Package: pika (via r-universe)

July 2, 2026

Title Evaluate Correlation Between Time Series

Version 0.2.0

Description This package takes two time series as inputs and first, determines the lag for the primary time series at which the cross-correlation between the two time series is highest. Second, the rolling correlation is calculated between the lagged primary time-series and the secondary time series.

License MIT + file LICENSE

URL https://mrc-ide.github.io/pika/, https://github.com/mrc-ide/pika

BugReports https://github.com/mrc-ide/pika/issues

Encoding UTF-8

LazyData true

Suggests knitr, pkgdown, testthat (>= 2.1.0)

Depends R (>= 3.0.0)

VignetteBuilder knitr

Imports EpiEstim, RColorBrewer, TTR, dplyr, ggplot2, purrr, scales, stats, tidyr, tidyselect

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libicu-dev libssl-dev

Repository https://ncov-ic.r-universe.dev

Date/Publication 2026-07-02 05:44:55 UTC

RemoteUrl https://github.com/mrc-ide/pika

RemoteRef master

RemoteSha 40bc8202e39ea22dd1f2ebf23bad2d8c060a8960

Contents

`calc_percent_change` *Convert a count time series to fractional change relative to a baseline period*

Description

For each group, computes the mean of `count_var` over a baseline period of `n_baseline_periods` consecutive time steps starting at `start_date` (or the earliest date if `start_date` is `NULL`). Each observation is then expressed as a fractional change relative to that baseline mean:

Usage

```
calc_percent_change(
  dat,
  date_var = "date",
  grp_var,
  count_var,
  n_baseline_periods = 7,
  start_date = NULL
)
```

Arguments

`dat`	A data frame containing a count column, a date column, and a grouping column.
`date_var`	Character string giving the name of the date column (class `Date`). Default is `"date"`.
`grp_var`	Character string giving the name of the grouping column. The baseline mean is computed separately per group.
`count_var`	Character string giving the name of the count column.
`n_baseline_periods`	Integer. Number of consecutive time steps used to compute the baseline mean. For daily data, `7` gives a one-week baseline. Default is 7.
`start_date`	Start date of the baseline period. Accepts a `Date` object or a character string in `"YYYY-MM-DD"` format (e.g. `"2020-01-13"`). If `NULL` (default), the earliest date across the combined dataset is used as the baseline start.

Details

$$\texttt{perc\_change} = \frac{\texttt{count} - \texttt{baseline mean}}{\texttt{baseline mean}}$$

A value of 0 indicates no change from baseline; -0.5 indicates a 50% decrease; 1.0 indicates a doubling. Originally developed for population mobility data but applicable to any non-negative count series.

Value

The input data frame with one additional numeric column, `perc_change`, giving each observation as a fractional change relative to the group-specific baseline mean (0 = no change from baseline, -1 = zero counts, positive values = above baseline).

Examples

```
## Not run:
dat_pct <- calc_percent_change(
  dat                = mobility_data,
  date_var           = "date",
  grp_var            = "region",
  count_var          = "trips",
  n_baseline_periods = 7,
  start_date         = "2020-01-13"
)

## End(Not run)
```

---

china_case_data *Daily confirmed cases of COVID-19 in China*

Description

The case data has daily confirmed cases for different provinces in China from 16 January to 24 March 2020 from the dashboard maintained by Chinese Center for Disease Prevention and Control (CCDC). The CCDC dashboard collates numbers of confirmed cases reported by national and local health commissions in each province in mainland China, and Hong Kong SAR and Macau SAR. Confirmed cases are defined as suspected cases, who have epidemiological links and/or clinical symptoms, and are detected with SARS-CoV-2 by PCR tests. However, in Hubei province, clinically diagnosed cases were additionally included between 12 and 19 February.

Usage

```
china_case_data
```

Format

A data frame with 483 rows and 3 variables:

date date, in YYYY-MM-DD format

province name of province/region in China where cases were reported

cases number of daily cases reported of COVID-19

Source

http://2019ncov.chinacdc.cn/2019-nCoV/

`cross_corr` *Find the lag at which cross-correlation between two time series is maximised*

Description

Computes the cross-correlation function (CCF) between `x_var` and `y_var` for lags from `-max_lag` to 0 using `ccf`, then returns the lag with the highest CCF for each group. Only non-positive lags are considered (i.e. `x_var` leading `y_var`), reflecting the assumption that changes in the primary series precede changes in the secondary series.

Usage

```
cross_corr(
  dat,
  date_var = NULL,
  grp_var,
  x_var,
  y_var,
  max_lag = 20,
  subset_date = NULL
)
```

Arguments

`dat`	A data frame containing the two time series and a grouping column.
`date_var`	Character string giving the name of the date column. Required when `subset_date` is non-`NULL`.
`grp_var`	Character string giving the name of the grouping column. The CCF is computed separately within each group.
`x_var`	Character string giving the name of the primary (leading) time series column.
`y_var`	Character string giving the name of the secondary (lagged) time series column.
`max_lag`	Integer. Maximum number of lags to evaluate. CCF is computed for lags `-max_lag` to 0. Default is 20.
`subset_date`	Character string in the same format as `date_var`. If supplied, only rows with dates on or before `subset_date` are used. Requires `date_var` to be specified.

Value

A tibble with one row per group containing:

`<grp_var>` Group identifier; column name matches `grp_var`.

`lag` Integer ≤ 0. The lag at which the CCF between `x_var` and `y_var` is highest within that group.

See Also

`rolling_corr` to compute rolling correlation at the identified lag; `ccf` for the underlying CCF method.

Examples

```
## Not run:
lags <- cross_corr(
  dat      = my_data,
  date_var = "date",
  grp_var  = "region",
  x_var    = "r_mean",
  y_var    = "movement",
  max_lag  = 14
)

## End(Not run)
```

---

estimate_rt *Estimate the effective reproduction number (Rt) over time by group*

Description

A grouped wrapper around `estimate_R` (Cori et al. 2013). For each group, Rt is estimated in a sliding weekly window using a Bayesian framework with a Gamma-distributed serial interval. Results from all groups are combined into a single data frame.

Usage

```
estimate_rt(
  dat,
  grp_var,
  date_var,
  incidence_var,
  est_method = "parametric_si",
  si_mean = 6.48,
  si_std = 3.83
)
```

Arguments

dat	A data frame with at least a date column, an incidence column, and a grouping column. No NA values are permitted in incidence_var.
grp_var	Character string giving the name of the grouping column. Rt is estimated independently for each group.
date_var	Character string giving the name of the date column.
incidence_var	Character string giving the name of the daily incidence (case count) column.
est_method	Character string specifying the serial interval estimation method passed to estimate_R. One of "parametric_si" (default), "non_parametric_si", "uncertain_si", "si_from_data", or "si_from_sample".
si_mean	Mean of the serial interval distribution (days). Used when est_method = "parametric_si". Default is 6.48 (COVID-19; Nishiura et al. 2020).
si_std	Standard deviation of the serial interval distribution (days). Used when est_method = "parametric_si". Default is 3.83 (COVID-19; Nishiura et al. 2020).

Details

The default serial interval parameters (si_mean = 6.48, si_std = 3.83) are from Nishiura et al. (2020) for COVID-19 and should be updated for other pathogens.

Value

A data frame with one row per estimation window per group, containing:

date_start Start date of the estimation window.

date_end End date of the estimation window.

<grp_var> Group identifier; column name matches grp_var.

r_mean Posterior mean Rt.

r_median Posterior median Rt.

r_q2.5 2.5th percentile of the posterior (lower 95% credible interval).

r_q97.5 97.5th percentile of the posterior (upper 95% credible interval).

References

Cori A, Ferguson NM, Fraser C, Cauchemez S (2013). A new framework and software to estimate time-varying reproduction numbers during epidemics. *American Journal of Epidemiology*, 178(9), 1505–1512. doi:10.1093/aje/kwt133

Nishiura H, Linton NM, Akhmetzhanov AR (2020). Serial interval of novel coronavirus (COVID-19) infections. *International Journal of Infectious Diseases*, 93, 284–286. doi:10.1016/j.ijid.2020.02.060

See Also

estimate_R for full estimation control, including non-parametric serial intervals.

Examples

```
## Not run:
rt_estimates <- estimate_rt(
  dat           = china_case_data,
  grp_var       = "province",
  date_var      = "date",
  incidence_var = "cases"
)

## End(Not run)
```

`exante_movement_data` *Daily within-city movement data for different regions in China*

Description

The daily within-city movement data, used as a proxy for economic activity, is available from 1 January to 24 March 2020 for major metropolitan cities within each province in mainland China, Hong Kong SAR, and Macau SAR. These data, provided by Exante Data Inc, measured travel activity relative to the 2019 average (excluding Lunar New Year). The underlying data are based on near real-time people movement statistics from Baidu.

Usage

```
exante_movement_data
```

Format

A data frame with 672 rows and 3 variables:

date date, in YYYY-MM-DD format

province name of province/region in China

movement daily population-weighted within-city movement index

---

`plot_corr` *Plot time series and rolling correlation over time by group*

Description

Produces a faceted line plot showing the primary series (`x_var`), secondary series (`y_var`), and rolling correlation (`roll_corr`) over time, with one facet per group. Horizontal reference lines are drawn at y = -1 (dashed), 0 (solid), and 1 (dashed). Optionally adds a shaded confidence ribbon around `x_var`. If `dat` contains a column named `roll_corr` (e.g. from `rolling_corr`) it will be included in the plot; if absent the line is simply omitted.

Usage

```
plot_corr(
  dat,
  date_var,
  grp_var,
  x_var,
  y_var,
  x_var_lower = NULL,
  x_var_upper = NULL,
  facet_labels = NULL,
  legend_labels = NULL,
  y_max = NULL,
 col_values = c(brewer.pal(8, "RdPu")[8], brewer.pal(8, "Greens")[5], brewer.pal(8,
    "Blues")[8])
)
```

Arguments

dat	A data frame containing the two time series, a roll_corr column, a date column, and a grouping column.
date_var	Character string giving the name of the date column (class Date).
grp_var	Character string giving the name of the grouping column used for faceting.
x_var	Character string giving the name of the primary time series column.
y_var	Character string giving the name of the secondary time series column.
x_var_lower	Character string giving the name of the column containing the lower confidence bound for x_var. If NULL (default), no ribbon is drawn. Both x_var_lower and x_var_upper must be supplied to draw a ribbon.
x_var_upper	Character string giving the name of the column containing the upper confidence bound for x_var. If NULL (default), no ribbon is drawn.
facet_labels	Named character vector of display labels for the facets, passed to as_labeller. Names must match values in the grouping column. If NULL (default), raw group values are shown.
legend_labels	Character vector of length 3 giving legend labels for roll_corr, x_var, and y_var respectively. If NULL (default), the internal metric names (roll_corr, x_var, y_var) are shown in the legend.
y_max	Numeric. Maximum value for the y-axis. If supplied, the axis is set to [-1, y_max] and confidence bounds are clamped to this value. Default is NULL (auto-scaled).
col_values	Character vector of length 3 specifying line colours for roll_corr, x_var, and y_var respectively. Defaults to dark purple, mid-green, and dark blue from RColorBrewer palettes.

Value

A ggplot object.

See Also

rolling_corr to compute roll_corr; plot_lag to visualise the lag distribution.

Examples

```
## Not run:
plot_corr(
  dat           = data_corr,
  date_var      = "date_end",
  grp_var       = "province",
  x_var         = "r_mean",
  y_var         = "movement",
  x_var_lower   = "r_q2.5",
  x_var_upper   = "r_q97.5",
  legend_labels = c("Rolling correlation", "Rt", "Mobility")
)

## End(Not run)
```

---

plot_lag *Plot a histogram of optimal lags across groups*

Description

Produces a histogram of the lag values returned by cross_corr, showing the distribution of optimal lags across groups.

Usage

```
plot_lag(dat, lag_var, bins = 2)
```

Arguments

dat	A data frame containing a lag column, typically the output of cross_corr.
lag_var	Character string giving the name of the lag column to plot.
bins	Numeric. Bin width for the histogram. Default is 2.

Value

A ggplot object.

See Also

cross_corr to compute the lag values; plot_corr to visualise the time series and rolling correlation.

Examples

```
## Not run:
lags <- cross_corr(
  dat     = my_data,
  grp_var = "region",
  x_var   = "r_mean",
  y_var   = "movement"
)
plot_lag(lags, lag_var = "lag")

## End(Not run)
```

`rolling_corr` *Calculate rolling (moving-window) correlation between two time series*

Description

Computes the Pearson correlation between `x_var` and `y_var` over a rolling window of `n` time periods within each group, using `runCor`. The first `n - 1` observations in each group will be `NA` because there are insufficient data to fill the window.

Usage

```
rolling_corr(dat, date_var = "date", grp_var, x_var, y_var, n = 14)
```

Arguments

`dat`	A data frame containing the two time series, a date column, and a grouping column.
`date_var`	Character string giving the name of the date column. Must be of class `Date`. Default is `"date"`.
`grp_var`	Character string giving the name of the grouping column. Rolling correlation is computed separately within each group.
`x_var`	Character string giving the name of the primary time series column.
`y_var`	Character string giving the name of the secondary time series column.
`n`	Integer. Width of the rolling window in time periods. Default is 14.

Value

A data frame with the same columns as the input plus one additional numeric column, `roll_corr`, containing the rolling Pearson correlation between `x_var` and `y_var`. Values range from -1 to 1. The first `n - 1` observations per group are `NA`. Note that rows where `x_var` or `y_var` are `NA` are removed before the rolling correlation is computed, so the returned frame may have fewer rows than the input.

See Also

cross_corr to identify the optimal lag before computing rolling correlation; runCor for the underlying method; plot_corr to visualise the result.

Examples

```
## Not run:
data_corr <- rolling_corr(
  dat      = my_data,
  date_var = "date",
  grp_var  = "region",
  x_var    = "r_mean",
  y_var    = "movement",
  n        = 14
)

## End(Not run)
```

Index