Package 'EpiNow2'

September 1, 2020

Type Package

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

Version 1.1.0

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

Encoding UTF-8

LazyData true

Imports cowplot, data.table, futile.logger (>= 1.4), future.apply, ggplot2, HDInterval, lubridate, methods, patchwork, purrr, Rcpp (>= 0.12.0), rstan (>= 2.18.1), rstantools (>= 2.0.0), scales, stats, truncnorm,

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

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)

SystemRequirements GNU make

Language en-US

NeedsCompilation yes

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

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

Repository CRAN

Date/Publication 2020-09-01 16:10:03 UTC

R topics documented:

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

Description

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 <- EpiNow2::example_confirmed[, `:=`(cases = as.integer(confirm))]

## Define a single report delay distribution
delay_def <- EpiNow2::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 <- EpiNow2::lognorm_dist_def(mean = EpiNow2::covid_incubation_period[1, ]$mean,
                                    mean_sd = EpiNow2::covid_incubation_period[1, ]$mean_sd,
                                             sd = EpiNow2::covid_incubation_period[1, ]$sd,
                                        sd_sd = EpiNow2::covid_incubation_period[1, ]$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)
```

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

Description

Fit a Subsampled Bootstrap to Integer Values and Summarise Distribution Parameters

Usage

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

Arguments

`values`	Numeric vector of integer 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.
`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")

## Inspect
out
```

---

clean_nowcasts *Clean Nowcasts for a Supplied Date*

Description

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 = NULL)
```

Arguments

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

---

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

Description

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.

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")

}
```

`covid_generation_times` *Covid Generation Time Estimates*

Description

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

Usage

```
covid_generation_times
```

Format

A `data.table` of summarising the distribution

covid_incubation_period *Covid Incubation Period*

Description

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

Usage

```
covid_incubation_period
```

Format

A vector giving the probability for each day

covid_serial_intervals *Covid Generation Serial Intervals*

Description

Default Covid serial interval estimates. See here for details: https://github.com/epiforecasts/EpiNow2/blob/master/data-raw/nishiura-serial-interval.R

Usage

```
covid_serial_intervals
```

Format

A vector giving the probability for each day

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

Description

Fit an Integer Adjusted Exponential, Gamma or Lognormal distributions

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

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

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

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

Estimate Rt and cases by date of infection, forecast into the future, transform to date of report and then save summary measures and plots.

Usage

```
epinow(
  reported_cases,
  family = "negbin",
  generation_time,
  delays,
 gp = list(basis_prop = 0.3, boundary_scale = 2, lengthscale_mean = 0, lengthscale_sd
    = 2),
  rt_prior = list(mean = 1, sd = 1),
  model,
```

```
  prior_smoothing_window = 7,
  cores = 1,
  chains = 4,
  samples = 1000,
  warmup = 200,
  adapt_delta = 0.99,
  max_treedepth = 15,
  estimate_rt = TRUE,
  estimate_week_eff = TRUE,
  estimate_breakpoints = FALSE,
  burn_in = 0,
  stationary = FALSE,
  fixed = FALSE,
  fixed_future_rt = FALSE,
  return_fit = FALSE,
  forecast_model,
  horizon = 7,
  ensemble_type = "mean",
  return_estimates = TRUE,
  target_folder,
  target_date,
  verbose = TRUE,
  debug = FALSE
)
```

Arguments

`reported_cases`	A data frame of confirmed cases (confirm) by date (date). confirm must be integer and date must be in date format.
`family`	A character string indicating the reporting model to use. Defaults to negative binomial ("negbin") with poisson ("poisson") also supported.
`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 list of delays (i.e incubation period/reporting delay) between infection and report. Each list entry must also be 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 that delay (assuming a lognormal distribution with all parameters excepting the max allowed value on the log scale).
`gp`	List controlling the Gaussian process approximation. Must contain the `basis_prop` (number of basis functions based on scaling the time points) which defaults to 0.3 and must be between 0 and 1 (increasing this increases the accuracy of the approximation and the cost of additional compute. Must also contain the `boundary_scale` (multiplied by half the range of the input time series). Increasing this increases the accuracy of the approximation at the cost of additional compute. See here: https://arxiv.org/abs/2004.11408 for more information on

setting these parameters. Can optionally also contain the `lengthscale_mean` and `lengthscale_sd`. If these are specified this will override the defaults of 0 and 2 (normal distributed truncated at zero).

`rt_prior` A list contain the mean and standard deviation (sd) of the gamma distributed prior for Rt. By default this is assumed to be mean 1 with a standard deviation of 1.

`model` A compiled stan model. By default uses the internal package model.

`prior_smoothing_window`
Numeric defaults to 7. The number of days over which to take a rolling average for the prior based on reported cases.

`cores` Numeric, defaults to 2. The number of cores to use when fitting the stan model.

`chains` Numeric, defaults to 2. The number of MCMC chains to use.

`samples` Numeric, defaults to 1000. Number of samples post warmup.

`warmup` Numeric, defaults to 200. Number of iteration of warmup to use.

`adapt_delta` Numeric, defaults to 0.99. See ?rstan::sampling.

`max_treedepth` Numeric, defaults to 15. See ?rstan::sampling.

`estimate_rt` Logical, defaults TRUE. Should Rt be estimated when imputing infections.

`estimate_week_eff`
Logical, defaults TRUE. Should weekly reporting effects be estimated.

`estimate_breakpoints`
Logical, defaults to FALSE. Should breakpoints in Rt be estimated. If true then `reported_cases` must contain a `breakpoint` variable that is 1 on the dates with breakpoints and otherwise 0. Breakpoints are fit jointly with a global non-parametric effect and so represent a conservative estimate of breakpoint changes.

`burn_in` Numeric, defaults to 0. The number of initial estimates to discard. This argument may be used to reduce spurious findings when running `estimate_infections` on a partial timeseries (as the earliest estimates will not be informed by all cases that occurred only those supplied to `estimate_infections`). The combined delays used will inform the appropriate length of this burn in but 7 days is likely a sensible starting point.

`stationary` Logical, defaults to FALSE. Should Rt be estimated with a global mean. When estimating Rt this should substantially improve run times but will revert to the global average for real time and forecasted estimates. This setting is most appropriate when estimating historic Rt or when combined with breakpoints.

`fixed` Logical, defaults to FALSE. If TRUE then a Gaussian process is not used and Rt is assumed to be constant over time (apart from any manually included breakpoints). If `estimate_rt` is FALSE then this reduces the backcalculation to a simple mean shift. This option can be used to produce a null model estimate, to produce a single Rt estimate for a short timeseries or as part of a wider analysis on the impact of interventions.

`fixed_future_rt`
Logical, defaults to FALSE. IF TRUE then the estimated Rt from the last time point with data is used for all future time points without data.

`return_fit` Logical, defaults to FALSE. Should the fitted stan model be returned.

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)}.
horizon	Numeric, defaults to 7. Number of days into the future to forecast.
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.
return_estimates	Logical, defaults to TRUE. Should estimates be returned.
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.
verbose	Logical, defaults to TRUE. Should verbose progress messages be printed.
debug	Logical, defaults to FALSE. Enables debug model in which additional diagnostics are available

Value

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

Examples

```
## Construct example distributions
generation_time <- list(mean = EpiNow2::covid_generation_times[1, ]$mean,
                        mean_sd = EpiNow2::covid_generation_times[1, ]$mean_sd,
                        sd = EpiNow2::covid_generation_times[1, ]$sd,
                        sd_sd = EpiNow2::covid_generation_times[1, ]$sd_sd,
                        max = 30)

incubation_period <- list(mean = EpiNow2::covid_incubation_period[1, ]$mean,
                          mean_sd = EpiNow2::covid_incubation_period[1, ]$mean_sd,
                          sd = EpiNow2::covid_incubation_period[1, ]$sd,
                          sd_sd = EpiNow2::covid_incubation_period[1, ]$sd_sd,
                          max = 30)

reporting_delay <- EpiNow2::bootstrapped_dist_fit(rlnorm(100, log(6), 1))
## Set max allowed delay to 60 days to truncate computation
reporting_delay$max <- 60

## Example case data
reported_cases <- EpiNow2::example_confirmed[1:40]

## Report Rt along with forecasts
out <- epinow(reported_cases = reported_cases, generation_time = generation_time,
              delays = list(incubation_period, reporting_delay),
              rt_prior = list(mean = 1, sd = 1),
             samples = 1000, warmup = 200, cores = ifelse(interactive(), 4, 1), chains = 4,
              verbose = TRUE, return_fit = TRUE)

out
```

```
## For optional forecasting
if(requireNamespace("EpiSoon")){
   if(requireNamespace("forecastHybrid")){

   ## Report Rt along with forecasts
   out <- epinow(reported_cases = cases, generation_time = generation_time,
                 delays = list(incubation_period, reporting_delay),
                 rt_prior = list(mean = 1, sd = 1),
                 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"), ...)},
               samples = 1000, warmup = 500, cores = ifelse(interactive(), 4, 1), chains = 4,
                 verbose = TRUE, return_fit = TRUE)

out
   }
}
```

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

Description

This function uses a non-parametric approach to reconstruct cases by date of infection from reported cases. It can optionally then estimate the time-varying reproduction number and the rate of growth.

Usage

```
estimate_infections(
  reported_cases,
  family = "negbin",
  generation_time,
  delays,
  gp = list(basis_prop = 0.3, boundary_scale = 2),
  rt_prior = list(mean = 1, sd = 1),
  prior_smoothing_window = 7,
  horizon = 7,
  model,
  cores = 1,
  chains = 4,
  samples = 1000,
  warmup = 200,
```

```
  estimate_rt = TRUE,
  estimate_week_eff = TRUE,
  estimate_breakpoints = FALSE,
  burn_in = 0,
  stationary = FALSE,
  fixed = FALSE,
  fixed_future_rt = FALSE,
  adapt_delta = 0.99,
  max_treedepth = 15,
  return_fit = FALSE,
  verbose = TRUE,
  debug = FALSE
)
```

Arguments

reported_cases	A data frame of confirmed cases (confirm) by date (date). confirm must be integer and date must be in date format.
family	A character string indicating the reporting model to use. Defaults to negative binomial ("negbin") with poisson ("poisson") also supported.
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 list of delays (i.e incubation period/reporting delay) between infection and report. Each list entry must also be 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 that delay (assuming a lognormal distribution with all parameters excepting the max allowed value on the log scale).
gp	List controlling the Gaussian process approximation. Must contain the basis_prop (number of basis functions based on scaling the time points) which defaults to 0.3 and must be between 0 and 1 (increasing this increases the accuracy of the approximation and the cost of additional compute. Must also contain the boundary_scale (multiplied by half the range of the input time series). Increasing this increases the accuracy of the approximation at the cost of additional compute. See here: https://arxiv.org/abs/2004.11408 for more information on setting these parameters. Can optionally also contain the lengthscale_mean and lengthscale_sd. If these are specified this will override the defaults of 0 and 2 (normal distributed truncated at zero).
rt_prior	A list contain the mean and standard deviation (sd) of the gamma distributed prior for Rt. By default this is assumed to be mean 1 with a standard deviation of 1.
prior_smoothing_window	Numeric defaults to 7. The number of days over which to take a rolling average for the prior based on reported cases.
horizon	Numeric, defaults to 7. Number of days into the future to forecast.

model	A compiled stan model. By default uses the internal package model.
cores	Numeric, defaults to 2. The number of cores to use when fitting the stan model.
chains	Numeric, defaults to 2. The number of MCMC chains to use.
samples	Numeric, defaults to 1000. Number of samples post warmup.
warmup	Numeric, defaults to 200. Number of iteration of warmup to use.
estimate_rt	Logical, defaults TRUE. Should Rt be estimated when imputing infections.
estimate_week_eff	Logical, defaults TRUE. Should weekly reporting effects be estimated.
estimate_breakpoints	Logical, defaults to FALSE. Should breakpoints in Rt be estimated. If true then reported_cases must contain a breakpoint variable that is 1 on the dates with breakpoints and otherwise 0. Breakpoints are fit jointly with a global non-parametric effect and so represent a conservative estimate of breakpoint changes.
burn_in	Numeric, defaults to 0. The number of initial estimates to discard. This argument may be used to reduce spurious findings when running estimate_infections on a partial timeseries (as the earliest estimates will not be informed by all cases that occurred only those supplied to estimate_infections). The combined delays used will inform the appropriate length of this burn in but 7 days is likely a sensible starting point.
stationary	Logical, defaults to FALSE. Should Rt be estimated with a global mean. When estimating Rt this should substantially improve run times but will revert to the global average for real time and forecasted estimates. This setting is most appropriate when estimating historic Rt or when combined with breakpoints.
fixed	Logical, defaults to FALSE. If TRUE then a Gaussian process is not used and Rt is assumed to be constant over time (apart from any manually included breakpoints). If estimate_rt is FALSE then this reduces the backcalculation to a simple mean shift. This option can be used to produce a null model estimate, to produce a single Rt estimate for a short timeseries or as part of a wider analysis on the impact of interventions.
fixed_future_rt	Logical, defaults to FALSE. IF TRUE then the estimated Rt from the last time point with data is used for all future time points without data.
adapt_delta	Numeric, defaults to 0.99. See ?rstan::sampling.
max_treedepth	Numeric, defaults to 15. See ?rstan::sampling.
return_fit	Logical, defaults to FALSE. Should the fitted stan model be returned.
verbose	Logical, defaults to TRUE. Should verbose progress messages be printed.
debug	Logical, defaults to FALSE. Enables debug model in which additional diagnostics are available

Examples

```
# Get example case counts
reported_cases <- EpiNow2::example_confirmed[1:50]
```

```
# Add a dummy breakpoint (used only when optionally estimating breakpoints)
reported_cases <- reported_cases[, breakpoint := data.table::fifelse(date == as.Date("2020-03-16"),
                                                                     1, 0)]
# Set up example generation time
generation_time <- list(mean = EpiNow2::covid_generation_times[1, ]$mean,
                        mean_sd = EpiNow2::covid_generation_times[1, ]$mean_sd,
                        sd = EpiNow2::covid_generation_times[1, ]$sd,
                        sd_sd = EpiNow2::covid_generation_times[1, ]$sd_sd,
                        max = 30)
# Set delays between infection and case report
# (any number of delays can be specifed here)
incubation_period <- list(mean = EpiNow2::covid_incubation_period[1, ]$mean,
                          mean_sd = EpiNow2::covid_incubation_period[1, ]$mean_sd,
                          sd = EpiNow2::covid_incubation_period[1, ]$sd,
                          sd_sd = EpiNow2::covid_incubation_period[1, ]$sd_sd,
                          max = 30)

reporting_delay <- list(mean = log(5),
                        mean_sd = log(2),
                        sd = log(2),
                        sd_sd = log(1.5),
                        max = 30)

# Run model with default settings
def <- estimate_infections(reported_cases, family = "negbin",
                           generation_time = generation_time,
                           delays = list(incubation_period, reporting_delay),
                       samples = 1000, warmup = 200, cores = ifelse(interactive(), 4, 1),
                       chains = 4, estimate_rt = TRUE, verbose = FALSE, return_fit = TRUE)


# Plot output
plots <- report_plots(summarised_estimates = def$summarised,
                      reported = reported_cases)

plots$summary

# Run model with Rt fixed into the future
fixed_rt <- estimate_infections(reported_cases, family = "negbin",
                                generation_time = generation_time,
                                delays = list(incubation_period, reporting_delay),
                          samples = 1000, warmup = 200, cores = ifelse(interactive(), 4, 1),
                                chains = 4, estimate_rt = TRUE, fixed_future_rt = TRUE,
                                return_fit = TRUE)


# Plot output
plots <- report_plots(summarised_estimates = fixed_rt$summarised,
                      reported = reported_cases)

plots$summary
```

```
# Run the model with default settings on a later snapshot of
# data (use burn_in here to remove the first week of
# estimates that may be impacted by this most).
snapshot_cases <- EpiNow2::example_confirmed[80:130]
snapshot <- estimate_infections(snapshot_cases, family = "negbin",
                                generation_time = generation_time,
                                delays = list(incubation_period, reporting_delay),
                          samples = 1000, warmup = 400, cores = ifelse(interactive(), 4, 1),
                          chains = 4, estimate_rt = TRUE, verbose = FALSE, return_fit = TRUE,
                                burn_in = 7)


# Plot output
plots <- report_plots(summarised_estimates = snapshot$summarised,
                      reported = snapshot_cases)

plots$summary

## Run model with stationary Rt assumption (likely to provide biased real-time estimates)
stat <- estimate_infections(reported_cases, family = "negbin",
                            generation_time = generation_time,
                            delays = list(incubation_period, reporting_delay),
                       samples = 1000, warmup = 200, cores = ifelse(interactive(), 4, 1),
                            chains = 4, estimate_rt = TRUE, stationary = TRUE,
                            verbose = FALSE, return_fit = TRUE)


# Plot output
plots <- report_plots(summarised_estimates = stat$summarised,
                      reported = reported_cases)

plots$summary

# Run model with fixed Rt assumption
fixed <- estimate_infections(reported_cases, family = "negbin",
                             generation_time = generation_time,
                             delays = list(incubation_period, reporting_delay),
                        samples = 1000, warmup = 200, cores = ifelse(interactive(), 4, 1),
                             chains = 4, estimate_rt = TRUE, fixed = TRUE,
                             verbose = FALSE, return_fit = TRUE)


# Plot output
plots <- report_plots(summarised_estimates = fixed$summarised,
                      reported = reported_cases)


plots$summary


# Run model with no delays
no_delay <- estimate_infections(reported_cases, family = "negbin",
                             generation_time = generation_time,
```

```
                              samples = 1000, warmup = 200,
                              cores = ifelse(interactive(), 4, 1),
                              chains = 4, estimate_rt = TRUE,
                              verbose = FALSE, return_fit = TRUE)


# Plot output
plots <- report_plots(summarised_estimates = no_delay$summarised,
                      reported = reported_cases)


plots$summary

# Run model with breakpoints
bkp <- estimate_infections(reported_cases, family = "negbin",
                           generation_time = generation_time,
                           delays = list(incubation_period, reporting_delay),
                         samples = 1000, warmup = 200, cores = ifelse(interactive(), 4, 1),
                           chains = 4, estimate_rt = TRUE, estimate_breakpoints = TRUE,
                           verbose = FALSE, return_fit = TRUE)


# Plot output
plots <- report_plots(summarised_estimates = bkp$summarised,
                      reported = reported_cases)

plots$summary

# Run model with breakpoints but with constrained non-linear change over time
# This formulation may increase the apparent effect of the breakpoint but needs to be tested using
# model fit criteria (i.e LFO).
cbkp <- estimate_infections(reported_cases, family = "negbin",
                            generation_time = generation_time,
                            gp = list(basis_prop = 0.3, boundary_scale = 2,
                                      lengthscale_mean = 20, lengthscale_sd = 1),
                            delays = list(incubation_period, reporting_delay),
                          samples = 1000, warmup = 200, cores = ifelse(interactive(), 4, 1),
                            chains = 4, estimate_rt = TRUE, estimate_breakpoints = TRUE,
                            verbose = FALSE, return_fit = TRUE)


# Plot output
plots <- report_plots(summarised_estimates = cbkp$summarised,
             reported = reported_cases)

plots$summary

# Pull out breakpoint summary
cbkp$summarised[variable == "breakpoints"]

# Run model with breakpoints but otherwise static Rt
# This formulation may increase the apparent effect of the breakpoint but needs to be tested using
# model fit criteria (i.e LFO).
```

```
fbkp <- estimate_infections(reported_cases, family = "negbin",
                            generation_time = generation_time,
                            delays = list(incubation_period, reporting_delay),
                          samples = 1000, warmup = 200, cores = ifelse(interactive(), 4, 1),
                            chains = 4, estimate_breakpoints = TRUE, fixed = TRUE,
                            verbose = FALSE, return_fit = TRUE)


# Plot output
plots <- report_plots(summarised_estimates = fbkp$summarised,
              reported = reported_cases)

plots$summary

# Pull out breakpoint summary
fbkp$summarised[variable == "breakpoints"]

# Run model without Rt estimation (just backcalculation)
backcalc <- estimate_infections(reported_cases, family = "negbin",
                                generation_time = generation_time,
                                delays = list(incubation_period, reporting_delay),
                           samples = 1000, warmup = 200, cores = ifelse(interactive(), 4, 1),
                          chains = 4, estimate_rt = FALSE, verbose = FALSE, return_fit = TRUE)


# plot just infections as report_plots does not support the backcalculation only model
plot_estimates(estimate = backcalc$summarised[variable == "infections"],
               reported = reported_cases, ylab = "Cases")
```

`example_confirmed` *Example Confirmed Case Data Set*

Description

An example data frame of observed cases

Usage

```
example_confirmed
```

Format

A data frame containing cases reported on each date.

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

Description

This function 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,
  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)}.
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 <- EpiNow2::example_confirmed[1:40]

generation_time <- list(mean = EpiNow2::covid_generation_times[1, ]$mean,
                        mean_sd = EpiNow2::covid_generation_times[1, ]$mean_sd,
                        sd = EpiNow2::covid_generation_times[1, ]$sd,
                        sd_sd = EpiNow2::covid_generation_times[1, ]$sd_sd,
                        max = 30)

incubation_period <- list(mean = EpiNow2::covid_incubation_period[1, ]$mean,
                          mean_sd = EpiNow2::covid_incubation_period[1, ]$mean_sd,
                          sd = EpiNow2::covid_incubation_period[1, ]$sd,
                          sd_sd = EpiNow2::covid_incubation_period[1, ]$sd_sd,
                          max = 30)

reporting_delay <- list(mean = log(5),
                        mean_sd = log(2),
                        sd = log(2),
                        sd_sd = log(1.5),
                        max = 30)

## Estimate Rt and infections from data
out <- estimate_infections(reported_cases, family = "negbin",
                           generation_time = generation_time,
                           delays = list(incubation_period, reporting_delay),
                           rt_prior = list(mean = 1, sd = 1),
                           samples = 1000, warmup = 500,
                           cores = ifelse(interactive(), 4, 1), chains = 4,
                           estimate_rt = TRUE,
                           verbose = FALSE, return_fit = TRUE)

## 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
```

```
 }
}
```

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

Description

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]]
```

---

get_raw_result *Get a Single Raw Result*

Description

Get a Single Raw Result

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

Get Combined Regional Results

Usage

```
get_regional_results(regional_output, results_dir, date, 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.
`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 <- list(mean = EpiNow2::covid_generation_times[1, ]$mean,
                        mean_sd = EpiNow2::covid_generation_times[1, ]$mean_sd,
                        sd = EpiNow2::covid_generation_times[1, ]$sd,
                        sd_sd = EpiNow2::covid_generation_times[1, ]$sd_sd,
                        max = 30)

incubation_period <- list(mean = EpiNow2::covid_incubation_period[1, ]$mean,
                          mean_sd = EpiNow2::covid_incubation_period[1, ]$mean_sd,
                          sd = EpiNow2::covid_incubation_period[1, ]$sd,
                          sd_sd = EpiNow2::covid_incubation_period[1, ]$sd_sd,
                          max = 30)

reporting_delay <- list(mean = log(10),
                        mean_sd = 0.8,
                        sd = log(2),
                        sd_sd = 0.1,
                        max = 30)
```

```
# Uses example case vector from EpiSoon
cases <- EpiNow2::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,
                                incubation_period = incubation_period,
                                reporting_delay = reporting_delay,
                           samples = 2000, warmup = 200, cores = ifelse(interactive(), 4, 1),
                                adapt_delta = 0.95, chains = 4, verbose = TRUE,
                                summary = FALSE)

get_regional_results(regional_out$regional, forecast = TRUE)
```

`get_regions` *Get Folders with Nowcast Results*

Description

Get Folders with Nowcast Results

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.

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

Description

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.

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)
}
```

`growth_to_R` *Convert Growth Rates to Reproduction numbers.*

Description

See here for justification.

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)
```

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

Description

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

Format Credible Intervals

Usage

```
make_conf(value, round_type = NULL, digits = 0)
```

Arguments

value	List of value to map into a string. Requires, point, lower, and upper.
round_type	Function, type of rounding to apply. Defaults to round.
digits	Numeric, defaults to 0. Amount of rounding to apply

Value

A character vector formatted for reporting

Examples

```
value <- list(point = 1, lower = 0, upper = 3)

make_conf(value, round_type = round, digits = 0)
```

map_prob_change *Categorise the Probability of Change for Rt*

Description

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)
```

plot_estimates *Plot Estimates*

Description

Plot Estimates

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 <- EpiNow2::example_confirmed[1:40]


## Set up example generation time
generation_time <- list(mean = EpiNow2::covid_generation_times[1, ]$mean,
                        mean_sd = EpiNow2::covid_generation_times[1, ]$mean_sd,
                        sd = EpiNow2::covid_generation_times[1, ]$sd,
                        sd_sd = EpiNow2::covid_generation_times[1, ]$sd_sd,
                        max = 30)
## Set
incubation_period <- list(mean = EpiNow2::covid_incubation_period[1, ]$mean,
                          mean_sd = EpiNow2::covid_incubation_period[1, ]$mean_sd,
                          sd = EpiNow2::covid_incubation_period[1, ]$sd,
                          sd_sd = EpiNow2::covid_incubation_period[1, ]$sd_sd,
                          max = 30)

reporting_delay <- list(mean = log(5),
                        mean_sd = log(2),
                        sd = log(2),
                        sd_sd = log(1.5),
                        max = 30)

## Run model
out <- EpiNow2::estimate_infections(cases, family = "negbin",
                                    generation_time = generation_time,
                                    delays = list(incubation_period, reporting_delay),
                                    samples = 1000, warmup = 200,
                                    cores = ifelse(interactive(), 4, 1),
                                  chains = 4, horizon = 7, estimate_rt = TRUE, verbose = TRUE)
## 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

Plot a Summary of the Latest Results

Usage

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

Arguments

`summary_results`	A data.able 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

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

Description

Estimates Rt by region. See the documentation for `epinow` for further information.

Usage

```
regional_epinow(
  reported_cases,
  target_folder,
  target_date,
  non_zero_points = 2,
  cores = 1,
  summary = TRUE,
  summary_dir,
  region_scale = "Region",
  all_regions_summary = TRUE,
  return_estimates = TRUE,
  max_plot = 10,
  ...
)
```

Arguments

`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.
`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 2.
`cores`	Numeric, defaults to 2. The number of cores to use when fitting the stan model.
`summary`	Logical, should summary measures be calculated.
`summary_dir`	A character string giving the directory in which to store summary of results.
`region_scale`	A character string indicating the name to give the regions being summarised.
`all_regions_summary`	Logical, defaults to `TRUE`. Should summary plots for all regions be returned rather than just regions of interest.
`return_estimates`	Logical, defaults to TRUE. Should estimates be returned.
`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.
`...`	Pass additional arguments to `epinow`

Value

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

Examples

```
## Construct example distributions
generation_time <- list(mean = EpiNow2::covid_generation_times[1, ]$mean,
                        mean_sd = EpiNow2::covid_generation_times[1, ]$mean_sd,
                        sd = EpiNow2::covid_generation_times[1, ]$sd,
                        sd_sd = EpiNow2::covid_generation_times[1, ]$sd_sd,
                        max = 30)

incubation_period <- list(mean = EpiNow2::covid_incubation_period[1, ]$mean,
                          mean_sd = EpiNow2::covid_incubation_period[1, ]$mean_sd,
                          sd = EpiNow2::covid_incubation_period[1, ]$sd,
                          sd_sd = EpiNow2::covid_incubation_period[1, ]$sd_sd,
                          max = 30)

reporting_delay <- list(mean = log(10),
                        mean_sd = log(2),
                        sd = log(2),
                        sd_sd = log(1.1),
                        max = 30)

## Uses example case vector
cases <- EpiNow2::example_confirmed[1:40]

cases <- data.table::rbindlist(list(
  data.table::copy(cases)[, region := "testland"],
  cases[, region := "realland"]))

## Run basic nowcasting pipeline
## Here we reduce the accuracy of the GP approximation in order to reduce runtime
out <- regional_epinow(reported_cases = cases,
                       generation_time = generation_time,
                       delays = list(incubation_period, reporting_delay),
                       adapt_delta = 0.9,
                       samples = 2000, warmup = 200, verbose = TRUE,
                       cores = ifelse(interactive(), 4, 1), chains = 4)
```

---

regional_summary *Generate Regional Summary Output*

Description

Generate Regional Summary Output

Usage

```
regional_summary(
  regional_output,
  reported_cases,
  results_dir,
  summary_dir,
  target_date,
  region_scale = "Region",
  all_regions = TRUE,
  return_summary = TRUE,
  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_summary	Logical, defaults to TRUE. Should summary measures be returned.
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

Examples

```
# Construct example distributions
generation_time <- list(mean = EpiNow2::covid_generation_times[1, ]$mean,
                        mean_sd = EpiNow2::covid_generation_times[1, ]$mean_sd,
                        sd = EpiNow2::covid_generation_times[1, ]$sd,
                        sd_sd = EpiNow2::covid_generation_times[1, ]$sd_sd,
                        max = 30)

incubation_period <- list(mean = EpiNow2::covid_incubation_period[1, ]$mean,
                          mean_sd = EpiNow2::covid_incubation_period[1, ]$mean_sd,
```

```
                          sd = EpiNow2::covid_incubation_period[1, ]$sd,
                          sd_sd = EpiNow2::covid_incubation_period[1, ]$sd_sd,
                          max = 30)

reporting_delay <- list(mean = log(10),
                        mean_sd = 0.8,
                        sd = log(2),
                        sd_sd = 0.1,
                        max = 30)

# Uses example case vector from EpiSoon
cases <- EpiNow2::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 = list(incubation_period, reporting_delay),
                                samples = 2000, warmup = 200, cores = 4,
                                adapt_delta = 0.95, chains = 4, verbose = TRUE,
                                summary = FALSE)

results_dir <- tempdir()

regional_summary(regional_output = regional_out$regional,
                 reported_cases = cases,
                 summary_dir = results_dir,
                 region_scale = "Country", all_regions = FALSE)
```

`report_cases` *Report case counts by date of report*

Description

Report case counts by date of report

Usage

```
report_cases(
  case_estimates,
  case_forecast = NULL,
  delays,
  type = "sample",
  reporting_effect
)
```

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 list of delays (i.e incubation period/reporting delay) between infection and report. Each list entry must also be 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 that delay (assuming a lognormal distribution with all parameters excepting the max allowed value on the log scale).
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.

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 <- EpiNow2::example_confirmed[1:40]


## Set up example generation time
generation_time <- list(mean = EpiNow2::covid_generation_times[1, ]$mean,
                        mean_sd = EpiNow2::covid_generation_times[1, ]$mean_sd,
                        sd = EpiNow2::covid_generation_times[1, ]$sd,
                        sd_sd = EpiNow2::covid_generation_times[1, ]$sd_sd,
                        max = 30)
## Set
incubation_period <- list(mean = EpiNow2::covid_incubation_period[1, ]$mean,
                          mean_sd = EpiNow2::covid_incubation_period[1, ]$mean_sd,
                          sd = EpiNow2::covid_incubation_period[1, ]$sd,
                          sd_sd = EpiNow2::covid_incubation_period[1, ]$sd_sd,
                          max = 30)

reporting_delay <- list(mean = log(5),
                        mean_sd = log(2),
                        sd = log(2),
                        sd_sd = log(1.5),
                        max = 30)
```

```
## Run model
out <- EpiNow2::estimate_infections(cases, family = "negbin",
                                    generation_time = generation_time,
                                    delays = list(incubation_period, reporting_delay),
                                    samples = 1000, warmup = 200,
                                    cores = ifelse(interactive(), 4, 1), chains = 4,
                                    estimate_rt =  FALSE, verbose = TRUE)


reported_cases <- report_cases(case_estimates = out$samples[variable == "infections"][,
                                                            cases := value][, value := NULL],
                               delays = list(incubation_period, reporting_delay),
                               type = "sample")

print(reported_cases)
```

`report_plots` *Report plots*

Description

Report plots

Usage

```
report_plots(summarised_estimates, reported, target_folder, 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 `ggplot2` object

Examples

```
## Define example cases
cases <- EpiNow2::example_confirmed[1:40]


## Set up example generation time
generation_time <- list(mean = EpiNow2::covid_generation_times[1, ]$mean,
                        mean_sd = EpiNow2::covid_generation_times[1, ]$mean_sd,
                        sd = EpiNow2::covid_generation_times[1, ]$sd,
                        sd_sd = EpiNow2::covid_generation_times[1, ]$sd_sd,
                        max = 30)
## Set
incubation_period <- list(mean = EpiNow2::covid_incubation_period[1, ]$mean,
                          mean_sd = EpiNow2::covid_incubation_period[1, ]$mean_sd,
                          sd = EpiNow2::covid_incubation_period[1, ]$sd,
                          sd_sd = EpiNow2::covid_incubation_period[1, ]$sd_sd,
                          max = 30)

reporting_delay <- list(mean = log(5),
                        mean_sd = log(2),
                        sd = log(2),
                        sd_sd = log(1.5),
                        max = 30)

## Run model
out <- EpiNow2::estimate_infections(cases, family = "negbin",
                                    generation_time = generation_time,
                                    delays = list(incubation_period, reporting_delay),
                                    samples = 1000, warmup = 200, cores = 4, chains = 4,
                                    horizon = 7, estimate_rt = TRUE, verbose = TRUE)


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

plots
```

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

Description

Provide Summary Statistics for Estimated Infections and Rt

Usage

```
report_summary(summarised_estimates, rt_samples)
```

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.

Value

A data.table containing formatted and numeric summary measures

---

R_to_growth *Convert Reproduction Numbers to Growth Rates*

Description

See here for justification.

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

Approximate Sampling a Distribution using Counts

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 <- EpiNow2::example_confirmed

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

## Reported case distribution
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")
```

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

Description

Simulate Cases by Date of Infection, Onset and Report

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`.

Examples

```
if(requireNamespace("EpiSoon")){

# Define an initial rt vector
rts <- c(rep(2, 20), (2 - 1:15 * 0.1), rep(0.5, 10))
rts
# Use the mean default generation interval for covid
# Generation time
generation_defs <- EpiNow2::gamma_dist_def(mean = generation_time$mean,
                                           mean_sd = generation_time$mean_sd,
                                           sd = generation_time$sd,
                                           sd_sd = generation_time$sd_sd,
                                           max_value = generation_time$max, samples = 1)

generate_pdf <- function(dist, max_value) {
   ## Define with 0 day padding
   sample_fn <- function(n, ...) {
     c(0, EpiNow2::dist_skel(n = n,
                 model = dist$model[[1]],
                 params = dist$params[[1]],
                 max_value = dist$max_value[[1]] - 1,
                 ...))
                  }
 dist_pdf <- sample_fn(0:(max_value - 1), dist = TRUE, cum = FALSE)

 return(dist_pdf)}

 generation_pdf <- generate_pdf(generation_defs, max_value = generation_defs$max)

# Sample a report delay as a lognormal
delay_def <- EpiNow2::lognorm_dist_def(mean = 5, mean_sd = 1,
                                       sd = 3, sd_sd = 1, max_value = 30,
                                       samples = 1, to_log = TRUE)


# Sample a incubation period (again using the default for covid)
incubation_def <- EpiNow2::lognorm_dist_def(mean = EpiNow2::covid_incubation_period[1, ]$mean,
                                  mean_sd = EpiNow2::covid_incubation_period[1, ]$mean_sd,
                                            sd = EpiNow2::covid_incubation_period[1, ]$sd,
                                     sd_sd = EpiNow2::covid_incubation_period[1, ]$sd_sd,
                                            max_value = 30, samples = 1)

# Simulate cases with a decrease in reporting at weekends
# and an increase on Monday
simulated_cases <- simulate_cases(rts, initial_cases = 100 ,
                                  initial_date = as.Date("2020-03-01"),
                                  generation_interval = generation_pdf,
```

```
                                  delay_defs = list(incubation_def, delay_def),
                                  reporting_effect = c(1.1, rep(1, 4), 0.95, 0.95))

print(simulated_cases)



# Simulate cases with a weekly reporting effect and
# stochastic noise in reporting (beyond the delay)
simulated_cases <- simulate_cases(rts, initial_cases = 100 ,
                                  initial_date = as.Date("2020-03-01"),
                                  generation_interval = generation_pdf,
                                  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(simulated_cases)
}
```

`summarise_key_measures` *Summarise rt and cases*

Description

Summarise rt and cases

Usage

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

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

---

`summarise_results` *Summarise Real-time Results*

Description

Summarise Real-time Results

Usage

```
summarise_results(
  regions,
  summaries,
  results_dir,
  target_date,
  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

theme_map *Custom Map Theme*

Description

Custom Map Theme

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

Index