

# Package: epinowcast (via r-universe)

October 9, 2026

**Title** A Bayesian Framework for Real-time Infectious Disease Surveillance

**Version** 0.7.0

**Description** A modular Bayesian framework for real-time infectious disease surveillance. Provides tools for nowcasting, reproduction number estimation, delay estimation, and forecasting from data subject to reporting delays, right-truncation, missing data, and incomplete ascertainment. Users can build models suited to their setting using a flexible formula interface supporting fixed effects, random effects, random walks, and time-varying parameters, with options including parametric and non-parametric delay distributions with optional modifiers (via discrete-time hazard models), renewal processes, observation models, missing data imputation, and stratified analyses with partial pooling. By jointly estimating disease dynamics and reporting patterns, our framework enables earlier and more reliable detection of trends. While designed with epidemiological applications in mind, the framework can be applied to any right-truncated time series count data.

**License** MIT + file LICENSE

**URL** <https://package.epinowcast.org>,  
<https://github.com/epinowcast/epinowcast/>

**BugReports** <https://github.com/epinowcast/epinowcast/issues/>

**Depends** R (>= 4.3.0)

**Imports** cli, data.table, reformulas, ggplot2, posterior, purrr, rlang, scales, lifecycle

**Suggests** bookdown, scoringutils, cmdstanr, cyclocomp, dplyr, lme4, loo (>= 2.4.1), primarycensored, knitr, RcppEigen, rmarkdown, spelling, testthat (>= 3.1.9), usethis, vdiff, withr

**VignetteBuilder** knitr

**Additional\_repositories** <https://stan-dev.r-universe.dev>

**Config/Needs/website** r-lib/pkgdown, epinowcast/enwtheme

**Config/Needs/hexsticker** hexSticker, sysfonts

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-GB

**LazyData** true

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**Repository** <https://epinowcast.r-universe.dev>

**Date/Publication** 2026-07-11 16:21:35 UTC

**RemoteUrl** <https://github.com/epinowcast/epinowcast>

**RemoteRef** v0.7.0

**RemoteSha** a5b5ecdfe588649a0b704739cd5d522954861f

## Contents

|                                         |    |
|-----------------------------------------|----|
| .check_primarycensored . . . . .        | 5  |
| .delay_draw_columns . . . . .           | 6  |
| .discretise_parametric_pmf . . . . .    | 6  |
| .expectation_is_minimal . . . . .       | 7  |
| add_max_observed_delay . . . . .        | 8  |
| add_pmfs . . . . .                      | 9  |
| ar . . . . .                            | 10 |
| arima . . . . .                         | 10 |
| arima_terms . . . . .                   | 11 |
| arma . . . . .                          | 12 |
| as_forecast_sample.epinowcast . . . . . | 13 |
| as_string_formula . . . . .             | 14 |
| build_ord_obs . . . . .                 | 14 |
| check_design_matrix_sparsity . . . . .  | 15 |
| check_group . . . . .                   | 16 |
| check_group_date_unique . . . . .       | 17 |
| check_max_delay . . . . .               | 17 |
| check_module . . . . .                  | 19 |
| check_modules_compatible . . . . .      | 20 |
| check_numeric_timestep . . . . .        | 20 |
| check_observation_indicator . . . . .   | 21 |
| check_quantiles . . . . .               | 22 |
| check_timestep . . . . .                | 22 |
| check_timestep_by_date . . . . .        | 23 |
| check_timestep_by_group . . . . .       | 24 |
| coerce_date . . . . .                   | 25 |
| coerce_dt . . . . .                     | 26 |
| construct_arima . . . . .               | 27 |

|                                                      |    |
|------------------------------------------------------|----|
| construct_gp . . . . .                               | 28 |
| construct_re . . . . .                               | 29 |
| construct_rw . . . . .                               | 30 |
| convolution_matrix . . . . .                         | 31 |
| date_to_numeric_modulus . . . . .                    | 32 |
| delay_only_ltotal . . . . .                          | 33 |
| delay_only_total . . . . .                           | 33 |
| enw_add_cumulative . . . . .                         | 34 |
| enw_add_cumulative_membership . . . . .              | 35 |
| enw_add_delay . . . . .                              | 36 |
| enw_add_incidence . . . . .                          | 37 |
| enw_add_latest_obs_to_nowcast . . . . .              | 38 |
| enw_add_max_reported . . . . .                       | 39 |
| enw_add_metaobs_features . . . . .                   | 40 |
| enw_add_pooling_effect . . . . .                     | 41 |
| enw_aggregate_cumulative . . . . .                   | 42 |
| enw_assign_group . . . . .                           | 44 |
| enw_complete_dates . . . . .                         | 45 |
| enw_construct_data . . . . .                         | 47 |
| enw_dayofweek_structural_reporting . . . . .         | 49 |
| enw_delay_categories . . . . .                       | 50 |
| enw_delay_quantiles . . . . .                        | 51 |
| enw_design . . . . .                                 | 52 |
| enw_effects_metadata . . . . .                       | 53 |
| enw_example . . . . .                                | 54 |
| enw_expectation . . . . .                            | 55 |
| enw_extend_date . . . . .                            | 56 |
| enw_filter_delay . . . . .                           | 57 |
| enw_filter_reference_dates . . . . .                 | 58 |
| enw_filter_reference_dates_by_report_start . . . . . | 60 |
| enw_filter_report_dates . . . . .                    | 61 |
| enw_fit_opts . . . . .                               | 62 |
| enw_flag_observed_observations . . . . .             | 63 |
| enw_formula . . . . .                                | 64 |
| enw_formula_as_data_list . . . . .                   | 67 |
| enw_get_cache . . . . .                              | 69 |
| enw_get_data . . . . .                               | 69 |
| enw_impute_na_observations . . . . .                 | 70 |
| enw_incidence_to_linelist . . . . .                  | 71 |
| enw_latest_data . . . . .                            | 72 |
| enw_linelist_to_incidence . . . . .                  | 72 |
| enw_manual_formula . . . . .                         | 74 |
| enw_metadata . . . . .                               | 75 |
| enw_metadata_delay . . . . .                         | 76 |
| enw_missing . . . . .                                | 78 |
| enw_missing_reference . . . . .                      | 79 |
| enw_model . . . . .                                  | 80 |
| enw_nowcast_samples . . . . .                        | 81 |

|                                             |     |
|---------------------------------------------|-----|
| enw_nowcast_summary . . . . .               | 82  |
| enw_obs . . . . .                           | 84  |
| enw_obs_at_delay . . . . .                  | 85  |
| enw_one_hot_encode_feature . . . . .        | 86  |
| enw_pathfinder . . . . .                    | 87  |
| enw_plot_delay_counts . . . . .             | 88  |
| enw_plot_delay_cumulative . . . . .         | 89  |
| enw_plot_delay_fraction . . . . .           | 90  |
| enw_plot_delay_quantiles . . . . .          | 91  |
| enw_plot_nowcast_quantiles . . . . .        | 91  |
| enw_plot_obs . . . . .                      | 92  |
| enw_plot_pp_quantiles . . . . .             | 93  |
| enw_plot_quantiles . . . . .                | 94  |
| enw_plot_theme . . . . .                    | 95  |
| enw_posterior . . . . .                     | 96  |
| enw_posterior_delay . . . . .               | 97  |
| enw_pp_summary . . . . .                    | 98  |
| enw_preprocess_data . . . . .               | 99  |
| enw_priors_as_data_list . . . . .           | 102 |
| enw_quantiles_to_long . . . . .             | 102 |
| enw_reference . . . . .                     | 103 |
| enw_reference_by_report . . . . .           | 105 |
| enw_replace_priors . . . . .                | 106 |
| enw_report . . . . .                        | 107 |
| enw_reporting_triangle . . . . .            | 108 |
| enw_reporting_triangle_to_long . . . . .    | 109 |
| enw_reps_with_complete_refs . . . . .       | 110 |
| enw_retrospective . . . . .                 | 111 |
| enw_rolling_sum . . . . .                   | 112 |
| enw_sample . . . . .                        | 113 |
| enw_set_cache . . . . .                     | 114 |
| enw_simulate_missing_reference . . . . .    | 115 |
| enw_stan_to_r . . . . .                     | 116 |
| enw_structural_reporting_metadata . . . . . | 117 |
| enw_summarise_samples . . . . .             | 118 |
| enw_unset_cache . . . . .                   | 119 |
| epinowcast . . . . .                        | 120 |
| extract_obs_metadata . . . . .              | 123 |
| extract_sparse_matrix . . . . .             | 124 |
| germany_covid19_hosp . . . . .              | 125 |
| get_internal_timestep . . . . .             | 125 |
| gp . . . . .                                | 126 |
| gp_terms . . . . .                          | 128 |
| is.Date . . . . .                           | 128 |
| latest_obs_as_matrix . . . . .              | 129 |
| ma . . . . .                                | 130 |
| parse_formula . . . . .                     | 130 |
| plot.enw_preprocess_data . . . . .          | 132 |

|                                                |     |
|------------------------------------------------|-----|
| <code>plot.epinowcast</code>                   | 133 |
| <code>print.enw_preprocess_data</code>         | 134 |
| <code>print.epinowcast</code>                  | 135 |
| <code>print.summary.enw_preprocess_data</code> | 136 |
| <code>re</code>                                | 136 |
| <code>remove_arima_terms</code>                | 137 |
| <code>remove_gp_terms</code>                   | 138 |
| <code>remove_profiling</code>                  | 139 |
| <code>remove_rw_terms</code>                   | 139 |
| <code>rw</code>                                | 140 |
| <code>rw_terms</code>                          | 141 |
| <code>split_formula_to_terms</code>            | 142 |
| <code>stan_fns_as_string</code>                | 143 |
| <code>subset_obs</code>                        | 143 |
| <code>summary.enw_preprocess_data</code>       | 144 |
| <code>summary.epinowcast</code>                | 145 |
| <code>write_stan_files_no_profile</code>       | 146 |

|              |            |
|--------------|------------|
| <b>Index</b> | <b>148</b> |
|--------------|------------|

---

`.check_primarycensored`

*Error if the primarycensored package is not installed*

---

## Description

Error if the primarycensored package is not installed

## Usage

```
.check_primarycensored(caller)
```

## Arguments

`caller`                      A string naming the calling function for the error message.

## Value

Invisibly TRUE; called for its side effect.

## See Also

Functions used for postprocessing of model fits `.delay_draw_columns()`, `.discretise_parametric_pmf()`, `build_ord_obs()`, `enw_add_latest_obs_to_nowcast()`, `enw_nowcast_samples()`, `enw_nowcast_summary()`, `enw_posterior()`, `enw_posterior_delay()`, `enw_pp_summary()`, `enw_quantiles_to_long()`, `enw_summarise_samples()`, `subset_obs()`

---

|                                  |                                                                            |
|----------------------------------|----------------------------------------------------------------------------|
| <code>.delay_draw_columns</code> | <i>Extract the index-ordered draw columns for a delay vector parameter</i> |
|----------------------------------|----------------------------------------------------------------------------|

---

### Description

Extract the index-ordered draw columns for a delay vector parameter

### Usage

```
.delay_draw_columns(fit, var)
```

### Arguments

|                  |                                                                |
|------------------|----------------------------------------------------------------|
| <code>fit</code> | A cmdstanr fit object.                                         |
| <code>var</code> | A scalar string naming the vector variable (e.g. "refp_mean"). |

### Value

A list with `draws` (a `draws_df`, or `NULL` if `var` is absent) and `cols` (the matching column names in index order).

### See Also

Functions used for postprocessing of model fits `.check_primarycensored()`, `.discretise_parametric_pmf()`, `build_ord_obs()`, `enw_add_latest_obs_to_nowcast()`, `enw_nowcast_samples()`, `enw_nowcast_summary()`, `enw_posterior()`, `enw_posterior_delay()`, `enw_pp_summary()`, `enw_quantiles_to_long()`, `enw_summarise_samples()`, `subset_obs()`

---

|                                         |                                                   |
|-----------------------------------------|---------------------------------------------------|
| <code>.discretise_parametric_pmf</code> | <i>Discretise a parametric delay distribution</i> |
|-----------------------------------------|---------------------------------------------------|

---

### Description

Computes the discretised probability mass function of a parametric delay distribution using primary-event censoring via `primarycensored::dprimarycensored()`, the same method `epinowcast` relies on for delay distributions. The PMF is over delays  $0:(\text{max\_delay} - 1)$ , truncated at `max_delay`. Parameters use the distribution's natural parameterisation (e.g. `meanlog/sdlog` for the lognormal); the `refp_mean_int/refp_sd_int` parameters are mapped accordingly.

### Usage

```
.discretise_parametric_pmf(mu, sigma, max_delay, distribution = "lognormal")
```

**Arguments**

|                           |                                                                                                               |
|---------------------------|---------------------------------------------------------------------------------------------------------------|
| <code>mu</code>           | Location parameter on the modelled scale ( <code>refp_mean_int</code> ).                                      |
| <code>sigma</code>        | Scale parameter on the modelled scale ( <code>refp_sd_int</code> ). Ignored for the exponential distribution. |
| <code>max_delay</code>    | Maximum delay (number of delay slots, delays <code>'0:(max_delay • 1)'</code> ).                              |
| <code>distribution</code> | One of "lognormal", "gamma", "exponential", or "loglogistic".                                                 |

**Value**

A numeric vector of length `max_delay` summing to 1.

**See Also**

Functions used for postprocessing of model fits `.check_primarycensored()`, `.delay_draw_columns()`, `build_ord_obs()`, `enw_add_latest_obs_to_nowcast()`, `enw_nowcast_samples()`, `enw_nowcast_summary()`, `enw_posterior()`, `enw_posterior_delay()`, `enw_pp_summary()`, `enw_quantiles_to_long()`, `enw_summarise_samples()`, `subset_obs()`

---

`.expectation_is_minimal`

*Is an expectation module minimal (intercept-only, no convolution)?*

---

**Description**

Used by `epinowcast()` to decide whether a delay-only fit can leave the supplied expectation untouched. A minimal expectation has no growth-rate covariates or random effects (`expr_fncol` and `expr_rncol` both zero), so replacing it with `enw_expectation(~1, ...)` would change nothing.

**Usage**

```
.expectation_is_minimal(expectation)
```

**Arguments**

`expectation` An expectation module as returned by `enw_expectation()`.

**Value**

A logical scalar.

**See Also**

Model modules `enw_expectation()`, `enw_fit_opts()`, `enw_missing()`, `enw_obs()`, `enw_reference()`, `enw_report()`

---

add\_max\_observed\_delay

*Add maximum observed delay*


---

## Description

This function calculates and adds the maximum observed delay for each group and reference date in the provided dataset. It first checks the validity of the observation indicator and then computes the maximum delay. If an observation indicator is provided, it further adjusts the maximum observed delay for unobserved data to be negative 1 (indicating no maximum observed).

## Usage

```
add_max_observed_delay(new_confirm, observation_indicator = NULL)
```

## Arguments

**new\_confirm** A data.table containing the columns: "reference\_date", "delay", ".group", "new\_confirm", and "max\_obs\_delay". As produced by [enw\\_preprocess\\_data\(\)](#) in the new\_confirm output with the addition of the "max\_obs\_delay" column as produced by [add\\_max\\_observed\\_delay\(\)](#).

**observation\_indicator** A character string specifying the column name in new\_confirm that indicates whether an observation is observed or not. This column should be a logical vector. If NULL (default), all observations are considered observed.

## Value

A data.table with the original columns of new\_confirm and an additional "max\_obs\_delay" column representing the maximum observed delay for each group and reference date. If an observation indicator is provided, unobserved data will have a "max\_obs\_delay" value of -1.

## See Also

Helper functions for model modules [add\\_pmfs\(\)](#), [convolution\\_matrix\(\)](#), [delay\\_only\\_ltotal\(\)](#), [delay\\_only\\_total\(\)](#), [enw\\_dayofweek\\_structural\\_reporting\(\)](#), [enw\\_reference\\_by\\_report\(\)](#), [enw\\_reps\\_with\\_complete\\_refs\(\)](#), [enw\\_structural\\_reporting\\_metadata\(\)](#), [extract\\_obs\\_metadata\(\)](#), [extract\\_sparse\\_matrix\(\)](#), [latest\\_obs\\_as\\_matrix\(\)](#)

add\_pmfs

*Add probability mass functions***Description**

This function allows the addition of probability mass functions (PMFs) to produce a new PMF. This is useful for example in the context of reporting delays where the PMF of the sum of two Poisson distributions is the convolution of the PMFs.

**Usage**

```
add_pmfs(pmfs)
```

**Arguments**

pmfs                      A list of vectors describing the probability mass functions to

**Value**

A vector describing the probability mass function of the sum of the

**See Also**

Helper functions for model modules [add\\_max\\_observed\\_delay\(\)](#), [convolution\\_matrix\(\)](#), [delay\\_only\\_ltotal\(\)](#), [delay\\_only\\_total\(\)](#), [enw\\_dayofweek\\_structural\\_reporting\(\)](#), [enw\\_reference\\_by\\_report\(\)](#), [enw\\_reps\\_with\\_complete\\_refs\(\)](#), [enw\\_structural\\_reporting\\_metadata\(\)](#), [extract\\_obs\\_metadata\(\)](#), [extract\\_sparse\\_matrix\(\)](#), [latest\\_obs\\_as\\_matrix\(\)](#)

**Examples**

```
# Sample and analytical PMFs for two Poisson distributions
x <- rpois(10000, 5)
xpmf <- dpois(0:20, 5)
y <- rpois(10000, 7)
ypmf <- dpois(0:20, 7)
# Add sampled Poisson distributions up to get combined distribution
z <- x + y
# Analytical convolution of PMFs
conv_pmf <- add_pmfs(list(xpmf, ypmf))
conv_cdf <- cumsum(conv_pmf)
# Empirical convolution of PMFs
cdf <- ecdf(z)(0:42)
# Compare sampled and analytical CDFs
plot(conv_cdf)
lines(cdf, col = "black")
```

---

|    |                                                         |
|----|---------------------------------------------------------|
| ar | <i>Autoregressive alias for <a href="#">arima()</a></i> |
|----|---------------------------------------------------------|

---

### Description

Thin wrapper around [arima\(\)](#) that fixes  $d = 0$  and  $q = 0$ . Matches the in-formula [ar\(\)](#) helper that brms users will be familiar with. Equivalent to `arima(time, by, p = p, d = 0, q = 0)`.

### Usage

```
ar(time, by, p = 1)
```

### Arguments

|      |                                                                                                                                                        |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| time | Time variable for the latent series; numeric.                                                                                                          |
| by   | Optional grouping variable. Each group draws an independent shock series; AR/MA parameters and the latent standard deviation are shared across groups. |
| p    | Autoregressive order. Defaults to 1.                                                                                                                   |

### Value

An `enw_arima_term` interpretable by [construct\\_arima\(\)](#).

### See Also

Functions used to help convert formulas into model designs [arima\(\)](#), [arima\\_terms\(\)](#), [arma\(\)](#), [as\\_string\\_formula\(\)](#), [construct\\_arima\(\)](#), [construct\\_gp\(\)](#), [construct\\_re\(\)](#), [construct\\_rw\(\)](#), [enw\\_formula\(\)](#), [enw\\_manual\\_formula\(\)](#), [gp\(\)](#), [gp\\_terms\(\)](#), [ma\(\)](#), [parse\\_formula\(\)](#), [re\(\)](#), [remove\\_arima\\_terms\(\)](#), [remove\\_gp\\_terms\(\)](#), [remove\\_rw\\_terms\(\)](#), [rw\(\)](#), [rw\\_terms\(\)](#), [split\\_formula\\_to\\_terms\(\)](#)

### Examples

```
ar(time)
ar(time, location, p = 2)
```

---

|       |                                                                          |
|-------|--------------------------------------------------------------------------|
| arima | <i>Adds an ARIMA(<math>p, d, q</math>) latent residual to the model.</i> |
|-------|--------------------------------------------------------------------------|

---

### Description

A call to `arima()` can be used in the formula argument of model construction functions in the `epinowcast` package such as [enw\\_formula\(\)](#). It declares an ARIMA( $p, d, q$ ) latent series indexed by time (and optionally a grouping variable `by`) whose value at each observation is added to the linear predictor. As with [rw\(\)](#), arguments are not evaluated; they are passed by name for use in model construction. Setting  $p = d = q = 0$  is not allowed; use [rw\(\)](#) (equivalent to `arima(time, d = 1)`) for a random walk.

**Usage**

```
arima(time, by, p = 1, d = 0, q = 0)
```

**Arguments**

|      |                                                                                                                                              |
|------|----------------------------------------------------------------------------------------------------------------------------------------------|
| time | Defines the time index of the ARIMA process.                                                                                                 |
| by   | Optional grouping variable. If supplied, an independent ARIMA series is fitted for each level of by. Currently limited to a single variable. |
| p    | Non-negative integer. Order of the autoregressive part. Defaults to 1.                                                                       |
| d    | Non-negative integer. Order of differencing (d = 1 gives an integrated series, equivalent to rw() when p = q = 0). Defaults to 0.            |
| q    | Non-negative integer. Order of the moving-average part. Defaults to 0.                                                                       |

**Value**

A list of class `enw_arima_term` describing the ARIMA term, interpretable by `construct_arima()`. Each group draws an independent shock series; phi, theta, and sigma are shared across groups (per-group parameters are a planned extension).

**See Also**

Functions used to help convert formulas into model designs `ar()`, `arima_terms()`, `arma()`, `as_string_formula()`, `construct_arima()`, `construct_gp()`, `construct_re()`, `construct_rw()`, `enw_formula()`, `enw_manual_formula()`, `gp()`, `gp_terms()`, `ma()`, `parse_formula()`, `re()`, `remove_arima_terms()`, `remove_gp_terms()`, `remove_rw_terms()`, `rw()`, `rw_terms()`, `split_formula_to_terms()`

**Examples**

```
arima(time)
arima(time, location)
arima(time, location, p = 2, d = 1, q = 1)
```

---

arima\_terms

*Finds ARIMA terms in a formula object*


---

**Description**

This function extracts ARIMA terms from a formula so that they can be processed on their own. Matches all four user-facing helpers that produce an `enw_arima_term`: `arima()`, plus the convenience aliases `ar()`, `ma()`, and `arma()`.

**Usage**

```
arima_terms(formula)
```

**Arguments**

**formula** A model formula that may use standard fixed effects, random effects using [lme4](#) syntax (see [re\(\)](#)), and random walks defined using the [rw\(\)](#) helper function. See the Details section below for a comprehensive explanation of the supported syntax.

**Value**

A character vector containing the ARIMA terms identified in the supplied formula.

**See Also**

Functions used to help convert formulas into model designs [ar\(\)](#), [arima\(\)](#), [arma\(\)](#), [as\\_string\\_formula\(\)](#), [construct\\_arima\(\)](#), [construct\\_gp\(\)](#), [construct\\_re\(\)](#), [construct\\_rw\(\)](#), [enw\\_formula\(\)](#), [enw\\_manual\\_formula\(\)](#), [gp\(\)](#), [gp\\_terms\(\)](#), [ma\(\)](#), [parse\\_formula\(\)](#), [re\(\)](#), [remove\\_arima\\_terms\(\)](#), [remove\\_gp\\_terms\(\)](#), [remove\\_rw\\_terms\(\)](#), [rw\(\)](#), [rw\\_terms\(\)](#), [split\\_formula\\_to\\_terms\(\)](#)

**Examples**

```
epinowcast:::arima_terms(~ 1 + age_group + arima(week))
epinowcast:::arima_terms(~ 1 + ar(week, p = 2))
epinowcast:::arima_terms(~ 1 + arma(week, location, p = 1, q = 1))
```

---

 arma

*ARMA alias for [arima\(\)](#)*


---

**Description**

Thin wrapper around [arima\(\)](#) that fixes  $d = 0$ . Equivalent to `arima(time, by, p = p, d = 0, q = q)`. For an integrated (random-walk) series use [rw\(\)](#) or `arima(time, by, p = 0, d = 1, q = 0)` directly.

**Usage**

```
arma(time, by, p = 1, q = 1)
```

**Arguments**

**time** Time variable for the latent series; numeric.

**by** Optional grouping variable. Each group draws an independent shock series; AR/MA parameters and the latent standard deviation are shared across groups.

**p** Autoregressive order. Defaults to 1.

**q** Moving-average order. Defaults to 1.

**Value**

An `enw_arima_term` interpretable by [construct\\_arima\(\)](#).

**See Also**

Functions used to help convert formulas into model designs [ar\(\)](#), [arima\(\)](#), [arima\\_terms\(\)](#), [as\\_string\\_formula\(\)](#), [construct\\_arima\(\)](#), [construct\\_gp\(\)](#), [construct\\_re\(\)](#), [construct\\_rw\(\)](#), [enw\\_formula\(\)](#), [enw\\_manual\\_formula\(\)](#), [gp\(\)](#), [gp\\_terms\(\)](#), [ma\(\)](#), [parse\\_formula\(\)](#), [re\(\)](#), [remove\\_arima\\_terms\(\)](#), [remove\\_gp\\_terms\(\)](#), [remove\\_rw\\_terms\(\)](#), [rw\(\)](#), [rw\\_terms\(\)](#), [split\\_formula\\_to\\_terms\(\)](#)

**Examples**

```
arma(time)
arma(time, location, p = 1, q = 1)
```

---

```
as_forecast_sample.epinowcast
```

*Convert an epinowcast object to a forecast\_sample object*

---

**Description**

This function is used to convert an epinowcast as returned by [epinowcast\(\)](#) object to a forecast\_sample object which can be used for scoring using the [scoringutils](#) package.

**Usage**

```
## S3 method for class 'epinowcast'
as_forecast_sample(data, latest_obs, ...)
```

**Arguments**

|            |                                                                                             |
|------------|---------------------------------------------------------------------------------------------|
| data       | An epinowcast nowcast object as returned by <a href="#">epinowcast()</a> .                  |
| latest_obs | Latest observations to use for the true values must contain confirm and observed variables. |
| ...        | Additional arguments passed to <a href="#">scoringutils::as_forecast_sample()</a>           |

**Value**

A forecast\_sample object as returned by [scoringutils::as\\_forecast\\_sample\(\)](#)

**Examples**

```
library(scoringutils)

nowcast <- enw_example("nowcast")
latest_obs <- enw_example("observations")
as_forecast_sample(nowcast, latest_obs)
```

---

|                   |                                     |
|-------------------|-------------------------------------|
| as_string_formula | <i>Converts formulas to strings</i> |
|-------------------|-------------------------------------|

---

**Description**

Converts formulas to strings

**Usage**

```
as_string_formula(formula)
```

**Arguments**

|         |                                                                                                                                                                                                                                                                                                    |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| formula | A model formula that may use standard fixed effects, random effects using <a href="#">lme4</a> syntax (see <a href="#">re()</a> ), and random walks defined using the <a href="#">rw()</a> helper function. See the Details section below for a comprehensive explanation of the supported syntax. |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Value**

A character string of the supplied formula

**See Also**

Functions used to help convert formulas into model designs [ar\(\)](#), [arima\(\)](#), [arima\\_terms\(\)](#), [arma\(\)](#), [construct\\_arima\(\)](#), [construct\\_gp\(\)](#), [construct\\_re\(\)](#), [construct\\_rw\(\)](#), [enw\\_formula\(\)](#), [enw\\_manual\\_formula\(\)](#), [gp\(\)](#), [gp\\_terms\(\)](#), [ma\(\)](#), [parse\\_formula\(\)](#), [re\(\)](#), [remove\\_arima\\_terms\(\)](#), [remove\\_gp\\_terms\(\)](#), [remove\\_rw\\_terms\(\)](#), [rw\(\)](#), [rw\\_terms\(\)](#), [split\\_formula\\_to\\_terms\(\)](#)

**Examples**

```
epinowcast:::as_string_formula(~ 1 + age_group)
```

---

|               |                                      |
|---------------|--------------------------------------|
| build_ord_obs | <i>Build the ord_obs data.table.</i> |
|---------------|--------------------------------------|

---

**Description**

Build the ord\_obs data.table.

**Usage**

```
build_ord_obs(obs, max_delay, internal_timestep, timestep, nowcast = NULL)
```

**Arguments**

|                   |                                                                                                                                                   |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| obs               | Observations as pulled from <code>nowcast\$latest[[1]]</code> .                                                                                   |
| max_delay         | Whole number representing the maximum delay in units of the timestep.                                                                             |
| internal_timestep | The internal timestep in days.                                                                                                                    |
| timestep          | The timestep to be used. This can be a string ("day", "week", "month") or a numeric whole number representing the number of days.                 |
| nowcast           | If getting posterior samples, a data frame with a <code>draws</code> column to get the draws from, as pulled from the fit attribute of a nowcast. |

**Value**

A `data.table`.

**See Also**

Functions used for postprocessing of model fits [.check\\_primarycensored\(\)](#), [.delay\\_draw\\_columns\(\)](#), [.discretise\\_parametric\\_pmf\(\)](#), [enw\\_add\\_latest\\_obs\\_to\\_nowcast\(\)](#), [enw\\_nowcast\\_samples\(\)](#), [enw\\_nowcast\\_summary\(\)](#), [enw\\_posterior\(\)](#), [enw\\_posterior\\_delay\(\)](#), [enw\\_pp\\_summary\(\)](#), [enw\\_quantiles\\_to\\_long\(\)](#), [enw\\_summarise\\_samples\(\)](#), [subset\\_obs\(\)](#)

---

check\_design\_matrix\_sparsity

*Check design matrix sparsity*

---

**Description**

This function checks the sparsity of a design matrix and provides a recommendation if the matrix is considered sparse.

**Usage**

```
check_design_matrix_sparsity(
  matrix,
  sparsity_threshold = 0.9,
  min_matrix_size = 50,
  name = "checked"
)
```

**Arguments**

|                    |                                                                                                           |
|--------------------|-----------------------------------------------------------------------------------------------------------|
| matrix             | A numeric matrix to be checked for sparsity.                                                              |
| sparsity_threshold | A numeric value between 0 and 1 indicating the threshold for considering a matrix sparse. Default is 0.9. |

`min_matrix_size` An integer indicating the minimum size of the matrix for which to perform the sparsity check. Default is 50.

`name` A character string specifying the name of the design matrix. Default is "checked".

### Value

This function is used for its side effect of providing an informational message if the matrix is sparse. It returns NULL invisibly.

### See Also

Functions used for checking inputs [check\\_group\(\)](#), [check\\_group\\_date\\_unique\(\)](#), [check\\_max\\_delay\(\)](#), [check\\_module\(\)](#), [check\\_modules\\_compatible\(\)](#), [check\\_numeric\\_timestep\(\)](#), [check\\_observation\\_indicator\(\)](#), [check\\_quantiles\(\)](#), [check\\_timestep\(\)](#), [check\\_timestep\\_by\\_date\(\)](#), [check\\_timestep\\_by\\_group\(\)](#)

---

|             |                                                           |
|-------------|-----------------------------------------------------------|
| check_group | <i>Check observations for reserved grouping variables</i> |
|-------------|-----------------------------------------------------------|

---

### Description

Check observations for reserved grouping variables

### Usage

```
check_group(obs)
```

### Arguments

`obs` An object that will be `coerce_dtd` in place, that does not contain `.group`, `.old_group`, or `.new_group`. These are reserved names.

### Value

The `obs` object, which will be modifiable in place.

### See Also

Functions used for checking inputs [check\\_design\\_matrix\\_sparsity\(\)](#), [check\\_group\\_date\\_unique\(\)](#), [check\\_max\\_delay\(\)](#), [check\\_module\(\)](#), [check\\_modules\\_compatible\(\)](#), [check\\_numeric\\_timestep\(\)](#), [check\\_observation\\_indicator\(\)](#), [check\\_quantiles\(\)](#), [check\\_timestep\(\)](#), [check\\_timestep\\_by\\_date\(\)](#), [check\\_timestep\\_by\\_group\(\)](#)

---

 check\_group\_date\_unique

*Check observations for uniqueness of grouping variables with respect to reference\_date and report\_date*

---

## Description

This function checks that the input data is stratified by reference\_date, report\_date, and .group. It does this by counting the number of observations for each combination of these variables, and throwing a warning if any combination has more than one observation.

## Usage

```
check_group_date_unique(obs)
```

## Arguments

|     |                                                                                                            |
|-----|------------------------------------------------------------------------------------------------------------|
| obs | An object that will be coerce_dtd in place, that contains .group, reference_date, and report_date columns. |
|-----|------------------------------------------------------------------------------------------------------------|

## See Also

Functions used for checking inputs [check\\_design\\_matrix\\_sparsity\(\)](#), [check\\_group\(\)](#), [check\\_max\\_delay\(\)](#), [check\\_module\(\)](#), [check\\_modules\\_compatible\(\)](#), [check\\_numeric\\_timestep\(\)](#), [check\\_observation\\_indicator\(\)](#), [check\\_quantiles\(\)](#), [check\\_timestep\(\)](#), [check\\_timestep\\_by\\_date\(\)](#), [check\\_timestep\\_by\\_group\(\)](#)

---

 check\_max\_delay

*Check appropriateness of maximum delay*

---

## Description

Check if maximum delay specified by the user is long enough and raise potential warnings. This is achieved by computing the share of reference dates where the cumulative case count is below some aspired coverage.

## Usage

```
check_max_delay(
  data,
  max_delay = data$max_delay,
  cum_coverage = 0.8,
  maxdelay_quantile_outlier = 0.97,
  warn = TRUE,
  warn_internal = FALSE
)
```

## Arguments

|                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| data                      | Output from <code>enw_preprocess_data()</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| max_delay                 | <p>The maximum delay to model in the delay distribution, specified in units of the timestep (e.g., if <code>timestep = "week"</code>, then <code>max_delay = 3</code> means 3 weeks). If not specified the maximum observed delay is assumed to be the true maximum delay in the model. Otherwise, an integer greater than or equal to 1 can be specified. Observations with delays larger than the maximum delay will be dropped. If the specified maximum delay is too short, nowcasts can be biased as important parts of the true delay distribution are cut off. At the same time, computational cost scales non-linearly with this setting, so you want the maximum delay to be as long as necessary, but not much longer.</p> <p>Steps to take to determine the maximum delay:</p> <ul style="list-style-type: none"> <li>• Consider what is realistic and relevant for your application.</li> <li>• Check the proportion of observations reported (<code>prop_reported</code>) by delay in the <code>new_confirm</code> output of <code>enw_preprocess_obs</code>.</li> <li>• Use <code>check_max_delay()</code> to check the coverage of a candidate <code>max_delay</code>.</li> <li>• If in doubt, check if increasing the maximum delay noticeably changes the delay distribution or nowcasts as estimated by <code>epinowcast</code>. If it does, your maximum delay may still be too short.</li> </ul> <p>Note that delays are zero indexed and so include the reference date and <code>max_delay - 1</code> other intervals (i.e. a <code>max_delay</code> of 1 corresponds to no delay).</p> |
| cum_coverage              | The aspired percentage of cases that the maximum delay should cover. Defaults to 0.8 (80%).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| maxdelay_quantile_outlier | Only reference dates sufficiently far in the past, determined based on the maximum observed delay, are included (see details). Instead of the overall maximum observed delay, a quantile of the maximum observed delay over all reference dates is used. This is more robust against outliers. Defaults to 0.97 (97%).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| warn                      | Should a warning be issued if the cumulative case count is below <code>cum_coverage</code> for the majority of reference dates?                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| warn_internal             | Should only be TRUE if this function is called internally by another <code>epinowcast</code> function. Then, warnings are adjusted to avoid confusing the user.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

## Details

When data is very sparse (e.g., predominantly zero counts), the function may not be able to compute meaningful coverage statistics. In such cases, a warning is issued and the function treats the data as having no coverage issues. This typically occurs when groups have very few non-zero observations or when the specified `max_delay` is too large relative to available data.

The coverage is with respect to the maximum observed case count for the corresponding reference date. As the maximum observed case count is likely smaller than the true overall case count for not yet fully observed reference dates (due to right truncation), only reference dates that are more than the maximum observed delay ago are included. Still, because we can only use the maximum observed delay, not the unknown true maximum delay, the computed coverage values should be interpreted with care, as they are only proxies for the true coverage.

**Value**

A data.frame with the share of reference dates where the cumulative case count is below cum\_coverage, stratified by group.

**See Also**

Functions used for checking inputs [check\\_design\\_matrix\\_sparsity\(\)](#), [check\\_group\(\)](#), [check\\_group\\_date\\_unique\(\)](#), [check\\_module\(\)](#), [check\\_modules\\_compatible\(\)](#), [check\\_numeric\\_timestep\(\)](#), [check\\_observation\\_indicator\(\)](#), [check\\_quantiles\(\)](#), [check\\_timestep\(\)](#), [check\\_timestep\\_by\\_date\(\)](#), [check\\_timestep\\_by\\_group\(\)](#)

**Examples**

```
pobs <- enw_example(type = "preprocessed_observations")
check_max_delay(pobs, max_delay = 20, cum_coverage = 0.8)
```

---

check\_module

---

*Check a model module contains the required components*


---

**Description**

Check a model module contains the required components

**Usage**

```
check_module(module)
```

**Arguments**

module                    A model module. For example [enw\\_expectation\(\)](#).

**See Also**

Functions used for checking inputs [check\\_design\\_matrix\\_sparsity\(\)](#), [check\\_group\(\)](#), [check\\_group\\_date\\_unique\(\)](#), [check\\_max\\_delay\(\)](#), [check\\_modules\\_compatible\(\)](#), [check\\_numeric\\_timestep\(\)](#), [check\\_observation\\_indicator\(\)](#), [check\\_quantiles\(\)](#), [check\\_timestep\(\)](#), [check\\_timestep\\_by\\_date\(\)](#), [check\\_timestep\\_by\\_group\(\)](#)

---

check\_modules\_compatible

*Check that model modules have compatible specifications*

---

### Description

Check that model modules have compatible specifications

### Usage

```
check_modules_compatible(modules)
```

### Arguments

modules            A list of model modules.

### See Also

Functions used for checking inputs [check\\_design\\_matrix\\_sparsity\(\)](#), [check\\_group\(\)](#), [check\\_group\\_date\\_unique\(\)](#), [check\\_max\\_delay\(\)](#), [check\\_module\(\)](#), [check\\_numeric\\_timestep\(\)](#), [check\\_observation\\_indicator\(\)](#), [check\\_quantiles\(\)](#), [check\\_timestep\(\)](#), [check\\_timestep\\_by\\_date\(\)](#), [check\\_timestep\\_by\\_group\(\)](#)

---

check\_numeric\_timestep

*Check Numeric Timestep*

---

### Description

This function verifies if the difference in numeric dates in the provided observations corresponds to the provided timestep.

### Usage

```
check_numeric_timestep(dates, date_var, timestep, exact = TRUE)
```

### Arguments

dates            Vector of Date class representing dates.

date\_var        The variable in obs representing dates.

timestep        Numeric timestep for date difference.

exact           Logical, if TRUE, checks if all differences exactly match the timestep. If FALSE, checks if the sum of the differences modulo the timestep equals zero. Default is TRUE.

**Value**

This function is used for its side effect of stopping if the check fails. If the check passes, the function returns invisibly.

**See Also**

Functions used for checking inputs [check\\_design\\_matrix\\_sparsity\(\)](#), [check\\_group\(\)](#), [check\\_group\\_date\\_unique\(\)](#), [check\\_max\\_delay\(\)](#), [check\\_module\(\)](#), [check\\_modules\\_compatible\(\)](#), [check\\_observation\\_indicator\(\)](#), [check\\_quantiles\(\)](#), [check\\_timestep\(\)](#), [check\\_timestep\\_by\\_date\(\)](#), [check\\_timestep\\_by\\_group\(\)](#)

---

check\_observation\_indicator

*Check observation indicator*

---

**Description**

This function verifies if the `observation_indicator` within the provided `new_confirm` observations is logical. The check is performed to ensure that the `observation_indicator` is of the correct type.

**Usage**

```
check_observation_indicator(new_confirm, observation_indicator = NULL)
```

**Arguments**

`new_confirm` A data frame containing the observations to be checked.

`observation_indicator` A character string specifying the column name in `new_confirm` that represents the observation indicator. This column should be of logical type. If `NULL`, no check is performed.

**Value**

This function is used for its side effect of checking the observation indicator in `new_confirm`. If the check passes, the function returns invisibly. Otherwise, it stops and returns an error message.

**See Also**

Functions used for checking inputs [check\\_design\\_matrix\\_sparsity\(\)](#), [check\\_group\(\)](#), [check\\_group\\_date\\_unique\(\)](#), [check\\_max\\_delay\(\)](#), [check\\_module\(\)](#), [check\\_modules\\_compatible\(\)](#), [check\\_numeric\\_timestep\(\)](#), [check\\_quantiles\(\)](#), [check\\_timestep\(\)](#), [check\\_timestep\\_by\\_date\(\)](#), [check\\_timestep\\_by\\_group\(\)](#)

---

|                 |                                             |
|-----------------|---------------------------------------------|
| check_quantiles | <i>Check required quantiles are present</i> |
|-----------------|---------------------------------------------|

---

### Description

Check required quantiles are present

### Usage

```
check_quantiles(posterior, req_probs = c(0.5, 0.95, 0.2, 0.8))
```

### Arguments

|           |                                                                                                                                           |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------|
| posterior | A <code>data.table</code> that will be <code>coerce_dt()</code> d in place; must contain quantiles identified using the q5 naming scheme. |
| req_probs | A numeric vector of required probabilities. Default: <code>c(0.5, 0.95, 0.2, 0.8)</code> .                                                |

### See Also

Functions used for checking inputs `check_design_matrix_sparsity()`, `check_group()`, `check_group_date_unique()`, `check_max_delay()`, `check_module()`, `check_modules_compatible()`, `check_numeric_timestep()`, `check_observation_indicator()`, `check_timestep()`, `check_timestep_by_date()`, `check_timestep_by_group()`

---

|                |                       |
|----------------|-----------------------|
| check_timestep | <i>Check timestep</i> |
|----------------|-----------------------|

---

### Description

This function verifies if the difference in dates in the provided observations corresponds to the provided timestep. If the `exact` argument is set to `TRUE`, the function checks if all differences exactly match the timestep; otherwise, it checks if the sum of the differences modulo the timestep equals zero. If the check fails, the function stops and returns an error message.

### Usage

```
check_timestep(
  obs,
  date_var,
  timestep = "day",
  exact = TRUE,
  check_nrow = TRUE
)
```

**Arguments**

|            |                                                                                                                                                                                                                              |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| obs        | Any of the types supported by <code>data.table::as.data.table()</code> .                                                                                                                                                     |
| date_var   | The variable in obs representing dates.                                                                                                                                                                                      |
| timestep   | The timestep to used. This can be a string ("day", "week") or a numeric whole number representing the number of days. Note that "month" is not currently supported in user-facing functions and will throw an error if used. |
| exact      | Logical, if TRUE, checks if all differences exactly match the timestep. If FALSE, checks if the sum of the differences modulo the timestep equals zero. Default is TRUE.                                                     |
| check_nrow | Logical, if TRUE, checks if there are at least two observations. Default is TRUE. If FALSE, the function returns invisibly if there is only one observation.                                                                 |

**Value**

This function is used for its side effect of stopping if the check fails. If the check passes, the function returns invisibly.

**See Also**

Functions used for checking inputs `check_design_matrix_sparsity()`, `check_group()`, `check_group_date_unique()`, `check_max_delay()`, `check_module()`, `check_modules_compatible()`, `check_numeric_timestep()`, `check_observation_indicator()`, `check_quantiles()`, `check_timestep_by_date()`, `check_timestep_by_group()`

---

check\_timestep\_by\_date

*Check timestep by date*

---

**Description**

This function verifies if the difference in dates within each date in the provided observations corresponds to the provided timestep. This check is performed for both `report_date` and `reference_date` and for each group in obs.

**Usage**

```
check_timestep_by_date(obs, timestep = "day", exact = TRUE)
```

**Arguments**

|          |                                                                                                                                                                                                                              |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| obs      | Any of the types supported by <code>data.table::as.data.table()</code> .                                                                                                                                                     |
| timestep | The timestep to used. This can be a string ("day", "week") or a numeric whole number representing the number of days. Note that "month" is not currently supported in user-facing functions and will throw an error if used. |
| exact    | Logical, if TRUE, checks if all differences exactly match the timestep. If FALSE, checks if the sum of the differences modulo the timestep equals zero. Default is TRUE.                                                     |

**Value**

This function is used for its side effect of checking the timestep by date in obs. If the check passes for all dates, the function returns invisibly. Otherwise, it stops and returns an error message.

**See Also**

Functions used for checking inputs `check_design_matrix_sparsity()`, `check_group()`, `check_group_date_unique()`, `check_max_delay()`, `check_module()`, `check_modules_compatible()`, `check_numeric_timestep()`, `check_observation_indicator()`, `check_quantiles()`, `check_timestep()`, `check_timestep_by_group()`

---

check\_timestep\_by\_group

*Check timestep by group*

---

**Description**

This function verifies if the difference in dates within each group in the provided observations corresponds to the provided timestep. This check is performed for the specified `date_var` and for each group in obs.

**Usage**

```
check_timestep_by_group(obs, date_var, timestep = "day", exact = TRUE)
```

**Arguments**

|                       |                                                                                                                                                                                                                              |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>obs</code>      | Any of the types supported by <code>data.table::as.data.table()</code> .                                                                                                                                                     |
| <code>date_var</code> | The variable in obs representing dates.                                                                                                                                                                                      |
| <code>timestep</code> | The timestep to used. This can be a string ("day", "week") or a numeric whole number representing the number of days. Note that "month" is not currently supported in user-facing functions and will throw an error if used. |
| <code>exact</code>    | Logical, if TRUE, checks if all differences exactly match the timestep. If FALSE, checks if the sum of the differences modulo the timestep equals zero. Default is TRUE.                                                     |

**Value**

This function is used for its side effect of checking the timestep by group in obs. If the check passes for all groups, the function returns invisibly. Otherwise, it stops and returns an error message.

**See Also**

Functions used for checking inputs `check_design_matrix_sparsity()`, `check_group()`, `check_group_date_unique()`, `check_max_delay()`, `check_module()`, `check_modules_compatible()`, `check_numeric_timestep()`, `check_observation_indicator()`, `check_quantiles()`, `check_timestep()`, `check_timestep_by_date()`

---

coerce\_date*Coerce Dates*

---

## Description

Provides consistent coercion of inputs to [IDate](#) with error handling

## Usage

```
coerce_date(dates = NULL)
```

## Arguments

|       |                                                                                                                           |
|-------|---------------------------------------------------------------------------------------------------------------------------|
| dates | A vector-like input, which the function attempts to coerce via <a href="#">data.table::as.IDate()</a> . Defaults to NULL. |
|-------|---------------------------------------------------------------------------------------------------------------------------|

## Details

If any of the elements of dates cannot be coerced, this function will result in an error, indicating all indices which cannot be coerced to [IDate](#).

Internal methods of [epinowcast](#) assume dates are represented as [IDate](#).

## Value

An [IDate](#) vector.

## See Also

Utility functions [coerce\\_dt\(\)](#), [date\\_to\\_numeric\\_modulus\(\)](#), [enw\\_get\\_data\(\)](#), [enw\\_rolling\\_sum\(\)](#), [get\\_internal\\_timestep\(\)](#), [is.Date\(\)](#), [stan\\_fns\\_as\\_string\(\)](#)

## Examples

```
# works
coerce_date(c("2020-05-28", "2020-05-29"))
# does not, indicates index 2 is problem
tryCatch(
  coerce_date(c("2020-05-28", "2020-o5-29")),
  error = function(e) {
    print(e)
  }
)
```

---

|           |                           |
|-----------|---------------------------|
| coerce_dt | <i>Coerce data.tables</i> |
|-----------|---------------------------|

---

## Description

Provides consistent coercion of inputs to [data.table](#) with error handling, column checking, and optional selection.

## Usage

```
coerce_dt(
  data,
  select = NULL,
  required_cols = select,
  forbidden_cols = NULL,
  group = FALSE,
  dates = FALSE,
  copy = TRUE,
  msg_required = "The following columns are required: ",
  msg_forbidden = "The following columns are forbidden: "
)
```

## Arguments

|                |                                                                                                                                                    |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| data           | Any of the types supported by <a href="#">data.table::as.data.table()</a>                                                                          |
| select         | An optional character vector of columns to return; <i>unchecked</i> n.b. it is an error to include ".group"; use group argument for that           |
| required_cols  | An optional character vector of required columns                                                                                                   |
| forbidden_cols | An optional character vector of forbidden columns                                                                                                  |
| group          | A logical; ensure the presence of a .group column?                                                                                                 |
| dates          | A logical; ensure the presence of report_date and reference_date? If TRUE (default), those columns will be coerced with <a href="#">as.IDate</a> . |
| copy           | A logical; if TRUE (default), a new data.table is returned                                                                                         |
| msg_required   | A character string; for required_cols-related error message                                                                                        |
| msg_forbidden  | A character string; for forbidden_cols-related error message                                                                                       |

## Details

This function provides a single-point function for getting a "local" version of data provided by the user, in the internally used data.table format. It also enables selectively copying versus not, as well as checking for the presence and/or absence of various columns.

While it is intended to address garbage in from the *user*, it does not generally attempt to address garbage in from the *developer* - e.g. if asking for overlapping required and forbidden columns (though that will lead to an always-error condition).

When `dates = TRUE`, this function ensures that `report_date` and `reference_date` columns are coerced to `IDate` class with integer storage mode. This is necessary because some operations (such as `dplyr::filter()`) can convert `IDate` columns to double storage mode whilst preserving the class, which violates `data.table`'s requirements and causes errors in subsequent date arithmetic operations.

### Value

A `data.table`; the returned object will be a copy, unless `copy = FALSE`, in which case modifications are made in-place

### See Also

Utility functions `coerce_date()`, `date_to_numeric_modulus()`, `enw_get_data()`, `enw_rolling_sum()`, `get_internal_timestep()`, `is.Date()`, `stan_fns_as_string()`

---

|                 |                                       |
|-----------------|---------------------------------------|
| construct_arima | <i>Constructs ARIMA term metadata</i> |
|-----------------|---------------------------------------|

---

### Description

Takes an ARIMA term as defined by `arma()` and returns the metadata required to wire the term into a Stan model. Unlike `construct_rw()`, this does not modify the data or produce design matrix columns; ARIMA latent residuals enter the linear predictor through a parameter-dependent kernel applied to unit-normal shocks (see `inst/stan/functions/arma_kernel.stan`).

### Usage

```
construct_arima(arma, data)
```

### Arguments

|                   |                                                                                                                                      |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <code>arma</code> | An ARIMA term as defined by <code>arma()</code> .                                                                                    |
| <code>data</code> | A <code>data.frame</code> of observations used to define the ARIMA term. Must contain the time and (if specified) grouping variable. |

### Value

A list with the following elements:

- `time`, `by`, `p`, `d`, `q`: passed through from the `arma()` term.
- `T`: number of distinct time points in the series.
- `G`: number of groups (1 if `by` is unspecified).
- `time_idx`: integer vector mapping each row of data to a time index in `1:T`.
- `group_idx`: integer vector mapping each row of data to a group index in `1:G`.
- `time_vals`, `group_levels`: lookup vectors so the indices can be inverted.
- `name`: a label for the term, suitable as a parameter prefix.

**See Also**

Functions used to help convert formulas into model designs `ar()`, `arma()`, `arma_terms()`, `arma()`, `as_string_formula()`, `construct_gp()`, `construct_re()`, `construct_rw()`, `enw_formula()`, `enw_manual_formula()`, `gp()`, `gp_terms()`, `ma()`, `parse_formula()`, `re()`, `remove_arma_terms()`, `remove_gp_terms()`, `remove_rw_terms()`, `rw()`, `rw_terms()`, `split_formula_to_terms()`

**Examples**

```
data <- enw_example("preproc")$metareference[[1]]
epinowcast::construct_arma(arma(week), data)
epinowcast::construct_arma(
  arma(week, day_of_week, p = 2, d = 1), data
)
```

---

construct\_gp

Constructs Gaussian process term metadata

---

**Description**

Takes a Gaussian process term as defined by `gp()` and returns the metadata required to wire the term into a Stan model. Like `construct_arma()`, this does not modify the data or produce design matrix columns; the Gaussian process enters the linear predictor through a Hilbert-space reduced-rank approximation (see `inst/stan/functions/gaussian_process.stan`).

**Usage**

```
construct_gp(gp, data)
```

**Arguments**

|                   |                                                                                                                   |
|-------------------|-------------------------------------------------------------------------------------------------------------------|
| <code>gp</code>   | A Gaussian process term as defined by <code>gp()</code> .                                                         |
| <code>data</code> | A data.frame of observations used to define the term. Must contain the time and (if specified) grouping variable. |

**Value**

A list with the following elements:

- `time`, `by`, `kernel`, `gp_type`, `nu`, `d`, `basis_prop`, `boundary_scale`: passed through from the `gp()` term.
- `T`: number of distinct time points in the integrated series.
- `G`: number of groups (1 if `by` is unspecified).
- `M`: number of basis functions, `ceiling(basis_prop * (T - d))`.
- `PHI`: the  $(T - d) \times M$  basis matrix. For  $d \geq 1$  the basis is built on the  $T - d$  free values that are integrated  $d$  times in Stan; the first  $d$  values of the realisation are anchored to zero.
- `time_idx`, `group_idx`: per-observation lookup indices.
- `time_vals`, `group_levels`: lookup vectors so the indices can be inverted.
- `name`: a label for the term, suitable as a parameter prefix.

See Also

Functions used to help convert formulas into model designs `ar()`, `arima()`, `arima_terms()`, `arma()`, `as_string_formula()`, `construct_arima()`, `construct_re()`, `construct_rw()`, `enw_formula()`, `enw_manual_formula()`, `gp()`, `gp_terms()`, `ma()`, `parse_formula()`, `re()`, `remove_arima_terms()`, `remove_gp_terms()`, `remove_rw_terms()`, `rw()`, `rw_terms()`, `split_formula_to_terms()`

Examples

```
data <- enw_example("preproc")$metareference[[1]]
epinowcast::construct_gp(gp(week), data)
epinowcast::construct_gp(gp(week, day_of_week, kernel = "se"), data)
```

---

|              |                                       |
|--------------|---------------------------------------|
| construct_re | <i>Constructs random effect terms</i> |
|--------------|---------------------------------------|

---

Description

Constructs random effect terms

Usage

```
construct_re(re, data)
```

Arguments

- re                    A random effect as defined using `re()` which itself takes random effects specified in a model formula using the `lme4` syntax.
- data                 A `data.frame` of observations used to define the random effects. Must contain the variables specified in the `re()` term.

Value

A list containing the transformed data ("data"), fixed effects terms ("terms") and a `data.frame` specifying the random effect structure between these terms (effects). Note that if the specified random effect was not a factor it will have been converted into one.

See Also

Functions used to help convert formulas into model designs `ar()`, `arima()`, `arima_terms()`, `arma()`, `as_string_formula()`, `construct_arima()`, `construct_gp()`, `construct_rw()`, `enw_formula()`, `enw_manual_formula()`, `gp()`, `gp_terms()`, `ma()`, `parse_formula()`, `re()`, `remove_arima_terms()`, `remove_gp_terms()`, `remove_rw_terms()`, `rw()`, `rw_terms()`, `split_formula_to_terms()`

## Examples

```
# Simple examples
form <- epinowcast:::parse_formula(~ 1 + (1 | day_of_week))
data <- enw_example("prepr")$metareference[[1]]
random_effect <- re(form$random[[1]])
epinowcast:::construct_re(random_effect, data)

# A more complex example
form <- epinowcast:::parse_formula(
  ~ 1 + disp + (1 + gear | cyl) + (0 + wt | am)
)
random_effect <- re(form$random[[1]])
epinowcast:::construct_re(random_effect, mtcars)

random_effect2 <- re(form$random[[2]])
epinowcast:::construct_re(random_effect2, mtcars)
```

---

|              |                                     |
|--------------|-------------------------------------|
| construct_rw | <i>Constructs random walk terms</i> |
|--------------|-------------------------------------|

---

## Description

This function takes random walks as defined by `rw()`, produces the required additional variables (denoted using a "c" prefix and constructed using `enw_add_cumulative_membership()`), and then returns the extended data.frame along with the new fixed effects and the random effect structure.

## Usage

```
construct_rw(rw, data)
```

## Arguments

|                   |                                                                                                                                                                 |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>rw</code>   | A random walk term as defined by <code>rw()</code> .                                                                                                            |
| <code>data</code> | A data.frame of observations used to define the random walk term. Must contain the time and grouping variables defined in the <code>rw()</code> term specified. |

## Value

A list containing the following:

- `data`: The input data.frame with the addition of the new variables required by the specified random walk. These are added using `enw_add_cumulative_membership()`. `-terms`: A character vector of new fixed effects terms to add to a model formula.
- `effects`: A data.frame describing the random effect structure of the new effects.

**See Also**

Functions used to help convert formulas into model designs `ar()`, `arima()`, `arima_terms()`, `arma()`, `as_string_formula()`, `construct_arima()`, `construct_gp()`, `construct_re()`, `enw_formula()`, `enw_manual_formula()`, `gp()`, `gp_terms()`, `ma()`, `parse_formula()`, `re()`, `remove_arima_terms()`, `remove_gp_terms()`, `remove_rw_terms()`, `rw()`, `rw_terms()`, `split_formula_to_terms()`

**Examples**

```
data <- enw_example("preproc")$metareference[[1]]

epinowcast:::construct_rw(rw(week), data)

epinowcast:::construct_rw(rw(week, day_of_week), data)
```

---

|                    |                                       |
|--------------------|---------------------------------------|
| convolution_matrix | <i>Construct a convolution matrix</i> |
|--------------------|---------------------------------------|

---

**Description**

This function allows the construction of convolution matrices which can be combined with a vector of primary events to produce a vector of secondary events for example in the form of a renewal equation or to simulate reporting delays. Time-varying delays are supported as well as distribution padding (to allow for use in renewal equation like approaches).

**Usage**

```
convolution_matrix(dist, t, include_partial = FALSE)
```

**Arguments**

|                              |                                                                                                         |
|------------------------------|---------------------------------------------------------------------------------------------------------|
| <code>dist</code>            | A vector of list of vectors describing the distribution to be convolved as a probability mass function. |
| <code>t</code>               | Integer value indicating the number of time steps to convolve over.                                     |
| <code>include_partial</code> | Logical, defaults to FALSE. If TRUE, the convolution include partially complete secondary events.       |

**Value**

A matrix with each column indicating a primary event and each row indicating a secondary event.

**See Also**

Helper functions for model modules `add_max_observed_delay()`, `add_pmfs()`, `delay_only_ltotal()`, `delay_only_total()`, `enw_dayofweek_structural_reporting()`, `enw_reference_by_report()`, `enw_reps_with_complete_refs()`, `enw_structural_reporting_metadata()`, `extract_obs_metadata()`, `extract_sparse_matrix()`, `latest_obs_as_matrix()`

**Examples**

```
# Simple convolution matrix with a static distribution
convolution_matrix(c(1, 2, 3), 10)
# Include partially reported convolutions
convolution_matrix(c(1, 2, 3), 10, include_partial = TRUE)
# Use a list of distributions
convolution_matrix(rep(list(c(1, 2, 3)), 10), 10)
# Use a time-varying list of distributions
convolution_matrix(c(rep(list(c(1, 2, 3)), 10), list(c(4, 5, 6))), 11)
```

---

date\_to\_numeric\_modulus

*Convert date column to numeric and calculate its modulus with given timestep.*

---

**Description**

This function processes a date column in a `data.table`, converting it to a numeric representation and then computing the modulus with the provided timestep.

**Usage**

```
date_to_numeric_modulus(dt, date_column, timestep)
```

**Arguments**

|             |                                                                    |
|-------------|--------------------------------------------------------------------|
| dt          | A <code>data.table</code> .                                        |
| date_column | A character string representing the name of the date column in dt. |
| timestep    | An integer representing the internal timestep.                     |

**Value**

A modified `data.table` with two new columns: one for the numeric representation of the date minus the minimum date and another for its modulus with the timestep.

**See Also**

Utility functions [coerce\\_date\(\)](#), [coerce\\_dt\(\)](#), [enw\\_get\\_data\(\)](#), [enw\\_rolling\\_sum\(\)](#), [get\\_internal\\_timestep\(\)](#), [is.Date\(\)](#), [stan\\_fns\\_as\\_string\(\)](#)

---

|                   |                                                                 |
|-------------------|-----------------------------------------------------------------|
| delay_only_ltotal | <i>Known per-reference-date totals for the delay-only model</i> |
|-------------------|-----------------------------------------------------------------|

---

**Description**

Builds the `dlo_ltotal` data entry for the delay-only model. The known totals are the latest available confirm per reference date and group, supplied to Stan on the log scale as a `g` by `t` matrix (cmdstanr's layout for `array[g] vector[t]`). The log total is only an offset on the expected cells and cancels in the multinomial likelihood, so reference dates with no observed cells (a non-positive or missing latest total, e.g. the most recent dates under an observation indicator) are floored to 1 (log 0); such dates contribute nothing to the likelihood.

**Usage**

```
delay_only_ltotal(data, delay_only)
```

**Arguments**

|                         |                                                                                                                               |
|-------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| <code>data</code>       | Output from <a href="#">enw_preprocess_data()</a> .                                                                           |
| <code>delay_only</code> | Logical; if FALSE an empty ( <code>g</code> by <code>0</code> ) matrix is returned so the model carries no delay-only totals. |

**Value**

A `g` by `t` matrix of log totals (or `g` by `0` when not in delay-only mode).

**See Also**

Helper functions for model modules [add\\_max\\_observed\\_delay\(\)](#), [add\\_pmfs\(\)](#), [convolution\\_matrix\(\)](#), [delay\\_only\\_total\(\)](#), [enw\\_dayofweek\\_structural\\_reporting\(\)](#), [enw\\_reference\\_by\\_report\(\)](#), [enw\\_reps\\_with\\_complete\\_refs\(\)](#), [enw\\_structural\\_reporting\\_metadata\(\)](#), [extract\\_obs\\_metadata\(\)](#), [extract\\_sparse\\_matrix\(\)](#), [latest\\_obs\\_as\\_matrix\(\)](#)

---

|                  |                                                                   |
|------------------|-------------------------------------------------------------------|
| delay_only_total | <i>Known integer totals per snapshot for the delay-only model</i> |
|------------------|-------------------------------------------------------------------|

---

**Description**

Builds the `dlo_total` data entry: the known integer total per snapshot (the latest cumulative confirm, i.e. the cutoff running total), ordered by group then reference date to match the snapshot order. These totals size the residual category when an observation indicator leaves some before-cutoff cells unobserved.

**Usage**

```
delay_only_total(data, delay_only)
```

**Arguments**

|            |                                                                                                              |
|------------|--------------------------------------------------------------------------------------------------------------|
| data       | Output from <a href="#">enw_preprocess_data()</a> .                                                          |
| delay_only | Logical; if FALSE an empty (g by $\emptyset$ ) matrix is returned so the model carries no delay-only totals. |

**Value**

An integer vector of length snapshots (or length 0 when not in delay-only mode).

**See Also**

Helper functions for model modules [add\\_max\\_observed\\_delay\(\)](#), [add\\_pmfs\(\)](#), [convolution\\_matrix\(\)](#), [delay\\_only\\_ltotal\(\)](#), [enw\\_dayofweek\\_structural\\_reporting\(\)](#), [enw\\_reference\\_by\\_report\(\)](#), [enw\\_reps\\_with\\_complete\\_refs\(\)](#), [enw\\_structural\\_reporting\\_metadata\(\)](#), [extract\\_obs\\_metadata\(\)](#), [extract\\_sparse\\_matrix\(\)](#), [latest\\_obs\\_as\\_matrix\(\)](#)

---

|                    |                                                                          |
|--------------------|--------------------------------------------------------------------------|
| enw_add_cumulative | <i>Calculate cumulative reported cases from incidence of new reports</i> |
|--------------------|--------------------------------------------------------------------------|

---

**Description**

Calculate cumulative reported cases from incidence of new reports

**Usage**

```
enw_add_cumulative(obs, by = NULL, copy = TRUE)
```

**Arguments**

|      |                                                                                                                                                                                                                      |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| obs  | A data.frame containing at least the following variables: reference_date (index date of interest), report_date (report date for observations), and new_confirm (incident observations by reference and report date). |
| by   | A character vector describing the stratification of observations. This defaults to no grouping. This should be used when modelling multiple time series in order to identify them for downstream modelling           |
| copy | Should obs be copied (default) or modified in place?                                                                                                                                                                 |

**Value**

The input data.frame with a new variable confirm.

**See Also**

Data converters [enw\\_add\\_incidence\(\)](#), [enw\\_aggregate\\_cumulative\(\)](#), [enw\\_incidence\\_to\\_linelist\(\)](#), [enw\\_linelist\\_to\\_incidence\(\)](#)

**Examples**

```
# Default reconstruct incidence
dt <- germany_covid19_hosp[location == "DE"][age_group == "00+"]
dt <- enw_filter_reference_dates_by_report_start(dt)
dt <- enw_add_incidence(dt)
dt <- dt[, confirm := NULL]
enw_add_cumulative(dt)

# Make use of maximum reported to calculate empirical daily reporting
enw_add_cumulative(dt)
```

---

enw\_add\_cumulative\_membership

*Add a cumulative membership effect to a data.frame*


---

**Description**

This function adds a cumulative membership effect to a data frame. This is useful for specifying models such as random walks (using `rw()`) where these features can be used in the design matrix with the appropriate formula. Supports grouping via the optional `.group` column. Note that cumulative membership is indexed to start with zero (i.e. the first observation is assigned a cumulative membership of zero).

**Usage**

```
enw_add_cumulative_membership(metaobs, feature, copy = TRUE)
```

**Arguments**

|         |                                                                                    |
|---------|------------------------------------------------------------------------------------|
| metaobs | A data.frame with a column named feature that contains a numeric vector of values. |
| feature | The name of the column in metaobs that contains the numeric vector of values.      |
| copy    | Should metaobs be copied (default) or modified in place?                           |

**Value**

A data.frame with a new columns `cfeature$` that contain the cumulative membership effect for each value of feature. For example if the original feature was week (with numeric entries 1, 2, 3) then the new columns will be `cweek1`, `cweek2`, and `cweek3`.

**See Also**

Functions used to formulate models `enw_add_pooling_effect()`, `enw_design()`, `enw_effects_metadata()`, `enw_one_hot_encode_feature()`

## Examples

```
metaobs <- data.frame(week = 1:2)
enw_add_cumulative_membership(metaobs, "week")

metaobs <- data.frame(week = 1:3, .group = c(1,1,2))
enw_add_cumulative_membership(metaobs, "week")
```

---

enw\_add\_delay

*Add a delay variable to the observations*


---

## Description

This helper function takes a `data.frame` or `data.table` of observations and adds the delay (numeric, in days) between `reference_date` and `report_date` for each observation.

## Usage

```
enw_add_delay(obs, timestep = "day", copy = TRUE)
```

## Arguments

|                       |                                                                                                                                                                                                                                                                        |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>obs</code>      | A <code>data.frame</code> containing at least the following variables: <code>reference_date</code> (index date of interest), <code>report_date</code> (report date for observations), and <code>confirm</code> (cumulative observations by reference and report date). |
| <code>timestep</code> | The timestep to used. This can be a string ("day", "week") or a numeric whole number representing the number of days. Note that "month" is not currently supported in user-facing functions and will throw an error if used.                                           |
| <code>copy</code>     | Should <code>obs</code> be copied (default) or modified in place?                                                                                                                                                                                                      |

## Value

A `data.table` of observations with a new column `delay`.

## See Also

Preprocessing functions [enw\\_add\\_max\\_reported\(\)](#), [enw\\_add\\_metaobs\\_features\(\)](#), [enw\\_assign\\_group\(\)](#), [enw\\_complete\\_dates\(\)](#), [enw\\_construct\\_data\(\)](#), [enw\\_extend\\_date\(\)](#), [enw\\_filter\\_delay\(\)](#), [enw\\_filter\\_reference\\_dates\(\)](#), [enw\\_filter\\_reference\\_dates\\_by\\_report\\_start\(\)](#), [enw\\_filter\\_report\\_dates\(\)](#), [enw\\_flag\\_observed\\_observations\(\)](#), [enw\\_impute\\_na\\_observations\(\)](#), [enw\\_latest\\_data\(\)](#), [enw\\_metadata\(\)](#), [enw\\_metadata\\_delay\(\)](#), [enw\\_missing\\_reference\(\)](#), [enw\\_obs\\_at\\_delay\(\)](#), [enw\\_preprocess\\_data\(\)](#), [enw\\_reporting\\_triangle\(\)](#), [enw\\_reporting\\_triangle\\_to\\_long\(\)](#), [enw\\_retrospective\(\)](#)

## Examples

```
obs <- data.frame(report_date = as.Date("2021-01-01") + -2:0)
obs$reference_date <- as.Date("2021-01-01")
enw_add_delay(obs)
```

---

|                   |                                                                   |
|-------------------|-------------------------------------------------------------------|
| enw_add_incidence | <i>Calculate incidence of new reports from cumulative reports</i> |
|-------------------|-------------------------------------------------------------------|

---

## Description

Computes incident counts from cumulative reports. Users should typically call [enw\\_filter\\_reference\\_dates\\_by\\_report\\_start](#) before this function to remove reference dates that precede the earliest report date, which would otherwise produce spurious leading entries.

## Usage

```
enw_add_incidence(obs, set_negatives_to_zero = TRUE, by = NULL, copy = TRUE)
```

## Arguments

|                       |                                                                                                                                                                                                                                                             |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| obs                   | A <code>data.frame</code> containing at least the following variables: reference date (index date of interest), report_date (report date for observations), and confirm (cumulative observations by reference and report date).                             |
| set_negatives_to_zero | Logical, defaults to TRUE. Should negative counts (for calculated incidence of observations) be set to zero? Currently downstream modelling does not support negative counts and so setting must be TRUE if intending to use <a href="#">epinowcast()</a> . |
| by                    | A character vector describing the stratification of observations. This defaults to no grouping. This should be used when modelling multiple time series in order to identify them for downstream modelling                                                  |
| copy                  | Should obs be copied (default) or modified in place?                                                                                                                                                                                                        |

## Value

The input `data.frame` with a new variable `new_confirm`. If `max_confirm` is present in the `data.frame`, then the proportion reported on each day (`prop_reported`) will also be added.

## See Also

Data converters [enw\\_add\\_cumulative\(\)](#), [enw\\_aggregate\\_cumulative\(\)](#), [enw\\_incidence\\_to\\_linelist\(\)](#), [enw\\_linelist\\_to\\_incidence\(\)](#)

## Examples

```
# Default reconstruct incidence
dt <- germany_covid19_hosp[location == "DE"][age_group == "00+"]
dt <- enw_filter_reference_dates_by_report_start(dt)
enw_add_incidence(dt)

# Make use of maximum reported to calculate empirical
# daily reporting
dt <- germany_covid19_hosp[location == "DE"]
```

```

    age_group == "00+"
  ]
dt <- enw_add_max_reported(dt)
dt <- enw_filter_reference_dates_by_report_start(dt)
enw_add_incidence(dt)

```

---

enw\_add\_latest\_obs\_to\_nowcast

*Add latest observations to nowcast output*


---

## Description

Add the latest observations to the nowcast output. This is useful for plotting the nowcast against the latest observations.

## Usage

```
enw_add_latest_obs_to_nowcast(nowcast, obs)
```

## Arguments

|         |                                                                                                                                                                                                                                                                                                                                                                                    |
|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| nowcast | A data.frame of nowcast output from <a href="#">enw_nowcast_summary()</a> .                                                                                                                                                                                                                                                                                                        |
| obs     | An observation data.frame containing reference_date columns of the same length as the number of rows in the posterior and the most up to date observation for each date. This is used to align the posterior with the observations. The easiest source of this data is the output of latest output of <a href="#">enw_preprocess_data()</a> or <a href="#">enw_latest_data()</a> . |

## Value

A data.frame of nowcast output with the latest observations added.

## See Also

Functions used for postprocessing of model fits [.check\\_primarycensored\(\)](#), [.delay\\_draw\\_columns\(\)](#), [.discretise\\_parametric\\_pmf\(\)](#), [build\\_ord\\_obs\(\)](#), [enw\\_nowcast\\_samples\(\)](#), [enw\\_nowcast\\_summary\(\)](#), [enw\\_posterior\(\)](#), [enw\\_posterior\\_delay\(\)](#), [enw\\_pp\\_summary\(\)](#), [enw\\_quantiles\\_to\\_long\(\)](#), [enw\\_summarise\\_samples\(\)](#), [subset\\_obs\(\)](#)

## Examples

```

fit <- enw_example("nowcast")
obs <- enw_example("obs")
nowcast <- summary(fit, type = "nowcast")
enw_add_latest_obs_to_nowcast(nowcast, obs)

```

---

enw\_add\_max\_reported    *Add the maximum number of reported cases for each reference\_date*


---

## Description

This is a helper function which adds the maximum (in the sense of latest observed) number of reported cases for each reference\_date and computes the proportion of already reported cases for each combination of reference\_date and report\_date.

## Usage

```
enw_add_max_reported(obs, copy = TRUE)
```

## Arguments

|      |                                                                                                                                                                                                                    |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| obs  | A data.frame containing at least the following variables: reference_date (index date of interest), report_date (report date for observations), and confirm (cumulative observations by reference and report date). |
| copy | Should obs be copied (default) or modified in place?                                                                                                                                                               |

## Value

A data.table with new columns max\_confirm and cum\_prop\_reported. max\_confirm is the maximum number of cases reported for a certain reference\_date. cum\_prop\_reported is the proportion of cases for a certain reference\_date that are reported until a given report\_day, relative to all cases so far observed for this reference\_date.

## See Also

Preprocessing functions [enw\\_add\\_delay\(\)](#), [enw\\_add\\_metaobs\\_features\(\)](#), [enw\\_assign\\_group\(\)](#), [enw\\_complete\\_dates\(\)](#), [enw\\_construct\\_data\(\)](#), [enw\\_extend\\_date\(\)](#), [enw\\_filter\\_delay\(\)](#), [enw\\_filter\\_reference\\_dates\(\)](#), [enw\\_filter\\_reference\\_dates\\_by\\_report\\_start\(\)](#), [enw\\_filter\\_report\\_dates\(\)](#), [enw\\_flag\\_observed\\_observations\(\)](#), [enw\\_impute\\_na\\_observations\(\)](#), [enw\\_latest\\_data\(\)](#), [enw\\_metadata\(\)](#), [enw\\_metadata\\_delay\(\)](#), [enw\\_missing\\_reference\(\)](#), [enw\\_obs\\_at\\_delay\(\)](#), [enw\\_preprocess\\_data\(\)](#), [enw\\_reporting\\_triangle\(\)](#), [enw\\_reporting\\_triangle\\_to\\_long\(\)](#), [enw\\_retrospective\(\)](#)

## Examples

```
obs <- data.frame(report_date = as.Date("2021-01-01") + 0:2)
obs$reference_date <- as.Date("2021-01-01")
obs$confirm <- 1:3
enw_add_max_reported(obs)
```

---

enw\_add\_metaobs\_features

*Add common metadata variables*


---

## Description

If not already present, annotates time series data with metadata commonly used in models: day of week, and days, weeks, and months since start of time series.

## Usage

```
enw_add_metaobs_features(
  metaobs,
  holidays = NULL,
  holidays_to = "Sunday",
  datecol = "date"
)
```

## Arguments

|             |                                                                                                                                                                                                   |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| metaobs     | Raw data, coercible via <code>data.table::as.data.table()</code> . Coerced object must have <code>Dates</code> column corresponding to datecol name.                                              |
| holidays    | a (potentially empty) vector of dates (or input coercible to such; see <code>coerce_date()</code> ). The <code>day_of_week</code> column will be set to <code>holidays_to</code> for these dates. |
| holidays_to | A character string to assign to holidays, when <code>holidays</code> argument non-empty. Replaces the <code>day_of_week</code> column value                                                       |
| datecol     | The column in <code>metaobs</code> corresponding to pertinent dates.                                                                                                                              |

## Details

Effects models often need to include covariates for time-based features, such as day of the week (e.g. to reflect different care-seeking and/or reporting behaviour).

This function is called from within `enw_preprocess_data()` to systematically annotate `metaobs` with these commonly used metadata, if not already present.

However, it can also be used directly on other data.

## Value

A copy of the `metaobs` input, with additional columns:

- `day_of_week`, a factor of values as output from `weekdays()` and possibly as `holiday_to` if distinct from weekdays values
- `day`, numeric, 0 based from start of time series
- `week`, numeric, 0 based from start of time series
- `month`, numeric, 0 based from start of time series

**See Also**

Preprocessing functions `enw_add_delay()`, `enw_add_max_reported()`, `enw_assign_group()`, `enw_complete_dates()`, `enw_construct_data()`, `enw_extend_date()`, `enw_filter_delay()`, `enw_filter_reference_dates()`, `enw_filter_reference_dates_by_report_start()`, `enw_filter_report_dates()`, `enw_flag_observed_observation()`, `enw_impute_na_observations()`, `enw_latest_data()`, `enw_metadata()`, `enw_metadata_delay()`, `enw_missing_reference()`, `enw_obs_at_delay()`, `enw_preprocess_data()`, `enw_reporting_triangle()`, `enw_reporting_triangle_to_long()`, `enw_retrospective()`

**Examples**

```
# make some example date
nat_germany_hosp <- subset(
  germany_covid19_hosp,
  location == "DE" & age_group == "80+"
)[1:40]

basemeta <- enw_add_metaobs_features(
  nat_germany_hosp,
  datecol = "report_date"
)
basemeta

# with holidays - n.b.: holidays not found are silently ignored
holidaymeta <- enw_add_metaobs_features(
  nat_germany_hosp,
  datecol = "report_date",
  holidays = c(
    "2021-04-04", "2021-04-05",
    "2021-05-01", "2021-05-13",
    "2021-05-24"
  ),
  holidays_to = "Holiday"
)
holidaymeta
subset(holidaymeta, day_of_week == "Holiday")
```

---

`enw_add_pooling_effect`

*Add a pooling effect to model design metadata*

---

**Description**

This function adds a pooling effect to the metadata returned by `enw_effects_metadata()`. It does this updating the fixed column to 0 for the effects that match the string argument and adding a new column `var_name` that is 1 for the effects that match the string argument and 0 otherwise.

**Usage**

```
enw_add_pooling_effect(effects, var_name = "sd", finder_fn = startsWith, ...)
```

**Arguments**

|           |                                                                                                                                                                                                                                                                                                                                          |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| effects   | <p>A data.table with the following columns:</p> <ul style="list-style-type: none"> <li>• effects: the name of the effect</li> <li>• fixed: a logical indicating whether the effect is fixed (1) or random (0).</li> </ul> <p>This is the output of <code>enw_effects_metadata()</code>.</p>                                              |
| var_name  | The name of the new column that will be added to the effects data.table. This column will be 1 for the effects that match the string and 0 otherwise. Defaults to 'sd'.                                                                                                                                                                  |
| finder_fn | A function that will be used to find the effects that match the string. Defaults to <code>startsWith()</code> . This can be any function that takes a character as it's first argument (the effects\$effects column) and then any other other arguments in ... and returns a logical vector indicating whether the effects were matched. |
| ...       | Additional arguments to finder_fn. E.g. for the finder_fn = startsWith default, this should be prefix = "somestring".                                                                                                                                                                                                                    |

**Value**

A data.table with the following columns:

- effects: the name of the effect
- fixed: a logical indicating whether the effect is fixed (1) or random (0).
- Argument supplied to var\_name: a logical indicating whether the effect should be pooled (1) or not (0).

**See Also**

Functions used to formulate models `enw_add_cumulative_membership()`, `enw_design()`, `enw_effects_metadata()`, `enw_one_hot_encode_feature()`

**Examples**

```
data <- data.frame(a = 1:3, b = as.character(1:3), c = c(1,1,2))
design <- enw_design(a ~ b + c, data)$design
effects <- enw_effects_metadata(design)
enw_add_pooling_effect(effects, prefix = "b")
```

---

enw\_aggregate\_cumulative

*Aggregate observations over a given timestep for both report and reference dates.*

---

## Description

This function aggregates observations over a specified timestep, ensuring alignment on the same day of week for report and reference dates. It is useful for aggregating data to a weekly timestep, for example which may be desirable if testing using a weekly timestep or if you are very concerned about runtime. Note that the start of the timestep will be determined by `min_date` + a single timestep (i.e. the first timestep will be "2022-10-23" if the minimum reference date is "2022-10-16"). Observations where the report dates do not form a complete timestep will be dropped from the aggregated output.

## Usage

```
enw_aggregate_cumulative(
  obs,
  timestep = "day",
  by = NULL,
  min_reference_date = min(obs$reference_date, na.rm = TRUE),
  copy = TRUE
)
```

## Arguments

|                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>obs</code>                | An object coercible to a <code>data.table</code> (such as a <code>data.frame</code> ) which must have a <code>new_confirm</code> numeric column, and <code>report_date</code> and <code>reference_date</code> date columns. The input must have a timestep of a day and be complete. See <a href="#">enw_complete_dates()</a> for more information. If NA values are present in the <code>confirm</code> column then these will be set to zero before aggregation this may not be desirable if this missingness is meaningful. Before aggregation, dates will be completed up to the last reference date to ensure all reference dates have the required report dates for the specified timestep. |
| <code>timestep</code>           | The timestep to used. This can be a string ("day", "week") or a numeric whole number representing the number of days. Note that "month" is not currently supported in user-facing functions and will throw an error if used.                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>by</code>                 | A character vector of variables to also aggregate by (i.e. as well as using the <code>reference_date</code> and <code>report_date</code> ). If not supplied then the function will aggregate by just the <code>reference_date</code> and <code>report_date</code> .                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>min_reference_date</code> | The minimum reference date to start the aggregation from. Note that the timestep will start from the minimum reference date + a single time step (i.e. the first timestep will be "2022-10-23" if the minimum reference date is "2022-10-16"). The default is the minimum reference date in the <code>obs</code> object. Other sensible values would be the minimum report date in the <code>obs</code> object + 1 day if reporting is already weekly and you wish to ensure that the timestep of the output matches the reporting timestep.                                                                                                                                                      |
| <code>copy</code>               | Should <code>obs</code> be copied (default) or modified in place?                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

## Value

A `data.table` with aggregated observations.

**See Also**

Data converters [enw\\_add\\_cumulative\(\)](#), [enw\\_add\\_incidence\(\)](#), [enw\\_incidence\\_to\\_linelist\(\)](#), [enw\\_linelist\\_to\\_incidence\(\)](#)

**Examples**

```
nat_hosp <- germany_covid19_hosp[location == "DE"][age_group == "00+"]
enw_aggregate_cumulative(nat_hosp, timestep = "week")
```

---

|                  |                                                   |
|------------------|---------------------------------------------------|
| enw_assign_group | <i>Assign a group to each row of a data.table</i> |
|------------------|---------------------------------------------------|

---

**Description**

Assign a group to each row of a data.table. If by is specified, then each unique combination of the columns in by will be assigned a unique group. If by is not specified, then all rows will be assigned to the same group.

**Usage**

```
enw_assign_group(obs, by = NULL, copy = TRUE)
```

**Arguments**

|      |                                                                              |
|------|------------------------------------------------------------------------------|
| obs  | A data.table or data.frame without a .group column.                          |
| by   | A character vector of column names to group by. Defaults to an empty vector. |
| copy | A logical; make a copy (default) of obs or modify it in place?               |

**Value**

A data.table with a .group column added ordered by .group and the existing key of obs.

**See Also**

Preprocessing functions [enw\\_add\\_delay\(\)](#), [enw\\_add\\_max\\_reported\(\)](#), [enw\\_add\\_metaobs\\_features\(\)](#), [enw\\_complete\\_dates\(\)](#), [enw\\_construct\\_data\(\)](#), [enw\\_extend\\_date\(\)](#), [enw\\_filter\\_delay\(\)](#), [enw\\_filter\\_reference\\_dates\(\)](#), [enw\\_filter\\_reference\\_dates\\_by\\_report\\_start\(\)](#), [enw\\_filter\\_report\\_dates\(\)](#), [enw\\_flag\\_observed\\_observations\(\)](#), [enw\\_impute\\_na\\_observations\(\)](#), [enw\\_latest\\_data\(\)](#), [enw\\_metadata\(\)](#), [enw\\_metadata\\_delay\(\)](#), [enw\\_missing\\_reference\(\)](#), [enw\\_obs\\_at\\_delay\(\)](#), [enw\\_preprocess\\_data\(\)](#), [enw\\_reporting\\_triangle\(\)](#), [enw\\_reporting\\_triangle\\_to\\_long\(\)](#), [enw\\_retrospective\(\)](#)

**Examples**

```
obs <- data.frame(x = 1:3, y = 1:3)
enw_assign_group(obs)
enw_assign_group(obs, by = "x")
```

---

enw\_complete\_dates      *Complete missing reference and report dates*


---

## Description

Ensures that all reference and report dates are present for all groups based on the maximum and minimum dates found in the data. This function may be of use to users when preprocessing their data. In general all features that you may consider using as grouping variables or as covariates need to be included in the by variable.

## Usage

```
enw_complete_dates(
  obs,
  by = NULL,
  max_delay,
  min_date = min(obs$reference_date, na.rm = TRUE),
  max_date = max(obs$report_date, na.rm = TRUE),
  timestep = "day",
  missing_reference = TRUE,
  completion_beyond_max_report = FALSE,
  flag_observation = FALSE
)
```

## Arguments

- |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| obs       | A data.frame containing at least the following variables: reference_date (index date of interest), report_date (report date for observations), and confirm (cumulative observations by reference and report date).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| by        | A character vector describing the stratification of observations. This defaults to no grouping. This should be used when modelling multiple time series in order to identify them for downstream modelling                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| max_delay | <p>The maximum delay to model in the delay distribution, specified in units of the timestep (e.g., if timestep = "week", then max_delay = 3 means 3 weeks). If not specified the maximum observed delay is assumed to be the true maximum delay in the model. Otherwise, an integer greater than or equal to 1 can be specified. Observations with delays larger than the maximum delay will be dropped. If the specified maximum delay is too short, nowcasts can be biased as important parts of the true delay distribution are cut off. At the same time, computational cost scales non-linearly with this setting, so you want the maximum delay to be as long as necessary, but not much longer.</p> <p>Steps to take to determine the maximum delay:</p> <ul style="list-style-type: none"> <li>• Consider what is realistic and relevant for your application.</li> <li>• Check the proportion of observations reported (prop_reported) by delay in the new_confirm output of enw_preprocess_obs.</li> <li>• Use <a href="#">check_max_delay()</a> to check the coverage of a candidate max_delay.</li> </ul> |

- If in doubt, check if increasing the maximum delay noticeably changes the delay distribution or nowcasts as estimated by `epinowcast`. If it does, your maximum delay may still be too short.

Note that delays are zero indexed and so include the reference date and `max_delay - 1` other intervals (i.e. a `max_delay` of 1 corresponds to no delay).

|                                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>min_date</code>                     | The minimum date to include in the data. Defaults to the minimum reference date found in the data.                                                                                                                                                                                                                                                                                                                                               |
| <code>max_date</code>                     | The maximum date to include in the data. Defaults to the maximum report date found in the data.                                                                                                                                                                                                                                                                                                                                                  |
| <code>timestep</code>                     | The timestep to used. This can be a string ("day", "week") or a numeric whole number representing the number of days. Note that "month" is not currently supported in user-facing functions and will throw an error if used.                                                                                                                                                                                                                     |
| <code>missing_reference</code>            | Logical, should entries for cases with missing reference date be completed as well?, Default: TRUE                                                                                                                                                                                                                                                                                                                                               |
| <code>completion_beyond_max_report</code> | Logical, should entries be completed beyond the maximum date found in the data? Default: FALSE                                                                                                                                                                                                                                                                                                                                                   |
| <code>flag_observation</code>             | Logical, should observations that have been imputed as missing be flagged as not observed?. Makes use of <code>enw_flag_observed_observations()</code> to add a <code>.observed</code> logical vector which indicates if observations have been imputed. This vector can then be passed to the <code>observation_indicator</code> argument of <code>enw_obs()</code> to control if these observations are used in the likelihood. Default: FALSE |

## Value

A data.table with completed entries for all combinations of reference dates, groups and possible report dates.

## See Also

Preprocessing functions `enw_add_delay()`, `enw_add_max_reported()`, `enw_add_metaobs_features()`, `enw_assign_group()`, `enw_construct_data()`, `enw_extend_date()`, `enw_filter_delay()`, `enw_filter_reference_dates_by_report_start()`, `enw_filter_report_dates()`, `enw_flag_observed_observations()`, `enw_impute_na_observations()`, `enw_latest_data()`, `enw_metadata()`, `enw_metadata_delay()`, `enw_missing_reference()`, `enw_obs_at_delay()`, `enw_preprocess_data()`, `enw_reporting_triangle()`, `enw_reporting_triangle_to_long()`, `enw_retrospective()`

## Examples

```
obs <- data.frame(
  report_date = c("2021-10-01", "2021-10-03"), reference_date = "2021-10-01",
  confirm = 1
)
enw_complete_dates(obs)
```

```
# Allow completion beyond the maximum date found in the data
enw_complete_dates(obs, completion_beyond_max_report = TRUE, max_delay = 10)
```

---

enw\_construct\_data      *Construct preprocessed data*

---

## Description

This function is used internally by [enw\\_preprocess\\_data\(\)](#) to combine various pieces of processed observed data into a single object. It is exposed to the user in order to allow for modular data preprocessing though this is not currently recommended. See documentation and code of [enw\\_preprocess\\_data\(\)](#) for more on the expected inputs.

## Usage

```
enw_construct_data(
  obs,
  new_confirm,
  latest,
  missing_reference,
  reporting_triangle,
  metareport,
  metareference,
  metadelay,
  max_delay,
  timestep,
  by
)
```

## Arguments

|                    |                                                                                                                      |
|--------------------|----------------------------------------------------------------------------------------------------------------------|
| obs                | Observations with the addition of empirical reporting proportions and and restricted to the specified maximum delay. |
| new_confirm        | Incidence of notifications by reference and report date. Empirical reporting distributions are also added.           |
| latest             | The latest available observations.                                                                                   |
| missing_reference  | A data.frame of reported observations that are missing the reference date.                                           |
| reporting_triangle | Incident observations by report and reference date in the standard reporting triangle matrix format.                 |
| metareport         | Metadata for report dates.                                                                                           |
| metareference      | Metadata reference dates derived from observations.                                                                  |
| metadelay          | Metadata for reporting delays produced using <a href="#">enw_metadata_delay()</a> .                                  |

|           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| max_delay | <p>The maximum delay to model in the delay distribution, specified in units of the timestep (e.g., if timestep = "week", then max_delay = 3 means 3 weeks). If not specified the maximum observed delay is assumed to be the true maximum delay in the model. Otherwise, an integer greater than or equal to 1 can be specified. Observations with delays larger than the maximum delay will be dropped. If the specified maximum delay is too short, nowcasts can be biased as important parts of the true delay distribution are cut off. At the same time, computational cost scales non-linearly with this setting, so you want the maximum delay to be as long as necessary, but not much longer.</p> <p>Steps to take to determine the maximum delay:</p> <ul style="list-style-type: none"> <li>• Consider what is realistic and relevant for your application.</li> <li>• Check the proportion of observations reported (prop_reported) by delay in the new_confirm output of enw_preprocess_obs.</li> <li>• Use <code>check_max_delay()</code> to check the coverage of a candidate max_delay.</li> <li>• If in doubt, check if increasing the maximum delay noticeably changes the delay distribution or nowcasts as estimated by epinowcast. If it does, your maximum delay may still be too short.</li> </ul> <p>Note that delays are zero indexed and so include the reference date and max_delay - 1 other intervals (i.e. a max_delay of 1 corresponds to no delay).</p> |
| timestep  | <p>The timestep to used in the process model (i.e. the reference date model). This can be a string ("day", "week", "month") or a numeric whole number representing the number of days. If your data does not have this timestep then you may wish to make use of <code>enw_aggregate_cumulative()</code> to aggregate your data to the desired timestep.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| by        | <p>A character vector describing the stratification of observations. This defaults to no grouping. This should be used when modelling multiple time series in order to identify them for downstream modelling</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

## Value

A data.table containing processed observations as a series of nested data.frames as well as variables containing metadata. These are:

- obs: (observations with the addition of empirical reporting proportions and restricted to the specified maximum delay).
- new\_confirm: Incidence of notifications by reference and report date. Empirical reporting distributions are also added.
- latest: The latest available observations.
- missing\_reference: Observations missing reference dates.
- reporting\_triangle: Incident observations by report and reference date in the standard reporting triangle matrix format.
- metareference: Metadata reference dates derived from observations.
- metrareport: Metadata for report dates.
- metadelay: Metadata for reporting delays produced using `enw_metadata_delay()`.
- max\_delay: Maximum delay to be modelled by epinowcast.

- time: Numeric, number of timepoints in the data.
- snapshots: Numeric, number of available data snapshots to use for nowcasting.
- groups: Numeric, Number of groups/strata in the supplied observations (set using by).
- max\_date: The maximum available report date.

### See Also

Preprocessing functions [enw\\_add\\_delay\(\)](#), [enw\\_add\\_max\\_reported\(\)](#), [enw\\_add\\_metaobs\\_features\(\)](#), [enw\\_assign\\_group\(\)](#), [enw\\_complete\\_dates\(\)](#), [enw\\_extend\\_date\(\)](#), [enw\\_filter\\_delay\(\)](#), [enw\\_filter\\_reference\\_dates\(\)](#), [enw\\_filter\\_reference\\_dates\\_by\\_report\\_start\(\)](#), [enw\\_filter\\_report\\_dates\(\)](#), [enw\\_flag\\_observed\\_observation\(\)](#), [enw\\_impute\\_na\\_observations\(\)](#), [enw\\_latest\\_data\(\)](#), [enw\\_metadata\(\)](#), [enw\\_metadata\\_delay\(\)](#), [enw\\_missing\\_reference\(\)](#), [enw\\_obs\\_at\\_delay\(\)](#), [enw\\_preprocess\\_data\(\)](#), [enw\\_reporting\\_triangle\(\)](#), [enw\\_reporting\\_triangle\\_to\\_long\(\)](#), [enw\\_retrospective\(\)](#)

### Examples

```
pobs <- enw_example("preprocessed")
enw_construct_data(
  obs = pobs$obs[[1]],
  new_confirm = pobs$new_confirm[[1]],
  latest = pobs$latest[[1]],
  missing_reference = pobs$missing_reference[[1]],
  reporting_triangle = pobs$reporting_triangle[[1]],
  metareport = pobs$metareport[[1]],
  metareference = pobs$metareference[[1]],
  metadelay = pobs$metadelay[[1]],
  max_delay = pobs$max_delay,
  timestep = pobs$timestep[[1]],
  by = c()
)
```

---

```
enw_dayofweek_structural_reporting
```

*Create day-of-week structural reporting pattern*

---

### Description

Creates a structural reporting pattern for cases where reporting only occurs on specific days of the week (e.g., Wednesday-only reporting). This is a convenience function that builds on [enw\\_structural\\_reporting\\_metadata\(\)](#).

### Usage

```
enw_dayofweek_structural_reporting(pobs, day_of_week)
```

### Arguments

|             |                                                                                                          |
|-------------|----------------------------------------------------------------------------------------------------------|
| pobs        | A preprocessed observation list from <a href="#">enw_preprocess_data()</a> .                             |
| day_of_week | Character vector of weekday names when reporting occurs (e.g., "Wednesday" or c("Monday", "Wednesday")). |

**Value**

A data.table with columns:

- .group: Group identifier
- date: Reference date
- report\_date: Report date
- report: Binary indicator (1 = reporting occurs, 0 = no reporting)

**See Also**

Helper functions for model modules [add\\_max\\_observed\\_delay\(\)](#), [add\\_pmfs\(\)](#), [convolution\\_matrix\(\)](#), [delay\\_only\\_ltotal\(\)](#), [delay\\_only\\_total\(\)](#), [enw\\_reference\\_by\\_report\(\)](#), [enw\\_reps\\_with\\_complete\\_refs\(\)](#), [enw\\_structural\\_reporting\\_metadata\(\)](#), [extract\\_obs\\_metadata\(\)](#), [extract\\_sparse\\_matrix\(\)](#), [latest\\_obs\\_as\\_matrix\(\)](#)

**Examples**

```
## Not run:
pobs <- enw_preprocess_data(obs, max_delay = 30)

# Wednesday-only reporting
enw_dayofweek_structural_reporting(
  pobs,
  day_of_week = "Wednesday"
)

# Multiple reporting days
enw_dayofweek_structural_reporting(
  pobs,
  day_of_week = c("Monday", "Wednesday", "Friday")
)

## End(Not run)
```

---

enw\_delay\_categories    *Categorise new confirmations by delay group*

---

**Description**

Categorises incidence by delay group with empirical reporting proportions. Intended for use with the [plot.enw\\_preprocess\\_data\(\)](#) visualisation types that show reporting patterns by delay.

**Usage**

```
enw_delay_categories(pobs, delay_group_thresh)
```

**Arguments**

`pobs` A preprocessed data object as produced by [enw\\_preprocess\\_data\(\)](#).

`delay_group_thresh` A numeric vector defining left-closed interval thresholds for grouping reporting delays. The smallest value should be zero and the largest should exceed `max_delay` (intervals are left-closed, right-open). Delays outside the range are dropped.

**Value**

A `data.table` of notification incidence by reference date and delay group, including columns `prop_reported` and `cum_prop_reported`.

**See Also**

Plotting functions [enw\\_delay\\_quantiles\(\)](#), [enw\\_plot\\_delay\\_counts\(\)](#), [enw\\_plot\\_delay\\_cumulative\(\)](#), [enw\\_plot\\_delay\\_fraction\(\)](#), [enw\\_plot\\_delay\\_quantiles\(\)](#), [enw\\_plot\\_nowcast\\_quantiles\(\)](#), [enw\\_plot\\_obs\(\)](#), [enw\\_plot\\_pp\\_quantiles\(\)](#), [enw\\_plot\\_quantiles\(\)](#), [enw\\_plot\\_theme\(\)](#), [plot.enw\\_preprocess\\_data\(\)](#), [plot.epinowcast\(\)](#)

**Examples**

```
pobs <- enw_example("preprocessed_observations")
enw_delay_categories(pobs, delay_group_thresh = c(0, 2, 5, 10, 21))
```

---

|                                  |                                                    |
|----------------------------------|----------------------------------------------------|
| <code>enw_delay_quantiles</code> | <i>Empirical delay quantiles by reference date</i> |
|----------------------------------|----------------------------------------------------|

---

**Description**

Computes empirical quantiles of the reporting delay distribution for each reference date. Intended for use with the "delay\_quantiles" plot type in [plot.enw\\_preprocess\\_data\(\)](#).

**Usage**

```
enw_delay_quantiles(pobs, quantiles = c(0.1, 0.5, 0.9))
```

**Arguments**

`pobs` A preprocessed data object as produced by [enw\\_preprocess\\_data\(\)](#).

`quantiles` A numeric vector of probabilities for which quantiles are computed. Defaults to `c(0.1, 0.5, 0.9)`.

**Value**

A `data.table` with columns for each quantile by reference date.

**See Also**

Plotting functions `enw_delay_categories()`, `enw_plot_delay_counts()`, `enw_plot_delay_cumulative()`, `enw_plot_delay_fraction()`, `enw_plot_delay_quantiles()`, `enw_plot_nowcast_quantiles()`, `enw_plot_obs()`, `enw_plot_pp_quantiles()`, `enw_plot_quantiles()`, `enw_plot_theme()`, `plot.enw_preprocess_data()`, `plot.epinowcast()`

**Examples**

```
pobs <- enw_example("preprocessed_observations")
enw_delay_quantiles(pobs)
```

---

enw\_design

*A helper function to construct a design matrix from a formula*


---

**Description**

This function is a wrapper around `stats::model.matrix()` that can optionally return a sparse design matrix defined as the unique number of rows in the design matrix and an index vector that allows the full design matrix to be reconstructed. This is useful for models that have many repeated rows in the design matrix and that are computationally expensive to fit. This function also allows for the specification of contrasts for categorical variables.

**Usage**

```
enw_design(formula, data, no_contrasts = FALSE, sparse = TRUE, ...)
```

**Arguments**

|                           |                                                                                                                                                                                                                                                     |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>formula</code>      | An R formula.                                                                                                                                                                                                                                       |
| <code>data</code>         | A <code>data.frame</code> containing the variables in the formula.                                                                                                                                                                                  |
| <code>no_contrasts</code> | A vector of variable names that should not be converted to contrasts. If <code>no_contrasts = FALSE</code> then all categorical variables will use contrasts. If <code>no_contrasts = TRUE</code> then no categorical variables will use contrasts. |
| <code>sparse</code>       | Logical, if <code>TRUE</code> return a sparse design matrix. Defaults to <code>TRUE</code> .                                                                                                                                                        |
| <code>...</code>          | Arguments passed on to <code>stats::model.matrix</code>                                                                                                                                                                                             |
|                           | <code>object</code> an object of an appropriate class. For the default method, a model <code>formula</code> or a <code>terms</code> object.                                                                                                         |

**Value**

A list containing the formula, the design matrix, and the index.

**See Also**

Functions used to formulate models `enw_add_cumulative_membership()`, `enw_add_pooling_effect()`, `enw_effects_metadata()`, `enw_one_hot_encode_feature()`

**Examples**

```
data <- data.frame(a = 1:3, b = as.character(1:3), c = c(1,1,2))
enw_design(a ~ b + c, data)
enw_design(a ~ b + c, data, no_contrasts = TRUE)
enw_design(a ~ b + c, data, no_contrasts = c("b"))
enw_design(a ~ c, data, sparse = TRUE)
enw_design(a ~ c, data, sparse = FALSE)
```

---

enw\_effects\_metadata    *Extracts metadata from a design matrix*

---

**Description**

This function extracts metadata from a design matrix and returns a `data.table` with the following columns:

- `effects`: the name of the effect
- `fixed`: a logical indicating whether the effect is fixed (1) or random (0).

It automatically drops the intercept (defined as "(Intercept)").

This function is useful for constructing a model design object for random effects when used in combination with `enw_add_pooling_effect`.

**Usage**

```
enw_effects_metadata(design)
```

**Arguments**

`design`                      A design matrix as returned by `stats::model.matrix()`.

**Value**

A `data.table` with the following columns:

- `effects`: the name of the effect
- `fixed`: a logical indicating whether the effect is fixed (1) or random (0)

**See Also**

Functions used to formulate models `enw_add_cumulative_membership()`, `enw_add_pooling_effect()`, `enw_design()`, `enw_one_hot_encode_feature()`

**Examples**

```
data <- data.frame(a = 1:3, b = as.character(1:3), c = c(1,1,2))
design <- enw_design(a ~ b + c, data)$design
enw_effects_metadata(design)
```

---

`enw_example`*Load a package example*

---

## Description

Loads examples of nowcasts produced using example scripts. Used to streamline examples, in package tests and to enable users to explore package functionality without needing to install `cmdstanr`.

## Usage

```
enw_example(  
  type = c("nowcast", "preprocessed_observations", "observations", "script")  
)
```

## Arguments

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>type</code> | A character string indicating the example to load. Supported options are <ul style="list-style-type: none"><li>• "nowcast", for <code>epinowcast()</code> applied to <a href="#">germany_covid19_hosp</a></li><li>• "preprocessed_observations", for <code>enw_preprocess_data()</code> applied to <a href="#">germany_covid19_hosp</a></li><li>• "observations", for <code>enw_latest_data()</code> applied to <a href="#">germany_covid19_hosp</a></li><li>• "script", the code used to generate these examples.</li></ul> |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Value

Depending on type, a `data.table` of the requested output OR the file name(s) to generate these outputs (`type = "script"`)

## See Also

Package data sets [germany\\_covid19\\_hosp](#)

## Examples

```
# Load the nowcast  
enw_example(type = "nowcast")  
  
# Load the preprocessed observations  
enw_example(type = "preprocessed_observations")  
  
# Load the latest observations  
enw_example(type = "observations")  
  
# Load the script used to generate these examples  
# Optionally source this script to regenerate the example  
readLines(enw_example(type = "script"))
```

---

|                 |                                 |
|-----------------|---------------------------------|
| enw_expectation | <i>Expectation model module</i> |
|-----------------|---------------------------------|

---

## Description

Expectation model module

## Usage

```
enw_expectation(
  r = ~0 + (1 | day:.group),
  generation_time = 1,
  observation = ~1,
  latent_reporting_delay = 1,
  data,
  ...
)
```

## Arguments

- |                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| r                      | A formula (as implemented in <a href="#">enw_formula()</a> ) describing the generative process used for expected incidence. This can use features defined by reference date as defined in <a href="#">metareference</a> as produced by <a href="#">enw_preprocess_data()</a> . By default this is set to use a daily random effect by group. This parameterisation is highly flexible and may not be the most appropriate choice when data is sparsely reported or reporting delays are substantial. In these settings an alternative could be a group-specific weekly random walk (specified as <code>rw(week, by = .group)</code> ). Setting to <code>~0</code> will produce an error as an expectation model is required. See <a href="#">enw_formula()</a> for details on formula syntax. |
| generation_time        | A numeric vector that sums to 1 and defaults to 1. Describes the weighting to apply to previous generations (i.e as part of a renewal equation). When set to 1 (the default) this corresponds to modelling the daily growth rate.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| observation            | A formula (as implemented in <a href="#">enw_formula()</a> ) describing the modifiers used to adjust expected observations. This can use features defined by reference date as defined in <a href="#">metareference</a> as produced by <a href="#">enw_preprocess_data()</a> . By default no modifiers are used but a common choice might be to adjust for the day of the week. Note that as the baseline is no modification, an intercept is always used and is set to 0. Set to <code>~0</code> to disable observation modifiers (internally converted to <code>~1</code> and flagged as inactive). See <a href="#">enw_formula()</a> for details on formula syntax.                                                                                                                        |
| latent_reporting_delay | A numeric vector that defaults to 1. Describes the weighting to apply to past and current latent expected observations (from most recent to least). This can be used both to convolve based on some assumed reporting delay and to rescale observations (by multiplying a probability mass function by some fraction) to                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

account ascertainment etc. A list of PMFs can be provided to allow for time-varying PMFs. This should be the same length as the modelled time period plus the length of the generation time if supplied.

data           Output from `enw_preprocess_data()`.

...           Additional parameters passed to `enw_add_metaobs_features()`. The same arguments as passed to `enw_preprocess_data()` should be used here.

Value

A list containing the supplied formulas, data passed into a list describing the models, a `data.frame` describing the priors used, and a function that takes the output data and priors and returns a function that can be used to sample from a tightened version of the prior distribution.

See Also

Model modules `.expectation_is_minimal()`, `enw_fit_opts()`, `enw_missing()`, `enw_obs()`, `enw_reference()`, `enw_report()`

Examples

```
enw_expectation(data = enw_example("preprocessed"))
```

---

|                 |                                                   |
|-----------------|---------------------------------------------------|
| enw_extend_date | <i>Extend a time series with additional dates</i> |
|-----------------|---------------------------------------------------|

---

Description

Extend a time series with additional dates. This is useful when extending the report dates of a time series to include future dates for nowcasting purposes or to include additional dates for backcasting when using a renewal process as the expectation model.

Usage

```
enw_extend_date(  
  metaobs,  
  days = 20,  
  direction = c("end", "start"),  
  timestep = "day"  
)
```

Arguments

|           |                                                                                                                                                                                                                              |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| metaobs   | A <code>data.frame</code> with a date column.                                                                                                                                                                                |
| days      | Number of days to add to the time series. Defaults to 20.                                                                                                                                                                    |
| direction | Should new dates be added at the beginning or end of the data. Default is "end" with "start" also available.                                                                                                                 |
| timestep  | The timestep to used. This can be a string ("day", "week") or a numeric whole number representing the number of days. Note that "month" is not currently supported in user-facing functions and will throw an error if used. |

**Value**

A data.table with the same columns as metaobs but with additional rows for each date in the range of date to date + days (or date - days if direction = "start"). An additional variable observed is added with a value of FALSE for all new dates and TRUE for all existing dates.

**See Also**

Preprocessing functions `enw_add_delay()`, `enw_add_max_reported()`, `enw_add_metaobs_features()`, `enw_assign_group()`, `enw_complete_dates()`, `enw_construct_data()`, `enw_filter_delay()`, `enw_filter_reference_dates()`, `enw_filter_reference_dates_by_report_start()`, `enw_filter_report_dates()`, `enw_flag_observed_observations()`, `enw_impute_na_observations()`, `enw_latest_data()`, `enw_metadata()`, `enw_metadata_delay()`, `enw_missing_reference()`, `enw_obs_at_delay()`, `enw_preprocess_data()`, `enw_reporting_triangle()`, `enw_reporting_triangle_to_long()`, `enw_retrospective()`

**Examples**

```
metaobs <- data.frame(date = as.Date("2021-01-01") + 0:4)
enw_extend_date(metaobs, days = 2)
enw_extend_date(metaobs, days = 2, direction = "start")
```

---

enw\_filter\_delay

---

*Filter observations to have a consistent maximum delay period*


---

**Description**

Filter observations to have a consistent maximum delay period

**Usage**

```
enw_filter_delay(obs, max_delay, timestep = "day")
```

**Arguments**

|           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| obs       | A data.frame containing at least the following variables: reference_date (index date of interest), report_date (report date for observations), and confirm (cumulative observations by reference and report date).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| max_delay | The maximum delay to model in the delay distribution, specified in units of the timestep (e.g., if timestep = "week", then max_delay = 3 means 3 weeks). If not specified the maximum observed delay is assumed to be the true maximum delay in the model. Otherwise, an integer greater than or equal to 1 can be specified. Observations with delays larger than the maximum delay will be dropped. If the specified maximum delay is too short, nowcasts can be biased as important parts of the true delay distribution are cut off. At the same time, computational cost scales non-linearly with this setting, so you want the maximum delay to be as long as necessary, but not much longer.<br>Steps to take to determine the maximum delay: |

- Consider what is realistic and relevant for your application.
- Check the proportion of observations reported (`prop_reported`) by delay in the `new_confirm` output of `enw_preprocess_obs`.
- Use `check_max_delay()` to check the coverage of a candidate `max_delay`.
- If in doubt, check if increasing the maximum delay noticeably changes the delay distribution or nowcasts as estimated by `epinowcast`. If it does, your maximum delay may still be too short.

Note that delays are zero indexed and so include the reference date and `max_delay - 1` other intervals (i.e. a `max_delay` of 1 corresponds to no delay).

**timestep** The timestep to used in the process model (i.e. the reference date model). This can be a string ("day", "week", "month") or a numeric whole number representing the number of days. If your data does not have this timestep then you may wish to make use of `enw_aggregate_cumulative()` to aggregate your data to the desired timestep.

### Value

A `data.frame` filtered so that dates by report are less than or equal the reference date plus the maximum delay.

### See Also

Preprocessing functions `enw_add_delay()`, `enw_add_max_reported()`, `enw_add_metaobs_features()`, `enw_assign_group()`, `enw_complete_dates()`, `enw_construct_data()`, `enw_extend_date()`, `enw_filter_reference_dates()`, `enw_filter_reference_dates_by_report_start()`, `enw_filter_report_dates()`, `enw_flag_observed_observations()`, `enw_impute_na_observations()`, `enw_latest_data()`, `enw_metadata()`, `enw_metadata_delay()`, `enw_missing_reference()`, `enw_obs_at_delay()`, `enw_preprocess_data()`, `enw_reporting_triangle()`, `enw_reporting_triangle_to_long()`, `enw_retrospective()`

### Examples

```
obs <- enw_example("preprocessed")$obs[[1]]
enw_filter_delay(obs, max_delay = 2)
```

---

`enw_filter_reference_dates`

*Filter by reference dates*

---

### Description

This is a helper function which allows users to filter datasets by reference date. This is useful, for example, when evaluating nowcast performance against fully observed data. Users may wish to combine this function with `enw_filter_report_dates()`. Note that by definition it is assumed that report dates must be equal or greater than the corresponding reference date (i.e a report cannot happen before the event being reported occurs). This means that this function will also filter out any report dates that are earlier than their corresponding reference date.

**Usage**

```
enw_filter_reference_dates(
  obs,
  earliest_date,
  include_days,
  latest_date,
  remove_days
)
```

**Arguments**

|                            |                                                                                                                                                                                                                                                                                             |
|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>obs</code>           | A <code>data.frame</code> ; must have <code>report_date</code> and <code>reference_date</code> columns.                                                                                                                                                                                     |
| <code>earliest_date</code> | earliest reference date to include in the data set                                                                                                                                                                                                                                          |
| <code>include_days</code>  | if <code>earliest_date</code> is not given, the number of reference dates to include, ending with the latest reference date included (determined by <code>latest_date</code> or <code>remove_days</code> ). For example, <code>include_days = 10</code> returns exactly 10 reference dates. |
| <code>latest_date</code>   | Date, the latest reference date to include in the returned dataset.                                                                                                                                                                                                                         |
| <code>remove_days</code>   | Integer, if <code>latest_date</code> is not given, the number of reference dates to remove, starting from the latest date included.                                                                                                                                                         |

**Details**

The `include_days` parameter filters to include exactly the specified number of most recent reference dates. For example, if the latest reference date is 2021-10-20 and `include_days = 10`, the filtered data will contain reference dates from 2021-10-11 to 2021-10-20 (10 days inclusive).

**Value**

A `data.table` filtered by report date

**See Also**

Preprocessing functions `enw_add_delay()`, `enw_add_max_reported()`, `enw_add_metaobs_features()`, `enw_assign_group()`, `enw_complete_dates()`, `enw_construct_data()`, `enw_extend_date()`, `enw_filter_delay()`, `enw_filter_reference_dates_by_report_start()`, `enw_filter_report_dates()`, `enw_flag_observed_observations()`, `enw_impute_na_observations()`, `enw_latest_data()`, `enw_metadata()`, `enw_metadata_delay()`, `enw_missing_reference()`, `enw_obs_at_delay()`, `enw_preprocess_data()`, `enw_reporting_triangle()`, `enw_reporting_triangle_to_long()`, `enw_retrospective()`

**Examples**

```
# Filter by date
enw_filter_reference_dates(
  germany_covid19_hosp,
  earliest_date = "2021-09-01",
  latest_date = "2021-10-01"
)
```

```
#
# Filter by days
enw_filter_reference_dates(
  germany_covid19_hosp,
  include_days = 10, remove_days = 10
)
```

---

```
enw_filter_reference_dates_by_report_start
```

*Filter reference dates that precede the earliest report date*

---

## Description

Removes observations where the reference\_date is earlier than the minimum report\_date within each group. Rows with missing reference\_date are retained. This is useful for ensuring that observations are only included from the first available report date onwards.

This function is typically called before `enw_add_incidence()` so that the incidence calculation starts from a valid reporting window. Without this step, reference dates that predate any report date produce spurious leading entries in the incidence output.

## Usage

```
enw_filter_reference_dates_by_report_start(obs, by = NULL, copy = TRUE)
```

## Arguments

|      |                                                                                                                                                                                                            |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| obs  | A data.frame with reference_date and report_date columns.                                                                                                                                                  |
| by   | A character vector describing the stratification of observations. This defaults to no grouping. This should be used when modelling multiple time series in order to identify them for downstream modelling |
| copy | Should obs be copied (default) or modified in place?                                                                                                                                                       |

## Value

A data.table filtered so that each reference\_date is on or after the minimum report\_date in its group. Rows with NA reference\_date are kept.

## See Also

Preprocessing functions `enw_add_delay()`, `enw_add_max_reported()`, `enw_add_metaobs_features()`, `enw_assign_group()`, `enw_complete_dates()`, `enw_construct_data()`, `enw_extend_date()`, `enw_filter_delay()`, `enw_filter_reference_dates()`, `enw_filter_report_dates()`, `enw_flag_observed_observations()`, `enw_impute_na_observations()`, `enw_latest_data()`, `enw_metadata()`, `enw_metadata_delay()`, `enw_missing_reference()`, `enw_obs_at_delay()`, `enw_preprocess_data()`, `enw_reporting_triangle()`, `enw_reporting_triangle_to_long()`, `enw_retrospective()`

**Examples**

```
library(data.table)
obs <- data.table(
  reference_date = as.IDate(c(
    "2021-10-01", "2021-10-02", "2021-10-03"
  )),
  report_date = as.IDate(c(
    "2021-10-02", "2021-10-02", "2021-10-03"
  ))
)
# The first row has reference_date before the minimum
# report_date, so it is removed
enw_filter_reference_dates_by_report_start(obs)
```

---

```
enw_filter_report_dates
```

*Filter by report dates*

---

**Description**

This is a helper function which allows users to create truncated data sets at past time points from a given larger data set. This is useful when evaluating nowcast performance against fully observed data. Users may wish to combine this function with [enw\\_filter\\_reference\\_dates\(\)](#).

**Usage**

```
enw_filter_report_dates(obs, latest_date, remove_days)
```

**Arguments**

|             |                                                                                                                     |
|-------------|---------------------------------------------------------------------------------------------------------------------|
| obs         | A data.frame; must have report_date and reference_date columns.                                                     |
| latest_date | Date, the latest report date to include in the returned dataset.                                                    |
| remove_days | Integer, if latest_date is not given, the number of report dates to remove, starting from the latest date included. |

**Value**

A data.table filtered by report date

**See Also**

Preprocessing functions [enw\\_add\\_delay\(\)](#), [enw\\_add\\_max\\_reported\(\)](#), [enw\\_add\\_metaobs\\_features\(\)](#), [enw\\_assign\\_group\(\)](#), [enw\\_complete\\_dates\(\)](#), [enw\\_construct\\_data\(\)](#), [enw\\_extend\\_date\(\)](#), [enw\\_filter\\_delay\(\)](#), [enw\\_filter\\_reference\\_dates\(\)](#), [enw\\_filter\\_reference\\_dates\\_by\\_report\\_start\(\)](#), [enw\\_flag\\_observed\\_observations\(\)](#), [enw\\_impute\\_na\\_observations\(\)](#), [enw\\_latest\\_data\(\)](#), [enw\\_metadata\(\)](#), [enw\\_metadata\\_delay\(\)](#), [enw\\_missing\\_reference\(\)](#), [enw\\_obs\\_at\\_delay\(\)](#), [enw\\_preprocess\\_data\(\)](#), [enw\\_reporting\\_triangle\(\)](#), [enw\\_reporting\\_triangle\\_to\\_long\(\)](#), [enw\\_retrospective\(\)](#)

## Examples

```
# Filter by date
enw_filter_report_dates(germany_covid19_hosp, latest_date = "2021-09-01")

# Filter by days
enw_filter_report_dates(germany_covid19_hosp, remove_days = 10)
```

---

enw\_fit\_opts

*Format model fitting options for use with stan*


---

## Description

Format model fitting options for use with stan

## Usage

```
enw_fit_opts(
  sampler = epinowcast::enw_sample,
  nowcast = TRUE,
  pp = FALSE,
  likelihood = TRUE,
  likelihood_aggregation = c("snapshots", "groups"),
  threads_per_chain = 1L,
  debug = FALSE,
  output_loglik = FALSE,
  sparse_design = FALSE,
  ...
)
```

## Arguments

- |                        |                                                                                                                                                                                                                                                                                                                                                                       |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sampler                | A function that creates an object that be used to extract posterior samples from the specified model. By default this is <a href="#">enw_sample()</a> which makes use of <a href="#">cmdstanr::sample()</a> .                                                                                                                                                         |
| nowcast                | Logical, defaults to TRUE. Should a nowcast be made using posterior predictions of the unobserved future reported notifications.                                                                                                                                                                                                                                      |
| pp                     | Logical, defaults to FALSE. Should posterior predictions be made for observed data. Useful for evaluating the performance of the model.                                                                                                                                                                                                                               |
| likelihood             | Logical, defaults to TRUE. Should the likelihood be included in the model                                                                                                                                                                                                                                                                                             |
| likelihood_aggregation | <p>Character string, aggregation over which stratify the likelihood when threads_per_chain is greater than 1; enforced by <a href="#">base::match.arg()</a>. Currently supported options:</p> <ul style="list-style-type: none"> <li>• "snapshots" which aggregates over report dates and groups (i.e the lowest level that observations are reported at),</li> </ul> |

- "groups" which aggregates across user defined groups.

Note that some model modules override this setting depending on model requirements. For example, the `enw_missing()` module model forces "groups" option. Generally, Users should typically want the default "snapshots" aggregation.

|                                |                                                                                                                                                                                                                                         |
|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>threads_per_chain</code> | Integer, defaults to 1. The number of threads to use within each MCMC chain. If this is greater than 1 then components of the likelihood will be calculated in parallel within each chain.                                              |
| <code>debug</code>             | Logical, defaults to FALSE. Should within model debug information be returned.                                                                                                                                                          |
| <code>output_loglik</code>     | Logical, defaults to FALSE. Should the log-likelihood be output. Disabling this will speed up fitting if evaluating the model fit is not required.                                                                                      |
| <code>sparse_design</code>     | Logical, defaults to FALSE. Should a sparse design matrices be used for all design matrices. This reduces memory requirements and may reduce computation time when fitting models with very sparse design matrices (90% or more zeros). |
| <code>...</code>               | Additional arguments to pass to the fitting function being used by <code>epinowcast()</code> . By default this will be <code>enw_sample()</code> and so <code>cmdstanr</code> options should be used.                                   |

### Value

A list containing the specified sampler function, data as a list specifying the fitting options to use, and additional arguments to pass to the sampler function when it is called.

### See Also

Model modules `.expectation_is_minimal()`, `enw_expectation()`, `enw_missing()`, `enw_obs()`, `enw_reference()`, `enw_report()`

### Examples

```
# Default options along with settings to pass to enw_sample
enw_fit_opts(iter_sampling = 1000, iter_warmup = 1000)
```

---

`enw_flag_observed_observations`  
*Flag observed observations*

---

### Description

Flags observations based on the 'confirm' column. If the 'observed' column does not exist, it is created. Observations are flagged as observed (TRUE) if 'confirm' is not NA.

### Usage

```
enw_flag_observed_observations(obs, copy = TRUE)
```

**Arguments**

|      |                                                                                    |
|------|------------------------------------------------------------------------------------|
| obs  | A data.frame with at least a have confirm column.                                  |
| copy | A logical; if TRUE (the default) creates a copy; otherwise, modifies obs in place. |

**Value**

A data.table with an additional column '.observed' indicating observed observations.

**See Also**

Preprocessing functions `enw_add_delay()`, `enw_add_max_reported()`, `enw_add_metaobs_features()`, `enw_assign_group()`, `enw_complete_dates()`, `enw_construct_data()`, `enw_extend_date()`, `enw_filter_delay()`, `enw_filter_reference_dates()`, `enw_filter_reference_dates_by_report_start()`, `enw_filter_report_dates()`, `enw_impute_na_observations()`, `enw_latest_data()`, `enw_metadata()`, `enw_metadata_delay()`, `enw_missing_reference()`, `enw_obs_at_delay()`, `enw_preprocess_data()`, `enw_reporting_triangle()`, `enw_reporting_triangle_to_long()`, `enw_retrospective()`

**Examples**

```
dt <- data.frame(id = 1:3, confirm = c(NA, 1, 2))
enw_flag_observed_observations(dt)
```

---

enw\_formula

*Define a model using a formula interface*


---

**Description**

This function allows models to be defined using a flexible formula interface that supports fixed effects, random effects (using [lme4](#) syntax), and random walks. The formula syntax builds on standard R formula notation and extends it with [lme4](#) style random effects and custom random walk terms. Users familiar with mixed models in [lme4](#) or [brms](#) will recognise the syntax. Note that the returned fixed effects design matrix is sparse and so the index supplied is required to link observations to the appropriate design matrix row.

**Usage**

```
enw_formula(formula, data, sparse = TRUE)
```

**Arguments**

|         |                                                                                                                                                                                                                                                                                                    |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| formula | A model formula that may use standard fixed effects, random effects using <a href="#">lme4</a> syntax (see <a href="#">re()</a> ), and random walks defined using the <a href="#">rw()</a> helper function. See the Details section below for a comprehensive explanation of the supported syntax. |
| data    | A data.frame of observations. It must include all variables used in the supplied formula.                                                                                                                                                                                                          |
| sparse  | Logical, defaults to TRUE. Should the fixed effects design matrix be sparsely defined.                                                                                                                                                                                                             |

## Details

### Formula syntax overview:

The formula interface supports three types of model components:

**Fixed effects:** Standard R formula syntax as used in `stats::lm()` and similar functions. For example:

- `~ 1`: intercept only
- `~ age_group`: intercept plus categorical predictor
- `~ age_group + location`: multiple predictors
- `~ 0 + age_group`: no intercept (contrasts)

**Random effects:** Uses `lme4` syntax with vertical bar notation. Random effects allow parameters to vary by group whilst sharing information across groups through partial pooling. Note that `epinowcast` assumes independent standard deviations for random effects rather than correlated random effects as supported by `lme4`. For example:

- `~ 1 + (1 | location)`: random intercepts by location
- `~ 1 + age_group + (1 | location)`: fixed age effect with random location intercepts
- `~ (age_group | location)`: random slopes for age within each location
- `~ (1 + week | location:month)`: random intercepts and week effects for each location-month combination (using interaction to create independent random effects per strata)

Interactions (e.g., `location:month`) can be used on the right-hand side of the vertical bar to specify independent random effects for each combination of the interacting variables.

See the `lme4` package documentation for more details on random effects syntax.

**Random walks:** Uses the `rw()` helper function to specify time-varying effects that evolve smoothly over time. For example:

- `~ rw(week)`: a random walk over weeks
- `~ rw(week, location)`: independent random walks for each location
- `~ rw(week, location)`: random walks with shared variance across locations (per-group variance is a planned extension)

**ARIMA residuals:** Uses the `arma()` helper to add an ARIMA(p, d, q) latent residual series to the linear predictor. Unlike `rw()`, the kernel that maps unit-normal shocks to the latent series depends on the autoregressive and moving-average parameters, so the term does not produce design-matrix columns; it carries lookup metadata that the Stan layer uses with the kernel from `inst/stan/functions/arma_kernel.stan`. For example:

- `~ arma(week)`: AR(1) on weekly residuals
- `~ arma(week, location, p = 2, d = 1, q = 1)`: ARIMA(2, 1, 1) driven by independent shocks per location, with  $\phi$ ,  $\theta$ , and  $\sigma$  shared across locations (per-group parameters are a planned extension)
- `arma(time, d = 1, p = 0, q = 0)` is equivalent to `rw(time)`

Convenience aliases match `brms`'s in-formula vocabulary:

- `ar(time, by, p)` is `arma(time, by, p = p, d = 0, q = 0)`
- `ma(time, by, q)` is `arma(time, by, p = 0, d = 0, q = q)`
- `arma(time, by, p, q)` is `arma(time, by, p = p, d = 0, q = q)`

These four types of effects can be combined in a single formula, for example: `~ 1 + age_group + (1 | location) + rw(week, location)` specifies fixed age effects, random location intercepts, and location-specific random walks over time.

### Turning off model components:

In `epinowcast` model specification functions (such as `enw_reference()`, `enw_report()`, `enw_expectation()`), formula arguments can be set to `~0` to disable that model component entirely. This is a package-specific convention. Note that when a formula is specified as `~0`, it is typically converted internally to `~1` (intercept only) to ensure valid model structure, but the component is flagged as inactive.

### How formulas map to the model:

The formula you specify controls which covariates and effects enter the linear predictor of the model. For instance, in the reference date model (`enw_reference()`), the formula determines how reporting delay parameters vary by covariates and groups. The formula is converted to design matrices: a fixed effects matrix (which may be sparse for computational efficiency) and a random effects matrix that defines the hierarchical structure.

## Value

A list containing the following:

- `formula`: The user supplied formula
- `parsed_formula`: The formula as parsed by `parse_formula()`
- `expanded_formula`: The flattened version of the formula with both user supplied terms and terms added for the user supplied complex model components.
- `fixed`: A list containing the fixed effect formula, sparse design matrix, and the index linking the design matrix with observations.
- `random`: A list containing the random effect formula, sparse design matrix, and the index linking the design matrix with random effects.

## References

For users new to formula syntax in R:

- **Fixed effects**: See `?formula` and the "Statistical Models in R" chapter of "An Introduction to R" at the URL:

<https://cran.r-project.org/doc/manuals/r-release/R-intro.html#Statistical-models-in-R>  
# nolint: line\_length\_linter

- **Random effects**: See the [lme4](#) package documentation and vignettes.
- **Mixed models**: Bates et al. (2015) "Fitting Linear Mixed-Effects Models Using lme4". Journal of Statistical Software, 67(1), 1-48. doi:10.18637/jss.v067.i01

## See Also

Functions used to help convert formulas into model designs `ar()`, `arma()`, `arma_terms()`, `arma()`, `as_string_formula()`, `construct_arma()`, `construct_gp()`, `construct_re()`, `construct_rw()`, `enw_manual_formula()`, `gp()`, `gp_terms()`, `ma()`, `parse_formula()`, `re()`, `remove_arma_terms()`, `remove_gp_terms()`, `remove_rw_terms()`, `rw()`, `rw_terms()`, `split_formula_to_terms()`

**Examples**

```

# Use meta data for references dates from the Germany COVID-19
# hospitalisation data.
obs <- enw_filter_report_dates(
  germany_covid19_hosp[location == "DE"],
  remove_days = 40
)
obs <- enw_filter_reference_dates(obs, include_days = 40)
pobs <- enw_preprocess_data(
  obs,
  by = c("age_group", "location"), max_delay = 20
)
data <- pobs$metareference[[1]]

# Intercept only
enw_formula(~1, data)

# Fixed effect
enw_formula(~ 1 + age_group, data)

# Random intercepts
enw_formula(~ 1 + (1 | age_group), data)

# Random walk
enw_formula(~ 1 + rw(week), data)

# Model with a random effect for age group and a random walk
enw_formula(~ 1 + (1 | age_group) + rw(week), data)

# Model defined without a sparse fixed effects design matrix
enw_formula(~1, data[1:20, ], sparse = FALSE)

# Model using an interaction in the right hand side of a random effect
# to specify an independent random effect per strata.
enw_formula(~ (1 + day | week:month), data = data)

```

---

enw\_formula\_as\_data\_list

*Format formula data for use with stan*


---

**Description**

Format formula data for use with stan

**Usage**

```
enw_formula_as_data_list(formula, prefix, drop_intercept = FALSE)
```

**Arguments**

|                |                                                                                                                                                                                 |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| formula        | The output of <code>enw_formula()</code> .                                                                                                                                      |
| prefix         | A character string indicating variable label to use as a prefix.                                                                                                                |
| drop_intercept | Logical, defaults to FALSE. Should the intercept be included as a fixed effect or excluded. This is used internally in model modules where an intercept must be present/absent. |

**Value**

A list defining the model formula. This includes:

- `prefix_fintercept`: Is an intercept present for the fixed effects design matrix.
- `prefix_fdesign`: The fixed effects design matrix
- `prefix_fnrow`: The number of rows of the fixed design matrix
- `prefix_findex`: The index linking design matrix rows to observations
- `prefix_fnindex`: The length of the index
- `prefix_fncol`: The number of columns (i.e effects) in the fixed effect design matrix (minus 1 if an intercept is present).
- `prefix_rdesign`: The random effects design matrix
- `prefix_rncol`: The number of columns (i.e random effects) in the random effect design matrix (minus 1 as the intercept is dropped).
- `prefix_arma_present`: 1 if the formula contains an `arma()` term, 0 otherwise.
- `prefix_arma_T`, `prefix_arma_G`: ARIMA series length and group count.
- `prefix_arma_p`, `prefix_arma_d`, `prefix_arma_q`: ARIMA orders.
- `prefix_arma_flat_idx`: per-observation column-major index into a (T x G) ARIMA residual matrix, used by Stan to gather residuals with `to_vector(eps)[flat_idx]`.
- `prefix_arma_n_obs`: length of the lookup vectors.

**See Also**

Functions used to help convert models into the format required for stan `enw_get_cache()`, `enw_model()`, `enw_pathfinder()`, `enw_priors_as_data_list()`, `enw_replace_priors()`, `enw_sample()`, `enw_set_cache()`, `enw_stan_to_r()`, `enw_unset_cache()`, `remove_profiling()`, `write_stan_files_no_profile()`

**Examples**

```
f <- enw_formula(~ 1 + (1 | cyl), mtcars)
enw_formula_as_data_list(f, "mtcars")

# A missing formula produces the default list
enw_formula_as_data_list(prefix = "missing")
```

---

|               |                                     |
|---------------|-------------------------------------|
| enw_get_cache | <i>Retrieve Stan cache location</i> |
|---------------|-------------------------------------|

---

**Description**

Retrieves the user set cache location for Stan models. This path can be set through the `enw_cache_location` function call. If no environmental variable is available the output from `tempdir()` will be returned.

**Usage**

```
enw_get_cache()
```

**Value**

A string representing the file path for the cache location

**See Also**

Functions used to help convert models into the format required for stan `enw_formula_as_data_list()`, `enw_model()`, `enw_pathfinder()`, `enw_priors_as_data_list()`, `enw_replace_priors()`, `enw_sample()`, `enw_set_cache()`, `enw_stan_to_r()`, `enw_unset_cache()`, `remove_profiling()`, `write_stan_files_no_profile()`

---

|              |                                                       |
|--------------|-------------------------------------------------------|
| enw_get_data | <i>Extract data from preprocessed nowcast objects</i> |
|--------------|-------------------------------------------------------|

---

**Description**

Extracts a named component from an `enw_preprocess_data()` or `epinowcast()` object. List columns are unwrapped automatically so you get the underlying `data.table` or value directly.

**Usage**

```
enw_get_data(x, name)
```

**Arguments**

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| <code>x</code>    | An <code>enw_preprocess_data</code> or <code>epinowcast</code> object. |
| <code>name</code> | Character string naming the component to extract.                      |

**Value**

The extracted component. For list columns this is the first element (typically a `data.table`); for scalar columns the value is returned as-is.

**See Also**

Utility functions [coerce\\_date\(\)](#), [coerce\\_dt\(\)](#), [date\\_to\\_numeric\\_modulus\(\)](#), [enw\\_rolling\\_sum\(\)](#), [get\\_internal\\_timestep\(\)](#), [is.Date\(\)](#), [stan\\_fns\\_as\\_string\(\)](#)

**Examples**

```
pobs <- enw_example("preprocessed_observations")
enw_get_data(pobs, "obs")
enw_get_data(pobs, "max_delay")
```

---

```
enw_impute_na_observations
```

*Impute NA observations*

---

**Description**

Imputes NA values in the 'confirm' column. NA values are replaced with the last available observation or 0.

**Usage**

```
enw_impute_na_observations(obs, by = NULL, copy = TRUE)
```

**Arguments**

|      |                                                                                    |
|------|------------------------------------------------------------------------------------|
| obs  | A data.frame with at least 'confirm' and 'reference_date' columns.                 |
| by   | A character vector of column names to group by. Defaults to an empty vector.       |
| copy | A logical; if TRUE (the default) creates a copy; otherwise, modifies obs in place. |

**Value**

A data.table with imputed 'confirm' column where NA values have been replaced with zero.

**See Also**

Preprocessing functions [enw\\_add\\_delay\(\)](#), [enw\\_add\\_max\\_reported\(\)](#), [enw\\_add\\_metaobs\\_features\(\)](#), [enw\\_assign\\_group\(\)](#), [enw\\_complete\\_dates\(\)](#), [enw\\_construct\\_data\(\)](#), [enw\\_extend\\_date\(\)](#), [enw\\_filter\\_delay\(\)](#), [enw\\_filter\\_reference\\_dates\(\)](#), [enw\\_filter\\_reference\\_dates\\_by\\_report\\_start\(\)](#), [enw\\_filter\\_report\\_dates\(\)](#), [enw\\_flag\\_observed\\_observations\(\)](#), [enw\\_latest\\_data\(\)](#), [enw\\_metadata\(\)](#), [enw\\_metadata\\_delay\(\)](#), [enw\\_missing\\_reference\(\)](#), [enw\\_obs\\_at\\_delay\(\)](#), [enw\\_preprocess\\_data\(\)](#), [enw\\_reporting\\_triangle\(\)](#), [enw\\_reporting\\_triangle\\_to\\_long\(\)](#), [enw\\_retrospective\(\)](#)

**Examples**

```
dt <- data.frame(
  id = 1:3, confirm = c(NA, 1, 2),
  reference_date = as.Date("2021-01-01")
)
enw_impute_na_observations(dt)
```

---

`enw_incidence_to_linelist`*Convert Aggregate Counts (Incidence) to a Line List*

---

### Description

This function takes a `data.table` of aggregate counts or something coercible to a `data.table` (such as a `data.frame`) and converts it to a line list where each row represents a case.

### Usage

```
enw_incidence_to_linelist(  
  obs,  
  reference_date = "reference_date",  
  report_date = "report_date"  
)
```

### Arguments

|                             |                                                                                                                                          |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <code>obs</code>            | An object coercible to a <code>data.table</code> (such as a <code>data.frame</code> ) which must have a <code>new_confirm</code> column. |
| <code>reference_date</code> | A character string of the variable name to use for the <code>reference_date</code> in the line list. The default is "reference_date".    |
| <code>report_date</code>    | A character string of the variable name to use for the <code>report_date</code> in the line list. The default is "report_date".          |

### Value

A `data.table` with the following variables: `id`, `reference_date`, `report_date`, and any other variables in the `obs` object. Rows in `obs` will be duplicated based on the `new_confirm` column. `reference_date` and `report_date` may be renamed if `reference_date` and `report_date` are supplied.

### See Also

Data converters [enw\\_add\\_cumulative\(\)](#), [enw\\_add\\_incidence\(\)](#), [enw\\_aggregate\\_cumulative\(\)](#), [enw\\_linelist\\_to\\_incidence\(\)](#)

### Examples

```
incidence <- enw_filter_reference_dates_by_report_start(  
  germany_covid19_hosp  
)  
incidence <- enw_add_incidence(incidence)  
incidence <- enw_filter_reference_dates(  
  incidence[location == "DE"], include_days = 10  
)  
enw_incidence_to_linelist(incidence, reference_date = "onset_date")
```

---

|                 |                                                             |
|-----------------|-------------------------------------------------------------|
| enw_latest_data | <i>Filter observations to the latest available reported</i> |
|-----------------|-------------------------------------------------------------|

---

### Description

Filter observations for the latest available reported data for each reference date. Note this is not the same as filtering for the maximum report date in all cases as data may only be updated up to some maximum number of days.

### Usage

```
enw_latest_data(obs)
```

### Arguments

obs                      A `data.frame`; must have `report_date` and `reference_date` columns.

### Value

A `data.table` of observations filtered for the latest available data for each reference date.

### See Also

Preprocessing functions `enw_add_delay()`, `enw_add_max_reported()`, `enw_add_metaobs_features()`, `enw_assign_group()`, `enw_complete_dates()`, `enw_construct_data()`, `enw_extend_date()`, `enw_filter_delay()`, `enw_filter_reference_dates()`, `enw_filter_reference_dates_by_report_start()`, `enw_filter_report_dates()`, `enw_flag_observed_observations()`, `enw_impute_na_observations()`, `enw_metadata()`, `enw_metadata_delay()`, `enw_missing_reference()`, `enw_obs_at_delay()`, `enw_preprocess_data()`, `enw_reporting_triangle()`, `enw_reporting_triangle_to_long()`, `enw_retrospective()`

### Examples

```
# Filter for latest reported data
enw_latest_data(germany_covid19_hosp)
```

---

|                           |                                                            |
|---------------------------|------------------------------------------------------------|
| enw_linelist_to_incidence | <i>Convert a Line List to Aggregate Counts (Incidence)</i> |
|---------------------------|------------------------------------------------------------|

---

### Description

This function takes a line list (i.e. tabular data where each row represents a case) and aggregates to a count (`new_confirm`) of cases by user-specified `reference_dates` and `report_dates`. This enables the use of `enw_preprocess_data()` and other `epinowcast()` preprocessing functions.

**Usage**

```
enw_linelist_to_incidence(
  linelist,
  reference_date = "reference_date",
  report_date = "report_date",
  by = NULL,
  max_delay,
  completion_beyond_max_report = FALSE,
  copy = TRUE
)
```

**Arguments**

|                                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>linelist</code>                     | An object coercible to a <code>data.table</code> (such as a <code>data.frame</code> ) where each row represents a case. Must contain at least two date variables or variables that can be coerced to dates.                                                                                                                                                                                                                                                                                                                                 |
| <code>reference_date</code>               | A date or a variable that can be coerced to a date that represents the date of interest for the case. For example, if the <code>reference_date</code> is the date of symptom onset then the <code>new_confirm</code> will be the number of new cases reported (based on <code>report_date</code> ) on each day that had onset on that day. The default is <code>"reference_date"</code> .                                                                                                                                                   |
| <code>report_date</code>                  | A date or a variable that can be coerced to a date that represents the date the case was reported. The default is <code>"report_date"</code> .                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>by</code>                           | A character vector of variables to also aggregate by (i.e. as well as using the <code>reference_date</code> and <code>report_date</code> ). If not supplied then the function will aggregate by just the <code>reference_date</code> and <code>report_date</code> .                                                                                                                                                                                                                                                                         |
| <code>max_delay</code>                    | The maximum delay (in days) between the <code>reference_date</code> and the <code>report_date</code> . If not supplied then the function will use the maximum observed delay. Note that this function operates before timestep conversion, so <code>max_delay</code> is always in days here. If the <code>max_delay</code> is less than the maximum number of days between the <code>reference_date</code> and the <code>report_date</code> in the <code>linelist</code> then the function will use this value instead and inform the user. |
| <code>completion_beyond_max_report</code> | Logical, should entries be completed beyond the maximum date found in the data? Default: <code>FALSE</code>                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <code>copy</code>                         | Should obs be copied (default) or modified in place?                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

**Value**

A `data.table` with the following variables: `reference_date`, `report_date`, `new_confirm`, `confirm`, `delay`, and any variables specified in `by`.

**See Also**

Data converters [enw\\_add\\_cumulative\(\)](#), [enw\\_add\\_incidence\(\)](#), [enw\\_aggregate\\_cumulative\(\)](#), [enw\\_incidence\\_to\\_linelist\(\)](#)

## Examples

```
linelist <- data.frame(
  onset_date = as.Date(c("2021-01-02", "2021-01-03", "2021-01-02")),
  report_date = as.Date(c("2021-01-03", "2021-01-05", "2021-01-04"))
)
enw_linelist_to_incidence(linelist, reference_date = "onset_date")

# Specify a custom maximum delay and allow completion beyond the maximum
# observed delay
enw_linelist_to_incidence(
  linelist, reference_date = "onset_date", max_delay = 5,
  completion_beyond_max_report = TRUE
)
```

---

|                    |                                                               |
|--------------------|---------------------------------------------------------------|
| enw_manual_formula | <i>Define a model manually using fixed and random effects</i> |
|--------------------|---------------------------------------------------------------|

---

## Description

For most typical use cases `enw_formula()` should provide sufficient flexibility to allow models to be defined. However, there may be some instances where more manual model specification is required. This function supports this by allowing the user to supply vectors of fixed, random, and customised random effects (where they are not first treated as fixed effect terms). Prior to 1.0.0 this was the main interface for specifying models and it is still used internally to handle some parts of the model specification process.

## Usage

```
enw_manual_formula(
  data,
  fixed = NULL,
  random = NULL,
  custom_random = NULL,
  no_contrasts = FALSE,
  add_intercept = TRUE
)
```

## Arguments

|               |                                                                                                                                                                                                   |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| data          | A <code>data.frame</code> of observations. It must include all variables used in the supplied formula.                                                                                            |
| fixed         | A character vector of fixed effects.                                                                                                                                                              |
| random        | A character vector of random effects. Random effects specified here will be added to the fixed effects.                                                                                           |
| custom_random | A vector of random effects. Random effects added here will not be added to the vector of fixed effects. This can be used to random effects for fixed effects that only have a partial name match. |

- no\_contrasts Logical, defaults to FALSE. TRUE means that no variable uses contrast. Alternatively a character vector of variables can be supplied indicating which variables should not have contrasts.
- add\_intercept Logical, defaults to FALSE. Should an intercept be added to the fixed effects.

Value

A list specifying the fixed effects (formula, design matrix, and design matrix index), and random effects (formula and design matrix).

See Also

Functions used to help convert formulas into model designs [ar\(\)](#), [arima\(\)](#), [arima\\_terms\(\)](#), [arma\(\)](#), [as\\_string\\_formula\(\)](#), [construct\\_arima\(\)](#), [construct\\_gp\(\)](#), [construct\\_re\(\)](#), [construct\\_rw\(\)](#), [enw\\_formula\(\)](#), [gp\(\)](#), [gp\\_terms\(\)](#), [ma\(\)](#), [parse\\_formula\(\)](#), [re\(\)](#), [remove\\_arima\\_terms\(\)](#), [remove\\_gp\\_terms\(\)](#), [remove\\_rw\\_terms\(\)](#), [rw\(\)](#), [rw\\_terms\(\)](#), [split\\_formula\\_to\\_terms\(\)](#)

Examples

```
data <- enw_example("prep")$metareference[[1]]
enw_manual_formula(data, fixed = "week", random = "day_of_week")
```

---

|              |                                       |
|--------------|---------------------------------------|
| enw_metadata | <i>Extract metadata from raw data</i> |
|--------------|---------------------------------------|

---

Description

Extract metadata from raw data, either by reference or by report date. For the target date chosen (reference or report), `confirm`, `max_confirm``, and `cum_prop_reported`` are dropped and the first observation for each group and date is retained.

Usage

```
enw_metadata(obs, target_date = c("reference_date", "report_date"))
```

Arguments

- obs A `data.frame` or `data.table` with columns: `reference_date` and / or `report_date`; at least one must be provided, `.group`, a grouping column and a date, a [Date](#) column.
- target\_date A character string, either "reference\_date" or "report\_date". The column corresponding to this string will be used as the target date for metadata extraction.

**Value**

A data.table with columns:

- date, a [Date](#) column
- .group, a grouping column

and the first observation for each group and date. The data.table is sorted by .group and date.

**See Also**

Preprocessing functions [enw\\_add\\_delay\(\)](#), [enw\\_add\\_max\\_reported\(\)](#), [enw\\_add\\_metaobs\\_features\(\)](#), [enw\\_assign\\_group\(\)](#), [enw\\_complete\\_dates\(\)](#), [enw\\_construct\\_data\(\)](#), [enw\\_extend\\_date\(\)](#), [enw\\_filter\\_delay\(\)](#), [enw\\_filter\\_reference\\_dates\(\)](#), [enw\\_filter\\_reference\\_dates\\_by\\_report\\_start\(\)](#), [enw\\_filter\\_report\\_dates\(\)](#), [enw\\_flag\\_observed\\_observations\(\)](#), [enw\\_impute\\_na\\_observations\(\)](#), [enw\\_latest\\_data\(\)](#), [enw\\_metadata\\_delay\(\)](#), [enw\\_missing\\_reference\(\)](#), [enw\\_obs\\_at\\_delay\(\)](#), [enw\\_preprocess\\_data\(\)](#), [enw\\_reporting\\_triangle\(\)](#), [enw\\_reporting\\_triangle\\_to\\_long\(\)](#), [enw\\_retrospective\(\)](#)

**Examples**

```
obs <- data.frame(
  reference_date = as.Date("2021-01-01"),
  report_date = as.Date("2022-01-01"), x = 1:10
)
enw_metadata(obs, target_date = "reference_date")
```

---

|                    |                                                                     |
|--------------------|---------------------------------------------------------------------|
| enw_metadata_delay | <i>Calculate reporting delay metadata for a given maximum delay</i> |
|--------------------|---------------------------------------------------------------------|

---

**Description**

Calculate delay metadata based on the supplied maximum delay and independent of other metadata or date indexing. These data are meant to be used in conjunction with metadata on the date of reference. Users can build additional features with this data.frame or regenerate it using this function in the output of [enw\\_preprocess\\_data\(\)](#).

**Usage**

```
enw_metadata_delay(max_delay = 20, breaks = 4, timestep = "day")
```

**Arguments**

|           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| max_delay | The maximum delay to model in the delay distribution, specified in units of the timestep (e.g., if timestep = "week", then max_delay = 3 means 3 weeks). If not specified the maximum observed delay is assumed to be the true maximum delay in the model. Otherwise, an integer greater than or equal to 1 can be specified. Observations with delays larger than the maximum delay will be dropped. If the specified maximum delay is too short, nowcasts can be biased as important |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

parts of the true delay distribution are cut off. At the same time, computational cost scales non-linearly with this setting, so you want the maximum delay to be as long as necessary, but not much longer.

Steps to take to determine the maximum delay:

- Consider what is realistic and relevant for your application.
- Check the proportion of observations reported (`prop_reported`) by delay in the `new_confirm` output of `enw_preprocess_obs`.
- Use `check_max_delay()` to check the coverage of a candidate `max_delay`.
- If in doubt, check if increasing the maximum delay noticeably changes the delay distribution or nowcasts as estimated by `epinowcast`. If it does, your maximum delay may still be too short.

Note that delays are zero indexed and so include the reference date and `max_delay - 1` other intervals (i.e. a `max_delay` of 1 corresponds to no delay).

|                       |                                                                                                                                                                                                                              |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>breaks</code>   | Numeric, defaults to 4. The number of breaks to use when constructing a categorised version of numeric delays.                                                                                                               |
| <code>timestep</code> | The timestep to used. This can be a string ("day", "week") or a numeric whole number representing the number of days. Note that "month" is not currently supported in user-facing functions and will throw an error if used. |

## Value

A data.frame of delay metadata. This includes:

- `delay`: The numeric delay from reference date to report.
- `delay_cat`: The categorised delay. This may be useful for model building.
- `delay_week`: The numeric week since the delay was reported. This again may be useful for model building.
- `delay_head`: A logical variable defining if the delay is in the lower 25% of the potential delays. This may be particularly useful when building models that assume a parametric distribution in order to increase the weight of the head of the reporting distribution in a pragmatic way.
- `delay_tail`: A logical variable defining if the delay is in the upper 75% of the potential delays. This may be particularly useful when building models that assume a parametric distribution in order to increase the weight of the tail of the reporting distribution in a pragmatic way.

## See Also

Preprocessing functions `enw_add_delay()`, `enw_add_max_reported()`, `enw_add_metaobs_features()`, `enw_assign_group()`, `enw_complete_dates()`, `enw_construct_data()`, `enw_extend_date()`, `enw_filter_delay()`, `enw_filter_reference_dates()`, `enw_filter_reference_dates_by_report_start()`, `enw_filter_report_dates()`, `enw_flag_observed_observations()`, `enw_impute_na_observations()`, `enw_latest_data()`, `enw_metadata()`, `enw_missing_reference()`, `enw_obs_at_delay()`, `enw_preprocess_data()`, `enw_reporting_triangle()`, `enw_reporting_triangle_to_long()`, `enw_retrospective()`

## Examples

```
enw_metadata_delay(max_delay = 20, breaks = 4)
```

---

|             |                                            |
|-------------|--------------------------------------------|
| enw_missing | <i>Missing reference data model module</i> |
|-------------|--------------------------------------------|

---

## Description

Missing reference data model module

## Usage

```
enw_missing(formula = ~1, data)
```

## Arguments

|         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| formula | A formula (as implemented in <a href="#">enw_formula()</a> ) describing the missing data proportion on the logit scale by reference date. This can use features defined by reference date as defined in <a href="#">metareference</a> as produced by <a href="#">enw_preprocess_data()</a> . Set to <code>~0</code> to disable the missing data model when no missingness is present (the model is not estimated in this case). When a model is specified, an intercept is always required. See <a href="#">enw_formula()</a> for details on formula syntax. |
| data    | Output from <a href="#">enw_preprocess_data()</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

## Value

A list containing the supplied formulas, data passed into a list describing the models, a `data.frame` describing the priors used, and a function that takes the output data and priors and returns a function that can be used to sample from a tightened version of the prior distribution.

## See Also

Model modules [.expectation\\_is\\_minimal\(\)](#), [enw\\_expectation\(\)](#), [enw\\_fit\\_opts\(\)](#), [enw\\_obs\(\)](#), [enw\\_reference\(\)](#), [enw\\_report\(\)](#)

## Examples

```
# Missingness model with a fixed intercept only
enw_missing(data = enw_example("preprocessed"))

# No missingness model specified
enw_missing(~0, data = enw_example("preprocessed"))
```

---

enw\_missing\_reference *Extract reports with missing reference dates*

---

## Description

Returns reports with missing reference dates as well as calculating the proportion of reports for a given reference date that were missing.

## Usage

```
enw_missing_reference(obs)
```

## Arguments

|     |                                                                                                                                                                            |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| obs | A data.frame as produced by <a href="#">enw_add_incidence()</a> . Must contain the following variables: report_date, reference_date, .group, and confirm, and new_confirm. |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Value

A data.table of missing counts and proportions by report date and group.

## See Also

Preprocessing functions [enw\\_add\\_delay\(\)](#), [enw\\_add\\_max\\_reported\(\)](#), [enw\\_add\\_metaobs\\_features\(\)](#), [enw\\_assign\\_group\(\)](#), [enw\\_complete\\_dates\(\)](#), [enw\\_construct\\_data\(\)](#), [enw\\_extend\\_date\(\)](#), [enw\\_filter\\_delay\(\)](#), [enw\\_filter\\_reference\\_dates\(\)](#), [enw\\_filter\\_reference\\_dates\\_by\\_report\\_start\(\)](#), [enw\\_filter\\_report\\_dates\(\)](#), [enw\\_flag\\_observed\\_observations\(\)](#), [enw\\_impute\\_na\\_observations\(\)](#), [enw\\_latest\\_data\(\)](#), [enw\\_metadata\(\)](#), [enw\\_metadata\\_delay\(\)](#), [enw\\_obs\\_at\\_delay\(\)](#), [enw\\_preprocess\\_data\(\)](#), [enw\\_reporting\\_triangle\(\)](#), [enw\\_reporting\\_triangle\\_to\\_long\(\)](#), [enw\\_retrospective\(\)](#)

## Examples

```
obs <- data.frame(
  report_date = c("2021-10-01", "2021-10-03"), reference_date = "2021-10-01",
  confirm = 1
)
obs <- rbind(
  obs,
  data.frame(report_date = "2021-10-04", reference_date = NA, confirm = 4)
)
obs <- enw_complete_dates(obs)
obs <- enw_assign_group(obs)
obs <- enw_filter_reference_dates_by_report_start(obs)
obs <- enw_add_incidence(obs)
enw_missing_reference(obs)
```

enw\_model

*Load and compile the nowcasting model***Description**

Load and compile the nowcasting model

**Usage**

```
enw_model(
  model = system.file("stan", "epinowcast.stan", package = "epinowcast"),
  include = system.file("stan", package = "epinowcast"),
  compile = TRUE,
  threads = TRUE,
  profile = FALSE,
  target_dir = epinowcast::enw_get_cache(),
  stanc_options = list(),
  cpp_options = list(),
  verbose = TRUE,
  ...
)
```

**Arguments**

|               |                                                                                                                                                                                                                                                                                                                                                |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| model         | A character string indicating the path to the model. If not supplied the package default model is used.                                                                                                                                                                                                                                        |
| include       | A character string specifying the path to any stan files to include in the model. If missing the package default is used.                                                                                                                                                                                                                      |
| compile       | Logical, defaults to TRUE. Should the model be loaded and compiled using <code>cmdstanr::cmdstan_model()</code> .                                                                                                                                                                                                                              |
| threads       | Logical, defaults to TRUE. Should the model compile with support for multi-thread support in chain. Note that setting this will produce a warning that <code>threads_to_chain</code> is set and ignored. Changing this to FALSE is not expected to yield any performance benefits even when not using multithreading and thus not recommended. |
| profile       | Logical, defaults to FALSE. Should the model be profiled? For more on profiling see the <a href="#">cmdstanr documentation</a> . # nolint                                                                                                                                                                                                      |
| target_dir    | The path to a directory in which the manipulated .stan files without profiling statements should be stored. To avoid overriding of the original .stan files, this should be different from the directory of the original model and the <code>include_paths</code> .                                                                            |
| stanc_options | A list of options to pass to the <code>stanc_options</code> of <code>cmdstanr::cmdstan_model()</code> . By default nothing is passed but potentially users may wish to pass optimisation flags for example. See the documentation for <code>cmdstanr::cmdstan_model()</code> for further details.                                              |

|             |                                                                                                                                                                                                                                                                                                                                          |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| cpp_options | A list of options to pass to the cpp_options of <code>cmdstanr::cmdstan_model()</code> . By default nothing is passed but potentially users may wish to pass optimisation flags for example. See the documentation for <code>cmdstanr::cmdstan_model()</code> for further details. Note that the threads argument replaces stan_threads. |
| verbose     | Logical, defaults to TRUE. Should verbose messages be shown.                                                                                                                                                                                                                                                                             |
| ...         | Additional arguments passed to <code>cmdstanr::cmdstan_model()</code> .                                                                                                                                                                                                                                                                  |

**Value**

A cmdstanr model.

**See Also**

Functions used to help convert models into the format required for stan `enw_formula_as_data_list()`, `enw_get_cache()`, `enw_pathfinder()`, `enw_priors_as_data_list()`, `enw_replace_priors()`, `enw_sample()`, `enw_set_cache()`, `enw_stan_to_r()`, `enw_unset_cache()`, `remove_profiling()`, `write_stan_files_no_profile()`

**Examples**

```
mod <- enw_model()
```

---

|                     |                                                             |
|---------------------|-------------------------------------------------------------|
| enw_nowcast_samples | <i>Extract posterior samples for the nowcast prediction</i> |
|---------------------|-------------------------------------------------------------|

---

**Description**

A generic wrapper around `posterior::draws_df()` with opinionated defaults to extract the posterior samples for the nowcast ("pp\_inf\_obs" from the stan code). The functionality of this function can be used directly on the output of `epinowcast()` using the supplied `summary.epinowcast()` method.

**Usage**

```
enw_nowcast_samples(fit, obs, max_delay = NULL, timestep = "day")
```

**Arguments**

|     |                                                                                                                                                                                                                                                                                                                                                                              |
|-----|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| fit | A cmdstanr fit object.                                                                                                                                                                                                                                                                                                                                                       |
| obs | An observation data.frame containing reference_date columns of the same length as the number of rows in the posterior and the most up to date observation for each date. This is used to align the posterior with the observations. The easiest source of this data is the output of latest output of <code>enw_preprocess_data()</code> or <code>enw_latest_data()</code> . |

|           |                                                                                                                                                                                                                                                                                                                              |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| max_delay | Maximum delay to which nowcasts should be extracted, in units of the timestep used during preprocessing. Must be equal (default) or larger than the modelled maximum delay. If it is larger, then nowcasts for unmodelled dates are added by assuming that case counts beyond the modelled maximum delay are fully observed. |
| timestep  | The timestep to used. This can be a string ("day", "week") or a numeric whole number representing the number of days. Note that "month" is not currently supported in user-facing functions and will throw an error if used.                                                                                                 |

**Value**

A `data.frame` of posterior samples for the nowcast prediction. This uses observed data where available and the posterior prediction where not.

**See Also**

Functions used for postprocessing of model fits `.check_primarycensored()`, `.delay_draw_columns()`, `.discretise_parametric_pmf()`, `build_ord_obs()`, `enw_add_latest_obs_to_nowcast()`, `enw_nowcast_summary()`, `enw_posterior()`, `enw_posterior_delay()`, `enw_pp_summary()`, `enw_quantiles_to_long()`, `enw_summarise_samples()`, `subset_obs()`

**Examples**

```
fit <- enw_example("nowcast")
enw_nowcast_samples(
  fit$fit[[1]],
  fit$latest[[1]],
  fit$max_delay,
  "day"
)
```

---

|                     |                                                   |
|---------------------|---------------------------------------------------|
| enw_nowcast_summary | <i>Summarise the posterior nowcast prediction</i> |
|---------------------|---------------------------------------------------|

---

**Description**

A generic wrapper around `enw_posterior()` with opinionated defaults to extract the posterior prediction for the nowcast ("pp\_inf\_obs" from the stan code). The functionality of this function can be used directly on the output of `epinowcast()` using the supplied `summary.epinowcast()` method.

**Usage**

```
enw_nowcast_summary(
  fit,
  obs,
  max_delay = NULL,
  timestep = "day",
```

```

    probs = c(0.05, 0.2, 0.35, 0.5, 0.65, 0.8, 0.95)
  )

```

### Arguments

|                        |                                                                                                                                                                                                                                                                                                                                                                                    |
|------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>fit</code>       | A cmdstanr fit object.                                                                                                                                                                                                                                                                                                                                                             |
| <code>obs</code>       | An observation data.frame containing reference_date columns of the same length as the number of rows in the posterior and the most up to date observation for each date. This is used to align the posterior with the observations. The easiest source of this data is the output of latest output of <a href="#">enw_preprocess_data()</a> or <a href="#">enw_latest_data()</a> . |
| <code>max_delay</code> | Maximum delay to which nowcasts should be summarised, in units of the timestep used during preprocessing. Must be equal (default) or larger than the modelled maximum delay. If it is larger, then nowcasts for unmodelled dates are added by assuming that case counts beyond the modelled maximum delay are fully observed.                                                      |
| <code>timestep</code>  | The timestep to used. This can be a string ("day", "week") or a numeric whole number representing the number of days. Note that "month" is not currently supported in user-facing functions and will throw an error if used.                                                                                                                                                       |
| <code>probs</code>     | A vector of numeric probabilities to produce quantile summaries for. By default these are the 5%, 20%, 80%, and 95% quantiles which are also the minimum set required for plotting functions to work.                                                                                                                                                                              |

### Value

A data.frame summarising the model posterior nowcast prediction. This uses observed data where available and the posterior prediction where not.

### See Also

[summary.epinowcast\(\)](#)

Functions used for postprocessing of model fits [.check\\_primarycensored\(\)](#), [.delay\\_draw\\_columns\(\)](#), [.discretise\\_parametric\\_pmf\(\)](#), [build\\_ord\\_obs\(\)](#), [enw\\_add\\_latest\\_obs\\_to\\_nowcast\(\)](#), [enw\\_nowcast\\_samples\(\)](#), [enw\\_posterior\(\)](#), [enw\\_posterior\\_delay\(\)](#), [enw\\_pp\\_summary\(\)](#), [enw\\_quantiles\\_to\\_long\(\)](#), [enw\\_summarise\\_samples\(\)](#), [subset\\_obs\(\)](#)

### Examples

```

fit <- enw_example("nowcast")
enw_nowcast_summary(
  fit$fit[[1]],
  fit$latest[[1]],
  fit$max_delay
)

```

enw\_obs

*Setup observation model and data***Description**

Setup observation model and data

**Usage**

```
enw_obs(
  family = c("negbin", "negbin1d", "poisson"),
  observation_indicator = NULL,
  delay_only = FALSE,
  data
)
```

**Arguments**

- |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| family                | Character string, the observation model to use in the likelihood; enforced by <code>base::match.arg()</code> . By default this is a negative binomial with a quadratic mean-variance relationship ("negbin"). Negative binomial with a linear mean-variance relationship ("negbin1d") and Poisson ("poisson") are also available. This is ignored when <code>delay_only = TRUE</code> , in which case the delay-only multinomial likelihood is used regardless of family; supplying a family alongside <code>delay_only = TRUE</code> emits a warning.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| observation_indicator | A character string, the name of the column in the data that indicates whether an observation is observed or not (using a logical variable) and therefore whether or not it should be used in the likelihood. This variable should be present in the data input to <code>enw_preprocess_data()</code> . It can be generated using <code>flag_observation</code> in <code>enw_complete_dates()</code> or it can be created directly using <code>enw_flag_observed_observations()</code> . If either of these approaches are used then the variable will be name <code>.observed</code> . Default is <code>NULL</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| delay_only            | Logical, defaults to <code>FALSE</code> . If <code>TRUE</code> , fit only the reporting-delay distribution conditional on the known per-reference-date totals, treating those totals as fixed truth. The latent process and per-cell observation model are replaced by a (truncated) multinomial over the reported cells of each reference date, so <code>family</code> is ignored. With final retrospective totals this is the plain multinomial; with running totals observed up to some horizon the renormalisation over the observed delay range gives the truncated multinomial. An <code>observation_indicator</code> is supported and renormalises over all delays up to the observation cutoff. Because the known totals override the expected observations, the latent process is inert; <code>epinowcast()</code> therefore minimises the expectation automatically, so a delay-only fit is just <code>epinowcast(data, obs = enw_obs(delay_only = TRUE, data = data))</code> with no separate expectation module to configure. This mode estimates delays and does not nowcast (refit with the full model for a nowcast). Not compatible with the missing reference model. See the delay estimation vignette for |

a worked example. Based on the conditional delay likelihood of Kalbfleisch and Lawless ([doi:10.1080/01621459.1989.10478780](https://doi.org/10.1080/01621459.1989.10478780)) and Höhle and an der Heiden ([doi:10.1111/biom.12194](https://doi.org/10.1111/biom.12194)).

data      Output from `enw_preprocess_data()`.

### Value

A list as required by stan.

### See Also

Model modules `.expectation_is_minimal()`, `enw_expectation()`, `enw_fit_opts()`, `enw_missing()`, `enw_reference()`, `enw_report()`

### Examples

```
enw_obs(data = enw_example("preprocessed"))
# Delay-only model conditional on known totals
enw_obs(delay_only = TRUE, data = enw_example("preprocessed"))
```

---

|                  |                                                             |
|------------------|-------------------------------------------------------------|
| enw_obs_at_delay | <i>Extract latest observations at a given maximum delay</i> |
|------------------|-------------------------------------------------------------|

---

### Description

Filter observations to a maximum delay and then extract the latest observations. This is useful for model evaluation where you want to assess performance against the data as the model would have seen it.

### Usage

```
enw_obs_at_delay(obs, max_delay, timestep = "day")
```

### Arguments

|           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| obs       | A data.frame containing at least the following variables: reference_date (index date of interest), report_date (report date for observations), and confirm (cumulative observations by reference and report date).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| max_delay | The maximum delay to model in the delay distribution, specified in units of the timestep (e.g., if timestep = "week", then max_delay = 3 means 3 weeks). If not specified the maximum observed delay is assumed to be the true maximum delay in the model. Otherwise, an integer greater than or equal to 1 can be specified. Observations with delays larger than the maximum delay will be dropped. If the specified maximum delay is too short, nowcasts can be biased as important parts of the true delay distribution are cut off. At the same time, computational cost scales non-linearly with this setting, so you want the maximum delay to be as long as necessary, but not much longer.<br>Steps to take to determine the maximum delay: |

- Consider what is realistic and relevant for your application.
- Check the proportion of observations reported (`prop_reported`) by delay in the `new_confirm` output of `enw_preprocess_obs`.
- Use `check_max_delay()` to check the coverage of a candidate `max_delay`.
- If in doubt, check if increasing the maximum delay noticeably changes the delay distribution or nowcasts as estimated by `epinowcast`. If it does, your maximum delay may still be too short.

Note that delays are zero indexed and so include the reference date and `max_delay - 1` other intervals (i.e. a `max_delay` of 1 corresponds to no delay).

**timestep** The timestep to used in the process model (i.e. the reference date model). This can be a string ("day", "week", "month") or a numeric whole number representing the number of days. If your data does not have this timestep then you may wish to make use of `enw_aggregate_cumulative()` to aggregate your data to the desired timestep.

### Value

A `data.table` of observations filtered for the latest available data for each reference date at the specified maximum delay.

### See Also

Preprocessing functions `enw_add_delay()`, `enw_add_max_reported()`, `enw_add_metaobs_features()`, `enw_assign_group()`, `enw_complete_dates()`, `enw_construct_data()`, `enw_extend_date()`, `enw_filter_delay()`, `enw_filter_reference_dates()`, `enw_filter_reference_dates_by_report_start()`, `enw_filter_report_dates()`, `enw_flag_observed_observations()`, `enw_impute_na_observations()`, `enw_latest_data()`, `enw_metadata()`, `enw_metadata_delay()`, `enw_missing_reference()`, `enw_preprocess_data()`, `enw_reporting_triangle()`, `enw_reporting_triangle_to_long()`, `enw_retrospective()`

### Examples

```
obs <- enw_example("preprocessed")$obs[[1]]
enw_obs_at_delay(obs, max_delay = 2)
```

---

`enw_one_hot_encode_feature`

*One-hot encode a variable and column-bind it to the original data.table*

---

### Description

This function takes a `data.frame` and a categorical variable, performs one-hot encoding, and column-binds the encoded variables back to the `data.frame`.

### Usage

```
enw_one_hot_encode_feature(metaobs, feature, contrasts = FALSE)
```

**Arguments**

|           |                                                                                                                                |
|-----------|--------------------------------------------------------------------------------------------------------------------------------|
| metaobs   | A data.frame containing the data to be encoded.                                                                                |
| feature   | The name of the categorical variable to one-hot encode as a character string.                                                  |
| contrasts | Logical. If TRUE, create one-hot encoded variables with contrasts; if FALSE, create them without contrasts. Defaults to FALSE. |

**See Also**

Functions used to formulate models [enw\\_add\\_cumulative\\_membership\(\)](#), [enw\\_add\\_pooling\\_effect\(\)](#), [enw\\_design\(\)](#), [enw\\_effects\\_metadata\(\)](#)

**Examples**

```
metaobs <- data.frame(week = 1:2)
enw_one_hot_encode_feature(metaobs, "week")
enw_one_hot_encode_feature(metaobs, "week", contrasts = TRUE)

metaobs <- data.frame(week = 1:6)
enw_one_hot_encode_feature(metaobs, "week")
enw_one_hot_encode_feature(metaobs, "week", contrasts = TRUE)
```

---

enw\_pathfinder

*Fit a CmdStan model using the pathfinder algorithm*


---

**Description**

For more information on the pathfinder algorithm see the [CmdStan documentation](#). # nolint

**Usage**

```
enw_pathfinder(
  data,
  model = epinowcast::enw_model(),
  diagnostics = TRUE,
  init = NULL,
  ...
)
```

**Arguments**

|             |                                                                                                                                                                              |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| data        | A list of data as produced by model modules (for example <a href="#">enw_expectation()</a> , <a href="#">enw_obs()</a> , etc.) and as required for use the model being used. |
| model       | A cmdstanr model object as loaded by <a href="#">enw_model()</a> or as supplied by the user.                                                                                 |
| diagnostics | Logical, defaults to TRUE. Should fitting diagnostics be returned as a data.frame.                                                                                           |
| init        | A list of initial values or a function to generate initial values. If not provided, the model will attempt to generate initial values                                        |
| ...         | Additional parameters to be passed to CmdStanModel\$pathfinder().                                                                                                            |

**Details**

Note that the `threads_per_chain` argument is renamed to `num_threads` to match the `CmdStanModel$pathfinder()` method.

This fitting method is faster but more approximate than the NUTS sampler used in `enw_sample()` and as such is recommended for use in exploratory analysis and model development.

**Value**

A `data.table` containing the fit, data, and `fit_args`. If `diagnostics` is `TRUE`, it also includes the `run_time` column with the timing information.

**See Also**

Functions used to help convert models into the format required for stan `enw_formula_as_data_list()`, `enw_get_cache()`, `enw_model()`, `enw_priors_as_data_list()`, `enw_replace_priors()`, `enw_sample()`, `enw_set_cache()`, `enw_stan_to_r()`, `enw_unset_cache()`, `remove_profiling()`, `write_stan_files_no_profile()`

**Examples**

```
pobs <- enw_example("preprocessed")

nowcast <- epinowcast(pobs,
  expectation = enw_expectation(~1, data = pobs),
  fit = enw_fit_opts(enw_pathfinder, pp = TRUE),
  obs = enw_obs(family = "poisson", data = pobs),
)

summary(nowcast)
```

---

`enw_plot_delay_counts` *Plot notifications by delay group*

---

**Description**

Stacked bar plot of notifications by reference date, coloured by delay group.

**Usage**

```
enw_plot_delay_counts(pobs, delay_group_thresh, facet = TRUE)
```

**Arguments**

|                                 |                                                                                                                                                                                                    |
|---------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>pobs</code>               | A preprocessed data object as produced by <code>enw_preprocess_data()</code> .                                                                                                                     |
| <code>delay_group_thresh</code> | A numeric vector defining left-closed interval thresholds for delay groups.                                                                                                                        |
| <code>facet</code>              | Logical. When <code>TRUE</code> (the default), plots with more than one <code>.group</code> are automatically wrapped by group. Set to <code>FALSE</code> to disable and add a custom facet layer. |

**Value**

A ggplot2 plot.

**See Also**

Plotting functions [enw\\_delay\\_categories\(\)](#), [enw\\_delay\\_quantiles\(\)](#), [enw\\_plot\\_delay\\_cumulative\(\)](#), [enw\\_plot\\_delay\\_fraction\(\)](#), [enw\\_plot\\_delay\\_quantiles\(\)](#), [enw\\_plot\\_nowcast\\_quantiles\(\)](#), [enw\\_plot\\_obs\(\)](#), [enw\\_plot\\_pp\\_quantiles\(\)](#), [enw\\_plot\\_quantiles\(\)](#), [enw\\_plot\\_theme\(\)](#), [plot.enw\\_preprocess\\_data\(\)](#), [plot.epinowcast\(\)](#)

**Examples**

```
pobs <- enw_example("preprocessed_observations")
enw_plot_delay_counts(pobs, c(0, 2, 5, 10, 21))
```

---

enw\_plot\_delay\_cumulative

*Plot cumulative empirical reporting delay*

---

**Description**

Stacked ribbon plot showing the cumulative fraction reported by delay group over reference dates.

**Usage**

```
enw_plot_delay_cumulative(pobs, delay_group_thresh, facet = TRUE)
```

**Arguments**

|                    |                                                                                                                                                             |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pobs               | A preprocessed data object as produced by <a href="#">enw_preprocess_data()</a> .                                                                           |
| delay_group_thresh | A numeric vector defining left-closed interval thresholds for delay groups.                                                                                 |
| facet              | Logical. When TRUE (the default), plots with more than one .group are automatically wrapped by group. Set to FALSE to disable and add a custom facet layer. |

**Value**

A ggplot2 plot.

**See Also**

Plotting functions [enw\\_delay\\_categories\(\)](#), [enw\\_delay\\_quantiles\(\)](#), [enw\\_plot\\_delay\\_counts\(\)](#), [enw\\_plot\\_delay\\_fraction\(\)](#), [enw\\_plot\\_delay\\_quantiles\(\)](#), [enw\\_plot\\_nowcast\\_quantiles\(\)](#), [enw\\_plot\\_obs\(\)](#), [enw\\_plot\\_pp\\_quantiles\(\)](#), [enw\\_plot\\_quantiles\(\)](#), [enw\\_plot\\_theme\(\)](#), [plot.enw\\_preprocess\\_data\(\)](#), [plot.epinowcast\(\)](#)

**Examples**

```
pobs <- enw_example("preprocessed_observations")
enw_plot_delay_cumulative(pobs, c(0, 2, 5, 10, 21))
```

---

```
enw_plot_delay_fraction
```

*Plot empirical reporting delay heatmap*

---

**Description**

Tile plot showing the fraction reported by delay group and reference date.

**Usage**

```
enw_plot_delay_fraction(pobs, delay_group_thresh, facet = TRUE)
```

**Arguments**

|                    |                                                                                                                                                             |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pobs               | A preprocessed data object as produced by <a href="#">enw_preprocess_data()</a> .                                                                           |
| delay_group_thresh | A numeric vector defining left-closed interval thresholds for delay groups.                                                                                 |
| facet              | Logical. When TRUE (the default), plots with more than one .group are automatically wrapped by group. Set to FALSE to disable and add a custom facet layer. |

**Value**

A ggplot2 plot.

**See Also**

Plotting functions [enw\\_delay\\_categories\(\)](#), [enw\\_delay\\_quantiles\(\)](#), [enw\\_plot\\_delay\\_counts\(\)](#), [enw\\_plot\\_delay\\_cumulative\(\)](#), [enw\\_plot\\_delay\\_quantiles\(\)](#), [enw\\_plot\\_nowcast\\_quantiles\(\)](#), [enw\\_plot\\_obs\(\)](#), [enw\\_plot\\_pp\\_quantiles\(\)](#), [enw\\_plot\\_quantiles\(\)](#), [enw\\_plot\\_theme\(\)](#), [plot.enw\\_preprocess\\_data\(\)](#), [plot.epinowcast\(\)](#)

**Examples**

```
pobs <- enw_example("preprocessed_observations")
enw_plot_delay_fraction(pobs, c(0, 2, 5, 10, 21))
```

---

enw\_plot\_delay\_quantiles

*Plot empirical reporting delay quantiles*


---

### Description

Line plot showing quantiles of the empirical delay distribution over reference dates.

### Usage

```
enw_plot_delay_quantiles(pobs, quantiles = c(0.1, 0.5, 0.9), facet = TRUE)
```

### Arguments

|           |                                                                                                                                                                          |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pobs      | A preprocessed data object as produced by <a href="#">enw_preprocess_data()</a> .                                                                                        |
| quantiles | A numeric vector of probabilities. Defaults to <code>c(0.1, 0.5, 0.9)</code> .                                                                                           |
| facet     | Logical. When TRUE (the default), plots with more than one <code>.group</code> are automatically wrapped by group. Set to FALSE to disable and add a custom facet layer. |

### Value

A ggplot2 plot.

### See Also

Plotting functions [enw\\_delay\\_categories\(\)](#), [enw\\_delay\\_quantiles\(\)](#), [enw\\_plot\\_delay\\_counts\(\)](#), [enw\\_plot\\_delay\\_cumulative\(\)](#), [enw\\_plot\\_delay\\_fraction\(\)](#), [enw\\_plot\\_nowcast\\_quantiles\(\)](#), [enw\\_plot\\_obs\(\)](#), [enw\\_plot\\_pp\\_quantiles\(\)](#), [enw\\_plot\\_quantiles\(\)](#), [enw\\_plot\\_theme\(\)](#), [plot.enw\\_preprocess\\_data\(\)](#), [plot.epinowcast\(\)](#)

### Examples

```
pobs <- enw_example("preprocessed_observations")
enw_plot_delay_quantiles(pobs)
```

---

enw\_plot\_nowcast\_quantiles

*Plot nowcast quantiles*


---

### Description

Plot nowcast quantiles

Usage

```
enw_plot_nowcast_quantiles(nowcast, latest_obs = NULL, log = FALSE, ...)
```

Arguments

|            |                                                                                                                                                    |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| nowcast    | A data.frame of summarised posterior nowcast estimates containing at least a confirm count column and a reference_date date variable.              |
| latest_obs | A data.frame of observed data containing at least a confirm count variable and the same date variable as in the main data.frame used for plotting. |
| log        | Logical, defaults to FALSE. Should counts be plot on the log scale.                                                                                |
| ...        | Additional arguments passed to <a href="#">enw_plot_pp_quantiles()</a> .                                                                           |

Value

A ggplot2 plot.

See Also

Plotting functions [enw\\_delay\\_categories\(\)](#), [enw\\_delay\\_quantiles\(\)](#), [enw\\_plot\\_delay\\_counts\(\)](#), [enw\\_plot\\_delay\\_cumulative\(\)](#), [enw\\_plot\\_delay\\_fraction\(\)](#), [enw\\_plot\\_delay\\_quantiles\(\)](#), [enw\\_plot\\_obs\(\)](#), [enw\\_plot\\_pp\\_quantiles\(\)](#), [enw\\_plot\\_quantiles\(\)](#), [enw\\_plot\\_theme\(\)](#), [plot.enw\\_preprocess\\_data\(\)](#), [plot.epinowcast\(\)](#)

Examples

```
nowcast <- enw_example("nowcast")
nowcast <- summary(nowcast, probs = c(0.05, 0.2, 0.8, 0.95))
enw_plot_nowcast_quantiles(nowcast)
```

---

|              |                              |
|--------------|------------------------------|
| enw_plot_obs | <i>Generic quantile plot</i> |
|--------------|------------------------------|

---

Description

Generic quantile plot

Usage

```
enw_plot_obs(obs, latest_obs = NULL, log = TRUE, ...)
```

**Arguments**

|                         |                                                                                                                                                                              |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>obs</code>        | A <code>data.frame</code> of summarised posterior estimates containing at least a confirm count column and a date variable                                                   |
| <code>latest_obs</code> | A <code>data.frame</code> of observed data containing at least a confirm count variable and the same date variable as in the main <code>data.frame</code> used for plotting. |
| <code>log</code>        | Logical, defaults to <code>FALSE</code> . Should counts be plot on the log scale.                                                                                            |
| <code>...</code>        | Additional arguments passed to <code>ggplot2::aes()</code> must at least specify the x date variable.                                                                        |

**Value**

A `ggplot2` plot.

**See Also**

Plotting functions `enw_delay_categories()`, `enw_delay_quantiles()`, `enw_plot_delay_counts()`, `enw_plot_delay_cumulative()`, `enw_plot_delay_fraction()`, `enw_plot_delay_quantiles()`, `enw_plot_nowcast_quantiles()`, `enw_plot_pp_quantiles()`, `enw_plot_quantiles()`, `enw_plot_theme()`, `plot.enw_preprocess_data()`, `plot.epinowcast()`

**Examples**

```
nowcast <- enw_example("nowcast")
obs <- enw_example("obs")

# Plot observed data by reference date
enw_plot_obs(obs, x = reference_date)

# Plot observed data by reference date with more recent data
enw_plot_obs(nowcast$latest[[1]], obs, x = reference_date)
```

---

`enw_plot_pp_quantiles` *Plot posterior prediction quantiles*

---

**Description**

Plot posterior prediction quantiles

**Usage**

```
enw_plot_pp_quantiles(pp, log = FALSE, ...)
```

**Arguments**

|                  |                                                                                                                                                                  |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>pp</code>  | A <code>data.frame</code> of summarised posterior predictions estimates containing at least a confirm count column and a <code>report_date</code> date variable. |
| <code>log</code> | Logical, defaults to <code>FALSE</code> . Should counts be plot on the log scale.                                                                                |
| <code>...</code> | Additional arguments passed to <code>enw_plot_pp_quantiles()</code> .                                                                                            |

**Value**

A ggplot2 plot.

**See Also**

Plotting functions `enw_delay_categories()`, `enw_delay_quantiles()`, `enw_plot_delay_counts()`, `enw_plot_delay_cumulative()`, `enw_plot_delay_fraction()`, `enw_plot_delay_quantiles()`, `enw_plot_nowcast_quantiles()`, `enw_plot_obs()`, `enw_plot_quantiles()`, `enw_plot_theme()`, `plot.enw_preprocess_data()`, `plot.epinowcast()`

**Examples**

```
nowcast <- enw_example("nowcast")
nowcast <- summary(
  nowcast, type = "posterior_prediction", probs = c(0.05, 0.2, 0.8, 0.95)
)
enw_plot_pp_quantiles(nowcast) +
  ggplot2::facet_wrap(ggplot2::vars(reference_date), scales = "free")
```

---

|                    |                              |
|--------------------|------------------------------|
| enw_plot_quantiles | <i>Generic quantile plot</i> |
|--------------------|------------------------------|

---

**Description**

Generic quantile plot

**Usage**

```
enw_plot_quantiles(posterior, latest_obs = NULL, log = FALSE, ...)
```

**Arguments**

|            |                                                                                                                                                                                                                                                                                                                                             |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| posterior  | A data.frame of summarised posterior estimates containing at least a confirm count column a date variable, quantile estimates for the 5%, 20%, 80%, and 95% quantiles and the mean and median. This function is wrapped in <code>enw_plot_nowcast_quantiles()</code> and <code>enw_plot_pp_quantiles()</code> with sensible default labels. |
| latest_obs | A data.frame of observed data containing at least a confirm count variable and the same date variable as in the main data.frame used for plotting.                                                                                                                                                                                          |
| log        | Logical, defaults to FALSE. Should counts be plot on the log scale.                                                                                                                                                                                                                                                                         |
| ...        | Additional arguments passed to <code>ggplot2::aes()</code> must at least specify the x date variable.                                                                                                                                                                                                                                       |

**Value**

A ggplot2 plot.

**See Also**

`enw_plot_nowcast_quantiles()`, `enw_plot_pp_quantiles()`

Plotting functions `enw_delay_categories()`, `enw_delay_quantiles()`, `enw_plot_delay_counts()`, `enw_plot_delay_cumulative()`, `enw_plot_delay_fraction()`, `enw_plot_delay_quantiles()`, `enw_plot_nowcast_quantiles()`, `enw_plot_obs()`, `enw_plot_pp_quantiles()`, `enw_plot_theme()`, `plot.enw_preprocess_data()`, `plot.epinowcast()`

**Examples**

```
nowcast <- enw_example("nowcast")
nowcast <- summary(nowcast, probs = c(0.05, 0.2, 0.8, 0.95))
enw_plot_quantiles(nowcast, x = reference_date)
```

---

enw\_plot\_theme

*Package plot theme*


---

**Description**

Package plot theme

**Usage**

```
enw_plot_theme(plot)
```

**Arguments**

`plot` `ggplot2` plot object.

**Value**

`ggplot2` plot object.

**See Also**

Plotting functions `enw_delay_categories()`, `enw_delay_quantiles()`, `enw_plot_delay_counts()`, `enw_plot_delay_cumulative()`, `enw_plot_delay_fraction()`, `enw_plot_delay_quantiles()`, `enw_plot_nowcast_quantiles()`, `enw_plot_obs()`, `enw_plot_pp_quantiles()`, `enw_plot_quantiles()`, `plot.enw_preprocess_data()`, `plot.epinowcast()`

---

|               |                                |
|---------------|--------------------------------|
| enw_posterior | <i>Summarise the posterior</i> |
|---------------|--------------------------------|

---

## Description

A generic wrapper around `posterior::summarise_draws()` with opinionated defaults.

## Usage

```
enw_posterior(fit, variables = NULL, probs = c(0.05, 0.2, 0.8, 0.95), ...)
```

## Arguments

|                        |                                                                                                                                                                                                       |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>fit</code>       | A cmdstanr fit object.                                                                                                                                                                                |
| <code>variables</code> | A character vector of variables to return posterior summaries for. By default summaries for all parameters are returned.                                                                              |
| <code>probs</code>     | A vector of numeric probabilities to produce quantile summaries for. By default these are the 5%, 20%, 80%, and 95% quantiles which are also the minimum set required for plotting functions to work. |
| <code>...</code>       | Additional arguments that may be passed but will not be used.                                                                                                                                         |

## Value

A data.frame summarising the model posterior.

## See Also

Functions used for postprocessing of model fits `.check_primarycensored()`, `.delay_draw_columns()`, `.discretise_parametric_pmf()`, `build_ord_obs()`, `enw_add_latest_obs_to_nowcast()`, `enw_nowcast_samples()`, `enw_nowcast_summary()`, `enw_posterior_delay()`, `enw_pp_summary()`, `enw_quantiles_to_long()`, `enw_summarise_samples()`, `subset_obs()`

## Examples

```
fit <- enw_example("nowcast")
enw_posterior(fit$fit[[1]], variables = "expr_beta")
```

---

enw\_posterior\_delay      *Posterior samples of the parametric reporting-delay distribution*


---

## Description

Extracts posterior draws of the parametric reporting-delay distribution from a fit (e.g. a delay-only fit, see the delay estimation vignette) and returns the discretised delay probability mass function for each draw. This supports comparing the estimated distribution against a known truth and plotting posterior samples rather than only summaries.

## Usage

```
enw_posterior_delay(fit, max_delay, distribution = "lognormal", draws = NULL)
```

## Arguments

|              |                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| fit          | A cmdstanr fit object (the fit[[1]] element of an epinowcast output).                                                                                                                                                                                                                                                                                                                                               |
| max_delay    | Maximum delay (number of delay slots, delays 0:(max_delay - 1)).                                                                                                                                                                                                                                                                                                                                                    |
| distribution | The parametric distribution used in <a href="#">enw_reference()</a> , one of "lognormal", "gamma", "exponential", or "loglogistic". Defaults to "lognormal". A cmdstanr fit does not record which distribution was used (the model_refp id is Stan data, not saved to the draws), so this must match the distribution passed to <a href="#">enw_reference()</a> . An incorrect value silently yields the wrong PMF. |
| draws        | Optional integer; if supplied, a random subset of this many posterior draws is returned (useful for plotting). The same draws are used across all reference-design rows.                                                                                                                                                                                                                                            |

## Details

The function reads the full refp\_mean and refp\_sd vectors saved by the Stan model (one entry per unique combination of parametric reference-date covariates) and discretises a delay PMF for each. For an intercept-only delay (the common case, refp\_mean of length one) the output is a long data.table with columns .draw, delay, and pmf. For a delay model with reference-date covariates, random effects, or time- or group-varying delays (refp\_mean of length greater than one) an additional integer row column identifies the reference-design row, and each row may have a different PMF. The mapping from row to reference date and group is set by the reference module's fixed-effect design (see [enw\\_reference\(\)](#) and enw\_formula\_as\_data\_list()'s refp\_findex); it is not recovered here because a cmdstanr fit does not retain that design on its own.

## Value

A long data.table with columns .draw, delay, and pmf, plus an integer row column when the delay model has more than one reference-design row.

See Also

Functions used for postprocessing of model fits `.check_primarycensored()`, `.delay_draw_columns()`, `.discretise_parametric_pmf()`, `build_ord_obs()`, `enw_add_latest_obs_to_nowcast()`, `enw_nowcast_samples()`, `enw_nowcast_summary()`, `enw_posterior()`, `enw_pp_summary()`, `enw_quantiles_to_long()`, `enw_summarise_samples()`, `subset_obs()`

Examples

```
fit <- enw_example("nowcast")
# Intercept-only delay
enw_posterior_delay(fit$fit[[1]], max_delay = 20, draws = 50)

# A delay model with a reference-date covariate yields one PMF per
# reference-design row, identified by the `row` column:
## Not run:
pobs <- enw_example("preprocessed")
covariate_fit <- epinowcast(
  pobs,
  reference = enw_reference(~ 1 + day_of_week, data = pobs),
  obs = enw_obs(delay_only = TRUE, data = pobs)
)
enw_posterior_delay(covariate_fit$fit[[1]], max_delay = 20, draws = 50)

## End(Not run)
```

---

|                |                                     |
|----------------|-------------------------------------|
| enw_pp_summary | <i>Posterior predictive summary</i> |
|----------------|-------------------------------------|

---

Description

This function summarises posterior predictives for observed data (by report and reference date). The functionality of this function can be used directly on the output of `epinowcast()` using the supplied `summary.epinowcast()` method.

Usage

```
enw_pp_summary(fit, diff_obs, probs = c(0.05, 0.2, 0.35, 0.5, 0.65, 0.8, 0.95))
```

Arguments

|          |                                                                                                                                                                                                       |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| fit      | A cmdstanr fit object.                                                                                                                                                                                |
| diff_obs | A data.frame of observed data with at least a date variable <code>reference_date</code> , and a grouping variable <code>.group</code> .                                                               |
| probs    | A vector of numeric probabilities to produce quantile summaries for. By default these are the 5%, 20%, 80%, and 95% quantiles which are also the minimum set required for plotting functions to work. |

**Value**

A data.table summarising the posterior predictions.

**See Also**

Functions used for postprocessing of model fits [.check\\_primarycensored\(\)](#), [.delay\\_draw\\_columns\(\)](#), [.discretise\\_parametric\\_pmf\(\)](#), [build\\_ord\\_obs\(\)](#), [enw\\_add\\_latest\\_obs\\_to\\_nowcast\(\)](#), [enw\\_nowcast\\_samples\(\)](#), [enw\\_nowcast\\_summary\(\)](#), [enw\\_posterior\(\)](#), [enw\\_posterior\\_delay\(\)](#), [enw\\_quantiles\\_to\\_long\(\)](#), [enw\\_summarise\\_samples\(\)](#), [subset\\_obs\(\)](#)

**Examples**

```
fit <- enw_example("nowcast")
enw_pp_summary(fit$fit[[1]], fit$new_confirm[[1]], probs = c(0.5))
```

---

|                     |                                |
|---------------------|--------------------------------|
| enw_preprocess_data | <i>Preprocess observations</i> |
|---------------------|--------------------------------|

---

**Description**

This function preprocesses raw observations under the assumption they are reported as cumulative counts by a reference and report date and is used to assign groups. It also constructs data objects used by visualisation and modelling functions including the observed empirical probability of a report on a given day, the cumulative probability of report, the latest available observations, incidence of observations, and metadata about the date of reference and report (used to construct models). This function wraps other preprocessing functions that may be instead used individually if required. Note that internally reports beyond the user specified delay are dropped for modelling purposes with the `cum_prop_reported` and `max_confirm` variables allowing the user to check the impact this may have (if `cum_prop_reported` is significantly below 1 a longer `max_delay` may be appropriate). Also note that if missing reference or report dates are suspected to occur in your data then these need to be completed with [enw\\_complete\\_dates\(\)](#).

**Usage**

```
enw_preprocess_data(
  obs,
  by = NULL,
  max_delay,
  timestep = "day",
  set_negatives_to_zero = TRUE,
  ...,
  copy = TRUE
)
```

## Arguments

|                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| obs                   | A data.frame containing at least the following variables: reference_date (index date of interest), report_date (report date for observations), confirm (cumulative observations by reference and report date).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| by                    | A character vector describing the stratification of observations. This defaults to no grouping. This should be used when modelling multiple time series in order to identify them for downstream modelling                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| max_delay             | <p>The maximum delay to model in the delay distribution, specified in units of the timestep (e.g., if timestep = "week", then max_delay = 3 means 3 weeks). If not specified the maximum observed delay is assumed to be the true maximum delay in the model. Otherwise, an integer greater than or equal to 1 can be specified. Observations with delays larger than the maximum delay will be dropped. If the specified maximum delay is too short, nowcasts can be biased as important parts of the true delay distribution are cut off. At the same time, computational cost scales non-linearly with this setting, so you want the maximum delay to be as long as necessary, but not much longer.</p> <p>Steps to take to determine the maximum delay:</p> <ul style="list-style-type: none"> <li>• Consider what is realistic and relevant for your application.</li> <li>• Check the proportion of observations reported (prop_reported) by delay in the new_confirm output of enw_preprocess_obs.</li> <li>• Use <a href="#">check_max_delay()</a> to check the coverage of a candidate max_delay.</li> <li>• If in doubt, check if increasing the maximum delay noticeably changes the delay distribution or nowcasts as estimated by epinowcast. If it does, your maximum delay may still be too short.</li> </ul> <p>Note that delays are zero indexed and so include the reference date and max_delay - 1 other intervals (i.e. a max_delay of 1 corresponds to no delay).</p> |
| timestep              | The timestep to used in the process model (i.e. the reference date model). This can be a string ("day", "week", "month") or a numeric whole number representing the number of days. If your data does not have this timestep then you may wish to make use of <a href="#">enw_aggregate_cumulative()</a> to aggregate your data to the desired timestep.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| set_negatives_to_zero | Logical, defaults to TRUE. Should negative counts (for calculated incidence of observations) be set to zero? Currently downstream modelling does not support negative counts and so setting must be TRUE if intending to use <a href="#">epinowcast()</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| ...                   | Other arguments to <a href="#">enw_add_metaobs_features()</a> , e.g. holidays, which sets commonly used metadata (e.g. day of week, days since start of time series)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| copy                  | A logical; if TRUE (the default) creates a copy; otherwise, modifies obs in place.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

## Details

If max\_delay is numeric, it will be internally coerced to integer using [as.integer\(\)](#).

**Value**

A data.table containing processed observations as a series of nested data.frames as well as variables containing metadata. These are:

- obs: (observations with the addition of empirical reporting proportions and restricted to the specified maximum delay).
- new\_confirm: Incidence of notifications by reference and report date. Empirical reporting distributions are also added.
- latest: The latest available observations.
- missing\_reference: Observations missing reference dates.
- reporting\_triangle: Incident observations by report and reference date in the standard reporting triangle matrix format.
- metareference: Metadata reference dates derived from observations.
- metrareport: Metadata for report dates.
- metadelay: Metadata for reporting delays produced using [enw\\_metadata\\_delay\(\)](#).
- max\_delay: Maximum delay to be modelled by epinowcast.
- time: Numeric, number of timepoints in the data.
- snapshots: Numeric, number of available data snapshots to use for nowcasting.
- groups: Numeric, Number of groups/strata in the supplied observations (set using by).
- max\_date: The maximum available report date.

**See Also**

Preprocessing functions [enw\\_add\\_delay\(\)](#), [enw\\_add\\_max\\_reported\(\)](#), [enw\\_add\\_metaobs\\_features\(\)](#), [enw\\_assign\\_group\(\)](#), [enw\\_complete\\_dates\(\)](#), [enw\\_construct\\_data\(\)](#), [enw\\_extend\\_date\(\)](#), [enw\\_filter\\_delay\(\)](#), [enw\\_filter\\_reference\\_dates\(\)](#), [enw\\_filter\\_reference\\_dates\\_by\\_report\\_start\(\)](#), [enw\\_filter\\_report\\_dates\(\)](#), [enw\\_flag\\_observed\\_observations\(\)](#), [enw\\_impute\\_na\\_observations\(\)](#), [enw\\_latest\\_data\(\)](#), [enw\\_metadata\(\)](#), [enw\\_metadata\\_delay\(\)](#), [enw\\_missing\\_reference\(\)](#), [enw\\_obs\\_at\\_delay\(\)](#), [enw\\_reporting\\_triangle\(\)](#), [enw\\_reporting\\_triangle\\_to\\_long\(\)](#), [enw\\_retrospective\(\)](#)

**Examples**

```
library(data.table)

# Filter example hospitalisation data to be national and over all ages
nat_germany_hosp <- germany_covid19_hosp[location == "DE"]
nat_germany_hosp <- nat_germany_hosp[age_group == "00+"]

# Preprocess with default settings
pobs <- enw_preprocess_data(nat_germany_hosp)
pobs
```

---

enw\_priors\_as\_data\_list

*Convert prior data.frame to list*


---

### Description

Converts priors defined in a `data.frame` into a list format for use by stan. In addition it adds "\_p" to all variable names in order too allow them to be distinguished from their standard usage within modelling code.

### Usage

```
enw_priors_as_data_list(priors)
```

### Arguments

`priors` A `data.frame` with columns `variable`, `mean`, and `sd` describing normal priors. Default priors in the appropriate format are returned by the `$priors` element of [enw\\_reference\(\)](#) and other model module functions.

### Value

A named list with each entry specifying a prior as a length two vector (specifying the mean and standard deviation of the prior).

### See Also

Functions used to help convert models into the format required for stan [enw\\_formula\\_as\\_data\\_list\(\)](#), [enw\\_get\\_cache\(\)](#), [enw\\_model\(\)](#), [enw\\_pathfinder\(\)](#), [enw\\_replace\\_priors\(\)](#), [enw\\_sample\(\)](#), [enw\\_set\\_cache\(\)](#), [enw\\_stan\\_to\\_r\(\)](#), [enw\\_unset\\_cache\(\)](#), [remove\\_profiling\(\)](#), [write\\_stan\\_files\\_no\\_profile\(\)](#)

### Examples

```
priors <- data.frame(variable = "x", mean = 1, sd = 2)
enw_priors_as_data_list(priors)
```

---

enw\_quantiles\_to\_long *Convert summarised quantiles from wide to long format*


---

### Description

Convert summarised quantiles from wide to long format

### Usage

```
enw_quantiles_to_long(posterior)
```

**Arguments**

posterior      A data.frame as output by [enw\\_posterior\(\)](#).

**Value**

A data.frame of quantiles in long format.

**See Also**

Functions used for postprocessing of model fits [.check\\_primarycensored\(\)](#), [.delay\\_draw\\_columns\(\)](#), [.discretise\\_parametric\\_pmf\(\)](#), [build\\_ord\\_obs\(\)](#), [enw\\_add\\_latest\\_obs\\_to\\_nowcast\(\)](#), [enw\\_nowcast\\_samples\(\)](#), [enw\\_nowcast\\_summary\(\)](#), [enw\\_posterior\(\)](#), [enw\\_posterior\\_delay\(\)](#), [enw\\_pp\\_summary\(\)](#), [enw\\_summarise\\_samples\(\)](#), [subset\\_obs\(\)](#)

**Examples**

```
fit <- enw_example("nowcast")
posterior <- enw_posterior(fit$fit[[1]], var = "expr_llatent_int[1,1]")
enw_quantiles_to_long(posterior)
```

---

enw\_reference

*Reference date logit hazard reporting model module*


---

**Description**

Specifies the reference date reporting delay model using parametric and/or non-parametric hazard formulations.

**Usage**

```
enw_reference(
  parametric = ~1,
  distribution = c("lognormal", "none", "exponential", "gamma"),
  non_parametric = ~0,
  data
)
```

**Arguments**

parametric      A formula (as implemented in [enw\\_formula\(\)](#)) describing the parametric reference date delay model. This can use features defined by report date as defined in [metareference](#) as produced by [enw\\_preprocess\\_data\(\)](#). Note that this formula will be applied to all summary statistics of the chosen parametric distribution but each summary parameter will have separate effects. Set to `~0` to disable the parametric component (internally converted to `~1` with distribution set to "none"). See [enw\\_formula\(\)](#) for details on formula syntax including fixed effects, random effects, and random walks.

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| distribution   | A character vector describing the parametric delay distribution to use. Current options are: "none", "lognormal", "gamma", and "exponential", with the default being "lognormal". These distributions are discretised using the double interval censoring approach from the <code>primarycensored</code> package, which accounts for primary event censoring, secondary interval censoring, and right truncation. The log-logistic distribution is not currently supported, pending log-logistic support in <code>primarycensored</code> (see <a href="https://github.com/epinowcast/primarycensored/issues/321">https://github.com/epinowcast/primarycensored/issues/321</a> ). See <code>vignette("distributions")</code> for details.                                                                                                        |
| non_parametric | A formula (as implemented in <code>enw_formula()</code> ) describing the non-parametric logit hazard model. This can use features defined by reference date and by delay. It draws on a linked <code>data.frame</code> using <code>metareference</code> and <code>metadelay</code> as produced by <code>enw_preprocess_data()</code> . When an effect per delay is specified this approximates the Cox proportional hazards model in discrete time with a single strata. Set to <code>~0</code> to disable the non-parametric component (internally converted to <code>~1</code> and flagged as inactive). When used in conjunction with a parametric model it often makes sense to disable the intercept to ensure model identifiability (e.g., <code>~0 + (1   delay)</code> ). See <code>enw_formula()</code> for details on formula syntax. |
| data           | Output from <code>enw_preprocess_data()</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

### Value

A list containing the supplied formulas, data passed into a list describing the models, a `data.frame` describing the priors used, and a function that takes the output data and priors and returns a function that can be used to sample from a tightened version of the prior distribution.

### See Also

Model modules `.expectation_is_minimal()`, `enw_expectation()`, `enw_fit_opts()`, `enw_missing()`, `enw_obs()`, `enw_report()`

### Examples

```
# Parametric model with a lognormal distribution
enw_reference(
  parametric = ~1, distribution = "lognormal",
  data = enw_example("preprocessed")
)

# Non-parametric model with a random effect per delay
enw_reference(
  parametric = ~0, non_parametric = ~ 1 + (1 | delay),
  data = enw_example("preprocessed")
)

# Combined parametric and non-parametric model
enw_reference(
  parametric = ~1, non_parametric = ~ 0 + (1 | delay_cat),
  data = enw_example("preprocessed")
)
```

---

enw\_reference\_by\_report

*Construct a lookup of references dates by report*


---

## Description

Construct a lookup of references dates by report

## Usage

```
enw_reference_by_report(
  missing_reference,
  reps_with_complete_refs,
  metareference,
  max_delay
)
```

## Arguments

missing\_reference

missing\_reference data.frame output from [enw\\_preprocess\\_data\(\)](#).

reps\_with\_complete\_refs

A data.frame of report dates with complete (i.e fully reported) reference dates as produced using [enw\\_reps\\_with\\_complete\\_refs\(\)](#).

metareference

metareference data.frame output from [enw\\_preprocess\\_data\(\)](#).

max\_delay

The maximum delay to model in the delay distribution, specified in units of the timestep (e.g., if timestep = "week", then max\_delay = 3 means 3 weeks). If not specified the maximum observed delay is assumed to be the true maximum delay in the model. Otherwise, an integer greater than or equal to 1 can be specified. Observations with delays larger than the maximum delay will be dropped. If the specified maximum delay is too short, nowcasts can be biased as important parts of the true delay distribution are cut off. At the same time, computational cost scales non-linearly with this setting, so you want the maximum delay to be as long as necessary, but not much longer.

Steps to take to determine the maximum delay:

- Consider what is realistic and relevant for your application.
- Check the proportion of observations reported (prop\_reported) by delay in the new\_confirm output of [enw\\_preprocess\\_obs\(\)](#).
- Use [check\\_max\\_delay\(\)](#) to check the coverage of a candidate max\_delay.
- If in doubt, check if increasing the maximum delay noticeably changes the delay distribution or nowcasts as estimated by [epinowcast](#). If it does, your maximum delay may still be too short.

Note that delays are zero indexed and so include the reference date and max\_delay - 1 other intervals (i.e. a max\_delay of 1 corresponds to no delay).

**Value**

A wide `data.frame` with each row being a complete report date and' the columns being the observation index for each reporting delay

**See Also**

Helper functions for model modules `add_max_observed_delay()`, `add_pmfs()`, `convolution_matrix()`, `delay_only_ltotal()`, `delay_only_total()`, `enw_dayofweek_structural_reporting()`, `enw_reps_with_complete_enw_structural_reporting_metadata()`, `extract_obs_metadata()`, `extract_sparse_matrix()`, `latest_obs_as_matrix()`

---

|                    |                                                          |
|--------------------|----------------------------------------------------------|
| enw_replace_priors | <i>Replace default priors with user specified priors</i> |
|--------------------|----------------------------------------------------------|

---

**Description**

Replaces default model priors with user specified ones (restricted to normal priors with specified mean and standard deviations). A common use is extracting the posterior from a previous `epinowcast()` run (using `summary(nowcast, type = "fit")`) and using it as a prior for subsequent fits.

**Usage**

```
enw_replace_priors(priors, custom_priors)
```

**Arguments**

|               |                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| priors        | A <code>data.frame</code> with columns <code>variable</code> , <code>mean</code> , and <code>sd</code> describing normal priors. Default priors in the appropriate format are returned by the <code>\$priors</code> element of <code>enw_reference()</code> and other model module functions.                                                                                                                                     |
| custom_priors | A <code>data.frame</code> with columns <code>variable</code> , <code>mean</code> , and <code>sd</code> describing normal priors to use as replacements. Rows in <code>custom_priors</code> replace matching rows in <code>priors</code> by the <code>variable</code> column. Vectorised prior names (i.e. those of the form <code>variable[n]</code> ) are matched after stripping the index (treated as <code>variable</code> ). |

**Details**

Default priors can be obtained from each model module's `$priors` element, e.g. `enw_reference(data = pobs)$priors`. See the `priors` argument of `epinowcast()` for a list of available prior variable names by module.

**Value**

A `data.table` of prior definitions (`variable`, `mean` and `sd`).

**See Also**

Functions used to help convert models into the format required for stan [enw\\_formula\\_as\\_data\\_list\(\)](#), [enw\\_get\\_cache\(\)](#), [enw\\_model\(\)](#), [enw\\_pathfinder\(\)](#), [enw\\_priors\\_as\\_data\\_list\(\)](#), [enw\\_sample\(\)](#), [enw\\_set\\_cache\(\)](#), [enw\\_stan\\_to\\_r\(\)](#), [enw\\_unset\\_cache\(\)](#), [remove\\_profiling\(\)](#), [write\\_stan\\_files\\_no\\_profile\(\)](#)

**Examples**

```
# Update priors from a data.frame
priors <- data.frame(variable = c("x", "y"), mean = c(1, 2), sd = c(1, 2))
custom_priors <- data.frame(variable = "x[1]", mean = 10, sd = 2)
enw_replace_priors(priors, custom_priors)

# Update priors from a previous model fit
default_priors <- enw_reference(
  distribution = "lognormal",
  data = enw_example("preprocessed"),
)$priors
print(default_priors)

fit_priors <- summary(
  enw_example("nowcast"),
  type = "fit",
  variables = c("refp_mean_int", "refp_sd_int", "sqrt_phi")
)
fit_priors

enw_replace_priors(default_priors, fit_priors)
```

---

enw\_report

---

*Report date logit hazard reporting model module*


---

**Description**

Report date logit hazard reporting model module

**Usage**

```
enw_report(non_parametric = ~0, structural = NULL, data)
```

**Arguments**

**non\_parametric** A formula (as implemented in [enw\\_formula\(\)](#)) describing the non-parametric logit hazard model for report date effects. This can use features defined by report date as defined in `metareport` as produced by [enw\\_preprocess\\_data\(\)](#). Note that the intercept for this model is set to 0 as it should be used for specifying report date related hazards rather than time-invariant hazards, which should instead be modelled using the `non_parametric` argument of [enw\\_reference\(\)](#). Set to `~0` to disable (internally converted to `~1` and flagged as inactive). See [enw\\_formula\(\)](#) for details on formula syntax.

|            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| structural | A <code>data.table</code> describing the known reporting structure (e.g., weekday-only reporting). This should be created using <code>enw_dayofweek_structural_reporting()</code> for day-of-week patterns, or <code>enw_structural_reporting_metadata()</code> as a base for custom patterns. The <code>data.table</code> must have columns: <code>.group</code> , <code>date</code> , <code>report_date</code> , and <code>report</code> (binary indicator where 1 = reporting occurs). This is particularly useful for modeling fixed reporting cycles, such as Wednesday-only reporting. Set to <code>NULL</code> to disable (default). |
| data       | Output from <code>enw_preprocess_data()</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

**Value**

A list containing the supplied formulas, data passed into a list describing the models, a `data.frame` describing the priors used, and a function that takes the output data and priors and returns a function that can be used to sample from a tightened version of the prior distribution.

**See Also**

Model modules `.expectation_is_minimal()`, `enw_expectation()`, `enw_fit_opts()`, `enw_missing()`, `enw_obs()`, `enw_reference()`

**Examples**

```
# Basic report model
enw_report(data = enw_example("preprocessed"))

## Not run:
# With Wednesday-only reporting structure
pobs <- enw_example("preprocessed")
structural <- enw_dayofweek_structural_reporting(
  pobs,
  day_of_week = "Wednesday"
)
enw_report(structural = structural, data = pobs)

## End(Not run)
```

---

enw\_reporting\_triangle

*Construct the reporting triangle*

---

**Description**

Constructs the reporting triangle with each row representing a reference date and columns being observations by report date

**Usage**

```
enw_reporting_triangle(obs)
```

**Arguments**

**obs** A data.frame as produced by [enw\\_add\\_incidence\(\)](#). Must contain the following variables: reference\_date, .group, delay.

**Value**

A data.frame with each row being a reference date, and columns being observations by reporting delay.

**See Also**

Preprocessing functions [enw\\_add\\_delay\(\)](#), [enw\\_add\\_max\\_reported\(\)](#), [enw\\_add\\_metaobs\\_features\(\)](#), [enw\\_assign\\_group\(\)](#), [enw\\_complete\\_dates\(\)](#), [enw\\_construct\\_data\(\)](#), [enw\\_extend\\_date\(\)](#), [enw\\_filter\\_delay\(\)](#), [enw\\_filter\\_reference\\_dates\(\)](#), [enw\\_filter\\_reference\\_dates\\_by\\_report\\_start\(\)](#), [enw\\_filter\\_report\\_dates\(\)](#), [enw\\_flag\\_observed\\_observations\(\)](#), [enw\\_impute\\_na\\_observations\(\)](#), [enw\\_latest\\_data\(\)](#), [enw\\_metadata\(\)](#), [enw\\_metadata\\_delay\(\)](#), [enw\\_missing\\_reference\(\)](#), [enw\\_obs\\_at\\_delay\(\)](#), [enw\\_preprocess\\_data\(\)](#), [enw\\_reporting\\_triangle\\_to\\_long\(\)](#), [enw\\_retrospective\(\)](#)

**Examples**

```
obs <- enw_example("preprocessed")$new_confirm
enw_reporting_triangle(obs)
```

---

```
enw_reporting_triangle_to_long
```

*Recast the reporting triangle from wide to long format*

---

**Description**

Recast the reporting triangle from wide to long format

**Usage**

```
enw_reporting_triangle_to_long(obs)
```

**Arguments**

**obs** A data.frame in the format produced by [enw\\_reporting\\_triangle\(\)](#).

**Value**

A long format reporting triangle as a data.frame with additional variables new\_confirm and delay.

**See Also**

Preprocessing functions `enw_add_delay()`, `enw_add_max_reported()`, `enw_add_metaobs_features()`, `enw_assign_group()`, `enw_complete_dates()`, `enw_construct_data()`, `enw_extend_date()`, `enw_filter_delay()`, `enw_filter_reference_dates()`, `enw_filter_reference_dates_by_report_start()`, `enw_filter_report_dates()`, `enw_flag_observed_observations()`, `enw_impute_na_observations()`, `enw_latest_data()`, `enw_metadata()`, `enw_metadata_delay()`, `enw_missing_reference()`, `enw_obs_at_delay()`, `enw_preprocess_data()`, `enw_reporting_triangle()`, `enw_retrospective()`

**Examples**

```
obs <- enw_example("preprocessed")$new_confirm
rt <- enw_reporting_triangle(obs)
enw_reporting_triangle_to_long(rt)
```

---

```
enw_reps_with_complete_refs
```

*Identify report dates with complete (i.e up to the maximum delay) reference dates*

---

**Description**

Identify report dates with complete (i.e up to the maximum delay) reference dates

**Usage**

```
enw_reps_with_complete_refs(new_confirm, max_delay, by = NULL, copy = TRUE)
```

**Arguments**

|                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>new_confirm</code> | <code>new_confirm</code> data.frame output from <code>enw_preprocess_data()</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>max_delay</code>   | <p>The maximum delay to model in the delay distribution, specified in units of the timestep (e.g., if timestep = "week", then max_delay = 3 means 3 weeks). If not specified the maximum observed delay is assumed to be the true maximum delay in the model. Otherwise, an integer greater than or equal to 1 can be specified. Observations with delays larger than the maximum delay will be dropped. If the specified maximum delay is too short, nowcasts can be biased as important parts of the true delay distribution are cut off. At the same time, computational cost scales non-linearly with this setting, so you want the maximum delay to be as long as necessary, but not much longer.</p> <p>Steps to take to determine the maximum delay:</p> <ul style="list-style-type: none"> <li>• Consider what is realistic and relevant for your application.</li> <li>• Check the proportion of observations reported (prop_reported) by delay in the new_confirm output of <code>enw_preprocess_obs</code>.</li> <li>• Use <code>check_max_delay()</code> to check the coverage of a candidate max_delay.</li> <li>• If in doubt, check if increasing the maximum delay noticeably changes the delay distribution or nowcasts as estimated by <code>epinowcast</code>. If it does, your maximum delay may still be too short.</li> </ul> |

|                   |                                                                                                                                                                                                            |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                   | Note that delays are zero indexed and so include the reference date and <code>max_delay - 1</code> other intervals (i.e. a <code>max_delay</code> of 1 corresponds to no delay).                           |
| <code>by</code>   | A character vector describing the stratification of observations. This defaults to no grouping. This should be used when modelling multiple time series in order to identify them for downstream modelling |
| <code>copy</code> | A logical; if TRUE (the default) creates a copy; otherwise, modifies obs in place.                                                                                                                         |

**Value**

A `data.frame` containing a `report_date` variable, and grouping variables specified for report dates that have complete reporting.

**See Also**

Helper functions for model modules `add_max_observed_delay()`, `add_pmfs()`, `convolution_matrix()`, `delay_only_ltotal()`, `delay_only_total()`, `enw_dayofweek_structural_reporting()`, `enw_reference_by_report_date()`, `enw_structural_reporting_metadata()`, `extract_obs_metadata()`, `extract_sparse_matrix()`, `latest_obs_as_matrix()`

---

|                                |                                                          |
|--------------------------------|----------------------------------------------------------|
| <code>enw_retrospective</code> | <i>Convert preprocessed data to retrospective format</i> |
|--------------------------------|----------------------------------------------------------|

---

**Description**

Takes output of `enw_preprocess_data()` and returns a new preprocessed dataset with `max_delay = 1`, suitable for retrospective Rt estimation without delay modelling. Observations are taken at the specified delay (or the latest available) and treated as final counts. In the returned data, `report_date` is set equal to `reference_date` for all rows (i.e. all observations appear to be reported on the same day they occurred).

**Usage**

```
enw_retrospective(data, max_delay = NULL)
```

**Arguments**

|                        |                                                                                                                                     |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| <code>data</code>      | Output of <code>enw_preprocess_data()</code> .                                                                                      |
| <code>max_delay</code> | Integer delay at which to freeze observations. If NULL (default), the latest available observation for each reference date is used. |

**Value**

A preprocessed data object (as from `enw_preprocess_data()`) with `max_delay = 1`.

**See Also**

Preprocessing functions `enw_add_delay()`, `enw_add_max_reported()`, `enw_add_metaobs_features()`, `enw_assign_group()`, `enw_complete_dates()`, `enw_construct_data()`, `enw_extend_date()`, `enw_filter_delay()`, `enw_filter_reference_dates()`, `enw_filter_reference_dates_by_report_start()`, `enw_filter_report_dates()`, `enw_flag_observed_observations()`, `enw_impute_na_observations()`, `enw_latest_data()`, `enw_metadata()`, `enw_metadata_delay()`, `enw_missing_reference()`, `enw_obs_at_delay()`, `enw_preprocess_data()`, `enw_reporting_triangle()`, `enw_reporting_triangle_to_long()`

**Examples**

```
pobs <- enw_example("preprocessed")
enw_retrospective(pobs)
```

---

|                 |                                        |
|-----------------|----------------------------------------|
| enw_rolling_sum | <i>Perform rolling sum aggregation</i> |
|-----------------|----------------------------------------|

---

**Description**

This function takes a `data.table` and applies a rolling sum over a given timestep, aggregating by specified columns. It's particularly useful for aggregating observations over certain periods.

**Usage**

```
enw_rolling_sum(dt, internal_timestep, by = NULL, value_col = "confirm")
```

**Arguments**

|                                |                                                                               |
|--------------------------------|-------------------------------------------------------------------------------|
| <code>dt</code>                | A <code>data.table</code> to be aggregated.                                   |
| <code>internal_timestep</code> | An integer indicating the period over which to aggregate.                     |
| <code>by</code>                | A character vector specifying the columns to aggregate by.                    |
| <code>value_col</code>         | A character string specifying the column to aggregate. Defaults to "confirm". |

**Value**

A modified `data.table` with aggregated observations.

**See Also**

Utility functions `coerce_date()`, `coerce_dt()`, `date_to_numeric_modulus()`, `enw_get_data()`, `get_internal_timestep()`, `is.Date()`, `stan_fns_as_string()`

enw\_sample

*Fit a CmdStan model using NUTS***Description**

Fit a CmdStan model using NUTS

**Usage**

```
enw_sample(
  data,
  model = epinowcast::enw_model(),
  init = NULL,
  init_method = c("prior", "pathfinder"),
  init_method_args = list(),
  diagnostics = TRUE,
  ...
)
```

**Arguments**

|                               |                                                                                                                                                                                                                                |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>data</code>             | A list of data as produced by model modules (for example <a href="#">enw_expectation()</a> , <a href="#">enw_obs()</a> , etc.) and as required for use the model being used.                                                   |
| <code>model</code>            | A cmdstanr model object as loaded by <a href="#">enw_model()</a> or as supplied by the user.                                                                                                                                   |
| <code>init</code>             | A list of initial values or a function to generate initial values. If not provided, the model will attempt to generate initial values                                                                                          |
| <code>init_method</code>      | The method to use for initializing the model. Defaults to "prior" which samples initial values from the prior. "pathfinder", which uses the pathfinder algorithm ( <a href="#">enw_pathfinder()</a> ) to initialize the model. |
| <code>init_method_args</code> | A list of additional arguments to pass to the initialization method.                                                                                                                                                           |
| <code>diagnostics</code>      | Logical, defaults to TRUE. Should fitting diagnostics be returned as a data.frame.                                                                                                                                             |
| <code>...</code>              | Additional parameters passed to the sample method of cmdstanr.                                                                                                                                                                 |

**Value**

A data.frame containing the cmdstanr fit, the input data, the fitting arguments, and optionally summary diagnostics.

**See Also**

Functions used to help convert models into the format required for stan [enw\\_formula\\_as\\_data\\_list\(\)](#), [enw\\_get\\_cache\(\)](#), [enw\\_model\(\)](#), [enw\\_pathfinder\(\)](#), [enw\\_priors\\_as\\_data\\_list\(\)](#), [enw\\_replace\\_priors\(\)](#), [enw\\_set\\_cache\(\)](#), [enw\\_stan\\_to\\_r\(\)](#), [enw\\_unset\\_cache\(\)](#), [remove\\_profiling\(\)](#), [write\\_stan\\_files\\_no\\_profile\(\)](#)

## Examples

```
pobs <- enw_example("preprocessed")

nowcast <- epinowcast(pobs,
  expectation = enw_expectation(~1, data = pobs),
  fit = enw_fit_opts(enw_sample, pp = TRUE),
  obs = enw_obs(family = "poisson", data = pobs),
)

summary(nowcast)

# Use pathfinder initialization
nowcast_pathfinder <- epinowcast(pobs,
  expectation = enw_expectation(~1, data = pobs),
  fit = enw_fit_opts(enw_sample, pp = TRUE, init_method = "pathfinder"),
  obs = enw_obs(family = "poisson", data = pobs),
)

summary(nowcast_pathfinder)
```

---

enw\_set\_cache

Set caching location for Stan models

---

## Description

This function allows the user to set a cache location for Stan models rather than a temporary directory. This can reduce the need for model compilation on every new model run across sessions or within a session. For R version 4.0.0 and above, it's recommended to use the persistent cache as shown in the example.

## Usage

```
enw_set_cache(path, type = c("session", "persistent", "all"))
```

## Arguments

|      |                                                                                                                                                                                                                                                                     |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| path | A valid filepath representing the desired cache location. If the directory does not exist it will be created.                                                                                                                                                       |
| type | A character string specifying the cache type. It can be one of "session", "persistent", or "all". Default is "session". "session" sets the cache for the current session, "persistent" writes the cache location to the user's .Renvirom file, and "all" does both. |

## Value

The string of the filepath set.

**See Also**

Functions used to help convert models into the format required for stan `enw_formula_as_data_list()`, `enw_get_cache()`, `enw_model()`, `enw_pathfinder()`, `enw_priors_as_data_list()`, `enw_replace_priors()`, `enw_sample()`, `enw_stan_to_r()`, `enw_unset_cache()`, `remove_profiling()`, `write_stan_files_no_profile()`

**Examples**

```
# Set to local directory
my_enw_cache <- enw_set_cache(file.path(tempdir(), "test"))
enw_get_cache()
## Not run:
# Use the package cache in R >= 4.0
if (R.version.string >= "4.0.0") {
  enw_set_cache(
    tools::R_user_dir(package = "epinowcast", "cache"),
    type = "all"
  )
}

## End(Not run)
```

---

`enw_simulate_missing_reference`

*Simulate observations with a missing reference date.*

---

**Description**

A simple binomial simulator of missing data by reference date using simulated or observed data as an input. This function may be used to validate missing data models, as part of examples and case studies, or to explore the implications of missing data for your use case.

**Usage**

```
enw_simulate_missing_reference(obs, proportion = 0.2, by = NULL)
```

**Arguments**

|                         |                                                                                                                                                                                                                    |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>obs</code>        | A data.frame containing at least the following variables: reference date (index date of interest), report_date (report date for observations), and confirm (cumulative observations by reference and report date). |
| <code>proportion</code> | Numeric, the proportion of observations that are missing a reference date, indexed by reference date. Currently only a fixed proportion are supported and this defaults to 0.2.                                    |
| <code>by</code>         | A character vector describing the stratification of observations. This defaults to no grouping. This should be used when modelling multiple time series in order to identify them for downstream modelling         |

**Value**

A data.table of the same format as the input but with a simulated proportion of observations now having a missing reference date.

**Examples**

```
# Load and filter germany hospitalisations
nat_germany_hosp <- subset(
  germany_covid19_hosp, location == "DE" & age_group == "00+"
)
nat_germany_hosp <- enw_filter_report_dates(
  nat_germany_hosp,
  latest_date = "2021-08-01"
)

# Make sure observations are complete
nat_germany_hosp <- enw_complete_dates(
  nat_germany_hosp,
  by = c("location", "age_group"), missing_reference = FALSE
)

# Simulate
enw_simulate_missing_reference(
  nat_germany_hosp,
  proportion = 0.35, by = c("location", "age_group")
)
```

---

enw\_stan\_to\_r

---

*Expose epinowcast stan functions in R*


---

**Description**

This function facilitates the exposure of Stan functions from the **epinowcast** package in R. It utilizes the [expose\\_functions](#) method of `cmdstanr::CmdStanModel` for this purpose. This function is useful for developers and contributors to the **epinowcast** package, as well as for users interested in exploring and prototyping with model functionalities.

**Usage**

```
enw_stan_to_r(
  files = list.files(include, pattern = "\\..stan$"),
  include = system.file("stan", "functions", package = "epinowcast"),
  global = TRUE,
  verbose = TRUE,
  ...
)
```

**Arguments**

|         |                                                                                                                                                                                                                                                                                                                                                     |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| files   | A character vector specifying the names of Stan files to be exposed. These must be in the include directory. Defaults to all Stan files in the include directory. Note that the following files contain overloaded functions and cannot be exposed: "delay_lpmf.stan", "allocate_observed_obs.stan", "obs_lpmf.stan", and "effects_priors_lp.stan". |
| include | A character string specifying the directory containing Stan files. Defaults to the 'stan/functions' directory of the <code>epinowcast()</code> package.                                                                                                                                                                                             |
| global  | A logical value indicating whether to expose the functions globally. Defaults to TRUE. Passed to the <code>expose_functions</code> method of <code>CmdStanModel</code> .                                                                                                                                                                            |
| verbose | Logical, defaults to TRUE. Should verbose messages be shown.                                                                                                                                                                                                                                                                                        |
| ...     | Additional arguments passed to <code>enw_model</code> .                                                                                                                                                                                                                                                                                             |

**Value**

An object of class `CmdStanModel` with functions from the model exposed for use in R.

**See Also**

Functions used to help convert models into the format required for stan `enw_formula_as_data_list()`, `enw_get_cache()`, `enw_model()`, `enw_pathfinder()`, `enw_priors_as_data_list()`, `enw_replace_priors()`, `enw_sample()`, `enw_set_cache()`, `enw_unset_cache()`, `remove_profiling()`, `write_stan_files_no_profile()`

**Examples**

```
# Compile functions in stan/functions/hazard.stan
stan_functions <- enw_stan_to_r("hazard.stan")
# These functions can now be used in R
stan_functions$functions$prob_to_hazard(c(0.5, 0.1, 0.1))
# or exposed globally and used directly
prob_to_hazard(c(0.5, 0.1, 0.1))
```

---

enw\_structural\_reporting\_metadata

*Create structural reporting metadata grid*

---

**Description**

Creates a base metadata grid for structural reporting patterns by generating all combinations of reference dates, delays, and report dates. This grid serves as the foundation for defining custom reporting patterns.

**Usage**

```
enw_structural_reporting_metadata(pobs)
```

**Arguments**

`pobs` A preprocessed observation list from `enw_preprocess_data()`.

**Value**

A data.table with columns:

- `.group`: Group identifier
- `date`: Reference date
- `report_date`: Report date (reference date + delay)

**See Also**

Helper functions for model modules `add_max_observed_delay()`, `add_pmfs()`, `convolution_matrix()`, `delay_only_ltotal()`, `delay_only_total()`, `enw_dayofweek_structural_reporting()`, `enw_reference_by_report`, `enw_reps_with_complete_refs()`, `extract_obs_metadata()`, `extract_sparse_matrix()`, `latest_obs_as_matrix()`

**Examples**

```
## Not run:
pobs <- enw_preprocess_data(obs, max_delay = 30)
metadata <- enw_structural_reporting_metadata(pobs)

# Add custom reporting pattern (e.g., only report on first day of month)
metadata[, report := as.integer(format(report_date, "%d") == "01")]

## End(Not run)
```

---

`enw_summarise_samples` *Summarise posterior samples*

---

**Description**

This function summarises posterior samples for arbitrary strata. It optionally holds out the observed data (variables that are not `".draw"`, `".iteration"`, `".sample"`, `".chain"`) joins this to the summarised posterior.

**Usage**

```
enw_summarise_samples(
  samples,
  probs = c(0.05, 0.2, 0.35, 0.5, 0.65, 0.8, 0.95),
  by = c("reference_date", ".group"),
  link_with_obs = TRUE
)
```

**Arguments**

|                            |                                                                                                                                                                                                       |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>samples</code>       | A data.frame of posterior samples with at least a numeric sample variable.                                                                                                                            |
| <code>probs</code>         | A vector of numeric probabilities to produce quantile summaries for. By default these are the 5%, 20%, 80%, and 95% quantiles which are also the minimum set required for plotting functions to work. |
| <code>by</code>            | A character vector of variables to summarise by. Defaults to <code>c("reference_date", ".group")</code> .                                                                                             |
| <code>link_with_obs</code> | Logical, should the observed data be linked to the posterior summary? This is useful for plotting the posterior against the observed data. Defaults to TRUE.                                          |

**Value**

A data.frame summarising the posterior samples.

**See Also**

Functions used for postprocessing of model fits `.check_primarycensored()`, `.delay_draw_columns()`, `.discretise_parametric_pmf()`, `build_ord_obs()`, `enw_add_latest_obs_to_nowcast()`, `enw_nowcast_samples()`, `enw_nowcast_summary()`, `enw_posterior()`, `enw_posterior_delay()`, `enw_pp_summary()`, `enw_quantiles_to_long()`, `subset_obs()`

**Examples**

```
fit <- enw_example("nowcast")
samples <- summary(fit, type = "nowcast_sample")
enw_summarise_samples(samples, probs = c(0.05, 0.5, 0.95))
```

---

|                              |                                  |
|------------------------------|----------------------------------|
| <code>enw_unset_cache</code> | <i>Unset Stan cache location</i> |
|------------------------------|----------------------------------|

---

**Description**

Optionally removes the `enw_cache_location` environment variable from the user .Renviro file and/or removes it from the local environment. If you unset the local cache and want to switch back to using the persistent cache, you can reload the .Renviro file using `readRenviro("~/Renviro")`.

**Usage**

```
enw_unset_cache(type = c("session", "persistent", "all"))
```

**Arguments**

|                   |                                                                                                                                                                                                                                                                                            |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>type</code> | A character string specifying the type of cache to unset. It can be one of "session", "persistent", or "all". Default is "session". "session" unsets the cache for the current session, "persistent" removes the cache location from the user's .Renviro file, and "all" does all options. |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Value**

The prior cache location, if it existed otherwise NULL.

**See Also**

Functions used to help convert models into the format required for stan `enw_formula_as_data_list()`, `enw_get_cache()`, `enw_model()`, `enw_pathfinder()`, `enw_priors_as_data_list()`, `enw_replace_priors()`, `enw_sample()`, `enw_set_cache()`, `enw_stan_to_r()`, `remove_profiling()`, `write_stan_files_no_profile()`

**Examples**

```
enw_unset_cache()
```

---

epinowcast

*Nowcast using partially observed data*


---

**Description**

Provides a user friendly interface around package functionality to produce a nowcast from observed preprocessed data, and a series of user defined models. By default a model that assumes a fixed parametric reporting distribution with a flexible expectation model is used. Explore the individual model components for additional documentation and see the package case studies for example model specifications for different tasks.

**Usage**

```
epinowcast(
  data,
  reference = epinowcast::enw_reference(parametric = ~1, distribution = "lognormal",
    non_parametric = ~0, data = data),
  report = epinowcast::enw_report(non_parametric = ~0, structural = NULL, data = data),
  expectation = epinowcast::enw_expectation(r = ~0 + (1 | day:.group), generation_time =
    1, observation = ~1, latent_reporting_delay = 1, data = data),
  missing = epinowcast::enw_missing(formula = ~0, data = data),
  obs = epinowcast::enw_obs(family = "negbin", data = data),
  fit = epinowcast::enw_fit_opts(sampler = epinowcast::enw_sample, nowcast = TRUE, pp =
    FALSE, likelihood = TRUE, debug = FALSE, output_loglik = FALSE),
  model = epinowcast::enw_model(),
  priors,
  ...
)
```

**Arguments**

|                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>data</code>        | Output from <code>enw_preprocess_data()</code> .                                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>reference</code>   | The reference date indexed reporting process model specification as defined using <code>enw_reference()</code> .                                                                                                                                                                                                                                                                                                                                        |
| <code>report</code>      | The report date indexed reporting process model specification as defined using <code>enw_report()</code> .                                                                                                                                                                                                                                                                                                                                              |
| <code>expectation</code> | The expectation model specification as defined using <code>enw_expectation()</code> . By default this is set to be a highly flexible random effect by reference date for each group and thus weakly informed. Depending on your context (and in particular the density of data reporting) other choices that enforce more assumptions may be more appropriate (for example a weekly random walk (specified using <code>rw(week, by = .group)</code> )). |
| <code>missing</code>     | The missing reference date model specification as defined using <code>enw_missing()</code> . By default this is set to not be used.                                                                                                                                                                                                                                                                                                                     |
| <code>obs</code>         | The observation model as defined by <code>enw_obs()</code> . Observations are also processed within this function for use in modelling.                                                                                                                                                                                                                                                                                                                 |
| <code>fit</code>         | Model fit options as defined using <code>enw_fit_opts()</code> . This includes the sampler function to use (with the package default being <code>enw_sample()</code> ), whether or now a nowcast should be used, etc. See <code>enw_fit_opts()</code> for further details.                                                                                                                                                                              |
| <code>model</code>       | The model to use within <code>fit</code> . By default this uses <code>enw_model()</code> .                                                                                                                                                                                                                                                                                                                                                              |
| <code>priors</code>      | A <code>data.frame</code> with columns <code>variable</code> , <code>mean</code> , and <code>sd</code> describing normal priors that replace the module defaults. Custom priors are merged with the defaults automatically. See Details for how to inspect and modify priors.                                                                                                                                                                           |
| <code>...</code>         | Additional model modules to pass to <code>model</code> . User modules may be used but currently require the supplied model to be adapted.                                                                                                                                                                                                                                                                                                               |

**Details****Priors:**

Each model module defines its own default priors. To inspect them, call the module function and access the `$priors` element. See `enw_reference()`, `enw_report()`, `enw_expectation()`, `enw_missing()`, and `enw_obs()` for the prior variables available in each module.

To replace specific defaults, pass a `data.frame` to the `priors` argument. Vectorised prior names (e.g. `"refp_mean_int[1]"`) are matched after stripping the index. See `enw_replace_priors()` for details on the merging behaviour.

**Value**

A object of the class "epinowcast" which inherits from `enw_preprocess_data()` and `data.table`, and combines the input data, priors, and output from the sampler specified in `enw_fit_opts()`.

**See Also**

Other epinowcast: `plot.enw_preprocess_data()`, `plot.epinowcast()`, `print.enw_preprocess_data()`, `print.epinowcast()`, `print.summary.enw_preprocess_data()`, `summary.enw_preprocess_data()`, `summary.epinowcast()`

## Examples

```
# Load data.table and ggplot2
library(data.table)
library(ggplot2)

# Use 2 cores
options(mc.cores = 2)
# Load and filter germany hospitalisations
nat_germany_hosp <-
  germany_covid19_hosp[location == "DE"][age_group == "00+"]
nat_germany_hosp <- enw_filter_report_dates(
  nat_germany_hosp,
  latest_date = "2021-10-01"
)
# Make sure observations are complete
nat_germany_hosp <- enw_complete_dates(
  nat_germany_hosp,
  by = c("location", "age_group")
)
# Make a retrospective dataset
retro_nat_germany <- enw_filter_report_dates(
  nat_germany_hosp,
  remove_days = 40
)
retro_nat_germany <- enw_filter_reference_dates(
  retro_nat_germany,
  include_days = 40
)
# Get latest observations for the same time period
latest_obs <- nat_germany_hosp |>
  enw_obs_at_delay(max_delay = 20) |>
  enw_filter_reference_dates(
    remove_days = 40, include_days = 20
  )
# Preprocess observations (note this maximum delay is likely too short)
pobs <- enw_preprocess_data(retro_nat_germany, max_delay = 20)

# Fit with custom priors
my_priors <- data.frame(
  variable = "refp_mean_int", mean = 2, sd = 0.5
)
nowcast <- epinowcast(pobs,
  priors = my_priors,
  fit = enw_fit_opts(
    save_warmup = FALSE, pp = TRUE,
    chains = 2, iter_warmup = 500, iter_sampling = 500
  )
)
nowcast
# plot the nowcast vs latest available observations
plot(nowcast, latest_obs = latest_obs)
```

```
# plot posterior predictions for the delay distribution by date
plot(nowcast, type = "posterior") +
  facet_wrap(vars(reference_date), scale = "free")
```

---

extract\_obs\_metadata    *Extract observation metadata*

---

## Description

This function extracts metadata from the provided dataset to be used in the observation model.

## Usage

```
extract_obs_metadata(new_confirm, observation_indicator = NULL)
```

## Arguments

**new\_confirm**      A data.table containing the columns: "reference\_date", "delay", ".group", "new\_confirm", and "max\_obs\_delay". As produced by [enw\\_preprocess\\_data\(\)](#) in the new\_confirm output with the addition of the "max\_obs\_delay" column as produced by [add\\_max\\_observed\\_delay\(\)](#).

**observation\_indicator**      A character string specifying the column name in new\_confirm that indicates whether an observation is observed or not. This column should be a logical vector. If NULL (default), all observations are considered observed.

## Value

A list containing:

- st: time index of each snapshot (snapshot time).
- ts: snapshot index by time and group.
- sl: number of reported observations per snapshot (snapshot length).
- csl: cumulative version of sl.
- lsl: number of consecutive reported observations per snapshot accounting for missing data.
- clsl: cumulative version of lsl.
- nsl: number of observed observations per snapshot (snapshot length).
- cnsl: cumulative version of nsl.
- sg: group index of each snapshot (snapshot group).

## See Also

Helper functions for model modules [add\\_max\\_observed\\_delay\(\)](#), [add\\_pmfs\(\)](#), [convolution\\_matrix\(\)](#), [delay\\_only\\_ltotal\(\)](#), [delay\\_only\\_total\(\)](#), [enw\\_dayofweek\\_structural\\_reporting\(\)](#), [enw\\_reference\\_by\\_report\(\)](#), [enw\\_reps\\_with\\_complete\\_refs\(\)](#), [enw\\_structural\\_reporting\\_metadata\(\)](#), [extract\\_sparse\\_matrix\(\)](#), [latest\\_obs\\_as\\_matrix\(\)](#)

---

extract\_sparse\_matrix *Extract sparse matrix elements*

---

## Description

This helper function allows the extraction of a sparse matrix from a matrix using a similar approach to that implemented in `rstan::extract_sparse_parts()` and returns these elements in a named list for use in stan. This function is used in the construction of the expectation model (see `enw_expectation()`).

## Usage

```
extract_sparse_matrix(mat, prefix = "")
```

## Arguments

|        |                                                              |
|--------|--------------------------------------------------------------|
| mat    | A matrix to extract the sparse matrix from.                  |
| prefix | A character string to prefix the names of the returned list. |

## Value

A list representing the sparse matrix, containing:

- nw: Count of non-zero elements in mat.
- w: Vector of non-zero elements in mat. Equivalent to the numeric values from mat excluding zeros.
- nv: Length of v.
- v: Vector of row indices corresponding to each non-zero element in w. Indicates the row location in mat for each non-zero value.
- nu: Length of u.
- u: Vector indicating the starting indices in w for non-zero elements of each row in mat. Helps identify the partition of w into different rows of mat.

## See Also

`enw_expectation()`

Helper functions for model modules `add_max_observed_delay()`, `add_pmfs()`, `convolution_matrix()`, `delay_only_ltotal()`, `delay_only_total()`, `enw_dayofweek_structural_reporting()`, `enw_reference_by_report`, `enw_reps_with_complete_refs()`, `enw_structural_reporting_metadata()`, `extract_obs_metadata()`, `latest_obs_as_matrix()`

## Examples

```
mat <- matrix(1:12, nrow = 4)
mat[2, 2] <- 0
mat[3, 1] <- 0
extract_sparse_matrix(mat)
```

---

germany\_covid19\_hosp    *Hospitalisations in Germany by date of report and reference*

---

**Description**

Hospitalisations in Germany by date of report and reference

**Usage**

```
germany_covid19_hosp
```

**Format**

An object of class `data.table` (inherits from `data.frame`) with 1536885 rows and 5 columns.

**Value**

A `data.table`

**See Also**

Package data sets [enw\\_example\(\)](#)

---

get\_internal\_timestep    *Get internal timestep*

---

**Description**

This function converts the string representation of the timestep to its corresponding numeric value or returns the numeric input (if it is a whole number). For "day", "week", it returns 1 and 7 respectively. "month" is not supported and will throw an error. If the input is a numeric whole number, it is returned as is.

**Usage**

```
get_internal_timestep(timestep)
```

**Arguments**

|          |                                                                                                                                                                                                                              |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| timestep | The timestep to used. This can be a string ("day", "week") or a numeric whole number representing the number of days. Note that "month" is not currently supported in user-facing functions and will throw an error if used. |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Value**

A numeric value representing the number of days for "day" and "week", or the input value if it is a numeric whole number.

**See Also**

Utility functions `coerce_date()`, `coerce_dt()`, `date_to_numeric_modulus()`, `enw_get_data()`, `enw_rolling_sum()`, `is.Date()`, `stan_fns_as_string()`

gp

*Adds an approximate Gaussian process to the model.***Description**

A call to `gp()` can be used in the formula argument of model construction functions in the `epinowcast` package such as `enw_formula()`. It declares a Hilbert-space reduced-rank (spectral) approximate Gaussian process indexed by time (and optionally a grouping variable by) whose value at each observation is added to the linear predictor. As with `arima()`, arguments are not evaluated; they are passed by name for use in model construction.

Like `arima()` and `rw()`, a `gp()` term works on every module that takes a formula, each with its own prior prefix:

- `enw_expectation()` — the growth rate (`expr`) and the latent-to-obs proportion (`expl`).
- `enw_reference()` — the parametric delay mean (`refp`) and the non-parametric logit hazards (`refnp`).
- `enw_report()` — report-date logit hazards (`rep`).
- `enw_missing()` — the missing-reference proportion (`miss`).

At most one `gp()` term is currently supported per formula (the multiple-term example shown for `gp_terms()` only illustrates term detection, not a supported model). The default alpha (magnitude) and length-scale priors are inherited from `EpiNow2` and are set on `EpiNow2`'s scale; on a given module's scale (for example the log growth rate or a logit hazard) they may need tuning with `enw_replace_priors()`.

**Usage**

```
gp(
  time,
  by,
  d = 0,
  kernel = c("matern32", "matern52", "ou", "se", "periodic"),
  basis_prop = 0.2,
  boundary_scale = 1.5
)
```

**Arguments**

|                   |                                                                                                                                                                                                                        |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>time</code> | Defines the time index of the Gaussian process. Must be numeric.                                                                                                                                                       |
| <code>by</code>   | Optional grouping variable. If supplied, an independent Gaussian process is fitted for each level of <code>by</code> (sharing the length scale and magnitude hyperparameters). Currently limited to a single variable. |

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| d              | Non-negative integer, defaults to 0. Order of differencing, matching the d of <code>arima()</code> : the per-group realisation is integrated (cumulative-summed) d times before it is added to the predictor. d = 0 gives stationary deviations (the default, equivalent to EpiNow2's <code>gp_on = "R0"</code> ). d = 1 integrates once, giving a smoothly drifting, random-walk-like trajectory (equivalent to EpiNow2's default <code>gp_on = "R_t-1"</code> ). For d >= 1 the first d values of the realisation are anchored to zero, so the free level (and, for d >= 2, slope) is carried by the module's fixed effects rather than the GP. Differencing is intended for the latent expectation modules (the growth rate <code>expr</code> and latent-to-obs proportion <code>expl</code> ); integrating a logit-hazard term ( <code>refnp</code> , <code>rep</code> , <code>miss</code> ) is unusual but permitted for API consistency with <code>arima()</code> . |
| kernel         | Character string selecting the covariance kernel. One of "matern32" (the default, a Matern 3/2 kernel), "matern52" (Matern 5/2), "ou" (Ornstein-Uhlenbeck, equivalent to Matern 1/2), "se" (squared exponential), or "periodic".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| basis_prop     | Numeric in (0, 1]. Proportion of time points to use as basis functions, controlling the accuracy-speed trade-off of the reduced-rank approximation. Defaults to 0.2 (the EpiNow2 default). The number of basis functions is <code>ceiling(basis_prop * T)</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| boundary_scale | Numeric, defaults to 1.5. Boundary factor L of the Hilbert-space approximation; the process is approximated on the interval scaled by this factor. This has no effect when <code>kernel = "periodic"</code> , which uses a fundamental-frequency basis rather than the boundary-scaled basis.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

### Value

A list of class `enw_gp_term` describing the Gaussian process term, interpretable by `construct_gp()`.

### Reference

The Stan implementation of the approximate Gaussian process is adapted from EpiNow2 (<https://github.com/epiforecasts/EpiNow2> MIT licensed). The Hilbert-space approximation follows Riutort-Mayol et al. (2023), doi:10.1007/s11222-022-10167-2.

### See Also

Functions used to help convert formulas into model designs `ar()`, `arima()`, `arima_terms()`, `arma()`, `as_string_formula()`, `construct_arima()`, `construct_gp()`, `construct_re()`, `construct_rw()`, `enw_formula()`, `enw_manual_formula()`, `gp_terms()`, `ma()`, `parse_formula()`, `re()`, `remove_arima_terms()`, `remove_gp_terms()`, `remove_rw_terms()`, `rw()`, `rw_terms()`, `split_formula_to_terms()`

### Examples

```
gp(time)
gp(time, location)
gp(time, kernel = "se", basis_prop = 0.3)
gp(time, d = 1)
```

---

 gp\_terms

*Finds Gaussian process terms in a formula object*


---

### Description

This function extracts Gaussian process terms denoted using `gp()` from a formula so that they can be processed on their own.

### Usage

```
gp_terms(formula)
```

### Arguments

`formula` A model formula that may use standard fixed effects, random effects using `lme4` syntax (see `re()`), and random walks defined using the `rw()` helper function. See the Details section below for a comprehensive explanation of the supported syntax.

### Value

A character vector containing the Gaussian process terms identified in the supplied formula.

### See Also

Functions used to help convert formulas into model designs `ar()`, `arma()`, `arma_terms()`, `arma()`, `as_string_formula()`, `construct_arma()`, `construct_gp()`, `construct_re()`, `construct_rw()`, `enw_formula()`, `enw_manual_formula()`, `gp()`, `ma()`, `parse_formula()`, `re()`, `remove_arma_terms()`, `remove_gp_terms()`, `remove_rw_terms()`, `rw()`, `rw_terms()`, `split_formula_to_terms()`

### Examples

```
epinowcast::gp_terms(~ 1 + age_group + gp(week))
epinowcast::gp_terms(~ 1 + gp(week, kernel = "se") + gp(day))
```

---

 is.Date

*Check an object is a Date*


---

### Description

Checks that an object is a date

### Usage

```
is.Date(x)
```

**Arguments**

`x`                      An object

**Value**

A logical

**See Also**

Utility functions [coerce\\_date\(\)](#), [coerce\\_dt\(\)](#), [date\\_to\\_numeric\\_modulus\(\)](#), [enw\\_get\\_data\(\)](#), [enw\\_rolling\\_sum\(\)](#), [get\\_internal\\_timestep\(\)](#), [stan\\_fns\\_as\\_string\(\)](#)

---

`latest_obs_as_matrix`    *Convert latest observed data to a matrix*

---

**Description**

Convert latest observed data to a matrix

**Usage**

```
latest_obs_as_matrix(latest)
```

**Arguments**

`latest`                      latest data.frame output from [enw\\_preprocess\\_data\(\)](#).

**Value**

A matrix with each column being a group and each row a reference date

**See Also**

Helper functions for model modules [add\\_max\\_observed\\_delay\(\)](#), [add\\_pmfs\(\)](#), [convolution\\_matrix\(\)](#), [delay\\_only\\_ltotal\(\)](#), [delay\\_only\\_total\(\)](#), [enw\\_dayofweek\\_structural\\_reporting\(\)](#), [enw\\_reference\\_by\\_report](#), [enw\\_reps\\_with\\_complete\\_refs\(\)](#), [enw\\_structural\\_reporting\\_metadata\(\)](#), [extract\\_obs\\_metadata\(\)](#), [extract\\_sparse\\_matrix\(\)](#)

---

|    |                                                         |
|----|---------------------------------------------------------|
| ma | <i>Moving-average alias for <a href="#">arima()</a></i> |
|----|---------------------------------------------------------|

---

### Description

Thin wrapper around [arima\(\)](#) that fixes  $p = 0$  and  $d = 0$ . Equivalent to `arima(time, by, p = 0, d = 0, q = q)`.

### Usage

```
ma(time, by, q = 1)
```

### Arguments

|      |                                                                                                                                                        |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| time | Time variable for the latent series; numeric.                                                                                                          |
| by   | Optional grouping variable. Each group draws an independent shock series; AR/MA parameters and the latent standard deviation are shared across groups. |
| q    | Moving-average order. Defaults to 1.                                                                                                                   |

### Value

An `enw_arima_term` interpretable by [construct\\_arima\(\)](#).

### See Also

Functions used to help convert formulas into model designs [ar\(\)](#), [arima\(\)](#), [arima\\_terms\(\)](#), [arma\(\)](#), [as\\_string\\_formula\(\)](#), [construct\\_arima\(\)](#), [construct\\_gp\(\)](#), [construct\\_re\(\)](#), [construct\\_rw\(\)](#), [enw\\_formula\(\)](#), [enw\\_manual\\_formula\(\)](#), [gp\(\)](#), [gp\\_terms\(\)](#), [parse\\_formula\(\)](#), [re\(\)](#), [remove\\_arima\\_terms\(\)](#), [remove\\_gp\\_terms\(\)](#), [remove\\_rw\\_terms\(\)](#), [rw\(\)](#), [rw\\_terms\(\)](#), [split\\_formula\\_to\\_terms\(\)](#)

### Examples

```
ma(time)
ma(time, location, q = 2)
```

---

|               |                                        |
|---------------|----------------------------------------|
| parse_formula | <i>Parse a formula into components</i> |
|---------------|----------------------------------------|

---

### Description

This function uses a series internal functions to break an input formula into its component parts each of which can then be handled separately. Currently supported components are fixed effects, [lme4](#) style random effects, and random walks using the [rw\(\)](#) helper function.

## Usage

```
parse_formula(formula)
```

## Arguments

**formula** A model formula that may use standard fixed effects, random effects using [lme4](#) syntax (see [re\(\)](#)), and random walks defined using the [rw\(\)](#) helper function. See the Details section below for a comprehensive explanation of the supported syntax.

## Value

A list of formula components. These currently include:

- **fixed**: A character vector of fixed effect terms
- **random**: A list of of [lme4](#) style random effects
- **rw**: A character vector of [rw\(\)](#) random walk terms.
- **arima**: A character vector of [arima\(\)](#) ARIMA(p, d, q) terms.
- **gp**: A character vector of [gp\(\)](#) Gaussian process terms.

## Reference

The random walk functions used internally by this function were adapted from code written by J Scott (under an MIT license) as part of the [epidemia](https://github.com/ImperialCollegeLondon/epidemia/) package (<https://github.com/ImperialCollegeLondon/epidemia/>).

## See Also

Functions used to help convert formulas into model designs [ar\(\)](#), [arima\(\)](#), [arima\\_terms\(\)](#), [arma\(\)](#), [as\\_string\\_formula\(\)](#), [construct\\_arima\(\)](#), [construct\\_gp\(\)](#), [construct\\_re\(\)](#), [construct\\_rw\(\)](#), [enw\\_formula\(\)](#), [enw\\_manual\\_formula\(\)](#), [gp\(\)](#), [gp\\_terms\(\)](#), [ma\(\)](#), [re\(\)](#), [remove\\_arima\\_terms\(\)](#), [remove\\_gp\\_terms\(\)](#), [remove\\_rw\\_terms\(\)](#), [rw\(\)](#), [rw\\_terms\(\)](#), [split\\_formula\\_to\\_terms\(\)](#)

## Examples

```
epinowcast:::parse_formula(~ 1 + age_group + location)

epinowcast:::parse_formula(~ 1 + age_group + (1 | location))

epinowcast:::parse_formula(~ 1 + (age_group | location))

epinowcast:::parse_formula(~ 1 + (1 | location) + rw(week, location))
```

---

plot.enw\_preprocess\_data

*Plot method for enw\_preprocess\_data*


---

## Description

plot method for preprocessed data of class "enw\_preprocess\_data". Creates descriptive plots of the empirical reporting delay distribution and notification time series.

## Usage

```
## S3 method for class 'enw_preprocess_data'
plot(
  x,
  type = c("obs", "delay_cumulative", "delay_fraction", "delay_quantiles",
    "delay_counts"),
  delay_group_thresh = NULL,
  quantiles = c(0.1, 0.5, 0.9),
  log = FALSE,
  facet = TRUE,
  ...
)
```

## Arguments

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x                  | A preprocessed data object as produced by <a href="#">enw_preprocess_data()</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| type               | Character string indicating the plot type; enforced by <a href="#">base::match.arg()</a> . Options: <ul style="list-style-type: none"> <li>• "obs" – latest observations (via <a href="#">enw_plot_obs()</a>)</li> <li>• "delay_cumulative" – cumulative empirical delay (via <a href="#">enw_plot_delay_cumulative()</a>)</li> <li>• "delay_fraction" – delay heatmap (via <a href="#">enw_plot_delay_fraction()</a>)</li> <li>• "delay_quantiles" – delay quantiles (via <a href="#">enw_plot_delay_quantiles()</a>)</li> <li>• "delay_counts" – notifications by delay group (via <a href="#">enw_plot_delay_counts()</a>)</li> </ul> |
| delay_group_thresh | A numeric vector of left-closed interval thresholds for delay grouping (use <code>right = FALSE</code> semantics, so the upper bound should exceed <code>max_delay</code> ). Used by "delay_cumulative", "delay_fraction", and "delay_counts". Defaults to <code>NULL</code> , which auto-generates thresholds from <code>max_delay</code> .                                                                                                                                                                                                                                                                                             |
| quantiles          | A numeric vector of probabilities for the "delay_quantiles" type. Defaults to <code>c(0.1, 0.5, 0.9)</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| log                | Logical, defaults to <code>FALSE</code> . Should counts be plotted on the log scale (only for "obs" type).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| facet              | Logical. When <code>TRUE</code> (the default), delay-based plots with more than one <code>.group</code> are automatically wrapped by group. Set to <code>FALSE</code> to disable and add a custom facet layer.                                                                                                                                                                                                                                                                                                                                                                                                                           |
| ...                | Additional arguments passed to the underlying plot function.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

**Value**

A ggplot2 object.

**See Also**

Other epinowcast: [epinowcast\(\)](#), [plot.epinowcast\(\)](#), [print.enw\\_preprocess\\_data\(\)](#), [print.epinowcast\(\)](#), [print.summary.enw\\_preprocess\\_data\(\)](#), [summary.enw\\_preprocess\\_data\(\)](#), [summary.epinowcast\(\)](#)

Plotting functions [enw\\_delay\\_categories\(\)](#), [enw\\_delay\\_quantiles\(\)](#), [enw\\_plot\\_delay\\_counts\(\)](#), [enw\\_plot\\_delay\\_cumulative\(\)](#), [enw\\_plot\\_delay\\_fraction\(\)](#), [enw\\_plot\\_delay\\_quantiles\(\)](#), [enw\\_plot\\_nowcast\\_quantiles\(\)](#), [enw\\_plot\\_obs\(\)](#), [enw\\_plot\\_pp\\_quantiles\(\)](#), [enw\\_plot\\_quantiles\(\)](#), [enw\\_plot\\_theme\(\)](#), [plot.epinowcast\(\)](#)

**Examples**

```
pobs <- enw_example("preprocessed_observations")

# Latest observations
plot(pobs, type = "obs")

# Cumulative reporting delay
plot(pobs, type = "delay_cumulative")

# Reporting delay heatmap
plot(pobs, type = "delay_fraction")

# Reporting delay quantiles
plot(pobs, type = "delay_quantiles")

# Notifications by delay group
plot(pobs, type = "delay_counts")
```

---

|                 |                                   |
|-----------------|-----------------------------------|
| plot.epinowcast | <i>Plot method for epinowcast</i> |
|-----------------|-----------------------------------|

---

**Description**

plot method for class "epinowcast".

**Usage**

```
## S3 method for class 'epinowcast'
plot(
  x,
  latest_obs = NULL,
  type = c("nowcast", "posterior_prediction"),
  log = FALSE,
  ...
)
```

**Arguments**

|            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x          | A data.table of output as produced by <code>epinowcast()</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| latest_obs | A data.frame of observed data which may be passed to lower level methods.                                                                                                                                                                                                                                                                                                                                                                                                                               |
| type       | Character string indicating the plot required; enforced by <code>base::match.arg()</code> .<br>Currently supported options: <ul style="list-style-type: none"> <li>• "nowcast" which plots the nowcast for each dataset along with latest available observed data using <code>enw_plot_nowcast_quantiles()</code>,</li> <li>• "posterior_prediction" which plots observations reported at the time against simulated observations from the model using <code>enw_plot_pp_quantiles()</code>.</li> </ul> |
| log        | Logical, defaults to FALSE. Should counts be plot on the log scale.                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| ...        | Additional arguments to the plot function specified by type.                                                                                                                                                                                                                                                                                                                                                                                                                                            |

**Value**

ggplot2 object

**See Also**

Other epinowcast: `epinowcast()`, `plot.enw_preprocess_data()`, `print.enw_preprocess_data()`, `print.epinowcast()`, `print.summary.enw_preprocess_data()`, `summary.enw_preprocess_data()`, `summary.epinowcast()`

Plotting functions `enw_delay_categories()`, `enw_delay_quantiles()`, `enw_plot_delay_counts()`, `enw_plot_delay_cumulative()`, `enw_plot_delay_fraction()`, `enw_plot_delay_quantiles()`, `enw_plot_nowcast_quantiles()`, `enw_plot_obs()`, `enw_plot_pp_quantiles()`, `enw_plot_quantiles()`, `enw_plot_theme()`, `plot.enw_preprocess_data()`

**Examples**

```
nowcast <- enw_example("nowcast")
latest_obs <- enw_example("obs")

# Plot nowcast
plot(nowcast, latest_obs = latest_obs, type = "nowcast")

# Plot posterior predictions by reference date
plot(nowcast, type = "posterior_prediction") +
  ggplot2::facet_wrap(ggplot2::vars(reference_date), scales = "free")
```

---

```
print.enw_preprocess_data
```

*Print method for enw\_preprocess\_data*

---

**Description**

print method for class "enw\_preprocess\_data".

**Usage**

```
## S3 method for class 'enw_preprocess_data'
print(x, ...)
```

**Arguments**

x                    A data.table output from [enw\\_preprocess\\_data\(\)](#) or [enw\\_construct\\_data\(\)](#).  
 ...                Additional arguments (not used).

**Value**

Invisibly returns x.

**See Also**

Other epinowcast: [epinowcast\(\)](#), [plot.enw\\_preprocess\\_data\(\)](#), [plot.epinowcast\(\)](#), [print.epinowcast\(\)](#), [print.summary.enw\\_preprocess\\_data\(\)](#), [summary.enw\\_preprocess\\_data\(\)](#), [summary.epinowcast\(\)](#)

**Examples**

```
pobs <- enw_example("preprocessed_observations")
pobs
```

---

|                  |                                    |
|------------------|------------------------------------|
| print.epinowcast | <i>Print method for epinowcast</i> |
|------------------|------------------------------------|

---

**Description**

print method for class "epinowcast".

**Usage**

```
## S3 method for class 'epinowcast'
print(x, ...)
```

**Arguments**

x                    A data.table output from [epinowcast\(\)](#).  
 ...                Additional arguments (not used).

**Value**

Invisibly returns x.

**See Also**

Other epinowcast: [epinowcast\(\)](#), [plot.enw\\_preprocess\\_data\(\)](#), [plot.epinowcast\(\)](#), [print.enw\\_preprocess\\_data\(\)](#), [print.summary.enw\\_preprocess\\_data\(\)](#), [summary.enw\\_preprocess\\_data\(\)](#), [summary.epinowcast\(\)](#)

**Examples**

```
nowcast <- enw_example("nowcast")
nowcast
```

---

```
print.summary.enw_preprocess_data
```

*Print method for summary.enw\_preprocess\_data*

---

**Description**

print method for the output of [summary.enw\\_preprocess\\_data\(\)](#).

**Usage**

```
## S3 method for class 'summary.enw_preprocess_data'
print(x, ...)
```

**Arguments**

x                    A `summary.enw_preprocess_data` object.  
 ...                Additional arguments (not used).

**Value**

Invisibly returns x.

**See Also**

Other `epinowcast`: [epinowcast\(\)](#), [plot.enw\\_preprocess\\_data\(\)](#), [plot.epinowcast\(\)](#), [print.enw\\_preprocess\\_data\(\)](#), [print.epinowcast\(\)](#), [summary.enw\\_preprocess\\_data\(\)](#), [summary.epinowcast\(\)](#)

---

re                    *Defines random effect terms using the lme4 syntax*

---

**Description**

Defines random effect terms using the `lme4` syntax

**Usage**

```
re(formula)
```

**Arguments**

formula            A random effect as returned by [findbars](#) when a random effect is defined using the `lme4` syntax in formula. Currently only simplified random effects (i.e LHS | RHS) are supported.

**Value**

A list defining the fixed and random effects of the specified random effect

**See Also**

Functions used to help convert formulas into model designs [ar\(\)](#), [arima\(\)](#), [arima\\_terms\(\)](#), [arma\(\)](#), [as\\_string\\_formula\(\)](#), [construct\\_arima\(\)](#), [construct\\_gp\(\)](#), [construct\\_re\(\)](#), [construct\\_rw\(\)](#), [enw\\_formula\(\)](#), [enw\\_manual\\_formula\(\)](#), [gp\(\)](#), [gp\\_terms\(\)](#), [ma\(\)](#), [parse\\_formula\(\)](#), [remove\\_arima\\_terms\(\)](#), [remove\\_gp\\_terms\(\)](#), [remove\\_rw\\_terms\(\)](#), [rw\(\)](#), [rw\\_terms\(\)](#), [split\\_formula\\_to\\_terms\(\)](#)

**Examples**

```
form <- epinowcast:::parse_formula(~ 1 + (1 | age_group))
re(form$random[[1]])

form <- epinowcast:::parse_formula(~ 1 + (location | age_group))
re(form$random[[1]])
```

---

|                    |                                                 |
|--------------------|-------------------------------------------------|
| remove_arima_terms | <i>Remove ARIMA terms from a formula object</i> |
|--------------------|-------------------------------------------------|

---

**Description**

This function removes ARIMA terms — [arima\(\)](#), [ar\(\)](#), [ma\(\)](#), and [arma\(\)](#) — from a formula so they can be processed on their own.

**Usage**

```
remove_arima_terms(formula)
```

**Arguments**

|         |                                                                                                                                                                                                                                                                                                    |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| formula | A model formula that may use standard fixed effects, random effects using <a href="#">lme4</a> syntax (see <a href="#">re()</a> ), and random walks defined using the <a href="#">rw()</a> helper function. See the Details section below for a comprehensive explanation of the supported syntax. |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Value**

A formula object with the ARIMA terms removed.

**See Also**

Functions used to help convert formulas into model designs [ar\(\)](#), [arima\(\)](#), [arima\\_terms\(\)](#), [arma\(\)](#), [as\\_string\\_formula\(\)](#), [construct\\_arima\(\)](#), [construct\\_gp\(\)](#), [construct\\_re\(\)](#), [construct\\_rw\(\)](#), [enw\\_formula\(\)](#), [enw\\_manual\\_formula\(\)](#), [gp\(\)](#), [gp\\_terms\(\)](#), [ma\(\)](#), [parse\\_formula\(\)](#), [re\(\)](#), [remove\\_gp\\_terms\(\)](#), [remove\\_rw\\_terms\(\)](#), [rw\(\)](#), [rw\\_terms\(\)](#), [split\\_formula\\_to\\_terms\(\)](#)

## Examples

```
epinowcast::remove_arma_terms(~ 1 + age_group + arma(week))
epinowcast::remove_arma_terms(~ 1 + age_group + ar(week, p = 2))
```

---

remove\_gp\_terms

*Remove Gaussian process terms from a formula object*

---

## Description

This function removes Gaussian process terms denoted using `gp()` from a formula so they can be processed on their own.

## Usage

```
remove_gp_terms(formula)
```

## Arguments

|         |                                                                                                                                                                                                                                                                                                    |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| formula | A model formula that may use standard fixed effects, random effects using <a href="#">lme4</a> syntax (see <a href="#">re()</a> ), and random walks defined using the <a href="#">rw()</a> helper function. See the Details section below for a comprehensive explanation of the supported syntax. |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Value

A formula object with the Gaussian process terms removed.

## See Also

Functions used to help convert formulas into model designs [ar\(\)](#), [arma\(\)](#), [arma\\_terms\(\)](#), [arma\(\)](#), [as\\_string\\_formula\(\)](#), [construct\\_arma\(\)](#), [construct\\_gp\(\)](#), [construct\\_re\(\)](#), [construct\\_rw\(\)](#), [enw\\_formula\(\)](#), [enw\\_manual\\_formula\(\)](#), [gp\(\)](#), [gp\\_terms\(\)](#), [ma\(\)](#), [parse\\_formula\(\)](#), [re\(\)](#), [remove\\_arma\\_terms\(\)](#), [remove\\_rw\\_terms\(\)](#), [rw\(\)](#), [rw\\_terms\(\)](#), [split\\_formula\\_to\\_terms\(\)](#)

## Examples

```
epinowcast::remove_gp_terms(~ 1 + age_group + gp(week))
```

---

|                  |                                                                                   |
|------------------|-----------------------------------------------------------------------------------|
| remove_profiling | <i>Remove profiling statements from a character vector representing stan code</i> |
|------------------|-----------------------------------------------------------------------------------|

---

**Description**

Remove profiling statements from a character vector representing stan code

**Usage**

```
remove_profiling(s)
```

**Arguments**

|   |                                         |
|---|-----------------------------------------|
| s | Character vector representing stan code |
|---|-----------------------------------------|

**Value**

A character vector of the stan code without profiling statements

**See Also**

Functions used to help convert models into the format required for stan [enw\\_formula\\_as\\_data\\_list\(\)](#), [enw\\_get\\_cache\(\)](#), [enw\\_model\(\)](#), [enw\\_pathfinder\(\)](#), [enw\\_priors\\_as\\_data\\_list\(\)](#), [enw\\_replace\\_priors\(\)](#), [enw\\_sample\(\)](#), [enw\\_set\\_cache\(\)](#), [enw\\_stan\\_to\\_r\(\)](#), [enw\\_unset\\_cache\(\)](#), [write\\_stan\\_files\\_no\\_profile\(\)](#)

---

|                 |                                                       |
|-----------------|-------------------------------------------------------|
| remove_rw_terms | <i>Remove random walk terms from a formula object</i> |
|-----------------|-------------------------------------------------------|

---

**Description**

This function removes random walk terms denoted using [rw\(\)](#) from a formula so that they can be processed on their own.

**Usage**

```
remove_rw_terms(formula)
```

**Arguments**

|         |                                                                                                                                                                                                                                                                                                    |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| formula | A model formula that may use standard fixed effects, random effects using <a href="#">lme4</a> syntax (see <a href="#">re()</a> ), and random walks defined using the <a href="#">rw()</a> helper function. See the Details section below for a comprehensive explanation of the supported syntax. |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Value**

A formula object with the random walk terms removed.

**Reference**

This function was adapted from code written by J Scott (under an MIT license) as part of the `epidemia` package (<https://github.com/ImperialCollegeLondon/epidemia/>).

**See Also**

Functions used to help convert formulas into model designs `ar()`, `arima()`, `arima_terms()`, `arma()`, `as_string_formula()`, `construct_arima()`, `construct_gp()`, `construct_re()`, `construct_rw()`, `enw_formula()`, `enw_manual_formula()`, `gp()`, `gp_terms()`, `ma()`, `parse_formula()`, `re()`, `remove_arima_terms()`, `remove_gp_terms()`, `rw()`, `rw_terms()`, `split_formula_to_terms()`

**Examples**

```
epinowcast:::remove_rw_terms(~ 1 + age_group + location)

epinowcast:::remove_rw_terms(~ 1 + age_group + location + rw(week, location))
```

---

 rw

*Adds random walks with Gaussian steps to the model.*

---

**Description**

A call to `rw()` can be used in the formula argument of model construction functions in the `epinowcast` package such as `enw_formula()`. Mathematically a Gaussian random walk is exactly an ARIMA(0, 1, 0) process; `rw(time, by, type)` is now a thin wrapper over `arima()` with  $p = 0$ ,  $d = 1$ ,  $q = 0$ . It is kept as a user-facing convenience because random walks are the most common time-series structure in `epinowcast` formulas.

**Usage**

```
rw(time, by)
```

**Arguments**

|                   |                                                                                                                                                                                                                                                                                                             |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>time</code> | Defines the random walk time period.                                                                                                                                                                                                                                                                        |
| <code>by</code>   | Defines the grouping parameter used for the random walk. If not specified no grouping is used. Currently this is limited to a single variable. Each group draws an independent shock series; the latent standard deviation is shared across groups (per-group standard deviations are a planned extension). |

**Details**

Does not evaluate arguments but instead simply passes information for use in model construction.

**Value**

A list of class `enw_arma_term` (with  $p = 0$ ,  $d = 1$ ,  $q = 0$ ) that can be interpreted by `construct_arma()`.

**See Also**

Functions used to help convert formulas into model designs `ar()`, `arma()`, `arma_terms()`, `arma()`, `as_string_formula()`, `construct_arma()`, `construct_gp()`, `construct_re()`, `construct_rw()`, `enw_formula()`, `enw_manual_formula()`, `gp()`, `gp_terms()`, `ma()`, `parse_formula()`, `re()`, `remove_arma_terms()`, `remove_gp_terms()`, `remove_rw_terms()`, `rw_terms()`, `split_formula_to_terms()`

**Examples**

```
rw(time)

rw(time, location)

rw(time, location)
```

---

|          |                                                    |
|----------|----------------------------------------------------|
| rw_terms | <i>Finds random walk terms in a formula object</i> |
|----------|----------------------------------------------------|

---

**Description**

This function extracts random walk terms denoted using `rw()` from a formula so that they can be processed on their own.

**Usage**

```
rw_terms(formula)
```

**Arguments**

|         |                                                                                                                                                                                                                                                                                           |
|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| formula | A model formula that may use standard fixed effects, random effects using <code>lme4</code> syntax (see <code>re()</code> ), and random walks defined using the <code>rw()</code> helper function. See the Details section below for a comprehensive explanation of the supported syntax. |
|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Value**

A character vector containing the random walk terms that have been identified in the supplied formula.

**Reference**

This function was adapted from code written by J Scott (under an MIT license) as part of the `epidemia` package (<https://github.com/ImperialCollegeLondon/epidemia/>).

**See Also**

Functions used to help convert formulas into model designs `ar()`, `arima()`, `arima_terms()`, `arma()`, `as_string_formula()`, `construct_arima()`, `construct_gp()`, `construct_re()`, `construct_rw()`, `enw_formula()`, `enw_manual_formula()`, `gp()`, `gp_terms()`, `ma()`, `parse_formula()`, `re()`, `remove_arima_terms()`, `remove_gp_terms()`, `remove_rw_terms()`, `rw()`, `split_formula_to_terms()`

**Examples**

```
epinowcast:::rw_terms(~ 1 + age_group + location)
```

```
epinowcast:::rw_terms(~ 1 + age_group + location + rw(week, location))
```

---

```
split_formula_to_terms
```

*Split formula into individual terms*

---

**Description**

Split formula into individual terms

**Usage**

```
split_formula_to_terms(formula)
```

**Arguments**

|         |                                                                                                                                                                                                                                                                                              |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| formula | A model formula that may use standard fixed effects, random effects using <a href="#">lme4</a> syntax (see <code>re()</code> ), and random walks defined using the <code>rw()</code> helper function. See the Details section below for a comprehensive explanation of the supported syntax. |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

**Value**

A character vector of formula terms

**See Also**

Functions used to help convert formulas into model designs `ar()`, `arima()`, `arima_terms()`, `arma()`, `as_string_formula()`, `construct_arima()`, `construct_gp()`, `construct_re()`, `construct_rw()`, `enw_formula()`, `enw_manual_formula()`, `gp()`, `gp_terms()`, `ma()`, `parse_formula()`, `re()`, `remove_arima_terms()`, `remove_gp_terms()`, `remove_rw_terms()`, `rw()`, `rw_terms()`

**Examples**

```
epinowcast:::split_formula_to_terms(~ 1 + age_group + location)
```

---

|                    |                                                           |
|--------------------|-----------------------------------------------------------|
| stan_fns_as_string | <i>Read in a stan function file as a character string</i> |
|--------------------|-----------------------------------------------------------|

---

**Description**

Read in a stan function file as a character string

**Usage**

```
stan_fns_as_string(files, include)
```

**Arguments**

|         |                                                                                                                                                                                                                                                                                                                                                     |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| files   | A character vector specifying the names of Stan files to be exposed. These must be in the include directory. Defaults to all Stan files in the include directory. Note that the following files contain overloaded functions and cannot be exposed: "delay_lpmf.stan", "allocate_observed_obs.stan", "obs_lpmf.stan", and "effects_priors_lp.stan". |
| include | A character string specifying the directory containing Stan files. Defaults to the 'stan/functions' directory of the <a href="#">epinowcast()</a> package.                                                                                                                                                                                          |

**Value**

A character string in the of stan functions.

**See Also**

Utility functions [coerce\\_date\(\)](#), [coerce\\_dt\(\)](#), [date\\_to\\_numeric\\_modulus\(\)](#), [enw\\_get\\_data\(\)](#), [enw\\_rolling\\_sum\(\)](#), [get\\_internal\\_timestep\(\)](#), [is.Date\(\)](#)

---

|            |                                                                                                |
|------------|------------------------------------------------------------------------------------------------|
| subset_obs | <i>Subset observations data table for either modelled dates or not-modelled earlier dates.</i> |
|------------|------------------------------------------------------------------------------------------------|

---

**Description**

Subset observations data table for either modelled dates or not-modelled earlier dates.

**Usage**

```
subset_obs(ord_obs, max_delay, internal_timestep, reference_subset)
```

**Arguments**

|                   |                                                                                                                                                |
|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| ord_obs           | The observations data.table to be subset, as pulled from the result of calling <code>epinowcast()</code> and coerced to a data table.          |
| max_delay         | Whole number representing the maximum delay in units of the timestep.                                                                          |
| internal_timestep | A numeric value representing the number of days in the timestep, e.g. 7 when the timesteps are weeks.                                          |
| reference_subset  | String giving a relational operator to subset ord_obs by reference date; e.g. > to keep the modelled reference dates from after the max_delay. |

**Value**

A data.frame subset for the desired observations

**See Also**

Functions used for postprocessing of model fits `.check_primarycensored()`, `.delay_draw_columns()`, `.discretise_parametric_pmf()`, `build_ord_obs()`, `enw_add_latest_obs_to_nowcast()`, `enw_nowcast_samples()`, `enw_nowcast_summary()`, `enw_posterior()`, `enw_posterior_delay()`, `enw_pp_summary()`, `enw_quantiles_to_long()`, `enw_summarise_samples()`

---

```
summary.enw_preprocess_data
```

*Summary method for enw\_preprocess\_data*

---

**Description**

summary method for class "enw\_preprocess\_data". Returns a structured overview of the preprocessed data including a preview of the latest observations and a corner of the reporting triangle.

**Usage**

```
## S3 method for class 'enw_preprocess_data'
summary(object, n = 6, ...)
```

**Arguments**

|        |                                                                                                    |
|--------|----------------------------------------------------------------------------------------------------|
| object | A data.table output from <code>enw_preprocess_data()</code> or <code>enw_construct_data()</code> . |
| n      | Integer number of rows to show in previews. Defaults to 6.                                         |
| ...    | Additional arguments (not used).                                                                   |

**Value**

A list of class "summary.enw\_preprocess\_data" containing the preprocessed data object and preview parameters, printed via `print.summary.enw_preprocess_data()`.

**See Also**

Other epinowcast: [epinowcast\(\)](#), [plot.enw\\_preprocess\\_data\(\)](#), [plot.epinowcast\(\)](#), [print.enw\\_preprocess\\_data\(\)](#), [print.epinowcast\(\)](#), [print.summary.enw\\_preprocess\\_data\(\)](#), [summary.epinowcast\(\)](#)

**Examples**

```
pobs <- enw_example("preprocessed_observations")
summary(pobs)
```

---

|                    |                                      |
|--------------------|--------------------------------------|
| summary.epinowcast | <i>Summary method for epinowcast</i> |
|--------------------|--------------------------------------|

---

**Description**

summary method for class "epinowcast".

**Usage**

```
## S3 method for class 'epinowcast'
summary(
  object,
  type = c("nowcast", "nowcast_samples", "fit", "posterior_prediction"),
  max_delay = object$max_delay,
  ...
)
```

**Arguments**

|           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| object    | A data.table output from <a href="#">epinowcast()</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| type      | Character string indicating the summary to return; enforced by <a href="#">base::match.arg()</a> . Supported options are: <ul style="list-style-type: none"> <li>"nowcast" which summarises nowcast posterior with <a href="#">enw_nowcast_summary()</a>,</li> <li>"nowcast_samples" which samples latest with <a href="#">enw_nowcast_samples()</a>,</li> <li>"fit" which returns the summarised cmdstanr fit with <a href="#">enw_posterior()</a>,</li> <li>"posterior_prediction" which returns summarised posterior predictions for the observations after fitting using <a href="#">enw_pp_summary()</a>.</li> </ul> |
| max_delay | Maximum delay to which nowcasts should be summarised, in units of the timestep used during preprocessing. Must be equal (default) or larger than the modelled maximum delay. If it is larger, then nowcasts for unmodelled dates are added by assuming that case counts beyond the modelled maximum delay are fully observed.                                                                                                                                                                                                                                                                                             |
| ...       | Additional arguments passed to summary specified by type.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

**Value**

A summary data.frame

**See Also**

summary epinowcast

Other epinowcast: [epinowcast\(\)](#), [plot.enw\\_preprocess\\_data\(\)](#), [plot.epinowcast\(\)](#), [print.enw\\_preprocess\\_data\(\)](#), [print.epinowcast\(\)](#), [print.summary.enw\\_preprocess\\_data\(\)](#), [summary.enw\\_preprocess\\_data\(\)](#)

**Examples**

```
nowcast <- enw_example("nowcast")

# Summarise nowcast posterior
summary(nowcast, type = "nowcast")

# Nowcast posterior samples
summary(nowcast, type = "nowcast_samples")

# Nowcast model fit
summary(nowcast, type = "fit")

# Posterior predictions
summary(nowcast, type = "posterior_prediction")
```

---

```
write_stan_files_no_profile
```

*Write copies of the .stan files of a Stan model and its #include files with all profiling statements removed.*

---

**Description**

Write copies of the .stan files of a Stan model and its #include files with all profiling statements removed.

**Usage**

```
write_stan_files_no_profile(
  stan_file,
  include_paths = NULL,
  target_dir = epinowcast::enw_get_cache()
)
```

**Arguments**

|               |                                                                                                                                                                                                                                                       |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| stan_file     | The path to a .stan file containing a Stan program.                                                                                                                                                                                                   |
| include_paths | Paths to directories where Stan should look for files specified in #include directives in the Stan program.                                                                                                                                           |
| target_dir    | The path to a directory in which the manipulated .stan files without profiling statements should be stored. To avoid overriding of the original .stan files, this should be different from the directory of the original model and the include_paths. |

**Value**

A list containing the path to the .stan file without profiling statements and the include\_paths for the included .stan files without profiling statements

**See Also**

Functions used to help convert models into the format required for stan [enw\\_formula\\_as\\_data\\_list\(\)](#), [enw\\_get\\_cache\(\)](#), [enw\\_model\(\)](#), [enw\\_pathfinder\(\)](#), [enw\\_priors\\_as\\_data\\_list\(\)](#), [enw\\_replace\\_priors\(\)](#), [enw\\_sample\(\)](#), [enw\\_set\\_cache\(\)](#), [enw\\_stan\\_to\\_r\(\)](#), [enw\\_unset\\_cache\(\)](#), [remove\\_profiling\(\)](#)

# Index

## \* **check**

- check\_design\_matrix\_sparsity, 15
- check\_group, 16
- check\_group\_date\_unique, 17
- check\_max\_delay, 17
- check\_module, 19
- check\_modules\_compatible, 20
- check\_numeric\_timestep, 20
- check\_observation\_indicator, 21
- check\_quantiles, 22
- check\_timestep, 22
- check\_timestep\_by\_date, 23
- check\_timestep\_by\_group, 24

## \* **dataconverters**

- enw\_add\_cumulative, 34
- enw\_add\_incidence, 37
- enw\_aggregate\_cumulative, 42
- enw\_incidence\_to\_linelist, 71
- enw\_linelist\_to\_incidence, 72

## \* **datasets**

- germany\_covid19\_hosp, 125

## \* **data**

- enw\_example, 54
- germany\_covid19\_hosp, 125

## \* **epinowcast**

- epinowcast, 120
- plot.enw\_preprocess\_data, 132
- plot.epinowcast, 133
- print.enw\_preprocess\_data, 134
- print.epinowcast, 135
- print.summary.enw\_preprocess\_data, 136
- summary.enw\_preprocess\_data, 144
- summary.epinowcast, 145

## \* **formulatools**

- ar, 10
- arma, 10
- arma\_terms, 11
- arma, 12

- as\_string\_formula, 14
- construct\_arma, 27
- construct\_gp, 28
- construct\_re, 29
- construct\_rw, 30
- enw\_formula, 64
- enw\_manual\_formula, 74
- gp, 126
- gp\_terms, 128
- ma, 130
- parse\_formula, 130
- re, 136
- remove\_arma\_terms, 137
- remove\_gp\_terms, 138
- remove\_rw\_terms, 139
- rw, 140
- rw\_terms, 141
- split\_formula\_to\_terms, 142

## \* **modeldesign**

- enw\_add\_cumulative\_membership, 35
- enw\_add\_pooling\_effect, 41
- enw\_design, 52
- enw\_effects\_metadata, 53
- enw\_one\_hot\_encode\_feature, 86

## \* **modelmodulehelpers**

- add\_max\_observed\_delay, 8
- add\_pmfs, 9
- convolution\_matrix, 31
- delay\_only\_ltotal, 33
- delay\_only\_total, 33
- enw\_dayofweek\_structural\_reporting, 49
- enw\_reference\_by\_report, 105
- enw\_reps\_with\_complete\_refs, 110
- enw\_structural\_reporting\_metadata, 117
- extract\_obs\_metadata, 123
- extract\_sparse\_matrix, 124
- latest\_obs\_as\_matrix, 129

- \* **modelmodules**
  - .expectation\_is\_minimal, 7
  - enw\_expectation, 55
  - enw\_fit\_opts, 62
  - enw\_missing, 78
  - enw\_obs, 84
  - enw\_reference, 103
  - enw\_report, 107
- \* **modeltools**
  - enw\_formula\_as\_data\_list, 67
  - enw\_get\_cache, 69
  - enw\_model, 80
  - enw\_pathfinder, 87
  - enw\_priors\_as\_data\_list, 102
  - enw\_replace\_priors, 106
  - enw\_sample, 113
  - enw\_set\_cache, 114
  - enw\_stan\_to\_r, 116
  - enw\_unset\_cache, 119
  - remove\_profiling, 139
  - write\_stan\_files\_no\_profile, 146
- \* **modelvalidation**
  - as\_forecast\_sample.epinowcast, 13
- \* **plot**
  - enw\_delay\_categories, 50
  - enw\_delay\_quantiles, 51
  - enw\_plot\_delay\_counts, 88
  - enw\_plot\_delay\_cumulative, 89
  - enw\_plot\_delay\_fraction, 90
  - enw\_plot\_delay\_quantiles, 91
  - enw\_plot\_nowcast\_quantiles, 91
  - enw\_plot\_obs, 92
  - enw\_plot\_pp\_quantiles, 93
  - enw\_plot\_quantiles, 94
  - enw\_plot\_theme, 95
  - plot.enw\_preprocess\_data, 132
  - plot.epinowcast, 133
- \* **postprocess**
  - .check\_primarycensored, 5
  - .delay\_draw\_columns, 6
  - .discretise\_parametric\_pmf, 6
  - build\_ord\_obs, 14
  - enw\_add\_latest\_obs\_to\_nowcast, 38
  - enw\_nowcast\_samples, 81
  - enw\_nowcast\_summary, 82
  - enw\_posterior, 96
  - enw\_posterior\_delay, 97
  - enw\_pp\_summary, 98
  - enw\_quantiles\_to\_long, 102
  - enw\_summarise\_samples, 118
  - subset\_obs, 143
- \* **preprocess**
  - enw\_add\_delay, 36
  - enw\_add\_max\_reported, 39
  - enw\_add\_metaobs\_features, 40
  - enw\_assign\_group, 44
  - enw\_complete\_dates, 45
  - enw\_construct\_data, 47
  - enw\_extend\_date, 56
  - enw\_filter\_delay, 57
  - enw\_filter\_reference\_dates, 58
  - enw\_filter\_reference\_dates\_by\_report\_start, 60
  - enw\_filter\_report\_dates, 61
  - enw\_flag\_observed\_observations, 63
  - enw\_impute\_na\_observations, 70
  - enw\_latest\_data, 72
  - enw\_metadata, 75
  - enw\_metadata\_delay, 76
  - enw\_missing\_reference, 79
  - enw\_obs\_at\_delay, 85
  - enw\_preprocess\_data, 99
  - enw\_reporting\_triangle, 108
  - enw\_reporting\_triangle\_to\_long, 109
  - enw\_retrospective, 111
- \* **simulate**
  - enw\_simulate\_missing\_reference, 115
- \* **utils**
  - coerce\_date, 25
  - coerce\_dt, 26
  - date\_to\_numeric\_modulus, 32
  - enw\_get\_data, 69
  - enw\_rolling\_sum, 112
  - get\_internal\_timestep, 125
  - is.Date, 128
  - stan\_fns\_as\_string, 143
  - .check\_primarycensored, 5, 6, 7, 15, 38, 82, 83, 96, 98, 99, 103, 119, 144
  - .delay\_draw\_columns, 5, 6, 7, 15, 38, 82, 83, 96, 98, 99, 103, 119, 144
  - .discretise\_parametric\_pmf, 5, 6, 6, 15, 38, 82, 83, 96, 98, 99, 103, 119, 144
  - .expectation\_is\_minimal, 7, 56, 63, 78, 85, 104, 108

- add\_max\_observed\_delay, 8, 9, 31, 33, 34, 50, 106, 111, 118, 123, 124, 129
- add\_max\_observed\_delay(), 8, 123
- add\_pmfs, 8, 9, 31, 33, 34, 50, 106, 111, 118, 123, 124, 129
- ar, 10, 11–14, 28, 29, 31, 66, 75, 127, 128, 130, 131, 137, 138, 140–142
- ar(), 11
- arma, 10, 10, 12–14, 28, 29, 31, 66, 75, 127, 128, 130, 131, 137, 138, 140–142
- arma(), 10–12, 27, 65, 68, 126, 127, 130, 131, 140
- arma\_terms, 10, 11, 11, 13, 14, 28, 29, 31, 66, 75, 127, 128, 130, 131, 137, 138, 140–142
- arma, 10, 11, 12, 12, 14, 28, 29, 31, 66, 75, 127, 128, 130, 131, 137, 138, 140–142
- arma(), 11
- as.IDate, 26
- as.integer(), 100
- as\_forecast\_sample.epinowcast, 13
- as\_string\_formula, 10–13, 14, 28, 29, 31, 66, 75, 127, 128, 130, 131, 137, 138, 140–142
- base::match.arg(), 62, 84, 132, 134, 145
- build\_ord\_obs, 5–7, 14, 38, 82, 83, 96, 98, 99, 103, 119, 144
- check\_design\_matrix\_sparsity, 15, 16, 17, 19–24
- check\_group, 16, 16, 17, 19–24
- check\_group\_date\_unique, 16, 17, 19–24
- check\_max\_delay, 16, 17, 17, 19–24
- check\_max\_delay(), 18, 45, 48, 58, 77, 86, 100, 105, 110
- check\_module, 16, 17, 19, 19–24
- check\_modules\_compatible, 16, 17, 19, 20, 21–24
- check\_numeric\_timestep, 16, 17, 19, 20, 20–24
- check\_observation\_indicator, 16, 17, 19, 20, 21, 21–24
- check\_quantiles, 16, 17, 19–21, 22, 23, 24
- check\_timestep, 16, 17, 19–21, 22, 22, 24
- check\_timestep\_by\_date, 16, 17, 19–22, 23, 23, 24
- check\_timestep\_by\_group, 16, 17, 19–23, 24, 24
- CmdStanModel, 117
- cmdstanr::cmdstan\_model(), 80, 81
- cmdstanr::CmdStanModel, 116
- cmdstanr::sample(), 62
- coerce\_date, 25, 27, 32, 70, 112, 126, 129, 143
- coerce\_date(), 40
- coerce\_dt, 25, 26, 32, 70, 112, 126, 129, 143
- coerce\_dt(), 22
- construct\_arma, 10–14, 27, 29, 31, 66, 75, 127, 128, 130, 131, 137, 138, 140–142
- construct\_arma(), 10–12, 28, 130, 141
- construct\_gp, 10–14, 28, 28, 29, 31, 66, 75, 127, 128, 130, 131, 137, 138, 140–142
- construct\_gp(), 127
- construct\_re, 10–14, 28, 29, 29, 31, 66, 75, 127, 128, 130, 131, 137, 138, 140–142
- construct\_rw, 10–14, 28, 29, 30, 66, 75, 127, 128, 130, 131, 137, 138, 140–142
- construct\_rw(), 27
- convolution\_matrix, 8, 9, 31, 33, 34, 50, 106, 111, 118, 123, 124, 129
- data.table, 26
- data.table::as.data.table(), 23, 24, 26, 40
- data.table::as.IDate(), 25
- Date, 75, 76
- date\_to\_numeric\_modulus, 25, 27, 32, 70, 112, 126, 129, 143
- Dates, 40
- delay\_only\_ltotal, 8, 9, 31, 33, 34, 50, 106, 111, 118, 123, 124, 129
- delay\_only\_total, 8, 9, 31, 33, 33, 50, 106, 111, 118, 123, 124, 129
- enw\_add\_cumulative, 34, 37, 44, 71, 73
- enw\_add\_cumulative\_membership, 35, 42, 52, 53, 87
- enw\_add\_cumulative\_membership(), 30
- enw\_add\_delay, 36, 39, 41, 44, 46, 49, 57–61, 64, 70, 72, 76, 77, 79, 86, 101, 109, 110, 112
- enw\_add\_incidence, 34, 37, 44, 71, 73

- `enw_add_incidence()`, 60, 79, 109
- `enw_add_latest_obs_to_nowcast`, 5–7, 15, 38, 82, 83, 96, 98, 99, 103, 119, 144
- `enw_add_max_reported`, 36, 39, 41, 44, 46, 49, 57–61, 64, 70, 72, 76, 77, 79, 86, 101, 109, 110, 112
- `enw_add_metaobs_features`, 36, 39, 40, 44, 46, 49, 57–61, 64, 70, 72, 76, 77, 79, 86, 101, 109, 110, 112
- `enw_add_metaobs_features()`, 56, 100
- `enw_add_pooling_effect`, 35, 41, 52, 53, 87
- `enw_aggregate_cumulative`, 34, 37, 42, 71, 73
- `enw_aggregate_cumulative()`, 48, 58, 86, 100
- `enw_assign_group`, 36, 39, 41, 44, 46, 49, 57–61, 64, 70, 72, 76, 77, 79, 86, 101, 109, 110, 112
- `enw_complete_dates`, 36, 39, 41, 44, 45, 49, 57–61, 64, 70, 72, 76, 77, 79, 86, 101, 109, 110, 112
- `enw_complete_dates()`, 43, 84, 99
- `enw_construct_data`, 36, 39, 41, 44, 46, 47, 57–61, 64, 70, 72, 76, 77, 79, 86, 101, 109, 110, 112
- `enw_construct_data()`, 135, 144
- `enw_dayofweek_structural_reporting`, 8, 9, 31, 33, 34, 49, 106, 111, 118, 123, 124, 129
- `enw_dayofweek_structural_reporting()`, 108
- `enw_delay_categories`, 50, 52, 89–95, 133, 134
- `enw_delay_quantiles`, 51, 51, 89–95, 133, 134
- `enw_design`, 35, 42, 52, 53, 87
- `enw_effects_metadata`, 35, 42, 52, 53, 87
- `enw_effects_metadata()`, 41, 42
- `enw_example`, 54, 125
- `enw_expectation`, 7, 55, 63, 78, 85, 104, 108
- `enw_expectation()`, 7, 19, 66, 87, 113, 121, 124, 126
- `enw_extend_date`, 36, 39, 41, 44, 46, 49, 56, 58–61, 64, 70, 72, 76, 77, 79, 86, 101, 109, 110, 112
- `enw_filter_delay`, 36, 39, 41, 44, 46, 49, 57, 59–61, 64, 70, 72, 76, 77, 79, 86, 101, 109, 110, 112
- `enw_filter_reference_dates`, 36, 39, 41, 44, 46, 49, 57, 58, 58, 60, 61, 64, 70, 72, 76, 77, 79, 86, 101, 109, 110, 112
- `enw_filter_reference_dates()`, 61
- `enw_filter_reference_dates_by_report_start`, 36, 39, 41, 44, 46, 49, 57–59, 60, 61, 64, 70, 72, 76, 77, 79, 86, 101, 109, 110, 112
- `enw_filter_reference_dates_by_report_start()`, 37
- `enw_filter_report_dates`, 36, 39, 41, 44, 46, 49, 57–60, 61, 64, 70, 72, 76, 77, 79, 86, 101, 109, 110, 112
- `enw_filter_report_dates()`, 58
- `enw_fit_opts`, 7, 56, 62, 78, 85, 104, 108
- `enw_fit_opts()`, 121
- `enw_flag_observed_observations`, 36, 39, 41, 44, 46, 49, 57–61, 63, 70, 72, 76, 77, 79, 86, 101, 109, 110, 112
- `enw_flag_observed_observations()`, 46, 84
- `enw_formula`, 10–14, 28, 29, 31, 64, 75, 127, 128, 130, 131, 137, 138, 140–142
- `enw_formula()`, 10, 55, 68, 74, 78, 103, 104, 107, 126, 140
- `enw_formula_as_data_list`, 67, 69, 81, 88, 102, 107, 113, 115, 117, 120, 139, 147
- `enw_get_cache`, 68, 69, 81, 88, 102, 107, 113, 115, 117, 120, 139, 147
- `enw_get_data`, 25, 27, 32, 69, 112, 126, 129, 143
- `enw_impute_na_observations`, 36, 39, 41, 44, 46, 49, 57–61, 64, 70, 72, 76, 77, 79, 86, 101, 109, 110, 112
- `enw_incidence_to_linelist`, 34, 37, 44, 71, 73
- `enw_latest_data`, 36, 39, 41, 44, 46, 49, 57–61, 64, 70, 72, 76, 77, 79, 86, 101, 109, 110, 112
- `enw_latest_data()`, 38, 54, 81, 83
- `enw_linelist_to_incidence`, 34, 37, 44, 71, 72
- `enw_manual_formula`, 10–14, 28, 29, 31, 66, 74, 127, 128, 130, 131, 137, 138, 140–142
- `enw_metadata`, 36, 39, 41, 44, 46, 49, 57–61, 64, 70, 72, 75, 77, 79, 86, 101, 109,

- [110, 112](#)
- [enw\\_metadata\\_delay, 36, 39, 41, 44, 46, 49, 57–61, 64, 70, 72, 76, 76, 79, 86, 101, 109, 110, 112](#)
- [enw\\_metadata\\_delay\(\), 47, 48, 101](#)
- [enw\\_missing, 7, 56, 63, 78, 85, 104, 108](#)
- [enw\\_missing\(\), 63, 121, 126](#)
- [enw\\_missing\\_reference, 36, 39, 41, 44, 46, 49, 57–61, 64, 70, 72, 76, 77, 79, 86, 101, 109, 110, 112](#)
- [enw\\_model, 68, 69, 80, 88, 102, 107, 113, 115, 117, 120, 139, 147](#)
- [enw\\_model\(\), 87, 113, 121](#)
- [enw\\_nowcast\\_samples, 5–7, 15, 38, 81, 83, 96, 98, 99, 103, 119, 144](#)
- [enw\\_nowcast\\_samples\(\), 145](#)
- [enw\\_nowcast\\_summary, 5–7, 15, 38, 82, 82, 96, 98, 99, 103, 119, 144](#)
- [enw\\_nowcast\\_summary\(\), 38, 145](#)
- [enw\\_obs, 7, 56, 63, 78, 84, 104, 108](#)
- [enw\\_obs\(\), 46, 87, 113, 121](#)
- [enw\\_obs\\_at\\_delay, 36, 39, 41, 44, 46, 49, 57–61, 64, 70, 72, 76, 77, 79, 85, 101, 109, 110, 112](#)
- [enw\\_one\\_hot\\_encode\\_feature, 35, 42, 52, 53, 86](#)
- [enw\\_pathfinder, 68, 69, 81, 87, 102, 107, 113, 115, 117, 120, 139, 147](#)
- [enw\\_pathfinder\(\), 113](#)
- [enw\\_plot\\_delay\\_counts, 51, 52, 88, 89–95, 133, 134](#)
- [enw\\_plot\\_delay\\_counts\(\), 132](#)
- [enw\\_plot\\_delay\\_cumulative, 51, 52, 89, 89–95, 133, 134](#)
- [enw\\_plot\\_delay\\_cumulative\(\), 132](#)
- [enw\\_plot\\_delay\\_fraction, 51, 52, 89, 90, 91–95, 133, 134](#)
- [enw\\_plot\\_delay\\_fraction\(\), 132](#)
- [enw\\_plot\\_delay\\_quantiles, 51, 52, 89, 90, 91, 92–95, 133, 134](#)
- [enw\\_plot\\_delay\\_quantiles\(\), 132](#)
- [enw\\_plot\\_nowcast\\_quantiles, 51, 52, 89, 90, 91, 91, 93–95, 133, 134](#)
- [enw\\_plot\\_nowcast\\_quantiles\(\), 94, 95, 134](#)
- [enw\\_plot\\_obs, 51, 52, 89–91, 92, 92, 94, 95, 133, 134](#)
- [enw\\_plot\\_obs\(\), 132](#)
- [enw\\_plot\\_pp\\_quantiles, 51, 52, 89–92, 93, 93, 95, 133, 134](#)
- [enw\\_plot\\_pp\\_quantiles\(\), 92–95, 134](#)
- [enw\\_plot\\_quantiles, 51, 52, 89–93, 94, 94, 95, 133, 134](#)
- [enw\\_plot\\_theme, 51, 52, 89–94, 95, 95, 133, 134](#)
- [enw\\_posterior, 5–7, 15, 38, 82, 83, 96, 98, 99, 103, 119, 144](#)
- [enw\\_posterior\(\), 82, 103, 145](#)
- [enw\\_posterior\\_delay, 5–7, 15, 38, 82, 83, 96, 97, 99, 103, 119, 144](#)
- [enw\\_pp\\_summary, 5–7, 15, 38, 82, 83, 96, 98, 98, 103, 119, 144](#)
- [enw\\_pp\\_summary\(\), 145](#)
- [enw\\_preprocess\\_data, 36, 39, 41, 44, 46, 49, 57–61, 64, 70, 72, 76, 77, 79, 86, 99, 109, 110, 112](#)
- [enw\\_preprocess\\_data\(\), 8, 18, 33, 34, 38, 40, 47, 49, 51, 54–56, 69, 72, 76, 78, 81, 83–85, 88–91, 103–105, 107, 108, 110, 111, 118, 121, 123, 129, 132, 135, 144](#)
- [enw\\_priors\\_as\\_data\\_list, 68, 69, 81, 88, 102, 107, 113, 115, 117, 120, 139, 147](#)
- [enw\\_quantiles\\_to\\_long, 5–7, 15, 38, 82, 83, 96, 98, 99, 102, 119, 144](#)
- [enw\\_reference, 7, 56, 63, 78, 85, 103, 108](#)
- [enw\\_reference\(\), 66, 97, 102, 106, 107, 121, 126](#)
- [enw\\_reference\\_by\\_report, 8, 9, 31, 33, 34, 50, 105, 111, 118, 123, 124, 129](#)
- [enw\\_replace\\_priors, 68, 69, 81, 88, 102, 106, 113, 115, 117, 120, 139, 147](#)
- [enw\\_replace\\_priors\(\), 121, 126](#)
- [enw\\_report, 7, 56, 63, 78, 85, 104, 107](#)
- [enw\\_report\(\), 66, 121, 126](#)
- [enw\\_reporting\\_triangle, 36, 39, 41, 44, 46, 49, 57–61, 64, 70, 72, 76, 77, 79, 86, 101, 108, 110, 112](#)
- [enw\\_reporting\\_triangle\(\), 109](#)
- [enw\\_reporting\\_triangle\\_to\\_long, 36, 39, 41, 44, 46, 49, 57–61, 64, 70, 72, 76, 77, 79, 86, 101, 109, 109, 112](#)
- [enw\\_reps\\_with\\_complete\\_refs, 8, 9, 31, 33, 34, 50, 106, 110, 118, 123, 124, 129](#)
- [enw\\_reps\\_with\\_complete\\_refs\(\), 105](#)

- enw\_retrospective, [36](#), [39](#), [41](#), [44](#), [46](#), [49](#),  
[57–61](#), [64](#), [70](#), [72](#), [76](#), [77](#), [79](#), [86](#),  
[101](#), [109](#), [110](#), [111](#)
- enw\_rolling\_sum, [25](#), [27](#), [32](#), [70](#), [112](#), [126](#),  
[129](#), [143](#)
- enw\_sample, [68](#), [69](#), [81](#), [88](#), [102](#), [107](#), [113](#),  
[115](#), [117](#), [120](#), [139](#), [147](#)
- enw\_sample(), [62](#), [63](#), [88](#), [121](#)
- enw\_set\_cache, [68](#), [69](#), [81](#), [88](#), [102](#), [107](#), [113](#),  
[114](#), [117](#), [120](#), [139](#), [147](#)
- enw\_simulate\_missing\_reference, [115](#)
- enw\_stan\_to\_r, [68](#), [69](#), [81](#), [88](#), [102](#), [107](#), [113](#),  
[115](#), [116](#), [120](#), [139](#), [147](#)
- enw\_structural\_reporting\_metadata, [8](#), [9](#),  
[31](#), [33](#), [34](#), [50](#), [106](#), [111](#), [117](#), [123](#),  
[124](#), [129](#)
- enw\_structural\_reporting\_metadata(),  
[49](#), [108](#)
- enw\_summarise\_samples, [5–7](#), [15](#), [38](#), [82](#), [83](#),  
[96](#), [98](#), [99](#), [103](#), [118](#), [144](#)
- enw\_unset\_cache, [68](#), [69](#), [81](#), [88](#), [102](#), [107](#),  
[113](#), [115](#), [117](#), [119](#), [139](#), [147](#)
- epinowcast, [25](#), [116](#), [120](#), [133–136](#), [145](#), [146](#)
- epinowcast(), [7](#), [13](#), [37](#), [54](#), [63](#), [69](#), [72](#), [81](#),  
[82](#), [84](#), [98](#), [100](#), [106](#), [117](#), [134](#), [135](#),  
[143–145](#)
- expose\_functions, [116](#), [117](#)
- extract\_obs\_metadata, [8](#), [9](#), [31](#), [33](#), [34](#), [50](#),  
[106](#), [111](#), [118](#), [123](#), [124](#), [129](#)
- extract\_sparse\_matrix, [8](#), [9](#), [31](#), [33](#), [34](#), [50](#),  
[106](#), [111](#), [118](#), [123](#), [124](#), [129](#)
- findbars, [136](#)
- formula, [52](#)
- germany\_covid19\_hosp, [54](#), [125](#)
- get\_internal\_timestep, [25](#), [27](#), [32](#), [70](#), [112](#),  
[125](#), [129](#), [143](#)
- ggplot2::aes(), [93](#), [94](#)
- gp, [10–14](#), [28](#), [29](#), [31](#), [66](#), [75](#), [126](#), [128](#), [130](#),  
[131](#), [137](#), [138](#), [140–142](#)
- gp(), [28](#), [128](#), [131](#), [138](#)
- gp\_terms, [10–14](#), [28](#), [29](#), [31](#), [66](#), [75](#), [127](#), [128](#),  
[130](#), [131](#), [137](#), [138](#), [140–142](#)
- gp\_terms(), [126](#)
- IDate, [25](#)
- is.Date, [25](#), [27](#), [32](#), [70](#), [112](#), [126](#), [128](#), [143](#)
- latest\_obs\_as\_matrix, [8](#), [9](#), [31](#), [33](#), [34](#), [50](#),  
[106](#), [111](#), [118](#), [123](#), [124](#), [129](#)
- lme4, [12](#), [14](#), [29](#), [64–66](#), [128](#), [130](#), [131](#),  
[136–139](#), [141](#), [142](#)
- ma, [10–14](#), [28](#), [29](#), [31](#), [66](#), [75](#), [127](#), [128](#), [130](#),  
[131](#), [137](#), [138](#), [140–142](#)
- ma(), [11](#)
- parse\_formula, [10–14](#), [28](#), [29](#), [31](#), [66](#), [75](#),  
[127](#), [128](#), [130](#), [130](#), [137](#), [138](#),  
[140–142](#)
- parse\_formula(), [66](#)
- plot.enw\_preprocess\_data, [51](#), [52](#), [89–95](#),  
[121](#), [132](#), [134–136](#), [145](#), [146](#)
- plot.enw\_preprocess\_data(), [50](#), [51](#)
- plot.epinowcast, [51](#), [52](#), [89–95](#), [121](#), [133](#),  
[133](#), [135](#), [136](#), [145](#), [146](#)
- posterior::draws\_df(), [81](#)
- posterior::summarise\_draws(), [96](#)
- primarycensored::dprimarycensored(), [6](#)
- print.enw\_preprocess\_data, [121](#), [133](#), [134](#),  
[134–136](#), [145](#), [146](#)
- print.epinowcast, [121](#), [133](#), [134](#), [135](#), [135](#),  
[136](#), [145](#), [146](#)
- print.summary.enw\_preprocess\_data, [121](#),  
[133–135](#), [136](#), [145](#), [146](#)
- print.summary.enw\_preprocess\_data(),  
[144](#)
- re, [10–14](#), [28](#), [29](#), [31](#), [66](#), [75](#), [127](#), [128](#), [130](#),  
[131](#), [136](#), [137](#), [138](#), [140–142](#)
- re(), [12](#), [14](#), [29](#), [64](#), [128](#), [131](#), [137–139](#), [141](#),  
[142](#)
- remove\_arima\_terms, [10–14](#), [28](#), [29](#), [31](#), [66](#),  
[75](#), [127](#), [128](#), [130](#), [131](#), [137](#), [137](#),  
[138](#), [140–142](#)
- remove\_gp\_terms, [10–14](#), [28](#), [29](#), [31](#), [66](#), [75](#),  
[127](#), [128](#), [130](#), [131](#), [137](#), [138](#),  
[140–142](#)
- remove\_profiling, [68](#), [69](#), [81](#), [88](#), [102](#), [107](#),  
[113](#), [115](#), [117](#), [120](#), [139](#), [147](#)
- remove\_rw\_terms, [10–14](#), [28](#), [29](#), [31](#), [66](#), [75](#),  
[127](#), [128](#), [130](#), [131](#), [137](#), [138](#), [139](#),  
[141](#), [142](#)
- rstan::extract\_sparse\_parts(), [124](#)
- rw, [10–14](#), [28](#), [29](#), [31](#), [66](#), [75](#), [127](#), [128](#), [130](#),  
[131](#), [137](#), [138](#), [140](#), [140](#), [142](#)

`rw()`, [10](#), [12](#), [14](#), [30](#), [35](#), [64](#), [65](#), [126](#), [128](#), [130](#),  
[131](#), [137–139](#), [141](#), [142](#)

`rw_terms`, [10–14](#), [28](#), [29](#), [31](#), [66](#), [75](#), [127](#), [128](#),  
[130](#), [131](#), [137](#), [138](#), [140](#), [141](#), [141](#),  
[142](#)

`scoringutils::as_forecast_sample()`, [13](#)

`split_formula_to_terms`, [10–14](#), [28](#), [29](#), [31](#),  
[66](#), [75](#), [127](#), [128](#), [130](#), [131](#), [137](#), [138](#),  
[140](#), [141](#), [142](#), [142](#)

`stan_fns_as_string`, [25](#), [27](#), [32](#), [70](#), [112](#),  
[126](#), [129](#), [143](#)

`startsWith()`, [42](#)

`stats::lm()`, [65](#)

`stats::model.matrix`, [52](#)

`stats::model.matrix()`, [52](#), [53](#)

`subset_obs`, [5–7](#), [15](#), [38](#), [82](#), [83](#), [96](#), [98](#), [99](#),  
[103](#), [119](#), [143](#)

`summary.enw_preprocess_data`, [121](#),  
[133–136](#), [144](#), [146](#)

`summary.enw_preprocess_data()`, [136](#)

`summary.epinowcast`, [121](#), [133–136](#), [145](#),  
[145](#)

`summary.epinowcast()`, [81–83](#), [98](#)

`tempdir()`, [69](#)

`terms`, [52](#)

`weekdays()`, [40](#)

`write_stan_files_no_profile`, [68](#), [69](#), [81](#),  
[88](#), [102](#), [107](#), [113](#), [115](#), [117](#), [120](#),  
[139](#), [146](#)