Package 'feasts'

October 13, 2022

Title Feature Extraction and Statistics for Time Series

Version 0.3.0

Description Provides a collection of features, decomposition methods,
statistical summaries and graphics functions for the analysing tidy time
series data. The package name 'feasts' is an acronym comprising of its key
features: Feature Extraction And Statistics for Time Series.

Depends R (>= 3.5.0), fabletools (>= 0.3.1)

Imports rlang (>= 0.2.0), tibble (>= 1.4.1), tsibble (>= 0.9.0),
ggplot2 (>= 3.0.0), dplyr (>= 1.0.0), tidyr (>= 0.8.3), scales
(>= 1.1.0), vctrs, lubridate, grid, slider, utils, lifecycle,
gtable

Suggests tsibbledata, pillar (>= 1.0.1), knitr, rmarkdown, testthat,
covr, seasonal, urca, fracdiff, fable, ggrepel

ByteCompile true

VignetteBuilder knitr

License GPL-3

URL http://feasts.tidyverts.org/, https://github.com/tidyverts/feasts/

BugReports https://github.com/tidyverts/feasts/issues

Encoding UTF-8

RoxygenNote 7.2.1

Language en-GB

RdMacros lifecycle

NeedsCompilation no

Author Mitchell O'Hara-Wild [aut, cre],
Rob Hyndman [aut],
Earo Wang [aut],
Di Cook [ctb],
Thiyanga Talagala [ctb] (Correlation features),
Leanne Chhay [ctb] (Guerrero's method)

Maintainer Mitchell O'Hara-Wild <mail@mitchelloharawild.com>

R **topics documented:**

---

feasts-package   *feasts: Feature Extraction and Statistics for Time Series*

Description

Provides a collection of features, decomposition methods, statistical summaries and graphics functions for the analysing tidy time series data. The package name 'feasts' is an acronym comprising of its key features: Feature Extraction And Statistics for Time Series.

Author(s)

Maintainer: Mitchell O'Hara-Wild <mail@mitchelloharawild.com>

Authors:

- Rob Hyndman
- Earo Wang

Other contributors:

- Di Cook [contributor]
- Thiyanga Talagala (Correlation features) [contributor]
- Leanne Chhay (Guerrero's method) [contributor]

See Also

Useful links:

- <http://feasts.tidyverts.org/>
- <https://github.com/tidyverts/feasts/>
- Report bugs at <https://github.com/tidyverts/feasts/issues>

| ACF | *(Partial) Autocorrelation and Cross-Correlation Function Estimation* |

Description

The function ACF computes an estimate of the autocorrelation function of a (possibly multivariate) tsibble. Function PACF computes an estimate of the partial autocorrelation function of a (possibly multivariate) tsibble. Function CCF computes the cross-correlation or cross-covariance of two columns from a tsibble.

Usage

```
ACF(
  .data,
  y,
  ...,
  lag_max = NULL,
  type = c("correlation", "covariance", "partial"),
  na.action = na.contiguous,
  demean = TRUE,
  tapered = FALSE
)

PACF(.data, y, ..., lag_max = NULL, na.action = na.contiguous, tapered = FALSE)
```

```
CCF(
  .data,
  y,
  x,
  ...,
  lag_max = NULL,
  type = c("correlation", "covariance"),
  na.action = na.contiguous
)
```

Arguments

`.data`	A tsibble
`...`	The column(s) from the tsibble used to compute the ACF, PACF or CCF.
`lag_max`	maximum lag at which to calculate the acf. Default is 10*log10(N/m) where N is the number of observations and m the number of series. Will be automatically limited to one less than the number of observations in the series.
`type`	character string giving the type of acf to be computed. Allowed values are `"correlation"` (the default), `"covariance"` or `"partial"`. Will be partially matched.
`na.action`	function to be called to handle missing values. `na.pass` can be used.
`demean`	logical. Should the covariances be about the sample means?
`tapered`	Produces banded and tapered estimates of the (partial) autocorrelation.
`x, y`	a univariate or multivariate (not `ccf`) numeric time series object or a numeric vector or matrix, or an `"acf"` object.

Details

The functions improve the [stats::acf()](), [stats::pacf()]() and [stats::ccf()]() functions. The main differences are that ACF does not plot the exact correlation at lag 0 when `type=="correlation"` and the horizontal axes show lags in time units rather than seasonal units.

The resulting tables from these functions can also be plotted using [autoplot.tbl_cf()]().

Value

The `ACF`, `PACF` and `CCF` functions return objects of class "tbl_cf", which is a tsibble containing the correlations computed.

Author(s)

Mitchell O'Hara-Wild and Rob J Hyndman

References

Hyndman, R.J. (2015). Discussion of "High-dimensional autocovariance matrices and optimal linear prediction". *Electronic Journal of Statistics*, 9, 792-796.

McMurry, T. L., & Politis, D. N. (2010). Banded and tapered estimates for autocovariance matrices and the linear process bootstrap. *Journal of Time Series Analysis*, 31(6), 471-482.

See Also

stats::acf(), stats::pacf(), stats::ccf()

Examples

```
library(tsibble)
library(tsibbledata)
library(dplyr)

vic_elec %>% ACF(Temperature)

vic_elec %>% ACF(Temperature) %>% autoplot()

vic_elec %>% PACF(Temperature)

vic_elec %>% PACF(Temperature) %>% autoplot()

global_economy %>%
  filter(Country == "Australia") %>%
  CCF(GDP, Population)

global_economy %>%
  filter(Country == "Australia") %>%
  CCF(GDP, Population) %>%
  autoplot()
```

---

autoplot.tbl_cf   *Auto- and Cross- Covariance and -Correlation plots*

Description

Produces an appropriate plot for the result of ACF(), PACF(), or CCF().

Usage

```
## S3 method for class 'tbl_cf'
autoplot(object, level = 95, ...)
```

Arguments

object	A tbl_cf object (the result ACF(), PACF(), or CCF()).
level	The level of confidence for the blue dashed lines.
...	Unused.

Value

A ggplot object showing the correlations.

```
classical_decomposition
```
Classical Seasonal Decomposition by Moving Averages

Description

Decompose a time series into seasonal, trend and irregular components using moving averages. Deals with additive or multiplicative seasonal component.

Usage

```
classical_decomposition(formula, type = c("additive", "multiplicative"), ...)
```

Arguments

formula	Decomposition specification (see "Specials" section).
type	The type of seasonal component. Can be abbreviated.
...	Other arguments passed to stats::decompose().

Details

The additive model used is:

$$Y_t = T_t + S_t + e_t$$

The multiplicative model used is:

$$Y_t = T_t \, S_t \, e_t$$

The function first determines the trend component using a moving average (if filter is NULL, a symmetric window with equal weights is used), and removes it from the time series. Then, the seasonal figure is computed by averaging, for each time unit, over all periods. The seasonal figure is then centered. Finally, the error component is determined by removing trend and seasonal figure (recycled as needed) from the original time series.

This only works well if x covers an integer number of complete periods.

Value

A fabletools::dable() containing the decomposed trend, seasonality and remainder from the classical decomposition.

Specials

season: The season special is used to specify seasonal attributes of the decomposition.

```
season(period = NULL)
```

period   The periodic nature of the seasonality. This can be either a number indicating the number of observations in each sea

Examples

```
as_tsibble(USAccDeaths) %>%
  model(classical_decomposition(value)) %>%
  components()

as_tsibble(USAccDeaths) %>%
  model(classical_decomposition(value ~ season(12), type = "mult")) %>%
  components()
```

---

coef_hurst   *Hurst coefficient*

Description

Computes the Hurst coefficient indicating the level of fractional differencing of a time series.

Usage

```
coef_hurst(x)
```

Arguments

x   a vector. If missing values are present, the largest contiguous portion of the
                     vector is used.

Value

A numeric value.

Author(s)

Rob J Hyndman

---

feat_acf   *Autocorrelation-based features*

Description

Computes various measures based on autocorrelation coefficients of the original series, first-differenced
series and second-differenced series

Usage

```
feat_acf(x, .period = 1, lag_max = NULL, ...)
```

Arguments

x	a univariate time series
.period	The seasonal period (optional)
lag_max	maximum lag at which to calculate the acf. The default is `max(.period, 10L)` for `feat_acf`, and `max(.period, 5L)` for `feat_pacf`
...	Further arguments passed to `stats::acf()` or `stats::pacf()`

Value

A vector of 6 values: first autocorrelation coefficient and sum of squared of first ten autocorrelation coefficients of original series, first-differenced series, and twice-differenced series. For seasonal data, the autocorrelation coefficient at the first seasonal lag is also returned.

Author(s)

Thiyanga Talagala

---

feat_intermittent   *Intermittency features*

Description

Computes various measures that can indicate the presence and structures of intermittent data.

Usage

```
feat_intermittent(x)
```

Arguments

x	A vector to extract features from.

Value

A vector of named features:

- zero_run_mean: The average interval between non-zero observations
- nonzero_squared_cv: The squared coefficient of variation of non-zero observations
- zero_start_prop: The proportion of data which starts with zero
- zero_end_prop: The proportion of data which ends with zero

References

Kostenko, A. V., & Hyndman, R. J. (2006). A note on the categorization of demand patterns. *Journal of the Operational Research Society*, 57(10), 1256-1257.

---

feat_pacf *Partial autocorrelation-based features*

Description

Computes various measures based on partial autocorrelation coefficients of the original series, first-differenced series and second-differenced series.

Usage

```
feat_pacf(x, .period = 1, lag_max = NULL, ...)
```

Arguments

x	a univariate time series
.period	The seasonal period (optional)
lag_max	maximum lag at which to calculate the acf. The default is max(.period, 10L) for feat_acf, and max(.period, 5L) for feat_pacf
...	Further arguments passed to stats::acf() or stats::pacf()

Value

A vector of 3 values: Sum of squared of first 5 partial autocorrelation coefficients of the original series, first differenced series and twice-differenced series. For seasonal data, the partial autocorrelation coefficient at the first seasonal lag is also returned.

Author(s)

Thiyanga Talagala

---

feat_spectral *Spectral features of a time series*

Description

Computes spectral entropy from a univariate normalized spectral density, estimated using an AR model.

Usage

```
feat_spectral(x, .period = 1, ...)
```

Arguments

x	a univariate time series
.period	The seasonal period.
...	Further arguments for `stats::spec.ar()`

Details

The *spectral entropy* equals the Shannon entropy of the spectral density $f_x(\lambda)$ of a stationary process x_t:

$$H_s(x_t) = -\int_{-\pi}^{\pi} f_x(\lambda) \log f_x(\lambda) d\lambda,$$

where the density is normalized such that $\int_{-\pi}^{\pi} f_x(\lambda) d\lambda = 1$. An estimate of $f(\lambda)$ can be obtained using `spec.ar` with the burg method.

Value

A non-negative real value for the spectral entropy $H_s(x_t)$.

Author(s)

Rob J Hyndman

References

Jerry D. Gibson and Jaewoo Jung (2006). "The Interpretation of Spectral Entropy Based Upon Rate Distortion Functions". IEEE International Symposium on Information Theory, pp. 277-281.

Goerg, G. M. (2013). "Forecastable Component Analysis". Journal of Machine Learning Research (JMLR) W&CP 28 (2): 64-72, 2013. Available at https://proceedings.mlr.press/v28/goerg13.html.

See Also

`spec.ar`

Examples

```
feat_spectral(rnorm(1000))
feat_spectral(lynx)
feat_spectral(sin(1:20))
```

---

feat_stl   *STL features*

Description

Computes a variety of measures extracted from an STL decomposition of the time series. This includes details about the strength of trend and seasonality.

Usage

```
feat_stl(x, .period, s.window = 11, ...)
```

Arguments

x	A vector to extract features from.
.period	The period of the seasonality.
s.window	The seasonal window of the data (passed to `stats::stl()`)
...	Further arguments passed to `stats::stl()`

Value

A vector of numeric features from a STL decomposition.

See Also

Forecasting Principle and Practices: Measuring strength of trend and seasonality

---

generate.stl_decomposition

Generate block bootstrapped series from an STL decomposition

Description

Produces new data with the same structure by resampling the residuals using a block bootstrap procedure. This method can only generate within sample, and any generated data out of the trained sample will produce NA simulations.

Usage

```
## S3 method for class 'stl_decomposition'
generate(x, new_data, specials = NULL, ...)
```

Arguments

x	A fitted model.
new_data	A `tsibble` containing future information used to forecast.
specials	(passed by [`fabletools::forecast.mdl_df()`](#)).
...	Additional arguments for forecast model methods.

References

Bergmeir, C., R. J. Hyndman, and J. M. Benitez (2016). Bagging Exponential Smoothing Methods using STL Decomposition and Box-Cox Transformation. International Journal of Forecasting 32, 303-312.

Examples

```
as_tsibble(USAccDeaths) %>%
  model(STL(log(value))) %>%
  generate(as_tsibble(USAccDeaths), times = 3)
```

---

gg_arma   *Plot characteristic ARMA roots*

Description

Produces a plot of the inverse AR and MA roots of an ARIMA model. Inverse roots outside the unit circle are shown in red.

Usage

```
gg_arma(data)
```

Arguments

data	A mable containing models with AR and/or MA roots.

Details

Only models which compute ARMA roots can be visualised with this function. That is to say, the `glance()` of the model contains `ar_roots` and `ma_roots`.

Value

A ggplot object the characteristic roots from ARMA components.

Examples

```
if (requireNamespace("fable", quietly = TRUE)) {
library(fable)
library(tsibble)
library(dplyr)

tsibbledata::aus_retail %>%
  filter(
    State == "Victoria",
    Industry == "Cafes, restaurants and catering services"
  ) %>%
  model(ARIMA(Turnover ~ pdq(0,1,1) + PDQ(0,1,1))) %>%
  gg_arma()
}
```

---

gg_lag   *Lag plots*

Description

A lag plot shows the time series against lags of itself. It is often coloured the seasonal period to identify how each season correlates with others.

Usage

```
gg_lag(
  data,
  y = NULL,
  period = NULL,
  lags = 1:9,
  geom = c("path", "point"),
  arrow = FALSE,
  ...
)
```

Arguments

data	A tidy time series object (tsibble)
y	The variable to plot (a bare expression). If NULL, it will automatically selected from the data.
period	The seasonal period to display.
lags	A vector of lags to display as facets.
geom	The geometry used to display the data.
arrow	Arrow specification to show the direction in the lag path. If TRUE, an appropriate default arrow will be used. Alternatively, a user controllable arrow created with grid::arrow() can be used.
...	Additional arguments passed to the geom.

Value

A ggplot object showing a lag plot of a time series.

Examples

```
library(tsibble)
library(dplyr)
tsibbledata::aus_retail %>%
  filter(
    State == "Victoria",
    Industry == "Cafes, restaurants and catering services"
  ) %>%
  gg_lag(Turnover)
```

---

gg_season   *Seasonal plot*

Description

Produces a time series seasonal plot. A seasonal plot is similar to a regular time series plot, except the x-axis shows data from within each season. This plot type allows the underlying seasonal pattern to be seen more clearly, and is especially useful in identifying years in which the pattern changes.

Usage

```
gg_season(
  data,
  y = NULL,
  period = NULL,
  facet_period = NULL,
  max_col = Inf,
  max_col_discrete = 7,
  pal = (scales::hue_pal())(9),
  polar = FALSE,
  labels = c("none", "left", "right", "both"),
  labels_repel = FALSE,
  labels_left_nudge = 0,
  labels_right_nudge = 0,
  ...
)
```

Arguments

data	A tidy time series object (tsibble)
y	The variable to plot (a bare expression). If NULL, it will automatically selected from the data.

period	The seasonal period to display.
facet_period	A secondary seasonal period to facet by (typically smaller than period).
max_col	The maximum number of colours to display on the plot. If the number of seasonal periods in the data is larger than max_col, the plot will not include a colour. Use max_col = 0 to never colour the lines, or Inf to always colour the lines. If labels are used, then max_col will be ignored.
max_col_discrete	
	The maximum number of colours to show using a discrete colour scale.
pal	A colour palette to be used.
polar	If TRUE, the season plot will be shown on polar coordinates.
labels	Position of the labels for seasonal period identifier.
labels_repel	If TRUE, the seasonal period identifying labels will be repelled with the ggrepel package.
labels_left_nudge, labels_right_nudge	
	Allows seasonal period identifying labels to be nudged to the left or right from their default position.
...	Additional arguments passed to geom_line()

Value

A ggplot object showing a seasonal plot of a time series.

References

Hyndman and Athanasopoulos (2019) Forecasting: principles and practice, 3rd edition, OTexts: Melbourne, Australia. https://OTexts.com/fpp3/

Examples

```
library(tsibble)
library(dplyr)
tsibbledata::aus_retail %>%
  filter(
    State == "Victoria",
    Industry == "Cafes, restaurants and catering services"
  ) %>%
  gg_season(Turnover)
```

gg_subseries *Seasonal subseries plots*

Description

A seasonal subseries plot facets the time series by each season in the seasonal period. These facets form smaller time series plots consisting of data only from that season. If you had several years of monthly data, the resulting plot would show a separate time series plot for each month. The first subseries plot would consist of only data from January. This case is given as an example below.

Usage

```
gg_subseries(data, y = NULL, period = NULL, ...)
```

Arguments

data	A tidy time series object (tsibble)
y	The variable to plot (a bare expression). If NULL, it will automatically selected from the data.
period	The seasonal period to display.
...	Additional arguments passed to geom_line()

Details

The horizontal lines are used to represent the mean of each facet, allowing easy identification of seasonal differences between seasons. This plot is particularly useful in identifying changes in the seasonal pattern over time.

similar to a seasonal plot (gg_season()), and

Value

A ggplot object showing a seasonal subseries plot of a time series.

References

Hyndman and Athanasopoulos (2019) Forecasting: principles and practice, 3rd edition, OTexts: Melbourne, Australia. https://OTexts.com/fpp3/

Examples

```
library(tsibble)
library(dplyr)
tsibbledata::aus_retail %>%
  filter(
    State == "Victoria",
    Industry == "Cafes, restaurants and catering services"
  ) %>%
```

```
gg_subseries(Turnover)
```

---

gg_tsdisplay   *Ensemble of time series displays*

Description

Plots a time series along with its ACF along with an customisable third graphic of either a PACF, histogram, lagged scatterplot or spectral density.

Usage

```
gg_tsdisplay(
  data,
  y = NULL,
  plot_type = c("auto", "partial", "season", "histogram", "scatter", "spectrum"),
  lag_max = NULL
)
```

Arguments

data	A tidy time series object (tsibble)
y	The variable to plot (a bare expression). If NULL, it will automatically selected from the data.
plot_type	type of plot to include in lower right corner. By default ("auto") a season plot will be shown for seasonal data, a spectrum plot will be shown for non-seasonal data without missing values, and a PACF will be shown otherwise.
lag_max	maximum lag at which to calculate the acf. Default is 10*log10(N/m) where N is the number of observations and m the number of series. Will be automatically limited to one less than the number of observations in the series.

Value

A list of ggplot objects showing useful plots of a time series.

Author(s)

Rob J Hyndman & Mitchell O'Hara-Wild

References

Hyndman and Athanasopoulos (2019) *Forecasting: principles and practice*, 3rd edition, OTexts: Melbourne, Australia. <https://OTexts.com/fpp3/>

See Also

[plot.ts](), [ACF](), [spec.ar]()

Examples

```
library(tsibble)
library(dplyr)
tsibbledata::aus_retail %>%
  filter(
    State == "Victoria",
    Industry == "Cafes, restaurants and catering services"
  ) %>%
  gg_tsdisplay(Turnover)
```

---

gg_tsresiduals   *Ensemble of time series residual diagnostic plots*

Description

Plots the residuals using a time series plot, ACF and histogram.

Usage

```
gg_tsresiduals(data, type = "innovation", ...)
```

Arguments

data	A mable containing one model with residuals.
type	The type of residuals to compute. If type="response", residuals on the back-transformed data will be computed.
...	Additional arguments passed to gg_tsdisplay().

Value

A list of ggplot objects showing a useful plots of a time series model's residuals.

References

Hyndman and Athanasopoulos (2019) *Forecasting: principles and practice*, 3rd edition, OTexts: Melbourne, Australia. https://OTexts.com/fpp3/

See Also

gg_tsdisplay()

Examples

```
if (requireNamespace("fable", quietly = TRUE)) {
library(fable)

tsibbledata::aus_production %>%
  model(ETS(Beer)) %>%
  gg_tsresiduals()
}
```

---

guerrero   *Guerrero's method for Box Cox lambda selection*

Description

Applies Guerrero's (1993) method to select the lambda which minimises the coefficient of variation for subseries of x.

Usage

```
guerrero(x, lower = -0.9, upper = 2, .period = 2L)
```

Arguments

x	A numeric vector. The data used to identify the transformation parameter lambda.
lower	The lower bound for lambda.
upper	The upper bound for lambda.
.period	The length of each subseries (usually the length of seasonal period). Subseries length must be at least 2.

Details

Note that this function will give slightly different results to `forecast::BoxCox.lambda(y)` if your data does not start at the start of the seasonal period. This function will make use of all of your data, whereas the forecast package will not use data that doesn't complete a seasonal period.

Value

A Box Cox transformation parameter (lambda) chosen by Guerrero's method.

References

Box, G. E. P. and Cox, D. R. (1964) An analysis of transformations. JRSS B 26 211–246.

Guerrero, V.M. (1993) Time-series analysis supported by power transformations. Journal of Forecasting, 12, 37–48.

---

ljung_box   *Portmanteau tests*

Description

Compute the Box–Pierce or Ljung–Box test statistic for examining the null hypothesis of independence in a given time series. These are sometimes known as 'portmanteau' tests.

Usage

```
ljung_box(x, lag = 1, dof = 0, ...)

box_pierce(x, lag = 1, dof = 0, ...)

portmanteau_tests
```

Arguments

x	A numeric vector
lag	The number of lag autocorrelation coefficients to use in calculating the statistic
dof	Degrees of freedom of the fitted model (useful if x is a series of residuals).
...	Unused.

Format

An object of class list of length 2.

Value

A vector of numeric features for the test's statistic and p-value.

See Also

stats::Box.test()

Examples

```
ljung_box(rnorm(100))

box_pierce(rnorm(100))
```

---

longest_flat_spot *Longest flat spot length*

Description

"Flat spots" are computed by dividing the sample space of a time series into ten equal-sized intervals, and computing the maximum run length within any single interval.

Usage

```
longest_flat_spot(x)
```

Arguments

x   a vector

Value

A numeric value.

Author(s)

Earo Wang and Rob J Hyndman

---

n_crossing_points *Number of crossing points*

Description

Computes the number of times a time series crosses the median.

Usage

```
n_crossing_points(x)
```

Arguments

x   a univariate time series

Value

A numeric value.

Author(s)

Earo Wang and Rob J Hyndman

---

shift_level_max   *Sliding window features*

Description

Computes feature of a time series based on sliding (overlapping) windows. shift_level_max finds the largest mean shift between two consecutive windows. shift_var_max finds the largest var shift between two consecutive windows. shift_kl_max finds the largest shift in Kulback-Leibler divergence between two consecutive windows.

Usage

```
shift_level_max(x, .size = NULL, .period = 1)

shift_var_max(x, .size = NULL, .period = 1)

shift_kl_max(x, .size = NULL, .period = 1)
```

Arguments

x	a univariate time series
.size	size of sliding window, if NULL .size will be automatically chosen using .period
.period	The seasonal period (optional)

Details

Computes the largest level shift and largest variance shift in sliding mean calculations

Value

A vector of 2 values: the size of the shift, and the time index of the shift.

Author(s)

Earo Wang, Rob J Hyndman and Mitchell O'Hara-Wild

---

stat_arch_lm   *ARCH LM Statistic*

Description

Computes a statistic based on the Lagrange Multiplier (LM) test of Engle (1982) for autoregressive conditional heteroscedasticity (ARCH). The statistic returned is the R^2 value of an autoregressive model of order lags applied to x^2.

Usage

```
stat_arch_lm(x, lags = 12, demean = TRUE)
```

Arguments

x	a univariate time series
lags	Number of lags to use in the test
demean	Should data have mean removed before test applied?

Value

A numeric value.

Author(s)

Yanfei Kang

STL   *Multiple seasonal decomposition by Loess*

Description

Decompose a time series into seasonal, trend and remainder components. Seasonal components are estimated iteratively using STL. Multiple seasonal periods are allowed. The trend component is computed for the last iteration of STL. Non-seasonal time series are decomposed into trend and remainder only. In this case, supsmu is used to estimate the trend. Optionally, the time series may be Box-Cox transformed before decomposition. Unlike stl, mstl is completely automated.

Usage

```
STL(formula, iterations = 2, ...)
```

Arguments

formula	Decomposition specification (see "Specials" section).
iterations	Number of iterations to use to refine the seasonal component.
...	Other arguments passed to [stats::stl()]().

Value

A [fabletools::dable()]() containing the decomposed trend, seasonality and remainder from the STL decomposition.

Specials

trend: The trend special is used to specify the trend extraction parameters.

```
trend(window, degree, jump)
```

window   The span (in lags) of the loess window, which should be odd. If NULL, the default, nextodd(ceiling((1.5*period) / (
degree   The degree of locally-fitted polynomial. Should be zero or one.
jump   Integers at least one to increase speed of the respective smoother. Linear interpolation happens between every jumpt

season: The season special is used to specify the season extraction parameters.

```
season(period = NULL, window = NULL, degree, jump)
```

period   The periodic nature of the seasonality. This can be either a number indicating the number of observations in each se
window   The span (in lags) of the loess window, which should be odd. If the window is set to "periodic" or Inf, the seasona
degree   The degree of locally-fitted polynomial. Should be zero or one.
jump   Integers at least one to increase speed of the respective smoother. Linear interpolation happens between every jumpt

lowpass: The lowpass special is used to specify the low-pass filter parameters.

```
lowpass(window, degree, jump)
```

window   The span (in lags) of the loess window of the low-pass filter used for each subseries. Defaults to the smallest odd int
degree   The degree of locally-fitted polynomial. Must be zero or one.
jump   Integers at least one to increase speed of the respective smoother. Linear interpolation happens between every jumpt

References

R. B. Cleveland, W. S. Cleveland, J.E. McRae, and I. Terpenning (1990) STL: A Seasonal-Trend Decomposition Procedure Based on Loess. Journal of Official Statistics, 6, 3–73.

See Also

[stl](), [supsmu]()

Examples

```
as_tsibble(USAccDeaths) %>%
```

```
    model(STL(value ~ trend(window = 10))) %>%
    components()
```

---

unitroot_kpss   *Unit root tests*

Description

Performs a test for the existence of a unit root in the vector.

Usage

```
unitroot_kpss(x, type = c("mu", "tau"), lags = c("short", "long", "nil"), ...)

unitroot_pp(
  x,
  type = c("Z-tau", "Z-alpha"),
  model = c("constant", "trend"),
  lags = c("short", "long"),
  ...
)
```

Arguments

x	A vector to be tested for the unit root.
type	Type of deterministic part.
lags	Maximum number of lags used for error term correction.
...	Arguments passed to unit root test function.
model	Determines the deterministic part in the test regression.

Details

unitroot_kpss computes the statistic for the Kwiatkowski et al. unit root test with linear trend and lag 1.

unitroot_pp computes the statistic for the Z-tau version of Phillips & Perron unit root test with constant trend and lag 1.

Value

A vector of numeric features for the test's statistic and p-value.

See Also

urca::ur.kpss()
urca::ur.pp()

---

unitroot_ndiffs   *Number of differences required for a stationary series*

Description

Use a unit root function to determine the minimum number of differences necessary to obtain a stationary time series.

Usage

```
unitroot_ndiffs(
  x,
  alpha = 0.05,
  unitroot_fn = ~unitroot_kpss(.)["kpss_pvalue"],
  differences = 0:2,
  ...
)

unitroot_nsdiffs(
  x,
  alpha = 0.05,
  unitroot_fn = ~feat_stl(., .period)[2] < 0.64,
  differences = 0:2,
  .period = 1,
  ...
)
```

Arguments

x	A vector to be tested for the unit root.
alpha	The level of the test.
unitroot_fn	A function (or lambda) that provides a p-value for a unit root test.
differences	The possible differences to consider.
...	Additional arguments passed to the unitroot_fn function
.period	The period of the seasonality.

Details

Note that the default 'unit root function' for unitroot_nsdiffs() is based on the seasonal strength of an STL decomposition. This is not a test for the presence of a seasonal unit root, but generally works reasonably well in identifying the presence of seasonality and the need for a seasonal difference.

Value

A numeric corresponding to the minimum required differences for stationarity.

---

var_tiled_var *Time series features based on tiled windows*

Description

Computes feature of a time series based on tiled (non-overlapping) windows. Means or variances are produced for all tiled windows. Then stability is the variance of the means, while lumpiness is the variance of the variances.

Usage

```
var_tiled_var(x, .size = NULL, .period = 1)

var_tiled_mean(x, .size = NULL, .period = 1)
```

Arguments

x	a univariate time series
.size	size of sliding window, if NULL .size will be automatically chosen using .period
.period	The seasonal period (optional)

Value

A numeric vector of length 2 containing a measure of lumpiness and a measure of stability.

Author(s)

Earo Wang and Rob J Hyndman

X_13ARIMA_SEATS *X-13ARIMA-SEATS Seasonal Adjustment*

Description

X-13ARIMA-SEATS is a seasonal adjustment program developed and maintained by the U.S. Census Bureau.

Usage

```
X_13ARIMA_SEATS(
  formula,
  ...,
  na.action = seasonal::na.x13,
  defaults = c("seasonal", "none")
)
```

Arguments

formula	Decomposition specification.
...	Other arguments passed to seasonal::seas().
na.action	a function which indicates what should happen when the data contain NAs. na.omit (default), na.exclude or na.fail. If na.action = na.x13, NA handling is done by X-13, i.e. NA values are substituted by -99999.
defaults	If defaults="seasonal", the default options of seasonal::seas() will be used, which should work well in most circumstances. Setting defaults="none" gives an empty model specification, which can be added to in the model formula.

Details

The SEATS decomposition method stands for "Seasonal Extraction in ARIMA Time Series", and is the default method for seasonally adjusting the data. This decomposition method can extract seasonality from data with seasonal periods of 2 (biannual), 4 (quarterly), 6 (bimonthly), and 12 (monthly). This method is specified using the seats() function in the model formula.

Alternatively, the seasonal adjustment can be done using an enhanced X-11 decomposition method. The X-11 method uses weighted averages over a moving window of the time series. This is used in combination with the RegARIMA model to prepare the data for decomposition. To use the X-11 decomposition method, the x11() function can be used in the model formula.

Specials

The specials of the X-13ARIMA-SEATS model closely follow the individual specification options of the original function. Refer to Chapter 7 of the X-13ARIMA-SEATS Reference Manual for full details of the arguments.

The available specials for this model are:

#'

arima: The arima special is used to specify the ARIMA part of the regARIMA model. This defines a pure ARIMA model if the regression() special absent and if no exogenous regressors are specified. The lags of the ARIMA model can be specified in the model argument, potentially along with ar and ma coefficients.

arima(...)

> ...   Arguments described in the reference manual linked below.

automdl: The automdl special is used to specify the ARIMA part of the regARIMA model will be sought using an automatic model selection procedure derived from the one used by TRAMO (see Gomez and Maravall (2001a)). The maximum order of lags and differencing can be specified using maxorder and maxdiff arguments. Models containing mixtures of AR and MA components can be allowed or disallowed using the mixed argument.

automdl(...)

> ...   Arguments described in the reference manual linked below.

check: The check special is used to produce statistics for diagnostic checking of residuals from the estimated model. The computed statistics include ACF and PACF of residuals, along with some statistical tests. These calculations are included in the model object, but difficult to access. It is recommended that these checks are done in R after estimating the model, and that this special is not used.

```
check(...)
```

... Arguments described in the reference manual linked below.

estimate: The estimate special is used to specify optimisation parameters and estimation options for the regARIMA model specified by the regression() and arima() specials. Among other options, the tolerance can be set with tol, and maximum iterations can be set with maxiter.

```
estimate(...)
```

... Arguments described in the reference manual linked below.

force: The force is an optional special for invoking options that allow users to force yearly totals of the seasonally adjusted series to equal those of the original series for convenience.

```
force(...)
```

... Arguments described in the reference manual linked below.

forecast: The forecast special is used to specify options for forecasting and/or backcasting the time series using the estimated model. This process is used to enhance the decomposition procedure, especially its performance at the start and end of the series. The number of forecasts to produce is specified in the maxlead argument, and the number of backcasts in the maxback argument.

```
forecast(...)
```

... Arguments described in the reference manual linked below.

history: The history special is an optional special for requesting a sequence of runs from a sequence of truncated versions of the time series. Using this special can substantially slow down the program.

```
history(...)
```

... Arguments described in the reference manual linked below.

metadata: The metadata special is used to insert metadata into the diagnostic summary file. This is typically not needed when interacting with the program via R.

```
metadata(...)
```

... Arguments described in the reference manual linked below.

identify: The `identify` special is used to produce tables and line printer plots of sample ACFs and PACFs for identifying the ARIMA part of a regARIMA model.

`identify(...)`

          ...   Arguments described in the reference manual linked below.

outlier: The `outlier` special is used to perform automatic detection of additive (point) outliers, temporary change outliers, level shifts, or any combination of the three using the specified model. The `seasonal::seas()` defaults used when `defaults="seasonal"` will include the default automatic detection of outliers.

`outlier(...)`

          ...   Arguments described in the reference manual linked below.

pickmdl: The `pickmdl` special is used to specify the ARIMA part of the regARIMA model will be sought using an automatic model selectionprocedure similar to the one used by X-11-ARIMA/88 (see Dagum 1988).

`pickmdl(...)`

          ...   Arguments described in the reference manual linked below.

regression:
The `regression` special is used to specify including regression variables in a regARIMA model, or for specifying regression variables whose effects are to be removed by the `identify()` special to aid ARIMA model identification. Any exogenous regressors specified in the model formula will be passed into this specification via the `user` and `data` arguments. The [seasonal::seas()](#) defaults used when `defaults="seasonal"` will set `aictest = c("td", "easter")`, indicating that trading days and Easter effects will be included conditional on AIC-based selection methods.

`regression(...)`

          ...   Arguments described in the reference manual linked below.

seats: The `seats` special is optionally used to invoke the production of model based signal extraction using SEATS, a seasonal adjustment program developed by Victor Gomez and Agustin Maravall at the Bank of Spain.

`seats(...)`

          ...   Arguments described in the reference manual linked below.

slidingspans: The optional `slidingspans` special is to provide sliding spans stability analysis on the model. These compare different features of seasonal adjustment output from overlapping subspans of the time series data.

`slidingspans(...)`

... Arguments described in the reference manual linked below.

spectrum: The optional `spectrum` special is used to provide a choice between two spectrum diagnostics to detect seasonality or trading day effects in monthly series.

```
spectrum(...)
```

... Arguments described in the reference manual linked below.

transform: The `transform` special is used to transform or adjust the series prior to estimating a regARIMA model. This is comparable to transforming the response on the formula's left hand side, but offers X-13ARIMA-SEATS specific adjustment options.

```
transform(...)
```

... Arguments described in the reference manual linked below.

x11: The optional `x11` special is used to invoke seasonal adjustment by an enhanced version of the methodology of the Census Bureau X-11 and X-11Q programs. The user can control the type of seasonal adjustment decomposition calculated (`mode`), the seasonal and trend moving averages used (`seasonalma` and `trendma`), and the type of extreme value adjustment performed during seasonal adjustment (`sigmalim`).

```
x11(...)
```

... Arguments described in the reference manual linked below.

x11regression: The `x11regression` special is used in conjunction with the `x11()` special for series without missing observations. This special estimates calendar effects by regression modeling of the irregular component with predefined or user-defined regressors. Any exogenous regressors specified in the model formula will be passed into this specification via the `user` and `data` arguments.

```
x11regression(...)
```

... Arguments described in the reference manual linked below.

References

Gomez, Victor, and Agustin Maravall. "Automatic modeling methods for univariate series." A course in time series analysis (2001): 171-201.

Dagum, E.B. (1988), The X11 ARIMA/88 Seasonal Adjustment Method - Foundations And User's Manual, Time Series Research and Analysis Division Statistics Canada, Ottawa.

Dagum, E. B., & Bianconcini, S. (2016) "Seasonal adjustment methods and real time trend-cycle estimation". *Springer*.

X-13ARIMA-SEATS Documentation from the seasonal package's website: http://www.seasonal.website/seasonal.html

Official X-13ARIMA-SEATS manual: https://www2.census.gov/software/x-13arima-seats/

x13as/windows/documentation/docx13as.pdf

See Also

seasonal::seas()

Examples

```
fit <- tsibbledata::aus_production %>%
  model(X_13ARIMA_SEATS(Beer))

report(fit)
components(fit)

# Additive X-11 decomposition
fit <- tsibbledata::aus_production %>%
  model(X_13ARIMA_SEATS(Beer ~ transform(`function` = "none") + x11(mode = "add")))

report(fit)
components(fit)
```

Index