Package ‘rtestim’

July 5, 2025

Type Package

Title Estimate the Effective Reproductive Number with Trend Filtering

Version 1.0.0

Description Use trend filtering, a type of regularized nonparametric regression, to estimate the instantaneous reproduction number, also called Rt. This value roughly says how many new infections will result from each new infection today. Values larger than 1 indicate that an epidemic is growing while those less than 1 indicate decline. For more details about this methodology, see Liu, Cai, Gustafson, and McDonald (2024) <doi:10.1371/journal.pcbi.1012324>.

License MIT + file LICENSE

URL `https://github.com/dajmcdon/rtestim`,
`https://dajmcdon.github.io/rtestim/`

BugReports `https://github.com/dajmcdon/rtestim/issues`

Depends R (>= 3.6.2)

Imports checkmate, cli, dspline, ggplot2, Matrix, methods, Rcpp, rlang, tibble, tvdenoising, vctrs

Suggests dplyr, forcats, knitr, nnet, rmarkdown, testthat (>= 3.0.0), tidyr, xml2

LinkingTo BH, dspline, Rcpp, RcppEigen, testthat, tvdenoising

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

NeedsCompilation yes

Author Daniel J. McDonald [aut, cre, cph],
Jiaping Liu [aut],
Zhenglun Cai [ctb]

Maintainer Daniel J. McDonald <`daniel@stat.ubc.ca`>

Repository CRAN

Date/Publication 2025-07-05 18:50:02 UTC

Contents

`cancovid` *Canadian Incident COVID-19 Cases*

Description

This dataset contains 3+ years of incident COVID-19 case counts as reported by opencovid.ca as of July 4, 2023.

Usage

```
cancovid
```

Format

A data frame with 1,253 rows and 2 columns:

date The observed date. A `Date` object.

incident_cases The number of new recorded cases for this date.

Source

This data is available under the CC-BY-4.0 License.

See:

Berry, I., O'Neill, M., Sturrock, S. L., Wright, J. E., Acharya, K., Brankston, G., Harish, V., Kornas, K., Maani, N., Naganathan, T., Obress, L., Rossi, T., Simmons, A. E., Van Camp, M., Xie, X., Tuite, A. R., Greer, A. L., Fisman, D. N., & Soucy, J.-P. R. (2021). A sub-national real-time epidemiological and vaccination database for the COVID-19 pandemic in Canada. Scientific Data, 8(1). doi:10.1038/s41597021009552

`confband` *Add confidence bands to estimated Rt or incidence curves*

Description

Create an approximate confidence band for the Rt or incidence estimate. Note that the variance computation is approximate.

Usage

```
confband(object, lambda, level = 0.95, type = c("Rt", "Yt"), ...)
```

Arguments

`object`	a `poisson_rt` or `cv_poisson_rt` object.
`lambda`	the selected lambda. May be a scalar value, or in the case of `cv_poisson_rt` objects, `"lambda.min"` or `"lambda.max"`.
`level`	the desired confidence level(s). These will be sorted if necessary.
`type`	the type `Rt` or `Yt` for confidence intervals of fitted Rt or fitted incident cases
`...`	additional arguments for methods. Unused.

Value

A `data.frame` containing the estimates `Rt` or `Yt` at the chosen `lambda`, and confidence limits corresponding to `level`

Examples

```
y <- c(1, rpois(100, dnorm(1:100, 50, 15) * 500 + 1))
out <- estimate_rt(y, nsol = 10)
head(confband(out, out$lambda[2]))
head(confband(out, out$lambda[2], level = c(0.95, 0.8, 0.5)))

cv <- cv_estimate_rt(y, nfold = 3, nsol = 30)
head(confband(cv, "lambda.min", c(0.5, 0.9)))
```

`configure_rt_admm` *Rt estimation algorithm configuration*

Description

Rt estimation algorithm configuration

Usage

```
configure_rt_admm(
  rho = -1,
  alpha = 0.5,
  gamma = 0.9,
  tolerance = 1e-04,
  maxiter_newton = 50L,
  maxiter_line = 20L,
  verbose = 0,
  ...
)
```

Arguments

`rho`	Double. An ADMM parameter; coefficient of augmented term in the Lagrangian function.
`alpha`	Double. A parameter adjusting upper bound in line search algorithm in `prox_newton` algorithm.
`gamma`	Double. A parameter adjusting step size in line search algorithm in `prox_newton` algorithm.
`tolerance`	Double. Tolerance of ADMM convergence.
`maxiter_newton`	Integer. Maximum number of iterations for the outer Newton iteration.
`maxiter_line`	Integer. Maximum number of iterations for the linesearch algorithm in the proximal Newton method.
`verbose`	Integer.
`...`	space for future extensions

Value

a list of model parameters with class `rt_admm_configuration`

Examples

```
configure_rt_admm()
configure_rt_admm(tolerance = 1e-6, verbose = 1L)
```

`cv_estimate_rt` *Leave-kth-out cross validation for choosing a optimal parameter lambda*

Description

Leave-kth-out cross validation for choosing a optimal parameter lambda

Usage

```
cv_estimate_rt(
  observed_counts,
  korder = 3L,
  dist_gamma = c(2.5, 2.5),
  nfold = 3L,
  error_measure = c("deviance", "mse", "mae"),
  x = 1:n,
  lambda = NULL,
  maxiter = 1000000L,
  delay_distn = NULL,
  delay_distn_periodicity = NULL,
  regular_splits = FALSE,
  invert_splits = FALSE,
  ...
)
```

Arguments

`observed_counts`	vector of the observed daily infection counts
`korder`	Integer. Degree of the piecewise polynomial curve to be estimated. For example, `korder = 0` corresponds to a piecewise constant curve.
`dist_gamma`	Vector of length 2. These are the shape and scale for the assumed serial interval distribution. Roughly, this distribution describes the probability of an infectious individual infecting someone else after some period of time after having become infectious. As in most literature, we assume that this interval follows a gamma distribution with some shape and scale.
`nfold`	Integer. This number of folds to conduct the leave-kth-out cross validation. For leave-kth-out cross validation, every kth observed_counts and their corresponding position (evenly or unevenly spaced) are placed into the same fold. The first and last observed_counts are not assigned to any folds. Smallest allowable value is `nfold = 2`.
`error_measure`	Metric used to calculate cross validation scores. Must be choose from `mse`, `mae`, and `deviance`. `mse` calculates the mean square error; `mae` calculates the mean absolute error; `deviance` calculates the deviance

`x`	a vector of positions at which the counts have been observed. In an ideal case, we would observe data at regular intervals (e.g. daily or weekly) but this may not always be the case. May be numeric or Date.
`lambda`	Vector. A user supplied sequence of tuning parameters which determines the balance between data fidelity and smoothness of the estimated Rt; larger `lambda` results in a smoother estimate. The default, `NULL` results in an automatic computation based on `nlambda`, the largest value of `lambda` that would result in a maximally smooth estimate, and `lambda_min_ratio`. Supplying a value of `lambda` overrides this behaviour. It is likely better to supply a decreasing sequence of `lambda` values than a single (small) value. If supplied, the user-defined `lambda` sequence is automatically sorted in decreasing order.
`maxiter`	Integer. Maximum number of iterations for the estimation algorithm.
`delay_distn`	in the case of a non-gamma delay distribution, a vector or matrix (or `Matrix::Matrix()`) of delay probabilities may be passed here. For a vector, these will be coerced to sum to 1, and padded with 0 in the right tail if necessary. If a time-varying delay matrix, it must be lower-triangular. Each row will be silently coerced to sum to 1. See also `vignette("delay-distributions")`.
`delay_distn_periodicity`	Controls the relationship between the spacing of the computed delay distribution and the spacing of `x`. In the default case, `x` would be regular on the sequence `1:length(observed_cases)`, and this would be 1. But if `x` is a `Date` object or spaced irregularly, the relationship becomes more complicated. For example, weekly data when `x` is a date in the form `YYYY-MM-DD` requires specifying `delay_distn_periodicity = "1 week"`. Or if `observed_cases` were reported on Monday, Wednesday, and Friday, then `delay_distn_periodicity = "1 day"` would be most appropriate.
`regular_splits`	Logical. If `TRUE`, the folds for k-fold cross-validation are chosen by placing every kth point into the same fold. The first and last points are not included in any fold and are always included in building the predictive model. As an example, with 15 data points and `kfold = 4`, the points are assigned to folds in the following way: $$0\ 1\ 2\ 3\ 4\ 1\ 2\ 3\ 4\ 1\ 2\ 3\ 4\ 1\ 0$$ where 0 indicates no assignment. Therefore, the folds are not random and running `cv_estimate_rt()` twice will give the same result.
`invert_splits`	Logical. Typical K-fold CV would use K-1 folds for the training set while reserving 1 fold for evaluation (repeating the split K times). Setting this to true inverts this process, using a much smaller training set with a larger evaluation set. This tends to result in larger values of `lambda` that minimize CV.
`...`	additional parameters passed to `estimate_rt()` function

Value

An object with S3 class `"cv_poisson_rt"`. Among the list components:

- `full_fit` An object with S3 class `"poisson_rt"`, fitted with all `observed_counts` and `lambda`

- `cv_scores` leave-kth-out cross validation scores
- `cv_se` leave-kth-out cross validation standard error
- `lambda.min` lambda which achieved the optimal cross validation score
- `lambda.1se` lambda that gives the optimal cross validation score within one standard error.
- `lambda` the value of `lambda` used in the algorithm.

Examples

```
y <- c(1, rpois(100, dnorm(1:100, 50, 15) * 500 + 1))
cv <- cv_estimate_rt(y, korder = 3, nfold = 3, nsol = 30)
cv
```

`delay_calculator` *Calculate the total infectiousness at each observed time point.*

Description

The total infectiousness at each observed time point is calculated by $\sum_{s=1}^{t} I_{t-s} w_s$, where I denotes the vector containing observed incidence, and w denotes the generation interval distribution. Typically, the generation interval is challenging to estimate from data, so the serial interval is used instead. The serial interval distribution expresses the probability of a secondary infection caused by a primary infection which occurred s days earlier.

Usage

```
delay_calculator(
  observed_counts,
  x = NULL,
  dist_gamma = c(2.5, 2.5),
  delay_distn = NULL,
  delay_distn_periodicity = NULL,
  xout = x
)
```

Arguments

`observed_counts`	vector of the observed daily infection counts
`x`	a vector of positions at which the counts have been observed. In an ideal case, we would observe data at regular intervals (e.g. daily or weekly) but this may not always be the case. May be numeric or Date.
`dist_gamma`	Vector of length 2. These are the shape and scale for the assumed serial interval distribution. Roughly, this distribution describes the probability of an infectious individual infecting someone else after some period of time after having become infectious. As in most literature, we assume that this interval follows a gamma distribution with some shape and scale.

`delay_distn`	in the case of a non-gamma delay distribution, a vector or matrix (or `Matrix::Matrix()`) of delay probabilities may be passed here. For a vector, these will be coerced to sum to 1, and padded with 0 in the right tail if necessary. If a time-varying delay matrix, it must be lower-triangular. Each row will be silently coerced to sum to 1. See also `vignette("delay-distributions")`.
`delay_distn_periodicity`	Controls the relationship between the spacing of the computed delay distribution and the spacing of `x`. In the default case, `x` would be regular on the sequence `1:length(observed_cases)`, and this would be 1. But if `x` is a `Date` object or spaced irregularly, the relationship becomes more complicated. For example, weekly data when `x` is a date in the form `YYYY-MM-DD` requires specifying `delay_distn_periodicity = "1 week"`. Or if `observed_cases` were reported on Monday, Wednesday, and Friday, then `delay_distn_periodicity = "1 day"` would be most appropriate.
`xout`	a vector of positions at which the results should be returned. By default, this will be the same as `x`, but in the case that observations are unequally spaced, alternatives may be desired. Note that `xout` must satisfy `min(x) <= min(xout)` and `max(x) >= max(xout)`.

Value

A vector containing the total infectiousness at each point `xout`.

Examples

```
delay_calculator(c(3, 2, 5, 3, 1), dist_gamma = c(2.5, 2.5))
```

`discretize_gamma` *Compute the discretized density function for gamma distribution*

Description

The serial interval distribution expresses the probability of the symptom onset of a secondary infection occurred a given number of days after the primary infection. The serial interval distribution is commonly represented by a discretized Gamma distribution in literature, parametrized by the shape and scale parameters.

Usage

```
discretize_gamma(x, shape = 2.5, scale = 2.5, rate = 1/scale)
```

Arguments

`x`	locations (times) where cases are observed. Must be nonnegative.
`shape, scale`	shape and scale parameters. Must be positive, `scale` strictly.
`rate`	an alternative way to specify the scale.

Value

probability mass of the discretized gamma distribution

Examples

```
discretize_gamma(1:30, shape = 1, scale = 1)
```

`estimate_rt` *Estimate Rt using smoothness-penalized Poisson likelihood*

Description

The Effective Reproduction Number R_t of an infectious disease can be estimated by solving the smoothness penalized Poisson regression (trend filtering) of the form:

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} (w_i e^{\theta_i} - y_i \theta_i) + \lambda \| D^{(k+1)} \theta \|_1,$$

where $R_t = e^{\theta}$, y_i is the observed case count at day i, w_i is the weighted past counts at day i, λ is the smoothness penalty, and $D^{(k+1)}$ is the $(k+1)$-th order difference matrix.

Usage

```
estimate_rt(
  observed_counts,
  korder = 3L,
  dist_gamma = c(2.5, 2.5),
  x = 1:n,
  lambda = NULL,
  nsol = 50L,
  delay_distn = NULL,
  delay_distn_periodicity = NULL,
  lambdamin = NULL,
  lambdamax = NULL,
  lambda_min_ratio = 1e-04,
  maxiter = 1e+05,
  init = configure_rt_admm()
)
```

Arguments

`observed_counts`	vector of the observed daily infection counts
`korder`	Integer. Degree of the piecewise polynomial curve to be estimated. For example, `korder = 0` corresponds to a piecewise constant curve.

`dist_gamma`	Vector of length 2. These are the shape and scale for the assumed serial interval distribution. Roughly, this distribution describes the probability of an infectious individual infecting someone else after some period of time after having become infectious. As in most literature, we assume that this interval follows a gamma distribution with some shape and scale.
`x`	a vector of positions at which the counts have been observed. In an ideal case, we would observe data at regular intervals (e.g. daily or weekly) but this may not always be the case. May be numeric or Date.
`lambda`	Vector. A user supplied sequence of tuning parameters which determines the balance between data fidelity and smoothness of the estimated Rt; larger `lambda` results in a smoother estimate. The default, `NULL` results in an automatic computation based on `nlambda`, the largest value of `lambda` that would result in a maximally smooth estimate, and `lambda_min_ratio`. Supplying a value of `lambda` overrides this behaviour. It is likely better to supply a decreasing sequence of `lambda` values than a single (small) value. If supplied, the user-defined `lambda` sequence is automatically sorted in decreasing order.
`nsol`	Integer. The number of tuning parameters `lambda` at which to compute Rt.
`delay_distn`	in the case of a non-gamma delay distribution, a vector or matrix (or `Matrix::Matrix()`) of delay probabilities may be passed here. For a vector, these will be coerced to sum to 1, and padded with 0 in the right tail if necessary. If a time-varying delay matrix, it must be lower-triangular. Each row will be silently coerced to sum to 1. See also `vignette("delay-distributions")`.
`delay_distn_periodicity`	Controls the relationship between the spacing of the computed delay distribution and the spacing of `x`. In the default case, `x` would be regular on the sequence `1:length(observed_cases)`, and this would be 1. But if `x` is a `Date` object or spaced irregularly, the relationship becomes more complicated. For example, weekly data when `x` is a date in the form `YYYY-MM-DD` requires specifying `delay_distn_periodicity = "1 week"`. Or if `observed_cases` were reported on Monday, Wednesday, and Friday, then `delay_distn_periodicity = "1 day"` would be most appropriate.
`lambdamin`	Optional value for the smallest `lambda` to use. This should be greater than zero.
`lambdamax`	Optional value for the largest `lambda` to use.
`lambda_min_ratio`	If neither `lambda` nor `lambdamin` is specified, the program will generate a lambdamin by lambdamax * lambda_min_ratio. A multiplicative factor for the minimal lambda in the `lambda` sequence, where `lambdamin = lambda_min_ratio * lambdamax`. A very small value will lead to the solution `Rt = log(observed_counts)`. This argument has no effect if there is a user-defined `lambda` sequence.
`maxiter`	Integer. Maximum number of iterations for the estimation algorithm.
`init`	a list of internal configuration parameters of class `rt_admm_configuration`.

Value

An object with S3 class `poisson_rt`. Among the list components:

- `observed_counts` the observed daily infection counts.

- `x` a vector of positions at which the counts have been observed.
- `weighted_past_counts` the weighted sum of past infection counts.
- `Rt` the estimated effective reproduction rate. This is a matrix with each column corresponding to one value of `lambda`.
- `lambda` the values of `lambda` actually used in the algorithm.
- `korder` degree of the estimated piecewise polynomial curve.
- `dof` degrees of freedom of the estimated trend filtering problem.
- `niter` the required number of iterations for each value of `lambda`.
- `convergence` if number of iterations for each value of `lambda` is less than the maximum number of iterations for the estimation algorithm.

Examples

```
y <- c(1, rpois(100, dnorm(1:100, 50, 15) * 500 + 1))
out <- estimate_rt(y)
out
plot(out)

out0 <- estimate_rt(y, korder = 0L, nsol = 40)
out0
plot(out0)
```

`fitted.cv_poisson_rt` *Fitted cv_poisson_rt*

Description

Fitted cv_poisson_rt

Usage

```
## S3 method for class 'cv_poisson_rt'
fitted(object, which_lambda = c("lambda.min", "lambda.1se"), ...)
```

Arguments

`object`	result of cross validation of type `cv_poisson_rt`
`which_lambda`	select which Rt's to output. If not provided, all Rt's are returned. If provided a list of lambda,the corresponding Rt estimation will be returned. If provided a string, it must be either one of `lambda.min` or `lambda.1se`. • If provided `lambda.min`, return Rt which is generated from the lambda that minimizes the cross validation score. • If provided `lambda.1se`, return Rt which is generated from the lambda whose corresponding cross validation score is 1 standard error away of the minimal cross validation score.
`...`	not used.

Value

Rt's estimated from provided lambda

Examples

```
y <- c(1, rpois(100, dnorm(1:100, 50, 15) * 500 + 1))
cv <- cv_estimate_rt(y, korder = 3, nfold = 3, nsol = 30)
f <- fitted(cv)
f <- fitted(cv, which_lambda = cv$lambda[1])
f <- fitted(cv, which_lambda = "lambda.1se")
f <- fitted(cv, which_lambda = NULL)
```

`interpolate_rt` *Interpolate (or extrapolate) Rt estimates to intermediate design points*

Description

Interpolate (or extrapolate) Rt estimates to intermediate design points

Usage

```
interpolate_rt(object, xout, ...)

## S3 method for class 'cv_poisson_rt'
interpolate_rt(object, xout, which_lambda = c("lambda.min", "lambda.1se"), ...)

## S3 method for class 'poisson_rt'
interpolate_rt(object, xout, lambda = NULL, ...)
```

Arguments

`object`	A fitted object produced by `estimate_rt()` or `cv_estimate_rt()`.
`xout`	a vector of new positions at which Rt should be produced, but where counts may not have been observed.
`...`	additional arguments passed to methods.
`which_lambda`	Select which lambdas from the object to use. If not provided, all Rt's are returned. Note that new lambdas not originally used in the estimation procedure may be provided, but the results will be calculated by linearly interpolating the estimated Rt's. The strings `lambda.min` or `lambda.1se` are allowed to choose either the lambda that minimizes the cross validation score or the largest lambda whose corresponding cross validation score is within 1 standard error of the minimal cross validation score.

`lambda`	Vector. A user supplied sequence of tuning parameters which determines the balance between data fidelity and smoothness of the estimated Rt; larger `lambda` results in a smoother estimate. The default, `NULL` results in an automatic computation based on `nlambda`, the largest value of `lambda` that would result in a maximally smooth estimate, and `lambda_min_ratio`. Supplying a value of `lambda` overrides this behaviour. It is likely better to supply a decreasing sequence of `lambda` values than a single (small) value. If supplied, the user-defined `lambda` sequence is automatically sorted in decreasing order.

Value

A vector or matrix of interpolated Rt estimates.

Examples

```
y <- c(1, rpois(100, dnorm(1:100, 50, 15) * 500 + 1))
out <- estimate_rt(y)

# originally estimated at
out$x

# get the Rt at 3 new points (for all estimated lambdas)
int <- interpolate_rt(out, c(10.5, 11.5, 12.5))

# get the Rt at a single value of lambda
interpolate_rt(out, c(10.5, 11.5, 12.5), lambda = out$lambda[20])

y <- c(1, rpois(100, dnorm(1:100, 50, 15) * 500 + 1))
out <- estimate_rt(y, nsol = 10)
interpolate_rt(out, xout = c(1.5, 2.5))
```

`plot.cv_poisson_rt` *Plot cv_poisson_rt*

Description

Plot cv_poisson_rt

Usage

```
## S3 method for class 'cv_poisson_rt'
plot(x, which_lambda = c("cv_scores", "lambda.min", "lambda.1se"), ...)
```

Arguments

`x`	result of cv_estimate_rt of class `cv_poisson_rt`

`which_lambda`	select which Rt's to plot. If not provided, the cross validation score will be plotted. If provided a list of lambda, the corresponding Rt estimation will be plotted. If provided a string, it must be either one of `lambda.min`, `lambda.1se`, or `cv_scores`. • If provided `lambda.min`, plot Rt which is generated from the lambda that minimizes the cross validation score. • If provided `lambda.1se`, plot Rt which is generated from the lambda whose corresponding cross validation score is 1 standard error away of the minimal cross validation score. • If provided `cv_scores`, plot the cross validation score. • If NULL, all estimated Rt values are plotted.
`...`	Not used.

Value

a ggplot2::ggplot

Examples

```
y <- c(1, rpois(100, dnorm(1:100, 50, 15) * 500 + 1))
cv <- cv_estimate_rt(y, korder = 1, nfold = 3, nsol = 30)
plot(cv)
plot(cv, which_lambda = cv$lambda[1])
plot(cv, which_lambda = "lambda.min")
plot(cv, which_lambda = "lambda.1se")
plot(cv, NULL)
```

`plot.poisson_rt` *Plot estimated Rt values from a* `poisson_rt` *object*

Description

Produces a figure showing some or all estimated Rt values for different values of the penalty. The result is a `ggplot2::ggplot()`. Additional user modifications can be added as desired.

Usage

```
## S3 method for class 'poisson_rt'
plot(x, lambda = NULL, ...)
```

Arguments

`x`	output of the function `estimate_rt()` of class `poisson_rt`
`lambda`	select which Rt's to plot. If not provided, all Rt's are plotted.
`...`	Not used.

Value

a ggplot2::ggplot

Examples

```
y <- c(1, rpois(100, dnorm(1:100, 50, 15) * 500 + 1))
out <- estimate_rt(y, lambda = log(c(1.1, 1.3, 1.5)))
plot(out)
```

`plot.rt_confidence_band` *Plot estimated confidence bands for an estimate of Rt*

Description

Produces a figure showing a single estimated Rt value along with approximate confidence bands. The result is a `ggplot2::ggplot()`. Additional user modifications can be added as desired.

Usage

```
## S3 method for class 'rt_confidence_band'
plot(x, colour = "#3A448F", ...)
```

Arguments

`x`	An object of class `rt_confidence_band` as produced by `confband()`.
`colour`	The colour of the desired plot
`...`	Not used.

Value

A `ggplot2::ggplot()`.

Examples

```
y <- c(1, rpois(100, dnorm(1:100, 50, 15) * 500 + 1))
out <- estimate_rt(y, nsol = 10)
cb <- confband(out, out$lambda[2], level = c(0.95, 0.8, 0.5))
plot(cb)
cb_y <- confband(out, out$lambda[2], level = c(0.95, 0.8, 0.5), type = "Yt")
plot(cb_y)
```

`predict.cv_poisson_rt` *Predict observed data using estimated Rt*

Description

Given an object of class `poisson_rt` produced with `estimate_rt()`, calculate predicted observed cases for the estimated Rt values. Note: This function is not intended for "new x" or to produce forecasts, but rather to examine how Rt relates to observables.

Usage

```
## S3 method for class 'cv_poisson_rt'
predict(object, which_lambda = c("lambda.min", "lambda.1se"), ...)
```

Arguments

`object`	result of cross validation of type `cv_poisson_rt`
`which_lambda`	Select which lambdas from the object to use. If not provided, all Rt's are returned. Note that new lambdas not originally used in the estimation procedure may be provided, but the results will be calculated by linearly interpolating the estimated Rt's. The strings `lambda.min` or `lambda.1se` are allowed to choose either the lambda that minimizes the cross validation score or the largest lambda whose corresponding cross validation score is within 1 standard error of the minimal cross validation score.
`...`	not used.

Value

A vector or matrix of predicted case counts.

Examples

```
y <- c(1, rpois(100, dnorm(1:100, 50, 15) * 500 + 1))
cv <- cv_estimate_rt(y, korder = 3, nfold = 3, nsol = 30)
p <- predict(cv)
p <- predict(cv, which_lambda = cv$lambda[1])
p <- predict(cv, which_lambda = "lambda.1se")
p <- predict(cv, which_lambda = NULL)
plot(y)
matlines(p, lty = 2)
```

`predict.poisson_rt` *Predict observed data using estimated Rt*

Description

Given an object of class `poisson_rt` produced with `estimate_rt()`, calculate predicted observed cases for the estimated Rt values. Note: This function is not intended for "new x" or to produce forecasts, but rather to examine how Rt relates to observables.

Usage

```
## S3 method for class 'poisson_rt'
predict(object, lambda = NULL, ...)
```

Arguments

`object`	An object of class `poisson_rt` produced with `estimate_rt()`.
`lambda`	Select which lambdas from the object to use. If not provided (the default), all are returned. Note that new lambdas not originally used in the estimation procedure may be provided, but the results will be calculated by linearly interpolating the estimated Rt's.
`...`	Not used.

Value

A vector or matrix of predicted case counts.

Examples

```
y <- c(1, rpois(100, dnorm(1:100, 50, 15) * 500 + 1))
out <- estimate_rt(y, nsol = 10)
preds <- predict(out)
plot(y)
matlines(preds, lty = 1)
```

Index