Package 'nowcastr'

April 16, 2026

Version 0.2.0

Title Nowcasting with Chain-Ladder Method

Description Tools for performing nowcasting using the Chain-Ladder method <https://en.wikipedia.org/wiki/Chain-ladder_method>. It supports both non-cumulative delay-based estimation and model-based completeness fitting (e.g., using logistic or Gompertz curves) to predict final counts from partially reported data.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

LazyData true

Imports S7, rlang, magrittr, dplyr, tidyr, stats, purrr, ggplot2, scales, cli

Depends R (>= 4.1.0)

Suggests ISOweek, testthat (>= 3.0.0), quarto, shiny, bslib, DT

VignetteBuilder quarto

Config/testthat/edition 3

URL https://github.com/whocov/nowcastr, https://whocov.github.io/nowcastr/

BugReports https://github.com/whocov/nowcastr/issues

NeedsCompilation no

Author Mathias Leroy [aut, cre], Finlay Campbell [aut]

Maintainer Mathias Leroy <mathias.leroy.rpkg@gmail.com>

Repository CRAN

Date/Publication 2026-04-16 08:30:02 UTC

Contents

`calculate_retro_score` *Calculate retro-scores for all groups*

Description

The retro-score is the amount of retro-adjustments / max possible retro-adjustments The higher the better for nowcast_cl() retro_score = n_changes / max_changes or = retro_adjustments / max_retro_adj Notes: "retro-adjustments" = "value changes" retro-score = number of changes / number of ywks (max changes)

Usage

```
calculate_retro_score(
  df,
  col_date_occurrence,
  col_date_reporting,
  col_value,
  group_cols = NULL,
  method = "2D_allgroups",
  max_delay = Inf,
  aggrby
)
```

Arguments

df	A data.frame or tibble.
col_date_occurrence	Column name for the date of occurrence/reference.
col_date_reporting	Column name for the date of reporting.
col_value	Column name for the value.
group_cols	Optional character vector of column names for grouping.
method	'2D_allgroups' (number of changes in 2D triangle) or 'at_least_1_change_by_occ' (number of occurrence dates with at least 2 reported values)
max_delay	Maximum delay to consider. (only works with method '2D_allgroups')
aggrby	A character vector of column names to aggregate by.

Value

A tibble with group cols + retro_score (percentage 0-1)

Examples

```
generate_test_data() %>%
  calculate_retro_score(
    col_date_occurrence = date_occurrence,
    col_date_reporting = date_report,
    col_value = value,
    group_cols = NULL
    # , aggrby = country
    # , method = "at_least_1_change_by_occ"
  )
```

---

explore_nowcast *Explore Nowcast Results with Shiny*

Description

Launch an interactive Shiny application to explore and visualize nowcast results. The app displays a summary table of model statistics and allows users to select groups to view corresponding plots for input data, delay distributions, and nowcast results.

Usage

```
explore_nowcast(nc_obj)
```

Arguments

nc_obj	A nowcast_results S7 object.

Value

A Shiny app object.

`fill_future_reported_values` *Fill future reported values with last known values*

Description

This function completes a data frame to include all combinations of occurrence and reporting dates (within each group). It then fills in missing values by carrying the last known reported value backward in time from future reports to past reports for a given occurrence. This is useful for dealing with right-censored reporting data where reports are updated over time.

Usage

```
fill_future_reported_values(
  df,
  col_date_occurrence,
  col_date_reporting,
  col_value,
  group_cols = NULL,
  max_delay = Inf
)
```

Arguments

`df`	A data.frame or tibble.
`col_date_occurrence`	Column name for the date of occurrence/reference.
`col_date_reporting`	Column name for the date of reporting.
`col_value`	Column name for the value.
`group_cols`	Optional character vector of column names for grouping.
`max_delay`	Inf / 'auto' / NULL / integer. 'auto' or NULL will keep the same max_delay as the input.

Value

A data frame with the same columns as `df`, but with rows added for missing reporting dates and `NA` values in `col_value` filled with the last available observation for each occurrence date within each group.

Examples

```
library(dplyr)

generate_test_data() %>%
  fill_future_reported_values(
    col_date_occurrence = date_occurrence,
    col_date_reporting = date_report,
    col_value = value,
    group_cols = NULL
  )
```

`generate_test_data` *Generate asymptotic test data for nowcating*

Description

Create synthetic long-format test data following an asymptotic delay curve and constant final value. Formula: `value = final_value * (c + 1 * (1 - exp(-b * .data$delay)))`

Usage

```
generate_test_data(
  n_reportdates = 9,
  n_delays = 10,
  reportdate_from = "2025-02-01",
  delay_from = 0,
  time_units = "days",
  final_value = 100,
  c = 0.5,
  b = 0.9,
  remove_delay = FALSE
)
```

Arguments

`n_reportdates`	Integer. Number of consecutive report dates to generate.
`n_delays`	Integer. Number of delay values to generate.
`reportdate_from`	Character or Date. Start report date (e.g. "2025-02-01").
`delay_from`	Integer >= 0. Minimum delay value.
`time_units`	Time units. Accepted values: ("auto", "secs", "mins", "hours", "days", "weeks").
`final_value`	Numeric. Asymptotic target (a in the formula).
`c`	Numeric (0-1). Baseline fraction of final_value added to the curve.
`b`	Numeric. Rate constant controlling how fast y approaches a; larger b $\rightarrow$ faster approach.
`remove_delay`	Logical. If TRUE, remove the delay column in the output.

Details

With defaults this returns n_reportdates × (n_delays - delay_from + 1) rows (9 × 10 = 90). report-date_from may be provided as a Date or parsable character; lubridate units are used for stepping.

Value

A tibble (long format) with columns date_report (Date), date_occurrence (Date), delay (integer, optional) and value (numeric).

Examples

```
generate_test_data()
generate_test_data(n_reportdates = 3, n_delays = 3) # A tibble: 9 × 4
generate_test_data(time_units = "weeks", remove_delay = TRUE)
```

`nowcast_cl` *Nowcasting with Chain-Ladder Method*

Description

Performs nowcasting using non-cumulative Chain-Ladder Method. Input dataset with 2 date columns; 1 value column and a flexible number of group columns. Output dataset with latest reported data joined with `completeness` ratio and final `value_predicted` You have the option to use model-free completeness ratio calculation (faster) or use model-fitted completeness (slower).

Usage

```
nowcast_cl(
  df,
  col_date_occurrence,
  col_date_reporting,
  col_value,
  group_cols = NULL,
  time_units = "days",
  max_delay = NULL,
  max_reportunits = 10,
  max_completeness = 5,
  min_completeness_samples = 1,
  use_weighted_method = TRUE,
  do_propagate_missing_delays = FALSE,
  do_model_fitting = TRUE,
  model_names = c("monomolecular", "vonbertalanffy", "logistic", "gompertz",
    "asymptotic", "linear"),
  do_use_modelled_completeness = TRUE,
  rss_threshold = 0.01
)
```

Arguments

df	A data.frame or tibble.
col_date_occurrence	Column name for the date of occurrence/reference.
col_date_reporting	Column name for the date of reporting.
col_value	Column name for the value.
group_cols	Optional character vector of column names for grouping.
time_units	Time unit for delay calculation. Accepted values: ("auto", "secs", "mins", "hours", "days", "weeks").
max_delay	Max delay to keep in the analysis. Interger or NULL. If NULL, will take the max delay from the data.
max_reportunits	Max number of col_date_reporting to use (in the unit of time_units).
max_completeness	Maximum completeness ratio (e.g., 2 = 200%).
min_completeness_samples	Min number of samples required to calculate completeness. Integer from 1 to max_reportunits.
use_weighted_method	Use weighted method, linear weight to older reported completeness values
do_propagate_missing_delays	Fill missing completeness if lower delay has a value.
do_model_fitting	Fit a model through the completeness by delay. Models are useful for indicators like 'time to 95% compl.' and also soften variability.
model_names	Character vector with names of the models to test for best fit. Accepted values: "monomolecular", "vonbertalanffy", "logistic", "gompertz", "asymptotic", "linear"
do_use_modelled_completeness	Use the modelled completeness values for the nowcasting. Boolean or NULL (for auto selective). (unused if do_model_fitting=FALSE)
rss_threshold	Minimum RSS threshold to use model-fitted values. Only used if do_use_modelled_completeness is NULL. If the RSS of the fit is higher than this then observed completeness is used for nowcasting.

Value

Returns an **S7 object** of class `nowcast_results`. This object serves as a comprehensive container for the analysis results and metadata. Use the `@results` slot to access the primary prediction data frame.

The object includes:

- **Predictions**: The `results` slot contains the latest observed values, the calculated `completeness` ratio, and the `value_predicted`.
- **Calculation**: Predictions are derived using $value\_predicted = value/completeness$.
- **Metadata**: Slots for `params`, `time_start`, `max_delay`, and model diagnostics (`RSS`).

Examples

```
input <- generate_test_data()
res <- input %>%
  nowcast_cl(
    col_date_occurrence = date_occurrence,
    col_date_reporting = date_report,
    col_value = value,
    time_units = "days"
  )

# Access the predicted data:
head(res@results)
```

`nowcast_demo` *Nowcast Dataset for demo and testing*

Description

This dataset contains data for demonstration purposes.

Usage

```
nowcast_demo
```

Format

A data frame with X rows and Y variables:

group text, for grouping

date_occurrence text, representing week of onset

date_report text, representing week of reporting

value number, representing number of cases

`nowcast_eval` *Evaluate Nowcasting Performance*

Description

Evaluates the historical performance of `nowcast_cl()` by repeatedly peeling back the most recent reporting period and comparing predictions against the eventually-reported true values (highest `col_date_reporting` per occurrence date).

Usage

```
nowcast_eval(
  df,
  col_date_occurrence,
  col_date_reporting,
  col_value,
  group_cols = NULL,
  n_past = 10,
  time_units = "days",
  max_delay = NULL,
  max_reportunits = 10,
  max_completeness = 5,
  min_completeness_samples = 1,
  use_weighted_method = TRUE,
  do_propagate_missing_delays = FALSE,
  do_model_fitting = TRUE,
  model_names = c("monomolecular", "vonbertalanffy", "logistic", "gompertz",
    "asymptotic", "linear"),
  do_use_modelled_completeness = TRUE,
  rss_threshold = 0.01
)
```

Arguments

df	A data.frame or tibble.
col_date_occurrence	Column name for the date of occurrence/reference.
col_date_reporting	Column name for the date of reporting.
col_value	Column name for the value.
group_cols	Optional character vector of column names for grouping.
n_past	Integer. Number of past reporting periods to evaluate. Each iteration peels off one reporting period (in time_units).
time_units	Time unit for delay calculation. Accepted values: ("auto", "secs", "mins", "hours", "days", "weeks").
max_delay	Max delay to keep in the analysis. Interger or NULL. If NULL, will take the max delay from the data.
max_reportunits	Max number of col_date_reporting to use (in the unit of time_units).
max_completeness	Maximum completeness ratio (e.g., 2 = 200%).
min_completeness_samples	Min number of samples required to calculate completeness. Integer from 1 to max_reportunits.
use_weighted_method	Use weighted method, linear weight to older reported completeness values

`do_propagate_missing_delays`
: Fill missing completeness if lower delay has a value.

`do_model_fitting`
: Fit a model through the completeness by delay. Models are useful for indicators like 'time to 95% compl.' and also soften variability.

`model_names`
: Character vector with names of the models to test for best fit. Accepted values: "monomolecular", "vonbertalanffy", "logistic", "gompertz", "asymptotic", "linear"

`do_use_modelled_completeness`
: Use the modelled completeness values for the nowcasting. Boolean or NULL (for auto selective). (unused if do_model_fitting=FALSE)

`rss_threshold`
: Minimum RSS threshold to use model-fitted values. Only used if do_use_modelled_completeness is NULL. If the RSS of the fit is higher than this then observed completeness is used for nowcasting.

Value

An S7 object of class `nowcast_eval_results` with slots:

detail data.frame with per-prediction errors (one row per occurrence date x delay x past period).

summary data.frame with aggregated SMAPE and proportion_pred_is_better, by group and delay.

params list of parameters used.

n_past number of past periods evaluated.

time_start POSIXct start time.

time_end POSIXct end time.

Examples

```
input <- generate_test_data()
eval_res <- nowcast_eval(
  df = input,
  col_date_occurrence = date_occurrence,
  col_date_reporting = date_report,
  col_value = value,
  n_past = 10,
  time_units = "days"
)
```

`nowcast_eval_results` *S7 object class for* `nowcast_eval()` *Results*

Description

The `nowcast_eval()` function returns an object of this class.

Usage

```
nowcast_eval_results(
  detail,
  summary,
  params,
  n_past,
  time_start,
  time_end
)
```

Arguments

detail	data.frame. Per-prediction errors with columns for observed value, predicted value, true value.
summary	data.frame. Aggregated metrics per group x delay: SMAPE (pred and obs), SMAPE improvement, proportion_pred_is_better, Wilson CIs.
params	list. Parameters used for the evaluation run.
n_past	integer. Number of past reporting periods evaluated.
time_start	POSIXct. Time the function started.
time_end	POSIXct. Time the function ended.

Value

An S7 object of class `nowcast_eval_results` with the following slots:

detail Data frame. Per-prediction errors with columns for observed value, predicted value, true value.

summary Data frame. Aggregated metrics per group x delay: SMAPE (pred and obs), SMAPE improvement, proportion_pred_is_better, Wilson CIs.

params List. Parameters used for the evaluation run.

n_past Numeric. Number of past reporting periods evaluated.

time_start POSIXct. Time the function started.

time_end POSIXct. Time the function ended.

`nowcast_results` *S7 object class for* `nowcast_cl()` *Results*

Description

The `nowcast_cl()` function returns an object of this class.

Usage

```
nowcast_results(name, params, time_start, time_end, n_groups, max_delay,
  data, completeness, delays, models, results)
```

Arguments

name	A character string with a timestamp for the run.
params	A list with the parameters used for the nowcasting (unevaluated call).
time_start	the sys time at which the function started.
time_end	the sys time at which the function ended.
n_groups	The number of groups processed.
max_delay	The maximum delay used.
data	Dataframe. The original input data frame (with only required columns).
completeness	Dataframe. The original input data frame with delays and completeness columns.
delays	Dataframe. A data frame with the final aggregated completeness estimates per delay (+ modelled column if do_model_fitting was TRUE).
models	Dataframe. The resulting fitted models (empty data frame if do_model_fitting was FALSE)
results	Dataframe. A data frame with the resulting nowcasting predictions.

Value

An **S7 object** of class nowcast_results. This object is a structured container for the entire nowcasting pipeline output. It consists of the following properties (slots):

name Character. A unique timestamp identifier for the run (YYYYMMDD_HHMMSS).

params List. The evaluated parameters and arguments used in the function call.

time_start, time_end POSIXct. Timestamps marking the duration of the calculation.

n_groups Numeric. The total count of unique groups processed.

max_delay Numeric. The maximum reporting delay (in time_units) considered.

data Data frame. The subset of the original input used for the analysis.

completeness Data frame. Detailed row-level completeness calculations and delays.

delays Data frame. Aggregated completeness estimates per delay unit, including both observed and (optionally) modelled values.

models Data frame. Results of the non-linear model fitting, including RSS and model types. Returns an empty data frame if do_model_fitting was FALSE.

results Data frame. The final nowcasting table containing predicted values.

nowcast_test_data *Nowcast Minimal Dataset for testing*

Description

This dataset contains data for testing purposes.

Usage

```
nowcast_test_data
```

Format

A data frame with X rows and Y variables:

country text, for grouping

onset_week text, representing week of onset

report_week text, representing week of reporting

cases number, representing number of cases

plot_delays *Plot Reporting Completeness by Delay*

Description

Creates a scatter plot of reporting completeness against reporting delay. If a `col_completeness_modelled` is present, it will be shown as a dotted line.

Usage

```
plot_delays(
  df,
  col_completeness_obs,
  col_completeness_modelled = "",
  group_cols = NULL,
  color1 = "#222222",
  color2 = "firebrick2",
  limits_y = c(NA, NA)
)
```

Arguments

df	A data.frame containing 'delay' and col_completeness columns. An optional modelled column
col_completeness_obs	Column name for the Observed Completeness. (dots)
col_completeness_modelled	Column name for the Modelled Completeness. (line)
group_cols	Optional character vector of column names for grouping.
color1	Color for observed data. (dots)
color2	Color for modelled data. (line)
limits_y	vector to be passed to limits of ggplot2::scale_y_continuous.

Value

A ggplot object showing completeness vs. delay.

Examples

```
delays <- data.frame(
  delay = 0:9,
  completeness = c(0.509, 0.802, 0.920, 0.967, 0.987, 0.995, 0.998, 0.999, 1, NA),
  modelled = c(0.509, 0.802, 0.920, 0.968, 0.987, 0.995, 0.998, 0.999, 1, 1)
)
plot_delays(
  df = delays,
  col_completeness_obs = completeness,
  col_completeness_modelled = modelled
)
```

---

plot_millipede *Millipede plot*

Description

Draw a line plot where each line represents one reporting date:

- x = date of occurrence
- y = value (e.g., counts or proportions)
- group + fill = date of reporting

Usage

```
plot_millipede(
  df,
  col_value,
  col_date_occurrence,
  col_date_reporting,
  scale_percent = FALSE
)
```

Arguments

df	A data.frame or tibble.
col_value	Column name for the value.
col_date_occurrence	Column name for the date of occurrence/reference.
col_date_reporting	Column name for the date of reporting.
scale_percent	Logical; if TRUE formats the fill legend as percentages and labels it "Percentage".

Details

When there are reporting delays the plot look like this:

```
---\--\--\--\
    \  \  \  \
     \  \  \  \
```

Value

A ggplot object.

See Also

ggplot2::geom_line, ggplot2::scale_color_viridis_c

Examples

```
generate_test_data() %>%
  plot_millipede(
    col_value = value,
    col_date_occurrence = date_occurrence,
    col_date_reporting = date_report,
    scale_percent = FALSE
  )
```

plot_nc_input *Plot Nowcast Input Data*

Description

Can plot 2 types of plot: option="triangle" or "millipede"

Usage

```
plot_nc_input(
  df,
  col_value,
  col_date_occurrence,
  col_date_reporting,
  group_cols = NULL,
  option = "millipede",
  do_rescale = TRUE,
  do_facet_groups = TRUE
)
```

Arguments

df	A data.frame or tibble.
col_value	Column name for the value.
col_date_occurrence	Column name for the date of occurrence/reference.
col_date_reporting	Column name for the date of reporting.
group_cols	Optional character vector of column names for grouping.
option	"millipede" or "triangle".
do_rescale	Rescale values 0-1. Boolean.
do_facet_groups	Boolean. Should groups be faceted?

Value

A ggplot object.

plot_nowcast *Plot Nowcasting Predictions*

Description

Compares observed data with nowcasted predictions over the occurrence date. Observed values are plotted as a solid grey line, and predicted values as a dashed black line.

Usage

```
plot_nowcast(
  df,
  col_date_occurrence,
  col_value,
  col_value_predicted,
  group_cols = NULL,
  color1 = "#333333",
  color2 = "firebrick1"
)
```

Arguments

df	A data.frame or tibble.
col_date_occurrence	Column name for the date of occurrence/reference.
col_value	Column name for the value.
col_value_predicted	Column name for the Predicted Value.
group_cols	Optional character vector of column names for grouping.
color1	Color for observed data.
color2	Color for predicted data.

Value

A ggplot object comparing observed and predicted values.

Examples

```
df_nowcast <- data.frame(
  date_occurrence = as.Date("2023-01-01") + 0:9,
  value_observed = c(10, 12, 15, 13, 18, 20, 22, 24, 25, 20),
  value_predicted = c(10, 12, 15, 13, 18, 20, 22, 25, 28, 30)
)
plot_nowcast(
  df = df_nowcast,
  col_value = value_observed,
  col_date_occurrence = date_occurrence,
```

```
  col_value_predicted = value_predicted
)
```

`plot_nowcast_eval` *Plot Nowcast Evaluation Results*

Description

Plots a horizontal bar chart of nowcasting evaluation metrics per group, at a selected delay. Two panels are shown side by side:

- **SMAPE improvement**: median per-prediction SMAPE difference (obs minus pred; positive = prediction is better), with IQR as error bar.
- **Proportion better**: share of past periods where prediction beat raw observed, centered at 0 (0.5 = no improvement), with Wilson 95% CI.

Bars are coloured by whether the improvement is significant (IQR / CI fully above or below zero) or not.

Usage

```
plot_nowcast_eval(
  x,
  delay = NULL,
  color_good = "dodgerblue1",
  color_bad = "firebrick1",
  alpha_less = 0.35,
  ...
)
```

Arguments

`x`	A `nowcast_eval_results` S7 object from `nowcast_eval()`.
`delay`	Numeric. Which delay to plot. Defaults to the minimum delay in the data.
`color_good`	Character. Colour for significantly better predictions.‘.
`color_bad`	Character. Colour for significantly worse predictions.‘.
`alpha_less`	alpha value for the "less significant" bars, 0-1.
`...`	Ignored.

Value

A `ggplot` object.

Examples

```
input <- generate_test_data()
eval_res <- nowcast_eval(
  df = input,
  col_date_occurrence = date_occurrence,
  col_date_reporting = date_report,
  col_value = value,
  n_past = 10,
  time_units = "days"
)
plot(eval_res)
plot(eval_res, delay = 2)
```

---

plot_nowcast_eval_by_delay

Plot Nowcast Evaluation by Delay

Description

Plots evaluation metric as a function of delay, faceted by group. The y-axis shows how much the nowcast improves over raw observed values, across all delays. Background shading indicates the direction of improvement.

Usage

```
plot_nowcast_eval_by_delay(
  x,
  indicator = "SMAPE_improvement_med",
  color_good = "dodgerblue1",
  color_bad = "firebrick1",
  ...
)
```

Arguments

x	A nowcast_eval_results S7 object from nowcast_eval().
indicator	Character. Which metric to plot on the y-axis. One of: "SMAPE_improvement_med" (default), "SMAPE_improvement_mean", or "proportion_pred_is_better".
color_good	Character. Fill colour for the "better" region.‘.
color_bad	Character. Fill colour for the "worse" region.‘.
...	Ignored.

Value

A ggplot object.

Examples

```
input <- generate_test_data()
eval_res <- nowcast_eval(
  df = input,
  col_date_occurrence = date_occurrence,
  col_date_reporting = date_report,
  col_value = value,
  n_past = 10,
  time_units = "days"
)
plot_nowcast_eval_by_delay(eval_res)
plot_nowcast_eval_by_delay(eval_res, indicator = "proportion_pred_is_better")
```

---

plot_nowcast_eval_detail

Plot Nowcast Evaluation Detail Over Time

Description

For a selected delay, plots predicted and observed values over time alongside the eventual true value. Vertical segments show which estimate (raw observed or predicted) was closer to truth for each occurrence date.

Usage

```
plot_nowcast_eval_detail(
  x,
  delay = NULL,
  color_good = "dodgerblue1",
  color_bad = "firebrick1",
  ...
)
```

Arguments

x	A nowcast_eval_results S7 object from nowcast_eval().
delay	Numeric. Which delay to plot. Defaults to the minimum delay in the data.
color_good	Character. Colour when prediction beats raw observed.‘.
color_bad	Character. Colour when raw observed beats prediction.‘.
...	Ignored.

Value

A ggplot object.

Examples

```
input <- generate_test_data()
eval_res <- nowcast_eval(
  df = input,
  col_date_occurrence = date_occurrence,
  col_date_reporting = date_report,
  col_value = value,
  n_past = 10,
  time_units = "days"
)
plot_nowcast_eval_detail(eval_res)
plot_nowcast_eval_detail(eval_res, delay = 7)
```

---

plot_triangle *Triangle plot*

Description

Draw a triangular heatmap where:

- x = date of occurrence
- y = date of reporting
- fill = value (e.g., counts or proportions)

Usage

```
plot_triangle(
  df,
  col_value,
  col_date_occurrence,
  col_date_reporting,
  scale_percent = FALSE
)
```

Arguments

df	A data.frame or tibble.
col_value	Column name for the value.
col_date_occurrence	Column name for the date of occurrence/reference.
col_date_reporting	Column name for the date of reporting.
scale_percent	Logical; if TRUE formats the fill legend as percentages and labels it "Percentage".

Details

The plot is triangular because reporting dates cannot precede occurrence dates.

```
|---------/
|        /
|       /
|      /
|     /
|----/
```

Uses geom_raster with coord_fixed for square tiles.

Value

A ggplot object.

See Also

ggplot2::geom_raster, ggplot2::scale_fill_viridis_c

Examples

```
generate_test_data() %>%
  plot_triangle(
    col_value = value,
    col_date_occurrence = date_occurrence,
    col_date_reporting = date_report,
    scale_percent = FALSE
  )
```

---

rm_repeated_values *Remove duplicated reported values in reporting matrix*

Description

Remove duplicated reported values in reporting matrix

Usage

```
rm_repeated_values(
  df,
  col_date_occurrence,
  col_date_reporting,
  col_value,
  group_cols = NULL
)
```

Arguments

df	A data.frame or tibble.
col_date_occurrence	Column name for the date of occurrence/reference.
col_date_reporting	Column name for the date of reporting.
col_value	Column name for the value.
group_cols	Optional character vector of column names for grouping.

Value

A tibble with the same columns as df, but with rows removed.

Examples

```
library(dplyr)
generate_test_data(n_delays = 20, n_reportdates = 20) %>%
  mutate(value = round(value, 1)) %>% ## make values identical
  rm_repeated_values(
    col_value = value,
    col_date_occurrence = date_occurrence,
    col_date_reporting = date_report
  ) %>%
  plot_triangle(
    col_value = value,
    col_date_occurrence = date_occurrence,
    col_date_reporting = date_report
  )
```

Index