Package ‘Rnaught’

September 2, 2025

Type Package

Title Estimate the Basic Reproduction Number (R0)

Version 0.1.0

Description A collection of methods for estimating the basic reproduction number (R0) of infectious diseases. Features a web application to interface with the estimators. Uses the models from:
Fisman et al. (2013) <DOI:10.1371/journal.pone.0083622>,
Bettencourt and Ribeiro (2008) <DOI:10.1371/journal.pone.0002185>, and
White and Pagano (2008) <DOI:10.1002/sim.3136>.
Includes datasets for Canadian national and provincial COVID-19 case counts provided by Berry et al. (2021) <DOI:10.1038/s41597-021-00955-2>.

License AGPL (>= 3)

URL `https://mi2yorku.github.io/Rnaught/`,
`https://github.com/MI2YorkU/Rnaught`

BugReports `https://github.com/MI2YorkU/Rnaught/issues`

Imports stats, utils

Suggests knitr, rmarkdown, shiny, bslib, DT, plotly, testthat (>= 3.0.0)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Depends R (>= 2.10)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Naeem Model [aut, cph, cre],
Sawitree Boonpatcharanon [aut, cph],
Jane Heffernan [aut, cph],
Hanna Jankowski [aut, cph],
Tatiana Krikella [aut, cph],
Kseniia Lipikhin [ctb, cph]

Maintainer Naeem Model <me@nmode.ca>

Repository CRAN

Date/Publication 2025-09-02 21:00:07 UTC

Contents

COVIDCanada *COVID-19 Canada National Case Counts, 2020-2023*

Description

Daily national COVID-19 case counts in Canada, from the start of the pandemic until the end of 2023. Retrieved from the COVID-19 Canada Open Data Working Group on 2024-05-11.

Usage

```
COVIDCanada
```

Format

A data frame with 1439 observations on 3 variables:

date The date of reporting in YYYY-MM-DD format.

cases The daily number of cases.

cumulative_cases The cumulative number of cases.

Source

https://github.com/ccodwg/CovidTimelineCanada

`COVIDCanadaPT` *COVID-19 Canada Provincial and Territorial Case Counts, 2020-2023*

Description

Daily COVID-19 case counts for each Canadian province and territory, from the start of the pandemic until the end of 2023. Retrieved from the the COVID-19 Canada Open Data Working Group on 2024-05-11.

Usage

```
COVIDCanadaPT
```

Format

A data frame with 16799 observations on 4 variables:

region The two-letter code for the Canadian province or territory.

date The date of reporting in YYYY-MM-DD format.

cases The daily number of cases.

cumulative_cases The cumulative number of cases.

Source

`https://github.com/ccodwg/CovidTimelineCanada`

`id` *Incidence Decay (ID)*

Description

This function implements a least squares estimation method of R0 due to Fisman et al. (PloS One, 2013). See details for implementation notes.

Usage

```
id(cases, mu)
```

Arguments

`cases` Vector of case counts. The vector must be non-empty and only contain positive integers.

`mu` Mean of the serial distribution. This must be a positive number. The value should match the case counts in time units. For example, if case counts are weekly and the serial distribution has a mean of seven days, then `mu` should be set to 1. If case counts are daily and the serial distribution has a mean of seven days, then `mu` should be set to 7.

Details

The method is based on a straightforward incidence decay model. The estimate of R0 is the value which minimizes the sum of squares between observed case counts and cases counts expected under the model.

This method is based on an approximation of the SIR model, which is most valid at the beginning of an epidemic. The method assumes that the mean of the serial distribution (sometimes called the serial interval) is known. The final estimate can be quite sensitive to this value, so sensitivity testing is strongly recommended. Users should be careful about units of time (e.g., are counts observed daily or weekly?) when implementing.

Value

An estimate of the basic reproduction number (R0).

References

Fisman et al. (PloS One, 2013) doi:10.1371/journal.pone.0083622

See Also

`idea()` for a similar method.

Examples

```
# Weekly data.
cases <- c(1, 4, 10, 5, 3, 4, 19, 3, 3, 14, 4)

# Obtain R0 when the serial distribution has a mean of five days.
id(cases, mu = 5 / 7)

# Obtain R0 when the serial distribution has a mean of three days.
id(cases, mu = 3 / 7)
```

`idea` *Incidence Decay and Exponential Adjustment (IDEA)*

Description

This function implements a least squares estimation method of R0 due to Fisman et al. (PloS One, 2013). See details for implementation notes.

Usage

```
idea(cases, mu)
```

Arguments

`cases`	Vector of case counts. The vector must be of length at least two and only contain positive integers.
`mu`	Mean of the serial distribution. This must be a positive number. The value should match the case counts in time units. For example, if case counts are weekly and the serial distribution has a mean of seven days, then `mu` should be set to `1`. If case counts are daily and the serial distribution has a mean of seven days, then `mu` should be set to `7`.

Details

This method is closely related to that implemented in `id()`. The method is based on an incidence decay model. The estimate of R0 is the value which minimizes the sum of squares between observed case counts and case counts expected under the model.

This method is based on an approximation of the SIR model, which is most valid at the beginning of an epidemic. The method assumes that the mean of the serial distribution (sometimes called the serial interval) is known. The final estimate can be quite sensitive to this value, so sensitivity testing is strongly recommended. Users should be careful about units of time (e.g., are counts observed daily or weekly?) when implementing.

Value

An estimate of the basic reproduction number (R0).

References

Fisman et al. (PloS One, 2013) doi:10.1371/journal.pone.0083622

See Also

`id()` for a similar method.

Examples

```
# Weekly data.
cases <- c(1, 4, 10, 5, 3, 4, 19, 3, 3, 14, 4)

# Obtain R0 when the serial distribution has a mean of five days.
idea(cases, mu = 5 / 7)

# Obtain R0 when the serial distribution has a mean of three days.
idea(cases, mu = 3 / 7)
```

`seq_bayes` *Sequential Bayes (seqB)*

Description

This function implements a sequential Bayesian estimation method of R0 due to Bettencourt and Ribeiro (PloS One, 2008). See details for important implementation notes.

Usage

```
seq_bayes(cases, mu, kappa = 20, post = FALSE)
```

Arguments

`cases` Vector of case counts. The vector must only contain non-negative integers, and have at least two positive integers.

`mu` Mean of the serial distribution. This must be a positive number. The value should match the case counts in time units. For example, if case counts are weekly and the serial distribution has a mean of seven days, then `mu` should be set to `1`. If case counts are daily and the serial distribution has a mean of seven days, then `mu` should be set to `7`.

`kappa` Largest possible value of the uniform prior (defaults to `20`). This must be a number greater than or equal to `1`. It describes the prior belief on the ranges of R0, and should be set to a higher value if R0 is believed to be larger.

`post` Whether to return the posterior distribution of R0 instead of the estimate of R0 (defaults to `FALSE`). This must be a value identical to `TRUE` or `FALSE`.

Details

The method sets a uniform prior distribution on R0 with possible values between `0` and `kappa`, discretized to a fine grid. The distribution of R0 is then updated sequentially, with one update for each new case count observation. The final estimate of R0 is the mean of the (last) posterior distribution. The prior distribution is the initial belief of the distribution of R0, which is the uninformative uniform distribution with values between `0` and `kappa`. Users can change the value of `kappa` only (i.e., the prior distribution cannot be changed from the uniform). As more case counts are observed, the influence of the prior distribution should lessen on the final estimate.

This method is based on an approximation of the SIR model, which is most valid at the beginning of an epidemic. The method assumes that the mean of the serial distribution (sometimes called the serial interval) is known. The final estimate can be quite sensitive to this value, so sensitivity testing is strongly recommended. Users should be careful about units of time (e.g., are counts observed daily or weekly?) when implementing.

Our code has been modified to provide an estimate even if case counts equal to zero are present in some time intervals. This is done by grouping the counts over such periods of time. Without grouping, and in the presence of zero counts, no estimate can be provided.

Value

If post is identical to TRUE, a list containing the following components is returned:

- supp - the support of the posterior distribution of R0
- pmf - the probability mass function of the posterior distribution of R0

Otherwise, if post is identical to FALSE, only the estimate of R0 is returned. Note that the estimate is equal to sum(supp * pmf) (i.e., the posterior mean).

References

Bettencourt and Ribeiro (PloS One, 2008) doi:10.1371/journal.pone.0002185

See Also

vignette("seq_bayes_post", package = "Rnaught") for examples of using the posterior distribution.

Examples

```
# Weekly data.
cases <- c(1, 4, 10, 5, 3, 4, 19, 3, 3, 14, 4)

# Obtain R0 when the serial distribution has a mean of five days.
seq_bayes(cases, mu = 5 / 7)

# Obtain R0 when the serial distribution has a mean of three days.
seq_bayes(cases, mu = 3 / 7)

# Obtain R0 when the serial distribution has a mean of seven days, and R0 is
# believed to be at most 4.
estimate <- seq_bayes(cases, mu = 1, kappa = 4)

# Same as above, but return the posterior distribution of R0 instead of the
# estimate.
posterior <- seq_bayes(cases, mu = 1, kappa = 4, post = TRUE)

# Display the support and probability mass function of the posterior.
posterior$supp
posterior$pmf

# Note that the following always holds:
estimate == sum(posterior$supp * posterior$pmf)
```

web *Launch the Rnaught Web Application*

Description

This is the entry point of the Rnaught web application, which creates and returns a Shiny app object. When invoked directly, the web application is launched.

Usage

```
web()
```

Details

The following dependencies are required to run the application:

- shiny
- bslib
- DT
- plotly

If any of the above packages are missing during launch, a prompt will appear to install them.

To configure settings such as the port, host or default browser, set Shiny's global options (see `shiny::runApp()`).

Value

A Shiny app object for the Rnaught web application.

wp *White and Pagano (WP)*

Description

This function implements an R0 estimation due to White and Pagano (Statistics in Medicine, 2008). The method is based on maximum likelihood estimation in a Poisson transmission model. See details for important implementation notes.

Usage

```
wp(
  cases,
  mu = NA,
  serial = FALSE,
  grid_length = 100,
  max_shape = 10,
  max_scale = 10
)
```

Arguments

`cases` Vector of case counts. The vector must be of length at least two and only contain positive integers.

`mu` Mean of the serial distribution (defaults to `NA`). This must be a positive number or `NA`. If a number is specified, the value should match the case counts in time units. For example, if case counts are weekly and the serial distribution has a mean of seven days, then `mu` should be set to `1`. If case counts are daily and the serial distribution has a mean of seven days, then `mu` should be set to `7`.

`serial` Whether to return the estimated serial distribution in addition to the estimate of R0 (defaults to `FALSE`). This must be a value identical to `TRUE` or `FALSE`.

`grid_length` The length of the grid in the grid search (defaults to 100). This must be a positive integer. It will only be used if `mu` is set to `NA`. The grid search will go through all combinations of the shape and scale parameters for the gamma distribution, which are `grid_length` evenly spaced values from `0` (exclusive) to `max_shape` and `max_scale` (inclusive), respectively. Note that larger values will result in a longer search time.

`max_shape` The largest possible value of the shape parameter in the grid search (defaults to 10). This must be a positive number. It will only be used if `mu` is set to `NA`. Note that larger values will result in a longer search time, and may cause numerical instabilities.

`max_scale` The largest possible value of the scale parameter in the grid search (defaults to 10). This must be a positive number. It will only be used if `mu` is set to `NA`. Note that larger values will result in a longer search time, and may cause numerical instabilities.

Details

This method is based on a Poisson transmission model, and hence may be most most valid at the beginning of an epidemic. In their model, the serial distribution is assumed to be discrete with a finite number of possible values. In this implementation, if `mu` is not `NA`, the serial distribution is taken to be a discretized version of a gamma distribution with shape parameter `1` and scale parameter `mu` (and hence mean `mu`). When `mu` is `NA`, the function implements a grid search algorithm to find the maximum likelihood estimator over all possible gamma distributions with unknown shape and scale, restricting these to a prespecified grid (see the parameters `grid_length`, `max_shape` and `max_scale`). In both cases, the largest value of the support is chosen such that the cumulative distribution function of the original (pre-discretized) gamma distribution has cumulative probability of no more than 0.999 at this value.

When the serial distribution is known (i.e., `mu` is not `NA`), sensitivity testing of `mu` is strongly recommended. If the serial distribution is unknown (i.e., `mu` is `NA`), the likelihood function can be flat near the maximum, resulting in numerical instability of the optimizer. When `mu` is `NA`, the implementation takes considerably longer to run. Users should be careful about units of time (e.g., are counts observed daily or weekly?) when implementing.

The model developed in White and Pagano (2008) is discrete, and hence the serial distribution is finite discrete. In our implementation, the input value `mu` is that of a continuous distribution. The algorithm discretizes this input, and so the mean of the estimated serial distribution returned (when `serial` is set to `TRUE`) will differ from `mu` somewhat. That is to say, if the user notices that the input mu and the mean of the estimated serial distribution are different, this is to be expected, and is caused by the discretization.

Value

If serial is identical to TRUE, a list containing the following components is returned:

- r0 - the estimate of R0
- supp - the support of the estimated serial distribution
- pmf - the probability mass function of the estimated serial distribution

Otherwise, if serial is identical to FALSE, only the estimate of R0 is returned.

References

White and Pagano (Statistics in Medicine, 2008) doi:10.1002/sim.3136

See Also

vignette("wp_serial", package="Rnaught") for examples of using the serial distribution.

Examples

```
# Weekly data.
cases <- c(1, 4, 10, 5, 3, 4, 19, 3, 3, 14, 4)

# Obtain R0 when the serial distribution has a mean of five days.
wp(cases, mu = 5 / 7)

# Obtain R0 when the serial distribution has a mean of three days.
wp(cases, mu = 3 / 7)

# Obtain R0 when the serial distribution is unknown.
# Note that this will take longer to run than when `mu` is known.
wp(cases)

# Same as above, but specify custom grid search parameters. The larger any of
# the parameters, the longer the search will take, but with potentially more
# accurate estimates.
wp(cases, grid_length = 40, max_shape = 4, max_scale = 4)

# Return the estimated serial distribution in addition to the estimate of R0.
estimate <- wp(cases, serial = TRUE)

# Display the estimate of R0, as well as the support and probability mass
# function of the estimated serial distribution returned by the grid search.
estimate$r0
estimate$supp
estimate$pmf
```

Index