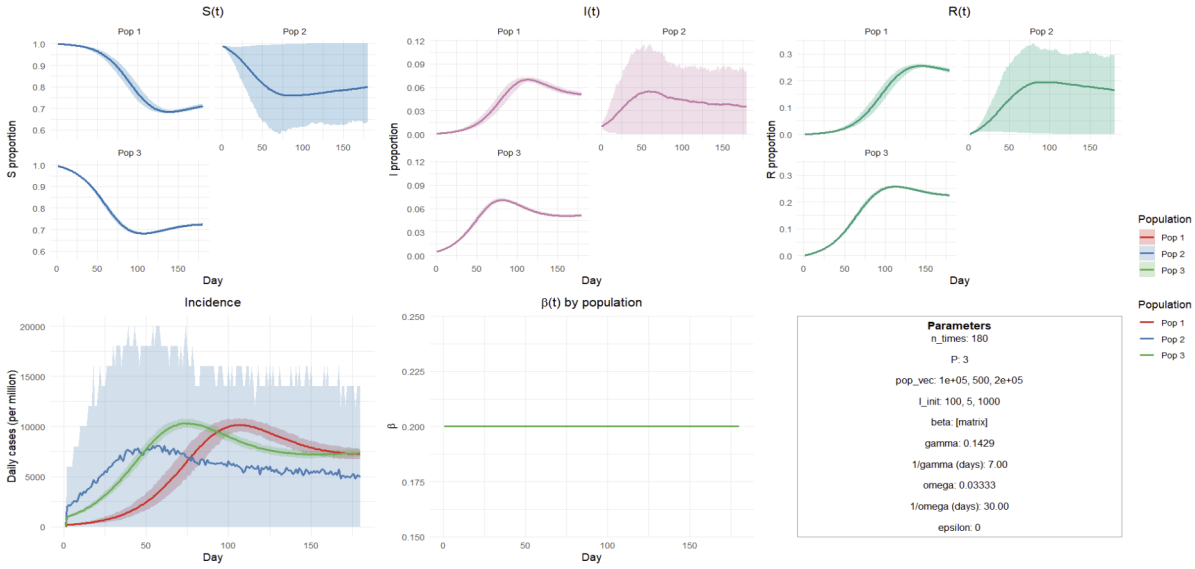

Figure 10: Multi-population stochastic dashboard 2x3 arrangement.

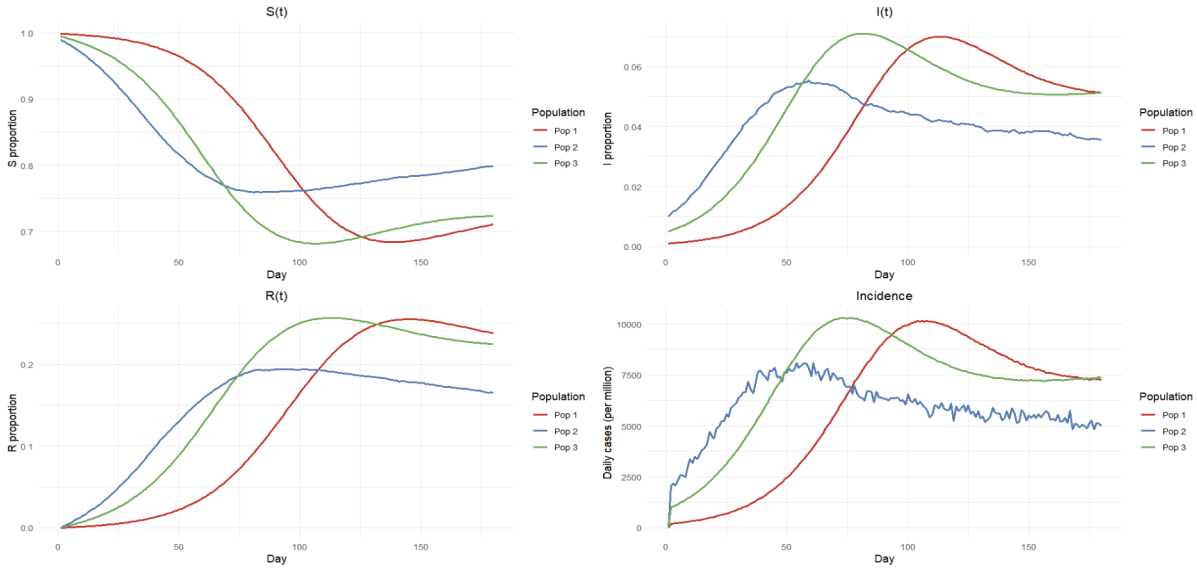

Figure 11: Multi-population stochastic dashboard 2x2 arrangement (combined style).

needs. The flexibility in layout and legend collection allows for customization based on the specific requirements of your study.

Summaries & helper functions

Summary functions

After running the simulation functions, we can use `summarize_sim()` to generate summary statistics from the simulation results. This function provides user have a quick, readable snapshot—especially of the peak infection.

Table 7: Summary Table Columns

Column	Description
sim	Simulation Rounds
pop	Populations
peak_I	Peak infected proportion
peak_I_day	Peak infected Day
peak_incidence	Peak incidence(Count)
peak_incidence_day	Peak incidence Day
final_R	Final Recovered proportion

If user want to run a large amount simulation rounds with many population sizes, the ouput table will be large. So inside the `User-Demo.Rmd` make it more user friendly, we use `DT`package. By using the datatable you can search and filter interactively.

For example in Table 8, we show the summary statistics from multi-population stochastic simulation. There is 3 populations and run 200 independent simulation rounds, which will return out 600 entries.

Helper functions

`to_tidy()`: Convert simulation output to tidy format for easier plotting and analysis.

After you run the simulation function, they will return a long list so we can use `to_tidy()` to convert it to a tidy data frame table for further analysis:

- **Time:** simulation date.
- **Group:** populations size.
- **Sim:** independent simulation runs.
- **State:** S, I, R compartments.
- **Value:** the number of individuals in each compartment at each time point for each simulation run.

Table 9 shows an example of tidy data frame from multi-population stochastic simulation output, filtered for group 1, state "I" and first 10 rows only.

`make_beta()`: Create time-varying transmission rate(beta) with seasonal pattern.

This function generates a vector of transmission rates(β) that vary over time according to a seasonal pattern. Users can specify the base transmission rate, amplitude of seasonal variation, and phase shift to model seasonal effects on disease transmission.

Table 10: make_beta() Function Parameter Definitions

Parameter	Description
n_times	Total time points to generate beta values for
mode	Type of beta pattern: 'constant' or 'seasonal'
base	Base transmission rate
amplitude	Amplitude of seasonal variation (0 amplitude $<$ 1)
phase	Phase shift in days

Figure 12 shows an example of seasonal beta over time with a base transmission rate of 0.7, amplitude of 0.25, and phase shift of 30 days.

Table 8: Summary Statistics from Multi-population Stochastic Simulation

Show 10 entries Search:

sims	pop	peak_I	peak_I_day	peak_incidence	peak_incidence_day	final_R
1	1	0.07265	117	1096	110	0.23481
1	2	0.108	80	12	74	0.16
1	3	0.069735	86	2074	72	0.22495
2	1	0.06643	122	1009	114	0.24156
2	2	0.084	128	11	72	0.126
2	3	0.07078	79	2083	79	0.222595
3	1	0.07188	112	1087	103	0.24514
3	2	0.11	95	11	95	0.144
3	3	0.06895	89	2089	81	0.22742
4	1	0.07451	106	1115	101	0.23156

Showing 1 to 10 of 600 entries

Previous 1 2 3 4 5 … 60 Next

Table 9: Tidy Data Frame from Multi-population Stochastic Simulation Output

	time	group	sim	state	value
1	1	1	1	I	0.00100
2	2	1	1	I	0.00112
3	3	1	1	I	0.00124
4	4	1	1	I	0.00127
5	5	1	1	I	0.00129
6	6	1	1	I	0.00145
7	7	1	1	I	0.00156
8	8	1	1	I	0.00156
9	9	1	1	I	0.00172
10	10	1	1	I	0.00185

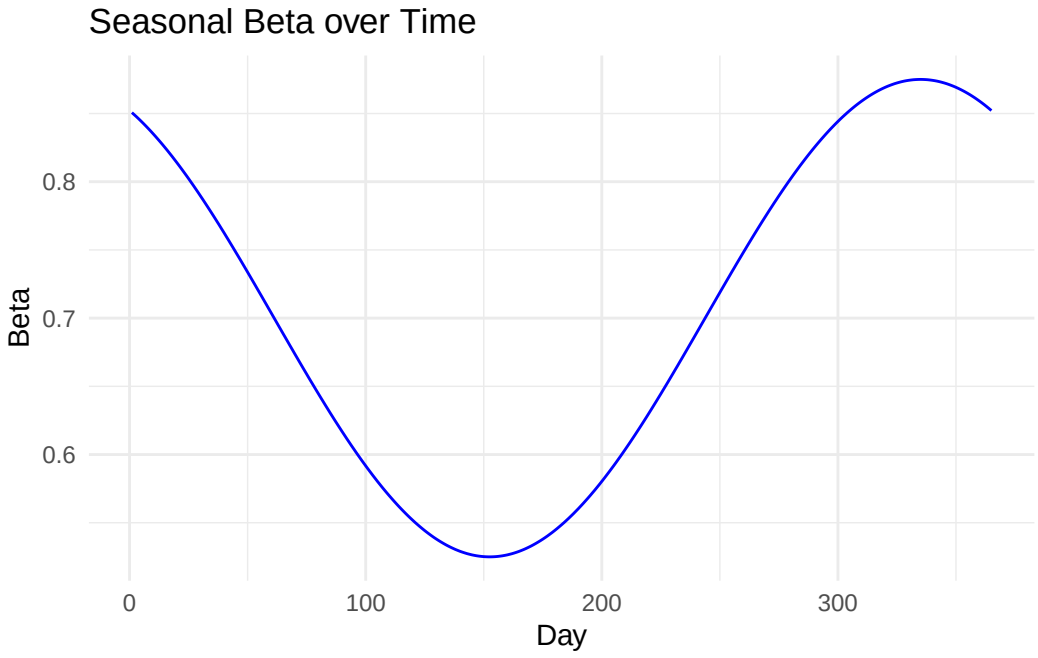

Figure 12: Seasonal Beta over Time

`adjust_beta()`: Adjust transmission rate(beta) by a scaling factor.

This function allows users to modify the transmission rate(beta) within specified day windows by applying scaling factors. This is useful for simulating interventions or changes in transmission dynamics over time.

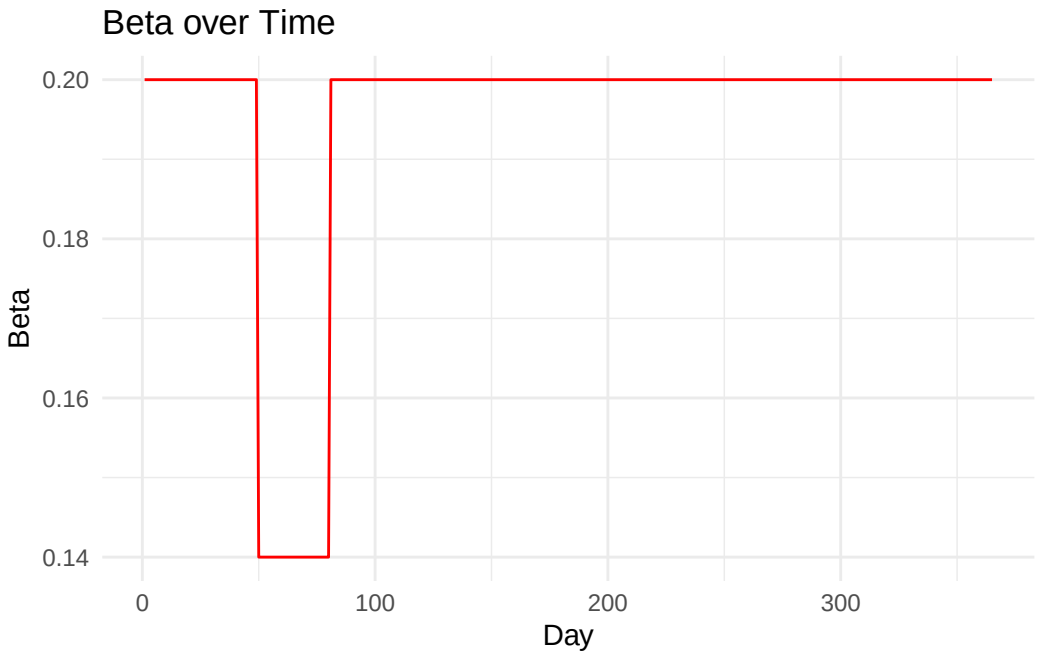

Figure 13: Adjusted Beta over Time

Figure 13 This is an example of adjusted beta over time starting from day 50 to day 80 with a scaling factor of 0.7. The initaial beta is 0.2, after scaling the beta becomes 0.14(0.2 * 0.7) during the specified period.

`cumulative_incidence()`: Calculate cumulative incidence from simulation output.

This function computes the cumulative number of new infections over time based on the simulation results. It helps to understand the overall impact of the epidemic in terms of total cases.

`sanity_check()`: Verify S + I + R 1 at each time step.

This function checks that the sum of the proportions in the S, I, and R compartments is approximately equal to 1 at each time step, ensuring the integrity of the simulation results.

```
[1] TRUE
```

Table 11: Cumulative Incidence from Simulation Output

Show 10 entries Search:

time	sim	group	cum_cases
1	2	1	0
2	2	1	15
3	2	1	27
4	2	1	44
5	2	1	61
6	2	1	77
7	2	1	104
8	2	1	125
9	2	1	149
10	2	1	170

Showing 1 to 10 of 540 entries

Previous 1 2 3 4 5 … 54 Next

Already considered inside the core simulation functions, we would like to drop this function in the future version.

Limitations

- **Model scope.** The project currently implements **SIRS only**; models with a latent stage (e.g., **SEIR/SEIRS**) are not included.
- **No data fitting.** We do not provide parameter estimation or calibration to real data. Examples use **simulated scenarios**, so results are illustrative rather than empirical. Users may adapt the code to fit their own datasets.
- **Dashboard status.** The dashboard is **exploratory** and under active development; some features are incomplete.
- **Project maturity.** This is **source-first research code** still in development and not yet released as a formal R package (e.g., not on CRAN). Installation is via the repo source.

Future developments

Since we still are developing this package, there are several planned improvements for future versions.

Planned improvements include:

- For all the plotting function, for "SIR','S', "I" and "R" are showing the proportion, we would like to add an option is showing the real number of individuals in each compartment.
- Dashboard function is still under development, we would like to add more plot options. For example add `plot_det_vs_stoch()` function to the dashboard so user can compare the deterministic and stochastic results in one panel.
- Added plotting function for `summarize_sim()` output, so user can visualise the summary statistics better. For example, plot the distribution of peak infection across multiple simulation rounds.
- Consider combine `make_beta()` and `adjust_beta()` functions into one function with more flexible options.
- `sanity_check()` function already considered inside the core simulation functions, we would like to drop this function in the future version.
- Package release. Finalise all the improvements, adding descriptions, unit tests, and vignettes to prepare for formal release package.

Conclusion

This report presents a lightweight, ready-to-use SIRS toolkit that turns core epidemiological assumptions into clear, like-for-like outputs for rapid scenario exploration. By standardising inputs and outputs across deterministic and stochastic engines, supporting time-varying transmission (β_t) (seasonality and intervention windows), and returning tidy tables with uncertainty-aware visuals (quantile ribbons, headline summaries), the package makes it straightforward to compare scenarios and communicate results responsibly. The workflow remains accessible to non-coders via an R Markdown demo while staying fully scriptable for analysts; the dashboard provides quick exploratory views without claiming to be a finished decision tool.

Findings from simulated scenarios illustrate expected patterns—seasonal β_t produces recurrent waves; simple interventions that reduce β_t lower and delay peaks; faster waning (ω) shortens inter-wave intervals—and show how uncertainty narrows with stronger control or larger populations. At the same time, the scope is intentionally lean: SIRS only (no SEIR/SEIRS), no data fitting or parameter uncertainty, homogeneous mixing, and an exploratory dashboard. Planned enhancements—counts as a plotting option, richer dashboard panels (including det-vs-stoch overlays), summary visualisations, a unified β_t utility, and package hardening—will further improve usability and comparability. Overall, the toolkit offers a simple, fast, and defensible baseline for outbreak scenario analysis, well suited to teaching, rapid "what-if" work, and transparent communication when time matters.

Acknowledgements

This report is utilized the tidyverse(Wickham et al. [4]), DT(Xie, Cheng, and Tan [5]), kableExtra(Zhu [6]), plotly(Sievert [2]), R.utils(Bengtsson [1]) for creating the visual included in this report. The developing version are available on GitHub. I also acknowledge the use of ChatGPT to improve the text, grammar, and spelling.

Appendix

Contribution Statement

Contributors	Functions	Description
Christy	`simulate_sirs_det()`	Deterministic single population.
	`simulate_sirs_stoch()`	Stochastic single population (many rounds).
	`simulate_sirs_multi()`	Deterministic multi-population.
	`simulate_sirs_multi_stoch()`	Stochastic multi-population.

Contributors	Functions	Description
	`plot_sirs()`	Single-pop SIR/overlay/incidence views.
	`plot_stoch()`	Single-pop Stochastic ribbons/lines & incidence.
	`plot_multi()`	Multi-pop S/I/R/incidence (combined or faceted).
	`plot_dashboard()` + `arrange_dashboard()`	Build a multi-panel dashboard.
	`summarize_sim()`	Compare deterministic vs stochastic outputs.
	`make_beta()`	Headline outbreak metrics (peaks, final sizes, etc.).
Varun	`plot_det_vs_stoch()`	Create time-varying beta (constant, seasonal).
	`adjust_beta()`	Scale beta within specified day windows.
	`to_tidy()`	Coerce outputs to a tidy long table (time, group, sim, state, value).
	`cumulative_incidence()`	Cumulative incidence over time.
	`sanity_check()`	Verify S + I + R 1 at each time step
	`attack_rate()`	Cumulative attack rate over the simulation horizon
	`reff_from_sim()`	Calculate effective reproduction number over time.

References

[1] Henrik Bengtsson. *R.utils: Various Programming Utilities*. R package version 2.13.0. 2025. URL: https://CRAN.R-project.org/package=R.utils.

[2] Carson Sievert. *Interactive Web-Based Data Visualization with R, plotly, and shiny*. Chapman and Hall/CRC, 2020. ISBN: 9781138331457. URL: https://plotly-r.com.

[3] THE STATE and PEACE BUILDING FUND (SPF). *An Introduction to Deterministic Infectious Disease Models*. 2021. URL: https://documents.worldbank.org/en/publication/documents-reports/documentdetail/888341625223820901/an-introduction-to-deterministic-infectious-disease-models.

[4] Hadley Wickham et al. "Welcome to the tidyverse". In: *Journal of Open Source Software* 4.43 (2019), p. 1686. DOI: 10.21105/joss.01686.

[5] Yihui Xie, Joe Cheng, and Xianying Tan. *DT: A Wrapper of the JavaScript Library 'DataTables'*. R package version 0.33. 2024. URL: https://CRAN.R-project.org/package=DT.

[6] Hao Zhu. *kableExtra: Construct Complex Table with 'kable' and Pipe Syntax*. R package version 1.4.0. 2024. URL: https://CRAN.R-project.org/package=kableExtra.