Package 'RtForecastR'

August 21, 2026

Type Package

Title Real-Time Effective Reproduction Number Estimation and Forecasting

Version 0.1.1

Description Filtered (real-time/causal) and smoothed (retrospective) estimation of the time-varying effective reproduction number (Rt) from case-count time series, using the EpiFilter algorithm of Parag (2021) <doi:10.1371/journal.pcbi.1009347>, together with a one-step-ahead in-sample prediction check, a genuine out-of-sample one-step forecast with predictive intervals, elimination probability P(Rt < 1), and forecast calibration metrics (mean absolute error, mean squared error, root mean squared error, empirical coverage, and the weighted interval score of Bracher et al. (2021) <doi:10.1371/journal.pcbi.1008618>). Disease-agnostic: works for any pathogen given a known generation interval.

Language en-US

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.5)

Imports graphics, grDevices, stats, utils

Suggests testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

RoxygenNote 8.0.0

VignetteBuilder knitr

URL https://github.com/rajsubediresearch/RtForecastR

BugReports https://github.com/rajsubediresearch/RtForecastR/issues

NeedsCompilation no

Author Raj Subedi [aut, cre, cph] (Copyright holder for all files except epiFilter.R, epiSmoother.R, and the original recursPredict.R logic (see Kris V. Parag entry); author of R/recursPredict.R's configurable-grid maxI extension and R/recursPredictQuantiles.R), Kris V. Parag [ctb, cph] (Author/copyright holder of the original EpiFilter algorithm (epiFilter, epiSmoother, recursPredict); files R/epiFilter.R, R/epiSmoother.R and R/recursPredict.R are unmodified or lightly modified ports of that work, released under GPL-3)

Maintainer Raj Subedi <rajsubediresearch@gmail.com>

Repository CRAN

Date/Publication 2026-08-21 13:10:37 UTC

Contents

`compute_lambda` *Total infectiousness (renewal-equation convolution)*

Description

Computes the "total infectiousness" Lambda_t: the convolution of past incidence with a discretized Gamma(mean, variance) generation interval distribution, as required by `epi_filter()` and `epi_smoother()` alongside raw incidence. Written to avoid a dependency on the EpiEstim package's `overall_infectivity()` for this single step.

Usage

```
compute_lambda(incidence, mean_GI, var_GI, max_si = NULL)
```

Arguments

`incidence`	numeric vector of case counts, length n
`mean_GI`	mean generation interval, in the same time units as `incidence` (e.g. weeks if `incidence` is weekly)
`var_GI`	variance of the generation interval, same time units
`max_si`	maximum serial interval to consider (defaults to `length(incidence) - 1`)

Details

The Gamma(mean, variance) discretization approach follows the convention used in `get_Rt.m`, part of the GrowthPredict toolbox: Chowell G, Bleichrodt A, Dahal S, Tariq A, Roosa K, Hyman JM, Luo R. (2024) "GrowthPredict: A toolbox and tutorial-based primer for fitting and forecasting growth trajectories using phenomenological growth models." Scientific Reports 14, 1630. doi:10.1038/s41598024518528

Value

numeric vector of length n; `Lambda[1]` is `NA` (no prior incidence exists to convolve with)

Examples

```
data(measles_cdmx)
Lambda <- compute_lambda(measles_cdmx$cases, mean_GI = 11/7, var_GI = (4/7)^2)
```

`coverage` *Empirical coverage of a prediction interval*

Description

Empirical coverage of a prediction interval

Usage

```
coverage(observed, lo, hi)
```

Arguments

`observed`	numeric vector of realized values
`lo, hi`	numeric vectors giving the lower/upper interval bounds

Value

the fraction of observations falling within `[lo, hi]`

Examples

```
coverage(c(5, 15, 25), c(0, 10, 20), c(10, 20, 30))
```

interval_score *Interval score for a single central prediction interval*

Description

The building block of the weighted interval score (Gneiting & Raftery 2007).

Usage

```
interval_score(observed, lo, hi, level)
```

Arguments

observed	realized value
lo, hi	interval bounds
level	the interval's alpha (e.g. 0.025 for a 95% interval)

Value

a single numeric value (lower is better)

Examples

```
interval_score(observed = 5, lo = 2, hi = 8, level = 0.05)
```

jalisco *Weekly measles cases, Jalisco (larger-outbreak example)*

Description

A second worked example with a larger peak incidence than measles_cdmx, useful for exercising the `maxI` prediction-grid argument of `rt_forecast()`.

Usage

```
jalisco
```

Format

A data frame with columns:

time integer, weekly time index

cases integer, case count for that week

Source

Worked example shipped with RtForecastR; not an official surveillance release.

mae *Mean absolute error*

Description

Mean absolute error

Usage

```
mae(observed, predicted)
```

Arguments

observed	numeric vector of realized values
predicted	numeric vector of point predictions (same length/order)

Value

a single numeric value

Examples

```
mae(c(10, 20, 30), c(12, 18, 33))
```

measles_cdmx *Weekly measles cases, Ciudad de Mexico (worked example)*

Description

A two-column weekly case-count series used as the package's worked example. Time is a weekly index, not a calendar date; see `vignette("rtforecastr-walkthrough")` for the recommended `mean_GI`/`var_GI` values to use with this series.

Usage

```
measles_cdmx
```

Format

A data frame with columns:

time integer, weekly time index

cases integer, case count for that week

Source

Worked example shipped with RtForecastR; not an official surveillance release.

mse *Mean squared error*

Description

Mean squared error

Usage

```
mse(observed, predicted)
```

Arguments

observed	numeric vector of realized values
predicted	numeric vector of point predictions (same length/order)

Value

a single numeric value

Examples

```
mse(c(10, 20, 30), c(12, 18, 33))
```

plot.rtforecast *Plot method for rtforecast objects*

Description

Draws to the current graphics device (matches normal R plotting conventions - use grDevices::png()/grDevices::pdf() yourself around the call if you want a file, the way you would for any base R plot). Ports the "current situation" view from the original run_rtforecast.R script.

Usage

```
## S3 method for class 'rtforecast'
plot(
  x,
  which = c("Rt", "forecast", "observed_vs_predicted"),
  n_recent = 8,
  ...
)
```

Arguments

x	an "rtforecast" object, as returned by rt_forecast()
which	one of "Rt" (filtered vs smoothed R_t), "forecast" (recent observed/predicted cases + one-step forecast), or "observed_vs_predicted" (full-period observed vs in-sample predicted + forecast)
n_recent	for which = "forecast", how many recent time points to show (default 8)
...	passed on to the underlying graphics::plot() call

Value

The input "rtforecast" object x, returned invisibly. Called primarily for its side effect of drawing to the current graphics device.

---

rmse *Root mean squared error*

Description

Root mean squared error

Usage

```
rmse(observed, predicted)
```

Arguments

observed	numeric vector of realized values
predicted	numeric vector of point predictions (same length/order)

Value

a single numeric value

Examples

```
rmse(c(10, 20, 30), c(12, 18, 33))
```

`rt_forecast` *Estimate and forecast the time-varying effective reproduction number*

Description

Fits filtered (real-time/causal) and smoothed (retrospective) R_t estimates to a case-count time series using the EpiFilter algorithm (Parag 2021), computes one-step-ahead in-sample predictions (a model adequacy check), a genuine out-of-sample one-step forecast with predictive intervals, and the elimination probability P(R_t < 1) at every time point. Disease-agnostic: supply the generation interval for your own pathogen via `mean_GI`/`var_GI`.

Usage

```
rt_forecast(
  time,
  cases,
  mean_GI,
  var_GI,
  Rmin = 0.01,
  Rmax = 10,
  grid_size = 200,
  eta = 0.1,
  ci_level = 0.025,
  quantile_levels = c(0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45),
  maxI = NULL
)
```

Arguments

`time`	numeric vector of time indices (e.g. epi weeks)
`cases`	numeric vector of case counts, same length as `time`
`mean_GI`	mean generation interval, in the same time units as `time` (e.g. weeks if your data is weekly)
`var_GI`	variance of the generation interval, same time units
`Rmin`, `Rmax`	bounds of the grid searched over for R_t (default 0.01, 10)
`grid_size`	number of points in the R_t grid (default 200)
`eta`	diffusion (state) noise controlling smoothness of R_t (default 0.1)
`ci_level`	tail probability defining the reported main credible interval (default 0.025, i.e. a 95% interval)
`quantile_levels`	additional quantile levels (each in (0, 0.5)) to compute for the out-of-sample forecast, used by `wis()`. Default adds the levels a standard 11-interval WIS wants; set to `NULL` to skip and only compute the 50%/95% forecast interval.
`maxI`	upper bound of the internal prediction grid; if `NULL` (default) it's set automatically to `3 * max(cases)` (minimum 2000)

Details

Unlike a script, this returns an object rather than writing files - use `plot.rtforecast()` if you want the plots `run_rtforecast.R` used to write directly.

Value

an object of class `"rtforecast"`: a list with elements `results` (per-time R_t estimates and elimination probability), `predictions` (in-sample one-step-ahead predictions), `forecast` (the one-step-ahead out-of-sample forecast, with a `quantiles` element if `quantile_levels` was set), and the raw filter/smoother objects for advanced use.

Examples

```
data(measles_cdmx)
fit <- rt_forecast(measles_cdmx$time, measles_cdmx$cases,
                   mean_GI = 11/7, var_GI = (4/7)^2)
head(fit$results)
fit$forecast
```

---

score_batches *Score realized-vs-forecast performance across weekly output batches*

Description

Recursively finds every `asof_week*/epifilter_forecast_next_week.csv` under `root_dir` and matches each batch's one-week-ahead forecast to the actual case count once it appears as an observed row in a *later* batch's `epifilter_results.csv` - i.e. this scores genuine prospective (not in-sample) forecast performance over time.

Usage

```
score_batches(root_dir, revision = c("first", "latest"))
```

Arguments

`root_dir`	path under which to search recursively for asof_week* folders (e.g. `"output"`)
`revision`	`"first"` or `"latest"` - see above

Details

Surveillance case counts are commonly revised between runs as reporting catches up (backfill), so the same (location, time) can legitimately have different case counts across different asof_week batches. This function makes that choice explicit via `revision`: `"first"` uses the count as it was first reported (matches classic prospective forecast evaluation); `"latest"` uses the most complete/least revised count available.

This function is a filesystem helper for scoring output produced by the standalone `run_rtforecast.R` script's weekly batch folders. If you're using `rt_forecast()` directly in R (recommended for new code), accumulate a data.frame of your own forecasts/actuals across weeks and use `mae()`, `coverage()`, and `wis()` directly instead.

Value

a data.frame, one row per batch whose forecast has since been realized, with location, time, forecast, actual, and per-point MAE/coverage (95% and 50%) columns.

`wis` *Weighted interval score (WIS)*

Description

Bracher J, Ray EL, Gneiting T, Reich NG. (2021) "Evaluating epidemic forecasts in an interval format." PLOS Computational Biology 17(2): e1008618. doi:10.1371/journal.pcbi.1008618

Usage

```
wis(observed, median_pred, quantiles)
```

Arguments

`observed`	numeric vector of realized values, length T
`median_pred`	numeric vector of median (or mean, if no median is available) point predictions, length T
`quantiles`	a named list, as returned by `rt_forecast()$forecast_quantiles` or directly from `recurs_predict_quantiles()`$quantiles: each element is a 2 x T matrix with rows lo/hi for a (level, 1-level) interval.

Value

a data.frame with per-time-point WIS and its components, plus the overall mean WIS as an attribute (`"mean_wis"`)

Examples

```
q <- list("0.025" = matrix(c(2, 8, 15, 25), nrow = 2),
          "0.25"  = matrix(c(4, 6, 18, 22), nrow = 2))
result <- wis(observed = c(5, 20), median_pred = c(5, 21), quantiles = q)
attr(result, "mean_wis")
```

Index