Package: csalert (via r-universe)

July 2, 2026

Title Alerts from Public Health Surveillance Data

Version 2026.7.1

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>.

Depends R (>= 3.3.0)

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, magrittr, ggplot2, glm2, cstidy, cstime, digest, lubridate, matrixStats, stringr, surveillance

Suggests testthat, knitr, rmarkdown, rstudioapi, glue, covidnor, csdata, csmaps, ggrepel, mem, plnr

RoxygenNote 7.3.2

VignetteBuilder knitr

Remotes niphr/cstidy

Config/Needs/website niphr/cstemplate

Config/pak/sysreqs libicu-dev

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

Date/Publication 2026-07-02 08:47:03 UTC

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

RemoteRef HEAD

RemoteSha d481348e30e6090c45633b8fdf634077939eccfc

Contents

`add_holiday_effect` *Apply a public holiday effect to simulated data*

Description

Multiplies the daily counts on public holidays by a fixed factor, so that simulated data can reflect the effect of holidays on a time series of daily counts.

Usage

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

Arguments

data	A csfmt_rts_data_v1 data object, typically the output of simulate_baseline_data.
holiday_data	A data.table with a date column and a logical is_holiday column, used to flag which dates are public holidays.
holiday_effect	Multiplicative factor applied to the count n on holidays.

Value

A csfmt_rts_data_v1 (data.table) equal to data with the count n multiplied by holiday_effect on flagged holidays, and a holiday column indicating those dates.

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 csfmt result sets*

Description

Compare two collapsed csfmt result sets

Usage

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

Arguments

current, previous	data.tables (or csfmt_rts_data_v3) from two runs.

Value

A long data.table: identity + isoyearweek + column + role/q/level + ‘cur‘/‘prv‘.

`csfmt_ensemble_v3` *Construct a csfmt_ensemble_v3*

Description

Construct a csfmt_ensemble_v3

Usage

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

Arguments

data	data.table with the identity columns and ‘time_col‘.
id_cols	Character vector of identity columns defining a series.
time_col	Time-ordering column (default "isoyearweek").
draws	Optional named list of ‘[nrow(data) x n_draws]‘ matrices, given in ‘data‘’s input row order (they are reordered to match the canonical sort).

Value

A ‘csfmt_ensemble_v3‘.

csfmt_interpret *Interpret a dataset's columns via the naming grammar*

Description

Applies [csfmt_parse] to every value column (everything not in the structural schema) and returns a catalog: one row per column with its parsed components. This makes a dataset self-describing – generic tooling (QC, collapse, presentation) routes on the catalog instead of hardcoding column names.

Usage

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

Arguments

d	A data.table / data.frame.
value_cols	Optional columns to interpret; defaults to all non-structural.

Value

A data.table: ‘column, measure, denom, role, q, level, per, suffix, interpretable‘ (the last TRUE when a role/quantile/level coordinate was found).

csfmt_parse *Parse a csfmt measure column name into components (inverse of [csfmt_var])*

Description

Parse a csfmt measure column name into components (inverse of [csfmt_var])

Usage

```
csfmt_parse(varname)
```

Arguments

varname	Character scalar column name.

Value

Named list with the components that were present (e.g. ‘measure‘, ‘role‘, ‘q‘, ‘denom‘, ‘per‘), i.e. the inverse of [csfmt_var].

Examples

```
csfmt_parse("numerator_nowcasted_q50x0")
```

csfmt_reporting_triangle_v3 *Construct a csfmt_reporting_triangle_v3*

Description

Construct a csfmt_reporting_triangle_v3

Usage

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

Arguments

`data`	data.table with identity columns, a reference and a reporting ISO-week column, and a value column.
`id_cols`	Identity columns defining a series.
`reference_col, reporting_col`	ISO-week column names.
`value_col`	Count column name.

Value

A validated ‘csfmt_reporting_triangle_v3‘ (a data.table with the as-of boundary and column roles stored as attributes).

csfmt_var *Construct a csfmt measure column name from components*

Description

Construct a csfmt measure column name from components

Usage

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

Arguments

measure	Character scalar, the measure identity (e.g. "consults_r80").
denom	Optional denominator name; inserts '_vs_<denom>'.
role	Optional statistic role: observed/nowcasted/forecasted/trend/baseline/status.
q	Optional probability for a quantile coordinate (mutually exclusive with 'level').
level	Optional status level for a 'prob_<level>' coordinate.
per	Optional rate scaling (e.g. 100 -> '_pr100').
suffix	Optional unit suffix (e.g. "_n").

Value

Character scalar column name.

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 *Add a rate measure to an ensemble*

Description

An ensemble operation ('ens_' family): dispatches on the ensemble class, so the class – not a name prefix on the caller – carries the "operates on an ensemble" meaning, matching [nowcast_quasipoisson_v1()] / [short_term_trend()].

Usage

```
ens_add_rate(x, ...)

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

Arguments

x	A ‘csfmt_ensemble_v3‘.
...	Passed to methods.
numerator, denominator	Measure names present in ‘$draws‘.
per	Scaling factor (e.g. 100 for percent).
name	Optional output measure name (defaults to the grammar name).

Value

‘x‘ with the rate measure added to ‘$draws‘.

---

ens_collapse *Collapse a csfmt_ensemble_v3 to a quantile-summary*

Description

An ensemble operation (‘ens_‘ family): dispatches on the ensemble class, matching [nowcast_quasipoisson_v1()] / [short_term_trend()].

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	A ‘csfmt_ensemble_v3‘.
...	Passed to methods.
probs	Numeric vector of probabilities for the quantile columns.
heal	If TRUE, heal the result into a ‘cstidy::csfmt_rts_data_v3‘ (the clean weekly csfmt) instead of returning a plain data.table.

Value

A ‘data.table‘ (or ‘csfmt_rts_data_v3‘ if ‘heal=TRUE‘): ‘$data‘ plus ‘<measure>_qNNxN‘ columns for every measure in ‘$draws‘; no draws.

`mem_thresholds_v1` *MEM intensity thresholds*

Description

MEM intensity thresholds

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

`x`	Data object.
`...`	Passed to methods.
`measure`	The '$draws' measure to threshold on (a rate or count).
`min_seasons`	Hard floor of complete prior seasons needed to fit.
`prefer_seasons`	Preferred training depth (provisional below this).
`i.seasons`	Max seasons passed to mem::memmodel.
`min_weeks_per_season`	Weeks needed for a season to count as training.
`exclude_seasons`	Optional character vector of seasons (e.g. 'c("2009/2010", "2019/2020")', the 'isoyearweek_to_season_c' form) to drop from the MEM training baseline – anomalous seasons (pandemic years, data gaps) that would distort the thresholds. Thresholds are still ESTIMATED for every season (including excluded ones) from its remaining non-excluded prior seasons; only the baseline they are fit on changes.

Value

The 'csfmt_ensemble_v3' with per-draw MEM intensity columns added to '$draws' (the ordinal 1..5 status for 'measure' and its threshold levels), so the intensity level propagates through the later quantile collapse.

`nowcast_backtest` *Replay a nowcast method across as-of weeks (backtest)*

Description

For each ‘as_of‘ week, censor the triangle to what was known then, run the method, collapse to quantiles, and collect the nowcast for the reference weeks at the requested horizons (horizon = weeks between reference and as-of). An as-of week whose method call errors (e.g. too little history) is skipped with a warning rather than aborting the sweep.

Usage

```
nowcast_backtest(
  triangle,
  method,
  as_of_weeks = NULL,
  max_delay,
  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

`triangle`	A ‘csfmt_reporting_triangle_v3‘ (single series).
`method`	A function ‘f(triangle) -> csfmt_ensemble_v3‘ (params baked in).
`as_of_weeks`	ISO-week strings to replay. Default: every reference week after a ‘max_delay‘-week burn-in, replayed as-of itself.
`max_delay`	Delay horizon (used for the default as-of set and burn-in).
`horizons`	Integer weeks-back to keep (0 = the as-of week itself).
`probs`	Quantile probabilities to extract.
`measure`	Ensemble measure to score; default the numerator’s nowcast.
`seed`	Optional integer base seed. Each as-of is seeded as ‘seed + week-index‘, so a given cell is reproducible regardless of the as-of list order (the nowcast draws for week W depend only on ‘seed‘ and ‘W‘).

Value

A long data.table: ‘reference‘, ‘as_of‘, ‘horizon‘, ‘quantile_level‘, ‘predicted‘.

`nowcast_censor` *Censor a reporting triangle to what was known "as of" a past week*

Description

Keeps only cells reported on or before ‘as_of‘ and rebuilds the triangle, so its as-of boundary and delay structure are exactly what an engine would have seen at that week. The basis for replay-based backtesting.

Usage

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

Arguments

`triangle`	A ‘csfmt_reporting_triangle_v3‘.
`as_of`	An ISO-week string; cells reported after it are dropped.

Value

A ‘csfmt_reporting_triangle_v3‘ censored to ‘as_of‘.

`nowcast_evaluate_v1` *Evaluate nowcast method(s): interval coverage + point-estimate revision*

Description

Replays each method over the triangle (backtest) and scores it on interval coverage (are the intervals honest?) + point-estimate revision (how much will the number still move?), stacked into one per-horizon table with a ‘method‘ column. Pass a single method or a named list; a shared ‘seed‘ pairs them (common random numbers) so a head-to-head is apples-to-apples. Coverage is read straight off the interval quantiles, so this needs no ‘scoringutils‘.

Usage

```
nowcast_evaluate_v1(
  triangle,
  methods,
  max_delay,
  as_of_weeks = NULL,
  horizons = 1:2,
  probs = c(0.025, 0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95, 0.975),
  by = "horizon",
  thresholds = c(0.25, 0.5),
  seed = NULL
)
```

Arguments

`triangle`	A ‘csfmt_reporting_triangle_v3‘ (single series).
`methods`	A method ‘f(triangle) -> csfmt_ensemble_v3‘, or a NAMED list of them (each with its parameters baked in, e.g. via a closure).
`max_delay`	Delay horizon in weeks.
`as_of_weeks`, `horizons`, `probs`, `seed`	Passed to [nowcast_backtest]. ‘seed‘ is shared across methods, so the comparison is paired.
`by`	Grouping for the evaluation summary (default "horizon").
`thresholds`	Absolute-revision cut-offs to report the exceedance probability for (default 25% and 50%).

Value

A data.table, one row per group x method: ‘n‘, interval coverage (‘coverage_50‘, ‘coverage_90‘), the point-estimate revision (‘median_signed‘ bias, ‘median_abs‘, ‘q05‘/‘q95‘ band, ‘p_gt_<t>‘ tails) and ‘method‘.

Examples

```
# a small reporting triangle: 30 weeks, each reported over delays 0-2
w <- cstime::dates_by_isoyearweek$isoyearweek; i <- match("2023-01", w)
d <- data.table::data.table(
  isoyearweek_reference = w[i + rep(0:29, each = 3)],
  isoyearweek_reporting = w[i + rep(0:29, each = 3) + rep(0:2, 30)],
  numerator = 10, indicator = "x", location = "n", age = "total", sex = "total")
tri <- csfmt_reporting_triangle_v3(d, id_cols = c("indicator", "location", "age", "sex"))

# one method:
nowcast_evaluate_v1(tri, function(x) nowcast_passthrough_to_ensemble_v1(x, max_delay = 3),
                    max_delay = 3, horizons = 0:2, seed = 1)
# several methods, paired (common random numbers), stacked with a `method` column:
nowcast_evaluate_v1(tri, max_delay = 3, horizons = 0:2, seed = 1, methods = list(
  passthrough = function(x) nowcast_passthrough_to_ensemble_v1(x, max_delay = 3)))
```

---

nowcast_passthrough_to_ensemble_v1

Build an ensemble from a reporting triangle WITHOUT nowcasting (passthrough)

Description

Collapse the triangle to the observed (reported-so-far) totals per reference week and wrap them as a degenerate single-draw ensemble. An indicator that should NOT be nowcast-completed (because reporting is effectively complete, or the analyst has chosen not to model the delay) then flows through the SAME rate/trend/MEM/collapse pipeline with its observed values unchanged. It emits

the same '<measure>_nowcasted' columns as the modelling engines – here equal to the observed value – so all downstream code is identical; the single draw makes every collapsed quantile equal the observed point.

Usage

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

Arguments

x	A 'csfmt_reporting_triangle_v3'.
max_delay	Delay horizon (defines the contiguous reference grid).
denominator_col	Optional denominator column, carried through the same way (its observed total is also surfaced as '<denom>_observed').

Value

A 'csfmt_ensemble_v3' with single-column draw matrices.

---

nowcast_quasipoisson_v1

Nowcast a reporting triangle into an ensemble (quasipoisson reporting regression)

Description

A discriminative (regression) nowcast engine: for each horizon it regresses the settled total on the counts reported so far ('total ~ n[delay 0] + n[delay 1] + ...', quasipoisson/identity, no intercept) and completes the incomplete weeks by simulating from that fit – parameter uncertainty plus a dispersion-matched negbin. No per-week magnitude parameter, so it is robust for the recent weeks and honestly dispersed. Shares the contract 'f(reporting_triangle, ...) -> csfmt_ensemble_v3'.

Usage

```
nowcast_quasipoisson_v1(x, ...)

## S3 method for class 'csfmt_reporting_triangle_v3'
nowcast_quasipoisson_v1(
  x,
  max_delay,
  n_sim = 1000,
  denominator_col = NULL,
  delay_window = 26,
  ...
)
```

Arguments

x	A ‘csfmt_reporting_triangle_v3‘.
...	Passed to methods.
max_delay	Delay horizon in weeks.
n_sim	Number of nowcast draws.
denominator_col	Optional denominator column to nowcast alongside.
delay_window	Train on only settled weeks within roughly this many weeks (tracks a drifting regime). Default 26; ‘NULL‘ uses all settled weeks.

Value

A ‘csfmt_ensemble_v3‘ with one row per reference week and an ‘n_sim‘-column draw matrix of the nowcasted total per week (settled weeks degenerate at their observed total; incomplete weeks carry the regression’s parameter + dispersion uncertainty). A second measure is added when ‘denominator_col‘ is given.

---

nowcast_truth *The settled (eventually-observed) total per reference week*

Description

Sums each reference week’s counts across all delays up to ‘max_delay‘ – the quantity a nowcast is trying to predict – and keeps only weeks old enough that this total is settled (at least ‘max_delay‘ weeks before the triangle’s as-of).

Usage

```
nowcast_truth(triangle, max_delay)
```

Arguments

triangle	A ‘csfmt_reporting_triangle_v3‘ (single series).
max_delay	Delay horizon in weeks.

Value

A data.table ‘reference‘, ‘truth‘.

prediction_interval *Prediction thresholds*

Description

Prediction thresholds

Usage

```
prediction_interval(object, newdata, alpha = 0.05, z = NULL, ...)
```

Arguments

object	Object
newdata	New data
alpha	Two-sided alpha (e.g 0.05)
z	Similar to alpha (e.g. z=1.96 is the same as alpha=0.05)
...	dots

prediction_interval.glm *Prediction thresholds*

Description

Prediction thresholds

Usage

```
## S3 method for class 'glm'
prediction_interval(
  object,
  newdata,
  alpha = 0.05,
  z = NULL,
  skewness_transform = "none",
  ...
)
```

Arguments

object	Object
newdata	New data
alpha	Two-sided alpha (e.g 0.05)
z	Similar to alpha (e.g. z=1.96 is the same as alpha=0.05)
skewness_transform	"none", "1/2", "2/3"
...	dots

---

print.csfmt_ensemble_v3 *Print a ‘csfmt_ensemble_v3‘*

Description

Compact one-line summary: number of rows, number of time series, and the names of the per-measure draw matrices.

Usage

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

Arguments

x	A ‘csfmt_ensemble_v3‘.
...	Ignored (for S3 consistency).

Value

‘x‘, invisibly.

---

q_label *Probability -> controlled-vocabulary quantile label*

Description

‘0.025 -> "q02x5"‘, ‘0.5 -> "q50x0"‘, ‘0.975 -> "q97x5"‘, ‘0.005 -> "q00x5"‘. Two integer-percent digits, then ‘x‘, then one decimal-percent digit.

Usage

```
q_label(p)
```

Arguments

p	Numeric vector of probabilities in [0, 1].

Value

Character vector of quantile labels.

Examples

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

---

q_value *Quantile label -> probability (inverse of [q_label])*

Description

Quantile label -> probability (inverse of [q_label])

Usage

```
q_value(label)
```

Arguments

label	Character vector of quantile labels, e.g. "q02x5".

Value

Numeric vector of probabilities.

Examples

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

qc_surveillance_data *Quality-control checks on surveillance input data*

Description

Quality-control checks on surveillance input data

Usage

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

Arguments

d	A data.table of one indicator's data.
reference_col	The reference time column (default "isoyearweek_reference").
expect_latest	Optional: the latest reference period that *should* be present. If 'max(reference) < expect_latest', the feed is flagged stale.
min_rows	Minimum rows required (default 1).

Value

A list: 'ok' (logical) and 'reasons' (character vector; empty if ok).

qc_week_over_week *Week-over-week QC: settled-data integrity (A) + frontier status signal (B)*

Description

Week-over-week QC: settled-data integrity (A) + frontier status signal (B)

Usage

```
qc_week_over_week(current, previous, max_delay, tol = 1e-06)
```

Arguments

current, previous	Two runs' collapsed csfmt.
max_delay	Nowcast horizon (weeks); sets the settled/frontier boundary.
tol	Tolerance for "unchanged" in the integrity check.

Value

'list(integrity = <A>, signal = <B>)'.

`reporting_completion` *Empirical reporting-completion summary from a reporting triangle*

Description

Empirical reporting-completion summary from a reporting triangle

Usage

```
reporting_completion(
  triangle,
  max_delay,
  delay_window = NULL,
  period = c("all", "year", "month")
)
```

Arguments

`triangle`	A 'csfmt_reporting_triangle_v3'.
`max_delay`	Delay horizon in weeks.
`delay_window`	Optional: use only settled weeks within roughly this many weeks (drift-aware). 'NULL' uses all settled weeks. Ignored for the shape of 'period' stratification, which slices time itself.
`period`	Time stratification of the settled weeks, by the calendar year / month of each week's Thursday: '"all"' (one pooled curve, default), '"year"', or '"month"' (one row per period). Use '"year"'/'"month"' to see whether completion time is trending up or down.

Value

One row per series (and per period when stratified): identity columns + 'period' + 'n_settled', 'mean_delay', 'complete_by_md' (fraction in by 'max_delay'), and 'pct_w1'..'pct_w<max_delay>' (the pooled % of cases reported after that many weeks observed – the delay ECDF, no interpolation).

reporting_completion_trend_v1

Reporting-completion trend: the delay curve by year and recent months

Description

Convenience over [reporting_completion]: the completion curve sliced by calendar ‘year‘ (all years) and by ‘month‘ (the most recent ‘n_months‘, per series), stacked with a ‘scope‘ column. One table that shows whether reporting is speeding up or slowing down over time.

Usage

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

Arguments

triangle	A ‘csfmt_reporting_triangle_v3‘.
max_delay	Delay horizon in weeks.
n_months	Keep this many most-recent months per series. Default 12.

Value

A data.table: the [reporting_completion] columns plus a ‘scope‘ column ("year"/"month"), the year rows followed by the last-‘n_months‘ month rows. Empty when no series has enough settled data.

Examples

```
w <- cstime::dates_by_isoyearweek$isoyearweek; i <- match("2023-01", w)
d <- data.table::data.table(
  isoyearweek_reference = w[i + rep(0:39, each = 3)],
  isoyearweek_reporting = w[i + rep(0:39, each = 3) + rep(0:2, 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 = 3, n_months = 6)
```

reporting_triangle_matrix

Densify a reporting triangle into per-series reference x delay count matrices

Description

Densify a reporting triangle into per-series reference x delay count matrices

Usage

```
reporting_triangle_matrix(
  triangle,
  max_delay,
  value_col = attr(triangle, "value_col")
)
```

Arguments

triangle	A ‘csfmt_reporting_triangle_v3‘.
max_delay	Number of delay columns (delay 0 .. max_delay-1, in weeks).
value_col	Which value column to reshape (default the triangle’s ‘value_col‘; pass a denominator column to reshape that instead).

Value

Named list (by time_series_id) of ‘list(reference, mat)‘, where ‘mat‘ is a reference x delay count matrix (zeros filled within the observed region).

---

rolling_slope_matrix *Rolling OLS slope over a weeks x draws matrix*

Description

Closed-form simple linear regression of each window (length ‘width‘, time index 1..width) applied independently down every column. Returns matrices of the same shape; leading ‘width-1‘ rows of each column are NA.

Usage

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

Arguments

Y	Numeric matrix, rows = time (ordered), columns = draws.
width	Window width (>= 2).

Value

List of matrices: ‘beta0‘, ‘beta1‘, ‘se‘.

set_time_series_id *Assign content-hash time_series_id (+ readable label) by reference*

Description

Assign content-hash time_series_id (+ readable label) by reference

Usage

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

Arguments

d	data.table.
id_cols	Character vector of identity columns defining a series.
sep	Separator for the canonical key (default unit-separator).

Value

'd', modified by reference (invisibly).

short_term_trend *Determine the short term trend of a timeseries*

Description

Fits a quasi-Poisson regression over a moving window of recent weeks and classifies the short-term trend of the numerator (optionally per a denominator) as increasing or not, together with an estimated doubling time. The method is based upon a published analytics strategy by Benedetti (2019) <doi: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_ensemble_v3'
short_term_trend(x, measure, trend_isoyearweeks = 3, ...)
```

Arguments

x	Data object
...	Not in use.
numerator	Character of name of numerator
denominator	Character of name of denominator (optional)
prX	If using denominator, what scaling factor should be used for numerator/denominator?
trend_isoyearweeks	Rolling window width in isoyearweeks (>= 2).
remove_last_isoyearweeks	Same as remove_last_dates, but used if granularity_geo=='isoyearweek'
forecast_isoyearweeks	Same as forecast_dates, but used if granularity_geo=='isoyearweek'
numerator_naming_prefix	"from_numerator", "generic", or a custom prefix
denominator_naming_prefix	"from_denominator", "generic", or a custom prefix
statistics_naming_prefix	"universal" (one variable for trend status, one variable for doubling dates), "from_numerator_and_prX" (If denominator is NULL, then one variable corresponding to numerator. If denominator exists, then one variable for each of the prXs)
remove_training_data	Boolean. If TRUE, removes the training data (i.e. 1:(trend_dates-1) or 1:(trend_isoyearweeks-1)) from the returned dataset.
include_decreasing	If true, then *_trend*_status contains the levels c("training", "forecast", "decreasing", "null", "increasing"), otherwise the levels c("training", "forecast", "notincreasing", "increasing").
alpha	Significance level for change in trend.
measure	Character: the '$draws' measure to compute the trend on.

Value

The original csfmt_rts_data_v1 dataset with extra columns. *_trend*_status contains a factor with levels c("training", "forecast", "decreasing", "null", "increasing"), while *_doublingdays* contains the expected number of days before the numerator doubles.

The ‘csfmt_ensemble_v3‘ with per-draw short-term-trend columns added to ‘$draws‘ for ‘measure‘ (the rolling slope/level and a P(increasing)), ready for the quantile collapse.

Examples

```
d <- cstidy::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` *Determine the short term trend of a surveillance time series*

Description

Fits a quasi-Poisson regression over a moving window of recent observations of a `surveillance` `sts` object and sets the alarm slot to 1 for time points with a significant increasing trend (0 otherwise). The method is based upon a published analytics strategy by Benedetti (2019) <doi:10.5588/pha.19.0002>. This function was frozen on 2024-06-24 and operates on `sts` objects.

Usage

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

Arguments

`sts`	Data object of type sts.
`control`	Control object, a named list with several elements. **w** Length of the window that is being analyzed. **alpha** Significance level for change in trend.

Value

sts object with the alarms slot set to 0/1 if not-increasing/increasing.

Examples

```
d <- cstidy::nor_covid19_icu_and_hospitalization_csfmt_rts_v1
d <- d[granularity_time=="isoyearweek"]
sts <- surveillance::sts(
  observed = d$hospitalization_with_covid19_as_primary_cause_n, # weekly number of cases
  start = c(d$isoyear[1], d$isoweek[1]), # first week of the time series
  frequency = 52
)
x <- csalert::short_term_trend_sts_v1(
  sts,
  control = list(
    w = 5,
    alpha = 0.05
  )
)
plot(x)
```

`signal_detection_hlm` *Detect signals using the historical limits method*

Description

Flags weeks where the observed value is unusually high compared with a baseline built from the same weeks in previous years. For each week, a baseline mean and standard deviation are computed from the surrounding weeks (`week - 1`, `week`, `week + 1`) in each of the previous `baseline_isoyears` years. A week is flagged as `"high"` when its value exceeds the upper (99.5%) baseline prediction interval.

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_ensemble_v3'
signal_detection_hlm(x, measure, baseline_isoyears = 5, ...)
```

Arguments

x	Data object.
...	Not in use.
value	Character of name of value
baseline_isoyears	Years of history used for the baseline.
remove_last_isoyearweeks	Number of isoyearweeks you want to remove at the end (due to unreliable data)
forecast_isoyearweeks	Number of isoyearweeks you want to forecast into the future
value_naming_prefix	"from_numerator", "generic", or a custom prefix
remove_training_data	Boolean. If TRUE, removes the training data (i.e. the early weeks that have no baseline) from the returned dataset.
measure	The '$draws' measure to detect signals on.

Value

The original csfmt_rts_data_v1 dataset with extra columns. `*_status` is a factor with levels c("training", "forecast", "null", "high") flagging weeks above the baseline, `*_forecasted*` holds the observed value (or the baseline median for forecast weeks), and `*_baseline_predinterval_*` holds the lower (0.5%), median (50%) and upper (99.5%) baseline prediction interval.

The 'csfmt_ensemble_v3' with a per-draw exceedance column added to '$draws' for 'measure' (1 where the draw exceeds its HLM baseline threshold, else 0), so the exceedance probability falls out of the quantile collapse. Weeks without a full baseline are NA.

Examples

```
d <- cstidy::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 baseline surveillance data*

Description

Simulates a time series of daily counts in the absence of outbreaks. The counts are drawn from a Poisson or negative binomial model following the approach of Noufaily et al. (2019). The baseline frequency, linear trend, seasonal pattern and day-of-the-week pattern are all controlled through the function arguments.

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	Starting date of the simulation period. Date is in the format of 'yyyy-mm-dd'.
end_date	Ending date of the simulation period. Date is in the format of 'yyyy-mm-dd'.
seasonal_pattern_n	Number of seasonal patterns. For no seasonal pattern seasonal_pattern_n = 0. Seasonal_pattern_n = 1 represents annual pattern. Seasonal_pattern_n = 2 indicates biannual pattern.
weekly_pattern_n	Number of weekly patterns. For no specific weekly pattern, weekly_pattern_n = 0. Weekly_pattern_n = 1 represents one weekly peak.
alpha	The parameter is used to specify the baseline frequencies of reports
beta	The parameter is used to specify to specify linear trend
gamma_1	The parameter is used to specify the seasonal pattern
gamma_2	The parameter is used to specify the seasonal pattern
gamma_3	The parameter is used to specify day-of-the week pattern
gamma_4	The parameter is used to specify day-of-the week pattern

phi	Dispersion parameter. If phi =0, a Poisson model is used to simulate baseline data.
shift_1	Horizontal shift parameter to help control over week/month peaks.

Value

A csfmt_rts_data_v1 (data.table) holding one row per day over the simulation period, including the columns:

date Calendar date of the observation.

wday Day of the week.

mu Expected count from the baseline model.

n Simulated 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.

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 data

Description

Adds seasonal outbreaks to a simulated baseline time series, for syndromes or diseases that follow seasonal trends. Seasonal outbreaks vary more in size and timing than the underlying seasonal pattern. The number of outbreaks per affected year is set by `n_season_outbreak`, and `week_season_start` to `week_season_end` define the season window. The outbreak start is drawn from the season window, with a higher probability near the peak (`week_season_peak`). The outbreak size (the excess number of cases) is drawn from a Poisson distribution following Noufaily et al. (2019).

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`	A `csfmt_rts_data_v1` data object, typically the output of `simulate_baseline_data`.
`week_season_start`	Starting season week number.
`week_season_peak`	Peak of the season week number.
`week_season_end`	Ending season week number.
`n_season_outbreak`	Number of seasonal outbreaks to be simulated.
`m`	Parameter to determine the size of the outbreak (m times the standard deviation of the baseline count at the starting day of the seasonal outbreak).

Value

A `csfmt_rts_data_v1` (`data.table`) equal to `data` with the simulated seasonal outbreak counts added to column `n` and additional columns describing the outbreaks (e.g. `seasonal_outbreak`, `seasonal_outbreak_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.

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 spiked outbreaks to simulated data

Description

Adds spiked outbreaks to a simulated baseline time series, following Noufaily et al. (2019). The method is similar to simulate_seasonal_outbreak_data, but the outbreaks are shorter in duration and are added only within the last year of data (the prediction period). A spiked outbreak can start at any week during that period.

Usage

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

Arguments

data	A csfmt_rts_data_v1 data object, typically the output of simulate_baseline_data.
n_sp_outbreak	Number of spiked outbreaks to be simulated.
m	Parameter to determine the size of the outbreak (m times the standard deviation of the baseline count at the starting day of the spiked outbreak).

Value

A csfmt_rts_data_v1 (data.table) equal to data with the simulated spiked outbreak counts added to column n and additional columns describing the outbreaks (e.g. sp_outbreak, sp_outbreak_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.

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 *Validate a csfmt_ensemble_v3's invariants*

Description

Validate a csfmt_ensemble_v3's invariants

Usage

```
validate_ensemble(ens)
```

Arguments

ens	A 'csfmt_ensemble_v3'.

Value

‘ens‘ invisibly; errors on violation.

Index