Package: EpiStrainDynamics (via r-universe)

August 24, 2026

Title Infer temporal trends of multiple pathogens

Version 0.1.0

Description 'EpiStrainDynamics' is a statistical framework developed for inferring temporal trends of multiple pathogens from routinely collected surveillance data.

BugReports https://github.com/ropensci/EpiStrainDynamics/issues

License Apache License 2.0

URL https://docs.ropensci.org/EpiStrainDynamics, https://github.com/ropensci/EpiStrainDynamics

Encoding UTF-8

Language en

RoxygenNote 7.3.3

Roxygen list (markdown = TRUE, roclets = c (` `namespace", ` `rd", ` `srr::srr_stats_roclet"))

Biarch true

Depends R (>= 4.1.0)

Imports bayesplot, cli, dplyr, ggplot2, methods, purrr, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rlang, rstan (>= 2.18.1), rstantools (>= 2.4.0), timetk, tsibble, viridis

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

SystemRequirements GNU make

Suggests data.table, knitr, piggyback, rmarkdown, testthat (>= 3.0.0), tibble, tibbletime, units, xts, zoo

Config/testthat/edition 3

Config/testthat/parallel true

LazyData true

VignetteBuilder knitr

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libssl-dev libx11-dev

Repository https://ropensci.r-universe.dev

Date/Publication 2026-08-24 10:15:27 UTC

RemoteUrl https://github.com/ropensci/EpiStrainDynamics

RemoteRef main

RemoteSha d53c8416eb3bfebe9bdb16fafc2d12e8ae73eae2

Contents

construct_model . . . 2
diagnose_model . . . 3
dispersion_structure . . . 4
fit_model . . . 5
growth_rate . . . 9
incidence . . . 10
influenza . . . 12
multiple . . . 13
p_spline . . . 14
plot . . . 15
print.EpiStrainDynamics.model . . . 16
proportion . . . 17
random_walk . . . 19
Rt . . . 20
sarscov2 . . . 22
single . . . 23
smoothing_structure . . . 24
subtyped . . . 25

Index **27**

`construct_model` *Construct model*

Description

Construct model

Usage

```
construct_model(
  pathogen_structure,
  method,
  smoothing_params = smoothing_structure(),
  dispersion_params = dispersion_structure(),
  pathogen_noise = FALSE,
  dow_effect = FALSE
)
```

Arguments

pathogen_structure	either single(), multiple(), or subtyped()
method	either random_walk() or p_spline()
smoothing_params	argument is optional and defines the structure of the smoothing terms including optionally setting the smoothing prior tau. Created with smoothing_structure(). NULL option defaults to "shared" smoothing structure and default priors.
dispersion_params	argument is optional and defines priors for the overdispersion parameter of the negative binomial likelihood for the case timeseries. Created using dispersion_structure(). NULL option uses default priors for phi.
pathogen_noise	logical whether individual pathogen counts have additional gamma-distributed noise. Default is FALSE. Models with single pathogen structure will be set to FALSE.
dow_effect	logical whether to incorporate a day of week model.

Value

a list containing the data, the model parameters, and pathogen names of class EpiStrainDynamics.model

Examples

```
mod <- construct_model(
  pathogen_structure = multiple(
    data = sarscov2,
    case_timeseries = "cases",
    time = "date",
    component_pathogen_timeseries = c("alpha", "delta", "omicron", "other")
  ),
  method = p_spline(),
  smoothing_params = smoothing_structure(
    "independent",
    tau_mean = c(0, 0.1, 0.3, 0), tau_sd = rep(1, times = 4)
  ),
  dispersion_params = dispersion_structure(phi_mean = 0, phi_sd = 1),
  pathogen_noise = FALSE,
  dow_effect = TRUE
)
```

---

diagnose_model *Diagnose model convergence and fit*

Description

Checks MCMC convergence diagnostics — R-hat and effective sample size — against user-specified thresholds, and reports any parameters that fail either check.

Usage

```
diagnose_model(fitted_model, rhat_threshold = 1.1, eff_sample_threshold = 100)
```

Arguments

`fitted_model`	A fitted model object of class `EpiStrainDynamics.fit`, as returned by `fit_model()`.
`rhat_threshold`	R-hat threshold for convergence (default 1.1). Values above this threshold indicate chains have not mixed well.
`eff_sample_threshold`	Effective sample size threshold (default 100). Values below this threshold indicate the posterior samples for a parameter are too autocorrelated to reliably estimate its distribution.

Value

An object of class `list`, returned invisibly, containing:

`convergence`	Logical; `TRUE` if no parameter fails either threshold
`rhat_issues`	Character vector of parameter names with R-hat above `rhat_threshold`
`eff_sample_issues`	Character vector of parameter names with effective sample size below `eff_sample_threshold`
`max_rhat`	The largest R-hat value across all parameters
`min_neff`	The smallest effective sample size across all parameters
`summary`	The full `rstan::summary()` table the above are derived from

Examples

```
mod <- construct_model(
  pathogen_structure = single(
    case_timeseries = sarscov2$cases,
    time = sarscov2$date
  ),
  method = random_walk()
)
fit <- fit_model(mod)
diagnose_model(fit)
```

---

dispersion_structure *Create Dispersion Structure Specification*

Description

This function creates a dispersion structure object that specifies priors for the overdispersion parameter of the negative binomial likelihood for the case timeseries. `phi` controls how much the observed counts vary around the expected trend beyond Poisson noise: smaller values allow more overdispersion (noisier counts relative to the mean), larger values approach Poisson-like variance.

Usage

```
dispersion_structure(phi_mean = NULL, phi_sd = NULL)
```

Arguments

phi_mean	Numeric scalar specifying the prior mean for the negative binomial dispersion parameter. Must be positive. Optional - if not provided, no priors will be set.
phi_sd	Numeric scalar specifying the prior standard deviation for the dispersion parameter. Must be positive. Optional - if not provided, no priors will be set.

Value

An object of class `EpiStrainDynamics.dispersion` containing:

mean	The prior mean for phi
sd	The prior standard deviation for phi
priors_provided	Integer flag passed to the Stan model: 1 if no priors were supplied (Stan's built-in default prior is used), 2 if priors were supplied (the phi_mean/phi_sd values are used as the prior)

Examples

```
# Create dispersion structure
disp_struct <- dispersion_structure(phi_mean = 2.0, phi_sd = 0.5)
```

---

fit_model *Generic Method for fitting model*

Description

S3 generic for fitted models from constructed model object

Usage

```
fit_model(
  constructed_model,
  n_chain = 4,
  n_iter = 2000,
  n_warmup = floor(n_iter/2),
  thin = 1,
  adapt_delta = 0.9,
  multi_cores = TRUE,
  verbose = TRUE,
  suppress_warnings = FALSE,
  seed = NULL,
```

```
  ...
)

## S3 method for class 'rw_subtyped'
fit_model(
  constructed_model,
  n_chain = 4,
  n_iter = 2000,
  n_warmup = floor(n_iter/2),
  thin = 1,
  adapt_delta = 0.9,
  multi_cores = TRUE,
  verbose = TRUE,
  suppress_warnings = FALSE,
  seed = NULL,
  ...
)

## S3 method for class 'ps_subtyped'
fit_model(
  constructed_model,
  n_chain = 4,
  n_iter = 2000,
  n_warmup = floor(n_iter/2),
  thin = 1,
  adapt_delta = 0.9,
  multi_cores = TRUE,
  verbose = TRUE,
  suppress_warnings = FALSE,
  seed = NULL,
  ...
)

## S3 method for class 'rw_multiple'
fit_model(
  constructed_model,
  n_chain = 4,
  n_iter = 2000,
  n_warmup = floor(n_iter/2),
  thin = 1,
  adapt_delta = 0.9,
  multi_cores = TRUE,
  verbose = TRUE,
  suppress_warnings = FALSE,
  seed = NULL,
  ...
)
```

```
## S3 method for class 'ps_multiple'
fit_model(
  constructed_model,
  n_chain = 4,
  n_iter = 2000,
  n_warmup = floor(n_iter/2),
  thin = 1,
  adapt_delta = 0.9,
  multi_cores = TRUE,
  verbose = TRUE,
  suppress_warnings = FALSE,
  seed = NULL,
  ...
)

## S3 method for class 'rw_single'
fit_model(
  constructed_model,
  n_chain = 4,
  n_iter = 2000,
  n_warmup = floor(n_iter/2),
  thin = 1,
  adapt_delta = 0.9,
  multi_cores = TRUE,
  verbose = TRUE,
  suppress_warnings = FALSE,
  seed = NULL,
  ...
)

## S3 method for class 'ps_single'
fit_model(
  constructed_model,
  n_chain = 4,
  n_iter = 2000,
  n_warmup = floor(n_iter/2),
  thin = 1,
  adapt_delta = 0.9,
  multi_cores = TRUE,
  verbose = TRUE,
  suppress_warnings = FALSE,
  seed = NULL,
  ...
)
```

Arguments

`constructed_model`
: prepared model object of class `EpiStrainDynamics.model`

n_chain	number of MCMC chains, defaults to 4
n_iter	A positive integer specifying the number of iterations for each chain, default value is 2000
n_warmup	A positive integer specifying the number of warmup iterations,default value is half the number of iterations
thin	A positive integer specifying the period for saving samples, default value is 1.
adapt_delta	Numeric value between 0 and 1 indicating target average acceptance probability used in rstan::sampling. Default value is 0.9.
multi_cores	A logical value indicating whether to parallelize chains with multiple cores, default is TRUE and uses all available cores - 1.
verbose	Logical value controlling the verbosity of output. When TRUE (default), shows all messages, warnings, errors, and progress indicators. When FALSE, suppresses messages and progress while retaining warnings and errors.
suppress_warnings	Logical value indicating whether to suppress warnings from Stan. Default is FALSE. When TRUE, warnings are suppressed but errors are still raised.
seed	A positive integer seed used for random number generation in MCMC. Default is NULL, which means the seed is generated from 1 to the maximum integer supported by R.
...	additional arguments to rstan::sampling(), such as init

Value

fit model of class EpiStrainDynamics.fit, or if fitting fails, an error is raised that can be caught and inspected.

Examples

```
mod <- construct_model(
  pathogen_structure = single(
    case_timeseries = sarscov2$cases,
    time = sarscov2$date
  ),
  method = random_walk()
)

fit <- fit_model(mod)

# Suppress progress and messages but keep warnings/errors
fit <- fit_model(mod, verbose = FALSE)

# Suppress warnings too
fit <- fit_model(mod, verbose = FALSE, suppress_warnings = TRUE)

# Catch errors and inspect
result <- tryCatch(
  fit_model(mod),
  error = function(e) e
```

```
)
if (inherits(result, "EpiStrainDynamics.fit.error")) {
  cat("Fitting failed:", result$message, "\n")
  # Can still access the model: result$constructed_model
}
```

`growth_rate` *Generic Method for Growth Rate Analysis*

Description

Computes the epidemiological growth rate, defined as the instantaneous rate of change in log-incidence over time. Mathematically, it represents:

$$r_t = \log(I_t) - \log(I_{t-1}) = \log\left(\frac{I_t}{I_{t-1}}\right)$$

where I_t is incidence (see `incidence()`) at time t.

Usage

```
growth_rate(fitted_model, ...)

## S3 method for class 'ps'
growth_rate(fitted_model, ...)

## S3 method for class 'rw'
growth_rate(fitted_model, ...)

## S3 method for class 'ps_single'
growth_rate(fitted_model, ...)

## S3 method for class 'rw_single'
growth_rate(fitted_model, ...)
```

Arguments

`fitted_model`	Fitted model object with class `EpiStrainDynamics.fit`
`...`	Additional arguments passed to metrics calculation

Details

Growth rate and R_t both describe transmission trends, but on different scales: growth rate is a direct log-scale rate of change, while `Rt()` additionally accounts for the generation interval to translate that rate into an average number of secondary infections per case. Growth rate is positive whenever $R_t > 1$ and negative whenever $R_t < 1$, since both describe the same underlying growth or decline.

This metric quantifies the **proportional change** in disease incidence from one time period to the next on a logarithmic scale, where:

- Positive values ($r_t > 0$) indicate exponential growth
- Negative values ($r_t < 0$) indicate exponential decline
- Values near zero ($r_t \approx 0$) indicate stable incidence
- The magnitude indicates the rate of exponential change

For example:

- A growth rate of 0.1 means incidence increased by approximately 10.5% ($e^{0.1} - 1$)
- A growth rate of -0.05 means incidence decreased by approximately 4.9%
- A growth rate of 0 means no change in incidence

This metric function can be run directly on the fitted model output.

Value

named list of class `EpiStrainDynamics.metric` containing a dataframe of the calculated metric outcome (`$measure`), the fit object (`$fit`), and the constructed model object (`$constructed_model`). The `measure` data frame contains the median of the epidemiological quantity (`y`), the 50% credible interval of the quantity (`lb_50` & `ub_50`), the 95% credible interval (`lb_95` & `ub_95`), the proportion greater than a defined threshold value (`prop`), the pathogen name (`pathogen`), and the time label (`time`).

See Also

Other metrics: `Rt()`, `incidence()`, `proportion()`

Examples

```
mod <- construct_model(
  pathogen_structure = single(
    case_timeseries = sarscov2$cases,
    time = sarscov2$date
  ),
  method = random_walk()
)

fit <- fit_model(mod)
gr <- growth_rate(fit)
```

`incidence` *Generic Method for Incidence Analysis*

Description

Computes epidemiological incidence, defined as the number of new cases occurring at a specific time point, derived by exponentiating the log-incidence estimates from the fitted model:

$$I_t = \exp(\log\text{-incidence}_t)$$

Usage

```
incidence(fitted_model, dow = NULL, ...)

## S3 method for class 'ps'
incidence(fitted_model, dow = NULL, ...)

## S3 method for class 'rw'
incidence(fitted_model, dow = NULL, ...)

## S3 method for class 'ps_single'
incidence(fitted_model, dow = NULL, ...)

## S3 method for class 'rw_single'
incidence(fitted_model, dow = NULL, ...)
```

Arguments

`fitted_model`	Fitted model object with class `EpiStrainDynamics.fit`
`dow`	Logical indicating whether to include day-of-week effects. If `NULL` or `NA` (default), uses the day-of-week setting from the fitted model. If `TRUE`, includes day-of-week effects (model must have been fitted with `dow_effect = TRUE`). If `FALSE`, excludes day-of-week effects.
`...`	Additional arguments passed to metrics calculation

Details

This metric quantifies the **absolute number of new cases** at each time point, where it is:

- Always positive (since it's an exponentiated value)
- Represents the expected case count at time t
- Can be adjusted for day-of-week effects when modeled
- Provides uncertainty quantification through posterior credible intervals

Day-of-week adjustment: When day-of-week effects are included in the model, the incidence is further adjusted as:

$$I_t^{adj} = I_t \times \text{week_effect} \times \text{dow_simplex}[\text{DOW}(t)]$$

Where:

- week_effect is the number of distinct days modelled (7 for a full weekly cycle)
- dow_simplex gives the relative reporting weight for each day of the week, estimated from the data
- $\text{DOW}(t)$ maps time t to its day of the week

This accounts for systematic variations in case reporting (e.g. lower weekend reporting) that are not part of the underlying transmission trend.

This metric function can be run directly on the fitted model output.

Value

named list of class `EpiStrainDynamics.metric` containing a dataframe of the calculated metric outcome (`$measure`), the fit object (`$fit`), and the constructed model object (`$constructed_model`). The `measure` data frame contains the median of the epidemiological quantity (`y`), the 50% credible interval of the quantity (`lb_50` & `ub_50`), the 95% credible interval (`lb_95` & `ub_95`), the proportion greater than a defined threshold value (`prop`), the pathogen name (`pathogen`), and the time label (`time`).

See Also

Other metrics: `Rt()`, `growth_rate()`, `proportion()`

Examples

```
mod <- construct_model(
  pathogen_structure = single(
    case_timeseries = sarscov2$cases,
    time = sarscov2$date
  ),
  method = random_walk()
)

fit <- fit_model(mod)

# Use model's dow setting (default)
inc <- incidence(fit)

# Explicitly exclude dow effects
inc_no_dow <- incidence(fit, dow = FALSE)

# Explicitly include dow effects (if model has them)
inc_with_dow <- incidence(fit, dow = TRUE)
```

`influenza` *World Health Organisation Global Influenza Programme for Australia*

Description

Influenza-like illness data were retrieved from the World Health Organization's Global Influenza Programme. We collated weekly data for Australia from the week starting January 2, 2012 to the week starting December 25, 2023 inclusive. The data described: (1) the weekly number of cases of influenza-like illness; and (2) the weekly number of specimens positive for influenza by subtype. We grouped the influenza specimens into: influenza A subtype not determined; influenza A H3N2; influenza A H1N1 (influenza A H1N1, influenza A H1N1pdm09); influenza B (influenza B Yamagata, influenza B Victoria, influenza B lineage not determined); and other (unknown causes which we do not have data to determine).

Usage

```
influenza
```

Format

`influenza`:
A data frame with 426 rows and 7 columns:

ili Integer, daily total number of cases of influenza-like illness

week Date, between 1 January 2012 to week starting 1 March 2020

inf_A Integer, daily number of cases of unsubtyped influenza A

inf_B Integer, daily number of cases of influenza B

inf_H3N2 Integer, daily number of cases of influenza A subtype H3N2

inf_H1N1 Integer, daily number of cases of influenza A subtype H1N1

other Integer, number of cases of unspecified influenza-like illness

Source

https://www.who.int/teams/global-influenza-programme/surveillance-and-monitoring/influenza-surveillance-outputs

---

`multiple` *Multiple pathogen structure*

Description

Multiple pathogen structure

Usage

```
multiple(data, case_timeseries, component_pathogen_timeseries, time = NULL)
```

Arguments

`data`	dataframe containing columns with all relevant data, or a time series object (ts, xts, zoo, tsibble, etc.)
`case_timeseries`	Column name containing case counts. Must be numeric or a `units` object from the **units** package.
`component_pathogen_timeseries`	vector of column names with additional pathogen case count timeseries to model. Must be numeric or a `units` object from the **units** package.
`time`	Column name with time data. Required for non-time-series input data. Flexible format - can be date, index, or others, accepted as `index` identifiers in the `tsibble` time format. Optional when `data` is a time series class object (ts, mts, xts, zoo, zooreg, tsibble) as the time index will be automatically detected.

Value

named list including pathogen_structure, pathogen_names, and data of class `EpiStrainDynamics.pathogen_structure`

See Also

Other pathogen_structure: `single()`, `subtyped()`

Examples

```
# Using a data frame
multiple(
  data = sarscov2,
  case_timeseries = "cases",
  component_pathogen_timeseries = c("alpha", "delta", "omicron", "other"),
  time = "date"
)


# Using a time series object (time argument is optional)
sarscov2_xts <- xts::xts(
  sarscov2[, c("cases", "alpha", "delta", "omicron", "other")],
  order.by = sarscov2$date
)
multiple(
  data = sarscov2_xts,
  case_timeseries = "cases",
  component_pathogen_timeseries = c("alpha", "delta", "omicron", "other")
)
```

---

p_spline *Specify p_spline method*

Description

Penalised splines is one of two optional Bayesian smoothing prior methods that can be selected and used in the model definition with `EpiStrainDynamics`. The main benefit of selecting the penalised spline method over random walks is that it can capture dynamical effects (fine enough temporal resolution) while not being too computationally expensive.

Usage

```
p_spline(spline_degree = 3, days_per_knot = 3)
```

Arguments

`spline_degree`	polynomial degree of the individual spline segments used to construct the overall curve (must be a positive whole number)
`days_per_knot`	number of days for each knot (must be a positive whole number)

Value

list with method and model parameters of class `EpiStrainDynamics.method`

See Also

Other method: `random_walk()`

Examples

```
# Valid usage
p_spline(spline_degree = 2L, days_per_knot = 5L)
p_spline(spline_degree = 3, days_per_knot = 7)

# These will produce validation errors (as intended):

# Non-positive values
try(p_spline(spline_degree = 0, days_per_knot = 5))
try(p_spline(spline_degree = 3, days_per_knot = -1))

# Non-whole numbers
try(p_spline(spline_degree = 2.5, days_per_knot = 5))
try(p_spline(spline_degree = 3, days_per_knot = 4.2))

# Non-numeric values
try(p_spline(spline_degree = "invalid", days_per_knot = 5))
```

---

plot *Plot metrics calculation outputs*

Description

S3 methods for plotting metrics calculated from the output sof either `incidence()`, `growth_rate()`, `Rt()`, or `proportion()`.

Usage

```
## S3 method for class 'incidence'
plot(x, xlab = "Time", ...)

## S3 method for class 'growth_rate'
plot(x, xlab = "Time", ...)

## S3 method for class 'Rt'
plot(x, xlab = "Time", ...)

## S3 method for class 'proportion'
plot(x, xlab = "Time", ...)
```

Arguments

x	Metrics calculation output of class EpiStrainDynamics.metric
xlab	Time label for x axis, defaults to "Time"
...	Additional arguments passed to plot

Value

ggplot2 plot output

Examples

```
mod <- construct_model(
  pathogen_structure = single(
    case_timeseries = sarscov2$cases,
    time = sarscov2$date
  ),
  method = random_walk()
)

fit <- fit_model(mod)
gr <- growth_rate(mod)
plot(gr)
```

---

print.EpiStrainDynamics.model

Print method for constructed EpiStrainDynamics models

Description

Prints a concise summary instead of dumping the full data and standata list to the console.

Usage

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

Arguments

x	an EpiStrainDynamics.model object, as returned by construct_model()
...	further arguments passed to or from other methods (unused)

Value

x, invisibly

Examples

```
mod <- construct_model(
  pathogen_structure = single(
    data = sarscov2,
    case_timeseries = "cases",
    time = "date"
  ),
  method = random_walk()
)
print(mod)
```

`proportion` *Generic Method for proportion Analysis*

Description

Computes epidemiological proportion, defined as the relative fraction of cases attributable to specific pathogen(s) or strain(s) at a given time point, calculated as the ratio of incidence from selected pathogen(s) to a reference group:

$$P_t = \frac{I_{\text{numerator}}(t)}{I_{\text{denominator}}(t)}$$

Usage

```
proportion(
  fitted_model,
  numerator_combination = NULL,
  denominator_combination = NULL,
  ...
)
```

Arguments

`fitted_model`	Fitted model object with class `EpiStrainDynamics.fit` with `multiple` or `subtyped` pathogen structure.
`numerator_combination`	Named pathogens or subtypes to be included in proportion numerator, or NULL. If NULL, it will use each pathogen.
`denominator_combination`	Named pathogens or subtypes to be included in proportion denominator, or NULL. If NULL, it will use all pathogens.
`...`	Additional arguments passed to metrics calculation

Details

Where incidences are derived from the exponential of log-incidence estimates:

$$P_t = \frac{\sum_{i \in \text{numerator}} \exp(\log\text{-incidence}_{i,t})}{\sum_{j \in \text{denominator}} \exp(\log\text{-incidence}_{j,t})}$$

Where the numerator and denominator are each a user-specified set of pathogens or subtypes (via `numerator_combination`/`denominator_combination`), and i, j index the pathogens included in each.

This metric quantifies the **relative contribution** of specific pathogen(s) or strain(s) to the total disease burden. Key characteristics:

- Values between 0 and 1 ($0 \leq P_t \leq 1$) when denominator includes numerator components
- Values can exceed 1 ($P_t > 1$) when denominator excludes numerator components
- Represents the fractional share of cases at each time point
- Time-varying to capture changing pathogen/strain dynamics

Flexible combinations:

- Individual proportions: Each pathogen relative to all pathogens (default)
- Custom numerator: Specific pathogen(s) of interest (e.g., variant of concern)
- Custom denominator: Either 'all' pathogens or a specified subset
- Subset comparisons: Compare specific groups (e.g., Alpha vs. Delta + Omicron)

This metric function can be run directly on the fitted model output.

Value

named list of class `EpiStrainDynamics.metric` containing a dataframe of the calculated metric outcome (`$measure`), the fit object (`$fit`), the constructed model object (`$constructed_model`), the resolved pathogen names used as the numerator (`$numerator_combination`), and the resolved pathogen names used as the denominator (`$denominator_combination`). When `numerator_combination` is left as the default (`NULL`), every pathogen name is listed in `$numerator_combination`, but each was computed as its own separate proportion line (one per pathogen), not summed into a single group the way `$denominator_combination` always is. The `measure` data frame contains the median of the epidemiological quantity (`y`), the 50% credible interval of the quantity (`lb_50` & `ub_50`), the 95% credible interval (`lb_95` & `ub_95`), the proportion greater than a defined threshold value (`prop`), the pathogen name (`pathogen`), and the time label (`time`).

See Also

Other metrics: `Rt()`, `growth_rate()`, `incidence()`

Examples

```
mod <- construct_model(
  pathogen_structure = multiple(
    case_timeseries = sarscov2$cases,
    time = sarscov2$date,
    component_pathogen_timeseries = list(
      alpha = sarscov2$alpha,
      delta = sarscov2$delta,
      omicron = sarscov2$omicron,
      other = sarscov2$other
    )
  ),
  method = p_spline()
)

fit <- fit_model(mod)
prop <- proportion(fit)

# or a unique combination, compared to all pathogens
prop2 <- proportion(fit,
  numerator_combination = c("alpha", "delta", "omicron")
)

# or a user-specified combination in both numerator and denominator
prop3 <- proportion(fit,
  numerator_combination = "alpha",
  denominator_combination = c("alpha", "delta", "omicron")
)
```

---

random_walk *Specify random walk method*

Description

Random walk is one of two optional Bayesian smoothing prior methods that can be selected and used in the model definition with `EpiStrainDynamics`. The random walk is a stochastic process that describes a path of a series of random steps on a mathematical space, and each next step's direction only depends on the current position, not the previous path. No additional arguments must be supplied to define the random walk method.

Usage

```
random_walk()
```

Value

list with method identified as random walk of class `EpiStrainDynamics.method`

See Also

Other method: p_spline()

Examples

```
random_walk()
```

Rt *Generic Method for Rt Analysis*

Description

Computes the time-varying reproduction number (Rt), defined as the average number of secondary infections generated by each infected individual at time t, accounting for the generation interval distribution:

$$R_t = \frac{I_t}{\sum_{k=0}^{\tau_{\max}-1} I_{t-k} \cdot \frac{g(k)}{g_a}}$$

Usage

```
Rt(fitted_model, gi_dist, tau_max = 7, ...)

## S3 method for class 'ps'
Rt(fitted_model, gi_dist, tau_max = 7, ...)

## S3 method for class 'rw'
Rt(fitted_model, gi_dist, tau_max = 7, ...)

## S3 method for class 'ps_single'
Rt(fitted_model, gi_dist, tau_max = 7, ...)

## S3 method for class 'rw_single'
Rt(fitted_model, gi_dist, tau_max = 7, ...)
```

Arguments

fitted_model	Fitted model object with class EpiStrainDynamics.fit
gi_dist	Function returning the generation interval probability for a given day. Must accept a numeric vector and return a non-negative numeric vector of the same length; does not need to sum to 1 (see Description).
tau_max	Integer maximum generation interval in days (default: 7)
...	Additional arguments passed to metrics calculation

Details

Where:

- $I_t = \exp(\log\text{-incidence}_t)$ is the current incidence
- $g(k)$ is the generation interval probability for day k
- $g_a = \sum_{k=0}^{\tau_{\max}-1} g(k)$ is the normalization constant
- $\tau_{\max}$ is the maximum generation interval in days

This metric quantifies **current transmission potential** by comparing present incidence to the weighted historical incidence that could have generated it, where:

- $R_t > 1$: Epidemic is growing (each case generates >1 secondary case)
- $R_t = 1$: Epidemic is stable (replacement level transmission)
- $R_t < 1$: Epidemic is declining (each case generates <1 secondary case)
- Threshold of 1 is epidemiologically meaningful for control decisions

Generation interval considerations:

- Accounts for transmission timing using probability distribution $g(k)$
- Recent cases contribute more to current transmission potential
- Accepts user-defined generation interval distributions, validated for structural properties (vectorized, non-negative, finite)
- Does not need to sum to 1 – normalized internally using $g_a = \sum_{k=0}^{\tau_{\max}-1} g(k)$
- Validation does not assess whether the shape is epidemiologically plausible (e.g. unimodal, decaying); as with other Rt estimation methods, supplying a distribution informed by published estimates for the pathogen(s) being modelled is the user's responsibility

This metric function can be run directly on the fitted model output.

Value

named list of class `EpiStrainDynamics.metric` containing a dataframe of the calculated metric outcome (`$measure`), the fit object (`$fit`), and the constructed model object (`$constructed_model`). The `measure` data frame contains the median of the epidemiological quantity (`y`), the 50% credible interval of the quantity (`lb_50` & `ub_50`), the 95% credible interval (`lb_95` & `ub_95`), the proportion greater than a defined threshold value (`prop`), the pathogen name (`pathogen`), and the time label (`time`).

See Also

Other metrics: `growth_rate()`, `incidence()`, `proportion()`

Examples

```
mod <- construct_model(
  pathogen_structure = single(
    case_timeseries = sarscov2$cases,
    time = sarscov2$date
  ),
  method = random_walk()
)

fit <- fit_model(mod)

rt <- Rt(fit, gi_dist = function(x) 4 * x * exp(-2 * x), tau_max = 7)
```

---

sarscov2 *United Kingdom Health Security Agency SARS-CoV-2 case data*

Description

Daily SARS-CoV-2 case numbers by specimen date for the United Kingdom from 2020 - 2022 retrieved from the UK Health Security Agency's data dashboard. downloaded data describing the daily number of variants detected by collection date. The data classified all sequences based on 'major lineage calls'. We considered groupings of the variants based on their major lineage calls or other, consisting of all lineages with a designation not consistent with any of the major lineage calls.

Usage

```
sarscov2
```

Format

`sarscov2`**:**

A data frame with 830 rows and 6 columns:

date Date, between 23 September 2020 and 31 December 2022

cases Numeric, daily number of cases

alpha Integer, daily number of cases of B.1.1.7 (Alpha variant)

delta Integer, daily number of cases of B.1.617.2 (Delta variant)

omicron Integer, daily number of cases of BA.1, BA.2, BA.2.75, BA.4, BA.5, BQ.1 (Omicron variants)

other Integer, daily number of cases of B.1.177, XBB (a recombinant of omicron sub-variants), and all other lineages with a designation not consistent with any of the major lineage calls

Source

<https://ukhsa-dashboard.data.gov.uk/covid-19-archive-data-download; https://datadryad.org/dataset/doi:10.5061/dryad.hx3

`single` *Single pathogen structure*

Description

Single pathogen structure

Usage

```
single(data, case_timeseries, time = NULL)
```

Arguments

`data`	dataframe containing columns with all relevant data, or a time series object (ts, xts, zoo, tsibble, etc.)
`case_timeseries`	Column name containing case counts. Must be numeric or a `units` object from the **units** package.
`time`	name of column with time data. Required for non-time-series input data. Flexible format - can be date, index, or others, accepted as `index` identifiers in the `tsibble` time format. Optional when `data` is a time series class object (ts, mts, xts, zoo, zooreg, tsibble) as the time index will be automatically detected.

Value

formatted list with pathogen structure and data of class `EpiStrainDynamics.pathogen_structure`.

See Also

Other pathogen_structure: `multiple()`, `subtyped()`

Examples

```
# Using a data frame
single(
  data = sarscov2,
  case_timeseries = "cases",
  time = "date"
)


# Using a time series object (time argument is optional)
sarscov2_xts <- xts::xts(sarscov2[, c("cases", "alpha")], order.by = sarscov2$date)
single(
  data = sarscov2_xts,
  case_timeseries = "cases"
)
```

`smoothing_structure` *Create Smoothing Structure Specification with Priors*

Description

This function creates a standardized smoothing structure object that specifies both the smoothing structure and associated priors for EpiStrainDynamics models. `tau` (denoted ρ in Eales et al. 2022, *Epidemics*) is the smoothing parameter that penalises how much the underlying trend's growth rate is allowed to change over time. Smaller values enforce a smoother trend; larger values allow it to bend more sharply.

Usage

```
smoothing_structure(smoothing_type = "shared", tau_mean = NULL, tau_sd = NULL)
```

Arguments

`smoothing_type`	Character string specifying the smoothing type: • `"shared"`: All pathogens have the same smoothing parameter (equivalent to `tau[1]`). By default a model with a single pathogen will have `shared` smoothing type. • `"independent"`: Independent smoothing per pathogen (equivalent to `tau[number of pathogens]`) • `"correlated"`: Correlated smoothing type (equivalent to `Sigma[number of pathogens, number of pathogens]`)
`tau_mean`	Optional numeric vector specifying the prior mean(s) for tau parameter. Can be provided for `shared` (single value) and `independent` smoothing types (can provide a single value which will be repeated for each pathogen or can provide a unique prior for each pathogen). Prior for tau for `correlated` smoothing type is not currently supported.
`tau_sd`	Numeric vector specifying the prior standard deviation(s) for tau parameter. Can be provided for `shared` (single value) and `independent` smoothing types (can provide a single value which will be repeated for each pathogen or can provide a unique prior for each pathogen). Prior for tau for `correlated` smoothing type is not currently supported.

Value

An object of class `EpiStrainDynamics.smoothing` containing:

`smoothing_type`	The specified smoothing structure type
`tau_priors`	Prior specifications for tau
`priors_provided`	Integer flag passed to the Stan model: `1` if no priors were supplied (Stan's built-in default prior is used), `2` if priors were supplied (the `tau_mean`/`tau_sd` values are used as the prior)

Examples

```
# Shared smoothing with scalar priors
shared_smooth <- smoothing_structure("shared", tau_mean = 0, tau_sd = 1)

# Independent smoothing with vector priors
indep_smooth <- smoothing_structure("independent",
  tau_mean = c(0, 0, 0),
  tau_sd = c(1, 1, 1)
)
```

---

subtyped *Subtyped pathogen structure*

Description

Subtyped pathogen structure

Usage

```
subtyped(
  data,
  case_timeseries,
  unsubtyped_timeseries,
  subtyped_timeseries,
  other_pathogen_timeseries,
  time = NULL
)
```

Arguments

`data`	dataframe containing columns with all relevant data, or a time series object (ts, xts, zoo, tsibble, etc.)
`case_timeseries`	Column name containing case counts. Must be numeric or a `units` object from the **units** package.
`unsubtyped_timeseries`	vector of column names with additional unsubtyped pathogen case count timeseries. Must be numeric or a `units` object from the **units** package.
`subtyped_timeseries`	vector of column names with additional subtyped pathogen case count timeseries. Must be numeric or a `units` object from the **units** package.
`other_pathogen_timeseries`	vector of column names with additional pathogen case count timeseries to model. Must be numeric or a `units` object from the **units** package.

`time` name of column with time data. Required for non-time-series input data. Flexible format - can be date, index, or others, accepted as `index` identifiers in the `tsibble` time format. Optional when `data` is a time series class object (ts, mts, xts, zoo, zooreg, tsibble) as the time index will be automatically detected.

Value

named list including pathogen_structure, pathogen_names, and data of class `EpiStrainDynamics.pathogen_structure`

See Also

Other pathogen_structure: `multiple()`, `single()`

Examples

```
# Using a data frame
subtyped(
  data = influenza,
  case_timeseries = "ili",
  unsubtyped_timeseries = "inf_A",
  subtyped_timeseries = c("inf_H3N2", "inf_H1N1"),
  other_pathogen_timeseries = c("inf_B", "other"),
  time = "week"
)


# Using a time series object (time argument is optional)
influenza_xts <- xts::xts(
  influenza[, c("ili", "inf_A", "inf_H3N2", "inf_H1N1", "inf_B", "other")],
  order.by = influenza$week
)
subtyped(
  data = influenza_xts,
  case_timeseries = "ili",
  unsubtyped_timeseries = "inf_A",
  subtyped_timeseries = c("inf_H3N2", "inf_H1N1"),
  other_pathogen_timeseries = c("inf_B", "other")
)
```

Index

∗ **datasets**
  influenza, 12
  sarscov2, 22
∗ **method**
  p_spline, 14
  random_walk, 19
∗ **metrics**
  growth_rate, 9
  incidence, 10
  proportion, 17
  Rt, 20
∗ **pathogen_structure**
  multiple, 13
  single, 23
  subtyped, 25

construct_model, 2
construct_model(), *16*

diagnose_model, 3
dispersion_structure, 4
dispersion_structure(), *3*

fit_model, 5
fit_model(), *4*

growth_rate, 9, *12*, *18*, *21*
growth_rate(), *15*

incidence, 10, *10*, *18*, *21*
incidence(), *9*, *15*
influenza, 12

multiple, 13, *23*, *26*
multiple(), *3*

p_spline, 14, *20*
p_spline(), *3*
plot, 15
print.EpiStrainDynamics.model, 16
proportion, *10*, *12*, 17, *21*
proportion(), *15*

random_walk, *15*, 19
random_walk(), *3*
Rt, *10*, *12*, *18*, 20
Rt(), *9*, *15*

sarscov2, 22
single, *14*, 23, *26*
single(), *3*
smoothing_structure, 24
smoothing_structure(), *3*
subtyped, *14*, *23*, 25
subtyped(), *3*