# Package 'EpiNova'

April 22, 2026

**Title** Flexible Extended State-Space Epidemiological Models with Modern Inference

**Version** 0.1.0

**Description** An extended epidemiological modelling framework that goes beyond the classical SIR (Susceptible-Infectious-Recovered) model. Supports SEIR (Susceptible-Exposed-Infectious-Recovered), SEIRD (Susceptible-Exposed-Infectious-Recovered-Deceased), SVEIRD (Susceptible-Vaccinated-Exposed-Infectious-Recovered-Deceased), and age-stratified compartmental models with flexible intervention functions (spline-based, Gaussian process, or user-defined). Inference is available via maximum likelihood or sequential Monte Carlo (SMC, also known as particle filtering) with no external binary dependencies. Includes a dependency-free real-time effective reproduction number (Rt) estimator, spatial multi-patch models with gravity-model mobility, ensemble forecasting via Bayesian model averaging (BMA), and proper scoring rules including CRPS (Continuous Ranked Probability Score), coverage, and MAE (Mean Absolute Error) for forecast evaluation. Methods follow Anderson and May (1991, ISBN:9780198545996), Doucet, de Freitas, and Gordon (2001) <doi:10.1007/978-1-4757-3437-9>, Cori et al. (2013) <doi:10.1093/aje/kwt133>, and Gneiting and Raftery (2007) <doi:10.1198/016214506000001437>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Depends** R (>= 4.0.0)

**Imports** deSolve, ggplot2, dplyr, tidyr, splines, scales

**Suggests** DEoptim, MASS, numDeriv, rstan, TMB, EpiEstim, knitr, rmarkdown, testthat (>= 3.0.0), ggpubr, patchwork

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**URL** `https://github.com/causalfragility-lab/EpiNova`

**BugReports** `https://github.com/causalfragility-lab/EpiNova/issues`

**NeedsCompilation** no

**Author** Subir Hait [aut, cre] (ORCID: <`https://orcid.org/0009-0004-9871-9677`>)

**Maintainer** Subir Hait <`haitsubi@msu.edu`>

**Repository** CRAN

**Date/Publication** 2026-04-22 08:20:26 UTC

# Contents

---

## `build_multipatch_SEIR` *Build a mobility-coupled multi-patch SEIR ODE system*

---

### Description

Build a mobility-coupled multi-patch SEIR ODE system

### Usage

```
build_multipatch_SEIR(
  n_patches,
  M,
  beta_vec,
  gamma_vec,
  sigma_vec,
  pi_fn_list = NULL,
  N_vec = rep(1, n_patches)
)
```

### Arguments

| | |
|---|---|
| `n_patches` | Integer number of patches. |
| `M` | n_patches x n_patches mobility matrix. `M[i,j]` is the daily fraction of population in patch i that travels to patch j (rows sum to <= 1). |
| `beta_vec` | Length-n_patches vector of transmission rates. |
| `gamma_vec` | Length-n_patches vector of recovery rates. |
| `sigma_vec` | Length-n_patches vector of incubation rates. |
| `pi_fn_list` | List of n_patches pi(t) functions (one per patch). Use `NULL` for no intervention in that patch. |
| `N_vec` | Population sizes for each patch. |

### Value

A function suitable for `deSolve::ode`.

---

## `build_phi_pulse` *Build a smooth quarantine pulse phi(t)*

---

### Description

Approximates a Dirac delta at each change point with a narrow Gaussian pulse. The area under each pulse equals the quarantine fraction `phi_values[i]`.

### Usage

```
build_phi_pulse(change_times, phi_values, bandwidth = 0.5)
```

### Arguments

| | |
|---|---|
| `change_times` | Numeric vector of quarantine event days. |
| `phi_values` | Quarantine fractions in (0, 1). |
| `bandwidth` | Width (SD) of each Gaussian pulse (default 0.5 days). |

### Value

A function `phi_fn(t)`.

---

## `build_pi_exp` *Build an exponential decay pi(t) = exp(-lambda t)*

### Description

Build an exponential decay pi(t) = exp(-lambda t)

### Usage

```
build_pi_exp(lambda, t0 = 0)
```

### Arguments

| | |
|---|---|
| `lambda` | Decay rate (positive scalar). |
| `t0` | Start of decay (default 0). |

### Value

A function `pi_fn(t)`.

---

## `build_pi_spline` *Build a natural cubic spline pi(t)*

---

### Description

Build a natural cubic spline pi(t)

### Usage

```
build_pi_spline(knot_times, knot_values, extrapolate = c("flat", "linear"))
```

### Arguments

| | |
|---|---|
| `knot_times` | Numeric vector of days (knot positions). |
| `knot_values` | Numeric vector of pi values at each knot, all in $[0, 1]$. |
| `extrapolate` | How to handle t outside knot range: `"flat"` (default) or `"linear"`. |

### Value

A function `pi_fn(t)`.

### Examples

```
pi_spline <- build_pi_spline(
  knot_times  = c(0, 10, 20, 40, 100),
  knot_values = c(1, 0.8, 0.5, 0.3, 0.3)
)
curve(pi_spline(x), 0, 120, ylab = expression(pi(t)))
```

---

## `build_pi_step` *Build a step-function pi(t) (reproduces eSIR Model 1 behaviour)*

---

### Description

Build a step-function pi(t) (reproduces eSIR Model 1 behaviour)

### Usage

```
build_pi_step(change_times, pi_values)
```

### Arguments

| | |
|---|---|
| `change_times` | Numeric vector of change-point days. |
| `pi_values` | Numeric vector of length `length(change_times) + 1`. |

### Value

A function `pi_fn(t)`.

---

## `compose_pi` *Compose multiple pi(t) functions multiplicatively*

---

### Description

Each component function represents an independent non-pharmaceutical intervention (NPI). Their effects are assumed multiplicative on transmission.

### Usage

```
compose_pi(...)
```

### Arguments

| | |
|---|---|
| `...` | Any number of functions `pi_fn(t)`, each in $[0, 1]$. |

### Value

A composite function `pi_fn(t)` in $[0, 1]$.

### Examples

```
lockdown  <- build_pi_step(c(10, 60), c(1, 0.4, 0.7))
masks     <- build_pi_spline(c(0, 10, 20, 30), c(1, 0.92, 0.85, 0.75))
combined  <- compose_pi(lockdown, masks)
```

---

## `ensemble_forecast` *Ensemble forecast via Bayesian Model Averaging (BMA)*

---

### Description

Fits multiple compartmental models and combines their forecasts weighted by their marginal likelihoods (approximated by AIC weights).

### Usage

```
ensemble_forecast(
  obs_Y,
  obs_R,
  N,
  models = c("SEIR", "SEIRD", "SVEIRD"),
  par_inits,
  par_bounds,
  pi_fn = NULL,
  T_forecast = 60L,
  n_sim = 500L
)
```

### Arguments

| | |
|---|---|
| `obs_Y` | Numeric vector of observed infected proportions. |
| `obs_R` | Numeric vector of observed removed proportions. |
| `N` | Population size. |
| `models` | Character vector of model types to average over. |
| `par_inits` | Named list of initial parameter lists (one per model). |
| `par_bounds` | List with elements `lower` and `upper`, each a named list of bounds applying to all models. |
| `pi_fn` | Common transmission modifier (or `NULL`). |
| `T_forecast` | Integer number of days to forecast ahead. |
| `n_sim` | Number of simulation draws per model. |

### Value

A list with `ensemble_forecast` (data frame), `model_weights`, and `individual_fits`.

---

## `EpiNova-ensemble` *Ensemble Forecasting and Uncertainty Quantification*

### Description

eSIR produced credible intervals only for its fixed model class. EpiNova provides:

- Model-averaged forecasts (Bayesian model averaging over different compartmental structures).
- Scenario-based projection (what-if analysis).
- Proper scoring rules to evaluate forecast calibration.

---

## `EpiNova-inference` *Inference Engines for EpiNova*

### Description

eSIR relied exclusively on MCMC via JAGS, which requires an external binary install and can be slow for large models. EpiNova offers three inference backends that the user can switch between with a single argument:

1. **MLE** — Maximum likelihood via **TMB** (Template Model Builder). Automatic differentiation gives exact gradients, enabling fast optimisation and Laplace-approximation uncertainty. No external binary needed.
2. **HMC** — Full Bayesian posterior via **cmdstanr / rstan**. Hamiltonian Monte Carlo mixes far faster than Gibbs/Metropolis for continuous parameters.
3. **SMC** — Sequential Monte Carlo (particle filter). Ideal for real-time updating as new daily data arrive. Implemented in pure R so no extra dependencies are required.

---

## `EpiNova-interventions` *Flexible Intervention (pi) Functions*

---

### Description

eSIR restricted pi(t) to step functions or a single decaying exponential. EpiNova provides a richer library of transmission modifiers and allows users to compose them or supply a completely custom function.

All builders return a plain R function `pi_fn(t)` in $[0, 1]$.

---

## `EpiNova-multipatch` *Multi-patch Spatial SIR Models*

---

### Description

eSIR was a single-population model. EpiNova adds a network of patches (cities, regions, countries) connected by a mobility / commuting matrix. Each patch has its own compartmental dynamics; patches exchange infectious individuals according to a movement kernel.

This is particularly useful for modelling cross-regional seeding, airport-based importations, and the effect of travel restrictions.

---

## `EpiNova-ode` *EpiNova: Flexible Epidemiological Compartmental Models*

---

### Description

Core ODE system for multiple compartmental model types

---

## `EpiNova-plotting` *Visualisation Layer for EpiNova*

---

### Description

Publication-ready ggplot2 graphics. All plot functions return ggplot2 objects that can be further customised.

## `estimate_Rt` *Estimate time-varying effective reproduction number Rt*

### Description

Wraps **EpiEstim** with automatic serial interval specification. Returns a tidy data frame with posterior mean and 95 % CrI.

### Usage

```
estimate_Rt(incidence, mean_si = 5.2, sd_si = 2.8, window = 7L)
```

### Arguments

| | |
|---|---|
| `incidence` | Integer vector of daily new case counts. |
| `mean_si` | Mean of the serial interval distribution (days). |
| `sd_si` | SD of the serial interval distribution (days). |
| `window` | Sliding window for Rt estimation (default 7 days). |

### Value

Data frame with columns `t_end`, `Rt_mean`, `Rt_lower`, `Rt_upper`.

## `estimate_Rt_simple` *Lightweight built-in Rt estimator (no extra packages needed)*

### Description

Estimates the effective reproduction number using a sliding-window case ratio approach weighted by a discretised serial interval distribution (gamma). Uncertainty bands are obtained from 500 Poisson bootstrap replicates. No external packages are required.

### Usage

```
estimate_Rt_simple(
  incidence,
  mean_si = 5.2,
  sd_si = 2.8,
  window = 7L,
  n_boot = 500L
)
```

### Arguments

| | |
|---|---|
| `incidence` | Integer vector of daily new case counts. |
| `mean_si` | Mean serial interval in days (default 5.2). |
| `sd_si` | SD of serial interval in days (default 2.8). |
| `window` | Sliding window width in days (default 7). |
| `n_boot` | Number of bootstrap replicates for CIs (default 500). |

### Value

Data frame with columns `t_end`, `Rt_mean`, `Rt_lower`, `Rt_upper`.

### Examples

```
incidence <- c(1,1,2,4,6,8,13,21,18,14,10,7,5,3,2,1)
Rt_df <- estimate_Rt_simple(incidence, n_boot = 50L)
```

---

## `fit_mle` *Fit a EpiNova model by maximum likelihood*

### Description

Minimises the negative log-likelihood using a two-stage approach: (1) global search via differential evolution (`DEoptim`), (2) local refinement via `nlminb` with analytical gradients. Profile-likelihood confidence intervals are returned for all parameters.

### Usage

```
fit_mle(
  obs_Y,
  obs_R,
  N,
  model = "SEIRD",
  par_init,
  par_lower,
  par_upper,
  pi_fn = NULL,
  obs_model = c("betabin", "normal"),
  ...
)
```

### Arguments

| | |
|---|---|
| `obs_Y` | Numeric vector of observed infected proportions. |
| `obs_R` | Numeric vector of observed removed proportions. |
| `N` | Total population size. |

| | |
|---|---|
| `model` | Compartmental model type (see `solve_model`). |
| `par_init` | Named list of starting parameter values. |
| `par_lower` | Named list of lower bounds. |
| `par_upper` | Named list of upper bounds. |
| `pi_fn` | Transmission modifier function (see `build_pi_spline`, `build_pi_step`, etc.). |
| `obs_model` | Observation model: `"betabin"` (beta-binomial, recommended) or `"normal"`. |
| `...` | Additional arguments passed to `solve_model`. |

### Value

An object of class `"EpiNova_mle"` containing parameter estimates, standard errors, 95 % CIs, AIC, BIC, and the fitted trajectory.

---

## `fit_smc` *Sequential Monte Carlo (particle filter) inference*

### Description

Fits the model online, updating the parameter distribution each day as new observations arrive. Suitable for real-time outbreak monitoring dashboards.

### Usage

```
fit_smc(
  obs_Y,
  obs_R,
  N,
  model = "SEIR",
  prior_fn,
  n_particles = 2000L,
  pi_fn = NULL,
  resample_ess_thresh = 0.5
)
```

### Arguments

| | |
|---|---|
| `obs_Y` | Numeric vector of observed infected proportions. |
| `obs_R` | Numeric vector of observed removed proportions. |
| `N` | Population size. |
| `model` | Compartmental model type. |
| `prior_fn` | Function returning a named list of parameters drawn from the prior (one call = one particle). |
| `n_particles` | Integer number of particles (default 2000). |
| `pi_fn` | Transmission modifier function. |
| `resample_ess_thresh` | Resample when ESS < this fraction of n_particles (default 0.5). |

### Value

A list with elements `particles` (final weighted sample), `weights`, `log_evidence`, `Rt_trajectory` (median and 95 % CI of effective reproduction number over time).

---

## `gp_cov_sqexp` *Build a squared-exponential covariance matrix for GP pi(t)*

### Description

Used to construct the GP prior over transmission modifiers at a discrete set of time points before passing to the Stan sampler.

### Usage

```
gp_cov_sqexp(times, l = 14, sigma = 0.3)
```

### Arguments

| | |
|---|---|
| `times` | Numeric vector of time points. |
| `l` | Length-scale (controls smoothness). |
| `sigma` | Marginal standard deviation of the GP. |

### Value

A symmetric positive-definite matrix of dimension `length(times)` x `length(times)`.

---

## `gravity_mobility` *Build a gravity-model mobility matrix*

### Description

Uses the functional form

$$M_{ij} = \kappa \frac{N_i^\alpha N_j^\beta}{d_{ij}^\gamma}$$

normalised so that each row sums to at most `max_travel`.

### Usage

```
gravity_mobility(
  N_vec,
  dist_mat,
  kappa = 1e-05,
  alpha = 1,
  beta = 1,
  grav_gamma = 2,
  max_travel = 0.1
)
```

### Arguments

| | |
|---|---|
| `N_vec` | Length-n population sizes. |
| `dist_mat` | n x n symmetric matrix of pairwise distances. |
| `kappa` | Proportionality constant. |
| `alpha, beta` | Population exponents (default 1). |
| `grav_gamma` | Distance decay exponent (default 2). |
| `max_travel` | Maximum daily travel fraction per patch (0–1). |

### Value

A row-normalised n x n mobility matrix.

---

## `hubei_covid` *Hubei Province COVID-19 data (Jan 13 - Feb 11, 2020)*

### Description

Cumulative daily confirmed and removed (recovered + deceased) COVID-19 case counts for Hubei Province, China, covering 30 days from 13 January 2020 to 11 February 2020. Same dataset used in the original eSIR paper.

### Usage

```
hubei_covid
```

### Format

A named list with six elements:

**NI** Integer vector (length 30). Cumulative confirmed cases.

**RI** Integer vector (length 30). Cumulative removed cases (recovered + deceased).

**N** Numeric scalar. Hubei population size (58,500,000).

**begin_date** Character. First observation date ("2020-01-13").

**end_date** Character. Last observation date ("2020-02-11").

**description** Character. Plain-text description of the dataset.

### Source

DXY.cn daily COVID-19 situation reports.

### Examples

```
Y <- hubei_covid$NI / hubei_covid$N - hubei_covid$RI / hubei_covid$N
R <- hubei_covid$RI / hubei_covid$N
plot(Y, type = "l", ylab = "Infected proportion", xlab = "Day")
```

---

## `plot_forecast` *Plot forecast with uncertainty ribbon*

---

### Description

Plot forecast with uncertainty ribbon

### Usage

```
plot_forecast(
  forecast_df,
  obs_Y = NULL,
  T_obs_end = NULL,
  title = "Infection Forecast"
)
```

### Arguments

| | |
|---|---|
| `forecast_df` | Data frame with `time`, `I_median`, `I_lower`, `I_upper`. |
| `obs_Y` | Observed infected proportions. |
| `T_obs_end` | Last observed day. |
| `title` | Plot title. |

### Value

A ggplot2 object.

---

## `plot_multipatch_snapshot` *Multi-patch bar chart of infected proportion by patch*

---

### Description

Multi-patch bar chart of infected proportion by patch

### Usage

```
plot_multipatch_snapshot(multipatch_df, t_snapshot, patch_names = NULL)
```

### Arguments

| | |
|---|---|
| `multipatch_df` | Data frame from `solve_multipatch`. |
| `t_snapshot` | Day to visualise. |
| `patch_names` | Character vector of patch names. |

## Value

A ggplot2 bar chart.

---

## `plot_Rt` *Plot effective reproduction number Rt over time*

---

### Description

Plot effective reproduction number Rt over time

### Usage

```
plot_Rt(Rt_df, change_times = NULL)
```

### Arguments

| | |
|---|---|
| `Rt_df` | Data frame with `t_end`, `Rt_mean`, `Rt_lower`, `Rt_upper`. |
| `change_times` | Optional numeric vector of intervention change points to mark as vertical lines. |

### Value

A ggplot2 object.

---

## `plot_scenarios` *Plot scenario comparison*

---

### Description

Plot scenario comparison

### Usage

```
plot_scenarios(scenario_df, obs_Y = NULL)
```

### Arguments

| | |
|---|---|
| `scenario_df` | Data frame from `project_scenarios` or a manually constructed data frame with columns `time`, `I_median`, `I_lower`, `I_upper`, `scenario`. |
| `obs_Y` | Optional observed data to overlay. |

### Value

A ggplot2 object.

---

## `plot_trajectory` *Plot model trajectory with observed data*

### Description

Plot model trajectory with observed data

### Usage

```
plot_trajectory(
  traj_df,
  obs_Y = NULL,
  obs_R = NULL,
  T_obs_end = NULL,
  title = "Epidemic Trajectory"
)
```

### Arguments

| | |
|---|---|
| `traj_df` | Data frame from `solve_model`. |
| `obs_Y` | Observed infected proportions (optional). |
| `obs_R` | Observed removed proportions (optional). |
| `T_obs_end` | Last observed day (draws a vertical cutoff line). |
| `title` | Plot title. |

### Value

A ggplot2 object.

---

## `prep_proportions` *Prepare population proportions from a hubei_covid-style list*

### Description

Converts raw cumulative counts into the proportion vectors `Y` (active infected) and `R` (removed) expected by `solve_model` and the fitting functions.

### Usage

```
prep_proportions(dat)
```

### Arguments

| | |
|---|---|
| `dat` | A list with elements `NI`, `RI`, and `N`. |

### Value

A list with numeric vectors `Y` and `R`.

### Examples

```
props <- prep_proportions(hubei_covid)
str(props)
```

---

## `project_scenarios` *Project scenarios under alternative intervention strategies*

---

### Description

Project scenarios under alternative intervention strategies

### Usage

```
project_scenarios(fit, scenarios, T_forecast = 120L, n_sim = 500L)
```

### Arguments

| | |
|---|---|
| `fit` | A fitted model object (`EpiNova_mle`). |
| `scenarios` | Named list of pi(t) functions, one per scenario. |
| `T_forecast` | Integer forecast horizon in days. |
| `n_sim` | Simulation draws for uncertainty. |

### Value

A tidy data frame with columns `scenario`, `time`, `I_median`, `I_lower`, `I_upper`, `peak_day`, `peak_I`, `total_infected`.

---

## `score_forecast` *Evaluate forecast calibration with proper scoring rules*

---

### Description

Computes the Continuous Ranked Probability Score (CRPS) and interval coverage for each time point.

### Usage

```
score_forecast(forecast_df, actual_Y)
```

### Arguments

| | |
|---|---|
| `forecast_df` | Data frame with columns `I_median`, `I_lower`, `I_upper` (from `project_scenarios` or `ensemble_forecast`). |
| `actual_Y` | Numeric vector of actual observed values. |

### Value

Data frame with `CRPS`, `coverage_50`, `coverage_95`, and `MAE`.

---

## `solve_model` *Solve a compartmental ODE model*

### Description

Dispatches to the appropriate ODE system based on the model type. Supports SIR, SEIR, SEIRD, SVEIR, SVEIRD, and age-stratified variants. All models share a common calling convention so that downstream functions (fitting, forecasting, plotting) are model-agnostic.

### Usage

```
solve_model(
  params,
  init,
  times,
  model = c("SIR", "SEIR", "SEIRD", "SVEIR", "SVEIRD", "age_SEIR"),
  pi_fn = NULL,
  phi_fn = NULL
)
```

### Arguments

| | |
|---|---|
| `params` | Named list of epidemiological parameters. |
| `init` | Named numeric vector of initial compartment proportions (must sum to 1). |
| `times` | Numeric vector of time points (days). |
| `model` | Character string: one of `"SIR"`, `"SEIR"`, `"SEIRD"`, `"SVEIR"`, `"SVEIRD"`, `"age_SEIR"`. |
| `pi_fn` | Optional function `pi_fn(t)` returning a transmission modifier in $[0, 1]$ at time `t`. If `NULL`, the unmodified transmission rate is used. |
| `phi_fn` | Optional function `phi_fn(t)` returning the instantaneous quarantine removal rate at time `t` (Dirac delta approximated by a narrow Gaussian pulse). If `NULL`, no quarantine pulse is applied. |

### Value

A data frame with one row per time point and one column per compartment, plus a column `time`.

## Examples

```
p  <- list(beta = 0.3, gamma = 0.1, sigma = 0.2,
           delta = 0.005, vax_rate = 0.002, wane = 0.01, ve = 0.85)
y0 <- c(S = 0.990, E = 0.005, I = 0.004, R = 0.001, D = 0, V = 0)
t  <- seq(0, 30, by = 1)
result <- solve_model(p, y0, t, model = "SVEIRD")
```

---

# `solve_multipatch` *Solve a multi-patch SEIR model*

## Description

Solve a multi-patch SEIR model

## Usage

```
solve_multipatch(ode_fn, init_mat, times, n_patches)
```

## Arguments

| | |
|---|---|
| `ode_fn` | ODE function from `build_multipatch_SEIR`. |
| `init_mat` | n_patches x 4 matrix of initial conditions (columns: S, E, I, R). |
| `times` | Time points. |
| `n_patches` | Number of patches. |

## Value

A data frame with columns `time` and `S_i`, `E_i`, `I_i`, `R_i` for each patch i.

# Index