Package 'ViroReportR'

May 7, 2026

Title Respiratory Viral Infection Forecast Reporting

Version 1.0.4

Description Tools for reporting and forecasting viral respiratory infections,
using case surveillance data. Report generation tools for short-term forecasts,
and validation metrics for an arbitrary number of customizable respiratory viruses.
Estimation of the effective reproduction number is based on the 'EpiEstim' framework
described in work by 'Cori' and colleagues. (2013) <doi:10.1093/aje/kwt133>.

License GPL (>= 3)

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

Imports data.table, dplyr, EpiEstim, ggplot2, incidence, lubridate,
projections, purrr, rlang, tibble, stats, utils, tidyr, glue,
mgcv, kableExtra, cowplot

VignetteBuilder knitr

Depends R (>= 4.1.0)

URL https://github.com/BCCDC-PHSA/ViroReportR

BugReports https://github.com/BCCDC-PHSA/ViroReportR/issues

NeedsCompilation no

Author Mike Irvine [aut, cre, cph],
Caesar Wong [aut],
Nirupama Tamvada [aut],
Rebeca Falcao [aut],
Nelson Tang [aut]

Maintainer Mike Irvine <mike.irvine@bccdc.ca>

Repository CRAN

Date/Publication 2026-02-17 23:10:27 UTC

Contents

clean_sample_data . . . 2
create_quantiles . . . 3
current_forecast_text . . . 3
fit_epiestim_model . . . 4
forecast_metrics . . . 5
generate_forecast . . . 5
generate_forecast_report . . . 7
generate_validation . . . 8
generate_validation_metric . . . 10
get_aggregated_data . . . 12
plot_rt . . . 13
plot_validation . . . 14
project_epiestim_model . . . 15
simulate_data . . . 15

Index **17**

`clean_sample_data` *Clean and validate case count data for EpiEstim*

Description

This function prepares case count data for use with **EpiEstim** by performing a series of validation and cleaning steps:

Usage

```
clean_sample_data(data, start_date)
```

Arguments

`data`	A data frame containing at least the columns `"date"` and `"confirm"`. The `"date"` column should be of class `Date`, and `"confirm"` should be numeric.
`start_date`	A `Date` (or date-convertible string) indicating the starting date for analysis. Must exist within the `"date"` column.

Details

1. Ensures that the input data frame has the required columns: `"date"` and `"confirm"`.
2. Confirms that the specified `start_date` exists in the data and filters the data to include only records on or after that date.
3. Removes leading days before the first non-zero confirmed case.
4. Verifies that the resulting dataset contains at least 14 valid days (as required for estimation).

This function is primarily intended as a preprocessing step for EpiEstim modeling. It combines validation checks for input structure and time coverage with minimal data cleaning logic to ensure robust downstream estimation.

Value

A cleaned data frame filtered from `start_date`, starting at the first date with non-zero confirmed cases, and containing at least 14 days of data.

---

create_quantiles *summarise a data frame* d *by groups along a* variable

---

Description

summarise a data frame `d` by groups along a `variable`

Usage

```
create_quantiles(d, ..., variable = NULL)
```

Arguments

`d`	tibble data frame
`...`	group_by variables
`variable`	string

Value

Data frame containing sample quantiles at probabilities 0.05, 0.25, 0.50, 0.75 and 0.95

---

current_forecast_text *Print out text output for ViroReportR report detailing current number of case visits, last value of Rt and corresponding intervals*

Description

Print out text output for ViroReportR report detailing current number of case visits, last value of Rt and corresponding intervals

Usage

```
current_forecast_text(time_period_result, ...)
```

Arguments

`time_period_result`	output from `forecast_time_period`
`...`	optional arguments to be passed on to `forecast_metrics`

Value

current forecast metrics

`fit_epiestim_model` *fit_epiestim_model - Function to estimate the reproduction number of an epidemic*

Description

A wrapper function for `estimate_R` from the `EpiEstim` library to estimate the reproduction number of epidemics to support short-term forecasts

Usage

```
fit_epiestim_model(
  data,
  window_size = 7L,
  type = NULL,
  mean_si = NULL,
  std_si = NULL,
  recon_opt = "match",
  method = "parametric_si",
  mean_prior = NULL,
  std_prior = NULL
)
```

Arguments

`data`	*data frame* containing two columns: date and confirm (number of cases)
`window_size`	*Integer* Length of the sliding windows used for R estimates.
`type`	*character* Specifies type of epidemic. Must be one of "flu_a", "flu_b", "rsv", "sars_cov2" or "custom"
`mean_si`	*Numeric* User specification of mean of parametric serial interval
`std_si`	*Numeric* User specification of standard deviation of parametric serial interval
`recon_opt`	Not implemented. One of "naive" or "match" to pass on to `estimate_R` (see help page)
`method`	One of "non_parametric_si", "parametric_si", "uncertain_si", "si_from_data" or "si_from_sample" to pass on to `estimate_R` (see help page)
`mean_prior`	*Numeric* positive number giving the mean of the common prior distribution for all reproduction numbers
`std_prior`	*Numeric* positive number giving the standard deviation of the common prior distribution for all reproduction numbers

Details

`fit_epiestim_model` currently supports the following epidemics: Influenza, RSV and COVID-19. The default serial intervals for the estimation of R were retrieved from Cowling et al., 2011, Vink et al., 2014 and Madewell et al., 2023 for Influenza A, Influenza B, RSV and COVID (BA.5 Omicron variant) respectively

Value

Object of class estimate_R (see EpiEstim help page)

---

forecast_metrics *Extract current forecast metrics: forecast prediction, percentile interval and Rt value*

Description

Extract current forecast metrics: forecast prediction, percentile interval and Rt value

Usage

```
forecast_metrics(time_period_result, iter = 10)
```

Arguments

time_period_result	output from forecast_time_period
iter	number of MCMC iterations used to generate Rt posterior

Value

dataframe of current forecast metrics

---

generate_forecast *Forecast daily epidemic cases using EpiEstim*

Description

This function prepares epidemic data, estimates the reproduction number (R_t) using fit_epiestim_model, and produces short-term forecasts of daily confirmed cases with project_epiestim_model.

It removes early periods with no cases, checks data validity, optionally smooths the epidemic curve, and then generates forward projections of cases for a specified number of days.

Usage

```
generate_forecast(
  data,
  start_date,
  window_size = 7,
  n_days = 7,
  type = NULL,
  smooth_data = FALSE,
  smoothing_cutoff = 10,
  ...
)
```

Arguments

data	*data frame* Must contain two columns: • date: observation dates • confirm: daily confirmed cases
start_date	*Date* Date after which the epidemic is considered to have started. Data before this date is removed.
window_size	*Integer* Length of the sliding window (in days) used for reproduction number estimation. Default is 7.
n_days	*Integer* Number of future days to forecast. Default is 7.
type	*character* Type of epidemic. Must be one of "flu_a", "flu_b", "rsv", "sars_cov2", or "custom". Passed to fit_epiestim_model.
smooth_data	*logical* Whether to smooth the input daily case counts before estimation. Default is FALSE.
smoothing_cutoff	*Integer* Cutoff parameter for smoothing. Only used if smooth_data = TRUE. Default is 10.
...	Additional arguments passed to fit_epiestim_model.

Details

- Data prior to the first non-zero confirm value is excluded.
- Input is checked for validity (sufficient days, proper format).
- If smoothing is enabled, case counts are adjusted before fitting.
- Forecasts are generated from the fitted EpiEstim model and returned with quantiles (2.5%, 25%, 50%, 75%, 97.5%), minimum, and maximum.

Value

A data frame of forecasted daily incidence with columns:

- date: date of forecast
- p50, p25, p75, p025, p975: forecast quantiles
- min_sim, max_sim: forecast range

See Also

fit_epiestim_model for reproduction number estimation, project_epiestim_model for forward simulations.

Examples

```
# Create sample test rsv data
disease_type <- "rsv"
test_data <- simulate_data()
formatted_data <- get_aggregated_data(
  test_data,
```

```
  number_column = disease_type,
  date_column = "date",
  start_date = "2024-04-01",
  end_date = "2024-05-01"
)

# Run a 7 day forecast with smoothing
res_smooth <- generate_forecast(
  data = formatted_data,
  start_date = "2024-04-01",
  n_days = 7,
  type = "rsv",
  smooth_data = FALSE
)
```

`generate_forecast_report` *Generate Viral Respiratory Forecast Report*

Description

Generates a full-season forecast report for viral respiratory diseases as an HTML document.

Usage

```
generate_forecast_report(
  input_data_dir = NULL,
  output_dir = NULL,
  n_days = 7,
  validate_window_size = 7,
  smooth = FALSE,
  disease_season = NULL
)
```

Arguments

`input_data_dir`	Path to input CSV data. Must contain columns: `date`, `confirm`, `disease_type`. Allowed values for `disease_type`: `"flu_a"`, `"flu_b"`, `"rsv"`, `"sars_cov2"`, `"custom"`.
`output_dir`	Path to output directory for the rendered HTML report.
`n_days`	Number of days ahead to forecast. Default is 7.
`validate_window_size`	The number of days between each validation window. Default is 7.
`smooth`	Logical indicating whether smoothing should be applied in the forecast. Default is `TRUE`.

disease_season An optional named list specifying the seasonal date ranges for each disease. Each element should be either:

- NULL (indicating no defined season), or
- a two-date vector in "YY-MM-DD" format (e.g., c("2024-09-01", "2025-03-01")) defining the start and end of the season for that disease.

For example: disease_season = list(flu_a = c("2024-09-01", "2025-03-01"), rsv = c("2024-09-01", "2025-03-01"), sars_cov2 = NULL)

This will produce a report where influenza A and RSV seasons run from September 1, 2024 to March 1, 2025, while no season is defined for SARS-CoV-2.

Value

Invisibly returns the path to the rendered HTML report.

Examples

```
data <- simulate_data(start_date = "2024-01-07", #starting Sunday
)
diseases <- c("flu_a", "rsv", "sars_cov2")
data$date <- lubridate::ymd(data$date)
vri_data_list <- purrr::set_names( purrr::map2( rep(list(data), length(diseases)),
                                  diseases,
                                  ~ get_aggregated_data(.x, "date", .y)
                                 ),
                           diseases
)
# Save the simulated data
df <- purrr::imap_dfr(
vri_data_list,
\(df, disease) dplyr::mutate(df, disease_type = disease)
)
tmp_dir <- tempdir() # temporary directory for example for saving data
data_path <- file.path(tmp_dir, "simulated_data.csv")
write.csv(df, data_path, row.names = FALSE)

output_path <- tempdir() # output directory for report (temporary as example)
generate_forecast_report(input_data_dir = data_path,
                         output_dir = output_path,
                         n_days = 7,
                         validate_window_size = 7,
                         smooth = FALSE)
```

---

generate_validation *Validate forecast performance over multiple time windows*

Description

This function performs rolling validation of short-term forecasts generated by **EpiEstim** or similar models. It divides the input time series into overlapping validation windows and repeatedly runs forecasts to assess model performance across different time segments.

Usage

```
generate_validation(
  data,
  start_date,
  validate_window_size = 7,
  window_size = 7,
  n_days = 7,
  type = NULL,
  smooth_data = FALSE,
  smoothing_cutoff = 10,
  ...
)
```

Arguments

`data`	A data frame containing at least the columns `"date"` and `"confirm"`. The `"date"` column should be of class `Date`, and `"confirm"` should be numeric.
`start_date`	A `Date` (or date-convertible string) specifying the starting point for validation Must exist in the `"date"` column.
`validate_window_size`	Integer. The number of days between each validation window (default: `7`).
`window_size`	Integer. The sliding window size (in days) used by the forecasting model (default: `7`).
`n_days`	Integer. The number of future days to forecast in each validation iteration (default: `7`).
`type`	*character* Type of epidemic. Must be one of `"flu_a"`, `"flu_b"`, `"rsv"`, `"sars_cov2"`, or `"custom"`. Passed to `fit_epiestim_model`.
`smooth_data`	Logical. Whether to smooth the input case counts prior to forecasting (default: `FALSE`).
`smoothing_cutoff`	Numeric. Threshold used for smoothing when `smooth_data = TRUE` (default: `10`).
`...`	Additional arguments passed to `generate_forecast()`.

Details

The validation procedure ensures that forecasts are evaluated under realistic temporal conditions. Starting from the earliest date, the function repeatedly:

1. Takes a growing subset of data up to the current validation endpoint.
2. Runs the forecast using `generate_forecast()`.

3. Moves the validation window forward by `validate_window_size` days.

This results in a set of forecasts that can be compared to observed data to evaluate predictive performance across time.

Value

A list of forecast results, each element corresponding to one validation window. Each element contains the output returned by `generate_forecast()` for that particular window.

See Also

`clean_sample_data()`, `generate_forecast()`

Examples

```
data <- simulate_data()
formatted_data <- get_aggregated_data(data,"date", "flu_a", "2024-10-16", "2024-12-31")
start_date <- as.Date("2024-10-16")
validation_results <- generate_validation(formatted_data, start_date, type="flu_a")
```

---

generate_validation_metric

Compute Forecast Validation Metrics (SMAPE & MASE)

Description

This function evaluates forecast accuracy across multiple validation runs by computing two key performance metrics:

Usage

```
generate_validation_metric(data, validation_res)
```

Arguments

`data`	A data frame used in `generate_validation()`, containing the **original training data** for the model. It must include: • `date`: Dates of the observed case data (class `Date`). • `confirm`: Numeric values of observed confirmed cases.
`validation_res`	A list of forecast validation results, typically the output from `generate_validation()`. Each element should contain: • `forecast_res_quantiles`: A data frame with columns `date` and `p50` (median forecasted values). • `original_data`: A data frame representing the training data used for that forecast, with a `date` column.

Details

- **Symmetric Mean Absolute Percentage Error (SMAPE)**: Measures relative forecast accuracy while remaining robust to zero values in the actual data.
- **Mean Absolute Scaled Error (MASE)**: Scales forecast errors relative to the in-sample one-step naïve forecast, allowing comparison across series with different scales.

For each forecast result, the function also reports the corresponding training and forecast periods. Computation stops once the forecast period reaches the maximum date in the model data.

- **SMAPE** is defined as:

$$SMAPE = mean(|F - A|/((|A| + |F|)/2))$$

  where \(A\) are actual values and \(F\) are forecasts. It avoids division by zero and is suitable for count data with zeros.

- **MASE** compares the mean absolute forecast error against the mean absolute difference of successive actual:

$$MASE = mean(|A - F|)/mean(|diff(A)|)$$

The function automatically excludes forecasts extending beyond the latest date in the observed model data.

Value

A `tibble` (data frame) with one row per forecast result and the following columns:

- `train_period`: Date range of the training period used for the forecast.
- `forecast_period`: Date range of the forecasted period.
- `smape`: Symmetric Mean Absolute Percentage Error between forecasted and actual values, rounded to two decimals.
- `mase`: Mean Absolute Scaled Error, rounded to two decimals.

See Also

`generate_validation()`, `generate_forecast()`

Examples

```
data <- simulate_data()
formatted_data <- get_aggregated_data(data,"date", "flu_a", "2024-10-16", "2024-12-31")
start_date <- ("2024-10-16")
validation_results <- generate_validation(formatted_data, start_date, type ="flu_a")
generate_validation_metric(formatted_data, validation_results)
```

`get_aggregated_data` *Extract Aggregated Weekly Generic Data*

Description

`get_aggregated_data()` performs data transformation in the following steps:

1. Group the weekly or daily data by date.
2. Aggregate the number of confirmed cases by either day or week.
3. Select only the date and confirmed cases column.
4. Filter the data by given start and end date

The input dataframe `generic_data` must have the following columns:

- `<date name>`: date column (e.g. as.Date('2022-01-01')).
- `<cases count name>`: Confirmed Cases Count (e.g. 1, 2, ...).

Note that these columns can be defined in a generic name, and inputted as the other two function parameters for data transformation (`date_column`, `number_column`)

Assume the date column is the start of the epiweek.

Usage

```
get_aggregated_data(
  generic_data,
  date_column,
  number_column,
  start_date = NULL,
  end_date = NULL,
  unit = "day"
)
```

Arguments

`generic_data`	the weekly generic data from `get_data()`
`date_column`	date column name str
`number_column`	cases count column name str
`start_date`	start date string (e.g. '2022-01-01')(optional, default is NULL)
`end_date`	end date string (e.g. '2022-12-31')(optional, default is NULL)
`unit`	aggregation unit "day" or "week"

Value

aggregated weekly data of the generic confirmed cases data (filtered by date if any)

date Either day or week date

confirm number of confirmed cases

Examples

```
sim_data <- simulate_data()
aggregated_data <- get_aggregated_data(
  sim_data,
  "date", "flu_a", "2024-10-16", "2024-12-31"
)
```

`plot_rt` *Plot Mean Rt with time index (dates)*

Description

Plot Mean Rt with time index (dates)

Usage

```
plot_rt(forecast_results)
```

Arguments

`forecast_results`
: is the output of `generate_forecast`.

Value

Mean Rt with time index plot

Examples

```
# Create sample test rsv data
disease_type <- "rsv"
test_data <- simulate_data()
formatted_data <- get_aggregated_data(
  test_data,
  number_column = disease_type,
  date_column = "date",
  start_date = "2024-04-01",
  end_date = "2024-05-01"
)

# Run a 7 day forecast with smoothing
forecast_results <- generate_forecast(
  data = formatted_data,
  start_date = "2024-04-01",
  n_days = 7,
  type = "rsv",
  smooth_data = FALSE
)
plot_rt(forecast_results)
```

`plot_validation` *Plot a ribbon plot with each time horizon predictions against true values for validation*

Description

Plot a ribbon plot with each time horizon predictions against true values for validation

Usage

```
plot_validation(data, validation_res, pred_plot = "ribbon")
```

Arguments

`data`	A data frame used in `generate_validation()`, containing the original training data used for model fitting. Must include: • `date`: Dates of the observed data (class `Date`). • `confirm`: Numeric values of confirmed cases.
`validation_res`	A list of forecast validation results, typically produced by `generate_validation()`. Each element should include: • `forecast_res_quantiles`: A data frame containing forecasted quantiles (`p025`, `p10`, `p25`, `p50`, `p75`, `p90`, `p975`, and `date`). • `estimate_R`: A list with estimated reproduction numbers (used for grouping). • `smoothed_data`: (optional) A data frame of smoothed observations, if smoothing was applied before forecasting.
`pred_plot`	either `"ribbon"` or `"error_bar"` (by default) to produce either ribbon prediction plots or error_bar plots respectively

Value

error_bar validation plot or ribbon validation plot for a specific prediction horizon

Examples

```
data <- simulate_data()
formatted_data <- get_aggregated_data(data,"date", "flu_a", "2024-10-16", "2024-12-31")
start_date <- ("2024-10-16")
validation_results <- generate_validation(formatted_data, start_date, type ="flu_a")
plot_validation(formatted_data, validation_results)
```

project_epiestim_model *Extract daily forecast samples*

Description

Function to produce short-term daily projections from objects of class `estimate_R`

Usage

```
project_epiestim_model(data, model_fit, n_days = 7, n_sim = 1000)
```

Arguments

`data`	*data frame* containing two columns: date and confirm (number of cases per day)
`model_fit`	Object of class `estimate_R` generated by running `fit_epiestim_model`
`n_days`	The number of days to run simulations for. Defaults to 14
`n_sim`	The number of epicurves to simulate. Defaults to 1000

Value

Data-frame of daily forecast samples from all simulations

date date

incidence projected number of daily confirmed cases

sim simulation run number

simulate_data *Simulate Daily Virus Incidence Data*

Description

Generates simulated daily incidence data for specified respiratory viruses over a defined number of days. Each virus is modeled using a Gaussian-like curve, parameterized by peak day, amplitude, and scale.

Usage

```
simulate_data(
  days = 365,
  peaks = c(flu_a = 90, rsv = 110, sars_cov2 = 160),
  amplitudes = c(flu_a = 50, rsv = 40, sars_cov2 = 20),
  scales = c(flu_a = -0.004, rsv = -0.005, sars_cov2 = -0.001),
  time_offset = 0,
  noise_sd = 5,
  start_date = "2024-01-07"
)
```

Arguments

days	Integer. Number of days to simulate (default is 365).
peaks	Named numeric vector. Peak day for each virus (e.g., c("flua"=90,"rsv"=110,"sars_cov2"=160)).
amplitudes	Named numeric vector. Amplitude for each virus's peak (e.g., c("flua"=50,"rsv"=40,"sars_cov2"=20
scales	Named numeric vector. Scale controlling spread of the peak for each virus (e.g., c("flua"=-0.004,"rsv"=-0.005,"sars_cov2"=-0.001)).
time_offset	Integer. Number of days to offset start of the simulation. useful if want to test data with larger values in the middle of a respiratory season.
noise_sd	numeric or named numeric. Gaussian noise applied to each virus signal. can either be a single value or named for each virus e.g., c("flu_a"=2,"rsv"=5,"sars_cov2"=7)
start_date	string

Value

A data frame with daily simulated incidence counts for each virus, including a date column.

Examples

```
simulate_data()
simulate_data(days = 100, peaks = c(flu_a = 30), amplitudes = c(flu_a = 60),
scales = c(flu_a = -0.01), noise_sd = c(flu_a = 5))
```

Index

clean_sample_data, 2
clean_sample_data(), *10*
create_quantiles, 3
current_forecast_text, 3

estimate_R, *4, 5*, *15*

fit_epiestim_model, 4, *5, 6*, *9*
forecast_metrics, 5

generate_forecast, 5
generate_forecast(), *10, 11*
generate_forecast_report, 7
generate_validation, 8
generate_validation(), *10, 11*, *14*
generate_validation_metric, 10
get_aggregated_data, 12

plot_rt, 13
plot_validation, 14
project_epiestim_model, *5, 6*, 15

simulate_data, 15