Package 'VisitorCounts'

August 18, 2024

Type Package

Title Modeling and Forecasting Visitor Counts Using Social Media

Version 2.0.2

Date 2024-7-28

Author Robert Bowen [aut, cre],
Russell Goebel [aut],
Beth Ann Brackett [ctb],
Kimihiro Noguchi [aut],
Dylan Way [aut]

Maintainer Robert Bowen <robertbowen.bham@gmail.com>

Description Performs modeling and forecasting of park visitor counts
using social media data and (partial) on-site visitor counts.
Specifically, the model is built based on an automatic decomposition
of the trend and seasonal components of the social media-based park visitor counts,
from which short-term forecasts of the visitor counts and percent changes
in the visitor counts can be made. A reference for the underlying model that 'Visitor-
Counts' uses can be found at
Russell Goebel, Austin Schmaltz, Beth Ann Brackett, Spencer A. Wood, Kimihiro Noguchi (2023) <doi:10.1002/for.2965> .

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Imports Rssa, methods, ggplot2, zoo

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

Depends R (>= 3.5.0)

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2024-08-18 03:50:10 UTC

Contents

`auto_decompose` *Automatic Decomposition Function*

Description

Automatically decomposes a time series using singular spectrum analysis. See package Rssa for details on singular spectrum analysis.

Usage

```
auto_decompose(
  time_series,
  suspected_periods = c(12, 6, 4, 3),
  proportion_of_variance_type = c("leave_out_first", "total"),
  max_proportion_of_variance = 0.995,
  log_ratio_cutoff = 0.2,
  window_length = "auto",
  num_trend_components = 2
)
```

Arguments

`time_series`	A vector which stores the time series of interest in the log scale.
`suspected_periods`	A vector which stores the suspected periods in the descending order of importance. The default option is c(12,6,4,3), corresponding to 12, 6, 4, and 3 months.
`proportion_of_variance_type`	A character string specifying the option for choosing the maximum number of eigenvalues based on the proportion of total variance explained. If "leave_out_first" is chosen, then the contribution made by the first eigenvector is ignored; otherwise, if "total" is chosen, then the contribution made by all the eigenvectors is considered.
`max_proportion_of_variance`	A numeric specifying the proportion of total variance explained using the method specified in proportion_of_variance_type. The default option is 0.995.
`log_ratio_cutoff`	A numeric specifying the threshold for the deviation between the estimated period and candidate periods in suspected_periods. The default option is 0.2, which means that, if the absolute log ratio between the estimated and candidate period is within 0.2 (approximately a 20% difference), then the estimated period is deemed equal to the candidate period.
`window_length`	A character string or positive integer specifying the window length for the SSA estimation. If "auto" is chosen, then the algorithm automatically selects the window length by taking a multiple of 12 which does not exceed half the length of time_series. The default option is "auto".
`num_trend_components`	A positive integer specifying the number of eigenvectors to be chosen for describing the trend in SSA. The default option is 2.

Value

`reconstruction`	A list containing important information about the reconstructed time series. In particular, it contains the reconstructed main trend component, overall trend component, seasonal component for each period specified in suspected_periods, and overall seasonal component.

grouping	A matrix containing information about the locations of the eigenvalue groups for each period in suspected_periods and trend component. The locations are indicated by '1'.
window_length	A numeric indicating the window length.
ts_ssa	An ssa object storing the singular spectrum analysis decomposition.

Examples

```
data("park_visitation")

### Decompose national parks service visitor counts and flickr photo user-days

# parameters ---------------------------------------------
suspected_periods <- c(12,6,4,3)
proportion_of_variance_type = "leave_out_first"
max_proportion_of_variance <- 0.995
log_ratio_cutoff <- 0.2

# load data ----------------------------------------------

park <- "YELL" #for Yellowstone National Park

nps_ts <- ts(park_visitation[park_visitation$park == park,]$nps, start = 2005, freq = 12)
nps_ts <- log(nps_ts)

pud_ts <- ts(park_visitation[park_visitation$park == park,]$pud, start = 2005, freq = 12)
pud_ts <- log(pud_ts)

# decompose time series and plot decompositions -----------
decomp_pud <- auto_decompose(pud_ts,
                                    suspected_periods,
                               proportion_of_variance_type = proportion_of_variance_type,
                                    max_proportion_of_variance,
                                    log_ratio_cutoff)
plot(decomp_pud)

decomp_nps <- auto_decompose(nps_ts,suspected_periods,
                               proportion_of_variance_type = proportion_of_variance_type,
                                    max_proportion_of_variance,log_ratio_cutoff)

plot(decomp_nps)
```

---

check_arguments *Check Arguments*

Description

Check arguments.

Usage

```
check_arguments(
  popularity_proxy,
  onsite_usage,
  constant,
  omit_trend,
  trend,
  ref_series,
  is_input_logged,
  ...
)
```

Arguments

popularity_proxy	A vector which stores a time series which may be used as a proxy for the monthly popularity of social media over time. The length of popularity_proxy must be the same as that of onsite_usage. The default option is NULL, in which case, no proxy needs to be supplied. Note that this vector cannot have a value of 0.
onsite_usage	A vector which stores monthly on-site usage for a particular social media platform and recreational site.
constant	A numeric specifying the constant term (beta0) in the model. This constant is understood as the mean log adjusted monthly visitation relative to the base month. The default option is 0, implying that the (logged) onsite_usage does not require any constant shift, which is unusual. If ref_series is supplied, the constant is overwritten by the least squares estimate.
omit_trend	This is obsolete and is left only for compatibility. In other words, trend will overwrite any option chosen in omit_trend. If trend is NULL, then trend is overwritten according to omit_trend. It is a Boolean specifying whether or not to consider the trend component to be 0. The default option is TRUE, in which case, the trend component is 0. If it is set to FALSE, then it is estimated using data.
trend	A character string specifying how the trend is modeled. Can be any of NULL, "linear", "none", and "estimated", where "none" and "estimated" correspond to omit_trend being TRUE and FALSE, respectively. If NULL, then it follows the value specified in omit_trend.
ref_series	A numeric vector specifying the original visitation series. The default option is NULL, implying that no such series is available. If such series is available, then its length must be the same as that of onsite_usage.
is_input_logged	A boolean specifying if the input is logged or not
...	Additional arguments.

Value

No return value, called for extra information.

convert_ts_forecast_to_df *convert_ts_forecast_to_df*

Description

method for converting a timerseries to a dataframe so that it can be plotted with ggplot2 and keep a Date x-axis.

Usage

```
convert_ts_forecast_to_df(forecast)
```

Arguments

forecast	timeseries object to convert

decompose_proxy *Decompose Popularity Proxy*

Description

Decomposes the popularity proxy time series into trend and seasonality components.

Usage

```
decompose_proxy(
  onsite_usage,
  popularity_proxy = NULL,
  suspected_periods = c(12, 6, 4, 3),
  proportion_of_variance_type = c("leave_out_first", "total"),
  max_proportion_of_variance = 0.995,
  log_ratio_cutoff = 0.2,
  window_length = "auto",
  num_trend_components = 2,
  criterion = c("cross-correlation", "MSE", "rank"),
  possible_lags = -36:36,
  leave_off = 6,
  estimated_change = 0,
  order_of_polynomial_approximation = 7,
  order_of_derivative = 1,
  ref_series = NULL,
  constant = 0,
  beta = "estimate",
  slope = 0,
```

```
  is_input_logged = FALSE,
  spline = FALSE,
  parameter_estimates = c("separate", "joint"),
  omit_trend = TRUE,
  trend = c("linear", "none", "estimated"),
  onsite_usage_decomposition,
  ...
)
```

Arguments

`onsite_usage`	A vector which stores monthly on-site usage for a particular social media platform and recreational site.
`popularity_proxy`	A vector which stores a time series which may be used as a proxy for the monthly popularity of social media over time. The length of `popularity_proxy` must be the same as that of `onsite_usage`. The default option is NULL, in which case, no proxy needs to be supplied. Note that this vector cannot have a value of 0.
`suspected_periods`	A vector which stores the suspected periods in the descending order of importance. The default option is c(12,6,4,3), corresponding to 12, 6, 4, and 3 months if observations are monthly.
`proportion_of_variance_type`	A character string specifying the option for choosing the maximum number of eigenvalues based on the proportion of total variance explained. If "leave_out_first" is chosen, then the contribution made by the first eigenvector is ignored; otherwise, if "total" is chosen, then the contribution made by all the eigenvectors is considered.
`max_proportion_of_variance`	A numeric specifying the proportion of total variance explained using the method specified in `proportion_of_variance_type`. The default option is 0.995.
`log_ratio_cutoff`	A numeric specifying the threshold for the deviation between the estimated period and candidate periods in suspected_periods. The default option is 0.2, which means that if the absolute log ratio between the estimated and candidate period is within 0.2 (approximately a 20 percent difference), then the estimated period is deemed equal to the candidate period.
`window_length`	A character string or positive integer specifying the window length for the SSA estimation. If "auto" is chosen, then the algorithm automatically selects the window length by taking a multiple of 12 which does not exceed half the length of `onsite_usage`. The default option is "auto".
`num_trend_components`	A positive integer specifying the number of eigenvectors to be chosen for describing the trend in SSA. The default option is 2. This is relevant only when `trend` is "estimated".
`criterion`	A character string specifying the criterion for estimating the lag in `popularity_proxy`. If "cross-correlation" is chosen, it chooses the lag that maximizes the correlation

coefficient between lagged popularity_proxy and onsite_usage. If "MSE" is chosen, it does so by identifying the lagged popularity_proxy whose derivative is closest to that of onsite_usage by minimizing the mean squared error. If "rank" is chosen, it does so by firstly ranking the square errors of the derivatives and identifying the lag which would minimize the mean rank.

possible_lags A numeric vector specifying all the candidate lags for popularity_proxy. The default option is -36:36. This is relevant only when trend is "estimated".

leave_off A positive integer specifying the number of observations to be left off when estimating the lag. The default option is 6. This is relevant only when trend is "estimated".

estimated_change
A numeric specifying the estimated change in the visitation trend. The default option is 0, implying no change in the trend.

order_of_polynomial_approximation
A numeric specifying the order of the polynomial approximation of the difference between time series used in estimate_lag. The default option is 7, the seventh-degree polynomial. This is relevant only when trend is "estimated".

order_of_derivative
A numeric specifying the order of derivative for the approximated difference between lagged popularity_proxy and onsite_usage. The default option is 1, the first derivative. This is relevant only when trend is "estimated".

ref_series A numeric vector specifying the original visitation series. The default option is NULL, implying that no such series is available. If such series is available, then its length must be the same as that of onsite_usage.

constant A numeric specifying the constant term (beta0) in the model. This constant is understood as the mean log adjusted monthly visitation relative to the base month. The default option is 0, implying that the (logged) onsite_usage does not require any constant shift, which is unusual. If ref_series is supplied, the constant is overwritten by the least squares estimate.

beta A numeric or a character string specifying the seasonality adjustment factor (beta1). The default option is "estimate", in which case, it is estimated by using the Fisher's z-transformed lag-12 autocorrelation. Even if an actual value is supplied, if ref_series is supplied, it is overwritten by the least squares estimate.

slope A numeric specifying the slope coefficient (beta2) in the model. This constant is applicable only when trend is set to "linear". The default option is 0, implying that the linear trend is absent.

is_input_logged
A Boolean describing whether the onsite_usage, ref_series, and popularity_proxy are in the log scale. The default option is FALSE, in which case the inputs will be assumed to not be logged and will be logged before making forecasts. Setting it to TRUE will assume the inputs are logged.

spline A Boolean specifying whether or not to use a smoothing spline for the lag estimation. This is relevant only when trend is "estimated".

parameter_estimates
A character string specifying how to estimate beta and constant parameters should a reference series be supplied. Both options use least squares estimates,

but "separate" indicates that the differenced series should be used to estimate beta separately from the constant, while "joint" indicates to estimate both using non-differenced detrended series.

`omit_trend` This is obsolete and is left only for compatibility. In other words, `trend` will overwrite any option chosen in `omit_trend`. If `trend` is NULL, then `trend` is overwritten according to `omit_trend`. It is a Boolean specifying whether or not to consider the trend component to be 0. The default option is TRUE, in which case, the trend component is 0. If it is set to FALSE, then it is estimated using data.

`trend` A character string specifying how the trend is modeled. Can be any of NULL, "linear", "none", and "estimated", where "none" and "estimated" correspond to `omit_trend` being TRUE and FALSE, respectively. If NULL, then it follows the value specified in `omit_trend`.

`onsite_usage_decomposition`
A "decomposition" class object containing decomposition data for the onsite usage time series (outputs from ‘auto_decompose‘).

`...` Additional arguments to be passed onto the smoothing spline (`smooth.spline`).

Value

`proxy_decomposition`
A "decomposition" object representing the automatic decomposition obtained from popularity_proxy (see `auto_decompose`).

`lagged_proxy_trend_and_forecasts_window`
A ‘ts‘ object storing the potentially lagged popularity proxy trend and any forecasts needed due to the lag.

`ts_trend_window`
A ‘ts‘ object storing the trend component of the onsite social media usage. This trend component is potentially truncated to match available popularity proxy data.

`ts_seasonality_window`
A ‘ts‘ object storing the seasonality component of the onsite social media usage. This seasonality component is potentially truncated to match available popularity proxy data.

`latest_starttime`
A ‘tsp‘ attribute of a ‘ts‘ object representing the latest of the two start times of the potentially lagged populairty proxy and the onsite social media usage.

`endtime` A ‘tsp‘ attribute of a ‘ts‘ object representing the time of the final onsite usage observation.

`forecasts_needed`
An integer representing the number of forecasts of popularity_proxy needed to obtain all fitted values. Negative values indicate extra observations which may be useful for predictions.

`lag_estimate` A list storing both the MSE-based esitmate and rank-based estimates for the lag.

`estimate_lag` *Estimate Lag Function*

Description

Uses polynomial approximation and derivatives for time series objects to estimate lag between series.

Usage

```
estimate_lag(
  time_series1,
  time_series2,
  possible_lags,
  method = c("cross-correlation", "MSE", "rank"),
  leave_off,
  estimated_change = 0,
  order_of_polynomial_approximation = 7,
  order_of_derivative = 1,
  spline = FALSE,
  ...
)
```

Arguments

`time_series1`	A numeric vector which stores the time series of interest in the log scale.
`time_series2`	A numeric vector which stores the trend proxy time series in the log scale. The length of trend_proxy must be the same as that of time_series1.
`possible_lags`	A numeric vector specifying all the candidate lags for trend_proxy. The default option is -36:36.
`method`	A character vector specifying the method used to obtain the lag estimate. "polynomial" uses polynomial approximation, while "cross-correlation" uses cross-correlation.
`leave_off`	A positive integer specifying the number of observations to be left off when estimating the lag.
`estimated_change`	A numeric specifying the estimated change in the visitation trend. The default option is 0, implying no change in the trend.
`order_of_polynomial_approximation`	A numeric specifying the order of the polynomial approximation of the difference between time series used in `estimate_lag`. The default option is 7, the seventh-degree polynomial.
`order_of_derivative`	A numeric specifying the order of derivative for the approximated difference between time_series1 and lagged time_series2. The default option is 1, the first derivative.

spline	A Boolean specifying whether or not to use a smoothing spline for the lag estimation.
...	Additional arguments to be passed onto the smooth.spline function, if method is "polynomial".

Value

cc_lag	A numeric indicating the estimated lag with the cross-correlation criterion.
mse_criterion	A numeric indicating the estimated lag with the MSE criterion.
rank_criterion	A numeric indicating the estimate lag with the rank criterion.

Examples

```
# Generate dataset with known lag and recover this lag --------------#'

lag <- 3
n <- 156
start_year <- 2005
frequency <- 12
trend_function <- function(x) x^2

x <- seq(-3,3, length.out = n)

y1 <- ts(trend_function(x),start = start_year, freq = frequency)
y2 <- stats::lag(y1, k = lag)


# Recover lag
estimate_lag(y1,y2, possible_lags = -36:36,
             method = "rank",leave_off = 0, spline = FALSE)
```

---

estimate_parameters *Estimate Parameters for Visitation Model*

Description

Estimate the two parameters (y-intercept and seasonality factor) for the visitation model.

Usage

```
estimate_parameters(
  popularity_proxy_decomposition_data = NULL,
  onsite_usage,
  onsite_usage_decomposition,
  omit_trend,
  trend,
```

```
  ref_series,
  constant,
  beta,
  slope,
  parameter_estimates,
  is_input_logged,
  ...
)
```

Arguments

popularity_proxy_decomposition_data	A "decomposition" class object containing decomposition data for the popularity proxy time series (outputs from auto_decompose).
onsite_usage	A vector which stores monthly onsite usage for a particular social media platform and recreational site.
onsite_usage_decomposition	A "decomposition" class object containing decomposition data for the monthly onsite usage time series (outputs from auto_decompose).
omit_trend	This is obsolete and is left only for compatibility. In other words, trend will overwrite any option chosen in omit_trend. If trend is NULL, then trend is overwritten according to omit_trend. It is a Boolean specifying whether or not to consider the trend component to be 0. The default option is TRUE, in which case, the trend component is 0. If it is set to FALSE, then it is estimated using data.
trend	A character string specifying how the trend is modeled. Can be any of NULL, "linear", "none", and "estimated", where "none" and "estimated" correspond to omit_trend being TRUE and FALSE, respectively. If NULL, then it follows the value specified in omit_trend.
ref_series	A numeric vector specifying the original visitation series. The default option is NULL, implying that no such series is available. If such series is available, then its length must be the same as that of onsite_usage.
constant	A numeric specifying the constant term (beta0) in the model. This constant is understood as the mean log adjusted monthly visitation relative to the base month. The default option is 0, implying that the (logged) onsite_usage does not require any constant shift, which is unusual. If ref_series is supplied, the constant is overwritten by the least squares estimate.
beta	A numeric or a character string specifying the seasonality adjustment factor (beta1). The default option is "estimate", in which case, it is estimated by using the Fisher's z-transformed lag-12 autocorrelation. Even if an actual value is supplied, if ref_series is supplied, it is overwritten by the least squares estimate.
slope	A numeric specifying the slope coefficient (beta2) in the model. This constant is applicable only when trend is set to "linear". The default option is 0, implying that the linear trend is absent.
parameter_estimates	A character string specifying how to estimate beta and constant parameters should a reference series be supplied. Both options use least squares estimates,

but "separate" indicates that the differenced series should be used to estimate beta separately from the constant, while "joint" indicates to estimate both using non-differenced detrended series.

`is_input_logged`
: A boolean specifying if the input is logged or not

`...`
: Additional arguments.

Value

`lagged_proxy_trend_and_forecasts_window`
: A 'ts' object storing the potentially lagged popularity proxy trend and any forecasts needed due to the lag.

`ts_trend_window`
: A 'ts' object storing the trend component of the onsite social media usage. This trend component is potentially truncated to match available popularity proxy data.

`ts_seasonality_window`
: A 'ts' object storing the seasonality component of the onsite social media usage. This seasonality component is potentially truncated to match available popularity proxy data.

`latest_starttime`
: A 'tsp' attribute of a 'ts' object representing the latest of the two start times of the potentially lagged populairty proxy and the onsite social media usage.

`endtime`
: A 'tsp' attribute of a 'ts' object representing the time of the final onsite usage observation.

`beta`
: A numeric storing the estimated seasonality adjustment factor.

`constant`
: A numeric storing estimated constant term used in the model.

`slope`
: A numeric storing the estimated slope term used in the model. Applicable when the trend parameter is "linear". Otherwise, NULL is returned.

`fit_model` *Fit Model*

Description

Fit the visitation model.

Usage

```
fit_model(
  parameter_estimates_and_time_series_windows,
  omit_trend,
  trend,
  is_input_logged,
  ...
)
```

Arguments

parameter_estimates_and_time_series_windows	# a list storing the outputs of estimate_parameters, including parameter estimates 'constant', 'beta', and 'slope', as well as data pertaining to time series windows.
omit_trend	This is obsolete and is left only for compatibility. In other words, trend will overwrite any option chosen in omit_trend. If trend is NULL, then trend is overwritten according to omit_trend. It is a Boolean specifying whether or not to consider the trend component to be 0. The default option is TRUE, in which case, the trend component is 0. If it is set to FALSE, then it is estimated using data.
trend	A character string specifying how the trend is modeled. Can be any of NULL, "linear", "none", and "estimated", where "none" and "estimated" correspond to omit_trend being TRUE and FALSE, respectively. If NULL, then it follows the value specified in omit_trend.
is_input_logged	a Boolean specifying if the input is logged or not.
...	Additional arguments

Value

visitation_fit	A vector storing fitted values of visitation model.

---

flickr_userdays *Popularity of Flickr, in User-Days*

Description

A time series representing the popularity of Flickr in the United States, as measured in user-days. Here, user-days count the number of unique users posting on Flickr on a given day.

Usage

```
flickr_userdays
```

Format

A time series object with 156 observations.

Source

Flickr. (2019). Retrieved October, 2019, from https://flickr.com/

forest_visitation *National Forest Visitation Photo-User-Days Data.*

Description

A data frame storing monthly visitation counts by National Forest Service (NFS) for 4 popular US national parks and associated Flickr photo-user-days (PUD). Here, photo-user-days (PUD) count the number of unique users posting a photo on Flickr on a given day from within the boundaries of a given National Forest.

Usage

```
forest_visitation
```

Format

A data frame with 995 observations and 4 variables.

date Date of monthly observation, in year-month-day format.

forest National Forest 3 letter identifier code, except for San Juan County which is labled as SJC.

pud Flickr photo-user-days (PUD). Here, PUD count the number of unique users posting a photo on flickr on a given day from within the boundaries of a given National Forest.

nfs Annual Visitation count for the corresponding forest and year given by the National Forest Service (NFS) and then distributed monthly utilizing the PUD as a proxy.

Source

Flickr (2022). Retrieved August, 2022, from https://flickr.com/

generate_proxy_trend_forecasts *Generate Proxy Trend Forecasts*

Description

Generating proxy trend forecasts from objects of the class "visitation_model".

Usage

```
generate_proxy_trend_forecasts(
  object,
  n_ahead,
  starttime,
  endtime,
  proxy_trend_correction,
  ts_frequency
)
```

Arguments

object	A visitation model object.
n_ahead	The number of desired forecasts.
starttime	The start time of the desired forecasts.
endtime	The end time of the desired forecasts.
proxy_trend_correction	The lag correction needed on the proxy trend.
ts_frequency	Frequency of the time series to forecast.

Value

A time series object storing forecasts for the proxy trend.

---

imputation *Imputation*

Description

Imputation by replacing negative infinities with appropriate numbers.

Usage

```
imputation(x)
```

Arguments

x	A numeric vector (usually the log visitation counts or photo-user days).

Value

A numeric vector with the negative infinities replaced with appropriate numbers.

---

label_visitation_forecast *labeled_visitation_forecast Class*

Description

Class for visitation_model predictions (for use with predict.visitation_model()).

Usage

```
label_visitation_forecast(visitation_forecast, label)
```

Arguments

visitation_forecast	A visitation_forecast object
label	A character string of the label of forecast

Value

Object of class "visitation_forecast_ensemble".

---

new_decomposition *"decomposition" Constructor Function*

Description

Constructs objects of the "decomposition" class.

Usage

```
new_decomposition(reconstruction_list, grouping_matrix, window_length, ts_ssa)
```

Arguments

reconstruction_list	A list containing important information about the reconstructed time series. In particular, it contains the reconstructed main trend component, overall trend component, seasonal component for each period specified in suspected_periods, and overall seasonal component.
grouping_matrix	A matrix containing information about the locations of the eigenvalue groups for each period in suspected_periods and trend component. The locations are indicated by '1'.
window_length	A numeric indicating the window length.
ts_ssa	An object of the class "ssa".

Value

A list of the class "decomposition".

new_visitation_forecast *visitation_forecast Class*

Description

Class for visitation_model predictions (for use with predict.visitation_model()).

Usage

```
new_visitation_forecast(
  forecasts,
  logged_forecasts,
  differenced_logged_forecasts,
  differenced_standard_forecasts,
  n_ahead,
  proxy_forecasts,
  onsite_usage_forecasts,
  beta,
  constant,
  slope,
  criterion,
  past_observations,
  lag_estimate
)
```

Arguments

forecasts	A time series of forecasts for the visitation model. the forecasts will be in the standard scale of visitors per month
logged_forecasts	A time series of the logged forecasts for the visitation model.
differenced_logged_forecasts	A time series of the differenced logged forecasts for the visitation model.
differenced_standard_forecasts	A time series of the exponentiated differenced logged forecasts that are for the visitation model.
n_ahead	An integer describing the number of forecasts made.
proxy_forecasts	A time series of forecasts of the popularity proxy series.
onsite_usage_forecasts	A time series of forecasts of the original time series.
beta	A numeric or a character string specifying the seasonality adjustment factor. (beta_1)

`constant` A numeric specifying the constant term in the model. This constant is understood as the mean of the trend-adjusted time_series. (beta_0)

`slope` A numeric specifying the slope term in the model when a linear trend is assumed. (beta_2)

`criterion` One of "MSE" or "Nonparametric", to specify the criterion used to select the lag.

`past_observations` One of "none", "fitted", or "ref_series". If "fitted", past model fitted values are used. If "ref_series", the reference series in the visitation model object is used. Note that if difference = TRUE, one of these is needed to forecast the first difference.

`lag_estimate` A numeric value specifying the estimated lag in the visitation model.

Value

Object of class "labeled_visitation_forecast".

Object of class "Visitation_forecast".

`new_visitation_forecast_ensemble` *visitation_forecast_ensemble Class*

Description

Class for plotting an array of visitation_forecast objects

Usage

```
new_visitation_forecast_ensemble(visitation_forecasts, labels)
```

Arguments

`visitation_forecasts` An array of visitation_forecast object

`labels` An array of labels associated with visitation_forecast

`new_visitation_model` *"visitation_model" Constructor Function*

Description

Constructs objects of the "visitation_model" class.

Usage

```
new_visitation_model(
  visitation_fit,
  differenced_fit,
  beta,
  constant,
  slope,
  lag_estimate,
  proxy_decomposition,
  onsite_usage_decomposition,
  forecasts_needed,
  ref_series,
  criterion,
  omit_trend,
  trend,
  call
)
```

Arguments

`visitation_fit`	A time series storing the fitted values of the visitation model.
`differenced_fit`	A time series storing the differenced fitted values of the visitation model.
`beta`	Seasonality adjustment factor. (beta_1)
`constant`	A numeric describing the constant term used in the model. (beta_0)
`slope`	A numeric describing the slope term used in the model when trend is set to "linear". (beta_2)
`lag_estimate`	An integer representing the lag parameter for the model fit.
`proxy_decomposition`	A decomposition class object representing the decomposition of a popularity measure (e.g., US Photo-User-Days).
`onsite_usage_decomposition`	A decomposition class object representing the decomposition of time series (e.g., park Photo-User-Days).
`forecasts_needed`	An integer describing how many forecasts for the proxy_decomposition are needed for the fit.

ref_series	A reference time series (or NULL) used in the model fit.
criterion	A character string specifying the criterion for estimating the lag in popularity_proxy. If "cross-correlation" is chosen, it chooses the lag that maximizes the correlation coefficient between lagged popularity_proxy and onsite_usage. If "MSE" is chosen, it does so by identifying the lagged popularity_proxy whose derivative is closest to that of onsite_usage by minimizing the mean squared error. If "rank" is chosen, it does so by firstly ranking the square errors of the derivatives and identifying the lag which would minimize the mean rank.
omit_trend	This is obsolete and is left only for compatibility. A Boolean specifying whether or not to consider the NPS trend to be zero.
trend	A character string specifying how the trend is modeled. Can be any of NULL, "linear", "none", and "estimated", where "none" and "estimated" correspond to omit_trend being TRUE and FALSE, respectively. If NULL, then it follows the value specified in omit_trend.
call	A call for the visitation model.

Value

A list of the class "model_forecast".

---

park_visitation *National Park Visitation Counts and Associated Photo-User-Days Data.*

Description

A data frame storing monthly visitation counts by National Park Service (NPS) for 20 popular US national parks and associated Flickr photo-user-days (PUD). Here, photo-user-days (PUD) count the number of unique users posting a photo on Flickr on a given day from within the boundaries of a given National Park.

Usage

```
park_visitation
```

Format

A data frame with 3276 rows and 4 variables.

date Date of monthly observation, in year-month-day format.

park National Park alpha code identifying a National Park.

pud Flickr photo-user-days (PUD). Here, PUD count the number of unique users posting a photo on flickr on a given day from within the boundaries of a given National Park.

nps Visitation count for the corresponding park and month given by the National Park Service (NPS).

Source

National Park Service (2018). National park service visitor use statistics. Retrieved May 10, 2018 from https://irma.nps.gov/Stats/

Flickr (2019). Retrieved October, 2019, from https://flickr.com/

`plot.decomposition` *Decomposition Plot Methods*

Description

Methods for plotting objects of the class "decomposition".

Usage

```
## S3 method for class 'decomposition'
plot(x, type = c("full", "period", "classical"), legend = TRUE, ...)
```

Arguments

`x`	An object of class "decomposition".
`type`	A character string. One of "full","period", or "classical". If "full", the full reconstruction is plotted. If "period", the reconstruction of each period is plotted individually. If "classical", the trend and seasonality are plotted.
`legend`	A Boolean specifying whether a legend should be added when type is "full". The default option is TRUE.
`...`	Additional arguments.

Value

A plot of the reconstruction in the "decomposition" class object.

Examples

```
data("park_visitation")

park <- "YELL"
nps_ts <- ts(park_visitation[park_visitation$park == park,]$nps, start = 2005, frequency = 12)
nps_ts <- log(nps_ts)

pud_ts <- ts(park_visitation[park_visitation$park == park,]$pud, start = 2005, frequency = 12)
pud_ts <- log(pud_ts)
nps_ts <- ts(park_visitation[park_visitation$park == park,]$nps, start = 2005, frequency = 12)
nps_ts <- log(nps_ts)



decomposition_pud <- auto_decompose(pud_ts)
```

```
decomposition_nps <- auto_decompose(nps_ts)

plot(decomposition_pud,lwd = 2)
plot(decomposition_pud,type = "period")
plot(decomposition_pud,type = "classical")


plot(decomposition_nps,legend = TRUE)


plot(decomposition_nps,type = "period")
plot(decomposition_nps,type = "classical")
```

`plot.visitation_forecast` *visitation_forecast Plot Methods*

Description

Methods for plotting objects of the class "visitation_forecast".

Usage

```
## S3 method for class 'visitation_forecast'
plot(
  x,
  difference = FALSE,
  log_outputs = FALSE,
  actual_visitation = NULL,
  xlab = "Time",
  ylab = "Fitted Value",
  pred_color = "#228B22",
  actual_color = "#FF0000",
  size = 1.5,
  main = "Forecasts for Visitation Model",
  plot_points = FALSE,
  date_breaks = "1 month",
  date_labels = "%y %b",
  ...
)
```

Arguments

`x`	An object of the "visitation_forecast" class.
`difference`	A boolean to plot the differenced series.
`log_outputs`	A boolean to plot the logged outputs of the forecast.

actual_visitation	A timeseries object representing the actual visitation that will be plotted along site the visitation_forecast object.
xlab	A string that will be used for the xlabel of the plot.
ylab	A string that will be used for the ylabel of the plot.
pred_color	a String that will be used for the predicted series color of the plot.
actual_color	a String that will be used for the actual series color of the plot.
size	A number that represents the thickness of the lines being plotted.
main	A string that will be used for the title of the plot.
plot_points	a boolean to specify if the plot should be points or continous line.
date_breaks	A string to represent the distance between dates that the x-axis should be in. ex "1 month", "1 year".
date_labels	A string to represent the format of the x-axis time labels. ex
...	extra arguments to pass in

Value

No return value, called for plotting objects of the class "visitation_forecast".

Examples

```
#' #Example:

data("park_visitation")
data("flickr_userdays")

n_ahead <- 12
park <- "YELL"
pud_ts <- ts(park_visitation[park_visitation$park == park,]$pud, start = 2005, freq = 12)
pud_ts <- log(pud_ts)
trend_proxy <- log(flickr_userdays)

mf <- visitation_model(pud_ts,trend_proxy)
vf <- predict(mf,12, only_new = TRUE)
plot(vf)
```

---

plot.visitation_forecast_ensemble

visitation_model visitation_forecast_ensemble plot Methods

Description

Method for plotting forecast ensemble.

Usage

```
## S3 method for class 'visitation_forecast_ensemble'
plot(
  x,
  difference = FALSE,
  log_outputs = FALSE,
  plot_cumsum = FALSE,
  plot_percent_change = FALSE,
  actual_visitation = NULL,
  actual_visitation_label = "Actual",
  xlab = "Time",
  ylab = "Fitted Value",
  pred_colors = c("#ff6361", "#58508d", "#bc5090", "#003f5c"),
  actual_color = "#ffa600",
  size = 1.5,
  main = "Forecasts for Visitation Model",
  plot_points = FALSE,
  date_breaks = "1 month",
  date_labels = "%y %b",
  ...
)
```

Arguments

x	An object of class visitation_forecast_ensemble.
difference	A Boolean specifying whether to plot the original fit or differenced series. The default option is FALSE, in which case, the series is not differenced.
log_outputs	whether to log the outputted forecasts or not
plot_cumsum	whether to plot the cumulative sum or not
plot_percent_change	whether to plot the percent change or not
actual_visitation	A timeseries object representing the actual visitation that will be plotted along site the visitation_forecast object
actual_visitation_label	a string that will be used for the label of the actual visitation.
xlab	A string that will be used for the xlabel of the plot
ylab	A string that will be used for the ylabel of the plot
pred_colors	an array of Strings that will be used for the predicted series colors of the plot
actual_color	a String that will be used for the actual series color of the plot,
size	A number that represents the thickness of the lines being plotted
main	A string that will be used for the title of the plot
plot_points	a boolean to specify if the plot should be points or continous line.
date_breaks	A string to represent the distance between dates that the x-axis should be in. ex "1 month", "1 year"

date_labels	A string to represent the format of the x-axis time labels.
...	extra arguments to pass in

Value

No return value, called for plotting objects of the class "visitation_forecast".

`plot.visitation_model` *visitation_model Plot Methods*

Description

Methods for plotting objects of the class "decomposition".

Usage

```
## S3 method for class 'visitation_model'
plot(x, type = c("fitted"), difference = FALSE, ...)
```

Arguments

x	An object of class "decomposition".
type	A character string. One of "full","period", or "classical". If "full", the full reconstruction is plotted. If "period", the reconstruction of each period is plotted individually. If "classical", the trend and seasonality are plotted.
difference	A Boolean specifying whether to plot the original fit or differenced series. The default option is FALSE, in which case, the series is not differenced.
...	Additional arguments.

Value

No return value, called for plotting objects of the class "decomposition".

Examples

```
data("park_visitation")
data("flickr_userdays")

park <- "YELL"
pud_ts <- ts(park_visitation[park_visitation$park == park,]$pud, start = 2005, freq = 12)
pud_ts <- log(pud_ts)

nps_ts <- ts(park_visitation[park_visitation$park == park,]$nps, start = 2005, freq = 12)
nps_ts <- log(nps_ts)

nps_decomp <- auto_decompose(nps_ts)
```

```
trend_proxy <- log(flickr_userdays)

vm <- visitation_model(pud_ts,trend_proxy,ref_series = nps_ts)
plot(vm)
```

`predict.decomposition` *Predict Decomposition*

Description

Methods for generating predictions from objects of the class "decomposition".

Usage

```
## S3 method for class 'decomposition'
predict(object, n_ahead, only_new = TRUE, ...)
```

Arguments

`object`	An object of class "decomposition".
`n_ahead`	An integer describing the number of forecasts to make.
`only_new`	A Boolean describing whether or not to include past values.
`...`	Additional arguments.

Value

`forecasts`	A vector with overall forecast values.
`trend_forecasts`	A vector with trend forecast values.
`seasonality_forecasts`	A vector with seasonality forecast values.

Examples

```
data("park_visitation")
suspected_periods <- c(12,6,4,3)
proportion_of_variance_type = "leave_out_first"
max_proportion_of_variance <- 0.995
log_ratio_cutoff <- 0.2

park <- "DEVA"

nps_ts <- ts(park_visitation[park_visitation$park == park,]$nps, start = 2005, freq = 12)
nps_ts <- log(nps_ts)

pud_ts <- ts(park_visitation[park_visitation$park == park,]$pud, start = 2005, freq = 12)
pud_ts <- log(pud_ts)
```

```
nps_ts <- ts(park_visitation[park_visitation$park == park,]$nps, start = 2005, freq = 12)
nps_ts <- log(nps_ts)

decomp_pud <- auto_decompose(pud_ts,
                                      suspected_periods,
                                proportion_of_variance_type = proportion_of_variance_type,
                                      max_proportion_of_variance,
                                     log_ratio_cutoff)
n_ahead = 36
pud_predictions <- predict(decomp_pud,n_ahead = n_ahead, only_new = FALSE)
```

`predict.visitation_model` *Predict Visitation Model*

Description

Methods for generating predictions from objects of the class "visitation_model".

Usage

```
## S3 method for class 'visitation_model'
predict(
  object,
  n_ahead,
  only_new = TRUE,
  past_observations = c("fitted", "reference"),
  ...
)
```

Arguments

`object`	An object of class "visitation_model".
`n_ahead`	An integer indicating how many observations to forecast.
`only_new`	A Boolean specifying whether to include only the forecasts (if TRUE) or the full reconstruction (if FALSE). The default option is TRUE.
`past_observations`	A character string; one of "fitted" or "reference". Here, "fitted" uses the fitted values of the visitation model, while "reference" uses values supplied in 'ref_series'.
`...`	Additional arguments.

Value

A predictions for the automatic decomposition.

`forecasts` A vector with forecast values.

`n_ahead` A numeric that shows the number of steps ahead.

`proxy_forecasts` A vector for the proxy of trend forecasts.

`onsite_usage_forecasts` A vector for the visitation forecasts.

`beta` A numeric for the seasonality adjustment factor.

`constant` A numeric for the value of the constant in the model.

`slope` A numeric for the value of the slope term in the model when trend is set to "linear".

`criterion` A string which specifies the method used to select the appropriate lag. Only applicable if the trend component is part of the forecasts.

`past_observations` A vector which specifies the fitted values for the past observations.

`lag_estimate` A numeric for the estimated lag. Only applicable if the trend component is part of the forecasts.

Examples

```
data("park_visitation")
data("flickr_userdays")

n_ahead <- 36
park <- "ROMO"
pud_ts <- ts(park_visitation[park_visitation$park == park,]$pud, start = 2005, frequency = 12)
pud_ts <- log(pud_ts)

nps_ts <- ts(park_visitation[park_visitation$park == park,]$nps, start = 2005, frequency = 12)
nps_ts <- log(nps_ts)
popularity_proxy <- log(flickr_userdays)

vm <- visitation_model(pud_ts,popularity_proxy, ref_series = nps_ts, trend = "linear")
predict_vm <- predict(vm,n_ahead,
                      only_new = FALSE, past_observations = "reference")
plot(predict_vm, )
predict_vm2 <- predict(vm,n_ahead,
                       only_new = FALSE, past_observations = "reference")
plot(predict_vm2)
```

prediction_warning *Notify User prediction warning on constant is 0*

Description

Notfy the user of details related to the outputs of the model being potentially inaccurate when constant of model is 0.

Usage

```
prediction_warning(constant)
```

Arguments

constant	The B_0 parameter of the model.

Value

No return value

print.decomposition *Decomposition Summary Method*

Description

S3 method for summarizing objects of the class "decomposition".

Usage

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

Arguments

x	An object of class "decomposition".
...	Additional arguments.

Value

A "decomposition" class object.

Examples

```
 data("park_visitation")

park <- "YELL"
nps_ts <- ts(park_visitation[park_visitation$park == park,]$nps, start = 2005, freq = 12)
nps_ts <- log(nps_ts)

pud_ts <- ts(park_visitation[park_visitation$park == park,]$pud, start = 2005, freq = 12)
pud_ts <- log(pud_ts)
nps_ts <- ts(park_visitation[park_visitation$park == park,]$nps, start = 2005, freq = 12)
nps_ts <- log(nps_ts)


decomposition_pud <- auto_decompose(pud_ts)
decomposition_nps <- auto_decompose(nps_ts)
summary(decomposition_pud)
summary(decomposition_nps)
```

---

print.visitation_forecast

visitation_forecast Summary Method

Description

Methods for summarizing objects of the class "decomposition".

Usage

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

Arguments

x	An object of class "decomposition".
...	Additional arguments.

Value

A "decomposition" class object.

Examples

```
#Example:

data("park_visitation")
data("flickr_userdays")

n_ahead <- 12
park <- "YELL"
```

```
pud_ts <- ts(park_visitation[park_visitation$park == park,]$pud, start = 2005, freq = 12)
pud_ts <- log(pud_ts)
trend_proxy <- log(flickr_userdays)

mf <- visitation_model(pud_ts,trend_proxy)
vf <- predict(mf,12, only_new = FALSE)
summary(vf)
```

`print.visitation_model` *visitation_model Summary Method*

Description

Methods for summarizing objects of the class "decomposition".

Usage

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

Arguments

x	An object of class "decomposition".
...	Additional arguments.

Value

A "decomposition" class object.

Examples

```
#Example:

data("park_visitation")
data("flickr_userdays")

n_ahead <- 12
park <- "YELL"
pud_ts <- ts(park_visitation[park_visitation$park == park,]$pud, start = 2005, freq = 12)
pud_ts <- log(pud_ts)
trend_proxy <- log(flickr_userdays)

vm <- visitation_model(pud_ts,trend_proxy)
summary(vm)
```

summary.decomposition *Decomposition Summary Method*

Description

S3 method for summarizing objects of the class "decomposition".

Usage

```
## S3 method for class 'decomposition'
summary(object, ...)
```

Arguments

object	An object of class "decomposition".
...	Additional arguments.

Value

A "decomposition" class object.

Examples

```
data("park_visitation")

park <- "YELL"
nps_ts <- ts(park_visitation[park_visitation$park == park,]$nps, start = 2005, freq = 12)
nps_ts <- log(nps_ts)

pud_ts <- ts(park_visitation[park_visitation$park == park,]$pud, start = 2005, freq = 12)
pud_ts <- log(pud_ts)
nps_ts <- ts(park_visitation[park_visitation$park == park,]$nps, start = 2005, freq = 12)
nps_ts <- log(nps_ts)#'


decomposition_pud <- auto_decompose(pud_ts)
decomposition_nps <- auto_decompose(nps_ts)
summary(decomposition_pud)
summary(decomposition_nps)
```

summary.visitation_forecast *visitation_forecast Summary Method*

Description

Methods for summarizing objects of the class "decomposition".

Usage

```
## S3 method for class 'visitation_forecast'
summary(object, ...)
```

Arguments

object	An object of class "decomposition".
...	Additional arguments.

Value

A "decomposition" class object.

Examples

```
#Example:

data("park_visitation")
data("flickr_userdays")

n_ahead <- 12
park <- "YELL"
pud_ts <- ts(park_visitation[park_visitation$park == park,]$pud, start = 2005, freq = 12)
pud_ts <- log(pud_ts)
trend_proxy <- log(flickr_userdays)

mf <- visitation_model(pud_ts,trend_proxy)
vf <- predict(mf,12, only_new = FALSE)
summary(vf)
```

summary.visitation_model *visitation_model Summary Method*

Description

Methods for summarizing objects of the class "decomposition".

Usage

```
## S3 method for class 'visitation_model'
summary(object, ...)
```

Arguments

object	An object of class "decomposition".
...	Additional arguments.

Value

A "decomposition" class object.

Examples

```
#Example:

data("park_visitation")
data("flickr_userdays")

n_ahead <- 12
park <- "YELL"
pud_ts <- ts(park_visitation[park_visitation$park == park,]$pud, start = 2005, freq = 12)
pud_ts <- log(pud_ts)
trend_proxy <- log(flickr_userdays)

vm <- visitation_model(pud_ts,trend_proxy)
summary(vm)
```

trim_training_data *trim training data*

Description

Makes sure that the provided onsite_usage and ref_series have at least 12 counts and overlap.

Usage

```
trim_training_data(onsite_usage = NULL, ref_series = NULL)
```

Arguments

onsite_usage	A vector which stores monthly on-site usage for a particular social media platform and recreational site.
ref_series	A numeric vector specifying the original visitation series. The default option is NULL, implying that no such series is available. If such series is available, then its length must be the same as that of onsite_usage.

Value

a list of onsite_usage and ref_series that has been trimmed and modified to share same window of time.

---

visitation_model *Visitation Model*

Description

Fits a time series model that uses social media posts and popularity of the social media to model visitation to recreational sites.

Usage

```
visitation_model(
  onsite_usage,
  popularity_proxy = NULL,
  suspected_periods = c(12, 6, 4, 3),
  proportion_of_variance_type = c("leave_out_first", "total"),
  max_proportion_of_variance = 0.995,
  log_ratio_cutoff = 0.2,
  window_length = "auto",
  num_trend_components = 2,
  criterion = c("cross-correlation", "MSE", "rank"),
  possible_lags = -36:36,
  leave_off = 6,
  estimated_change = 0,
  order_of_polynomial_approximation = 7,
  order_of_derivative = 1,
  ref_series = NULL,
  constant = 0,
  beta = "estimate",
  slope = 0,
  is_input_logged = FALSE,
  spline = FALSE,
```

```
  parameter_estimates = c("joint", "separate"),
  omit_trend = TRUE,
  trend = c("linear", "none", "estimated"),
  ...
)
```

Arguments

onsite_usage	A vector which stores monthly on-site usage for a particular social media platform and recreational site.
popularity_proxy	A vector which stores a time series which may be used as a proxy for the monthly popularity of social media over time. The length of popularity_proxy must be the same as that of onsite_usage. The default option is NULL, in which case, no proxy needs to be supplied. Note that this vector cannot have a value of 0.
suspected_periods	A vector which stores the suspected periods in the descending order of importance. The default option is c(12,6,4,3), corresponding to 12, 6, 4, and 3 months if observations are monthly.
proportion_of_variance_type	A character string specifying the option for choosing the maximum number of eigenvalues based on the proportion of total variance explained. If "leave_out_first" is chosen, then the contribution made by the first eigenvector is ignored; otherwise, if "total" is chosen, then the contribution made by all the eigenvectors is considered.
max_proportion_of_variance	A numeric specifying the proportion of total variance explained using the method specified in proportion_of_variance_type. The default option is 0.995.
log_ratio_cutoff	A numeric specifying the threshold for the deviation between the estimated period and candidate periods in suspected_periods. The default option is 0.2, which means that if the absolute log ratio between the estimated and candidate period is within 0.2 (approximately a 20 percent difference), then the estimated period is deemed equal to the candidate period.
window_length	A character string or positive integer specifying the window length for the SSA estimation. If "auto" is chosen, then the algorithm automatically selects the window length by taking a multiple of 12 which does not exceed half the length of onsite_usage. The default option is "auto".
num_trend_components	A positive integer specifying the number of eigenvectors to be chosen for describing the trend in SSA. The default option is 2. This is relevant only when trend is "estimated".
criterion	A character string specifying the criterion for estimating the lag in popularity_proxy. If "cross-correlation" is chosen, it chooses the lag that maximizes the correlation coefficient between lagged popularity_proxy and onsite_usage. If "MSE" is chosen, it does so by identifying the lagged popularity_proxy whose derivative is closest to that of onsite_usage by minimizing the mean squared error. If

"rank" is chosen, it does so by firstly ranking the square errors of the derivatives and identifying the lag which would minimize the mean rank.

`possible_lags` A numeric vector specifying all the candidate lags for `popularity_proxy`. The default option is -36:36. This is relevant only when `trend` is "estimated".

`leave_off` A positive integer specifying the number of observations to be left off when estimating the lag. The default option is 6. This is relevant only when `trend` is "estimated".

`estimated_change`
A numeric specifying the estimated change in the visitation trend. The default option is 0, implying no change in the trend.

`order_of_polynomial_approximation`
A numeric specifying the order of the polynomial approximation of the difference between time series used in `estimate_lag`. The default option is 7, the seventh-degree polynomial. This is relevant only when `trend` is "estimated".

`order_of_derivative`
A numeric specifying the order of derivative for the approximated difference between lagged `popularity_proxy` and `onsite_usage`. The default option is 1, the first derivative. This is relevant only when `trend` is "estimated".

`ref_series` A numeric vector specifying the original visitation series. The default option is NULL, implying that no such series is available. If such series is available, then its length must be the same as that of `onsite_usage`.

`constant` A numeric specifying the constant term (beta0) in the model. This constant is understood as the mean log adjusted monthly visitation relative to the base month. The default option is 0, implying that the (logged) `onsite_usage` does not require any constant shift, which is unusual. If `ref_series` is supplied, the constant is overwritten by the least squares estimate.

`beta` A numeric or a character string specifying the seasonality adjustment factor (beta1). The default option is "estimate", in which case, it is estimated by using the Fisher's z-transformed lag-12 autocorrelation. Even if an actual value is supplied, if `ref_series` is supplied, it is overwritten by the least squares estimate.

`slope` A numeric specifying the slope coefficient (beta2) in the model. This constant is applicable only when `trend` is set to "linear". The default option is 0, implying that the linear trend is absent.

`is_input_logged`
A Boolean describing whether the `onsite_usage`, `ref_series`, and `popularity_proxy` are in the log scale. The default option is FALSE, in which case the inputs will be assumed to not be logged and will be logged before making forecasts. Setting it to TRUE will assume the inputs are logged.

`spline` A Boolean specifying whether or not to use a smoothing spline for the lag estimation. This is relevant only when `trend` is "estimated".

`parameter_estimates`
A character string specifying how to estimate beta and constant parameters should a reference series be supplied. Both options use least squares estimates, but "separate" indicates that the differenced series should be used to estimate beta separately from the constant, while "joint" indicates to estimate both using non-differenced detrended series.

omit_trend — This is obsolete and is left only for compatibility. In other words, trend will overwrite any option chosen in omit_trend. If trend is NULL, then trend is overwritten according to omit_trend. It is a Boolean specifying whether or not to consider the trend component to be 0. The default option is TRUE, in which case, the trend component is 0. If it is set to FALSE, then it is estimated using data.

trend — A character string specifying how the trend is modeled. Can be any of NULL, "linear", "none", and "estimated", where "none" and "estimated" correspond to omit_trend being TRUE and FALSE, respectively. If NULL, then it follows the value specified in omit_trend.

... — Additional arguments to be passed onto the smoothing spline (smooth.spline).

Value

visitation_fit — A vector storing fitted values of visitation model.

differenced_fit — A vector storing differenced fitted values of visitation model. (Equal to diff(visitation_fit).)

constant — A numeric storing estimated constant term used in the model (beta0).

beta — A numeric storing the estimated seasonality adjustment factor (beta1).

slope — A numeric storing estimated slope coefficient term used in the model (beta2).

proxy_decomposition — A "decomposition" object representing the automatic decomposition obtained from popularity_proxy (see auto_decompose).

time_series_decomposition — A "decomposition" object representing the automatic decomposition obtained from onsite_usage (see auto_decompose).

forecasts_needed — An integer representing the number of forecasts of popularity_proxy needed to obtain all fitted values. Negative values indicate extra observations which may be useful for predictions.

lag_estimate — A list storing both the MSE-based estimate and rank-based estimates for the lag.

criterion — A string; one of "cross-correlation", "MSE", or "rank", specifying the method used to select the appropriate lag.

ref_series — The reference series, if one was supplied.

omit_trend — Whether or not trend was considered 0 in the model. This is obsolete and is left only for compatibility.

trend — The trend used in the model.

call — The model call.

See Also

See predict.visitation_model for forecast methods, estimate_lag for details on the lag estimation, and auto_decompose for details on the automatic decomposition of time series using singular spectrum analysis (SSA). See the package Rssa for details regarding singular spectrum analysis.

Examples

```
### load data --------------------

data("park_visitation")
data("flickr_userdays")

park <- "YELL" #Yellowstone National Park
pud_ts <- ts(park_visitation[park_visitation$park == park,]$pud, start = 2005, frequency = 12)
nps_ts <- ts(park_visitation[park_visitation$park == park,]$nps, start = 2005, frequency = 12)


### fit three models ---------------

vm_pud_linear <- visitation_model(onsite_usage = pud_ts,
                                  ref_series = nps_ts,
                                  parameter_estimates = "joint",
                                  trend = "linear")
vm_pud_only <- visitation_model(onsite_usage = pud_ts,
                                popularity_proxy = flickr_userdays,
                                trend = "estimated")
vm_ref_series <- visitation_model(onsite_usage = pud_ts,
                                  popularity_proxy = flickr_userdays,
                                  ref_series = nps_ts,
                                  parameter_estimates = "separate",
                                  possible_lags = -36:36,
                                  trend = "none")


### visualize fit ------------------

plot(vm_pud_linear, ylim = c(-3,3), difference = TRUE)
lines(diff(nps_ts), col = "red")

plot(vm_pud_only, ylim = c(-3,3), difference = TRUE)
lines(diff(nps_ts), col = "red")

plot(vm_ref_series, ylim = c(-3,3), difference = TRUE)
lines(diff(nps_ts), col = "red")
```

---

yearsToMonths *Converting Annual Counts into Monthly Counts*

Description

Convert annual counts into monthly counts using photo-user-days.

Usage

```
yearsToMonths(visitation_years, pud)
```

Arguments

visitation_years	A numeric vector with annual visitation counts. If not available, NA should be entered.
pud	A numeric vector for the monthly photo-user-days corresponding to visitation_years. As such, the length of pud needs to be exactly 12 times as long as visitation_counts.

Value

A numeric vector with estimated monthly visitation counts based on the annual counts and monthly photo-user-days.

Index