

Package: csalert (via r-universe)

October 15, 2026

Title Alerts from Public Health Surveillance Data

Version 2026.10.15

Description Helps create alerts and determine trends by using various methods to analyze public health surveillance data. The primary analysis method is based upon a published analytics strategy by Benedetti (2019) <[doi:10.5588/pha.19.0002](https://doi.org/10.5588/pha.19.0002)>.

Depends R (>= 4.1.0)

License MIT + file LICENSE

URL <https://niphr.github.io/csalert/>, <https://github.com/niphr/csalert>

BugReports <https://github.com/niphr/csalert/issues>

Encoding UTF-8

LazyData true

Imports data.table, ggplot2, glm2, cstydy (>= 2026.7.1), cstime, digest, lubridate, matrixStats, stringr, surveillance, utils

Suggests testthat, knitr, rmarkdown, glue, csmaps, mem, scoringutils

VignetteBuilder knitr

Remotes niphr/cstydy

Config/Needs/website niphr/cstemplate

Config/roxygen2/version 8.0.0

Roxygen list(markdown = TRUE)

Config/pak/sysreqs libicu-dev

Repository <https://niphr.r-universe.dev>

Date/Publication 2026-10-15 17:13:18 UTC

RemoteUrl <https://github.com/niphr/csalert>

RemoteRef HEAD

RemoteSha dc73e05e12b96890c3b18b2b8d987f6f6e38b7c2

Contents

add_holiday_effect	2
compare_results	4
csfmt_ensemble_v3	5
csfmt_interpret	7
csfmt_parse	8
csfmt_reporting_triangle_v3	9
csfmt_var	10
ens_add_rate	11
ens_collapse	13
isoyearweek_week_start	14
mem_thresholds_v1	15
nowcast_backtest	17
nowcast_censor	19
nowcast_delay_ecdf_v1	20
nowcast_passthrough_to_ensemble_v1	22
nowcast_score_v1	24
nowcast_truth	25
print.csfmt_ensemble_v3	27
q_label	27
q_value	28
qc_surveillance_data_v1	29
qc_week_over_week_v1	30
reporting_completion_trend_v1	32
reporting_completion_v1	33
reporting_triangle_matrix	35
rolling_slope_matrix	36
set_time_series_id	38
short_term_trend	39
short_term_trend_sts_v1	43
signal_detection_hlm	44
simulate_baseline_data	46
simulate_seasonal_outbreak_data	48
simulate_spike_outbreak_data	50
validate_ensemble	52
Index	54

add_holiday_effect	<i>Multiply simulated counts on public holidays</i>
--------------------	---

Description

Multiplies the count `n` by `holiday_effect` on each date that `holiday_data` marks as a holiday.

Usage

```
add_holiday_effect(data, holiday_data, holiday_effect = 2)
```

Arguments

`data` A `csfmt_rts_data_v1` with date and `n`.
`holiday_data` A `data.table` with date and a logical `is_holiday`.
`holiday_effect` The factor for a holiday.

Details

An integer `n`, as from [simulate_baseline_data\(\)](#), stays an integer. So a factor that gives a fraction truncates the count, with a warning.

Value

A copy of `data` with `n` changed, and a holiday column: the value of `is_holiday` on the dates in `holiday_data`, and NA on other dates.

See Also

`vignette("csalert", package = "csalert")`, which runs it.

Other data simulation functions: [simulate_baseline_data\(\)](#), [simulate_seasonal_outbreak_data\(\)](#), [simulate_spike_outbreak_data\(\)](#)

Examples

```
library(data.table)
set.seed(4)
baseline <- simulate_baseline_data(
  start_date = as.Date("2018-01-01"),
  end_date = as.Date("2019-12-31"),
  seasonal_pattern_n = 1,
  weekly_pattern_n = 1,
  alpha = 3,
  beta = 0,
  gamma_1 = 0.8,
  gamma_2 = 0.6,
  gamma_3 = 0.8,
  gamma_4 = 0.4,
  phi = 4,
  shift_1 = 29
)
holidays <- data.table(
  date = as.Date(c("2018-12-25", "2019-01-01", "2019-12-25")),
  is_holiday = TRUE
)
d <- add_holiday_effect(baseline, holiday_data = holidays, holiday_effect = 2)
print(d[holiday == TRUE, .(date, n, holiday)])
```

compare_results

Compare two collapsed result sets, column by column

Description

Joins two runs on `time_series_id` and `isoyearweek`, and returns one row per series, week and value column. `csfmt_interpret()` finds the value columns of current, and the parts of each name.

Usage

```
compare_results(current, previous)
```

Arguments

current, previous

The output of `ens_collapse()` for two runs.

Details

`time_series_id` is a hash of the identity columns, so a series has the same id in both runs. A week in only one run gets NA for the other run.

Value

A long `data.table` with `time_series_id`, `isoyearweek`, the columns among `indicator_tag`, `location_code`, `age` and `sex` that exist, and:

- `column`: the name of the value column,
- `cur`, `prv`: its value in current and in previous,
- `role`, `q`, `level`: the parts of its name.

Identity columns MUST use the csfmt schema names

`csfmt_interpret()` reads every column outside the csfmt schema as a value column. `location_code` and `indicator_tag` are schema names, but `location` and `indicator` are not. So with `location` or `indicator`, `cur` and `prv` become character.

This function still returns a table. `qc_week_over_week_v1()` then stops with non-numeric argument to binary operator.

See Also

`vignette("pipeline", package = "csalert")`, section 9.

Other quality control functions: `qc_surveillance_data_v1()`, `qc_week_over_week_v1()`

Examples

```
# 40 reference weeks, each reported 3, 10 and 17 days after its Monday
mondays <- as.Date("2023-01-02") + 7 * (0:39)
set.seed(1)
d <- data.table::data.table(
  isoyearweek_reference = format(rep(mondays, each = 3), "%G-%V"),
  reporting_date = rep(mondays, each = 3) + rep(c(3, 10, 17), 40),
  numerator = rpois(120, c(30, 15, 5)),
  indicator_tag = "x", location_code = "nation", age = "total", sex = "total"
)
d <- d[reporting_date <= mondays[40] + 6]

id <- c("indicator_tag", "location_code", "age", "sex")

# Set the seed inside `run()`. Then the two runs differ only by the data
# revision, and not by their Monte-Carlo draws.
run <- function(x) {
  set.seed(2)
  ens_collapse(nowcast_delay_ecdf_v1(
    csfmt_reporting_triangle_v3(x, id_cols = id),
    max_delay_days = 21, n_sim = 200
  ))
}

# last week's run saw one reference week before a correction raised it
cur <- run(d)
d_prv <- data.table::copy(d)
d_prv[
  isoyearweek_reference == format(mondays[11], "%G-%V"),
  numerator := numerator - 5
]
prv <- run(d_prv)

# with the seed held fixed, only the corrected week moves
compare_results(cur, prv)[q == 0.5 & abs(cur - prv) > 0]
```

csfmt_ensemble_v3	<i>Build a csfmt_ensemble_v3</i>
-------------------	----------------------------------

Description

Builds the working format of the pipeline. `$data` is a `data.table` with one row per series and week. `$draws` holds one matrix per measure, with one row per row of `$data` and one column per draw.

Usage

```
csfmt_ensemble_v3(data, id_cols, time_col = "isoyearweek", draws = list())
```

Arguments

<code>data</code>	A <code>data.table</code> with the identity columns and <code>time_col</code> .
<code>id_cols</code>	The identity columns that define one series.
<code>time_col</code>	The column that orders time within a series.
<code>draws</code>	A named list of matrices, one per measure, with <code>nrow(data)</code> rows in the row order of data.

Details

The constructor copies data, adds the ids of `set_time_series_id()`, and sorts the rows by series and `time_col`. It adds `time_series_internal_id`, 1..n within each series, and puts the draw rows in the same order. The nowcast engines call it for you.

Value

A `csfmt_ensemble_v3`.

See Also

`vignette("pipeline", package = "csalert")`, which builds an ensemble with a nowcast engine.

Other ensemble format functions: `print.csfmt_ensemble_v3()`, `set_time_series_id()`, `validate_ensemble()`

Examples

```
d <- data.table::data.table(
  location_code = "nation",
  age = "total",
  isoyearweek = c("2023-01", "2023-02", "2023-03")
)
set.seed(1)
ens <- csfmt_ensemble_v3(
  d,
  id_cols = c("location_code", "age"),
  draws = list(numerator_nowcasted = matrix(rpois(3 * 100, 20), nrow = 3))
)
ens

# $data holds the identity columns and the sort keys. The trailing [] makes
# data.table print a table that was last changed by reference.
ens$data[]

# $draws holds one matrix per measure, with rows = weeks, columns = draws
dim(ens$draws$numerator_nowcasted)
```

csfmt_interpret	<i>Split every value column name of a table into its parts</i>
-----------------	--

Description

Runs `csfmt_parse()` on every value column, and returns one row per column. Code can then find a column by its parts.

Usage

```
csfmt_interpret(d, value_cols = NULL)
```

Arguments

d	A data.table or data.frame.
value_cols	The columns to read. NULL reads every value column.

Details

A value column is any column outside a fixed list of 23 structural names. The list includes location_code, age, sex, isoyearweek, indicator_tag, original and the time_series_* columns.

Value

A data.table with the columns column, measure, denom, role, q, level, per, suffix and interpretable. interpretable is TRUE when the name has a role, a quantile label or a level.

See Also

vignette("pipeline", package = "csalert"), whose naming-grammar section shows the grammar. `compare_results()` uses this function.

Other naming grammar functions: `csfmt_parse()`, `csfmt_var()`, `q_label()`, `q_value()`

Examples

```
d <- data.table::data.table(
  isoyearweek = "2023-01",
  numerator_nowcasted_q50x0 = 42,
  numerator_nowcasted_vs_denominator_nowcasted_pr100_q50x0 = 8.4,
  numerator_nowcasted_status_prob_high = 0.3,
  a_column_outside_the_grammar = 1
)

# isoyearweek is structural, so it has no row. The last column is a value
# column that the grammar cannot read, so its interpretable is FALSE.
csfmt_interpret(d)
```

csfmt_parse

Split a measure column name into its parts

Description

Reads a name that `csfmt_var()` wrote back into its parts. From the right, it removes a `_n` suffix, a `_pr<per>` scale, a quantile label and a `_prob_<level>` level. It then removes a role and `_vs_<denom>`, and the rest is the measure.

Usage

```
csfmt_parse(varname)
```

Arguments

varname The column name.

Details

The roles it knows are observed, nowcasted, forecasted, trend, baseline, status and hlmstatus.

Value

A named list of the parts that are present: measure, denom, role, q, level, per and suffix, in that order.

Where it does not invert csfmt_var

The parse cannot tell which of two role words was the role. On the rate name of the package, it gets the denominator wrong:

```
csfmt_var("numerator_nowcasted", denom = "denominator_nowcasted", per = 100)
#> "numerator_nowcasted_vs_denominator_nowcasted_pr100"
csfmt_parse("numerator_nowcasted_vs_denominator_nowcasted_pr100")$denom
#> "denominator"    # "_nowcasted" of the denominator was read as the role
```

The parse is reliable for a name with one role word, such as `numerator_nowcasted_q50x0`. Check the result when the measure or the denominator ends in a role word.

See Also

`vignette("pipeline", package = "csalert")`, whose naming-grammar section shows this limit.

Other naming grammar functions: `csfmt_interpret()`, `csfmt_var()`, `q_label()`, `q_value()`

Examples

```
csfmt_parse("numerator_nowcasted_q50x0")

# the limit: the role is removed before the _vs_ part is read, so a
# denominator that ends in a role word loses that word
csfmt_parse("numerator_nowcasted_vs_denominator_nowcasted_pr100")
```

```
csfmt_reporting_triangle_v3
```

```
    Build a csfmt_reporting_triangle_v3
```

Description

Builds the input of the nowcast engines: counts by reference ISO week and by the date each count was reported. The newest reporting date is the as-of boundary.

Usage

```
csfmt_reporting_triangle_v3(
  data,
  id_cols,
  reference_col = "isoyearweek_reference",
  reporting_col = "reporting_date",
  value_col = "numerator"
)
```

Arguments

<code>data</code>	A data.table with the identity, reference, reporting and count columns.
<code>id_cols</code>	The identity columns that define one series.
<code>reference_col</code>	The reference ISO-week column, written "YYYY-WW".
<code>reporting_col</code>	The column with the date each count was reported. Its class MUST be exactly Date, so an IDate is an error. It MUST NOT hold NA. One NA would make the as-of boundary NA. Every nowcast engine would then return the observed totals with no warning.
<code>value_col</code>	The count column.

Details

The triangle is sparse. An absent cell on or before the as-of date is a zero, and a cell after it is not reported yet. The constructor copies data and adds the ids of [set_time_series_id\(\)](#). It stops with an error when a reporting date comes before its reference Monday, or a count is negative.

Value

A `csfmt_reporting_triangle_v3`: a data.table with the attributes `as_of`, a Date, and `id_cols`, `reference_col`, `reporting_col` and `value_col`.

See Also

vignette("pipeline", package = "csalert"), which takes a triangle through the whole pipeline.
 Other reporting triangle functions: [isoyearweek_week_start\(\)](#), [reporting_triangle_matrix\(\)](#)

Examples

```
# 40 reference weeks, each reported 3, 10 and 17 days after its Monday. The
# data stops at one as-of date, so the newest weeks are still incomplete.
cal <- cstime::dates_by_isoyearweek
i <- match("2023-01", cal$isoyearweek)
set.seed(1)
monday <- as.Date(cal$mon[i + rep(0:39, each = 3)])
d <- data.table::data.table(
  isoyearweek_reference = cal$isoyearweek[i + rep(0:39, each = 3)],
  reporting_date = monday + rep(c(3, 10, 17), 40),
  numerator = rpois(120, c(30, 15, 5)),
  indicator_tag = "x", location_code = "nation", age = "total", sex = "total"
)
d <- d[reporting_date <= as.Date(cal$mon[i + 39]) + 6]

tri <- csfmt_reporting_triangle_v3(
  d,
  id_cols = c("indicator_tag", "location_code", "age", "sex")
)

# the as-of boundary is the newest reporting date in the data
attr(tri, "as_of")
head(tri, 3)
```

csfmt_var

Build a measure column name from its parts

Description

Joins the parts in the order <measure>[_vs_<denom>][_<role>][_<q-label> | _prob_<level>][_pr<per>][_<suffix>]
 The pipeline names its draw matrices and quantile columns with it, so build a name with it before
 you look a column up.

Usage

```
csfmt_var(
  measure,
  denom = NULL,
  role = NULL,
  q = NULL,
  level = NULL,
  per = NULL,
  suffix = NULL
)
```

Arguments

measure	The name of the measure, for example "consults_r80".
denom	A denominator. It adds _vs_<denom>.
role	A role, for example "nowcasted", "trend" or "status". It adds _<role>.
q	A probability. It adds the label from q_label() . Give q or level, not both.
level	A status level. It adds _prob_<level>.
per	A rate scale. 100 adds _pr100.
suffix	A unit suffix, added as written, for example "_n".

Value

The column name.

See Also

[vignette\("pipeline", package = "csalert"\)](#), whose naming-grammar section builds and parses a name.

Other naming grammar functions: [csfmt_interpret\(\)](#), [csfmt_parse\(\)](#), [q_label\(\)](#), [q_value\(\)](#)

Examples

```
csfmt_var("numerator", role = "nowcasted", q = 0.5) # "numerator_nowcasted_q50x0"
csfmt_var("consults", denom = "population", per = 100) # a rate column name
```

ens_add_rate	<i>Add a rate to an ensemble</i>
--------------	----------------------------------

Description

Divides one draw matrix by another, draw by draw, and adds the result as a new draw matrix. Column j is the same draw in every measure, so the rate carries the uncertainty of both.

Usage

```
ens_add_rate(x, ...)

## S3 method for class 'csfmt_ensemble_v3'
ens_add_rate(x, numerator, denominator, per = 100, name = NULL, ...)
```

Arguments

x	The csfmt_ensemble_v3 that holds both measures.
...	Passed to the method.
numerator, denominator	Two draw matrices in \$draws.
per	The scale of the rate. 100 gives a percentage.
name	The name of the new measure. NULL uses csfmt_var(numerator, denom = denominator, per = per).

Details

A zero denominator gives NA, not 0. The rate is capped at per, because the numerator is a subset of the denominator, and a draw above the cap gives a warning.

Value

x with the rate in \$draws.

See Also

vignette("pipeline", package = "csalert"), stage 4.

Other ensemble operations: [ens_collapse\(\)](#), [mem_thresholds_v1\(\)](#), [short_term_trend\(\)](#), [signal_detection_hlm\(\)](#)

Examples

```
d <- data.table::data.table(
  location_code = "nation",
  age = "total",
  isoyearweek = c("2023-01", "2023-02", "2023-03")
)
# The numerator is a SUBSET of the denominator: positive tests out of tests
# taken. So simulate the denominator first, and the numerator from it.
set.seed(1)
denom <- matrix(rpois(3 * 100, 200), nrow = 3)
numer <- matrix(rbinom(length(denom), size = denom, prob = 0.10), nrow = 3)
ens <- csfmt_ensemble_v3(
  d,
  id_cols = c("location_code", "age"),
  draws = list(
    numerator_nowcasted = numer,
    denominator_nowcasted = denom
  )
)

ens <- ens_add_rate(
  ens,
  numerator = "numerator_nowcasted",
  denominator = "denominator_nowcasted",
  per = 100
)
```

```
# the rate is a third draw matrix, named by the grammar
names(ens$draws)

# its interval carries the uncertainty of both measures
r <- ens_collapse(ens, probs = c(0.05, 0.5, 0.95))
r[, .(
  isoyearweek,
  lo = numerator_nowcasted_vs_denominator_nowcasted_pr100_q05x0,
  med = numerator_nowcasted_vs_denominator_nowcasted_pr100_q50x0,
  hi = numerator_nowcasted_vs_denominator_nowcasted_pr100_q95x0
)]
```

ens_collapse

Collapse the draws of an ensemble to quantiles

Description

Reduces every draw matrix to quantile columns, and drops the draws. It is the last stage: no stage takes a collapsed table as input.

Usage

```
ens_collapse(x, ...)

## S3 method for class 'csfmt_ensemble_v3'
ens_collapse(
  x,
  probs = c(0.025, 0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95, 0.975),
  heal = FALSE,
  ...
)
```

Arguments

x	The csfmt_ensemble_v3 to collapse.
...	Passed to the method.
probs	The probabilities of the quantile columns.
heal	If TRUE, return a csfmt_rts_data_v3 from <code>cstidy::set_csfmt_rts_data_v3()</code> , which adds the calendar columns.

Details

A status matrix, one with a `levels` attribute, also gives a `<measure>_prob_<level>` column per level, with the share of draws at that level. Its quantile columns hold the code of the lowest level whose cumulative share reaches the probability. Both ignore NA draws.

Value

A copy of `$data` with a `<measure>_<q-label>` column for each measure and probability. It is a `data.table`, or a `csfmt_rts_data_v3` when `heal = TRUE`.

See Also

`vignette("pipeline", package = "csalert")`, stage 8.

Other ensemble operations: `ens_add_rate()`, `mem_thresholds_v1()`, `short_term_trend()`, `signal_detection_hlm()`

Examples

```
d <- data.table::data.table(
  location_code = "nation",
  age = "total",
  isoyearweek = c("2023-01", "2023-02", "2023-03")
)
set.seed(1)
ens <- csfmt_ensemble_v3(
  d,
  id_cols = c("location_code", "age"),
  draws = list(numerator_nowcasted = matrix(rpois(3 * 100, 20), nrow = 3))
)

# one column per probability, named by the naming grammar
r <- ens_collapse(ens, probs = c(0.05, 0.5, 0.95))
r[, .(
  isoyearweek,
  lo = numerator_nowcasted_q05x0,
  med = numerator_nowcasted_q50x0,
  hi = numerator_nowcasted_q95x0
)]

# the quantile columns are all that the collapse adds, and the draws are gone
setdiff(names(r), names(ens$data))
```

isoyearweek_week_start

The Monday that starts an ISO week

Description

Returns the Monday that starts each ISO week, as a Date. Every delay in a reporting triangle counts days from this Monday.

Usage

```
isoyearweek_week_start(isoyearweek)
```

Arguments

isoyearweek A character vector of ISO weeks, written "YYYY-WW".

Value

A Date vector as long as isoyearweek, with NA for a value that is not an ISO week in `cstime::dates_by_isoyearweek`.

See Also

`vignette("pipeline", package = "csalert")`, which counts every delay and age in days from this Monday.

Other reporting triangle functions: `csfmt_reporting_triangle_v3()`, `reporting_triangle_matrix()`

Examples

```
# ISO week 2026-01 starts on Monday 2025-12-29
isoyearweek_week_start(c("2026-01", "2026-02"))

# a report on 2026-01-08 has delay day 10 in reference week 2026-01
as.Date("2026-01-08") - isoyearweek_week_start("2026-01")

# a string that is not an ISO week gives NA
isoyearweek_week_start("2026-99")
```

mem_thresholds_v1	<i>Classify every draw into MEM intensity levels</i>
-------------------	--

Description

Fits Moving Epidemic Method (MEM) thresholds for each season from the earlier seasons of the same series, and classifies every draw against them. After `ens_collapse()`, each week has the share of its draws at each of five levels.

Usage

```
mem_thresholds_v1(x, ...)

## S3 method for class 'csfmt_ensemble_v3'
mem_thresholds_v1(
  x,
  measure,
  min_seasons = 2,
  prefer_seasons = 5,
  i_seasons = 10,
  min_weeks_per_season = 30,
  exclude_seasons = NULL,
  ...
)
```

Arguments

<code>x</code>	The <code>csfmt_ensemble_v3</code> that holds measure.
<code>...</code>	Passed to the method.
<code>measure</code>	The draw matrix to classify, a count or a rate.
<code>min_seasons</code>	The fewest training seasons for a fit.
<code>prefer_seasons</code>	A fit on fewer training seasons is provisional.
<code>i.seasons</code>	The most training seasons in one fit, passed to <code>mem::memmodel()</code> .
<code>min_weeks_per_season</code>	The fewest weeks that a training season needs.
<code>exclude_seasons</code>	Seasons to leave out of every training set, written as <code>cstime::isoyearweek_to_season_c()</code> writes them, such as "2019/2020". Use it for a pandemic year or a season with a data gap. An excluded season still gets thresholds, and a message names the excluded seasons in the data.

Details

The fit uses the median of the draws of each week. A season is fit only on earlier seasons: those with at least `min_weeks_per_season` weeks, minus `exclude_seasons`, and at most the latest `i.seasons`.

A season gets no thresholds, and NA in every draw, in three cases:

- it has fewer than `min_seasons` training seasons,
- fewer than 2 of them hold a non-zero week,
- `mem::memmodel()` fails with its default method and with `i.method = 3`.

A fit on fewer than `prefer_seasons` seasons is provisional, and a message counts those seasons.

The levels, coded 1 to 5, are:

1. `preepidemic`: below `mem_preepidemic`, the MEM epidemic threshold,
2. `low`: below `mem_medium`, the 40% intensity threshold,
3. `medium`: below `mem_high`, the 90% intensity threshold,
4. `high`: below `mem_veryhigh`, the 97.5% intensity threshold,
5. `veryhigh`: at or above `mem_veryhigh`.

It needs the `mem` package.

Value

`x` with a new draw matrix, `<measure>_status`, of level codes with a `levels` attribute. It also adds `mem_preepidemic`, `mem_medium`, `mem_high`, `mem_veryhigh` and `mem_n_seasons`, the number of training seasons, to `$data`. It adds them by reference, so the input ensemble gets them too.

See Also

`vignette("pipeline", package = "csalert")`, stage 6.

Other ensemble operations: [ens_add_rate\(\)](#), [ens_collapse\(\)](#), [short_term_trend\(\)](#), [signal_detection_hlm\(\)](#)

Examples

```
# MEM needs several earlier seasons, so this series spans four:
# 212 weeks with a winter peak in each.
if (requireNamespace("mem", quietly = TRUE)) {
  w <- cstime::dates_by_isoyearweek$isoyearweek
  i <- match("2018-30", w)
  n <- 212L
  iyw <- w[i + 0:(n - 1)]
  set.seed(1)
  lam <- 20 + 60 * exp(-(cstime::isoyearweek_to_seasonweek_n(iyw) - 25)^2 / 40)
  ens <- csfmt_ensemble_v3(
    data.table::data.table(
      isoyearweek = iyw, location_code = "nation", age = "total"
    ),
    id_cols = c("location_code", "age"),
    draws = list(numerator_nowcasted = matrix(rpois(n * 50, lam), nrow = n))
  )

  ens <- mem_thresholds_v1(ens, measure = "numerator_nowcasted")

  # each season is fit on earlier seasons only, so the first seasons get none
  print(unique(ens$data[
    !is.na(mem_high),
    .(
      season = cstime::isoyearweek_to_season_c(isoyearweek),
      mem_preepidemic, mem_medium, mem_high, mem_veryhigh, mem_n_seasons
    )
  ]))

  # every draw of a week in the first two seasons is NA
  status <- ens$draws$numerator_nowcasted_status
  print(c(
    weeks = nrow(status),
    weeks_with_no_threshold = sum(apply(status, 1, function(r) all(is.na(r))))
  ))

  # a week with thresholds gets a share of draws at each level
  r <- ens_collapse(ens, probs = 0.5)
  pcols <- grep("_status_prob_", names(r), value = TRUE)
  print(r[
    isoyearweek %in% c("2021-45", "2022-02"),
    c("isoyearweek", pcols),
    with = FALSE
  ])
}
```

Description

For each as-of date, cuts the triangle back with `nowcast_censor()`, runs the method and collapses the result to quantiles. When the method fails on a date, the function warns and goes on.

Usage

```
nowcast_backtest(
  triangle,
  method,
  as_of_weeks = NULL,
  max_delay_days,
  horizons = 1:2,
  probs = c(0.025, 0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95, 0.975),
  measure = NULL,
  seed = NULL
)
```

Arguments

<code>triangle</code>	A <code>csfmt_reporting_triangle_v3</code> with one series.
<code>method</code>	A function that takes a triangle and returns a <code>csfmt_ensemble_v3</code> , with its other arguments fixed.
<code>as_of_weeks</code>	A Date vector of as-of dates. NULL uses the last day of each reference week after a burn-in of <code>max_delay_days</code> , rounded up to whole weeks. That range covers the weeks with a report at delay day 0 to <code>max_delay_days - 1</code> . With no such week, the result is empty.
<code>max_delay_days</code>	The delay horizon in days. It sets the default as-of dates and the burn-in.
<code>horizons</code>	The horizons to keep, in weeks.
<code>probs</code>	The probabilities of the quantiles to keep.
<code>measure</code>	The draw matrix to score. NULL is <code><value_col>_nowcasted</code> .
<code>seed</code>	A base seed, or NULL. The seed for a date is <code>seed + as.integer(as_of)</code> , so its draws do not depend on the order of the dates.

Details

The horizon of a reference week is the number of whole weeks from its Monday to the as-of date. Horizon 0 is the week of the as-of date.

Value

A long `data.table` with `reference`, `as_of`, a Date, `horizon`, `quantile_level` and `predicted`.

See Also

`vignette("pipeline", package = "csalert")`, stage 2. `nowcast_score_v1()` scores the result against `nowcast_truth()`.

Other nowcast diagnostics: `nowcast_censor()`, `nowcast_truth()`

Examples

```
monday <- as.Date("2023-01-02") + 7 * rep(0:39, each = 3)
set.seed(1)
d <- data.table::data.table(
  isoyearweek_reference = format(monday, "%G-%V"),
  reporting_date = monday + rep(c(3, 10, 17), 40),
  numerator = rpois(120, c(30, 15, 5)),
  indicator_tag = "x", location_code = "nation", age = "total", sex = "total"
)
d <- d[reporting_date <= as.Date("2023-01-02") + 7 * 39 + 6]
tri <- csfmt_reporting_triangle_v3(
  d,
  id_cols = c("indicator_tag", "location_code", "age", "sex")
)

# a method takes a triangle and returns an ensemble
method <- function(x) nowcast_delay_ecdf_v1(x, max_delay_days = 21, n_sim = 200)

# Replay 19 as-of dates, each the Sunday that ends a reference week. Fewer
# dates only save time: with fewer than 3 settled weeks the engine returns
# the observed totals and does not fail. `as_of_weeks = NULL` replays every
# week after the burn-in.
bt <- nowcast_backtest(
  tri, method,
  max_delay_days = 21,
  as_of_weeks = as.Date("2023-01-02") + 7 * (20:38) + 6,
  horizons = 0:1,
  probs = c(0.05, 0.5, 0.95),
  seed = 1
)
head(bt, 6)
```

nowcast_censor

Cut a reporting triangle back to what was known on a past date

Description

Keeps the cells reported on or before `as_of`, and rebuilds the triangle. That is exactly what an engine saw on that date only when the reporting system never corrects or deletes a count.

Usage

```
nowcast_censor(triangle, as_of)
```

Arguments

<code>triangle</code>	The <code>csfmt_reporting_triangle_v3</code> to cut back.
<code>as_of</code>	A Date. Any other class is an error. R compares reporting date <code><= as_of</code> by the type of <code>as_of</code> , so the number 18262 would cut to 2020-01-01 with no warning. A date with no report on or before it is an error.

Value

A `csfmt_reporting_triangle_v3`. Its as-of boundary is the newest reporting date that remains.

See Also

`vignette("pipeline", package = "csalert")`, stage 2.

Other nowcast diagnostics: `nowcast_backtest()`, `nowcast_truth()`

Examples

```
# 40 reference weeks, each reported 3, 10 and 17 days after its Monday
monday <- as.Date("2023-01-02") + 7 * rep(0:39, each = 3)
set.seed(1)
d <- data.table::data.table(
  isoyearweek_reference = format(monday, "%G-%V"),
  reporting_date = monday + rep(c(3, 10, 17), 40),
  numerator = rpois(120, c(30, 15, 5)),
  indicator_tag = "x", location_code = "nation", age = "total", sex = "total"
)
d <- d[reporting_date <= as.Date("2023-01-02") + 7 * 39 + 6]
tri <- csfmt_reporting_triangle_v3(
  d,
  id_cols = c("indicator_tag", "location_code", "age", "sex")
)

# go back to what was known nine weeks earlier
past <- nowcast_censor(tri, as_of = as.Date("2023-01-02") + 7 * 30 + 6)
c(now = attr(tri, "as_of"), then = attr(past, "as_of"))
c(rows_now = nrow(tri), rows_then = nrow(past))
```

`nowcast_delay_ecdf_v1` *Nowcast a reporting triangle from the delay pattern of settled weeks*

Description

Completes each incomplete reference week from a pool of settled reference weeks. Each draw is the count observed so far, times a quantile of the completion ratios that the pool shows at the same delay day.

Usage

```
nowcast_delay_ecdf_v1(x, ...)

## S3 method for class 'csfmt_reporting_triangle_v3'
nowcast_delay_ecdf_v1(
  x,
  max_delay_days,
  n_sim = 1000,
```

```

    denominator_col = NULL,
    delay_window = 26,
    ...
)

```

Arguments

<code>x</code>	The <code>csfmt_reporting_triangle_v3</code> to nowcast.
<code>...</code>	Passed to the method.
<code>max_delay_days</code>	The delay horizon in days, delay day 0 to <code>max_delay_days - 1</code> . The nowcast estimates the count reported within it.
<code>n_sim</code>	The number of draws.
<code>denominator_col</code>	A denominator column to nowcast on the same draws. Its observed total goes to <code>\$data as <denominator_col>_observed</code> .
<code>delay_window</code>	The span, in weeks, of the settled weeks in the pool, so that the pool follows a reporting pattern that changes. NULL uses every settled week.

Details

Take a reference week of age d days, so delay days 0 to d are observed. The pool is the settled reference weeks inside `delay_window`. For pool week s , T_s is its settled total and O_s is its count by delay day d . The draws are `observed_so_far * quantile(T_s / O_s)`, in random order. The quantiles sit at n_{sim} evenly spaced probabilities from 0 to 1, or at 0.5 when n_{sim} is 1.

A pool week with $O_s = 0$ has no finite ratio, so it cannot enter the pool. With fewer than 3 pool weeks left, the reference week keeps its observed count. The interval therefore does not describe a week whose reporting has not started, and a reference week with no observed count stays at 0.

The interval is empirical: the 5% and 95% draw quantiles are the band. Nothing parametric is added, because the spread of the pool ratios already carries the estimation error and the reporting noise. A nowcast never falls below the observed count.

A pool week is settled once it is `max_delay_days - 1` days old. The nowcast estimates the count reported within `max_delay_days` days, so T_s holds no report at delay `max_delay_days` or later. A late report is still in `original`, and in the draws of a settled week. A settled week whose only reports are late has no delay information. Such a week, as in a bulk load of old weeks, does not enter the pool.

The engine also forms $p(d)$, the pooled share of the counts of a week that arrive by delay day d . $p(d)$ cancels out of every draw. It only decides whether delay day d is completed: when $p(d)$ is 0, the reference week keeps its observed count. So the draws are not built from `observed_so_far / p(d)`, the closed-form maximum likelihood estimate under $n[\text{ref}, d] \sim \text{Poisson}(\text{lambda}[\text{ref}] * p[d])$.

There is no weekday term. Every reference week starts on a Monday, so delay day d is always the same weekday. The pool ratios at delay day d therefore absorb the weekly pattern.

Whether the intervals are calibrated for your series is an empirical question. Replay it with `nowcast_backtest()`, and score the replay with `nowcast_score_v1()`. Like every nowcast engine, this one takes a reporting triangle and returns a `csfmt_ensemble_v3`.

Value

A `csfmt_ensemble_v3` with one row per reference week and a draw matrix `<value_col>_nowcasted` with `n_sim` columns. A settled week has its observed total in every draw. `$data` holds original, the observed total. The observed total includes every report at delay `max_delay_days` or later.

See Also

`vignette("pipeline", package = "csalert")`, which runs this engine on a synthetic triangle and scores it.

Other nowcast engines: [nowcast_passthrough_to_ensemble_v1\(\)](#)

Examples

```
# 40 reference weeks, each reported 3, 10 and 17 days after its Monday. The
# data stops at one as-of date, so the newest weeks are still incomplete.
cal <- cstime::dates_by_isoyearweek
i <- match("2023-01", cal$isoyearweek)
set.seed(1)
monday <- as.Date(cal$mon[i + rep(0:39, each = 3)])
d <- data.table::data.table(
  isoyearweek_reference = cal$isoyearweek[i + rep(0:39, each = 3)],
  reporting_date = monday + rep(c(3, 10, 17), 40),
  numerator = rpois(120, c(30, 15, 5)),
  indicator_tag = "x", location_code = "nation", age = "total", sex = "total"
)
d <- d[reporting_date <= as.Date(cal$mon[i + 39]) + 6]
tri <- csfmt_reporting_triangle_v3(
  d,
  id_cols = c("indicator_tag", "location_code", "age", "sex")
)

set.seed(2)
ens <- nowcast_delay_ecdf_v1(tri, max_delay_days = 21, n_sim = 200)
ens

# a settled week sits on its observed total, and a newest week is completed
# with an interval
r <- ens_collapse(ens, probs = c(0.05, 0.5, 0.95))
tail(r[, .(
  isoyearweek, original,
  lo = numerator_nowcasted_q05x0,
  med = numerator_nowcasted_q50x0,
  hi = numerator_nowcasted_q95x0
)], 4)
```

nowcast_passthrough_to_ensemble_v1

Build an ensemble from a reporting triangle with no nowcast

Description

Returns the counts reported so far as an ensemble with one draw. Use it for an indicator that SHOULD NOT be nowcast, and the rest of the pipeline runs unchanged.

Usage

```
nowcast_passthrough_to_ensemble_v1(x, max_delay_days, denominator_col = NULL)
```

Arguments

x The csfmt_reporting_triangle_v3 to pass through.

max_delay_days The delay horizon in days. The totals do not depend on it, because a report at delay max_delay_days or later is in the total too.

denominator_col A denominator column to pass through in the same way. Its total also goes to \$data as <denominator_col>_observed.

Details

The draw matrix has the name <measure>_nowcasted, as from [nowcast_delay_ecdf_v1\(\)](#), and holds the observed total. So every collapsed quantile equals the observed total, also for the newest weeks.

Value

A csfmt_ensemble_v3 with one draw. \$data holds original, the observed total, which includes the reports at delay max_delay_days or later.

See Also

`vignette("pipeline", package = "csalert")`, stage 2, which scores it against [nowcast_delay_ecdf_v1\(\)](#).
Other nowcast engines: [nowcast_delay_ecdf_v1\(\)](#)

Examples

```
# 40 reference weeks, each reported 3, 10 and 17 days after its Monday. The
# data stops at one as-of date, so the newest weeks are still incomplete.
monday <- as.Date("2023-01-02") + 7 * rep(0:39, each = 3)
set.seed(1)
d <- data.table::data.table(
  isoyearweek_reference = format(monday, "%G-%V"),
  reporting_date = monday + rep(c(3, 10, 17), 40),
  numerator = rpois(120, c(30, 15, 5)),
  indicator_tag = "x", location_code = "nation", age = "total", sex = "total"
)
d <- d[reporting_date <= as.Date("2023-01-02") + 7 * 39 + 6]
tri <- csfmt_reporting_triangle_v3(
  d,
  id_cols = c("indicator_tag", "location_code", "age", "sex")
)
```

```

ens <- nowcast_passthrough_to_ensemble_v1(tri, max_delay_days = 21)
ens

# one draw, so every quantile equals the observed total, also for the
# newest weeks that are still incomplete
r <- ens_collapse(ens, probs = c(0.05, 0.5, 0.95))
tail(r[, .(
  isoyearweek, original,
  lo = numerator_nowcasted_q05x0,
  med = numerator_nowcasted_q50x0,
  hi = numerator_nowcasted_q95x0
)], 3)

```

nowcast_score_v1	<i>Score replayed nowcast quantiles against the settled truth</i>
------------------	---

Description

Scores a backtest from [nowcast_backtest\(\)](#) against the truth from [nowcast_truth\(\)](#). It reports the interval coverage, the revision of the median, the weighted interval score and a 95% interval summary.

Usage

```
nowcast_score_v1(backtest, truth, by = "horizon", thresholds = c(0.25, 0.5))
```

Arguments

backtest	A data.table from nowcast_backtest() , with the columns reference, quantile_level and predicted, and optionally as_of and horizon.
truth	A data.table from nowcast_truth() , with the columns reference and truth.
by	The columns to group the scores by.
thresholds	The absolute revisions for the p_gt_<t> columns.

Details

A forecast unit is one reference week at one as-of date and horizon. The function scores only the units that have a finite truth and the 0.05, 0.25, 0.5, 0.75 and 0.95 quantiles. The revision of a unit is $(\text{median} - \text{truth}) / \text{truth}$, over the units with a truth above 0. Every score is a measurement on the replayed weeks, not a property of a method.

The weighted interval score (WIS) follows Bracher et al. (2021). For a unit with truth y and median m :

$$\begin{aligned} \text{WIS} &= (0.5 * |y - m| + \sum_k (\alpha_k / 2) * \text{IS}_k) / (K + 0.5) \\ \text{IS}_k &= (u - 1) + (2 / \alpha_k) * (1 - y) * 1\{y < l\} + (2 / \alpha_k) * (y - u) * 1\{y > u\} \end{aligned}$$

The sum is over the K central intervals from the quantile pairs at levels p and $1 - p$, with $p < 0.5$. Interval k is from l to u , and $\alpha_k = 2 * p$.

Value

A data.table with one row per group. It has the by columns and these columns:

- n: the number of scored units,
- coverage_50, coverage_90: the share of truths from the 0.25 to the 0.75 quantile, and from the 0.05 to the 0.95 quantile,
- median_signed, median_abs, q05, q95: the median revision, the median absolute revision, and the 5% and 95% quantiles of the revision,
- p_gt_<t>: the share of absolute revisions above each threshold, such as p_gt_25 for 0.25,
- wis: the mean WIS over the units,
- wis_log: the mean WIS after log1p() of each quantile and of the truth,
- coverage_95: the share of truths from the 0.025 to the 0.975 quantile,
- width_95_rel_median: the median over the units of $(q_{0.975} - q_{0.025}) / \max(\text{truth}, 1)$.

coverage_95 and width_95_rel_median are NA for a group where a unit has no 0.025 or no 0.975 quantile.

References

Bracher J, Ray EL, Gneiting T, Reich NG (2021). Evaluating epidemic forecasts in an interval format. PLOS Computational Biology 17(2): e1008618. doi:10.1371/journal.pcbi.1008618

See Also

vignette("pipeline", package = "csalert"), stage 2, which explains how to read each column.

Examples

```
truth <- data.table::data.table(reference = "2024-01", truth = 100)
backtest <- data.table::data.table(
  reference = "2024-01", horizon = 0L,
  quantile_level = c(0.025, 0.05, 0.25, 0.5, 0.75, 0.95, 0.975),
  predicted = c(60, 70, 85, 95, 105, 120, 130))
nowcast_score_v1(backtest, truth)
```

nowcast_truth

The settled total of each reference week

Description

Sums the counts of each settled reference week within the delay horizon, the truth that a backtest scores a nowcast against. A report at delay max_delay_days or later is not in the total, because a nowcast estimates the count within the horizon.

Usage

```
nowcast_truth(triangle, max_delay_days)
```

Arguments

triangle A csfmt_reporting_triangle_v3 with one series.

max_delay_days The delay horizon in days. 35 is the 35 days from the reference Monday. The 5 weekly delay columns of the older format had the same span.

Details

A week is settled when its Monday is at least `max_delay_days - 1` days before the as-of date. With `max_delay_days = 21`, the newest settled week starts 20 days before the as-of date, not 21. A report at delay `max_delay_days` or later does not change the total of a settled week.

Value

A data.table with reference, the ISO week, and truth, one row per settled week.

See Also

`vignette("pipeline", package = "csalert")`, stage 2.

Other nowcast diagnostics: [nowcast_backtest\(\)](#), [nowcast_censor\(\)](#)

Examples

```
monday <- as.Date("2023-01-02") + 7 * rep(0:39, each = 3)
set.seed(1)
d <- data.table::data.table(
  isoyearweek_reference = format(monday, "%G-%V"),
  reporting_date = monday + rep(c(3, 10, 17), 40),
  numerator = rpois(120, c(30, 15, 5)),
  indicator_tag = "x", location_code = "nation", age = "total", sex = "total"
)
d <- d[reporting_date <= as.Date("2023-01-02") + 7 * 39 + 6]
tri <- csfmt_reporting_triangle_v3(
  d,
  id_cols = c("indicator_tag", "location_code", "age", "sex")
)

truth <- nowcast_truth(tri, max_delay_days = 21)

# the newest weeks are not settled yet, so they have no row
tail(truth, 3)
c(reference_weeks = 40L, settled = nrow(truth))
```

```
print.csfmt_ensemble_v3
      Print a csfmt_ensemble_v3
```

Description

Prints one line: the number of rows and of series, and the names of the draw matrices.

Usage

```
## S3 method for class 'csfmt_ensemble_v3'
print(x, ...)
```

Arguments

x	The csfmt_ensemble_v3 to print.
...	Not used, but the print() generic has it.

Value

x, invisibly.

See Also

vignette("pipeline", package = "csalert"), stage 1.

Other ensemble format functions: [csfmt_ensemble_v3\(\)](#), [set_time_series_id\(\)](#), [validate_ensemble\(\)](#)

q_label	<i>Write a probability as a quantile label</i>
---------	--

Description

Writes q, two integer-percent digits, x and one decimal-percent digit, so 0.025 becomes "q02x5".

Usage

```
q_label(p)
```

Arguments

p	A numeric vector of probabilities.
---	------------------------------------

Details

The label loses information in two ways. A finer probability is rounded: 0.0125 becomes "q01x2". And p = 1 gives "q100x0", which [q_value\(\)](#) cannot read. Keep p on the 0.001 grid and below 1.

Value

A character vector of labels, with NA for NA.

See Also

`vignette("pipeline", package = "csalert")`, whose naming-grammar section shows both limits.

Other naming grammar functions: `csfmt_interpret()`, `csfmt_parse()`, `csfmt_var()`, `q_value()`

Examples

```
q_label(c(0.025, 0.5, 0.975))

# limit 1: a probability finer than one decimal percent is rounded
q_value(q_label(0.0125))

# limit 2: p = 1 gives a three-digit label, and q_value() returns NA for it
q_label(1)
q_value(q_label(1))
```

q_value

Read the probability in a quantile label

Description

Reads a label that `q_label()` wrote: q, two digits, x and one digit. Any other string gives NA.

Usage

```
q_value(label)
```

Arguments

label A character vector of labels, for example "q02x5".

Details

`q_value(q_label(p))` returns p only for a p on the 0.001 grid and below 1. Every probability that the package uses is on that grid.

Value

A numeric vector of probabilities.

See Also

`vignette("pipeline", package = "csalert")`, whose naming-grammar section shows the round trip.

Other naming grammar functions: `csfmt_interpret()`, `csfmt_parse()`, `csfmt_var()`, `q_label()`

Examples

```
q_value(c("q02x5", "q50x0", "q97x5"))

# a string that is not a label gives NA, and so does the three-digit q100x0
q_value(c("q100x0", "not_a_label"))
```

```
qc_surveillance_data_v1
```

Check that a surveillance feed has data and is up to date

Description

Checks the input of one indicator for too few rows, a missing reference column, and a newest period older than `expect_latest`. It returns a verdict, and the caller decides what to do.

Usage

```
qc_surveillance_data_v1(
  d,
  reference_col = "isoyearweek_reference",
  expect_latest = NULL,
  min_rows = 1L
)
```

Arguments

<code>d</code>	A <code>data.table</code> with the data of one indicator.
<code>reference_col</code>	The reference-period column.
<code>expect_latest</code>	The newest period that the caller expects, or <code>NULL</code> to skip the check. The check uses <code><</code> , which orders zero-padded "YYYY-WW" strings correctly.
<code>min_rows</code>	The fewest rows that pass.

Details

The checks run in that order and stop at the first failure, so `reasons` has at most one entry.

Value

A list with `ok`, `TRUE` when every check passes, and `reasons`, a character vector that is empty when `ok` is `TRUE`.

See Also

`vignette("pipeline", package = "csalert")`, section 9.
 Other quality control functions: [compare_results\(\)](#), [qc_week_over_week_v1\(\)](#)

Examples

```
d <- data.table::data.table(
  isoyearweek_reference = c("2023-01", "2023-02"),
  numerator = c(10, 12)
)

qc_surveillance_data_v1(d)

# the feed has not been updated as far as the caller expected
qc_surveillance_data_v1(d, expect_latest = "2023-05")

# nothing arrived at all
qc_surveillance_data_v1(d[0])
```

qc_week_over_week_v1 *Compare this week's run with last week's run*

Description

Splits the output of [compare_results\(\)](#) at the nowcast horizon. `$integrity` lists the settled weeks whose median changed, and `$signal` lists the status changes in the newer weeks.

Usage

```
qc_week_over_week_v1(
  current,
  previous,
  max_delay_weeks,
  tol = 1e-06,
  status_roles = c("status", "hlmstatus"),
  model_pattern = "_trend_(gr|beta1)_"
)
```

Arguments

current, previous	The output of ens_collapse() for this run and for the previous run. Their identity columns MUST have the names that compare_results() needs.
max_delay_weeks	The nowcast horizon in ISO weeks. The engines take <code>max_delay_days</code> , in days. The names differ so that one name never means two units.
tol	The largest change in a settled median that counts as no change.
status_roles	The roles that hold an ordinal status. <code>\$signal</code> holds only these roles, and <code>\$integrity</code> leaves them out. The default holds "status" from mem_thresholds_v1() and "hlmstatus" from signal_detection_hlm() .

model_pattern A regular expression for the names of model-output columns that `$integrity` leaves out. The default matches the trend columns that `short_term_trend()` writes for an ensemble, `<measure>_trend_gr_q50x0` and `<measure>_trend_beta1_q50x0`. A trend estimate has Monte-Carlo noise, so its change in a settled week is not a data revision.

Details

A week is settled when it is at least `max_delay_weeks` ISO weeks older than the newest week of previous. A row in `$integrity` means that a published number for a settled week changed, so ideally that table is empty. It compares finite values only, and leaves out the columns that `model_pattern` matches. `$signal` counts a new week as a change. Both tables use the median, `q = 0.5`.

Value

A list of two `data.tables`:

- `integrity`, with `indicator_tag`, `isoyearweek`, `column`, `prv`, `cur` and `abs_diff`,
- `signal`, with `indicator_tag`, `isoyearweek`, `column`, `from`, `to` and `change`, which is "new" or "changed".

A table with no rows keeps `prv` and `cur`, and has no added column. `indicator_tag` is there only when the input has it.

See Also

`vignette("pipeline", package = "csalert")`, section 9.

Other quality control functions: `compare_results()`, `qc_surveillance_data_v1()`

Examples

```
# 40 reference weeks, each reported 3, 10 and 17 days after its Monday
mondays <- as.Date("2023-01-02") + 7 * (0:39)
set.seed(1)
d <- data.table::data.table(
  isoyearweek_reference = format(rep(mondays, each = 3), "%G-%V"),
  reporting_date = rep(mondays, each = 3) + rep(c(3, 10, 17), 40),
  numerator = rpois(120, c(30, 15, 5)),
  indicator_tag = "x", location_code = "nation", age = "total", sex = "total"
)
d <- d[reporting_date <= mondays[40] + 6]

id <- c("indicator_tag", "location_code", "age", "sex")

# set the seed inside `run()`, so the two runs differ only by the data
run <- function(x) {
  set.seed(2)
  ens_collapse(nowcast_delay_ecdf_v1(
    csfmt_reporting_triangle_v3(x, id_cols = id),
    max_delay_days = 21, n_sim = 200
  )
}
```

```

    ))
  }

  cur <- run(d)
  d_prv <- data.table::copy(d)
  d_prv[
    isoyearweek_reference == format(mondays[11], "%G-%V"),
    numerator := numerator - 5
  ]
  prv <- run(d_prv)

  # the engine horizon is 21 days, and this horizon is 3 weeks
  qc <- qc_week_over_week_v1(cur, prv, max_delay_weeks = 3)

  # the corrected week is settled, so its changed median is listed here
  qc$integrity

  # empty: these runs hold no status column, because neither
  # mem_thresholds_v1() nor signal_detection_hlm() ran
  qc$signal

```

reporting_completion_trend_v1

Reporting speed by year and by recent month, in one table

Description

Stacks [reporting_completion_v1\(\)](#) by year, and by month for the latest `n_months` months of each series, so one table shows reporting speed up or slow down.

Usage

```
reporting_completion_trend_v1(triangle, max_delay_days, n_months = 12L)
```

Arguments

<code>triangle</code>	The <code>csfmt_reporting_triangle_v3</code> to measure.
<code>max_delay_days</code>	The delay horizon in days.
<code>n_months</code>	The number of latest months to keep for each series.

Value

A `data.table` with the columns of [reporting_completion_v1\(\)](#) and `scope`, "year" or "month", with the year rows first. It is empty when no series has enough settled weeks.

See Also

`vignette("pipeline", package = "csalert")`, stage 3.

Other reporting completion functions: [reporting_completion_v1\(\)](#)

Examples

```
monday <- as.Date("2023-01-02") + 7 * rep(0:39, each = 3)
d <- data.table::data.table(
  isoyearweek_reference = format(monday, "%G-%V"),
  reporting_date = monday + rep(c(3, 10, 17), 40),
  numerator = 10, indicator = "x", location = "n", age = "total", sex = "total")
tri <- csfmt_reporting_triangle_v3(d, id_cols = c("indicator", "location", "age", "sex"))
reporting_completion_trend_v1(tri, max_delay_days = 21, n_months = 6)[,
  .(scope, period, n_settled, mean_delay)]
```

reporting_completion_v1

Measure how fast the counts of a reporting triangle arrive

Description

Pools the settled weeks of each series, and returns the share of cases reported by each delay day, and the mean delay. Use it to choose `max_delay_days`, and with `period` to see reporting speed up or slow down.

Usage

```
reporting_completion_v1(
  triangle,
  max_delay_days,
  delay_window = NULL,
  period = c("all", "year", "month")
)
```

Arguments

<code>triangle</code>	The <code>csfmt_reporting_triangle_v3</code> to measure.
<code>max_delay_days</code>	The delay horizon in days, delay day 0 to <code>max_delay_days - 1</code> .
<code>delay_window</code>	Use only the settled weeks of about this many recent weeks. It counts weeks, not days. <code>NULL</code> uses every settled week.
<code>period</code>	"all" gives one pooled row. "year" and "month" give one row per calendar year or month of the Thursday of each week, so the year is the ISO year.

Details

A week is settled when its Monday is at least `max_delay_days - 1` days before the as-of date. In each period, the function drops a week with a total of 0. A period with fewer than 3 weeks left gets no row, and no warning.

Value

A data.table with one row per series and period: the identity columns, period, n_settled, mean_delay, complete_by_md, and pct_delay0 to pct_delay<max_delay_days - 1>.

- n_settled: the settled weeks with a total above 0.
- mean_delay: the mean delay in days over the pooled cases. A report at delay max_delay_days or later counts at max_delay_days - 1, so it is a lower bound.
- pct_delayD: the percentage of pooled cases reported by the end of day D from the reference Monday, with no interpolation. pct_delay0 is the Monday itself.

complete_by_md is always 1

complete_by_md is the last cumulative share of the total. So it is 1 for every series and period, and pct_delay<max_delay_days - 1> is 100. It does NOT show whether reporting continues after max_delay_days.

This function adds the reports past the horizon to the last column, so $100 - \text{pct_delay} < \text{max_delay_days} - 2 >$ is the share reported on the last day or later. To see the tail, run it again with a larger max_delay_days, and compare mean_delay and the pct_delayD curve.

See Also

vignette("pipeline", package = "csalert"), stage 3, which shows how to read each column.

Other reporting completion functions: [reporting_completion_trend_v1\(\)](#)

Examples

```
monday <- as.Date("2023-01-02") + 7 * rep(0:39, each = 3)
set.seed(1)
d <- data.table::data.table(
  isoyearweek_reference = format(monday, "%G-%V"),
  reporting_date = monday + rep(c(3, 10, 17), 40),
  numerator = rpois(120, c(30, 15, 5)),
  indicator_tag = "x", location_code = "nation", age = "total", sex = "total"
)
d <- d[reporting_date <= as.Date("2023-01-02") + 7 * 39 + 6]
tri <- csfmt_reporting_triangle_v3(
  d,
  id_cols = c("indicator_tag", "location_code", "age", "sex")
)

# One pooled curve over 21 delay days. This series reports on days 3, 10 and
# 17 after the reference Monday, so the curve steps on those three days.
rc <- reporting_completion_v1(tri, max_delay_days = 21)
rc[, .(n_settled, mean_delay, pct_delay3, pct_delay10, pct_delay17)]

# there is one pct_delay column per delay day
sum(grepl("^pct_delay", names(rc)))

# sliced by month, to show a change in reporting speed
head(
```

```

reporting_completion_v1(tri, max_delay_days = 21, period = "month")[,
  .(period, n_settled, mean_delay, pct_delay10)
],
3
)

```

reporting_triangle_matrix

Turn a reporting triangle into one count matrix per series

Description

Returns, for each series, a matrix of counts with one row per reference week and one column per delay day. Every nowcast engine calls it first.

Usage

```

reporting_triangle_matrix(
  triangle,
  max_delay_days,
  value_col = attr(triangle, "value_col")
)

```

Arguments

triangle	The csfmt_reporting_triangle_v3 to turn into matrices.
max_delay_days	The number of delay columns, in days. With 35, the columns are delay days 0 to 34, and a report at delay 35 or 400 counts in late.
value_col	The count column. The default is the value_col of the triangle. Give a denominator column to get its matrix.

Details

The rows run over every ISO week from the first to the last reference week. A week with no report is a row of zeros. A report at delay max_delay_days or later goes to late, not to mat. A week whose only reports are late still has a row, and that row is zeros. A report before its reference Monday has a negative delay, and is dropped.

Value

A list named by time_series_id. Each element holds reference, the ISO weeks of the rows, mat, the matrix, and late. late is a numeric vector with one value per row: the count reported at delay max_delay_days or later. The observed total of a week is rowSums(mat) + late. A cell sums its counts with na.rm = TRUE, so a cell with only NA counts is 0.

See Also

`vignette("pipeline", package = "csalert")`, stage 3, which reads the delay distribution from these counts.

Other reporting triangle functions: `csfmt_reporting_triangle_v3()`, `isoyearweek_week_start()`

Examples

```
cal <- cstime::dates_by_isoyearweek
i <- match("2023-01", cal$isoyearweek)
set.seed(1)
monday <- as.Date(cal$mon[i + rep(0:39, each = 3)])
d <- data.table::data.table(
  isoyearweek_reference = cal$isoyearweek[i + rep(0:39, each = 3)],
  reporting_date = monday + rep(c(3, 10, 17), 40),
  numerator = rpois(120, c(30, 15, 5)),
  indicator_tag = "x", location_code = "nation", age = "total", sex = "total"
)
d <- d[reporting_date <= as.Date(cal$mon[i + 39]) + 6]
tri <- csfmt_reporting_triangle_v3(
  d,
  id_cols = c("indicator_tag", "location_code", "age", "sex")
)

m <- reporting_triangle_matrix(tri, max_delay_days = 21)
names(m)

# rows are reference weeks, columns are delay days 0 to 20
dim(m[[1]]$mat)
head(m[[1]]$reference, 3)

# this series reports on days 3, 10 and 17 after the reference Monday
m[[1]]$mat[1:3, c("3", "10", "17")]

# the newest weeks are only partly reported: the later delays are still zero
tail(m[[1]]$mat[, c("3", "10", "17")], 3)
```

`rolling_slope_matrix` *Rolling regression slope down every column of a matrix*

Description

Fits $y \sim 1 + t$, with t from 1 to width, to each window of width rows in every column. It is the kernel of the ensemble method of `short_term_trend()`.

Usage

```
rolling_slope_matrix(
  Y,
  width,
```

```

family = c("identity", "quasipoisson", "binomial"),
prior_weights = NULL,
tol = 1e-10,
maxit = 25L
)

```

Arguments

<code>Y</code>	A numeric matrix, rows = weeks in time order, columns = draws.
<code>width</code>	The window width in rows, at least 2.
<code>family</code>	"identity" is ordinary least squares. "quasipoisson" is a log link and "binomial" a logit link, and for those <code>beta1</code> is on the link scale.
<code>prior_weights</code>	The binomial denominators, a matrix the shape of <code>Y</code> or with one column. Only <code>family = "binomial"</code> takes it, and it needs it.
<code>tol</code>	The iteration stops when no coefficient moves more than <code>tol</code> .
<code>maxit</code>	The most iterations. A window that still moves more than <code>tol</code> did not converge.

Details

Row `i` holds the fit of the window that ends on row `i`, so the first `width - 1` rows are NA. The identity family has a closed form. The other two families run iteratively reweighted least squares (IRLS) on every window at once. A window with complete or quasi-complete separation cannot converge: its `beta0`, `beta1` and `se` are NA, and one warning gives the count.

Value

A list of four matrices, each the shape of `Y`: `beta0`, `beta1`, `se` and `converged`. `se` is the standard error of `beta1`, a Wald standard error under IRLS. The logical `converged` is NA on the first `width - 1` rows.

One deliberate difference from glm

The function agrees with `stats::glm()` except on an all-zero window. There it returns NA, and `stats::glm()` returns a slope of 0. `stats::glm()` stops on the relative change in deviance, which is 0 from the first step. This function stops on the coefficient step, which never settles, because the intercept goes to $-\infty$. An all-zero window identifies no slope, and NA says so.

See Also

`vignette("pipeline", package = "csalert")`, stage 5.

Other short-term trend functions: [short_term_trend\(\)](#), [short_term_trend_sts_v1\(\)](#)

Examples

```

# 10 weeks x 4 draws, all rising at a true slope of 2 per week
set.seed(1)
Y <- matrix(rep(1:10, 4) * 2 + rnorm(40), nrow = 10)

```

```

rs <- rolling_slope_matrix(Y, width = 4)

# the first three rows have no complete window, so they are NA
head(rs$beta1, 3)

# the later rows find the slope, one estimate per draw
round(rs$beta1[8:10, ], 2)

# `se` is the OLS standard error of the slope
round(rs$se[10, ], 2)

# a log link: counts that rise by 0.2 per week on the log scale
N <- matrix(rpois(40, rep(exp(1 + 0.2 * 1:10), 4)), nrow = 10)
round(rolling_slope_matrix(N, width = 4, family = "quasipoisson")$beta1[10, ], 2)

```

set_time_series_id	<i>Add a content-hash series id to a data.table</i>
--------------------	---

Description

Adds `time_series_label`, the identity values joined by `sep`, and `time_series_id`, the xxhash64 digest of that label. The id depends only on the identity values, so a series gets the same id in every table.

Usage

```
set_time_series_id(d, id_cols, sep = "\037")
```

Arguments

<code>d</code>	A <code>data.table</code> . The function changes it by reference.
<code>id_cols</code>	The identity columns that define one series.
<code>sep</code>	The separator in the label. The default is the ASCII unit separator, <code>"\037"</code> .

Details

`csfmt_ensemble_v3()` and `csfmt_reporting_triangle_v3()` call this function for you.

Value

`d`, invisibly.

See Also

`vignette("pipeline", package = "csalert")`, which shows the ensemble format.

Other ensemble format functions: `csfmt_ensemble_v3()`, `print.csfmt_ensemble_v3()`, `validate_ensemble()`

Examples

```
d <- data.table::data.table(
  location_code = c("nation", "nation", "county03"),
  age = "total",
  isoyearweek = c("2023-01", "2023-02", "2023-01"),
  numerator = c(10, 12, 4)
)
set_time_series_id(d, id_cols = c("location_code", "age"))

# the two nation rows share one id, and the county row has its own
d[]
```

short_term_trend	<i>Estimate the short-term trend of a series</i>
------------------	--

Description

The `csfmt_ensemble_v3` method fits a line through a rolling window of weeks in every draw. The deprecated `csfmt_rts_data_v1` method fits a rolling quasi-Poisson regression to a table, after Benedetti (2019) [doi:10.5588/pha.19.0002](https://doi.org/10.5588/pha.19.0002).

Usage

```
short_term_trend(x, ...)

## S3 method for class 'csfmt_rts_data_v1'
short_term_trend(
  x,
  numerator,
  denominator = NULL,
  prX = 100,
  trend_isoyearweeks = 6,
  remove_last_isoyearweeks = 0,
  forecast_isoyearweeks = trend_isoyearweeks,
  numerator_naming_prefix = "from_numerator",
  denominator_naming_prefix = "from_denominator",
  statistics_naming_prefix = "universal",
  remove_training_data = FALSE,
  include_decreasing = FALSE,
  alpha = 0.05,
  ...
)

## S3 method for class 'csfmt_rts_data_v3'
short_term_trend(x, ...)

## S3 method for class 'csfmt_ensemble_v3'
```

```

short_term_trend(
  x,
  measure,
  trend_isoyearweeks = 3,
  n_sim = 1000L,
  family = c("identity", "quasipoisson", "binomial"),
  denominator = NULL,
  error_reference = c("auto", "normal", "t"),
  ...
)

```

Arguments

<code>x</code>	A <code>csfmt_ensemble_v3</code> , or a deprecated <code>csfmt_rts_data_v1</code> .
<code>...</code>	Passed to the method.
<code>numerator</code>	The count column.
<code>denominator</code>	For <code>csfmt_rts_data_v1</code> , an optional denominator column. For <code>csfmt_ensemble_v3</code> , the draw matrix of the binomial denominator, which <code>family = "binomial"</code> needs and the other families reject. <code>measure</code> MUST then be a proportion.
<code>prX</code>	The scale of the rate numerator / denominator, such as 100. It MAY hold more than one scale.
<code>trend_isoyearweeks</code>	The width of the rolling window, in weeks. The ensemble method needs at least 3, or 2 with <code>family = "binomial"</code> . The <code>v1</code> method needs at least 2.
<code>remove_last_isoyearweeks</code>	The number of latest weeks to leave out of the fit. They get the status "forecast".
<code>forecast_isoyearweeks</code>	The number of weeks to forecast after the data.
<code>numerator_naming_prefix</code>	The start of the new column names. "from_numerator" is the numerator name without its last <code><word></code> , and "generic" is "numerator". Any other string is used as written.
<code>denominator_naming_prefix</code>	The same for the denominator.
<code>statistics_naming_prefix</code>	"universal" names the columns <code><prefix>_trend0_<days>_status</code> and <code><prefix>_doublingdays0_<days></code> with <code><days> = 7 * trend_isoyearweeks - 1</code> . "from_numerator_and_prX" adds the numerator suffix, or <code>_pr<prX></code> with a denominator.
<code>remove_training_data</code>	If TRUE, drop the first <code>trend_isoyearweeks - 1</code> rows, which have no window.
<code>include_decreasing</code>	If FALSE, the levels are "training", "forecast", "notincreasing" and "increasing". If TRUE, they are "training", "forecast", "decreasing", "null" and "increasing".
<code>alpha</code>	The significance level of the test on the slope.
<code>measure</code>	The draw matrix to fit the trend to.

n_sim	The number of draws to use when the ensemble has one draw, as the output of <code>nowcast_passthrough_to_ensemble_v1()</code> has.
family	"identity" is ordinary least squares, with the growth rate $100 * \text{beta1} / Y$. "quasipoisson" is a log link, and "binomial" a logit link that needs denominator. For those two, beta1 is on the link scale. The growth rate is then $100 * (\exp(\text{beta1}) - 1)$, a percent change per week, in the odds under "binomial". A zero count needs no offset.
error_reference	The distribution of the slope error. "auto" gives "identity" and "quasipoisson" a <code>t</code> on <code>trend_isoyearweeks - 2</code> degrees of freedom, the residual degrees of freedom of their dispersion. It gives "binomial" a normal. At width 3 that <code>t</code> is a Cauchy, with heavy tails. Use "normal" to match <code>stats::confint()</code> , which profiles the deviance against an asymptotic chi-squared. The quasi-Poisson dispersion is floored at 1, so its standard error is never smaller than the Poisson one.

Value

The `v1` method returns `x` with the forecast weeks added. The new columns are the status, the doubling time in days, the forecast with its 2.5% and 97.5% limits, and a logical forecast marker. With a denominator, it also adds the forecast denominator and rates.

The `csfmt_rts_data_v3` method always stops with an error.

The ensemble method returns `x` with the draw matrices `<measure>_trend_beta1`, the slope per week, and `<measure>_trend_gr`, the growth rate. It adds `<measure>_trend_increasing_pr`, the share of draws with a positive slope, to `$data` by reference, so the input ensemble gets it too.

The deprecated `csfmt_rts_data_v1` method

`short_term_trend.csfmt_rts_data_v1()` is **deprecated**. It still works and gives no warning. New work SHOULD run the ensemble method before `ens_collapse()`:

```
ens <- nowcast_delay_ecdf_v1(triangle, max_delay_days = 35)
ens <- short_term_trend(ens, measure = "numerator_nowcasted")
out <- ens_collapse(ens, heal = TRUE)
```

The replacement is not a drop-in. The `v1` method takes numerator, denominator, `prX` and the `*_naming_prefix` arguments. The ensemble method takes one draw matrix, `measure`. The `v1` method returns a status and a doubling time from a quasi-Poisson log-link fit. The ensemble method returns a slope, a growth rate and the share of draws with a positive slope. So the call changes, and the numbers do not match.

The `v1` method needs `granularity_time == "isoyearweek"`. A window is increasing when its slope is positive with a p-value at most `alpha`.

Why the `csfmt_rts_data_v3` method is an error

A `csfmt_rts_data_v3` is the collapsed output of the pipeline. It holds quantiles, not draws, so no per-draw trend can come from it. Run the trend before the collapse:

```
ens <- short_term_trend(ens, measure = "numerator_nowcasted") # before
out <- ens_collapse(ens, heal = TRUE) # then collapse
```

The csfmt_ensemble_v3 method

In every draw, the method fits a line to the window of `trend_isoyearweeks` weeks that ends on each week, with `rolling_slope_matrix()`. The first `trend_isoyearweeks - 1` weeks of each series get NA.

It then adds the sampling error of the slope to each draw, $\beta_1 + se * e$, with e from the distribution that `error_reference` names. So a settled week, with the same count in every draw, still gets an interval. The method returns no increasing label: choose your own cut-off on the share of draws with a positive slope.

See Also

`vignette("pipeline", package = "csalert")`, stage 5, and `vignette("csalert", package = "csalert")` on which method to use.

Other ensemble operations: `ens_add_rate()`, `ens_collapse()`, `mem_thresholds_v1()`, `signal_detection_hlm()`

Other short-term trend functions: `rolling_slope_matrix()`, `short_term_trend_sts_v1()`

Examples

```
# the ensemble method: 10 weeks x 200 draws of a rising count
set.seed(1)
ens <- csfmt_ensemble_v3(
  data.table::data.table(
    isoyearweek = sprintf("2023-%02d", 1:10),
    location_code = "nation",
    age = "total"
  ),
  id_cols = c("location_code", "age"),
  draws = list(numerator_nowcasted = matrix(rpois(2000, 20 + 3 * 1:10), 10))
)
ens <- short_term_trend(ens, measure = "numerator_nowcasted", trend_isoyearweeks = 5)
names(ens$draws)

# the share of draws with a positive slope; the first 4 weeks have no window
ens$data[, .(isoyearweek, numerator_nowcasted_trend_increasing_pr)]

# the deprecated csfmt_rts_data_v1 method
d <- cstdy::nor_covid19_icu_and_hospitalization_csfmt_rts_v1
d <- d[granularity_time=="isoyearweek"]
res <- csalert::short_term_trend(
  d,
  numerator = "hospitalization_with_covid19_as_primary_cause_n",
  trend_isoyearweeks = 6
)
print(res[, .(
  isoyearweek,
  hospitalization_with_covid19_as_primary_cause_n,
  hospitalization_with_covid19_as_primary_cause_trend0_41_status
```

```
)])
```

```
short_term_trend_sts_v1
```

Short-term trend alarms on a surveillance sts object

Description

Sets an alarm where a rolling quasi-Poisson regression finds a significant rise, in each column of a `surveillance::sts` object, after Benedetti (2019) [doi:10.5588/pha.19.0002](https://doi.org/10.5588/pha.19.0002). It has not changed since 2024-06-24.

Usage

```
short_term_trend_sts_v1(sts, control = list(w = 5, alpha = 0.05))
```

Arguments

<code>sts</code>	A <code>surveillance::sts</code> object.
<code>control</code>	A list with <code>w</code> , the window width, and <code>alpha</code> , the significance level. A missing element takes its default.

Details

Each time point gets the fit of `observed ~ trend + log(population)`, by `glm2::glm2()`, on the `w` time points that end there. Its alarm is 1 when the slope is positive with a p-value below `alpha`, and 0 otherwise. The first `w - 1` time points keep their alarm.

Value

`sts` with its alarms set.

See Also

`vignette("csalert", package = "csalert")`, which explains the deprecated trend methods. The `csfmt_rts_data_v1` method of `short_term_trend()` is the table form of this function.

Other short-term trend functions: `rolling_slope_matrix()`, `short_term_trend()`

Examples

```
d <- cstudy::nor_covid19_icu_and_hospitalization_csfmt_rts_v1
d <- d[granularity_time=="isoyearweek"]
sts <- surveillance::sts(
  observed = d$hospitalization_with_covid19_as_primary_cause_n, # weekly counts
  start = c(d$isoyear[1], d$isoweek[1]), # the first week of the series
  frequency = 52
)
x <- csalert::short_term_trend_sts_v1(
```

```

    sts,
    control = list(
      w = 5,
      alpha = 0.05
    )
  )
  plot(x)

```

signal_detection_hlm *Flag weeks above a historical limit*

Description

Flags a week as high above the historical limit, the 99.5% quantile of a normal fitted to the same weeks in earlier years. The ensemble method compares every draw with the limit.

Usage

```

signal_detection_hlm(x, ...)

## S3 method for class 'csfmt_rts_data_v1'
signal_detection_hlm(
  x,
  value,
  baseline_isoyears = 5,
  remove_last_isoyearweeks = 0,
  forecast_isoyearweeks = 2,
  value_naming_prefix = "from_numerator",
  remove_training_data = FALSE,
  ...
)

## S3 method for class 'csfmt_rts_data_v3'
signal_detection_hlm(x, ...)

## S3 method for class 'csfmt_ensemble_v3'
signal_detection_hlm(x, measure, baseline_isoyears = 5, ...)

```

Arguments

x	A csfmt_ensemble_v3, or a deprecated csfmt_rts_data_v1.
...	Passed to the method.
value	The value column.
baseline_isoyears	The number of earlier years in the baseline.
remove_last_isoyearweeks	It has no effect: the method never uses it.

forecast_isoyearweeks	The number of weeks to add after the data, with the baseline median as forecast and the status "forecast".
value_naming_prefix	The start of the new column names: "from_numerator" is value without its last _<word>, "generic" is "value", and any other string is used as written.
remove_training_data	If TRUE, drop the "training" weeks.
measure	The draw matrix to compare with the limit.

Details

The baseline of a week is the values 52, 104, and so on up to $52 * \text{baseline_isoyears}$ rows back, each with its two neighbours. The limit is $\text{qnorm}(0.995, \text{mean}, \text{sd})$ of those values. A week that lacks any of them gets none. A year back is 52 rows, so across a 53-week ISO year the baseline is one week off. The limit is treated as known.

Value

The `v1` method returns `x` with the forecast weeks added, and:

- `<value>_status`, a factor with the levels "training", "forecast", "null" and "high",
- `*_forecasted_*`, the value or the forecast, and a logical forecast marker,
- `*_baseline_predinterval_*`, the 0.5%, 50% and 99.5% quantiles of the baseline normal.

The `csfmt_rts_data_v3` method always stops with an error.

The ensemble method returns `x` with a draw matrix `<measure>_hlmstatus`: 1 for null, 2 for high, and NA with no limit, with a `levels` attribute. It adds `hlm_threshold`, the limit, to `$data` by reference, so the input ensemble gets it too.

The deprecated `csfmt_rts_data_v1` method

`signal_detection_hlm.csfmt_rts_data_v1()` is **deprecated**. It still works and gives no warning. New work SHOULD run the ensemble method before `ens_collapse()`:

```
ens <- nowcast_delay_ecdf_v1(triangle, max_delay_days = 35)
ens <- signal_detection_hlm(ens, measure = "numerator_nowcasted")
out <- ens_collapse(ens, heal = TRUE)
```

The replacement is not a drop-in. The `v1` method takes `value`, `remove_last_isoyearweeks`, `forecast_isoyearweeks` and `value_naming_prefix`, where the ensemble method takes a draw matrix, `measure`. The `v1` method returns a label per week. The ensemble method classifies every draw, so after the collapse it gives the share of draws above the limit.

The `v1` method labels a week high when its value is above the limit, and training when it has no baseline.

Why the `csfmt_rts_data_v3` method is an error

A `csfmt_rts_data_v3` is the collapsed output of the pipeline. It holds quantiles, not draws, so no per-draw comparison can come from it. Run the detection before the collapse:

```
ens <- signal_detection_hlm(ens, measure = "numerator_nowcasted") # before
out <- ens_collapse(ens, heal = TRUE)                          # then collapse
```

The `csfmt_ensemble_v3` method

The baseline uses the median of the draws of each earlier week. A draw at or above the limit is high, and otherwise null. After `ens_collapse()`, `<measure>_hlmstatus_prob_high` is the share of draws at or above the limit. That share describes the nowcast uncertainty. It is not a p-value, a posterior probability or a false-alarm rate.

See Also

`vignette("pipeline", package = "csalert")`, stage 7, and `vignette("csalert", package = "csalert")` on which method to use.

Other ensemble operations: `ens_add_rate()`, `ens_collapse()`, `mem_thresholds_v1()`, `short_term_trend()`

Examples

```
# the deprecated csfmt_rts_data_v1 method
d <- cstudy::nor_covid19_icu_and_hospitalization_csfmt_rts_v1
d <- d[granularity_time=="isoyearweek"]
res <- csalert::signal_detection_hlm(
  d,
  value = "hospitalization_with_covid19_as_primary_cause_n",
  baseline_isoyears = 1
)
print(res[, .(
  isoyearweek,
  hospitalization_with_covid19_as_primary_cause_n,
  hospitalization_with_covid19_as_primary_cause_forecasted_n,
  hospitalization_with_covid19_as_primary_cause_forecasted_n_forecast,
  hospitalization_with_covid19_as_primary_cause_baseline_predinterval_q50x0_n,
  hospitalization_with_covid19_as_primary_cause_baseline_predinterval_q99x5_n,
  hospitalization_with_covid19_as_primary_cause_n_status
)])
```

simulate_baseline_data

Simulate daily counts with no outbreak

Description

Simulates daily counts from a Poisson or negative binomial model with a trend, a seasonal pattern and a day-of-week pattern, after Noufaily et al. (2019). Use it to make a series with a known truth.

Usage

```
simulate_baseline_data(
  start_date,
  end_date,
  seasonal_pattern_n,
  weekly_pattern_n,
  alpha,
  beta,
  gamma_1,
  gamma_2,
  gamma_3,
  gamma_4,
  phi,
  shift_1
)
```

Arguments

start_date, end_date	The first and last day, as a Date or a "YYYY-MM-DD" string.
seasonal_pattern_n	The number of seasonal harmonics: 0 for none, 1 for an annual pattern, 2 to add a half-year harmonic.
weekly_pattern_n	The number of weekly harmonics.
alpha	The log of the baseline count.
beta	The trend on the log scale, per day.
gamma_1, gamma_2	The cosine and sine coefficients of the seasonal term.
gamma_3, gamma_4	The cosine and sine coefficients of the weekly term.
phi	The dispersion. 1 gives a Poisson count, and above 1 a negative binomial with variance $\phi * \mu$. Below 1 gives NA counts and a warning.
shift_1	A shift in days, added to t, that moves the peaks.

Details

On day t , from 1, the log of the mean is $\alpha + \beta * (t + \text{shift}_1)$ plus a seasonal term and a weekly term. With $u = 2 * \pi * (t + \text{shift}_1) / p$, each term sums $a * \cos(l * u) + b * \sin(l * u)$ over $l = 1, \dots, n$:

- seasonal: $p = 364$, $n = \text{seasonal_pattern_n}$, $a = \text{gamma}_1$, $b = \text{gamma}_2$,
- weekly: $p = 7$, $n = \text{weekly_pattern_n}$, $a = \text{gamma}_3$, $b = \text{gamma}_4$.

With both n at 0, the log of the mean is $\alpha + \beta * t$. With $\text{seasonal_pattern_n} > 0$, $\text{weekly_pattern_n} = 0$ still adds a weekly term, because $1:0$ in R is $c(1, 0)$.

Value

A `csfmt_rts_data_v1` with one row per day. Its columns include date, time (t), wday (1 is Sunday), mu, the mean, and n, the count.

References

Noufaily A, Enki DG, Farrington P, Garthwaite P, Andrews N, Charlett A. An improved algorithm for outbreak detection in multiple surveillance systems. *Statistics in Medicine*. 2013.

See Also

`vignette("csalert", package = "csalert")`, which simulates a series with a known outbreak.

Other data simulation functions: [add_holiday_effect\(\)](#), [simulate_seasonal_outbreak_data\(\)](#), [simulate_spike_outbreak_data\(\)](#)

Examples

```
library(data.table)
set.seed(4)
baseline <- simulate_baseline_data(
  start_date = as.Date("2018-01-01"),
  end_date = as.Date("2019-12-31"),
  seasonal_pattern_n = 1,
  weekly_pattern_n = 1,
  alpha = 3,
  beta = 0,
  gamma_1 = 0.8,
  gamma_2 = 0.6,
  gamma_3 = 0.8,
  gamma_4 = 0.4,
  phi = 4,
  shift_1 = 29
)
print(baseline[, .(date, wday, mu, n)])
```

`simulate_seasonal_outbreak_data`

Add seasonal outbreaks to simulated daily counts

Description

Adds outbreaks inside a season window to the output of [simulate_baseline_data\(\)](#), after Noufaily et al. (2019). It adds `n_season_outbreak` outbreaks in each of a random number of years.

Usage

```
simulate_seasonal_outbreak_data(
  data,
  week_season_start = 40,
  week_season_peak = 4,
  week_season_end = 20,
  n_season_outbreak = 1,
  m = 50
)
```

Arguments

data	The output of simulate_baseline_data() .
week_season_start	The ISO week that starts the season window.
week_season_peak	Not used: the start day is drawn with random weights, not near the peak.
week_season_end	The ISO week, in the next year, that ends the window.
n_season_outbreak	The number of outbreaks in each outbreak year.
m	The size factor of an outbreak.

Details

The candidate years are those in `calyear`, except the last, and the function prints the years it picks. The window runs from ISO week `week_season_start` of a year to `week_season_end` of the next. An outbreak starts on a day of the window drawn with random weights.

An outbreak adds a Poisson number of cases with mean $10 * m * sd$, where $sd = \sqrt{\mu * \phi}$ on the start day. The draw repeats until it is 2 or more, and it calls `set.seed()`, which resets the random stream. The cases spread over the next days by a lognormal delay. Each day is then weighted: 0.5 on Sunday, 2 on Friday and Saturday, and 1 on other days.

It needs `calyear`. Where `calyear` is NA, as from [simulate_baseline_data\(\)](#) with `cstidy 2026.8.21`, it adds no outbreak.

Value

A copy of data with the cases added to `n`, and these columns:

- `sd` and `weight`,
- `seasonal_outbreak`: 1 on an outbreak day,
- `seasonal_outbreak_n`: the cases,
- `seasonal_outbreak_n_rw`: the weighted cases, which are added to `n`.

References

Noufaily A, Enki DG, Farrington P, Garthwaite P, Andrews N, Charlett A. An improved algorithm for outbreak detection in multiple surveillance systems. *Statistics in Medicine*. 2013.

See Also

`vignette("csalert", package = "csalert")`, which simulates a series with a known outbreak.

Other data simulation functions: `add_holiday_effect()`, `simulate_baseline_data()`, `simulate_spike_outbreak_data()`.

Examples

```
library(data.table)
set.seed(4)
baseline <- simulate_baseline_data(
  start_date = as.Date("2018-01-01"),
  end_date = as.Date("2019-12-31"),
  seasonal_pattern_n = 1,
  weekly_pattern_n = 1,
  alpha = 3,
  beta = 0,
  gamma_1 = 0.8,
  gamma_2 = 0.6,
  gamma_3 = 0.8,
  gamma_4 = 0.4,
  phi = 4,
  shift_1 = 29
)
d <- simulate_seasonal_outbreak_data(
  baseline,
  week_season_start = 40,
  week_season_peak = 4,
  week_season_end = 20,
  n_season_outbreak = 1
)
print(d[, .(date, n, seasonal_outbreak, seasonal_outbreak_n)])
```

`simulate_spike_outbreak_data`

Add short outbreaks to the end of simulated daily counts

Description

Adds `n_sp_outbreak` short outbreaks to the output of `simulate_baseline_data()`, after Noufaily et al. (2019). Each starts on a random day in the last 344 days, 49 weeks, of the data.

Usage

```
simulate_spike_outbreak_data(data, n_sp_outbreak = 1, m)
```

Arguments

<code>data</code>	The output of <code>simulate_baseline_data()</code> .
<code>n_sp_outbreak</code>	The number of outbreaks.
<code>m</code>	The size factor of an outbreak: the mean size is $10 * m * sd$.

Details

The size and the spread are as in [simulate_seasonal_outbreak_data\(\)](#), over about half as many days, and with no day-of-week weight on `n`. The size draw also calls `set.seed()`.

Value

A copy of data with the cases added to `n`, and these columns:

- `sd` and `weight`,
- `sp_outbreak`: 2 on an outbreak day and 0 on other days,
- `sp_outbreak_n`: the cases, which are added to `n`,
- `sp_outbreak_n_rw`: the weighted cases, which `n` does not use.

References

Noufaily A, Enki DG, Farrington P, Garthwaite P, Andrews N, Charlett A. An improved algorithm for outbreak detection in multiple surveillance systems. *Statistics in Medicine*. 2013.

See Also

`vignette("csalert", package = "csalert")`, which runs it.

Other data simulation functions: [add_holiday_effect\(\)](#), [simulate_baseline_data\(\)](#), [simulate_seasonal_outbreak_data\(\)](#)

Examples

```
library(data.table)
set.seed(4)
baseline <- simulate_baseline_data(
  start_date = as.Date("2018-01-01"),
  end_date = as.Date("2019-12-31"),
  seasonal_pattern_n = 1,
  weekly_pattern_n = 1,
  alpha = 3,
  beta = 0,
  gamma_1 = 0.8,
  gamma_2 = 0.6,
  gamma_3 = 0.8,
  gamma_4 = 0.4,
  phi = 4,
  shift_1 = 29
)
d <- simulate_spike_outbreak_data(
  baseline,
  n_sp_outbreak = 1,
  m = 2
)
print(d[, .(date, n, sp_outbreak, sp_outbreak_n)])
```

validate_ensemble	<i>Check the shape of a csfmt_ensemble_v3</i>
-------------------	---

Description

Checks the class, and that `$data` is a `data.table` with `time_series_id` and `time_series_internal_id`. Checks that each entry of `$draws` is a matrix with one row per row of `$data`. Every ensemble stage calls it.

Usage

```
validate_ensemble(ens)
```

Arguments

`ens` The `csfmt_ensemble_v3` to check.

Value

`ens`, invisibly. A failed check is an error.

What it does NOT check

It does NOT check:

- the sort order or the key of `$data`,
- that `time_series_internal_id` counts 1..n in each series,
- that `time_series_label` exists.

It compares only the number of rows, so a draw matrix with its rows in the wrong order passes. If you edit `$data` or `$draws` yourself, rebuild the object with `csfmt_ensemble_v3()`.

See Also

`vignette("pipeline", package = "csalert")`, which shows the ensemble format.

Other ensemble format functions: `csfmt_ensemble_v3()`, `print.csfmt_ensemble_v3()`, `set_time_series_id()`

Examples

```
d <- data.table::data.table(
  location_code = "nation",
  age = "total",
  isoyearweek = c("2023-01", "2023-02", "2023-03")
)
ens <- csfmt_ensemble_v3(
  d,
  id_cols = c("location_code", "age"),
  draws = list(numerator_nowcasted = matrix(1:12, nrow = 3))
)
```

```
)

# returns invisibly when the checks pass
validate_ensemble(ens)

# a draw matrix with the wrong number of rows is an error
bad <- ens
bad$draws$numerator_nowcasted <- matrix(1, nrow = 2, ncol = 4)
try(validate_ensemble(bad))

# a draw matrix with PERMUTED rows has the right count, so it passes
scrambled <- ens
scrambled$draws$numerator_nowcasted <-
  ens$draws$numerator_nowcasted[c(2, 3, 1), , drop = FALSE]
validate_ensemble(scrambled)
"passed, although the draws no longer line up with $data"
```

Index

- * **data simulation functions**
 - add_holiday_effect, 2
 - simulate_baseline_data, 46
 - simulate_seasonal_outbreak_data, 48
 - simulate_spike_outbreak_data, 50
 - * **ensemble format functions**
 - csfmt_ensemble_v3, 5
 - print.csfmt_ensemble_v3, 27
 - set_time_series_id, 38
 - validate_ensemble, 52
 - * **ensemble operations**
 - ens_add_rate, 11
 - ens_collapse, 13
 - mem_thresholds_v1, 15
 - short_term_trend, 39
 - signal_detection_hlm, 44
 - * **naming grammar functions**
 - csfmt_interpret, 7
 - csfmt_parse, 8
 - csfmt_var, 10
 - q_label, 27
 - q_value, 28
 - * **nowcast diagnostics**
 - nowcast_backtest, 17
 - nowcast_censor, 19
 - nowcast_truth, 25
 - * **nowcast engines**
 - nowcast_delay_ecdf_v1, 20
 - nowcast_passthrough_to_ensemble_v1, 22
 - * **quality control functions**
 - compare_results, 4
 - qc_surveillance_data_v1, 29
 - qc_week_over_week_v1, 30
 - * **reporting completion functions**
 - reporting_completion_trend_v1, 32
 - reporting_completion_v1, 33
 - * **reporting triangle functions**
 - csfmt_reporting_triangle_v3, 9
 - isoyearweek_week_start, 14
 - reporting_triangle_matrix, 35
 - * **short-term trend functions**
 - rolling_slope_matrix, 36
 - short_term_trend, 39
 - short_term_trend_sts_v1, 43
- add_holiday_effect, 2
- add_holiday_effect(), 48, 50, 51
- compare_results, 4
- compare_results(), 7, 29–31
- csfmt_ensemble_v3, 5
- csfmt_ensemble_v3(), 27, 38, 52
- csfmt_interpret, 7
- csfmt_interpret(), 4, 8, 11, 28
- csfmt_parse, 8
- csfmt_parse(), 7, 11, 28
- csfmt_reporting_triangle_v3, 9
- csfmt_reporting_triangle_v3(), 15, 36, 38
- csfmt_var, 10
- csfmt_var(), 7, 8, 28
- ens_add_rate, 11
- ens_add_rate(), 14, 16, 42, 46
- ens_collapse, 13
- ens_collapse(), 4, 12, 15, 16, 30, 41, 42, 45, 46
- isoyearweek_week_start, 14
- isoyearweek_week_start(), 10, 36
- mem_thresholds_v1, 15
- mem_thresholds_v1(), 12, 14, 30, 42, 46
- nowcast_backtest, 17
- nowcast_backtest(), 20, 21, 24, 26
- nowcast_censor, 19
- nowcast_censor(), 18, 26

nowcast_delay_ecdf_v1, 20
nowcast_delay_ecdf_v1(), 23
nowcast_passthrough_to_ensemble_v1, 22
nowcast_passthrough_to_ensemble_v1(),
22, 41
nowcast_score_v1, 24
nowcast_score_v1(), 18, 21
nowcast_truth, 25
nowcast_truth(), 18, 20, 24

print.csfmt_ensemble_v3, 27
print.csfmt_ensemble_v3(), 6, 38, 52

q_label, 27
q_label(), 7, 8, 11, 28
q_value, 28
q_value(), 7, 8, 11, 27, 28
qc_surveillance_data_v1, 29
qc_surveillance_data_v1(), 4, 31
qc_week_over_week_v1, 30
qc_week_over_week_v1(), 4, 29

reporting_completion_trend_v1, 32
reporting_completion_trend_v1(), 34
reporting_completion_v1, 33
reporting_completion_v1(), 32
reporting_triangle_matrix, 35
reporting_triangle_matrix(), 10, 15
rolling_slope_matrix, 36
rolling_slope_matrix(), 42, 43

set_time_series_id, 38
set_time_series_id(), 6, 9, 27, 52
short_term_trend, 39
short_term_trend(), 12, 14, 16, 31, 36, 37,
43, 46
short_term_trend_sts_v1, 43
short_term_trend_sts_v1(), 37, 42
signal_detection_hlm, 44
signal_detection_hlm(), 12, 14, 16, 30, 42
simulate_baseline_data, 46
simulate_baseline_data(), 3, 48–51
simulate_seasonal_outbreak_data, 48
simulate_seasonal_outbreak_data(), 3,
48, 51
simulate_spike_outbreak_data, 50
simulate_spike_outbreak_data(), 3, 48,
50

validate_ensemble, 52
validate_ensemble(), 6, 27, 38