Package 'SIR'

December 28, 2025

Type Package

Title Simulation and Inference for Deterministic Epidemic Models

Version 0.1.1

Description Tools for simulating and analyzing deterministic compartmental epidemic models using ordinary differential equations. The package currently supports classical SIR and SIRS models, including cumulative infection tracking and stochastic observation processes based on Poisson or negative binomial likelihoods. The output of simulations is returned as a dedicated S3 object with associated plotting and summary methods, facilitating reproducible epidemic modeling workflows.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

Imports deSolve,
stats

Suggests ggplot2,
devtools,
roxygen2,
knitr,
rmarkdown

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

URL `https://github.com/vcastellar/SIR`

BugReports `https://github.com/vcastellar/SIR/issues`

Contents

`compute_growth_metrics` *Compute short-horizon growth metrics from an infection time series*

Description

Computes two related growth metrics from a univariate time series of (typically) **cumulative** counts:

1. **Daily percent increase** (`inc_pct`): for each time step t,

   $$inc\_pct_t = (x_{t+1} - x_t)/x_t$$

   This is the one-step *relative* change, using the current value x_t as the denominator.

2. **Doubling time in days** (`doubling_days`): first computes a 4-point **left-aligned** moving average of `inc_pct` (call it α_t), then converts that average growth rate into an implied doubling time under constant-rate exponential growth:

   $$\alpha_t = mean(inc\_pct_t, inc\_pct_{t+1}, inc\_pct_{t+2}, inc\_pct_{t+3})$$

   $$doubling\_days_t = \log(2)/\log(1 + \alpha_t)$$

The doubling-time formula comes from solving $2 = (1 + \alpha)^k$ for k. If $1 + \alpha_t \leq 0$ or $\alpha_t = 0$ then the logarithm or division may yield non-finite values; infinite results are replaced with `NA`.

Usage

```
compute_growth_metrics(infected)
```

Arguments

`infected`	A univariate time series (typically a `ts`) of counts ordered in time. In most epidemiological uses this is a **cumulative** count series, but the function will work for any numeric series with positive values.

Details

Windowing / alignment

The percent increase is computed for adjacent pairs (x_t, x_{t+1}). The 4-point moving average of `inc_pct` is **left-aligned**, meaning the value at position t summarizes growth from t through $t + 3$.

Practical notes

- If `infected` contains zeros, `inc_pct` will be `Inf` or `NaN` at those points. This will propagate into `alpha` and `doubling_days`.
- Negative or decreasing series can produce $\alpha < 0$, for which $\log(1 + \alpha)$ may be undefined when $1 + \alpha \leq 0$.

Value

A named list with three numeric vectors:

inc_pct Vector of length `length(infected) - 1` with the one-step relative increases.

doubling_days Vector of length `length(infected) - 4` with the implied doubling time (in days) computed from the 4-point moving average of `inc_pct`.

alpha Vector of length `length(infected) - 4` with the 4-point moving average of `inc_pct`.

Examples

```
infected <- ts(c(100, 110, 121, 133, 146, 161))
out <- compute_growth_metrics(infected)
out$inc_pct
out$doubling_days
```

`fit_epi_model` *Fit an* `epi_model` *to incidence data by likelihood minimization*

Description

Fits a deterministic compartmental epidemic model (an object of class `"epi_model"`, such as `SIR_MODEL` or `SIRS_MODEL`) to observed **incidence count data** by minimizing a **negative log-likelihood**.

The function returns a fitted model object of class `"fit_epi_model"`, which contains the estimated parameters together with additional information about the optimization process, the model definition, and the initial conditions used for fitting.

Usage

```
fit_epi_model(
  x,
  model = SIR_MODEL,
  distr = c("poisson", "negbin"),
  init = list(I = 10, N = 1e+06),
  control = list(maxit = 500),
  n_starts = 30,
  control_ms = list(maxit = 100),
  seed = 1,
  ...
)
```

Arguments

`x`	Numeric vector of observed incidence counts (e.g. daily cases).
`model`	An `epi_model` object to be fitted (e.g. `SIR_MODEL` or `SIRS_MODEL`). Defaults to `SIR_MODEL`.
`distr`	Character string specifying the observation distribution used in the likelihood. One of `"poisson"` or `"negbin"`.
`init`	Named list defining the initial conditions through `init$N` (population size) and `init$I` (initial number of infectious individuals).
`control`	List of control parameters passed to `optim()` in the final optimization.
`n_starts`	Integer. Number of random starting points used in the multi-start initialization.
`control_ms`	List of control parameters passed to `optim()` during the multi-start phase.
`seed`	Optional integer seed for reproducible multi-start initialization.
`...`	Reserved for future extensions.

Details

Fitting approach:
Internally, the function constructs an objective function using an internal helper (see `objective_epi`, not exported) that:

- solves the model ODE system using `deSolve::ode()` and `model$rhs`;
- computes expected incidence as increments of the cumulative state $C(t)$ via `diff(C)`;
- evaluates a Poisson or Negative Binomial log-likelihood for the observed incidence vector `x`.

To improve numerical stability and reduce sensitivity to initial values, the optimization is initialized via an internal **multi-start** strategy (see `multi_start_optim`, not exported), which samples random initial parameter vectors within the bounds defined by the model and retains the best solution as the starting point for the final optimization.

Data and time grid:
The input vector `x` must represent incidence counts on an equally spaced time grid (typically daily). The internal objective uses a time grid `times = 0:length(x)` so that `diff(C)` has the same length as `x`.

Model requirements:
The supplied `model` must:

- be an object of class `"epi_model"`;
- provide an ODE right-hand side function compatible with `deSolve::ode()`;
- define parameter names via `model$par_names`;
- define parameter bounds via `model$lower` and `model$upper`;
- include a cumulative state variable `C(t)` in the ODE output.

Value

An object of class "`fit_epi_model`" with components including:

par Named numeric vector of fitted model parameters.

model The `epi_model` object that was fitted.

optim The full object returned by `stats::optim()`.

value Final value of the negative log-likelihood.

convergence Convergence code returned by `optim()`.

message Optional convergence message from `optim()`.

init List of initial condition arguments supplied by the user.

ini0 Named numeric vector of initial state values used internally.

See Also

`simulate_epi`, `SIR_MODEL`, `SIRS_MODEL`, `optim`, `ode`

Examples

```
## Not run:
## Simulate a SIRS epidemic and fit the model to observed incidence
sim2 <- simulate_epi(
  model = SIRS_MODEL,
  n_days = 200,
  parms = c(beta = 0.30, gamma = 0.10, omega = 0.02),
  init_args = list(N = 1e6, I0 = 20, R0 = 0),
  obs = "poisson",
  rho = 1,
  seed = 22
)
plot(sim2)

x <- sim2$incidence_obs$inc
plot(x, type = "l", xlab = "Day", ylab = "Incidence")

fit <- fit_epi_model(x, model = SIRS_MODEL, init = list(I = 10, N = 1e6))
fit
fit$par

## Compare fitted incidence to observed incidence
times <- 0:length(x)
ini0  <- c(S = 1e6 - 10, I = 10, R = 0, C = 10)

out <- deSolve::ode(
  y = ini0,
  times = times,
  func = SIRS_MODEL$rhs,
  parms = fit$par,
  method = "lsoda"
)
```

```
lines(as.data.frame(out)$incidence, col = "red")

## End(Not run)
```

`new_epi_model` *Create a new epidemic model object*

Description

Constructs an object of class `"epi_model"` that encapsulates all the information required to define and simulate a deterministic compartmental epidemic model (such as SIR, SIRS, or user-defined extensions).

An `epi_model` object stores the model equations (ODE right-hand side), state variables, parameters, parameter bounds, default values, and optional helpers for building initial conditions and interpreting model output.

Usage

```
new_epi_model(
  name,
  rhs,
  state_names,
  par_names,
  lower = NULL,
  upper = NULL,
  defaults = NULL,
  make_init = NULL,
  output = list(incidence_col = "incidence", cumulative_col = "C")
)
```

Arguments

`name`	Character scalar. Human-readable name of the model (e.g. `"SIR"`, `"SIRS"`).
`rhs`	Function defining the right-hand side of the ODE system. Must have signature `function(time, state, parms)` and be compatible with `deSolve::ode()`.
`state_names`	Character vector giving the names of the state variables required by the model.
`par_names`	Character vector giving the names of the model parameters.
`lower`	Optional named numeric vector of lower bounds for parameters.
`upper`	Optional named numeric vector of upper bounds for parameters.
`defaults`	Optional named numeric vector of default parameter values.
`make_init`	Optional function to construct the initial state vector. Typically takes arguments such as population size and initial conditions (e.g. `N`, `I0`, `R0`) and returns a named numeric vector matching `state_names`.
`output`	Optional list defining output conventions. By default, `list(incidence_col = "incidence", cumulative_col = "C")`.

Details

The `epi_model` class provides a lightweight abstraction layer separating model definition from simulation, inference, and visualization. Generic tools such as `simulate_epi` operate on this object without requiring model-specific logic.

Required components:

At minimum, a model must define:

- a name (`name`);
- a right-hand side ODE function compatible with `deSolve::ode()` (`rhs`);
- the names of the state variables (`state_names`);
- the names of the model parameters (`par_names`).

Optional components:

Additional components may be provided to facilitate simulation and fitting:

- lower and upper bounds for parameters (`lower`, `upper`);
- default parameter values (`defaults`);
- a function to construct initial conditions (`make_init`);
- conventions for output columns such as incidence and cumulative counts (`output`).

Parameter bounds and defaults, when supplied, must be named numeric vectors whose names match `par_names`.

Value

An object of class `"epi_model"`, implemented as a named list.

See Also

`simulate_epi`

Examples

```
## Minimal example (structure only)
dummy_rhs <- function(time, state, parms) {
  list(rep(0, length(state)))
}

model <- new_epi_model(
  name = "DUMMY",
  rhs = dummy_rhs,
  state_names = c("X"),
  par_names = c("alpha")
)

model
```

`plot.sim_epi` *Plot a simulated epidemic*

Description

Plot method for objects of class `"sim_epi"` as returned by `simulate_epi`. The function can display either the compartmental state trajectories (S, I, R) or the observed incidence time series, depending on the value of the `what` argument.

Usage

```
## S3 method for class 'sim_epi'
plot(x, what = c("states", "incidence"), ...)
```

Arguments

`x`	Object of class `"sim_epi"` as returned by `simulate_epi`.
`what`	Character. Type of plot to produce. Either `"states"` for the compartmental trajectories (S, I, R) or `"incidence"` for the observed incidence time series.
`...`	Additional graphical parameters passed to base plotting functions such as `plot` and `lines`.

Details

When `what = "states"`, the function plots the time evolution of the susceptible (S), infectious (I), and recovered (R) compartments on the same figure. When `what = "incidence"`, the function plots the observed incidence counts generated by the observation model.

This method is automatically dispatched when calling `plot()` on an object of class `"sim_epi"`.

Value

Invisibly returns the input object `x`.

See Also

`simulate_epi`, `summary.epi_sim`

Examples

```
sim <- simulate_epi(n_days = 200, model = "sir", seed = 1)

# Plot S, I, R trajectories
plot(sim)

# Plot observed incidence
plot(sim, what = "incidence")
```

predict.fit_epi_model *Predict epidemic dynamics from a fitted epidemic model*

Description

Computes forward predictions from a fitted epidemic model by solving the underlying ODE system using the estimated parameters.

The prediction is deterministic and represents the expected epidemic trajectory conditional on the fitted parameters.

Usage

```
## S3 method for class 'fit_epi_model'
predict(
  object,
  n_days,
  init = NULL,
  init_args = NULL,
  times = NULL,
  type = c("both", "states", "incidence"),
  method = "lsoda",
  ...
)
```

Arguments

`object`	An object of class "`fit_epi_model`".
`n_days`	Integer. Number of days to predict ahead.
`init`	Named numeric vector of initial states (e.g. `S`, `I`, `R`, `C`). If `NULL`, initial conditions are constructed using `init_args`.
`init_args`	Named list passed to `model$make_init()` (e.g. `list(N = 1e6, I0 = 10, R0 = 0)`).
`times`	Optional numeric vector of time points. Overrides `n_days`.
`type`	Character. One of "`states`", "`incidence`", or "`both`".
`method`	Integration method passed to `deSolve::ode()`.
`...`	Currently unused.

Value

A list containing predicted epidemic quantities:

states Predicted compartment trajectories.

incidence Predicted incidence time series.

Examples

```
## Not run:
sim <- simulate_epi(
  model = SIRS_MODEL,
  n_days = 200,
  parms = c(beta = 0.30, gamma = 0.10, omega = 0.02),
  init_args = list(N = 1e6, I0 = 20, R0 = 0),
  obs = "poisson"
)
x <- sim$incidence_obs
plot(x, type = "l", xlab = "Day", ylab = "Incidence")
fit <- fit_epi_model(x, model = SIRS_MODEL, init = list(I = 6, N = 1e6))
ini0 <- fit$ini0
out <- deSolve::ode(
  y = ini0,
  times = 1:fit$x_len,
  func = SIR_MODEL$rhs,
  parms = fit$par,
  method = "lsoda"
)
fin <- tail(out, n = 1)
iniF <- c(S = fin[2], I = fin[3], R = fin[4], C = fin[5])
pred <- predict(
  object = fit,
  n_days = 300,
  init = iniF,
  type = "both"
)

plot(pred$states$time, pred$states$I, type = "l", ylim = c(0, max(c(pred$states$I, pred$states$IR, pred$states$S))
lines(pred$states$R, col = "red", lty = 2)
lines(pred$states$S, col = "blue", lty = 2)

## End(Not run)
```

`print.epi_model` *Print method for epidemic model objects*

Description

Provides a concise, human-readable summary of an `epi_model` object, including its name, state variables, parameters, and (when available) parameter bounds.

This method is automatically called when an `epi_model` object is printed at the console.

Usage

```
## S3 method for class 'epi_model'
print(x, ...)
```

Arguments

`x`	An object of class `"epi_model"`.
`...`	Further arguments (ignored).

Value

Invisibly returns the input object `x`.

Examples

```
SIR_MODEL
```

`print.sim_epi` *Print a simulated epidemic*

Description

Print method for objects of class `"sim_epi"` as returned by `simulate_epi`. The function provides a concise, human-readable summary of the simulated epidemic, including the model type, time horizon, key parameters, and basic outcome metrics.

Usage

```
## S3 method for class 'sim_epi'
print(x, ...)
```

Arguments

`x`	Object of class `"sim_epi"` as returned by `simulate_epi`.
`...`	Currently unused; included for compatibility with generic dispatch.

Details

This method is automatically called when an object of class `"sim_epi"` is printed. It is intended to give a quick overview without displaying the full internal structure of the object.

Value

Invisibly returns the input object `x`.

See Also

`simulate_epi`, `summary.sim_epi`, `plot.sim_epi`

Examples

```
sim <- simulate_epi(n_days = 200, model = "sirs",
                    beta = 0.30, gamma = 0.10, omega = 1/180)

sim
```

`simulate_epi` *Simulate an epidemic model defined by an* `epi_model` *object*

Description

Simulates a deterministic compartmental epidemic model (e.g. SIR, SIRS, or any user-defined extension) specified via an `epi_model` object. The model is solved as an ODE system using `deSolve::ode()`, and can optionally include an observation process to generate reported incidence counts.

The function is model-agnostic: all model-specific information (state variables, parameters, ODE right-hand side, bounds, and output conventions) is read from the supplied `epi_model`.

Usage

```
simulate_epi(
  model,
  n_days = 200,
  parms = NULL,
  init = NULL,
  init_args = list(N = 1e+06, I0 = 10, R0 = 0),
  times = NULL,
  rho = 1,
  obs = c("negbin", "poisson", "none"),
  size = 20,
  seed = NULL,
  method = "lsoda"
)
```

Arguments

`model`	An object of class `"epi_model"` defining the epidemic model to simulate (e.g. SIR, SIRS).
`n_days`	Integer. Number of days to simulate. Ignored if `times` is provided.
`parms`	Named numeric vector of model parameters. Must match `model$par_names`. Any missing parameters are taken from `model$defaults`, if available.
`init`	Named numeric vector with initial state values. If `NULL` (default), the function uses `model$make_init` together with `init_args`.
`init_args`	Named list of arguments passed to `model$make_init` to construct the initial state (e.g. `N`, `I0`, `R0`).

`times`	Optional numeric vector of time points at which to solve the ODE system. If supplied, it overrides `n_days`.
`rho`	Numeric in $[0, 1]$. Reporting fraction mapping true incidence to expected observed incidence.
`obs`	Character string specifying the observation model. One of `"poisson"`, `"negbin"`, or `"none"`.
`size`	Numeric. Dispersion (size) parameter for the negative binomial observation model. Larger values imply less overdispersion.
`seed`	Optional integer. If provided, sets the random seed for reproducible observation draws.
`method`	Character string. Integration method passed to `deSolve::ode()` (default: `"lsoda"`).

Details

Model structure:

The `model` argument must be an object of class `"epi_model"`, typically created with `new_epi_model()`. At minimum, the model must define:

- a right-hand side ODE function (`model$rhs`);
- the required state variables (`model$state_names`);
- the model parameters (`model$par_names`).

Optionally, the model may also define:

- default parameter values (`model$defaults`);
- parameter bounds (`model$lower`, `model$upper`);
- a helper function to construct initial conditions (`model$make_init`);
- standard output column names for incidence and cumulative infections (`model$output`).

Time grid:

By default, the model is simulated on a daily grid from day 0 to `n_days`. Alternatively, a custom numeric vector of times can be supplied via `times`.

Observation model:

The latent incidence produced by the ODE model can be converted into reported incidence counts using a simple observation model:

`obs = "poisson"` Reported counts are drawn from a Poisson distribution with mean $\mu(t) = \rho\lambda(t)$.

`obs = "negbin"` Reported counts are drawn from a negative binomial distribution with mean $\mu(t) = \rho\lambda(t)$ and dispersion parameter `size`.

`obs = "none"` No observation process is applied; observed incidence and cumulative counts are returned as `NA`.

Value

A named list with the following components:

model Name of the epidemic model.

params List of model parameters used in the simulation.

states Data frame with columns `time` and the model state variables (e.g. `S`, `I`, `R`, `C`).

incidence_true Data frame with columns `time` and `inc` containing the latent model incidence.

incidence_obs Data frame with columns `time` and `inc` containing the observed (reported) incidence counts.

cumulative_obs Data frame with columns `time` and `cases_cum` containing cumulative observed cases.

See Also

`new_epi_model`, `deSolve::ode`

Examples

```
## SIR simulation
sim <- simulate_epi(
  model = SIR_MODEL,
  n_days = 200,
  parms = c(beta = 0.30, gamma = 0.10),
  init_args = list(N = 1e6, I0 = 20, R0 = 0),
  rho = 0.3,
  obs = "poisson",
  seed = 1
)

plot(sim)

## SIRS simulation
sim2 <- simulate_epi(
  model = SIRS_MODEL,
  n_days = 200,
  parms = c(beta = 0.30, gamma = 0.10, omega = 0.02),
  init_args = list(N = 1e6, I0 = 20, R0 = 0),
  obs = "poisson"
)
plot(sim2)
```

`SIRS_MODEL` *SIRS epidemic model with waning immunity*

Description

An `epi_model` object representing a deterministic **SIRS** (Susceptible–Infectious–Recovered–Susceptible) compartmental epidemic model with waning immunity. Individuals who recover from infection lose immunity at rate `omega` and return to the susceptible compartment.

The model is extended with an auxiliary state variable `C(t)` that tracks the cumulative number of infections over time, and it provides the instantaneous incidence as an additional model output.

Usage

```
SIRS_MODEL
```

Format

An object of class `"epi_model"`.

Details

State variables:

The model is defined in terms of the following state variables:

S(t) Number of susceptible individuals at time t.

I(t) Number of infectious (actively infected) individuals at time t.

R(t) Number of recovered (temporarily immune) individuals at time t.

C(t) Cumulative number of infections up to time t.

The total population size is given by

$$N = S(t) + I(t) + R(t),$$

which is conserved by the model dynamics.

Parameters:

The SIRS model depends on the following parameters:

beta Transmission rate (per day).

gamma Recovery/removal rate (per day).

omega Rate of waning immunity from `R` back to `S` (per day).

Model equations:

New infections occur at rate

$$\lambda(t) = \beta \frac{S(t)\, I(t)}{N}.$$

The system of ordinary differential equations is:

$$\frac{dS}{dt} = -\lambda(t) + \omega R(t),$$
$$\frac{dI}{dt} = \lambda(t) - \gamma I(t),$$
$$\frac{dR}{dt} = \gamma I(t) - \omega R(t),$$
$$\frac{dC}{dt} = \lambda(t).$$

The auxiliary state $C(t)$ provides a continuous-time analogue of cumulative incidence and is useful for linking the latent epidemic dynamics to discrete-time observation models.

Usage:

This predefined model object is intended to be used with generic utilities such as `simulate_epi` and model-fitting functions that operate on `epi_model` objects.

See Also

simulate_epi, new_epi_model

Examples

```
## Simulate a SIRS epidemic without an observation model
sim <- simulate_epi(
  model = SIRS_MODEL,
  n_days = 200,
  parms = c(beta = 0.3, gamma = 0.1, omega = 0.02),
  init_args = list(N = 1e6, I0 = 20, R0 = 0),
  obs = "none"
)

plot(sim$states$time, sim$states$I, type = "l",
     xlab = "Days", ylab = "I(t)",
     main = "SIRS model: infectious individuals")
```

`SIR_MODEL` *SIR epidemic model with cumulative infections*

Description

An `epi_model` object representing a deterministic **SIR** (Susceptible–Infectious–Recovered) compartmental epidemic model. The model is extended with an auxiliary state variable `C(t)` that tracks the cumulative number of infections over time, and it provides the instantaneous incidence as an additional model output.

Usage

```
SIR_MODEL
```

Format

An object of class `"epi_model"`.

Details

State variables:

The model is defined in terms of the following state variables:

S(t) Number of susceptible individuals at time t.

I(t) Number of infectious (actively infected) individuals at time t.

R(t) Number of removed/recovered individuals at time t.

C(t) Cumulative number of infections up to time t.

The total population size is given by

$$N = S(t) + I(t) + R(t),$$

which is conserved by the model dynamics.

Parameters:
The SIR model depends on the following parameters:

beta Transmission rate (per day).
gamma Recovery/removal rate (per day).

Model equations:
New infections occur at rate

$$\lambda(t) = \beta \frac{S(t)\, I(t)}{N}.$$

The system of ordinary differential equations is:

$$\begin{aligned}\frac{dS}{dt} &= -\lambda(t),\\ \frac{dI}{dt} &= \lambda(t) - \gamma I(t),\\ \frac{dR}{dt} &= \gamma I(t),\\ \frac{dC}{dt} &= \lambda(t).\end{aligned}$$

The auxiliary state $C(t)$ provides a continuous-time analogue of cumulative incidence and is useful for linking the latent epidemic dynamics to discrete-time observation models.

Usage:
This predefined model object is intended to be used with generic utilities such as `simulate_epi` and model-fitting functions that operate on `epi_model` objects.

See Also

`simulate_epi`, `new_epi_model`

Examples

```
## Simulate a SIR epidemic without an observation model
sim <- simulate_epi(
  model = SIR_MODEL,
  n_days = 200,
  parms = c(beta = 0.3, gamma = 0.1),
  init_args = list(N = 1e6, I0 = 20, R0 = 0),
  obs = "none"
)

plot(sim$states$time, sim$states$I, type = "l",
     xlab = "Days", ylab = "I(t)",
     main = "SIR model: infectious individuals")
```

summary.epi_sim *Summarize a simulated epidemic*

Description

Summary method for objects of class "sim_epi" as returned by simulate_epi. The function computes and returns a small set of epidemiologically meaningful summary quantities derived from the simulation.

Usage

```
## S3 method for class 'epi_sim'
summary(object, ...)
```

Arguments

object	Object of class "sim_epi" as returned by simulate_epi.
...	Currently unused; included for compatibility with generic summary.

Details

The summary includes:

model The epidemic model simulated (e.g. "sir" or "sirs").

R0 The basic reproduction number, computed as $R_0 = \beta/\gamma$.

peak_I The maximum number of infectious individuals observed during the simulation.

total_infections The total number of infection events, computed as the maximum value of the cumulative infection variable C(t).

This method is automatically dispatched when calling summary() on an object of class "sim_epi".

Value

A named list with summary statistics describing the simulated epidemic.

See Also

simulate_epi, plot.sim_epi

Examples

```
sim <- simulate_epi(n_days = 300, model = "sirs", omega = 1/180, seed = 1)

summary(sim)
```

Index