Package ‘REffectivePred’

November 12, 2024

Type Package

Title Pandemic Prediction Model in an SIRS Framework

Version 1.0.1

Description A suite of methods to fit and predict case count data using
a compartmental SIRS (Susceptible – Infectious – Recovered – Susceptible)
model, based on an assumed specification of the effective reproduction
number. The significance of this approach is that it relates epidemic
progression to the average number of contacts of infected individuals,
which decays as a function of the total susceptible fraction remaining
in the population. The main functions are pred.curve(), which computes
the epidemic curve for a set of parameters, and estimate.mle(), which
finds the best fitting curve to observed data. The easiest way to pass
arguments to the functions is via a config file, which contains input
settings required for prediction, and the package offers two methods,
navigate_to_config() which points the user to the configuration file,
and re_predict() for starting the fit-predict process. The main model was published in
Razvan G. Romanescu et al. <doi:10.1016/j.epidem.2023.100708>.

Imports yaml, config, zoo, grDevices, utils

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.3.2

Collate 'prediction_model.v16.R' 'executions.v16.R'

NeedsCompilation no

Author Razvan Romanescu [aut, cre],
Songdi Hu [aut],
Md Ashiqul Haque [aut],
Olivier Tremblay-Savard [aut]

Maintainer Razvan Romanescu <razvan_romanescu@hotmail.com>

Repository CRAN

Date/Publication 2024-11-12 09:50:05 UTC

Contents

`ci.curve` *Confidence bands*

Description

Computes the pointwise confidence interval of the epidemic curve.

Usage

```
ci.curve(
  fit = NULL,
  H.E = NULL,
  H.W = NULL,
  scenario = NULL,
  cases = NULL,
  cfg = NULL,
  restrictions = NULL,
  restriction.starts = NULL,
  ranges = NULL,
  rt_func = 1,
  fit.t.pred = NULL,
  predict.beyond = 0,
  lt = NULL,
  adj.period = NULL,
  population = NULL,
  rho = NULL,
  serial_mean = NULL,
  serial_var = NULL,
  window_size = NULL,
  eps = .Machine$double.eps^(1/2)
)
```

Arguments

`fit`	Output from function estimate.mle.
`H.E`	Mobility metrics for category Retail & Entertainment. Currently unsupported.
`H.W`	Mobility metrics for category Workplaces. Currently unsupported.
`scenario`	A character string describing options to deal with restrictions. Currently unsupported.
`cases`	vector of case counts.
`cfg`	The object that contains all variables from the configuration file. `fit`, `H.E`, `H.W`, `scenario`, and `cases` are also required for the method to execute. All other parameters will not be used if `cfg` is passed to the method.
`restrictions`	A numeric integer vector giving the severity of restrictions. Zero means no restriction, and higher numbers means greater severity/disruption. The ordered unique values should be consecutive integers starting from zero. Each number (other than 0) adds a new parameter to the fit.
`restriction.starts`	A vector of same length as restrictions, of times when restrictions came into effect. Note: the first index time should be 1.
`ranges`	A vector of time ranges for the different waves. The wave ranges should be contiguous, with at least one unit of time between consecutive waves.
`rt_func`	The parametric form of function c(). Options are listed under function c_helper.
`fit.t.pred`	Time from which prediction is done. If use.actual.not.predicted is TRUE, values of S_t before this time will be computed using actual counts.
`predict.beyond`	Number of days to predict beyond the end of `cases`. See Details for usage notes.
`lt`	The length of cases.
`adj.period`	Adjustment period following a change in severity level. Restriction level (psi) is linearly interpolated from the old to the new value over this period.
`population`	Total population size.
`rho`	A vector of under-reporting rates of the same length as cases. If a scalar is supplied, the vector will be constant with this value.
`serial_mean`	Mean of the serial interval on the log scale.
`serial_var`	Variance of the serial interval on the log scale.
`window_size`	The maximum value for the serial interval.
`eps`	The epsilon value for computing finite differences.

Value

Returns a matrix with two rows containing Wald-style confidence bounds:

- ci_lower - lower bound of a 95% pointwise CI of the best fit curve.
- ci_upper - upper bound of a 95% pointwise CI of the best fit curve.

Examples

```
library(REffectivePred)
## Read in the data
path_to_data <- system.file("extdata/NY_OCT_4_2022.csv", package = "REffectivePred")
data <- read.csv(path_to_data)
head(data)
cases <- diff(c(0, data$cases)) # Convert cumulative cases into daily cases
lt <- length(cases)             # Length of cases
Time <- as.Date(data$date, tryFormats = c("%d-%m-%Y", "%d/%m/%Y"))

navigate_to_config() # Open the config file, make any necessary changes here.
path_to_config <- system.file("config.yml", package = "REffectivePred")  # Read config file
cfg <- load_config()    # Build the cfg object

# Estimate parameters
est <- estimate.mle(
    cases = cases,
    cfg = cfg,
    hessian = TRUE
    )
a1 <- est$a1
a2 <- est$a2
a3 <- est$a3
a4 <- est$a4
nu <- est$nu
vt <- c(1, est$vt_params_est)
psi <- est$Psi
betas <- est$betas

# Predict curve
r1 <- pred.curve(
a1 = a1,
a2 = a2,
a3 = a3,
a4 = a4,
nu = nu,
variant.transm = vt,
Psi = psi,
betas = betas,
cases = cases,
cfg = cfg
)

plot_outputs(Time = Time,
cases = cases,
cfg = cfg,
curve = r1,
option = 2
)

bounds <- ci.curve(fit = est,
                   cases = cases,
                    cfg = cfg)

# Adding CI bands
# lines(c(Time, Time[length(Time)]+(1:predict.beyond)), bounds[2,], lty = 2)
# lines(c(Time, Time[length(Time)]+(1:predict.beyond)), bounds[1,], lty = 2)
```

c_helper *Contact rate function.*

Description

Computes the c() function.

Usage

```
c_helper(
  rt_func = 1,
  st.inner = NULL,
  a1 = NULL,
  a2 = NULL,
  a3 = NULL,
  a4 = NULL,
  psi = NULL
)
```

Arguments

`rt_func`	Options are: • 1 - Two piece exponential. • 2 - Exponential power model adapted from Granich et al. (2009) • 3 - Mass action. • 4 - Shifted inverse. • 5 - Power. • 6 - Poisson. • 7 - Geometric.
`st.inner`	The susceptible fraction S_t.
`a1`, `a2`, `a3`, `a4`	Parameters of the contact rate curve specified by `rt_func`;
`psi`	A vector of same length as st.inner containing the corresponding psi restriction factor, or a scalar.

Details

See Romanescu et al. (2023) for the exact forms of the functions.

Value

The value of the contact rate, used to compute R_t.

estimate.mle *Fit the Model*

Description

Estimate the parameters of the model by maximizing the likelihood function (or, rather, by minimizing the negative log likelihood).

Usage

```
estimate.mle(
  hessian = FALSE,
  H.E = NULL,
  H.W = NULL,
  cases = NULL,
  cfg = NULL,
  ini_params = NULL,
  params_limits = NULL,
  restrictions = NULL,
  restriction.starts = NULL,
  ranges = NULL,
  rt_func = 1,
  silence.errors = FALSE,
  fit.t.pred = NULL,
  param_scale = NULL,
  num.iter = NULL,
  scenario = NULL,
  adj.period = NULL,
  population = NULL,
  rho = NULL,
  serial_mean = serial_mean,
  serial_var = serial_var,
  lt = NULL,
  window_size = NULL,
  verbose = FALSE
)
```

Arguments

`hessian`	Logical. If TRUE, computes the variance-covariance matrix at the MLE.
`H.E`	Mobility metrics for category Retail & Entertainment. Currently unsupported.
`H.W`	Mobility metrics for category Workplaces. Currently unsupported.
`cases`	Vector of case counts.

`cfg`	The object that contains all variables from the configuration file. This includes all function arguments except for `cases`, `hessian`, `H.E`, and `H.W`. All other arguments are overridden if `cfg` is passed to the method.
`ini_params`	Initial parameter values to be used in optimization. Includes the following sets of parameters in a vector, in this order: • (a1,a2,a3,a4) = parameters for curve c() specified by `rt_func`; • nu = loss of immunity rate; • (v2,v3,v4,v5) = transmissibility of variants in waves 2+, as relative multiplication factors compared to transmissibility in wave 1; • (psi1,psi2,psi3,psi4) = psi parameters for severity levels 1,2,3 and 4. • (u,v) = variance parameters. Only u is currently in use. • (beta0,beta.R,beta.E,beta.W), when restrictions = NULL. Currently unsupported.
`params_limits`	Boundaries/limits of the ini_params.
`restrictions`	A numeric integer vector giving the severity of restrictions. Zero means no restriction, and higher numbers means greater severity/disruption. The ordered unique values should be consecutive integers starting from zero. Each number (other than 0) adds a new parameter to the fit. restrictions = NULL causes the function to use mobility data instead of the psi values (currently unsupported).
`restriction.starts`	A vector of same length as restrictions, of times when restrictions came into effect. Note: the first index time should be 1.
`ranges`	An vector of time ranges for the different waves. The waves ranges should be contiguous, with at least one unit of time between consecutive waves.
`rt_func`	The parametric form of function c(). Options are listed under function c_helper.
`silence.errors`	Logical. If TRUE, ignores certain errors to allow optimization to proceed. Not all errors can be ignored.
`fit.t.pred`	Time from which prediction is done. If use.actual.not.predicted is TRUE, values of S_t before this time will be computed using actual counts.
`param_scale`	Parameter scale. Passed as argument parscale to optim.
`num.iter`	Maximum number of iterations. Passed as argument maxit to optim.
`scenario`	A character string describing options to deal with restrictions. Currently unsupported.
`adj.period`	Delays in society adjusting.
`population`	total population size.
`rho`	A vector of under-reporting rates of the same length as cases. If a scalar is supplied, the vector will be constant with this value.
`serial_mean`	Mean of the serial interval on the log scale.
`serial_var`	Variance of the serial interval on the log scale.
`lt`	The length of cases.
`window_size`	The maximum value for the serial interval.
`verbose`	Logical. If TRUE, provides additional details while running the function.

Value

A list of maximum likelihood estimates of the parameters. Includes:

- a1
- a2
- a3
- a4
- nu
- vt_params_est
- Psi
- betas
- negative_log_lik
- mle
- hessian
- SE

Examples

```
library(REffectivePred)
## Read in the data
path_to_data <- system.file("extdata/NY_OCT_4_2022.csv", package = "REffectivePred")
data <- read.csv(path_to_data)
head(data)
cases <- diff(c(0, data$cases)) # Convert cumulative cases into daily cases
lt <- length(cases)             # Length of cases
Time <- as.Date(data$date, tryFormats = c("%d-%m-%Y", "%d/%m/%Y"))

navigate_to_config() # Open the config file, make any necessary changes here.
path_to_config <- system.file("config.yml", package = "REffectivePred")  # Read config file
cfg <- load_config()    # Build the cfg object

##### Option 1: populate the global environment with args to pass to function.
population <- cfg$population # Population size
window_size <- cfg$window.size
adj.period <- cfg$adj.period
fit.t.pred <- cfg$fit.t.pred # Time of prediction
not.predict <- cfg$not.predict
rt.func.num <- cfg$rt.func.num # choose which Rt function you want to use
num.iter <- cfg$num.iter
silence.errors <- cfg$silence.errors
predict.beyond <- cfg$predict.beyond
curve_params <- as.double(unlist(cfg$curve_params))
vt_params <- as.double(unlist(cfg$vt_params)) # The vt initial values, starting at wave 2
restriction_levels <- as.double(unlist(cfg$restriction_levels)) # Psi, u, and v parameters
betas <- as.double(unlist(cfg$betas)) #   betas
ini_params <- c(curve_params, vt_params, restriction_levels, betas)
restrictions_params <- cfg$restrictions_params
restriction_st_params <- cfg$restriction_st_params
```

```
param_scale <- abs(ini_params) / 10
waves_list <- ranges_to_waves(cfg$waves_list)
params_limits <- cfg$params_limits
num_waves <- cfg$num_waves
waves <- waves_1d_list(num_waves, waves_list)
rho <- eval(parse(text = cfg$rho))
serial_mean <- cfg$serial_mean
serial_var <- cfg$serial_var

est <- estimate.mle(
  ini_params = ini_params,
  params_limits = params_limits,
  restrictions = restrictions_params,
  restriction.starts = restriction_st_params,
  ranges = waves,
  rt_func = rt.func.num,
  silence.errors = silence.errors,

  fit.t.pred = fit.t.pred,
  param_scale = param_scale,
  num.iter = num.iter,
  cases = cases,
  scenario = NULL,
  H.E = NULL,
  H.W = NULL,
  adj.period = adj.period,
  population = population,
  rho = rho,
  serial_mean = serial_mean,
  serial_var = serial_var,
  lt = lt,
  window_size = window_size,
  hessian = FALSE
)
print(est)

##### Option 2: pass the cfg object instead.
est <- estimate.mle(
    cases = cases,
    cfg = cfg,
    hessian = FALSE
    )
print(est)
```

find_ends *Detect end of waves*

Description

Find the approximate end times of waves given times of peaks. This is based on local minima of R_t.

Usage

```
find_ends(rt_values, peaks_x, search_range)
```

Arguments

`rt_values`	A vector containing rt values for each time point.
`peaks_x`	Time points of peaks based on daily cases (one for each wave).
`search_range`	The range of data points to go through for filtering invalid ends (a vector).

Details

Note: This is provided for convenience only, and is not meant to replace an analyst's determination of wave bounds.

Value

A list containing the detected end times of waves, as scalars.

`find_starts` *Detect start of waves*

Description

Find the approximate beginning times of waves given times of peaks. This is based on local maxima of R_t.

Usage

```
find_starts(rt_values, peaks_x, search_range)
```

Arguments

`rt_values`	A vector containing rt values for each time point.
`peaks_x`	Time points of peaks based on daily cases (one for each wave).
`search_range`	The range of data points to go through for filtering invalid beginnings (a vector).

Details

Note: This is provided for convenience only, and is not meant to replace an analyst's determination of wave bounds.

Value

A list containing the detected start times of waves, as scalars.

`load_config` *Load configuration file*

Description

Load the configuration file as an object in the global environment. This can be passed to the main functions instead of the individual arguments within, for user convenience.

Usage

```
load_config()
```

Value

A 'config' object, which is a list that stores input parameters and settings from config.yml. Variable names are imported exactly. See the configuration file for details.

`log_lklh` *The likelihood function*

Description

The negative log likelihood function of the model.

Usage

```
log_lklh(
  param,
  params_limits,
  restrictions = NULL,
  restriction.starts = NULL,
  ranges = NULL,
  rt_func = 1,
  silence.errors = FALSE,
  fit.t.pred,
  lt,
  cases,
  scenario = NULL,
  H.E,
  H.W,
  adj.period,
  population,
  rho,
  serial_mean,
  serial_var,
  window_size
)
```

Arguments

`param`	Includes the following sets of parameters in a vector, in this order: • a1,a2,a3,a4 - Parameters for curve c() specified by `rt_func`. • nu - Loss of Immunity rate. • v2,v3,v4,v5 - Transmissibility of variants in waves 2+, as relative multiplication factors compared to transmissibility in wave 1. • psi1,psi2,psi3,psi4 - Psi parameters for severity levels 1,2,3 and 4. • u,v - Variance parameters. Only u is currently in use. • beta0,beta.R,beta.E,beta.W - When restrictions = NULL. Currently unsupported.
`params_limits`	Boundaries/limits of the ini_params.
`restrictions`	A numeric integer vector giving the severity of restrictions. Zero means no restriction, and higher numbers means greater severity/disruption. The ordered unique values should be consecutive integers starting from zero. Each number (other than 0) adds a new parameter to the fit. restrictions = NULL causes the function to use mobility data instead of the psi values (currently unsupported).
`restriction.starts`	A vector of same length as restrictions, of times when restrictions came into effect. Note: the first index time should be 1.
`ranges`	An vector of time ranges for the different waves. The wave ranges should be contiguous, with at least one unit of time between consecutive waves.
`rt_func`	The parametric form of function c(). Options are listed under function c_helper.
`silence.errors`	Ignores (skips) NA or NaN values when summing up likelihood contributions over time.
`fit.t.pred`	Time of prediction.
`lt`	Length of cases.
`cases`	A vector containing cases for each time-point.
`scenario`	A character string describing options to deal with restrictions. Currently unsupported.
`H.E`	Mobility metrics for category Retail & Entertainment. Currently unsupported.
`H.W`	Mobility metrics for category Workplaces. Currently unsupported.
`adj.period`	Delays in society adjusting.
`population`	total population size.
`rho`	Under-reporting fraction.
`serial_mean`	Mean of the serial interval on the log scale.
`serial_var`	Variance of the serial interval on the log scale.
`window_size`	The maximum value for the serial interval.

Details

The predicted curve is computed based on parameters supplied, by first calling the prediction function `pred.curve`. The probability model used to compute the likelihood assumes that observed infection at time t are $\sim N(mean = I_t, sd = \sqrt{u * I_t})$, where I_t are predicted infections, and sums the log-likelihood contributions for each time t during waves, and up to `fit.t.pred`.

Value

The negative log likelihood value of the data.

`navigate_to_config` *Navigate to the config file*

Description

Prints the path to the config file and opens the config file.

Usage

```
navigate_to_config()
```

Value

The path to the configuration file.

`plot_outputs` *Plotting function*

Description

Various plots related to an epidemic curve.

Usage

```
plot_outputs(
  curve = NULL,
  Time = NULL,
  cases = NULL,
  cfg = NULL,
  window_size = NULL,
  serial_mean,
  serial_var,
  predict.beyond = 0,
  waves_list = NULL,
  num_waves = NULL,
  rt_func = NULL,
  restrictions = NULL,
  restriction.starts = NULL,
  a1 = NULL,
  a2 = NULL,
  a3 = NULL,
  a4 = NULL,
  rt.max = NULL,
  option = "all",
  verbose = FALSE
)
```

Arguments

`curve`	The output list from the prediction function, see `pred.curve`.
`Time`	A vector of dates corresponding to cases.
`cases`	A vector containing cases for each time point.
`cfg`	The object that contains all variables from the configuration file. `curve`, `Time`, and `cases` are also required for the method to execute. All other parameters will not be used if `cfg` is passed to the method.
`window_size`	The maximum value for the serial interval.
`serial_mean`	Mean of the serial interval on the log scale.
`serial_var`	Variance of the serial interval on the log scale.
`predict.beyond`	How many days to predict beyond the end of `cases`.
`waves_list`	A two-dimensional list containing the waves' time data.
`num_waves`	Total number of waves.
`rt_func`	A flag that indicates which rt function to use. Should match the shape of `curve`.
`restrictions`	A numeric integer vector giving the severity of restrictions.
`restriction.starts`	A vector of same length as restrictions, of times when restrictions came into effect. Note: the first index time should be 1.
`a1`, `a2`, `a3`, `a4`	Parameters of the contact rate curve specified by `rt_func`. These override the values given in `curve` for the last plot only. If not specified, will use the values from `curve`.
`rt.max`	An optional upper limit for the y-axis when plotting R_t.
`option`	A choice of which plot to return (1,2, or 3 - see Value for options). If set to "all" (the default) plots all three figures.
`verbose`	Logical. If TRUE, provides additional details while running the function.

Value

NULL. Generates a few plots: a plot of R_t over time, with waves shaded (for `option` = 1); the epidemic curve overlaid on top of observed cases (`option` = 2), where the shading reflects restriction measures; and a plot of the theoretical R_t versus S_t, in a fully susceptible population with no restrictions (`option` = 3).

Examples

```
library(REffectivePred)
## Read in the data
path_to_data <- system.file("extdata/NY_OCT_4_2022.csv", package = "REffectivePred")
data <- read.csv(path_to_data)
```

```
head(data)
cases <- diff(c(0, data$cases)) # Convert cumulative cases into daily cases
lt <- length(cases)             # Length of cases
Time <- as.Date(data$date, tryFormats = c("%d-%m-%Y", "%d/%m/%Y"))

navigate_to_config() # Open the config file, make any necessary changes here.
path_to_config <- system.file("config.yml", package = "REffectivePred")  # Read config file
cfg <- load_config()    # Build the cfg object

# Estimate parameters
est <- estimate.mle(
    cases = cases,
    cfg = cfg
    )
a1 <- est$a1
a2 <- est$a2
a3 <- est$a3
a4 <- est$a4
nu <- est$nu
vt <- c(1, est$vt_params_est)
psi <- est$Psi
betas <- est$betas

# Predict curve
r1 <- pred.curve(
a1 = a1,
a2 = a2,
a3 = a3,
a4 = a4,
nu = nu,
variant.transm = vt,
Psi = psi,
betas = betas,
cases = cases,
cfg = cfg
)

plot_outputs(Time = Time,
cases = cases,
window_size = cfg$window.size,
serial_mean = cfg$serial_mean,
serial_var = cfg$serial_var,
predict.beyond = cfg$predict.beyond,
waves_list = cfg$waves_list,
num_waves = cfg$num_waves,
rt_func = cfg$rt.func.num,
curve = r1,
restrictions = cfg$restrictions_params,
restriction.starts = cfg$restriction_st_params,
rt.max = 10
)
```

pred.curve *Epidemic Curve Model*

Description

Computes the epidemic curve and associated quantities for a given parameter set.

Usage

```
pred.curve(
  a1 = 0,
  a2 = 0,
  a3 = 0,
  a4 = 0,
  nu = 0,
  variant.transm = NULL,
  Psi = NULL,
  betas = NULL,
  cases = NULL,
  cfg = NULL,
  use.actual.not.predicted = FALSE,
  restrictions = NULL,
  restriction.starts = NULL,
  ranges = NULL,
  rt_func = 1,
  fit.t.pred = NULL,
  predict.beyond = 0,
  scenario = NULL,
  H.E = NULL,
  H.W = NULL,
  adj.period = NULL,
  population = NULL,
  rho = NULL,
  serial_mean = NULL,
  serial_var = NULL,
  lt = NULL,
  window_size = NULL,
  verbose = FALSE
)
```

Arguments

`a1, a2, a3, a4`	Parameters of the contact rate curve specified by `rt_func`.
`nu`	Loss of immunity rate beyond the first wave.
`variant.transm`	Vector of transmissibility of variants in each wave, as relative multiplication factors compared to transmissibility in wave 1. Should always be 1 for the first wave.

`Psi`	Vector of restriction parameters for severity levels 1 - 4.
`betas`	Vector containing (beta0,beta.R,beta.E,beta.W), when restrictions = NULL. Not currently implemented.
`cases`	vector of case counts.
`cfg`	The object that contains all variables from the configuration file. `a1`, `a2`, `a3`, `a4`, `nu`, `variant.transm`, `Psi`, `betas`, and `cases` are also required for the method to execute. All other parameters will not be used if `cfg` is passed to the method.
`use.actual.not.predicted`	Logical; if FALSE (default), the susceptible fraction is updated using predicted cases. Otherwise updated using actual cases.
`restrictions`	A numeric integer vector giving the severity of restrictions. Zero means no restriction, and higher numbers means greater severity/disruption. The ordered unique values should be consecutive integers starting from zero. Each number (other than 0) adds a new parameter to the fit.
`restriction.starts`	A vector of same length as restrictions, of times when restrictions came into effect. Note: the first index time should be 1.
`ranges`	A vector of time ranges for the different waves. The wave ranges should be contiguous, with at least one unit of time between consecutive waves.
`rt_func`	The parametric form of function c(). Options are listed under function `c_helper`.
`fit.t.pred`	Time from which prediction is done. If use.actual.not.predicted is TRUE, values of S_t before this time will be computed using actual counts.
`predict.beyond`	Number of days to predict beyond the end of `cases`. See Details for usage notes.
`scenario`	A character string describing options to deal with restrictions. Currently unsupported.
`H.E`	Mobility metrics for category Retail & Entertainment. Currently unsupported.
`H.W`	Mobility metrics for category Workplaces. Currently unsupported.
`adj.period`	Adjustment period following a change in severity level. Restriction level (psi) is linearly interpolated from the old to the new value over this period.
`population`	Total population size.
`rho`	A vector of under-reporting rates of the same length as cases. If a scalar is supplied, the vector will be constant with this value.
`serial_mean`	Mean of the serial interval on the log scale.
`serial_var`	Variance of the serial interval on the log scale.
`lt`	The length of cases.
`window_size`	The maximum value for the serial interval.
`verbose`	Logical. If TRUE, provides additional details while running the function.

Details

At each time step, R_t is computed using the contact rate function $c(S_t)$ implemented via c_helper. Then the number of cases is estimated using formula:

$$y_{t+1} = R_t \sum_{s=1}^{M} w_s y_{t+1-s}$$

Finally, the fraction S_{t+1} is updated. This creates a curve over the entire range of `ranges`. See Romanescu R, Hu S, Nanton D, Torabi M, Tremblay-Savard O, Haque MA. The effective reproductive number: modeling and prediction with application to the multi-wave Covid-19 pandemic. Epidemics. 2023 Jul 20:100708 doi:10.1016/j.epidem.2023.100708 for more details.

For predicting an ongoing wave beyond the end of cases, the end of `ranges` (or `waves_list`, if using `cfg`) should be specified to match the `predict.beyond` argument. As well, argument `use.actual.not.predicted` should be set to FALSE when predicting beyond the end of `cases`.

Value

Returns list:

- Predicted Infections - Vector of estimated infections, computed as predicted cases divided by rho.
- Predicted Cases - Vector of predicted cases.
- Predicted R_t - Vector of predicted susceptible fractions
- Predicted R_t - Vector of (model) predicted R_t.
- Predicted Lambda t - Vector of predicted Lambda_t, which is the numerator used in computing the empirical R_t.
- Psi.vec - Vector of psi values, which pastes together parameters psi over the period they apply, or 1 when there are no restrictions.
- Contact rate params - Vector of the curve parameters (a1, a2, a3, a4).

Examples

```
library(REffectivePred)
## Read in the data
path_to_data <- system.file("extdata/NY_OCT_4_2022.csv", package = "REffectivePred")
data <- read.csv(path_to_data)
head(data)
cases <- diff(c(0, data$cases)) # Convert cumulative cases into daily cases
lt <- length(cases)             # Length of cases
Time <- as.Date(data$date, tryFormats = c("%d-%m-%Y", "%d/%m/%Y"))

navigate_to_config() # Open the config file, make any necessary changes here.
path_to_config <- system.file("config.yml", package = "REffectivePred")  # Read config file
cfg <- load_config()    # Build the cfg object

# Example 1. Using fits from Romanescu et al. (2023)

r1 <- pred.curve(
a1 = 0.58,
a2 = 1.12,
nu = 0.56,
variant.transm = c(1,1.22,0.36,0.56),
Psi = c(0.58,0.52,0.49),
cases = cases,
cfg = cfg
)

plot(cases, xlab="Day", ylab="Predicted cases")
lines(r1$'Predicted Cases', col='red')

# Example 2. Best fit curve
est <- estimate.mle(
    cases = cases,
    cfg = cfg
    )
a1 <- est$a1
a2 <- est$a2
a3 <- est$a3
a4 <- est$a4
nu <- est$nu
vt <- c(1, est$vt_params_est)
psi <- est$Psi
betas <- est$betas

r1 <- pred.curve(
a1 = a1,
a2 = a2,
a3 = a3,
a4 = a4,
nu = nu,
variant.transm = vt,
Psi = psi,
betas = betas,
cases = cases,
cfg = cfg
)
plot(r1$'Predicted Infections', xlab="Day", ylab="Predicted infections")
```

`ranges_to_waves` *Utility function for range manipulation*

Description

Converts a list of waves to a two-dimensional list.

Usage

```
ranges_to_waves(waves_list)
```

Arguments

`waves_list`	A list containing ranges (start, end) of each wave.

Value

A two-dimensional list with individual waves as sub-lists.

`re_predict` *Demo of main functions*

Description

Fits the model to an example case data and predicts the epidemic curves and plots the outputs.

Arguments

`path_to_data`	Absolute path to the dataset in csv format.

Details

Please modify the config file before invoking this method. The config file contains all the settings and initial parameter values necessary for the algorithm to run. Path to the dataset (in csv format) is also set in the config file. This file should be updated to the desired specifications before running this demo. To open it, execute REffectivePred::navigate_to_config().

Value

No return value.

`rt_empirical` *Empirical estimate of R_t*

Description

Compute empirical R_t, via Cori et al. (2013) method.

Usage

```
rt_empirical(cases, window_size, serial_mean, serial_var)
```

Arguments

`cases`	Vector of (confirmed) cases.
`window_size`	The maximum value for the serial interval.
`serial_mean`	Mean of the serial interval on the log scale.
`serial_var`	Variance of the serial interval on the log scale.

Value

A vector of same length as cases, giving the empirical estimate of the effective reproductive number over time.

`serial.helper` *Serial interval*

Description

Helper function for computing weights `w` based on the serial interval.

Usage

```
serial.helper(window_size, serial_mean = log(4), serial_var = log(1.380715))
```

Arguments

`window_size`	The maximum value for the serial interval.
`serial_mean`	Mean of the serial interval on the log scale. See Details.
`serial_var`	Variance of the serial interval on the log scale. See Details.

Details

Computed based on a log normal density function, as in Nishiura et al. (2020). Parameters `serial_mean` and `serial_var` are arguments `meanlog` and `sdlog` of function `dlnorm`. Default values are taken from the same reference.

Value

A vector that stores the serial interval in reverse order. This is meant to be multiplied to infections in chronological order.

`waves_1d_list` *Utility function for range manipulation*

Description

Combines multiple waves into one vector.

Usage

```
waves_1d_list(num_waves, waves_list)
```

Arguments

`num_waves`	Total number of waves.
`waves_list`	A list containing individual waves as sub-lists.

Value

A vector with the first `num_waves` waves combined.

Index