Package 'EpiNow2'

December 14, 2020

Type Package

Title Estimate Real-Time Case Counts and Time-Varying Epidemiological Parameters

Version 1.3.2

Description Estimates the time-varying reproduction number, rate of spread, and doubling time using a range of open-source tools (Abbott et al. (2020) <doi:10.12688/wellcomeopenres.16006.1>), and current best practices (Gostic et al. (2020) <doi:10.1101/2020.06.18.20134858>). It aims to help users avoid some of the limitations of naive implementations in a framework that is informed by community feedback and is under active development.

License MIT + file LICENSE

URL https:/epiforecasts.io/EpiNow2/, https:/epiforecasts.io/EpiNow2/dev/, https://github.com/epiforecasts/EpiNow2

BugReports https://github.com/epiforecasts/EpiNow2/issues

Imports cowplot, data.table, futile.logger (>= 1.4), future, future.apply, ggplot2, lubridate, methods, patchwork, progressr, purrr, R.utils (>= 2.0.0), Rcpp (>= 0.12.0), rlang (>= 0.4.7), rstan (>= 2.21.1), runner, scales, stats, truncnorm, lifecycle, utils

Suggests countrycode, dplyr, EpiSoon, forecastHybrid, here, kableExtra, knitr, magrittr, rmarkdown, rnaturalearth, rstantools, spelling, tidyr, testthat, covr

Additional_repositories https://epiforecasts.io/drat/

RoxygenNote 7.1.1

Biarch true

Depends R (>= 3.4.0)

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), rstan (>= 2.21.1), StanHeaders (>= 2.21.0-5)

Language en-GB

Encoding UTF-8

LazyData true

NeedsCompilation yes

Author Sam Abbott [aut, cre] (<https://orcid.org/0000-0001-8057-8037>),
Joel Hellewell [aut] (<https://orcid.org/0000-0003-2683-0849>),
Katharine Sherratt [aut],
Katelyn Gostic [aut],
Joe Hickson [aut],
Hamada S. Badr [aut] (<https://orcid.org/0000-0002-9808-2344>),
Michael DeWitt [aut] (<https://orcid.org/0000-0001-8940-1967>),
Robin Thompson [aut],
Sophie Meakin [ctb],
James Munday [ctb],
Nikos Bosse [ctb],
Paul Mee [ctb],
Peter Ellis [ctb],
EpiForecasts [aut],
Sebastian Funk [aut]

Maintainer Sam Abbott <sam.abbott@lshtm.ac.uk>

Repository CRAN

Date/Publication 2020-12-14 09:00:15 UTC

R topics documented:

adjust_infection_to_report 5
allocate_delays 6
allocate_empty 7
backcalc_opts 8
bootstrapped_dist_fit 9
calc_CrI 10
calc_CrIs 11
calc_summary_measures 11
calc_summary_stats 12
clean_nowcasts 13
clean_regions 13
construct_output 14
convert_to_logmean 14
convert_to_logsd 15
copy_results_to_latest 16
country_map 16
create_backcalc_data 18
create_clean_reported_cases 19
create_future_rt 19
create_gp_data 20
create_initial_conditions 21
create_obs_model 21
create_rt_data 22

create_shifted_cases . . . 23
create_stan_args . . . 24
create_stan_data . . . 25
delay_opts . . . 26
dist_fit . . . 27
dist_skel . . . 28
epinow . . . 29
estimates_by_report_date . . . 32
estimate_delay . . . 33
estimate_infections . . . 33
estimate_secondary . . . 37
estimate_truncation . . . 40
example_confirmed . . . 42
expose_stan_fns . . . 43
extract_CrIs . . . 43
extract_inits . . . 44
extract_parameter . . . 44
extract_parameter_samples . . . 45
extract_stan_param . . . 46
extract_static_parameter . . . 46
filter_opts . . . 47
fit_model_with_nuts . . . 48
fit_model_with_vb . . . 48
forecast_infections . . . 49
forecast_secondary . . . 51
format_fit . . . 52
gamma_dist_def . . . 53
generation_times . . . 54
get_dist . . . 54
get_generation_time . . . 55
get_incubation_period . . . 56
get_raw_result . . . 56
get_regional_results . . . 57
get_regions . . . 58
get_regions_with_most_reports . . . 59
global_map . . . 59
gp_opts . . . 61
growth_to_R . . . 62
incubation_periods . . . 63
init_cumulative_fit . . . 63
lognorm_dist_def . . . 64
make_conf . . . 65
map_prob_change . . . 65
match_output_arguments . . . 66
obs_opts . . . 67
opts_list . . . 68
plot.epinow . . . 69
plot.estimate_infections . . . 69

plot.estimate_secondary . . . 70
plot.estimate_truncation . . . 71
plot_CrIs . . . 71
plot_estimates . . . 72
plot_summary . . . 73
process_region . . . 74
process_regions . . . 75
regional_epinow . . . 75
regional_runtimes . . . 78
regional_summary . . . 80
report_cases . . . 81
report_plots . . . 83
report_summary . . . 84
rstan_opts . . . 85
rstan_sampling_opts . . . 86
rstan_vb_opts . . . 87
rt_opts . . . 88
run_region . . . 89
R_to_growth . . . 91
sample_approx_dist . . . 92
save_estimate_infections . . . 93
save_forecast_infections . . . 94
save_input . . . 95
secondary_opts . . . 95
setup_default_logging . . . 96
setup_dt . . . 97
setup_future . . . 97
setup_logging . . . 98
setup_target_folder . . . 99
simulate_cases . . . 99
simulate_infections . . . 101
stan_opts . . . 102
summarise_key_measures . . . 103
summarise_results . . . 104
summary.epinow . . . 105
summary.estimate_infections . . . 106
theme_map . . . 106
trunc_opts . . . 107
tune_inv_gamma . . . 108
update_horizon . . . 109
update_list . . . 109

Index **110**

`adjust_infection_to_report` *Adjust from Case Counts by Infection Date to Date of Report*

Description

Soft-deprecated Maps from cases by date of infection to date of report via date of onset.

Usage

```
adjust_infection_to_report(
  infections,
  delay_defs,
  reporting_model,
  reporting_effect,
  type = "sample",
  truncate_future = TRUE
)
```

Arguments

`infections`	`data.table` containing a `date` variable and a numeric `cases` variable.
`delay_defs`	A list of single row data.tables that each defines a delay distribution (model, parameters and maximum delay for each model). See `lognorm_dist_def` for an example of the structure.
`reporting_model`	A function that takes a single numeric vector as an argument and returns a single numeric vector. Can be used to apply stochastic reporting effects. See the examples for details.
`reporting_effect`	A numeric vector of length 7 that allows the scaling of reported cases by the day on which they report (1 = Monday, 7 = Sunday). By default no scaling occurs.
`type`	Character string indicating the method to use to transform counts. Supports either "sample" which approximates sampling or "median" would shift by the median of the distribution.
`truncate_future`	Logical, should cases be truncated if they occur after the first date reported in the data. Defaults to `TRUE`.

Value

A `data.table` containing a `date` variable (date of report) and a `cases` variable. If `return_onset = TRUE` there will be a third variable `reference` which indicates what the date variable refers to.

Examples

```
# define example cases
cases <- data.table::copy(example_confirmed)[, cases := as.integer(confirm)]

# define a single report delay distribution
delay_def <- lognorm_dist_def(mean = 5, mean_sd = 1, sd = 3, sd_sd = 1,
                              max_value = 30, samples = 1, to_log = TRUE)

# define a single incubation period
incubation_def <- lognorm_dist_def(mean = incubation_periods[1, ]$mean,
                                   mean_sd = incubation_periods[1, ]$mean_sd,
                                   sd = incubation_periods[1, ]$sd,
                                   sd_sd = incubation_periods[1, ]$sd_sd,
                                   max_value = 30, samples = 1)

# simple mapping
report <- adjust_infection_to_report(cases, delay_defs = list(incubation_def, delay_def))
print(report)

# mapping with a weekly reporting effect
report_weekly <- adjust_infection_to_report(
                      cases, delay_defs = list(incubation_def, delay_def),
                      reporting_effect = c(1.1, rep(1, 4), 0.95, 0.95))
print(report_weekly)

# map using a deterministic median shift for both delays
report_median <- adjust_infection_to_report(cases, delay_defs = list(incubation_def, delay_def),
                                            type = "median")
print(report_median)

# map with a weekly reporting effect and stochastic reporting model
report_stochastic <- adjust_infection_to_report(
                      cases, delay_defs = list(incubation_def, delay_def),
                      reporting_effect = c(1.1, rep(1, 4), 0.95, 0.95),
                      reporting_model = function(n) {
                      out <- suppressWarnings(rnbinom(length(n), as.integer(n), 0.5))
                      out <- ifelse(is.na(out), 0, out)
                      })
print(report_stochastic)
```

`allocate_delays` *Allocate Delays into Required Stan Format*

Description

Stable Allocate delays for stan. Used in `delay_opts`.

Usage

```
allocate_delays(delay_var, no_delays)
```

Arguments

delay_var	List of numeric delays
no_delays	Numeric, number of delays

Value

A numeric array

---

allocate_empty *Allocate Empty Parameters to a List*

Description

Stable Allocate missing parameters to be empty two dimensional arrays. Used internally by `simulate_infections`.

Usage

```
allocate_empty(data, params, n = 0)
```

Arguments

data	A list of parameters
params	A character vector of parameters to allocate to empty if missing.
n	Numeric, number of samples to assign an empty array

Value

A list of parameters some allocated to be empty

Examples

```
data <- list(x = 1, y = 2, z = 30)
EpiNow2:::allocate_empty(data, params = c("x", "t"))
```

`backcalc_opts` *Back Calculation Options*

Description

Stable Defines a list specifying the optional arguments for the back calculation of cases. Only used if `rt = NULL`.

Usage

```
backcalc_opts(prior = "reports", prior_window = 14, rt_window = 1)
```

Arguments

`prior`	A character string defaulting to "reports". Defines the prior to use when deconvolving. Currently implemented options are to use smoothed mean delay shifted reported cases ("reports"), to use the estimated infections from the previous time step seeded for the first time step using mean shifted reported cases ("infections"), or no prior ("none"). Using no prior will result in poor real time performance. No prior and using infections are only supported when a Gaussian process is present. If observed data is not reliable then it a sensible first step is to explore increasing the `prior_window` with a sensible second step being to no longer use reported cases as a prior (i.e set `prior = "none"`).
`prior_window`	Integer, defaults to 14 days. The mean centred smoothing window to apply to mean shifted reports (used as a prior during back calculation). 7 days is minimum recommended settings as this smooths day of the week effects but depending on the quality of the data and the amount of information users wish to use as a prior (higher values equalling a less informative prior).
`rt_window`	Integer, defaults to 1. The size of the centred rolling average to use when estimating Rt. This must be odd so that the central estimate is included.

Value

A list of back calculation settings

Examples

```
# default settings
backcalc_opts()
```

`bootstrapped_dist_fit` *Fit a Subsampled Bootstrap to Integer Values and Summarise Distribution Parameters*

Description

Stable Fits an integer adjusted distribution to a subsampled bootstrap of data and then integrates the posterior samples into a single set of summary statistics. Can be used to generate a robust reporting delay that accounts for the fact the underlying delay likely varies over time or that the size of the available reporting delay sample may not be representative of the current case load.

Usage

```
bootstrapped_dist_fit(
  values,
  dist = "lognormal",
  samples = 2000,
  bootstraps = 10,
  bootstrap_samples = 250,
  max_value,
  verbose = FALSE
)
```

Arguments

`values`	Integer vector of values.
`dist`	Character string, which distribution to fit. Defaults to lognormal (`"lognormal"`) but gamma (`"gamma"`) is also supported.
`samples`	Numeric, number of samples to take overall from the bootstrapped posteriors.
`bootstraps`	Numeric, defaults to 1. The number of bootstrap samples (with replacement) of the delay distribution to take.
`bootstrap_samples`	Numeric, defaults to 100. The number of samples to take in each bootstrap. When the sample size of the supplied delay distribution is less than 100 this is used instead.
`max_value`	Numeric, defaults to the maximum value in the observed data. Maximum delay to allow (added to output but does impact fitting).
`verbose`	Logical, defaults to `FALSE`. Should progress messages be printed

Value

A list summarising the bootstrapped distribution

Examples

```
# lognormal
delays <- rlnorm(500, log(5), 1)
out <- bootstrapped_dist_fit(delays, samples = 1000, bootstraps = 10,
                             dist = "lognormal")
out
```

`calc_CrI` *Calculate Credible Interval*

Description

Stable Adds symmetric a credible interval based on quantiles.

Usage

```
calc_CrI(samples, summarise_by = c(), CrI = 0.9)
```

Arguments

`samples`	A data.table containing at least a value variable
`summarise_by`	A character vector of variables to group by.
`CrI`	Numeric between 0 and 1. The credible interval for which to return values. Defaults to 0.9.

Value

A data.table containing the upper and lower bounds for the specified credible interval

Examples

```
samples <- data.frame(value = 1:10, type = "car")
# add 90% credible interval
calc_CrI(samples)
# add 90% credible interval grouped by type
calc_CrI(samples, summarise_by = "type")
```

calc_CrIs *Calculate Credible Intervals*

Description

Stable Adds symmetric credible intervals based on quantiles.

Usage

```
calc_CrIs(samples, summarise_by = c(), CrIs = c(0.2, 0.5, 0.9))
```

Arguments

samples	A data.table containing at least a value variable
summarise_by	A character vector of variables to group by.
CrIs	Numeric vector of credible intervals to calculate.

Value

A data.table containing the summarise_by variables and the specified lower and upper credible intervals

Examples

```
samples <- data.frame(value = 1:10, type = "car")
# add credible intervals
calc_CrIs(samples)
# add 90% credible interval grouped by type
calc_CrIs(samples, summarise_by = "type")
```

calc_summary_measures *Calculate All Summary Measures*

Description

Stable Calculate summary statistics and credible intervals from a data frame by group.

Usage

```
calc_summary_measures(
  samples,
  summarise_by = NULL,
  order_by = NULL,
  CrIs = c(0.2, 0.5, 0.9)
)
```

Arguments

samples	A data.table containing at least a value variable
summarise_by	A character vector of variables to group by.
order_by	A character vector of parameters to order by, defaults to all summarise_by variables.
CrIs	Numeric vector of credible intervals to calculate.

Value

A data.table containing summary statistics by group.

Examples

```
samples <- data.frame(value = 1:10, type = "car")
# default
calc_summary_measures(samples)
#  by type
calc_summary_measures(samples, summarise_by = "type")
```

---

calc_summary_stats *Calculate Summary Statistics*

Description

Stable Calculate summary statistics from a data frame by group. Currently supports the mean, median and standard deviation.

Usage

```
calc_summary_stats(samples, summarise_by = c())
```

Arguments

samples	A data.table containing at least a value variable
summarise_by	A character vector of variables to group by.

Value

A data.table containing the upper and lower bounds for the specified credible interval

Examples

```
samples <- data.frame(value = 1:10, type = "car")
# default
calc_summary_stats(samples)
#  by type
calc_summary_stats(samples, summarise_by = "type")
```

clean_nowcasts *Clean Nowcasts for a Supplied Date*

Description

Stable This function removes nowcasts in the format produced by `EpiNow2` from a target directory for the date supplied.

Usage

```
clean_nowcasts(date = NULL, nowcast_dir = ".")
```

Arguments

`date`	Date object. Defaults to today's date
`nowcast_dir`	Character string giving the filepath to the nowcast results directory. Defaults to the current directory.

clean_regions *Clean Regions*

Description

Stable Removes regions with insufficient time points, and provides logging information on the input.

Usage

```
clean_regions(reported_cases, non_zero_points)
```

Arguments

`reported_cases`	A data frame of confirmed cases (confirm) by date (date), and region (`region`).
`non_zero_points`	Numeric, the minimum number of time points with non-zero cases in a region required for that region to be evaluated. Defaults to 7.

Value

A dataframe of cleaned regional data

See Also

regional_epinow

construct_output *Construct Output*

Description

Stable Combines the output produced internally by epinow into a single list.

Usage

```
construct_output(
  estimates,
  forecast = NULL,
  estimated_reported_cases,
  plots = NULL,
  summary = NULL,
  samples = TRUE
)
```

Arguments

estimates	List of data frames as output by estimate_infections
forecast	A list of data frames as output by forecast_infections
estimated_reported_cases	A list of dataframes as produced by estimates_by_report_date.
plots	A list of plots as produced by report_plots
summary	A list of summary output as produced by report_summary
samples	Logical, defaults to TRUE. Should samples be saved

Value

A list of output as returned by epinow

convert_to_logmean *Convert mean and sd to log mean for a log normal distribution*

Description

Stable Convert from mean and standard deviation to the log mean of the lognormal distribution. Useful for defining distributions supported by estimate_infections, epinow, and regional_epinow.

Usage

```
convert_to_logmean(mean, sd)
```

Arguments

mean	Numeric, mean of a distribution
sd	Numeric, standard deviation of a distribution

Value

The log mean of a lognormal distribution

Examples

```
convert_to_logmean(2, 1)
```

---

convert_to_logsd *Convert mean and sd to log standard deviation for a log normal distribution*

Description

Stable Convert from mean and standard deviation to the log standard deviation of the lognormal distribution. Useful for defining distributions supported by `estimate_infections`, `epinow`, and `regional_epinow`.

Usage

```
convert_to_logsd(mean, sd)
```

Arguments

mean	Numeric, mean of a distribution
sd	Numeric, standard deviation of a distribution

Value

The log standard deviation of a lognormal distribution

Examples

```
convert_to_logsd(2, 1)
```

copy_results_to_latest *Copy Results From Dated Folder to Latest*

Description

Questioning Copies output from the dated folder to a latest folder. May be undergo changes in later releases.

Usage

```
copy_results_to_latest(target_folder = NULL, latest_folder = NULL)
```

Arguments

target_folder	Character string specifying where to save results (will create if not present).
latest_folder	Character string containing the path to the latest target folder. As produced by setup_target_folder.

country_map *Generate a country map for a single variable.*

Description

Questioning This general purpose function can be used to generate a country map for a single variable. It has few defaults but the data supplied must contain a region_code variable for linking to mapping data. This function requires the installation of the rnaturalearth package. Status of this function is currently questioning as it is uncertain if it is in use. Future releases may depreciate it.

Usage

```
country_map(
  data = NULL,
  country = NULL,
  variable = NULL,
  variable_label = NULL,
  trans = "identity",
  fill_labels = NULL,
  scale_fill = NULL,
  region_col_ne = "provnum_ne",
  ...
)
```

Arguments

data	Dataframe containing variables to be mapped. Must contain a region_code variable.
country	Character string indicating the name of the country to be mapped.
variable	A character string indicating the variable to map data for. This must be supplied.
variable_label	A character string indicating the variable label to use. If not supplied then the underlying variable name is used.
trans	A character string specifying the transform to use on the specified metric. Defaults to no transform ("identity"). Other options include log scaling ("log") and log base 10 scaling ("log10"). For a complete list of options see ggplot2::continous_scale.
fill_labels	A function to use to allocate legend labels. An example (used below) is scales::percent, which can be used for percentage data.
scale_fill	Function to use for scaling the fill. Defaults to a custom ggplot2::scale_fill_manual, which expects the possible values to be "Increasing", "Likely increasing", "Likely decreasing", "Decreasing" or "Unsure".
region_col_ne	Character string indicating the name of a column in the data returned by rnaturalearth::ne_states() that data$region_code corresponds to. Possibilities include provnum_ne, name, fips and others and will depend on which country you are mapping.
...	Additional arguments passed to the scale_fill function

Value

A ggplot2 object containing a country map.

Examples

```
if(requireNamespace("rnaturalearth") & requireNamespace("scales")){
# Example 1
# if you know the provnum_ne codes you can use them directly
eg_data <- data.table::data.table(variable = c("Increasing",
                                               "Decreasing",
                                               "Unsure",
                                               "Likely decreasing",
                                               "Likely increasing"),
                                  region_code = c(5, 7, 6, 8, 9))
# make variable a factor so the ordering is sensible
eg_data$variable <- factor(eg_data$variable, levels = c("Decreasing", "Likely decreasing",
                                                        "Unsure", "Likely increasing",
                                                        "Increasing"))

country_map(data = eg_data, country = "Australia", variable = "variable")


# Example 2
# sometimes it will be more convenient to join your data by name than provnum_ne code:
us_data <- data.table::data.table(variable = c("Increasing",
                                               "Decreasing",
```

```
                                                "Unsure",
                                                "Likely decreasing",
                                                "Likely increasing"),
                                    region_code = c("California",
                                                    "Texas",
                                                    "Florida",
                                                    "Arizona",
                                                    "New York"))
  # make variable a factor so the ordering is sensible in the legend
  us_data$variable <- factor(us_data$variable, levels = c("Decreasing", "Likely decreasing",
                                                          "Unsure", "Likely increasing",
                                                          "Increasing"))

  country_map(data = us_data, country = "United States of America",
              variable = "variable", region_col_ne = "name")
  }
```

`create_backcalc_data` *Create Back Calculation Data*

Description

Stable Takes the output of `backcalc_opts()` and converts it into a list understood by `stan`.

Usage

```
create_backcalc_data(backcalc = backcalc_opts)
```

Arguments

`backcalc`	A list of options as generated by `backcalc_opts()` to define the back calculation. Defaults to `backcalc_opts()`.

Value

A list of settings defining the Gaussian process

See Also

backcalc_opts

Examples

```
# define input data required
data <- list(
    t = 30,
    seeding_time = 7,
    horizon = 7)
```

```
# default gaussian process data
create_gp_data(data = data)

# settings when no gaussian process is desired
create_gp_data(NULL, data)

# custom lengthscale
create_gp_data(gp_opts(ls_mean = 14), data)
```

`create_clean_reported_cases` *Create Clean Reported Cases*

Description

Stable Cleans a data frame of reported cases by replacing missing dates with 0 cases and applies an optional threshold at which point 0 cases are replaced with a moving average of observed cases. See `zero_threshold` for details.

Usage

```
create_clean_reported_cases(reported_cases, horizon, zero_threshold = 50)
```

Arguments

`reported_cases`	A data frame of confirmed cases (confirm) by date (date). confirm must be integer and date must be in date format.
`horizon`	Numeric, defaults to 7. Number of days into the future to forecast.
`zero_threshold`	Numeric defaults to 50. Indicates if detected zero cases are meaningful by using a threshold of 50 cases on average over the last 7 days. If the average is above this thresold then the zero is replaced with the

Value

A cleaned data frame of reported cases

`create_future_rt` *Construct the Required Future Rt assumption*

Description

Stable Converts the `future` argument from `rt_opts()` into arguments that can be passed to `stan`.

Usage

```
create_future_rt(future = "latest", delay = 0)
```

Arguments

future	A character string or integer. This argument indicates how to set future Rt values. Supported options are to project using the Rt model ("project"), to use the latest estimate based on partial data ("latest"), to use the latest estimate based on data that is over 50% complete ("estimate"). If an integer is supplied then the Rt estimate from this many days into the future (or past if negative) past will be used forwards in time.
delay	Numeric mean delay

Value

A list containing a logical called fixed and an integer called from

---

create_gp_data *Create Gaussian Process Data*

Description

Stable Takes the output of `gp_opts()` and converts it into a list understood by `stan`.

Usage

```
create_gp_data(gp = gp_opts(), data)
```

Arguments

gp	A list of options as generated by `gp_opts()` to define the Gaussian process. Defaults to `gp_opts()`.Set to NULL to disable the Gaussian process.
data	A list containing the following numeric values: `t`, `seeding_time`, `horizon`.

Value

A list of settings defining the Gaussian process

See Also

gp_opts

Examples

```
# define input data required
data <- list(
    t = 30,
    seeding_time = 7,
    horizon = 7)

# default gaussian process data
create_gp_data(data = data)
```

```
# settings when no gaussian process is desired
create_gp_data(NULL, data)

# custom lengthscale
create_gp_data(gp_opts(ls_mean = 14), data)
```

`create_initial_conditions` *Create Initial Conditions Generating Function*

Description

Stable Uses the output of `create_stan_data` to create a function which can be used to sample from the prior distributions (or as close as possible) for parameters. Used in order to initialise each `stan` chain within a range of plausible values.

Usage

```
create_initial_conditions(data)
```

Arguments

`data`	A list of data as produced by `create_stan_data`.

Value

An initial condition generating function

`create_obs_model` *Create Observation Model Settings*

Description

Stable Takes the output of `obs_opts()` and converts it into a list understood by `stan`.

Usage

```
create_obs_model(obs = obs_opts())
```

Arguments

`obs`	A list of options as generated by `obs_opts()` defining the observation model. Defaults to `obs_opts()`.

Value

A list of settings ready to be passed to stan defining the Observation Model

See Also

obs_opts

Examples

```
# default observation model data
create_obs_model()

# Poisson observation model
create_obs_model(obs_opts(family = "poisson"))

# Applying a observation scaling to the data
create_obs_model(obs_opts(scale = list(mean = 0.4, sd = 0.01)))
```

---

create_rt_data *Create Time-varying Reproduction Number Data*

Description

Stable Takes the output from `rt_opts()` and converts it into a list understood by `stan`.

Usage

```
create_rt_data(rt = rt_opts(), breakpoints = NULL, delay = 0, horizon = 0)
```

Arguments

`rt`	A list of options as generated by `rt_opts()` defining Rt estimation. Defaults to `rt_opts()`. Set to `NULL` to switch to using back calculation rather than generating infections using Rt.
`breakpoints`	An integer vector (binary) indicating the location of breakpoints.
`delay`	Numeric mean delay
`horizon`	Numeric, forecast horizon.

Value

A list of settings defining the time-varying reproduction number

See Also

rt_settings

Examples

```
# default Rt data
create_rt_data()

# settings when no Rt is desired
create_rt_data(rt = NULL)

# using breakpoints
create_rt_data(rt_opts(use_breakpoints = TRUE), breakpoints = rep(1, 10))
```

`create_shifted_cases` *Create Delay Shifted Cases*

Description

Stable This functions creates a data frame of reported cases that has been smoothed using a centred partial rolling average (with a period set by `smoothing_window`) and shifted back in time by some delay. It is used by `estimate_infections` to generate the mean shifted prior on which the back calculation method (see `backcalc_opts`) is based.

Usage

```
create_shifted_cases(reported_cases, shift, smoothing_window, horizon)
```

Arguments

`reported_cases`	A data frame of confirmed cases (confirm) by date (date). confirm must be integer and date must be in date format.
`shift`	Numeric, mean delay shift to apply.
`smoothing_window`	Numeric, the rolling average smoothing window to apply. Must be odd in order to be defined as a centred average.
`horizon`	Numeric, defaults to 7. Number of days into the future to forecast.

Value

A data frame for shifted reported cases

Examples

```
create_shifted_cases(example_confirmed, 7, 14, 7)
```

`create_stan_args` *Create a List of Stan Arguments*

Description

Stable Generates a list of arguments as required by `rstan::sampling` or `rstan::vb` by combining the required options, with data, and type of initialisation. Initialisation defaults to random but it is expected that `create_initial_conditions` will be used.

Usage

```
create_stan_args(
  stan = stan_opts(),
  data = NULL,
  init = "random",
  verbose = FALSE
)
```

Arguments

`stan`	A list of stan options as generated by `stan_opts()`. Defaults to `stan_opts()`. Can be used to override `data`, `init`, and `verbose` settings if desired.
`data`	A list of stan data as created by `create_stan_data`
`init`	Initial conditions passed to `rstan`. Defaults to "random" but can also be a function (as supplied by `create_intitial_conditions`).
`verbose`	Logical, defaults to `FALSE`. Should verbose progress messages be returned.

Value

A list of stan arguments

Examples

```
# default settings
create_stan_args()

# increasing warmup
create_stan_args(stan = stan_opts(warmup = 1000))
```

create_stan_data *Create Stan Data Required for estimate_infections*

Description

Stable Takes the output of `stan_opts()` and converts it into a list understood by `stan`. Internally calls the other `create_` family of functions to construct a single list for input into stan with all data required present.

Usage

```
create_stan_data(
  reported_cases,
  generation_time,
  rt,
  gp,
  obs,
  delays,
  horizon,
  backcalc,
  shifted_cases,
  truncation
)
```

Arguments

`reported_cases`	A data frame of confirmed cases (confirm) by date (date). confirm must be integer and date must be in date format.
`generation_time`	A list containing the mean, standard deviation of the mean (mean_sd), standard deviation (sd), standard deviation of the standard deviation and the maximum allowed value for the generation time (assuming a gamma distribution).
`rt`	A list of options as generated by `rt_opts()` defining Rt estimation. Defaults to `rt_opts()`. Set to `NULL` to switch to using back calculation rather than generating infections using Rt.
`gp`	A list of options as generated by `gp_opts()` to define the Gaussian process. Defaults to `gp_opts()`.Set to NULL to disable the Gaussian process.
`obs`	A list of options as generated by `obs_opts()` defining the observation model. Defaults to `obs_opts()`.
`delays`	A call to `delay_opts()` defining delay distributions and options. See the documentation of `delay_opts()` and the examples below for details.
`horizon`	Numeric, forecast horizon.
`backcalc`	A list of options as generated by `backcalc_opts()` to define the back calculation. Defaults to `backcalc_opts()`.
`shifted_cases`	A dataframe of delay shifted cases

truncation **Experimental** A list of options as generated by `trunc_opts()` defining the truncation of observed data. Defaults to `trunc_opts()`. See `estimate_truncation()` for an approach to estimating truncation from data.

Value

A list of stan data

`delay_opts` *Delay Distribution Options*

Description

Stable Returns delay distributions formatted for usage by downstream functions.

Usage

```
delay_opts(...)
```

Arguments

... Delay distributions as a list with the following parameters: "mean", "mean_sd", "sd_mean", "sd_sd", and "max" defining a truncated log normal (with all parameters except for max defined in logged form).

Value

A list summarising the input delay distributions.

See Also

convert_to_logmean convert_to_logsd bootstrapped_dist_fit

Examples

```
# no delays
delay_opts()
```

`dist_fit` *Fit an Integer Adjusted Exponential, Gamma or Lognormal distributions*

Description

Stable Fits an integer adjusted exponential, gamma or lognormal distribution using `stan`.

Usage

```
dist_fit(
  values = NULL,
  samples = NULL,
  cores = 1,
  chains = 2,
  dist = "exp",
  verbose = FALSE
)
```

Arguments

`values`	Numeric vector of values
`samples`	Numeric, number of samples to take
`cores`	Numeric, defaults to 1. Number of CPU cores to use (no effect if greater than the number of chains).
`chains`	Numeric, defaults to 2. Number of MCMC chains to use. More is better with the minimum being two.
`dist`	Character string, which distribution to fit. Defaults to exponential (`"exp"`) but gamma (`"gamma"`) and lognormal (`"lognorma"`) are also supported.
`verbose`	Logical, defaults to FALSE. Should verbose progress messages be printed.

Value

A `stan` fit of an interval censored distribution

Examples

```
# integer adjusted exponential model
dist_fit(rexp(1:100, 2), samples = 1000, dist = "exp",
         cores = ifelse(interactive(), 4, 1), verbose = TRUE)


# integer adjusted gamma model
dist_fit(rgamma(1:100, 5, 5), samples = 1000, dist = "gamma",
         cores = ifelse(interactive(), 4, 1), verbose = TRUE)
```

```
# integer adjusted lognormal model
dist_fit(rlnorm(1:100, log(5), 0.2), samples = 1000, dist = "lognormal",
         cores = ifelse(interactive(), 4, 1), verbose = TRUE)
```

`dist_skel` *Distribution Skeleton*

Description

Questioning This function acts as a skeleton for a truncated distribution defined by model type, maximum value and model parameters. It is designed to be used with the output from `get_dist`.

Usage

```
dist_skel(n, dist = FALSE, cum = TRUE, model, params, max_value = 120)
```

Arguments

`n`	Numeric vector, number of samples to take (or days for the probability density).
`dist`	Logical, defaults to `FALSE`. Should the probability density be returned rather than a number of samples.
`cum`	Logical, defaults to `TRUE`. If `dist = TRUE` should the returned distribution be cumulative.
`model`	Character string, defining the model to be used. Supported options are exponential ("exp"), gamma ("gamma"), and log normal ("lognorm")
`params`	A list of parameters values (by name) required for each model. For the exponential model this is a rate parameter and for the gamma model this is alpha and beta.
`max_value`	Numeric, the maximum value to allow. Defaults to 120. Samples outside of this range are resampled.

Value

A vector of samples or a probability distribution.

Examples

```
## Exponential model
# sample
dist_skel(10, model = "exp", params = list(rate = 1))

# cumulative prob density
dist_skel(1:10, model = "exp", dist = TRUE, params = list(rate = 1))

# probability density
```

```
dist_skel(1:10, model = "exp", dist = TRUE,
          cum = FALSE, params = list(rate = 1))

## Gamma model
# sample
dist_skel(10, model = "gamma", params = list(alpha = 1, beta = 2))

# cumulative prob density
dist_skel(0:10, model = "gamma", dist = TRUE,
          params = list(alpha = 1, beta = 2))

# probability density
dist_skel(0:10, model = "gamma", dist = TRUE,
          cum = FALSE, params = list(alpha = 2, beta = 2))

## Log normal model
# sample
dist_skel(10, model = "lognorm", params = list(mean = log(5), sd = log(2)))

# cumulative prob density
dist_skel(0:10, model = "lognorm", dist = TRUE,
          params = list(mean = log(5), sd = log(2)))

# probability density
dist_skel(0:10, model = "lognorm", dist = TRUE, cum = FALSE,
          params = list(mean = log(5), sd = log(2)))
```

---

epinow *Real-time Rt Estimation, Forecasting and Reporting*

Description

Maturing This function wraps the functionality of `estimate_infections()` and `forecast_infections()` in order to estimate Rt and cases by date of infection, forecast into these infections into the future. It also contains additional functionality to convert forecasts to date of report and produce summary output useful for reporting results and interpreting them. See here for an example of using `epinow` to estimate Rt for Covid-19 in a country from the ECDC data source.

Usage

```
epinow(
  reported_cases,
  generation_time,
  delays = delay_opts(),
  truncation = trunc_opts(),
  rt = rt_opts(),
  backcalc = backcalc_opts(),
  gp = gp_opts(),
```

```
  obs = obs_opts(),
  stan = stan_opts(),
  horizon = 7,
  CrIs = c(0.2, 0.5, 0.9),
  return_output = FALSE,
  output = c("samples", "plots", "latest", "fit", "timing"),
  target_folder = NULL,
  target_date,
  forecast_args = NULL,
  logs = tempdir(),
  id = "epinow",
  verbose = interactive()
)
```

Arguments

reported_cases	A data frame of confirmed cases (confirm) by date (date). confirm must be integer and date must be in date format.
generation_time	A list containing the mean, standard deviation of the mean (mean_sd), standard deviation (sd), standard deviation of the standard deviation and the maximum allowed value for the generation time (assuming a gamma distribution).
delays	A call to `delay_opts()` defining delay distributions and options. See the documentation of `delay_opts()` and the examples below for details.
truncation	**Experimental** A list of options as generated by `trunc_opts()` defining the truncation of observed data. Defaults to `trunc_opts()`. See `estimate_truncation()` for an approach to estimating truncation from data.
rt	A list of options as generated by `rt_opts()` defining Rt estimation. Defaults to `rt_opts()`. Set to `NULL` to switch to using back calculation rather than generating infections using Rt.
backcalc	A list of options as generated by `backcalc_opts()` to define the back calculation. Defaults to `backcalc_opts()`.
gp	A list of options as generated by `gp_opts()` to define the Gaussian process. Defaults to `gp_opts()`.Set to NULL to disable the Gaussian process.
obs	A list of options as generated by `obs_opts()` defining the observation model. Defaults to `obs_opts()`.
stan	A list of stan options as generated by `stan_opts()`. Defaults to `stan_opts()`. Can be used to override `data`, `init`, and `verbose` settings if desired.
horizon	Numeric, defaults to 7. Number of days into the future to forecast.
CrIs	Numeric vector of credible intervals to calculate.
return_output	Logical, defaults to FALSE. Should output be returned, this automatically updates to TRUE if no directory for saving is specified.
output	A character vector of optional output to return. Supported options are samples ("samples"), plots ("plots"), the run time ("timing"), copying the dated folder into a latest folder (if `target_folder` is not null, set using "latest"), and the

	stan fit ("fit"). The default is to return all options. This argument uses partial matching so for example passing "sam" will lead to samples being reported.
target_folder	Character string specifying where to save results (will create if not present).
target_date	Date, defaults to maximum found in the data if not specified.
forecast_args	A list of arguments to pass to forecast_infections(). Unless at a minimum a forecast_model is passed then forecast_infections will be bypassed.
logs	Character path indicating the target folder in which to store log information. Defaults to the temporary directory if not specified. Default logging can be disabled if logs is set to NULL. If specifying a custom logging setup then the code for setup_default_logging and the setup_logging function are a sensible place to start.
id	A character string used to assign logging information on error. Used by regional_epinow to assign errors to regions. Alter the default to run with error catching.
verbose	Logical, defaults to TRUE when used interactively and otherwise FALSE. Should verbose debug progress messages be printed. Corresponds to the "DEBUG" level from futile.logger. See setup_logging for more detailed logging options.

Value

A list of output from estimate_infections, forecast_infections, report_cases, and report_summary.

See Also

estimate_infections simulate_infections forecast_infections regional_epinow

Examples

```
#set number of cores to use
options(mc.cores = ifelse(interactive(), 4, 1))
# construct example distributions
generation_time <- get_generation_time(disease = "SARS-CoV-2", source = "ganyani")
incubation_period <- get_incubation_period(disease = "SARS-CoV-2", source = "lauer")
reporting_delay <- list(mean = convert_to_logmean(3, 1),
                        mean_sd = 0.1,
                        sd = convert_to_logsd(3, 1),
                        sd_sd = 0.1,
                        max = 10)

# example case data
reported_cases <- example_confirmed[1:40]

# estimate Rt and nowcast/forecast cases by date of infection
out <- epinow(reported_cases = reported_cases, generation_time = generation_time,
              rt = rt_opts(prior = list(mean = 2, sd = 0.1)),
              delays = delay_opts(incubation_period, reporting_delay))
# summary of the latest estimates
summary(out)
```

```
# plot estimates
plot(out)

# summary of R estimates
summary(out, type = "parameters", params = "R")
```

`estimates_by_report_date`

Estimate Cases by Report Date

Description

Questioning Either extracts or converts reported cases from an input data table. For output from `estimate_infections` this is a simple filtering step but for output from `forecast_infection` this is currently an approximate convolution. This step is likely to be updated/deprecated in new releases as `forecast_infections` evolves to be based on `stan` functionality.

Usage

```
estimates_by_report_date(
  estimates,
  forecast,
  delays,
  CrIs = c(0.2, 0.5, 0.9),
  target_folder = NULL,
  samples = TRUE
)
```

Arguments

`estimates`	List of data frames as output by `estimate_infections`
`forecast`	A list of data frames as output by `forecast_infections`
`delays`	A call to `delay_opts()` defining delay distributions and options. See the documentation of `delay_opts()` and the examples below for details.
`CrIs`	Numeric vector of credible intervals to calculate.
`target_folder`	Character string specifying where to save results (will create if not present).
`samples`	Logical, defaults to TRUE. Should samples be saved

Value

A list of samples and summarised estimates of estimated cases by date of report

estimate_delay *Estimate a Delay Distribution*

Description

Maturing Estimate a log normal delay distribution from a vector of integer delays. Currently this function is a simple wrapper for `bootstrapped_dist_fit`.

Usage

```
estimate_delay(delays, ...)
```

Arguments

`delays`	Integer vector of delays
`...`	Arguments to pass to internal methods.

Value

A list summarising the bootstrapped distribution

See Also

bootstrapped_dist_fit

Examples

```
delays <- rlnorm(500, log(5), 1)
estimate_delay(delays, samples = 1000, bootstraps = 10)
```

estimate_infections *Estimate Infections, the Time-Varying Reproduction Number and the Rate of Growth*

Description

Maturing Uses a non-parametric approach to reconstruct cases by date of infection from reported cases. It uses either a generative Rt model or non-parametric back calculation to estimate underlying latent infections and then maps these infections to observed cases via uncertain reporting delays and a flexible observation model. See the examples and function arguments for the details of all options. The default settings may not be sufficient for your use case so the number of warmup samples (`stan_args = list(warmup)`) may need to be increased as may the overall number of samples. Follow the links provided by any warnings messages to diagnose issues with the MCMC fit. It is recommended to explore several of the Rt estimation approaches supported as not all of them may be suited to users own use cases. See here for an example of using `estimate_infections` within the `epinow` wrapper to estimate Rt for Covid-19 in a country from the ECDC data source.

Usage

```
estimate_infections(
  reported_cases,
  generation_time,
  delays = delay_opts(),
  truncation = trunc_opts(),
  rt = rt_opts(),
  backcalc = backcalc_opts(),
  gp = gp_opts(),
  obs = obs_opts(),
  stan = stan_opts(),
  horizon = 7,
  CrIs = c(0.2, 0.5, 0.9),
  id = "estimate_infections",
  verbose = interactive()
)
```

Arguments

reported_cases	A data frame of confirmed cases (confirm) by date (date). confirm must be integer and date must be in date format.
generation_time	A list containing the mean, standard deviation of the mean (mean_sd), standard deviation (sd), standard deviation of the standard deviation and the maximum allowed value for the generation time (assuming a gamma distribution).
delays	A call to `delay_opts()` defining delay distributions and options. See the documentation of `delay_opts()` and the examples below for details.
truncation	**Experimental** A list of options as generated by `trunc_opts()` defining the truncation of observed data. Defaults to `trunc_opts()`. See `estimate_truncation()` for an approach to estimating truncation from data.
rt	A list of options as generated by `rt_opts()` defining Rt estimation. Defaults to `rt_opts()`. Set to `NULL` to switch to using back calculation rather than generating infections using Rt.
backcalc	A list of options as generated by `backcalc_opts()` to define the back calculation. Defaults to `backcalc_opts()`.
gp	A list of options as generated by `gp_opts()` to define the Gaussian process. Defaults to `gp_opts()`.Set to NULL to disable the Gaussian process.
obs	A list of options as generated by `obs_opts()` defining the observation model. Defaults to `obs_opts()`.
stan	A list of stan options as generated by `stan_opts()`. Defaults to `stan_opts()`. Can be used to override `data`, `init`, and `verbose` settings if desired.
horizon	Numeric, defaults to 7. Number of days into the future to forecast.
CrIs	Numeric vector of credible intervals to calculate.
id	A character string used to assign logging information on error. Used by `regional_epinow` to assign errors to regions. Alter the default to run with error catching.

verbose — Logical, defaults to TRUE when used interactively and otherwise FALSE. Should verbose debug progress messages be printed. Corresponds to the "DEBUG" level from futile.logger. See setup_logging for more detailed logging options.

See Also

epinow regional_epinow forecast_infections simulate_infections

Examples

```
# set number of cores to use
options(mc.cores = ifelse(interactive(), 4, 1))
# get example case counts
reported_cases <- example_confirmed[1:60]

# set up example generation time
generation_time <- get_generation_time(disease = "SARS-CoV-2", source = "ganyani")
# set delays between infection and case report
incubation_period <- get_incubation_period(disease = "SARS-CoV-2", source = "lauer")
reporting_delay <- list(mean = convert_to_logmean(3, 1), mean_sd = 0.1,
                        sd = convert_to_logsd(3, 1), sd_sd = 0.1, max = 10)

# default setting
# here we assume that the observed data is truncated by the same delay as
def <- estimate_infections(reported_cases, generation_time = generation_time,
                           delays = delay_opts(incubation_period, reporting_delay),
                           rt = rt_opts(prior = list(mean = 2, sd = 0.1)),
                           stan = stan_opts(control = list(adapt_delta = 0.95)))
# real time estimates
summary(def)
# summary plot
plot(def)

# decreasing the accuracy of the approximate Gaussian to speed up computation.
# These settings are an area of active research. See ?gp_opts for details.
agp <- estimate_infections(reported_cases, generation_time = generation_time,
                           delays = delay_opts(incubation_period, reporting_delay),
                           rt = rt_opts(prior = list(mean = 2, sd = 0.1)),
                           gp = gp_opts(ls_min = 10, basis_prop = 0.1),
                           stan = stan_opts(control = list(adapt_delta = 0.95)))
summary(agp)
plot(agp)

# Adjusting for future susceptible depletion
dep <- estimate_infections(reported_cases, generation_time = generation_time,
                           delays = delay_opts(incubation_period, reporting_delay),
                           rt = rt_opts(prior = list(mean = 2, sd = 0.1),
                                        pop = 1000000, future = "latest"),
                           gp = gp_opts(ls_min = 10, basis_prop = 0.1), horizon = 21,
                           stan = stan_opts(control = list(adapt_delta = 0.95)))
```

```
plot(dep)

# Adjusting for truncation of the most recent data
# See estimate_truncation for an approach to estimating this from data
trunc_dist <- list(mean = convert_to_logmean(0.5, 0.5), mean_sd = 0.1,
                   sd = convert_to_logsd(0.5, 0.5), sd_sd = 0.1,
                   max = 3)
trunc <- estimate_infections(reported_cases, generation_time = generation_time,
                             delays = delay_opts(incubation_period, reporting_delay),
                             truncation = trunc_opts(trunc_dist),
                             rt = rt_opts(prior = list(mean = 2, sd = 0.1)),
                             gp = gp_opts(ls_min = 10, basis_prop = 0.1),
                             stan = stan_opts(control = list(adapt_delta = 0.95)))
plot(trunc)

# using back calculation (combined here with under reporting)
# this model is in the order of 10 ~ 100 faster than the gaussian process method
# it is likely robust for retrospective Rt but less reliable for real time estimates
# the width of the prior window controls the reliance on observed data and can be
# optionally switched off using backcalc_opts(prior = "none"), see ?backcalc_opts for
# other options
backcalc <- estimate_infections(reported_cases, generation_time = generation_time,
                                delays = delay_opts(incubation_period, reporting_delay),
                                rt = NULL, backcalc = backcalc_opts(),
                                obs = obs_opts(scale = list(mean = 0.4, sd = 0.05)),
                                horizon = 0)
plot(backcalc)

# Rt projected into the future using the Gaussian process
project_rt <- estimate_infections(reported_cases, generation_time = generation_time,
                                delays = delay_opts(incubation_period, reporting_delay),
                                  rt = rt_opts(prior = list(mean = 2, sd = 0.1),
                                               future = "project"))
plot(project_rt)

# default settings on a later snapshot of data
snapshot_cases <- example_confirmed[80:130]
snapshot <- estimate_infections(snapshot_cases, generation_time = generation_time,
                                delays = delay_opts(incubation_period, reporting_delay),
                                rt = rt_opts(prior = list(mean = 1, sd = 0.1)))
plot(snapshot)

# stationary Rt assumption (likely to provide biased real-time estimates)
stat <- estimate_infections(reported_cases, generation_time = generation_time,
                            delays = delay_opts(incubation_period, reporting_delay),
                           rt = rt_opts(prior = list(mean = 2, sd = 0.1), gp_on = "R0"))
plot(stat)

# no gaussian process (i.e fixed Rt assuming no breakpoints)
fixed <- estimate_infections(reported_cases, generation_time = generation_time,
                             delays = delay_opts(incubation_period, reporting_delay),
                             gp = NULL)
plot(fixed)
```

```
# no delays
no_delay <- estimate_infections(reported_cases, generation_time = generation_time)
plot(no_delay)

# break point but otherwise static Rt
bp_cases <- data.table::copy(reported_cases)
bp_cases <- bp_cases[, breakpoint := ifelse(date == as.Date("2020-03-16"), 1, 0)]
bkp <- estimate_infections(bp_cases, generation_time = generation_time,
                           delays = delay_opts(incubation_period, reporting_delay),
                           rt = rt_opts(prior = list(mean = 2, sd = 0.1)),
                           gp = NULL)
# break point effect
summary(bkp, type = "parameters", params = "breakpoints")
plot(bkp)

# weekly random walk
rw <- estimate_infections(reported_cases, generation_time = generation_time,
                          delays = delay_opts(incubation_period, reporting_delay),
                          rt = rt_opts(prior = list(mean = 2, sd = 0.1), rw = 7),
                          gp = NULL)

# random walk effects
summary(rw, type = "parameters", params = "breakpoints")
plot(rw)
```

`estimate_secondary` *Estimate a Secondary Observation from a Primary Observation*

Description

Experimental Estimates the relationship between a primary and secondary observation, for example hospital admissions and deaths or hospital admissions and bed occupancy. See `secondary_opts()` for model structure options. See parameter documentation for model defaults and options. See the examples for case studies using synthetic data and here for an example of forecasting Covid-19 deaths from Covid-19 cases. See here for a prototype function that may be used to estimate and forecast a secondary observation from a primary across multiple regions and here for an application forecasting Covid-19 deaths in Germany and Poland.

Usage

```
estimate_secondary(
  reports,
  secondary = secondary_opts(),
  delays = delay_opts(list(mean = 2.5, mean_sd = 0.5, sd = 0.47, sd_sd = 0.25, max =
    30)),
  truncation = trunc_opts(),
  obs = obs_opts(),
```

```
  burn_in = 14,
  CrIs = c(0.2, 0.5, 0.9),
  model = NULL,
  verbose = interactive(),
  ...
)
```

Arguments

reports	A data frame containing the date of report and both primary and secondary reports.
secondary	A call to secondary_opts() or a list containing the following binary variables: cumulative, historic, primary_hist_additive, current, primary_current_additive. These parameters control the structure of the secondary model, see secondary_opts() for details.
delays	A call to delay_opts() defining delay distributions between primary and secondary observations See the documentation of delay_opts() for details. BY default a diffuse prior is assumed with a mean of 14 days and standard deviation of 7 days (with a standard deviation of 0.5 and 0.25 respectively on the log scale).
truncation	**Experimental** A list of options as generated by trunc_opts() defining the truncation of observed data. Defaults to trunc_opts(). See estimate_truncation() for an approach to estimating truncation from data.
obs	A list of options as generated by obs_opts() defining the observation model. Defaults to obs_opts().
burn_in	Integer, defaults to 14 days. The number of data points to use for estimation but not to fit to at the beginning of the time series. This must be less than the number of observations.
CrIs	Numeric vector of credible intervals to calculate.
model	A compiled stan model to override the default model. May be useful for package developers or those developing extensions.
verbose	Logical, should model fitting progress be returned. Defaults to interactive().
...	Additional parameters to pass to rstan::sampling.

Value

A list containing: predictions (a data frame ordered by date with the primary, and secondary observations, and a summary of the model estimated secondary observations), data (a list of data used to fit the model), and fit (the stanfit object).

Examples

```
#set number of cores to use
options(mc.cores = ifelse(interactive(), 4, 1))
#' # load data.table for manipulation
library(data.table)
```

```
# load lubridate for dates
library(lubridate)

#### Incidence data example ####

# make some example secondary incidence data
cases <- example_confirmed
cases <- as.data.table(cases)

# apply a convolution of a log normal to a vector of observations
weight_cmf <- function(x, ...) {
   set.seed(x[1])
   meanlog <- rnorm(1, 1.6, 0.2)
   sdlog <- rnorm(1, 0.8, 0.1)
   cmf <- cumsum(dlnorm(1:length(x), meanlog, sdlog)) -
           cumsum(dlnorm(0:(length(x) - 1), meanlog, sdlog))
   conv <- sum(x * rev(cmf), na.rm = TRUE)
   conv <- round(conv, 0)
 return(conv)
}
# roll over observed cases to produce a convolution
cases <- cases[, .(date, primary = confirm, secondary = confirm)]
cases <- cases[, secondary := frollapply(secondary, 15, weight_cmf, align = "right")]
cases <- cases[!is.na(secondary)]
# add a day of the week effect and scale secondary observations at 40% of primary
cases <- cases[lubridate::wday(date) == 1, secondary := round(0.5 * secondary, 0)]
cases <- cases[, secondary := round(secondary * rnorm(.N, 0.4, 0.025), 0)]
cases <- cases[secondary < 0, secondary := 0]

# fit model to example data assuming only a given fraction of primary observations
# become secondary observations
inc <- estimate_secondary(cases[1:60],
                          obs = obs_opts(scale = list(mean = 0.2, sd = 0.2)))
plot(inc, primary = TRUE)

# forecast future secondary cases from primary
inc_preds <- forecast_secondary(inc, cases[61:.N][, value := primary])
plot(inc_preds, new_obs = cases, from = "2020-05-01")

#### Prevalence data example ####

# make some example prevalence data
cases <- example_confirmed
cases <- as.data.table(cases)
cases <- cases[, .(date, primary = confirm,
                   scaled_primary = confirm * rnorm(.N, 0.4, 0.05))]
cases$secondary <- 0
cases$secondary[1] <- as.integer(cases$scaled_primary[1])
for (i in 2:nrow(cases)) {
  meanlog <- rnorm(1, 1.6, 0.1)
  sdlog <- rnorm(1, 0.8, 0.05)
  cmf <- cumsum(dlnorm(1:min(i-1,40), meanlog, sdlog)) -
           cumsum(dlnorm(0:min(39,i-2), meanlog, sdlog))
```

```
  reducing_cases <- sum(cases$scaled_primary[(i-1):max(1,i-20)] * cmf)
  reducing_cases <- ifelse(cases$secondary[i - 1] < reducing_cases,
                           cases$secondary[i - 1], reducing_cases)
  cases$secondary[i] <- as.integer(
  cases$secondary[i - 1] + cases$scaled_primary[i] - reducing_cases
  )
  cases$secondary[i] <- ifelse(cases$secondary[i] < 0, 0,
                               cases$secondary[i])
}
# fit model to example prevalence data
prev <- estimate_secondary(cases[1:100], secondary = secondary_opts(type = "prevalence"),
                           obs = obs_opts(week_effect = FALSE,
                                          scale = list(mean = 0.3, sd = 0.1)))
plot(prev, primary = TRUE)

# forecast future secondary cases from primary
prev_preds <- forecast_secondary(prev, cases[101:.N][, value := primary])
plot(prev_preds, new_obs = cases, from = "2020-06-01")
```

`estimate_truncation` *Estimate Truncation of Observed Data*

Description

Experimental Estimates a truncation distribution from multiple snapshots of the same data source over time. This distribution can then be used in `regional_epinow`, `epinow`, and `estimate_infections` to adjust for truncated data. See here for an example of using this approach on Covid-19 data in England.

The model of truncation is as follows:

1. The truncation distribution is assumed to be log normal with a mean and standard deviation that is informed by the data.
2. The data set with the latest observations is adjusted for truncation using the truncation distribution.
3. Earlier data sets are recreated by applying the truncation distribution to the adjusted latest observations in the time period of the earlier data set. These data sets are then compared to the earlier observations assuming a negative binomial observation model.

This model is then fit using `stan` with standard normal, or half normal, prior for the mean, standard deviation and 1 over the square root of the over dispersion.

This approach assumes that:

- Current truncation is related to past truncation.
- Truncation is a multiplicative scaling of underlying reported cases.
- Truncation is log normally distributed.

Usage

```
estimate_truncation(
  obs,
  max_truncation = 10,
  model = NULL,
  CrIs = c(0.2, 0.5, 0.9),
  verbose = TRUE,
  ...
)
```

Arguments

obs	A list of data frames each containing a date variable and a confirm (integer) variable. Each data set should be a snapshot of the reported data over time. All data sets must contain a complete vector of dates.
max_truncation	Integer, defaults to 10. Maximum number of days to include in the truncation distribution.
model	A compiled stan model to override the default model. May be useful for package developers or those developing extensions.
CrIs	Numeric vector of credible intervals to calculate.
verbose	Logical, should model fitting progress be returned.
...	Additional parameters to pass to rstan::sampling.

Value

A list containing: the summary parameters of the truncation distribution (dist), the estimated CMF of the truncation distribution (cmf, can be used to adjusted new data), a data frame containing the observed truncated data, latest observed data and the adjusted for truncation observations (obs), a data frame containing the last observed data (last_obs, useful for plotting and validation), the data used for fitting (data) and the fit object (fit).

Examples

```
#set number of cores to use
options(mc.cores = ifelse(interactive(), 4, 1))
# get example case counts
reported_cases <- example_confirmed[1:60]

# define example truncation distribution (note not integer adjusted)
trunc_dist <- list(mean = convert_to_logmean(3, 2),
                   mean_sd = 0.1,
                   sd = convert_to_logsd(3, 2),
                   sd_sd = 0.1,
                   max = 10)

# apply truncation to example data
construct_truncation <- function(index, cases, dist) {
set.seed(index)
```

```
  cmf <- cumsum(
     dlnorm(1:(dist$max + 1),
            rnorm(1, dist$mean, dist$mean_sd),
            rnorm(1, dist$sd, dist$sd_sd)))
  cmf <- cmf / cmf[dist$max + 1]
  cmf <- rev(cmf)[-1]
  trunc_cases <- data.table::copy(cases)[1:(.N - index)]
  trunc_cases[(.N - length(cmf) + 1):.N, confirm := as.integer(confirm * cmf)]
  return(trunc_cases)
 }
example_data <- purrr::map(c(20, 15, 10, 0),
                           construct_truncation,
                           cases = reported_cases,
                           dist = trunc_dist)

# fit model to example data
est <- estimate_truncation(example_data, verbose = interactive(),
                           chains = 2, iter = 2000)

# summary of the distribution
est$dist
# summary of the estimated truncation cmf (can be applied to new data)
print(est$cmf)
# observations linked to truncation adjusted estimates
print(est$obs)
# validation plot of observations vs estimates
plot(est)
```

`example_confirmed` *Example Confirmed Case Data Set*

Description

Stable An example data frame of observed cases

Usage

```
example_confirmed
```

Format

A data frame containing cases reported on each date.

expose_stan_fns *Expose internal package stan functions in R*

Description

Stable his function exposes internal stan functions in R from a user supplied list of target files. Allows for testing of stan functions in R and potentially user use in R code.

Usage

```
expose_stan_fns(files, target_dir, ...)
```

Arguments

files	A character vector indicating the target files
target_dir	A character string indicating the target directory for the file
...	Additional arguments passed to rstan::expose_stan_functions.

Examples

```
expose_stan_fns("rt.stan", target_dir = system.file("stan/functions", package = "EpiNow2"))

# test by updating Rt
update_Rt(rep(1, 10), log(1.2), rep(0.1, 9), rep(10, 0), numeric(0), 0)
```

extract_CrIs *Extract Credible Intervals Present*

Description

Stable Helper function to extract the credible intervals present in a data frame.

Usage

```
extract_CrIs(summarised)
```

Arguments

summarised	A data frame as processed by calc_CrIs

Value

A numeric vector of credible intervals detected in the data frame.

Examples

```
samples <- data.frame(value = 1:10, type = "car")
summarised <- calc_CrIs(samples, summarise_by = "type",
                        CrIs = c(seq(0.05, 0.95, 0.05)))
extract_CrIs(summarised)
```

---

extract_inits *Generate initial conditions from a Stan fit*

Description

Experimental Extracts posterior samples to use to initialise a full model fit. This may be useful for certain data sets where the sampler gets stuck or cannot easily be initialised. In `estimate_infections()`, `epinow()` and `regional_epinow()` this option can be engaged by setting stan_opts(init_fit = <stanfit>).

This implementation is based on the approach taken in epidemia authored by James Scott.

Usage

```
extract_inits(fit, current_inits, exclude_list = NULL, samples = 50)
```

Arguments

`fit`	A stanfit object
`current_inits`	A function that returns a list of initial conditions (such as `create_initial_conditions()`). Only used in `exclude_list` is specified.
`exclude_list`	A character vector of parameters to not initialise from the fit object, defaulting to `NULL`.
`samples`	Numeric, defaults to 50. Number of posterior samples.

Value

A function that when called returns a set of initial conditions as a named list.

---

extract_parameter *Extract Samples for a Parameter from a Stan model*

Description

Stable Extracts a single from a list of `stan` output and returns it as a `data.table`.

Usage

```
extract_parameter(param, samples, dates)
```

Arguments

param	Character string indicating the parameter to extract
samples	Extracted stan model (using rstan::extract)
dates	A vector identifying the dimensionality of the parameter to extract. Generally this will be a date

Value

A data frame containing the parameter name, date, sample id and sample value

---

extract_parameter_samples

Extract Parameter Samples from a Stan Model

Description

Stable Extracts a custom set of parameters from a stan object and adds stratification and dates where appropriate.

Usage

```
extract_parameter_samples(
  stan_fit,
  data,
  reported_dates,
  reported_inf_dates,
  drop_length_1 = FALSE,
  merge = FALSE
)
```

Arguments

stan_fit	A fit Stan model as returned by rstan:sampling
data	A list of the data supplied to the rstan::sampling call.
reported_dates	A vector of dates to report estimates for.
reported_inf_dates	A vector of dates to report infection estimates for.
drop_length_1	Logical; whether the first dimension should be dropped if it is if length 1; this is necessary when processing simulation results
merge	if TRUE, merge samples and data so that parameters can be extracted from data

Value

A list of dataframes each containing the posterior of a parameter

extract_stan_param *Extract a Parameter Summary from a Stan Object*

Description

Stable Extracts summarised parameter posteriors from a `stanfit` object using `rstan::summary` in a format consistent with other summary functions in `EpiNow2`.

Usage

```
extract_stan_param(
  fit,
  params = NULL,
  CrIs = c(0.2, 0.5, 0.9),
  var_names = FALSE
)
```

Arguments

`fit`	A `stanfit` object
`params`	A character vector of parameters to extract. Defaults to all parameters.
`CrIs`	Numeric vector of credible intervals to calculate.
`var_names`	Logical defaults to `FALSE`. Should variables be named. Automatically set to TRUE if multiple parameters are to be extracted.

Value

A `data.table` summarising parameter posteriors. Contains a following variables: `variable`, `mean`, `mean_se`, `sd`, `median`, and `lower_`, `upper_` followed by credible interval labels indicating the credible intervals present.

extract_static_parameter *Extract Samples from a Parameter with a Single Dimension*

Description

Extract Samples from a Parameter with a Single Dimension

Usage

```
extract_static_parameter(param, samples)
```

Arguments

param	Character string indicating the parameter to extract
samples	Extracted stan model (using rstan::extract)

Value

A data frame containing the parameter name, sample id and sample value

---

filter_opts *Filter Options for a Target Region*

Description

Maturing A helper function that allows the selection of region specific settings if present and otherwise applies the overarching settings

Usage

```
filter_opts(opts, region)
```

Arguments

opts	Either a list of calls to an _opts function or a single call to an _opts function.
region	A character string indicating a region of interest.

Value

A list of options

Examples

```
# uses example case vector
cases <- example_confirmed[1:40]
cases <- data.table::rbindlist(list(
  data.table::copy(cases)[, region := "testland"],
  cases[, region := "realland"]))

# regional options
regional_opts <- opts_list(rt_opts(), cases)
EpiNow2:::filter_opts(regional_opts, "realland")
# default only
EpiNow2:::filter_opts(rt_opts(), "realland")
#settings are NULL in one regions
regional_opts <- update_list(regional_opts, list(realland = NULL))
EpiNow2:::filter_opts(regional_opts, "realland")
```

fit_model_with_nuts *Fit a Stan Model using the NUTs sampler*

Description

Maturing Fits a stan model using `rstan::sampling`. Provides the optional ability to run chains using `future` with error catching, timeouts and merging of completed chains.

Usage

```
fit_model_with_nuts(
  args,
  future = FALSE,
  max_execution_time = Inf,
  id = "stan"
)
```

Arguments

`args`	List of stan arguments
`future`	Logical, defaults to `FALSE`. Should `future` be used to run stan chains in parallel.
`max_execution_time`	Numeric, defaults to Inf. What is the maximum execution time per chain in seconds. Results will still be returned as long as at least 2 chains complete successfully within the timelimit.
`id`	A character string used to assign logging information on error. Used by `regional_epinow` to assign errors to regions. Alter the default to run with error catching.

Value

A stan model object

fit_model_with_vb *Fit a Stan Model using Variational Inference*

Description

Maturing Fits a stan model using variational inference.

Usage

```
fit_model_with_vb(args, future = FALSE, id = "stan")
```

Arguments

args	List of stan arguments
future	Logical, defaults to FALSE. Should future be used to run stan chains in parallel.
id	A character string used to assign logging information on error. Used by regional_epinow to assign errors to regions. Alter the default to run with error catching.

Value

A stan model object

---

forecast_infections *Forecast Infections and the Time-Varying Reproduction Number*

Description

Experimental Provides optional tools for forecasting cases and Rt estimates using the timeseries methods (via the EpiSoon package). It requires the Episoon package. Installation instructions for the EpiSoon package are available here.

Usage

```
forecast_infections(
  infections,
  rts,
  gt_mean,
  gt_sd,
  gt_max = 30,
  ensemble_type = "mean",
  forecast_model,
  CrIs = c(0.2, 0.5, 0.9),
  horizon = 14,
  samples = 1000
)
```

Arguments

infections	A data frame of cases by date of infection containing the following variables: date, mean, sd
rts	A data frame of Rt estimates by date of infection containing the following variables: date, mean, sd
gt_mean	Numeric, the mean of the gamma distributed generation time.
gt_sd	Numeric, the standard deviation of the gamma distributed generation time.
gt_max	Numeric, the maximum allowed value of the gamma distributed generation time.
ensemble_type	Character string indicating the type of ensemble to use. By default this is an unweighted ensemble ("mean") with no other types currently supported.

forecast_model	An uninitialised forecast model function to be passed to EpiSoon::forecast_rt. Used for forecasting future Rt and case co An example of the required structure is: function(ss,y){bsts::AddSemilocalLinearTrend(ss,y = y)}.
CrIs	Numeric vector of credible intervals to calculate.
horizon	Numeric, defaults to 14. The horizon over which to forecast Rts and cases.
samples	Numeric, the number of forecast samples to take.

Value

A list of data.tables. The first entry ("samples") contains raw forecast samples and the second entry ("summarised") contains summarised forecasts.

Examples

```
if(requireNamespace("EpiSoon")){
   if(requireNamespace("forecastHybrid")){
# example case data
reported_cases <- example_confirmed[1:40]

generation_time <- get_generation_time(disease = "SARS-CoV-2", source = "ganyani")
incubation_period <- get_incubation_period(disease = "SARS-CoV-2", source = "lauer")
reporting_delay <- estimate_delay(rlnorm(100, log(6), 1), max_value = 15)

# estimate Rt and infections from data
out <- estimate_infections(reported_cases, generation_time = generation_time,
                           delays = delay_opts(incubation_period, reporting_delay),
                           rt = rt_opts(prior = list(mean = 2, sd = 0.1)))

# forecast Rt and infections from estimates
forecast <- forecast_infections(
    infections = out$summarised[variable == "infections"],
    rts = out$summarised[variable == "R"],
     gt_mean = out$summarised[variable == "gt_mean"]$mean,
     gt_sd = out$summarised[variable == "gt_sd"]$mean,
     gt_max = 30,
     forecast_model = function(y, ...){
       EpiSoon::forecastHybrid_model(y = y[max(1, length(y) - 21):length(y)],
       model_params = list(models = "aefz", weights = "equal"),
       forecast_params = list(PI.combination = "mean"), ...)},
     horizon = 14,
     samples = 1000)

forecast$summarised
  }
 }
```

forecast_secondary *Forecast Secondary Observations Given a Fit from estimate_secondary*

Description

Experimental This function forecasts secondary observations using the output of `estimate_secondary()` and either observed primary data or a forecast of primary observations. See the examples of `estimate_secondary()` for one use case. It can also be combined with `estimate_infections()` to produce a forecast for a secondary observation from a forecast of a primary observation. See the examples of `estimate_secondary()` for example use cases on synthetic data. See here for an example of forecasting Covid-19 deaths from Covid-19 cases.

Usage

```
forecast_secondary(
  estimate,
  primary,
  primary_variable = "reported_cases",
  model = NULL,
  samples = NULL,
  all_dates = FALSE,
  CrIs = c(0.2, 0.5, 0.9)
)
```

Arguments

`estimate`	An object of class "estimate_secondary" as produced by `estimate_secondary()`.
`primary`	A data.frame containing at least `date` and `value` (integer) variables and optionally `sample`. Used as the primary observation used to forecast the secondary observations. Alternatively, this may be an object of class "estimate_infections" as produced by `estimate_infections()`. If `primary` is of class "estimate_infections" then the internal samples will be filtered to have a minimum date ahead of those observed in the `estimate` object.
`primary_variable`	A character string indicating the primary variable, defaulting to "reported_cases". Only used when primary is of class "estimate_infections".
`model`	A compiled stan model as returned by `rstan::stan_model`.
`samples`	Numeric, number of posterior samples to simulate from. The default is to use all samples in the `primary` input when present. If not present the default is to use 1000 samples.
`all_dates`	Logical, defaults to FALSE. Should a forecast for all dates and not just those in the forecast horizon be returned.
`CrIs`	Numeric vector of credible intervals to calculate.

Value

A list containing: `predictions` (a data frame ordered by date with the primary, and secondary observations, and a summary of the forecast secondary observations. For primary observations in the forecast horizon when uncertainty is present the median is used), `samples` a data frame of forecast secondary observation posterior samples, and `forecast` a summary of the forecast secondary observation posterior.

See Also

estimate_secondary

`format_fit` *Format Posterior Samples*

Description

Stable Summaries posterior samples and adds additional custom variables.

Usage

```
format_fit(posterior_samples, horizon, shift, burn_in, start_date, CrIs)
```

Arguments

`posterior_samples`	A list of posterior samples as returned by `extract_parameter_samples`
`horizon`	Numeric, forecast horizon
`shift`	Numeric, the shift to apply to estimates
`burn_in`	Numeric, number of days to discard estimates for
`start_date`	Date, earliest date with data
`CrIs`	Numeric vector of credible intervals to calculate.

Value

A list of samples and summarised posterior parameter estimates

`gamma_dist_def` *Generate a Gamma Distribution Definition Based on Parameter Estimates*

Description

Soft-deprecated Generates a distribution definition when only parameter estimates are available for gamma distributed parameters. See `rgamma` for distribution information.

Usage

```
gamma_dist_def(
  shape,
  shape_sd,
  scale,
  scale_sd,
  mean,
  mean_sd,
  sd,
  sd_sd,
  max_value,
  samples
)
```

Arguments

`shape`	Numeric, shape parameter of the gamma distribution.
`shape_sd`	Numeric, standard deviation of the shape parameter.
`scale`	Numeric, scale parameter of the gamma distribution.
`scale_sd`	Numeric, standard deviation of the scale parameter.
`mean`	Numeric, log mean parameter of the gamma distribution.
`mean_sd`	Numeric, standard deviation of the log mean parameter.
`sd`	Numeric, log sd parameter of the gamma distribution.
`sd_sd`	Numeric, standard deviation of the log sd parameter.
`max_value`	Numeric, the maximum value to allow. Defaults to 120. Samples outside of this range are resampled.
`samples`	Numeric, number of sample distributions to generate.

Value

A data.table defining the distribution as used by `dist_skel`

Examples

```
# using estimated shape and scale
def <- gamma_dist_def(shape = 5.807, shape_sd = 0.2,
               scale = 0.9, scale_sd = 0.05,
               max_value = 20, samples = 10)
print(def)
def$params[[1]]

# using mean and sd
def <- gamma_dist_def(mean = 3, mean_sd = 0.5,
               sd = 3, sd_sd = 0.1,
               max_value = 20, samples = 10)
print(def)
def$params[[1]]
```

---

generation_times *Literature Estimates of Generation Times*

Description

Stable Generation time estimates. See here for details: https://github.com/epiforecasts/EpiNow2/blob/master/data-raw/generation-time.R

Usage

```
generation_times
```

Format

A `data.table` of summarising the distribution

---

get_dist *Get a Literature Distribution*

Description

Stable Search a data frame for a distribution and return it in the format expected by `delay_opts` and the `generation_time` argument of `epinow` and `estimate_infections`.

Usage

```
get_dist(data, disease, source, max_value = 15)
```

Arguments

data	A data.table in the format of generation_times.
disease	A character string indicating the disease of interest.
source	A character string indicating the source of interest.
max_value	Numeric, the maximum value to allow. Defaults to 15 days.

Value

A list defining a distribution

Examples

```
get_dist(EpiNow2::generation_times, disease = "SARS-CoV-2", source = "ganyani")
```

---

get_generation_time *Get a Literature Distribution for the Generation Time*

Description

Stable Extracts a literature distribution from generation_times

Usage

```
get_generation_time(disease, source, max_value = 15)
```

Arguments

disease	A character string indicating the disease of interest.
source	A character string indicating the source of interest.
max_value	Numeric, the maximum value to allow. Defaults to 15 days.

Value

A list defining a distribution

Examples

```
get_generation_time(disease = "SARS-CoV-2", source = "ganyani")
```

get_incubation_period *Get a Literature Distribution for the Incubation Period*

Description

Stable Extracts a literature distribution from `incubation_periods`

Usage

```
get_incubation_period(disease, source, max_value = 15)
```

Arguments

`disease`	A character string indicating the disease of interest.
`source`	A character string indicating the source of interest.
`max_value`	Numeric, the maximum value to allow. Defaults to 15 days.

Value

A list defining a distribution

Examples

```
get_incubation_period(disease = "SARS-CoV-2", source = "lauer")
```

get_raw_result *Get a Single Raw Result*

Description

Stable

Usage

```
get_raw_result(file, region, date, result_dir)
```

Arguments

`file`	Character string giving the result files name.
`region`	Character string giving the region of interest.
`date`	Target date (in the format "yyyy-mm-dd).
`result_dir`	Character string giving the location of the target directory

Value

An R object read in from the targeted .rds file

`get_regional_results` *Get Combined Regional Results*

Description

Stable Summarises results across regions either from input or from disk. See the examples for details.

Usage

```
get_regional_results(
  regional_output,
  results_dir,
  date,
  samples = TRUE,
  forecast = FALSE
)
```

Arguments

`regional_output`	A list of output as produced by `regional_epinow` and stored in the `regional` list.
`results_dir`	A character string indicating the folder containing the `EpiNow2` results to extract.
`date`	A Character string (in the format "yyyy-mm-dd") indicating the date to extract data for. Defaults to "latest" which finds the latest results available.
`samples`	Logical, defaults to `TRUE`. Should samples be returned.
`forecast`	Logical, defaults to `FALSE`. Should forecast results be returned.

Value

A list of estimates, forecasts and estimated cases by date of report.

Examples

```
# construct example distributions
generation_time <- get_generation_time(disease = "SARS-CoV-2", source = "ganyani")
incubation_period <- get_incubation_period(disease = "SARS-CoV-2", source = "lauer")
reporting_delay <- estimate_delay(rlnorm(100, log(6), 1), max_value = 10)

# example case vector from EpiSoon
cases <- example_confirmed[1:30]
cases <- data.table::rbindlist(list(
  data.table::copy(cases)[, region := "testland"],
  cases[, region := "realland"]))
```

```
# save results to tmp folder
dir <- file.path(tempdir(check = TRUE), "results")
# run multiregion estimates
regional_out <- regional_epinow(reported_cases = cases,
                                generation_time = generation_time,
                                delays = delay_opts(incubation_period, reporting_delay),
                                rt = rt_opts(rw = 7), gp = NULL,
                                output = c("regions", "latest"),
                                target_folder = dir,
                                return_output = TRUE)
# from output
results <- get_regional_results(regional_out$regional, samples = FALSE)
names(results)

# from a folder
folder_results <- get_regional_results(results_dir = dir, samples = FALSE)
names(folder_results)
```

`get_regions` *Get Folders with Results*

Description

Stable

Usage

```
get_regions(results_dir)
```

Arguments

`results_dir`	A character string giving the directory in which results are stored (as produced by `regional_rt_pipeline`).

Value

A named character vector containing the results to plot.

get_regions_with_most_reports *Get Regions with Most Reported Cases*

Description

Stable Extract a vector of regions with the most reported cases in a set time window.

Usage

```
get_regions_with_most_reports(reported_cases, time_window = 7, no_regions = 6)
```

Arguments

reported_cases	A data frame of confirmed cases (confirm) by date (date), and region (region).
time_window	Numeric, number of days to include from latest date in data. Defaults to 7 days.
no_regions	Numeric, number of regions to return. Defaults to 6.

Value

A character vector of regions with the highest reported cases

global_map *Generate a global map for a single variable.*

Description

Questioning This general purpose function can be used to generate a global map for a single variable. It has few defaults but the data supplied must contain a country variable for linking to mapping data. This function requires the installation of the rnaturalearth package. Status of this function is currently questioning as it is uncertain if it is in use. Future releases may depreciate it.

Usage

```
global_map(
  data = NULL,
  variable = NULL,
  variable_label = NULL,
  trans = "identity",
  fill_labels = NULL,
  scale_fill = NULL,
  ...
)
```

Arguments

data	Dataframe containing variables to be mapped. Must contain a country variable.
variable	A character string indicating the variable to map data for. This must be supplied.
variable_label	A character string indicating the variable label to use. If not supplied then the underlying variable name is used.
trans	A character string specifying the transform to use on the specified metric. Defaults to no transform ("identity"). Other options include log scaling ("log") and log base 10 scaling ("log10"). For a complete list of options see ggplot2::continous_scale.
fill_labels	A function to use to allocate legend labels. An example (used below) is scales::percent, which can be used for percentage data.
scale_fill	Function to use for scaling the fill. Defaults to a custom ggplot2::scale_fill_manual, which expects the possible values to be "Increasing", "Likely increasing", "Likely decreasing", "Decreasing" or "Unsure".
...	Additional arguments passed to the scale_fill function

Value

A ggplot2 object containing a global map.

Examples

```
if(requireNamespace("rnaturalearth") & requireNamespace("scales")){
# Example 1 - categorical data
# If values are "Increasing", "Likely increasing" etc (see ?EpiNow2::theme_map),
# then the default fill scale works
eg_data <- data.table::data.table(variable = c("Increasing",
                                               "Decreasing",
                                               "Unsure",
                                               "Likely decreasing",
                                               "Likely increasing"),
                                  country = c("France",
                                              "Germany",
                                              "United Kingdom",
                                              "Spain",
                                              "Australia") )
# make variable a factor so the ordering is sensible in the legend
eg_data$variable <- factor(eg_data$variable, levels = c("Decreasing", "Likely decreasing",
                                                        "Unsure", "Likely increasing",
                                                        "Increasing"))
global_map(eg_data, variable = "variable", variable_label = "Direction\nof change")


# Example 2 - numeric data
# numeric data requires scale_fill and a global viridis_palette specified
eg_data$second_variable <- runif(nrow(eg_data))
viridis_palette <- "A"
global_map(eg_data, variable = "second_variable", scale_fill = scale_fill_viridis_c)
}
```

gp_opts *Approximate Gaussian Process Settings*

Description

Stable Defines a list specifying the structure of the approximate Gaussian process. Custom settings can be supplied which override the defaults.

Usage

```
gp_opts(
  basis_prop = 0.2,
  boundary_scale = 1.5,
  ls_mean = 21,
  ls_sd = 7,
  ls_min = 3,
  ls_max = 60,
  alpha_sd = 0.1,
  kernel = "matern",
  matern_type = 3/2
)
```

Arguments

basis_prop	Numeric, proportion of time points to use as basis functions. Defaults to 0.1. Decreasing this value results in a decrease in accuracy but a faster compute time (with increasing it having the first effect). In general smaller posterior length scales require a higher proportion of basis functions. See (Riutort-Mayol et al. 2020 https://arxiv.org/abs/2004.11408) for advice on updating this default. This setting is an area of active research.
boundary_scale	Numeric, defaults to 1.5. Boundary scale of the approximate Gaussian process. See (Riutort-Mayol et al. 2020 https://arxiv.org/abs/2004.11408) for advice on updating this default.
ls_mean	Numeric, defaults to 21 days. The mean of the lognormal length scale.
ls_sd	Numeric, defaults to 7 days. The standard deviation of the log normal length scale with..
ls_min	Numeric, defaults to 7. The minimum value of the length scale.
ls_max	Numeric, defaults to 60. The maximum value of the length scale. Updated in create_gp_data to be the length of the input data if this is smaller.
alpha_sd	Numeric, defaults to 0.2. The standard deviation of the magnitude parameter of the Gaussian process kernel. Should be approximately the expected standard deviation of the logged Rt.
kernel	Character string, the type of kernel required. Currently supporting the squared exponential kernel ("se") and the 3 over 2 Matern kernel ("matern", with matern_type = 3/2). Defaulting to the Matern 3 over 2 kernel as discontinuities are expected in Rt and infections.

matern_type	Numeric, defaults to 3/2. Type of Matern Kernel to use. Currently only the Matern 3/2 kernel is supported.

Value

A list of settings defining the Gaussian process

Examples

```
# default settings
gp_opts()

# add a custom length scale
gp_opts(ls_mean = 4)
```

---

growth_to_R *Convert Growth Rates to Reproduction numbers.*

Description

Questioning See here for justification. Now handled internally by stan so may be removed in future updates if no user demand.

Usage

```
growth_to_R(r, gamma_mean, gamma_sd)
```

Arguments

r	Numeric, rate of growth estimates
gamma_mean	Numeric, mean of the gamma distribution
gamma_sd	Numeric, standard deviation of the gamma distribution

Value

Numeric vector of reproduction number estimates

Examples

```
growth_to_R(0.2, 4, 1)
```

incubation_periods *Literature Estimates of Incubation Periods*

Description

Stable Incubation period estimates. See here for details: https://github.com/epiforecasts/EpiNow2/blob/master/data-raw/incubation-period.R

Usage

```
incubation_periods
```

Format

A `data.table` of summarising the distribution

init_cumulative_fit *Generate initial conditions by fitting to cumulative cases*

Description

Experimental Fits a model to cumulative cases. This may be a useful approach to initialising a full model fit for certain data sets where the sampler gets stuck or cannot easily be initialised as fitting to cumulative cases changes the shape of the posterior distribution. In `estimate_infections()`, `epinow()` and `regional_epinow()` this option can be engaged by setting `stan_opts(init_fit = "cumulative")`.

This implementation is based on the approach taken in epidemia authored by James Scott.

Usage

```
init_cumulative_fit(
  args,
  samples = 50,
  warmup = 50,
  id = "init",
  verbose = FALSE
)
```

Arguments

`args`	List of stan arguments
`samples`	Numeric, defaults to 50. Number of posterior samples.
`warmup`	Numeric, defaults to 50. Number of warmup samples.
`id`	A character string used to assign logging information on error. Used by `regional_epinow` to assign errors to regions. Alter the default to run with error catching.
`verbose`	Logical, should fitting progress be returned. Defaults to `FALSE`.

Value

A stanfit object

`lognorm_dist_def` *Generate a Log Normal Distribution Definition Based on Parameter Estimates*

Description

Soft-deprecated Generates a distribution definition when only parameter estimates are available for log normal distributed parameters. See `rlnorm` for distribution information.

Usage

```
lognorm_dist_def(mean, mean_sd, sd, sd_sd, max_value, samples, to_log = FALSE)
```

Arguments

`mean`	Numeric, log mean parameter of the gamma distribution.
`mean_sd`	Numeric, standard deviation of the log mean parameter.
`sd`	Numeric, log sd parameter of the gamma distribution.
`sd_sd`	Numeric, standard deviation of the log sd parameter.
`max_value`	Numeric, the maximum value to allow. Defaults to 120. Samples outside of this range are resampled.
`samples`	Numeric, number of sample distributions to generate.
`to_log`	Logical, should parameters be logged before use.

Value

A data.table defining the distribution as used by `dist_skel`

Examples

```
def <- lognorm_dist_def(mean = 1.621, mean_sd = 0.0640,
                        sd = 0.418, sd_sd = 0.0691,
                        max_value = 20, samples = 10)
print(def)
def$params[[1]]

def <- lognorm_dist_def(mean = 5, mean_sd = 1,
                        sd = 3, sd_sd = 1,
                        max_value = 20, samples = 10,
                        to_log = TRUE)
print(def)
def$params[[1]]
```

make_conf *Format Credible Intervals*

Description

Stable Combines a list of values into formatted credible intervals.

Usage

```
make_conf(value, CrI = 90, reverse = FALSE)
```

Arguments

value	List of value to map into a string. Requires, point, lower, and upper.
CrI	Numeric, credible interval to report. Defaults to 90
reverse	Logical, defaults to FALSE. Should the reported credible interval be switched.

Value

A character vector formatted for reporting

Examples

```
value <- list(median = 2, lower_90 = 1, upper_90 = 3)
make_conf(value)
```

map_prob_change *Categorise the Probability of Change for Rt*

Description

Stable Categorises a numeric variable into "Increasing" (< 0.05), "Likely increasing" (<0.2), "Unsure" (< 0.8), "Likely decreasing" (< 0.95), "Decreasing" (<= 1)

Usage

```
map_prob_change(var)
```

Arguments

var	Numeric variable to be categorised

Value

A character variable.

Examples

```
var <- seq(0.01, 1, 0.01)
var

map_prob_change(var)
```

`match_output_arguments` *Match User Supplied Arguments with Supported Options*

Description

Stable Match user supplied arguments with supported options and return a logical list for internal usage

Usage

```
match_output_arguments(
  input_args = c(),
  supported_args = c(),
  logger = NULL,
  level = "info"
)
```

Arguments

`input_args`	A character vector of input arguments (can be partial).
`supported_args`	A character vector of supported output arguments.
`logger`	A character vector indicating the logger to target messages at. Defaults to no logging.
`level`	Character string defaulting to "info". Logging level see documentation of futile.logger for details. Supported options are "info" and "debug"

Value

A logical vector of named output arguments

Examples

```
# select nothing
EpiNow2:::match_output_arguments(supported_args = c("fit", "plots", "samples"))

# select just plots
EpiNow2:::match_output_arguments("plots", supported_args = c("fit", "plots", "samples"))

# select plots and samples
EpiNow2:::match_output_arguments(c("plots", "samples"),
```

```
                        supported_args = c("fit", "plots", "samples"))

# lazily select arguments
EpiNow2:::match_output_arguments("p",
                        supported_args = c("fit", "plots", "samples"))
```

---

obs_opts *Observation Model Options*

Description

Stable Defines a list specifying the structure of the observation model. Custom settings can be supplied which override the defaults.

Usage

```
obs_opts(family = "negbin", weight = 1, week_effect = TRUE, scale = list())
```

Arguments

family	Character string defining the observation model. Options are Negative binomial ("negbin"), the default, and Poisson.
weight	Numeric, defaults to 1. Weight to give the observed data in the log density.
week_effect	Logical defaulting to TRUE. Should a day of the week effect be used in the observation model.
scale	List, defaulting to an empty list. Should an scaling factor be applied to map latent infections (convolved to date of report). If none empty a mean (mean) and standard deviation (sd) needs to be supplied defining the normally distributed scaling factor.

Value

A list of observation model settings.

Examples

```
# default settings
obs_opts()

# Turn off day of the week effect
obs_opts(week_effect = TRUE)

# Scale reported data
obs_opts(scale = list(mean = 0.2, sd = 0.02))
```

opts_list *Return an _opts List per Region*

Description

Maturing Define a list of _opts to pass to `regional_epinow` _opts accepting arguments. This is useful when different settings are needed between regions within a single `regional_epinow` call. Using `opts_list` the defaults can be applied to all regions present with an override passed to regions as necessary (either within `opts_list` or externally).

Usage

```
opts_list(opts, reported_cases, ...)
```

Arguments

`opts`	An _opts function call such as `rt_opts()`
`reported_cases`	A data frame containing a `region` variable indicating the target regions
`...`	Optional override for region defaults. See the examples for use case.

Value

A named list of options per region which can be passed to the _opt accepting arguments of `regional_epinow`

See Also

regional_epinow rt_opts

Examples

```
# uses example case vector
cases <- example_confirmed[1:40]
cases <- data.table::rbindlist(list(
  data.table::copy(cases)[, region := "testland"],
  cases[, region := "realland"]))

# default settings
opts_list(rt_opts(), cases)

# add a weekly random walk in realland
opts_list(rt_opts(), cases, realland = rt_opts(rw = 7))

# add a weekly random walk externally
rt <- opts_list(rt_opts(), cases)
rt$realland$rw <- 7
rt
```

plot.epinow *Plot method for epinow*

Description

Maturing `plot` method for class "epinow".

Usage

```
## S3 method for class 'epinow'
plot(x, type = "summary", ...)
```

Arguments

`x`	A list of output as produced by `epinow`
`type`	A character vector indicating the name of plots to return. Defaults to "summary" with supported options being "infections", "reports", "R", "growth_rate", "summary", "all".
`...`	Pass additional arguments to report_plots

Value

List of plots as produced by `report_plots`

See Also

plot plot.estimate_infections report_plots estimate_infections

plot.estimate_infections *Plot method for estimate_infections*

Description

Maturing `plot` method for class "estimate_infections".

Usage

```
## S3 method for class 'estimate_infections'
plot(x, type = "summary", ...)
```

Arguments

x	A list of output as produced by estimate_infections
type	A character vector indicating the name of plots to return. Defaults to "summary" with supported options being "infections", "reports", "R", "growth_rate", "summary", "all".
...	Pass additional arguments to report_plots

Value

List of plots as produced by report_plots

See Also

plot report_plots estimate_infections

---

plot.estimate_secondary

Plot method for estimate_secondary

Description

Experimental plot method for class "estimate_secondary".

Usage

```
## S3 method for class 'estimate_secondary'
plot(x, primary = FALSE, from = NULL, to = NULL, new_obs = NULL, ...)
```

Arguments

x	A list of output as produced by estimate_secondary
primary	Logical, defaults to FALSE. Should primary reports also be plot?
from	Date object indicating when to plot from.
to	Date object indicating when to plot up to.
new_obs	A data.frame containing the columns date and secondary which replace the secondary observations stored in the estimate_secondary output.
...	Pass additional arguments to plot function. Not currently in use.

Value

ggplot2 object

See Also

plot estimate_secondary

plot.estimate_truncation *Plot method for estimate_truncation*

Description

Experimental plot method for class "estimate_truncation". Returns a plot faceted over each dataset used in fitting with the latest observations as columns, the data observed at the time (and so truncated) as dots and the truncation adjusted estimates as a ribbon.

Usage

```
## S3 method for class 'estimate_truncation'
plot(x, ...)
```

Arguments

x	A list of output as produced by estimate_truncation
...	Pass additional arguments to plot function. Not currently in use.

Value

ggplot2 object

See Also

plot estimate_truncation

plot_CrIs *Plot EpiNow2 Credible Intervals*

Description

Stable Adds lineranges for user specified credible intervals

Usage

```
plot_CrIs(plot, CrIs, alpha, size)
```

Arguments

plot	A ggplot2 plot
CrIs	Numeric list of credible intervals present in the data. As produced by extract_CrIs
alpha	Numeric, overall alpha of the target line range
size	Numeric, size of the default line range.

Value

A ggplot2 plot.

`plot_estimates` *Plot Estimates*

Description

Questioning Allows users to plot the output from `estimate_infections` easily. In future releases it may be depreciated in favour of increasing the functionality of the S3 plot methods.

Usage

```
plot_estimates(
  estimate,
  reported,
  ylab = "Cases",
  hline,
  obs_as_col = TRUE,
  max_plot = 10
)
```

Arguments

`estimate`	A data.table of estimates containing the following variables: date, type (must contain "estimate", "estimate based on partial data" and optionally "forecast"),
`reported`	A data.table of reported cases with the following variables: date, confirm.
`ylab`	Character string, defaulting to "Cases". Title for the plot y axis.
`hline`	Numeric, if supplied gives the horizontal intercept for a indicator line.
`obs_as_col`	Logical, defaults to `TRUE`. Should observed data, if supplied, be plotted using columns or as points (linked using a line).
`max_plot`	Numeric, defaults to 10. A multiplicative upper bound on the number of cases shown on the plot. Based on the maximum number of reported cases.

Value

A ggplot2 object

Examples

```
# define example cases
cases <- example_confirmed[1:40]

# set up example delays
generation_time <- get_generation_time(disease = "SARS-CoV-2", source = "ganyani")
```

```
incubation_period <- get_incubation_period(disease = "SARS-CoV-2", source = "lauer")
reporting_delay <- estimate_delay(rlnorm(100, log(6), 1), max_value = 10)

# run model
out <- estimate_infections(cases, generation_time = generation_time,
                           delays = delay_opts(incubation_period, reporting_delay))
# plot infections
plot_estimates(
  estimate = out$summarised[variable == "infections"],
  reported = cases,
  ylab = "Cases", max_plot = 2) + ggplot2::facet_wrap(~type, scales = "free_y")

# plot reported cases estimated via Rt
plot_estimates(estimate = out$summarised[variable == "reported_cases"],
               reported = cases,
               ylab = "Cases")

# plot Rt estimates
plot_estimates(estimate = out$summarised[variable == "R"],
               ylab = "Effective Reproduction No.",
               hline = 1)
```

`plot_summary` *Plot a Summary of the Latest Results*

Description

Questioning Used to return a summary plot across regions (using results generated by `summarise_results`). May be depreciated in later releases in favour of enhanced S3 methods.

Usage

```
plot_summary(summary_results, x_lab = "Region", log_cases = FALSE, max_cases)
```

Arguments

`summary_results`	A data.table as returned by `summarise_results` (the `data` object).
`x_lab`	A character string giving the label for the x axis, defaults to region.
`log_cases`	Logical, should cases be shown on a logged scale. Defaults to `FALSE`
`max_cases`	Numeric, no default. The maximum number of cases to plot.

Value

A `ggplot2` object

`process_region` *Process regional estimate*

Description

Maturing Internal function that removes output that is not required, and returns logging information.

Usage

```
process_region(
  out,
  target_region,
  timing,
  return_output = TRUE,
  return_timing = TRUE,
  complete_logger = "EpiNow2.epinow"
)
```

Arguments

`out`	List of output returned by `epinow`
`target_region`	Character string indicating the region being evaluated
`timing`	Output from `Sys.time`
`return_output`	Logical, defaults to FALSE. Should output be returned, this automatically updates to TRUE if no directory for saving is specified.
`return_timing`	Logical, should runtime be returned
`complete_logger`	Character string indicating the logger to output the completion of estimation to.

Value

A list of processed output

See Also

regional_epinow

process_regions *Process all Region Estimates*

Description

Stable Internal function that processes the output from multiple epinow runs, adds summary logging information.

Usage

```
process_regions(regional_out, regions)
```

Arguments

regional_out	A list of output from multiple runs of regional_epinow
regions	A character vector identifying the regions that have been run

Value

A list of all regional estimates and successful regional estimates

See Also

regional_epinow epinow

regional_epinow *Real-time Rt Estimation, Forecasting and Reporting by Region*

Description

Maturing Efficiently runs epinow() across multiple regions in an efficient manner and conducts basic data checks and cleaning such as removing regions with fewer than non_zero_points as these are unlikely to produce reasonable results whilst consuming significant resources. See the documentation for epinow for further information.

By default all arguments supporting input from _opts functions are shared across regions (including delays, truncation, Rt settings, stan settings, and gaussian process settings). Region specific settings are supported by passing a named list of _opts calls (with an entry per region) to the relevant argument. A helper function (opts_list) is available to facilitate building this list.

Regions can be estimated in parallel using the {future} package (see setup_future). The progress of producing estimates across multiple regions is tracked using the progressr package. Modify this behaviour using progressr::handlers and enable it in batch by setting R_PROGRESSR_ENABLE=TRUE as an environment variable.

Usage

```
regional_epinow(
  reported_cases,
  generation_time,
  delays = delay_opts(),
  truncation = trunc_opts(),
  rt = rt_opts(),
  backcalc = backcalc_opts(),
  gp = gp_opts(),
  obs = obs_opts(),
  stan = stan_opts(),
  horizon = 7,
  CrIs = c(0.2, 0.5, 0.9),
  target_folder = NULL,
  target_date,
  non_zero_points = 2,
  output = c("regions", "summary", "samples", "plots", "latest"),
  return_output = FALSE,
  summary_args = list(),
  verbose = FALSE,
  logs = tempdir(check = TRUE),
  ...
)
```

Arguments

reported_cases	A data frame of confirmed cases (confirm) by date (date), and region (region).
generation_time	A list containing the mean, standard deviation of the mean (mean_sd), standard deviation (sd), standard deviation of the standard deviation and the maximum allowed value for the generation time (assuming a gamma distribution).
delays	A call to delay_opts() defining delay distributions and options. See the documentation of delay_opts() and the examples below for details.
truncation	**Experimental** A list of options as generated by trunc_opts() defining the truncation of observed data. Defaults to trunc_opts(). See estimate_truncation() for an approach to estimating truncation from data.
rt	A list of options as generated by rt_opts() defining Rt estimation. Defaults to rt_opts(). Set to NULL to switch to using back calculation rather than generating infections using Rt.
backcalc	A list of options as generated by backcalc_opts() to define the back calculation. Defaults to backcalc_opts().
gp	A list of options as generated by gp_opts() to define the Gaussian process. Defaults to gp_opts().Set to NULL to disable the Gaussian process.
obs	A list of options as generated by obs_opts() defining the observation model. Defaults to obs_opts().

stan	A list of stan options as generated by stan_opts(). Defaults to stan_opts(). Can be used to override data, init, and verbose settings if desired.
horizon	Numeric, defaults to 7. Number of days into the future to forecast.
CrIs	Numeric vector of credible intervals to calculate.
target_folder	Character string specifying where to save results (will create if not present).
target_date	Date, defaults to maximum found in the data if not specified.
non_zero_points	Numeric, the minimum number of time points with non-zero cases in a region required for that region to be evaluated. Defaults to 7.
output	A character vector of optional output to return. Supported options are the individual regional estimates ("regions"), samples ("samples"), plots ("plots"), copying the individual region dated folder into a latest folder (if target_folder is not null, set using "latest"), the stan fit of the underlying model ("fit"), and an overall summary across regions ("summary"). The default is to return samples and plots alongside summarised estimates and summary statistics. If target_folder is not NULL then the default is also to copy all results into a latest folder.
return_output	Logical, defaults to FALSE. Should output be returned, this automatically updates to TRUE if no directory for saving is specified.
summary_args	A list of arguments passed to regional_summary. See the regional_summary documentation for details.
verbose	Logical defaults to FALSE. Outputs verbose progress messages to the console from epinow.
logs	Character path indicating the target folder in which to store log information. Defaults to the temporary directory if not specified. Default logging can be disabled if logs is set to NULL. If specifying a custom logging setup then the code for setup_default_logging and the setup_logging function are a sensible place to start.
...	Pass additional arguments to epinow. See the documentation for epinow for details.

Value

A list of output stratified at the top level into regional output and across region output summary output

See Also

epinow estimate_infections forecast_infections setup_future regional_summary

Examples

```
#set number of cores to use
options(mc.cores = ifelse(interactive(), 4, 1))

# construct example distributions
```

```
generation_time <- get_generation_time(disease = "SARS-CoV-2", source = "ganyani")
incubation_period <- get_incubation_period(disease = "SARS-CoV-2", source = "lauer")
reporting_delay <- list(mean = convert_to_logmean(3,1),
                        mean_sd = 0.1,
                        sd = convert_to_logsd(3,1),
                        sd_sd = 0.1, max = 15)

# uses example case vector
cases <- example_confirmed[1:60]
cases <- data.table::rbindlist(list(
  data.table::copy(cases)[, region := "testland"],
  cases[, region := "realland"]))

# run epinow across multiple regions and generate summaries
# samples and warmup have been reduced for this example
def <- regional_epinow(reported_cases = cases,
                       generation_time = generation_time,
                       delays = delay_opts(incubation_period, reporting_delay),
                       rt = rt_opts(prior = list(mean = 2, sd = 0.2)),
                       stan = stan_opts(samples = 100, warmup = 200,
                                        control = list(adapt_delta = 0.95)),
                       verbose = interactive())

# apply a different rt method per region
# (here a gaussian process and a weekly random walk)
gp <- opts_list(gp_opts(), cases)
gp <- update_list(gp, list(realland = NULL))
rt <- opts_list(rt_opts(), cases, realland = rt_opts(rw = 7))
region_rt <- regional_epinow(reported_cases = cases,
                             generation_time = generation_time,
                             delays = delay_opts(incubation_period, reporting_delay),
                             rt = rt, gp = gp,
                             stan = stan_opts(samples = 100, warmup = 200,
                                              control = list(adapt_delta = 0.95)),
                             verbose = interactive())
```

`regional_runtimes` *Summarise Regional Runtimes*

Description

Maturing Used internally by `regional_epinow` to summarise region run times.

Usage

```
regional_runtimes(
  regional_output = NULL,
  target_folder = NULL,
  target_date = NULL,
```

```
  return_output = FALSE
)
```

Arguments

regional_output	A list of output as produced by regional_epinow and stored in the regional list.
target_folder	Character string specifying where to save results (will create if not present).
target_date	A character string giving the target date for which to extract results (in the format "yyyy-mm-dd"). Defaults to latest available estimates.
return_output	Logical, defaults to FALSE. Should output be returned, this automatically updates to TRUE if no directory for saving is specified.

Value

A data.table of region run times

See Also

regional_summary regional_epinow

Examples

```
# example delays
generation_time <- get_generation_time(disease = "SARS-CoV-2", source = "ganyani")
incubation_period <- get_incubation_period(disease = "SARS-CoV-2", source = "lauer")
reporting_delay <- example_delay(rlnorm(100, log(6), 1), max_value = 15)

# example case vector from EpiSoon
cases <- example_confirmed[1:30]
cases <- data.table::rbindlist(list(
  data.table::copy(cases)[, region := "testland"],
  cases[, region := "realland"]))

# run basic nowcasting pipeline
regional_out <- regional_epinow(reported_cases = cases,
                                generation_time = generation_time,
                                delays = delay_opts(incubation_period, reporting_delay),
                                samples = 100, stan_args = list(warmup = 100),
                                output = c("region", "timing"))

regional_runtimes(regional_output = regional_out$regional)
```

`regional_summary` *Regional Summary Output*

Description

Maturing Used to produce summary output either internally in `regional_epinow` or externally.

Usage

```
regional_summary(
  regional_output = NULL,
  reported_cases,
  results_dir = NULL,
  summary_dir = NULL,
  target_date = NULL,
  region_scale = "Region",
  all_regions = TRUE,
  return_output = FALSE,
  max_plot = 10
)
```

Arguments

`regional_output`	A list of output as produced by `regional_epinow` and stored in the `regional` list.
`reported_cases`	A data frame of confirmed cases (confirm) by date (date), and region (`region`).
`results_dir`	An optional character string indicating the location of the results directory to extract results from.
`summary_dir`	A character string giving the directory in which to store summary of results.
`target_date`	A character string giving the target date for which to extract results (in the format "yyyy-mm-dd"). Defaults to latest available estimates.
`region_scale`	A character string indicating the name to give the regions being summarised.
`all_regions`	Logical, defaults to `TRUE`. Should summary plots for all regions be returned rather than just regions of interest.
`return_output`	Logical, defaults to FALSE. Should output be returned, this automatically updates to TRUE if no directory for saving is specified.
`max_plot`	Numeric, defaults to 10. A multiplicative upper bound on the number of cases shown on the plot. Based on the maximum number of reported cases.

Value

A list of summary measures and plots

See Also

regional_epinow

Examples

```
# example delays
generation_time <- get_generation_time(disease = "SARS-CoV-2", source = "ganyani")
incubation_period <- get_incubation_period(disease = "SARS-CoV-2", source = "lauer")
reporting_delay <- estimate_delay(rlnorm(100, log(6), 1), max_value = 30)

# example case vector from EpiSoon
cases <- example_confirmed[1:30]
cases <- data.table::rbindlist(list(
  data.table::copy(cases)[, region := "testland"],
  cases[, region := "realland"]))

# run basic nowcasting pipeline
out <- regional_epinow(reported_cases = cases,
                       generation_time = generation_time,
                       delays = delay_opts(incubation_period, reporting_delay),
                       output = "region",
                       rt = NULL)

regional_summary(regional_output = out$regional,
                 reported_cases = cases)
```

---

report_cases *Report case counts by date of report*

Description

Soft-deprecated Convolves latent infections to reported cases via an observation model. Likely to be removed/replaced in later releases by functionality drawing on the `stan` implementation.

Usage

```
report_cases(
  case_estimates,
  case_forecast = NULL,
  delays,
  type = "sample",
  reporting_effect,
  CrIs = c(0.2, 0.5, 0.9)
)
```

Arguments

case_estimates	A data.table of case estimates with the following variables: date, sample, cases
case_forecast	A data.table of case forecasts with the following variables: date, sample, cases. If not supplied the default is not to incorporate forecasts.
delays	A call to delay_opts() defining delay distributions and options. See the documentation of delay_opts() and the examples below for details.
type	Character string indicating the method to use to transform counts. Supports either "sample" which approximates sampling or "median" would shift by the median of the distribution.
reporting_effect	A data.table giving the weekly reporting effect with the following variables: sample (must be the same as in nowcast), effect (numeric scaling factor for each weekday), day (numeric 1 - 7 (1 = Monday and 7 = Sunday)). If not supplied then no weekly reporting effect is assumed.
CrIs	Numeric vector of credible intervals to calculate.

Value

A list of data.tables. The first entry contains the following variables sample, date and cases with the second being summarised across samples.

Examples

```
# define example cases
cases <- example_confirmed[1:40]

# set up example delays
generation_time <- get_generation_time(disease = "SARS-CoV-2", source = "ganyani")
incubation_period <- get_incubation_period(disease = "SARS-CoV-2", source = "lauer")
reporting_delay <- bootstrapped_dist_fit(rlnorm(100, log(6), 1), max_value = 30)

# run model
out <- estimate_infections(cases, samples = 100,
                           generation_time = generation_time,
                           delays = delay_opts(incubation_period, reporting_delay),
                           rt = NULL)

reported_cases <- report_cases(case_estimates =
                               out$samples[variable == "infections"][,
                               cases := as.integer(value)][, value := NULL],
                               delays = delay_opts(incubation_period, reporting_delay),
                               type = "sample")
print(reported_cases)
```

`report_plots` *Report plots*

Description

Questioning Returns key summary plots for estimates. May be depreciated in later releases as current S3 methods are enhanced.

Usage

```
report_plots(
  summarised_estimates,
  reported,
  target_folder = NULL,
  max_plot = 10
)
```

Arguments

`summarised_estimates`	A data.table of summarised estimates containing the following variables: variable, median, bottom, and top. It should contain the following estimates: R, infections, reported_cases_rt, and r (rate of growth).
`reported`	A data.table of reported cases with the following variables: date, confirm.
`target_folder`	Character string specifying where to save results (will create if not present).
`max_plot`	Numeric, defaults to 10. A multiplicative upper bound on the number of cases shown on the plot. Based on the maximum number of reported cases.

Value

A named list of ggplot2 objects, `list(infections,reports,R,growth_rate,summary)`, which correspond to a summary combination (last item) and for the leading items @seealso `plot_estimates()` of `summarised_estimates[variable == "infections"]`, `summarised_estimates[variable == "reported_cases"]`, `summarised_estimates[variable == "R"]`, and `summarised_estimates[variable == "growth_rate"]`, respectively.

Examples

```
# define example cases
cases <- example_confirmed[1:40]

# set up example delays
generation_time <- get_generation_time(disease = "SARS-CoV-2", source = "ganyani")
incubation_period <- get_incubation_period(disease = "SARS-CoV-2", source = "lauer")
reporting_delay <- bootstrapped_dist_fit(rlnorm(100, log(6), 1), max_value = 30)
```

```
# run model
out <- estimate_infections(cases, samples = 100,
                           generation_time = generation_time,
                           delays = delay_opts(incubation_period, reporting_delay),
                           rt = NULL)

# plot infections
plots <- report_plots(summarised_estimates = out$summarised,
                      reported = cases)
plots
```

`report_summary` *Provide Summary Statistics for Estimated Infections and Rt*

Description

Questioning Creates a snapshot summary of estimates. May be removed in later releases as S3 methods are enhanced.

Usage

```
report_summary(
  summarised_estimates,
  rt_samples,
  target_folder = NULL,
  return_numeric = FALSE
)
```

Arguments

`summarised_estimates`	A data.table of summarised estimates containing the following variables: variable, median, bottom, and top. It should contain the following estimates: R, infections, and r (rate of growth).
`rt_samples`	A data.table containing Rt samples with the following variables: sample and value.
`target_folder`	Character string specifying where to save results (will create if not present).
`return_numeric`	Should numeric summary information be returned.

Value

A data.table containing formatted and numeric summary measures

rstan_opts *Rstan Options*

Description

Stable Defines a list specifying the arguments passed to underlying rstan functions via rstan_sampling_opts and rstan_vb_opts.Custom settings can be supplied which override the defaults.

Usage

```
rstan_opts(object = NULL, samples = 2000, method = "sampling", ...)
```

Arguments

object	Stan model object. By default uses the compiled package default.
samples	Numeric, default 2000. Overall number of posterior samples. When using multiple chains iterations per chain is samples / chains.
method	A character string, defaulting to sampling. Currently supports rstan::sampling ("sampling") or rstan:vb ("vb").
...	Additional parameters to pass underlying option functions.

Value

A list of arguments to pass to the appropriate rstan functions.

See Also

rstan_sampling_opts rstan_vb_opts

Examples

```
rstan_opts(samples = 1000)

# using vb
rstan_opts(method = "vb")
```

`rstan_sampling_opts` *Rstan Sampling Options*

Description

Stable Defines a list specifying the arguments passed to `rstan::sampling`. Custom settings can be supplied which override the defaults.

Usage

```
rstan_sampling_opts(
  cores = getOption("mc.cores", 1L),
  warmup = 250,
  samples = 2000,
  chains = 4,
  control = list(),
  save_warmup = FALSE,
  seed = as.integer(runif(1, 1, 1e+08)),
  future = FALSE,
  max_execution_time = Inf,
  ...
)
```

Arguments

`cores`	Number of cores to use when executing the chains in parallel, which defaults to 1 but it is recommended to set the mc.cores option to be as many processors as the hardware and RAM allow (up to the number of chains).
`warmup`	Numeric, defaults to 250. Number of warmup samples per chain.
`samples`	Numeric, default 2000. Overall number of posterior samples. When using multiple chains iterations per chain is samples / chains.
`chains`	Numeric, defaults to 4. Number of MCMC chains to use.
`control`	List, defaults to empty. control parameters to pass to underlying `rstan` function. By default `adapt_delta` = `0.98` and `max_treedepth` = `15` though these settings can be overwritten.
`save_warmup`	Logical, defaults to FALSE. Should warmup progress be saved.
`seed`	Numeric, defaults uniform random number between 1 and 1e8. Seed of sampling process.
`future`	Logical, defaults to `FALSE`. Should stan chains be run in parallel using `future`. This allows users to have chains fail gracefully (i.e when combined with `max_execution_time`). Should be combined with a call to `future::plan`
`max_execution_time`	Numeric, defaults to Inf (seconds). If set will kill off processing of each chain if not finished within the specified timeout. When more than 2 chains finish successfully estimates will still be returned. If less than 2 chains return within the allowed time then estimation will fail with an informative error.

...	Additional parameters to pass to `rstan::sampling`.

Value

A list of arguments to pass to `rstan::sampling`

Examples

```
rstan_sampling_opts(samples = 2000)
```

`rstan_vb_opts` *Rstan Variational Bayes Options*

Description

Stable Defines a list specifying the arguments passed to `rstan::vb`. Custom settings can be supplied which override the defaults.

Usage

```
rstan_vb_opts(samples = 2000, trials = 10, iter = 10000, ...)
```

Arguments

`samples`	Numeric, default 2000. Overall number of approximate posterior samples.
`trials`	Numeric, defaults to 10. Number of attempts to use `rstan::vb` before failing.
`iter`	Numeric, defaulting to 10000. Number of iterations to use in `rtan::vb`.
...	Additional parameters to pass to `rstan::vb`.

Value

A list of arguments to pass to `rstan::vb`

Examples

```
rstan_vb_opts(samples = 1000)
```

`rt_opts` *Time-Varying Reproduction Number Options*

Description

Stable Defines a list specifying the optional arguments for the time-varying reproduction number. Custom settings can be supplied which override the defaults.

Usage

```
rt_opts(
  prior = list(mean = 1, sd = 1),
  use_rt = TRUE,
  rw = 0,
  use_breakpoints = TRUE,
  future = "latest",
  gp_on = "R_t-1",
  pop = 0
)
```

Arguments

`prior`	List containing named numeric elements "mean" and "sd". The mean and standard deviation of the log normal Rt prior. Defaults to mean of 1 and standard deviation of 1.
`use_rt`	Logical, defaults to `TRUE`. Should Rt be used to generate infections and hence reported cases.
`rw`	Numeric step size of the random walk, defaults to 0. To specify a weekly random walk set `rw = 7`. For more custom break point settings consider passing in a `breakpoints` variable as outlined in the next section.
`use_breakpoints`	Logical, defaults to `TRUE`. Should break points be used if present as a `breakpoint` variable in the input data. Break points should be defined as 1 if present and otherwise 0. By default breakpoints are fit jointly with a global non-parametric effect and so represent a conservative estimate of break point changes (alter this by setting `gp = NULL`).
`future`	A character string or integer. This argument indicates how to set future Rt values. Supported options are to project using the Rt model ("project"), to use the latest estimate based on partial data ("latest"), to use the latest estimate based on data that is over 50% complete ("estimate"). If an integer is supplied then the Rt estimate from this many days into the future (or past if negative) past will be used forwards in time.
`gp_on`	Character string, defaulting to "R_t-1". Indicates how the Gaussian process, if in use, should be applied to Rt. Currently supported options are applying the Gaussian process to the last estimated Rt (i.e Rt = Rt-1 * GP), and applying the Gaussian process to a global mean (i.e Rt = R0 * GP). Both should produced

comparable results when data is not sparse but the method relying on a global mean will revert to this for real time estimates, which may not be desirable.

pop — Integer, defaults to 0. Susceptible population initially present. Used to adjust Rt estimates when otherwise fixed based on the proportion of the population that is susceptible. When set to 0 no population adjustment is done.

Value

A list of settings defining the time-varying reproduction number

Examples

```
# default settings
rt_opts()

# add a custom length scale
rt_opts(prior = list(mean = 2, sd = 1))

# add a weekly random walk
rt_opts(rw = 7)
```

---

run_region *Run epinow with Regional Processing Code*

Description

Maturing Internal function that handles calling `epinow`. Future work will extend this function to better handle `stan` logs and allow the user to modify settings between regions.

Usage

```
run_region(
  target_region,
  generation_time,
  delays,
  truncation,
  rt,
  backcalc,
  gp,
  obs,
  stan,
  horizon,
  CrIs,
  reported_cases,
  target_folder,
  target_date,
  return_output,
  output,
```

```
  complete_logger,
  verbose,
  progress_fn,
  ...
)
```

Arguments

target_region	Character string indicating the region being evaluated
generation_time	A list containing the mean, standard deviation of the mean (mean_sd), standard deviation (sd), standard deviation of the standard deviation and the maximum allowed value for the generation time (assuming a gamma distribution).
delays	A call to delay_opts() defining delay distributions and options. See the documentation of delay_opts() and the examples below for details.
truncation	**Experimental** A list of options as generated by trunc_opts() defining the truncation of observed data. Defaults to trunc_opts(). See estimate_truncation() for an approach to estimating truncation from data.
rt	A list of options as generated by rt_opts() defining Rt estimation. Defaults to rt_opts(). Set to NULL to switch to using back calculation rather than generating infections using Rt.
backcalc	A list of options as generated by backcalc_opts() to define the back calculation. Defaults to backcalc_opts().
gp	A list of options as generated by gp_opts() to define the Gaussian process. Defaults to gp_opts().Set to NULL to disable the Gaussian process.
obs	A list of options as generated by obs_opts() defining the observation model. Defaults to obs_opts().
stan	A list of stan options as generated by stan_opts(). Defaults to stan_opts(). Can be used to override data, init, and verbose settings if desired.
horizon	Numeric, defaults to 7. Number of days into the future to forecast.
CrIs	Numeric vector of credible intervals to calculate.
reported_cases	A data frame of confirmed cases (confirm) by date (date), and region (region).
target_folder	Character string specifying where to save results (will create if not present).
target_date	Date, defaults to maximum found in the data if not specified.
return_output	Logical, defaults to FALSE. Should output be returned, this automatically updates to TRUE if no directory for saving is specified.
output	A character vector of optional output to return. Supported options are the individual regional estimates ("regions"), samples ("samples"), plots ("plots"), copying the individual region dated folder into a latest folder (if target_folder is not null, set using "latest"), the stan fit of the underlying model ("fit"), and an overall summary across regions ("summary"). The default is to return samples and plots alongside summarised estimates and summary statistics. If target_folder is not NULL then the default is also to copy all results into a latest folder.

complete_logger	Character string indicating the logger to output the completion of estimation to.
verbose	Logical defaults to FALSE. Outputs verbose progress messages to the console from epinow.
progress_fn	Function as returned by progressr::progressor. Allows the use of a progress bar.
...	Pass additional arguments to epinow. See the documentation for epinow for details.

Value

A list of processed output as produced by process_region

See Also

regional_epinow

R_to_growth *Convert Reproduction Numbers to Growth Rates*

Description

Questioning See here for justification. Now handled internally by stan so may be removed in future updates if no user demand.

Usage

```
R_to_growth(R, gamma_mean, gamma_sd)
```

Arguments

R	Numeric, Reproduction number estimates
gamma_mean	Numeric, mean of the gamma distribution
gamma_sd	Numeric, standard deviation of the gamma distribution

Value

Numeric vector of reproduction number estimates

Examples

```
R_to_growth(2.18, 4, 1)
```

`sample_approx_dist` *Approximate Sampling a Distribution using Counts*

Description

Soft-deprecated Convolves cases by a PMF function. This function will soon be removed or replaced with a more robust `stan` implementation.

Usage

```
sample_approx_dist(
  cases = NULL,
  dist_fn = NULL,
  max_value = 120,
  earliest_allowed_mapped = NULL,
  direction = "backwards",
  type = "sample",
  truncate_future = TRUE
)
```

Arguments

`cases`	A dataframe of cases (in date order) with the following variables: `date` and `cases`.
`dist_fn`	Function that takes two arguments with the first being numeric and the second being logical (and defined as `dist`). Should return the probability density or a sample from the defined distribution. See the examples for more.
`max_value`	Numeric, maximum value to allow. Defaults to 120 days
`earliest_allowed_mapped`	A character string representing a date ("2020-01-01"). Indicates the earliest allowed mapped value.
`direction`	Character string, defato "backwards". Direction in which to map cases. Supports either "backwards" or "forwards".
`type`	Character string indicating the method to use to transform counts. Supports either "sample" which approximates sampling or "median" would shift by the median of the distribution.
`truncate_future`	Logical, should cases be truncated if they occur after the first date reported in the data. Defaults to `TRUE`.

Value

A `data.table` of cases by date of onset

Examples

```
cases <- example_confirmed
cases <- cases[, cases := as.integer(confirm)]
print(cases)

# total cases
sum(cases$cases)

delay_fn <- function(n, dist, cum) {
              if(dist) {
                pgamma(n + 0.9999, 2, 1) - pgamma(n - 1e-5, 2, 1)
               }else{
                as.integer(rgamma(n, 2, 1))
               }
             }

onsets <- sample_approx_dist(cases = cases,
                             dist_fn = delay_fn)

# estimated onset distribution
print(onsets)

# check that sum is equal to reported cases
total_onsets <- median(
   purrr::map_dbl(1:100,
                  ~ sum(sample_approx_dist(cases = cases,
                  dist_fn = delay_fn)$cases)))
total_onsets


# map from onset cases to reported
reports <- sample_approx_dist(cases = cases,
                              dist_fn = delay_fn,
                              direction = "forwards")


# map from onset cases to reported using a mean shift
reports <- sample_approx_dist(cases = cases,
                              dist_fn = delay_fn,
                              direction = "forwards",
                              type = "median")
```

`save_estimate_infections`
Save Estimated Infections

Description

Stable Saves output from `estimate_infections` to a target directory.

Usage

```
save_estimate_infections(
  estimates,
  target_folder = NULL,
  samples = TRUE,
  return_fit = TRUE
)
```

Arguments

estimates	List of data frames as output by estimate_infections
target_folder	Character string specifying where to save results (will create if not present).
samples	Logical, defaults to TRUE. Should samples be saved
return_fit	Logical, defaults to TRUE. Should the fit stan object be returned.

See Also

estimate_infections

---

save_forecast_infections

Save Forecast Infections

Description

Experimental Saves the output from forecast_infections to a target directory.

Usage

```
save_forecast_infections(forecast, target_folder = NULL, samples = TRUE)
```

Arguments

forecast	A list of data frames as output by forecast_infections
target_folder	Character string specifying where to save results (will create if not present).
samples	Logical, defaults to TRUE. Should samples be saved

See Also

forecast_infections

save_input *Save Observed Data*

Description

Stable Saves observed data to a target location if given.

Usage

```
save_input(reported_cases, target_folder)
```

Arguments

reported_cases	A data frame of confirmed cases (confirm) by date (date). confirm must be integer and date must be in date format.
target_folder	Character string specifying where to save results (will create if not present).

secondary_opts *Secondary Reports Options*

Description

Experimental Returns a list of options defining the secondary model used in `estimate_secondary()`. This model is a combination of a convolution of previously observed primary reports combined with current primary reports (either additive or subtractive). This model can optionally be cumulative. See the documentation of `type` for sensible options to cover most use cases and the returned values of `secondary_opts()` for all currently supported options.

Usage

```
secondary_opts(type = "incidence", ...)
```

Arguments

type	A character string indicating the type of observation the secondary reports are. Options include: • "incidence": Assumes that secondary reports equal a convolution of previously observed primary reported cases. An example application is deaths from an infectious disease predicted by reported cases of that disease (or estimated infections). • "prevalence": Assumes that secondary reports are cumulative and are defined by currently observed primary reports minus a convolution of secondary reports. An example application is hospital bed usage predicted by hospital admissions.
...	Overwrite options defined by type. See the returned values for all options that can be passed.

Value

A list of binary options summarising secondary model used in `estimate_secondary()`. Options returned are `cumulative` (should the secondary report be cumulative), `historic` (should a convolution of primary reported cases be used to predict secondary reported cases), `primary_hist_additive` (should the historic convolution of primary reported cases be additive or subtractive), `current` (should currently observed primary reported cases contribute to current secondary reported cases), `primary_current_additive` (should current primary reported cases be additive or subtractive).

See Also

estimate_secondary

Examples

```
# incidence model
secondary_opts("incidence")

# prevalence model
secondary_opts("prevalence")
```

`setup_default_logging` *Setup Default Logging*

Description

Questioning Sets up default logging. Usage of logging is currently being explored as the current setup cannot log stan errors or progress.

Usage

```
setup_default_logging(
  logs = tempdir(check = TRUE),
  mirror_epinow = FALSE,
  target_date = NULL
)
```

Arguments

`logs`	Character path indicating the target folder in which to store log information. Defaults to the temporary directory if not specified. Default logging can be disabled if `logs` is set to NULL. If specifying a custom logging setup then the code for `setup_default_logging` and the `setup_logging` function are a sensible place to start.
`mirror_epinow`	Logical, defaults to FALSE. Should internal logging be returned from `epinow` to the console.
`target_date`	Date, defaults to maximum found in the data if not specified.

Examples

```
setup_default_logging()
```

`setup_dt` *Convert to Data Table*

Description

Stable Convenience function that sets the number of `data.table` cores to 1 and maps input to be a `data.table`

Usage

```
setup_dt(reported_cases)
```

Arguments

`reported_cases`	A data frame of confirmed cases (confirm) by date (date). confirm must be integer and date must be in date format.

Value

A data table

`setup_future` *Set up Future Backend*

Description

Stable A utility function that aims to streamline the set up of the required future backend with sensible defaults for most users of `regional_epinow`. More advanced users are recommended to setup their own `future` backend based on their available resources.

Usage

```
setup_future(
  reported_cases,
  strategies = c("multiprocess", "multiprocess"),
  min_cores_per_worker = 4
)
```

Arguments

reported_cases	A data frame of confirmed cases (confirm) by date (date), and region (region).
strategies	A vector length 1 to 2 of strategies to pass to future::plan. Nesting of parallisation is from the top level down. The default is to set up nesting parallisation with both using future::multiprocess. For single level parallisation use a single strategy or future::plan directly. See ?future::plan for options.
min_cores_per_worker	Numeric, the minimum number of cores per worker. Defaults to 4 which assumes 4 MCMC chains are in use per region.

Value

Numeric number of cores to use per worker. If greater than 1 pass to stan_args = list(cores = "output from setup future") or use future = TRUE. If only a single strategy is used then nothing is returned.

---

setup_logging *Setup Logging*

Description

Questioning Sets up futile.logger logging, which is integrated into EpiNow2. See the documentation for futile.logger for full details. By default EpiNow2 prints all logs at the "INFO" level and returns them to the console. Usage of logging is currently being explored as the current setup cannot log stan errors or progress.

Usage

```
setup_logging(
  threshold = "INFO",
  file = NULL,
  mirror_to_console = FALSE,
  name = "EpiNow2"
)
```

Arguments

threshold	Character string indicating the logging level see (?futile.logger for details of the available options). Defaults to "INFO".
file	Character string indicating the path to save logs to. By default logs will be written to the console.
mirror_to_console	Logical, defaults to FALSE. If saving logs to a file should they also be duplicated in the console.

name	Character string defaulting to EpiNow2. This indicates the name of the logger to setup. The default logger for EpiNow2 is called EpiNow2. Nested options include: Epinow2.epinow which controls all logging for epinow and nested functions, EpiNow2.epinow.estimate_infections (logging in estimate_infections), and EpiNow2.epinow.estimate_infections.fit (logging in fitting functions).

Value

Nothing

---

setup_target_folder *Setup Target Folder for Saving*

Description

Stable Sets up a folders for saving results

Usage

```
setup_target_folder(target_folder = NULL, target_date)
```

Arguments

target_folder	Character string specifying where to save results (will create if not present).
target_date	Date, defaults to maximum found in the data if not specified.

Value

A list containing the path to the dated folder and the latest folder

---

simulate_cases *Simulate Cases by Date of Infection, Onset and Report*

Description

\@description **Questioning** Simulate cases from a single Rt trace, an initial number of cases, and a reporting model This functionality has largely been superseded by simulate_infections and will likely to replaced or updated to depend on stan code.

Usage

```
simulate_cases(
  rts,
  initial_cases,
  initial_date,
  generation_interval,
  rdist = rpois,
  delay_defs,
  reporting_effect,
  reporting_model,
  truncate_future = TRUE,
  type = "sample"
)
```

Arguments

rts	A dataframe of containing two variables rt and date with rt being numeric and date being a date.
initial_cases	Integer, initial number of cases.
initial_date	Date, (i.e as.Date("2020-02-01")). Starting date of the simulation.
generation_interval	Numeric vector describing the generation interval probability density
rdist	A function to be used to sample the number of cases. Must take two arguments with the first specifying the number of samples and the second the mean. Defaults to rpois if not supplied
delay_defs	A list of single row data.tables that each defines a delay distribution (model, parameters and maximum delay for each model). See lognorm_dist_def for an example of the structure.
reporting_effect	A numeric vector of length 7 that allows the scaling of reported cases by the day on which they report (1 = Monday, 7 = Sunday). By default no scaling occurs.
reporting_model	A function that takes a single numeric vector as an argument and returns a single numeric vector. Can be used to apply stochastic reporting effects. See the examples for details.
truncate_future	Logical, should cases be truncated if they occur after the first date reported in the data. Defaults to TRUE.
type	Character string indicating the method to use to transform counts. Supports either "sample" which approximates sampling or "median" would shift by the median of the distribution.

Value

A dataframe containing three variables: date, cases and reference.

See Also

simulate_infections

`simulate_infections` *Simulate infections using a given trajectory of the time-varying reproduction number*

Description

Stable This function simulates infections using an existing fit to observed cases but with a modified time-varying reproduction number. This can be used to explore forecast models or past counterfactuals. Simulations can be run in parallel using `future::plan`.

Usage

```
simulate_infections(
  estimates,
  R = NULL,
  model = NULL,
  samples = NULL,
  batch_size = 10,
  verbose = interactive()
)
```

Arguments

`estimates`	The `estimates` element of an `epinow` run that has been done with output = "fit", or the result of `estimate_infections` with `return_fit` set to TRUE.
`R`	A numeric vector of reproduction numbers; these will overwrite the reproduction numbers contained in `estimates`, except elements set to NA. If it is longer than the time series of reproduction numbers contained in `estimates`, the values going beyond the length of estimated reproduction numbers are taken as forecast.
`model`	A compiled stan model as returned by `rstan::stan_model`.
`samples`	Numeric, number of posterior samples to simulate from. The default is to use all samples in the `estimates` input.
`batch_size`	Numeric, defaults to 100. Size of batches in which to simulate. May decrease run times due to reduced IO costs but this is still being evaluated. If set to NULL then all simulations are done at once.
`verbose`	Logical defaults to `interactive()`. Should a progress bar (from `progressr`) be shown.

Examples

```
#set number of cores to use
options(mc.cores = ifelse(interactive(), 4, 1))
# get example case counts
reported_cases <- example_confirmed[1:50]

# set up example generation time
generation_time <- get_generation_time(disease = "SARS-CoV-2", source = "ganyani")
# set delays between infection and case report
incubation_period <- get_incubation_period(disease = "SARS-CoV-2", source = "lauer")
reporting_delay <- list(mean = convert_to_logmean(3, 1), mean_sd = 0.1,
                        sd = convert_to_logsd(3, 1), sd_sd = 0.1, max = 15)

# fit model to data to recover Rt estimates
est <- estimate_infections(reported_cases, generation_time = generation_time,
                           delays = delay_opts(incubation_period, reporting_delay),
                           rt = rt_opts(prior = list(mean = 2, sd = 0.1)),
                           gp = gp_opts(ls_min = 10, boundary_scale = 1.5,,
                                        basis_prop = 0.1),
                           obs = obs_opts(scale = list(mean = 0.1, sd = 0.01)))

# update Rt trajectory and simulate new infections using it
R <- c(rep(NA_real_, 40), rep(0.5, 10), rep(0.8, 7))
sims <- simulate_infections(est, R)
plot(sims)
```

---

stan_opts *Stan Options*

Description

Stable Defines a list specifying the arguments passed to underlying stan backend functions via `rstan_sampling_opts` and `rstan_vb_opts`. Custom settings can be supplied which override the defaults.

Usage

```
stan_opts(
  samples = 2000,
  backend = "rstan",
  init_fit = NULL,
  return_fit = TRUE,
  ...
)
```

Arguments

samples	Numeric, default 2000. Overall number of posterior samples. When using multiple chains iterations per chain is samples / chains.
backend	Character string indicating the backend to use for fitting stan models. Currently only "rstan" is supported.
init_fit	**Experimental** Character string or stanfit object, defaults to NULL. Should an initial fit be used to initialise the full fit. An example scenario would be using a national level fit to parametrise regional level fits. Optionally a character string can be passed with the currently supported option being "cumulative". This fits the model to cumulative cases and may be useful for certain data sets where the sampler gets stuck or struggles to initialise. See init_cumulative_fit() for details. This implementation is based on the approach taken in epidemia authored by James Scott.
return_fit	Logical, defaults to TRUE. Should the fit stan model be returned.
...	Additional parameters to pass underlying option functions.

Value

A list of arguments to pass to the appropriate rstan functions.

See Also

rstan_opts

Examples

```
# using default of rstan::sampling
stan_opts(samples = 1000)

# using vb
stan_opts(method = "vb")
```

---

summarise_key_measures

Summarise rt and cases

Description

Maturing Produces summarised data frames of output across regions. Used internally by regional_summary.

Usage

```
summarise_key_measures(
  regional_results = NULL,
  results_dir = NULL,
  summary_dir = NULL,
  type = "region",
  date = "latest"
)
```

Arguments

regional_results	A list of dataframes as produced by `get_regional_results`
results_dir	Character string indicating the directory from which to extract results.
summary_dir	Character string the directory into which to save results as a csv.
type	Character string, the region identifier to apply (defaults to region).
date	A Character string (in the format "yyyy-mm-dd") indicating the date to extract data for. Defaults to "latest" which finds the latest results available.

Value

A list of summarised Rt, cases by date of infection and cases by date of report

See Also

regional_summary

---

summarise_results *Summarise Real-time Results*

Description

Questioning Used internally by `regional_summary` to produce a summary table of results. May be streamlined in later releases.

Usage

```
summarise_results(
  regions,
  summaries = NULL,
  results_dir = NULL,
  target_date = NULL,
  region_scale = "Region"
)
```

Arguments

regions	An character string containing the list of regions to extract results for (must all have results for the same target date).
summaries	A list of summary data frames as output by epinow
results_dir	An optional character string indicating the location of the results directory to extract results from.
target_date	A character string indicating the target date to extract results for. All regions must have results for this date.
region_scale	A character string indicating the name to give the regions being summarised.

Value

A list of summary data

---

summary.epinow *Summary output from epinow*

Description

Stable summary method for class "epinow".

Usage

```
## S3 method for class 'epinow'
summary(object, output = "estimates", date = NULL, params = NULL, ...)
```

Arguments

object	A list of output as produced by "epinow".
output	A character string of output to summarise. Defaults to "estimates" but also supports "forecast", and "estimated_reported_cases".
date	A date in the form "yyyy-mm-dd" to inspect estimates for.
params	A character vector of parameters to filter for.
...	Pass additional summary arguments to lower level methods

Value

Returns a data frame of summary output

See Also

summary.estimate_infections epinow

summary.estimate_infections *Summary output from estimate_infections*

Description

Stable summary method for class "estimate_infections".

Usage

```
## S3 method for class 'estimate_infections'
summary(object, type = "snapshot", date = NULL, params = NULL, ...)
```

Arguments

object	A list of output as produced by "estimate_infections".
type	A character vector of data types to return. Defaults to "snapshot" but also supports "parameters". "snapshot" returns a summary at a given date (by default the latest date informed by data). "parameters" returns summarised parameter estimates that can be further filtered using params to show just the parameters of interest and date.
date	A date in the form "yyyy-mm-dd" to inspect estimates for.
params	A character vector of parameters to filter for.
...	Pass additional arguments to report_summary

Value

Returns a data frame of summary output

See Also

summary estimate_infections report_summary

theme_map *Custom Map Theme*

Description

Questioning Applies a custom map theme to be used with global_map, country_map, and other ggplot2 maps. Status of this function is currently questioning as it is uncertain if it is in use. Future releases may depreciate it.

Usage

```
theme_map(
  map = NULL,
  continuous = FALSE,
  variable_label = NULL,
  trans = "identity",
  fill_labels = NULL,
  scale_fill = NULL,
  breaks = NULL,
  ...
)
```

Arguments

map	ggplot2 map object
continuous	Logical defaults to FALSE. Is the fill variable continuous.
variable_label	A character string indicating the variable label to use. If not supplied then the underlying variable name is used.
trans	A character string specifying the transform to use on the specified metric. Defaults to no transform ("identity"). Other options include log scaling ("log") and log base 10 scaling ("log10"). For a complete list of options see ggplot2::continous_scale.
fill_labels	A function to use to allocate legend labels. An example (used below) is scales::percent, which can be used for percentage data.
scale_fill	Function to use for scaling the fill. Defaults to a custom ggplot2::scale_fill_manual, which expects the possible values to be "Increasing", "Likely increasing", "Likely decreasing", "Decreasing" or "Unsure".
breaks	Breaks to use in legend. Defaults to ggplot2::waiver.
...	Additional arguments passed to the scale_fill function

Value

A ggplot2 object

---

trunc_opts *Truncation Distribution Options*

Description

Stable Returns a truncation distribution formatted for usage by downstream functions. See estimate_truncation for an approach to estimate this distribution.

Usage

```
trunc_opts(dist = NULL)
```

Arguments

dist	A list defining the truncation distribution, defaults to NULL in which case no truncation is used. Must have the following elements if defined: "mean", "mean_sd", "sd_mean", "sd_sd", and "max" defining a truncated log normal (with all parameters except for max defined in logged form).

Value

A list summarising the input truncation distribution.

See Also

convert_to_logmean convert_to_logsd bootstrapped_dist_fit

Examples

```
# no truncation
trunc_opts()
```

---

tune_inv_gamma *Tune an Inverse Gamma to Achieve the Target Truncation*

Description

Questioning Allows an inverse gamma distribution to be. tuned so that less than 0.01 of its probability mass function falls outside of the specified bounds. This is required when using an inverse gamma prior, for example for a Gaussian process. As no inverse gamma priors are currently in use and this function has some stability issues it may be deprecated at a later date.

Usage

```
tune_inv_gamma(lower = 2, upper = 21)
```

Arguments

lower	Numeric, defaults to 2. Lower truncation bound.
upper	Numeric, defaults to 21. Upper truncation bound.

Value

A list of alpha and beta values that describe a inverse gamma distribution that achieves the target truncation.

Examples

```
tune_inv_gamma(lower = 2, upper = 21)
```

update_horizon *Updates Forecast Horizon Based on Input Data and Target*

Description

Stable Makes sure that a forecast is returned for the user specified time period beyond the target date.

Usage

```
update_horizon(horizon, target_date, reported_cases)
```

Arguments

horizon	Numeric, defaults to 7. Number of days into the future to forecast.
target_date	Date, defaults to maximum found in the data if not specified.
reported_cases	A data frame of confirmed cases (confirm) by date (date). confirm must be integer and date must be in date format.

Value

Numeric forecast horizon adjusted for the users intention

update_list *Update a List*

Description

Stable Used to handle updating settings in a list. For example when making changes to `opts_list` output.

Usage

```
update_list(defaults = list(), optional = list())
```

Arguments

defaults	A list of default settings
optional	A list of optional settings to override defaults

Value

A list

Index

∗ **datasets**
  example_confirmed, 42
  generation_times, 54
  incubation_periods, 63

adjust_infection_to_report, 5
allocate_delays, 6
allocate_empty, 7

backcalc_opts, 8
bootstrapped_dist_fit, 9

calc_CrI, 10
calc_CrIs, 11
calc_summary_measures, 11
calc_summary_stats, 12
clean_nowcasts, 13
clean_regions, 13
construct_output, 14
convert_to_logmean, 14
convert_to_logsd, 15
copy_results_to_latest, 16
country_map, 16
create_backcalc_data, 18
create_clean_reported_cases, 19
create_future_rt, 19
create_gp_data, 20
create_initial_conditions, 21
create_obs_model, 21
create_rt_data, 22
create_shifted_cases, 23
create_stan_args, 24
create_stan_data, 25

delay_opts, 26
dist_fit, 27
dist_skel, 28

epinow, 29
estimate_delay, 33
estimate_infections, 33
estimate_secondary, 37
estimate_truncation, 40
estimates_by_report_date, 32
example_confirmed, 42
expose_stan_fns, 43
extract_CrIs, 43
extract_inits, 44
extract_parameter, 44
extract_parameter_samples, 45
extract_stan_param, 46
extract_static_parameter, 46

filter_opts, 47
fit_model_with_nuts, 48
fit_model_with_vb, 48
forecast_infections, 49
forecast_secondary, 51
format_fit, 52

gamma_dist_def, 53
generation_times, 54
get_dist, 54
get_generation_time, 55
get_incubation_period, 56
get_raw_result, 56
get_regional_results, 57
get_regions, 58
get_regions_with_most_reports, 59
global_map, 59
gp_opts, 61
growth_to_R, 62

incubation_periods, 63
init_cumulative_fit, 63

lognorm_dist_def, 64

make_conf, 65
map_prob_change, 65
match_output_arguments, 66

obs_opts, 67
opts_list, 68

plot *(*plot.estimate_infections*)*, 69
plot.epinow, 69
plot.estimate_infections, 69
plot.estimate_secondary, 70
plot.estimate_truncation, 71
plot_CrIs, 71
plot_estimates, 72
plot_estimates(), *83*
plot_summary, 73
process_region, 74
process_regions, 75

R_to_growth, 91
regional_epinow, 75
regional_runtimes, 78
regional_summary, 80
report_cases, 81
report_plots, 83
report_summary, 84
rstan_opts, 85
rstan_sampling_opts, 86
rstan_vb_opts, 87
rt_opts, 88
run_region, 89

sample_approx_dist, 92
save_estimate_infections, 93
save_forecast_infections, 94
save_input, 95
secondary_opts, 95
setup_default_logging, 96
setup_dt, 97
setup_future, 97
setup_logging, 98
setup_target_folder, 99
simulate_cases, 99
simulate_infections, 101
stan_opts, 102
summarise_key_measures, 103
summarise_results, 104
summary *(*summary.epinow*)*, 105
summary.epinow, 105
summary.estimate_infections, 106

theme_map, 106
trunc_opts, 107
tune_inv_gamma, 108

update_horizon, 109
update_list, 109