

# Package ‘feasts’

January 17, 2020

**Title** Feature Extraction And Statistics for Time Series

**Version** 0.1.2

**Description** Provides a collection of features, decompositions, 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 ( $\geq 3.5.0$ ), fabletools

**Imports** rlang ( $\geq 0.2.0$ ), tibble ( $\geq 1.4.1$ ), tsibble ( $\geq 0.8.0$ ), ggplot2 ( $\geq 3.0.0$ ), dplyr ( $\geq 0.8.0$ ), tidyr ( $\geq 0.8.3$ ), scales, lubridate, grid

**Suggests** tsibbledata, pillar ( $\geq 1.0.1$ ), knitr, rmarkdown, testthat, covr, seasonal, urca, fracdiff, fable, ForeCA

**ByteCompile** true

**VignetteBuilder** knitr

**License** GPL-3

**URL** <http://feasts.tidyverts.org/>

**BugReports** <https://github.com/tidyverts/feasts/issues>

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.0.2

**Language** en-GB

**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](mailto:mail@mitchelloharawild.com)>

**Repository** CRAN

**Date/Publication** 2020-01-17 09:40:22 UTC

**R topics documented:**

feasts-package . . . . .	2
ACF . . . . .	3
autoplot.tbl_cf . . . . .	5
classical_decomposition . . . . .	5
coef_hurst . . . . .	6
feat_acf . . . . .	7
feat_pacf . . . . .	8
feat_spectral . . . . .	8
feat_stl . . . . .	9
gg_arma . . . . .	10
gg_lag . . . . .	11
gg_season . . . . .	12
gg_subseries . . . . .	13
gg_tsdisplay . . . . .	14
gg_tsresiduals . . . . .	15
guerrero . . . . .	16
ljung_box . . . . .	17
n_crossing_points . . . . .	18
n_flat_spots . . . . .	18
scale_cf_lag . . . . .	19
shift_level_max . . . . .	19
stat_arch_lm . . . . .	20
STL . . . . .	21
unitroot_kpss . . . . .	22
unitroot_ndiffs . . . . .	23
var_tiled_var . . . . .	24
<b>Index</b>	<b>25</b>

feasts-package

*feasts: Feature Extraction And Statistics for Time Series***Description**

Provides a collection of features, decompositions, 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/>
- 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,
  ...,
  lag_max = NULL,
  demean = TRUE,
  type = c("correlation", "covariance", "partial")
)
```

```
PACF(.data, ..., lag_max = NULL)
```

```
CCF(.data, ..., lag_max = NULL, type = c("correlation", "covariance"))
```

### Arguments

<code>.data</code>	A tsibble
<code>...</code>	The column(s) from the tsibble used to compute the ACF, PACF or CCF.
<code>lag_max</code>	maximum lag at which to calculate the acf. Default is $10 * \log_{10}(N/m)$ 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.
<code>demean</code>	logical. Should the covariances be about the sample means?
<code>type</code>	character string giving the type of acf to be computed. Allowed values are "correlation" (the default), "covariance" or "partial". Will be partially matched.

### Details

The functions improve the `acf`, `pacf` and `ccf` functions. The main differences are that ACF does not plot a spike at lag 0 when `type=="correlation"` (which is redundant) 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 tibble 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

[acf](#), [pacf](#), [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	<i>Auto- and Cross- Covariance and -Correlation plots</i>
-----------------	---

---

**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 <a href="#">ACF()</a> , <a href="#">PACF()</a> , or <a href="#">CCF()</a> ).
level	The level of confidence for the blue dashed lines.
...	Unused.

**Value**

A ggplot object showing the correlations.

---

classical_decomposition	<i>Classical Seasonal Decomposition by Moving Averages</i>
-------------------------	--

---

**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 <code>\link[stats]{decompose}</code> .

**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 season.

**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	<i>Autocorrelation-based features</i>
----------	---------------------------------------

---

**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  $\max(\text{.period}, 10L)$  for `feat_acf`, and  $\max(\text{.period}, 5L)$  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_pacf	<i>Partial autocorrelation-based features</i>
-----------	---

---

**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 <code>stats::acf()</code> or <code>stats::pacf()</code>

**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	<i>Spectral features of a time series</i>
---------------	---

---

**Description**

Computes the spectral entropy of a time series

**Usage**

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

**Arguments**

x	a univariate time series
.period	The seasonal period.
...	Further arguments for <code>ForeCA::spectral_entropy()</code>



**Value**

A numeric value.

**Author(s)**

Rob J Hyndman

---

feat_stl	<i>STL features</i>
----------	---------------------

---

**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 = 13, ...)
```

**Arguments**

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

**Value**

A vector of numeric features from a STL decomposition.

**See Also**

[Forecasting Principle and Practices: Measuring strength of trend and seasonality](#)

---

`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

<code>data</code>	A tidy time series object (tsibble)
<code>y</code>	The variable to plot (a bare expression). If NULL, it will automatically selected from the data.
<code>period</code>	The seasonal period to display.
<code>lags</code>	A vector of lags to display as facets.
<code>geom</code>	The geometry used to display the data.
<code>arrow</code>	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 <code>grid::arrow()</code> can be used.
<code>...</code>	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 = 15,
  polar = FALSE,
  labels = c("none", "left", "right", "both"),
  ...
)
```

## Arguments

<code>data</code>	A tidy time series object (tsibble)
<code>y</code>	The variable to plot (a bare expression). If NULL, it will automatically selected from the data.
<code>period</code>	The seasonal period to display.
<code>facet_period</code>	A secondary seasonal period to facet by (typically smaller than period).
<code>max_col</code>	The maximum number of colours to display on the plot. If the number of seasonal periods in the data is larger than <code>max_col</code> , the plot will not include a colour. Use <code>max_col = 0</code> to never colour the lines, or <code>Inf</code> to always colour the lines. If labels are used, then <code>max_col</code> will be ignored.
<code>polar</code>	If TRUE, the season plot will be shown on polar coordinates.
<code>labels</code>	Position of the labels for seasonal period identifier.
<code>...</code>	Additional arguments passed to <code>geom_line()</code>

## 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.org/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 <code>geom_line()</code>

## 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.org/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 $10 * \log_{10}(N/m)$ 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.org/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, ...)
```

**Arguments**

`data` A mable containing one model with residuals.  
`...` 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.org/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 = -1, 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.

**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**

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

---

n_crossing_points	<i>Number of crossing points</i>
-------------------	----------------------------------

---

**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

---

n_flat_spots	<i>Number of flat spots</i>
--------------	-----------------------------

---

**Description**

Number of flat spots in a time series

**Usage**

```
n_flat_spots(x)
```

**Arguments**

x                    a vector

**Value**

A numeric value.

**Author(s)**

Earo Wang and Rob J Hyndman

---

scale_cf_lag	<i>lagged datetime scales This set of scales defines new scales for lagged time structures.</i>
--------------	---

---

**Description**

lagged datetime scales This set of scales defines new scales for lagged time structures.

**Usage**

```
scale_x_cf_lag(...)
```

**Arguments**

... Further arguments to be passed on to scale\_x\_continuous()

**Value**

A ggproto object inheriting from Scale

---

shift_level_max	<i>Sliding window features</i>
-----------------	--------------------------------

---

**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**

<code>x</code>	a univariate time series
<code>.size</code>	size of sliding window, if NULL <code>.size</code> will be automatically chosen using <code>.period</code>
<code>.period</code>	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

**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**

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

**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)
```

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

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

```
season(period = NULL, window = 13, degree, jump)
```

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

**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 integer greater than or equal to  $1.35 \times \sqrt{n}$ .  
 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 jump.

## 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	<i>Number of differences required for a stationary series</i>
-----------------	---

---

**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.

**Value**

A numeric corresponding to the minimum required differences for stationarity.

---

var_tiled_var	<i>Time series features based on tiled windows</i>
---------------	--

---

**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



# Index

## \*Topic **datasets**

ljung\_box, [17](#)

## \*Topic **package**

feasts-package, [2](#)

ACF, [3](#), [15](#)

acf, [4](#)

ACF(), [5](#)

autoplot.tbl\_cf, [5](#)

autoplot.tbl\_cf(), [4](#)

box\_pierce (ljung\_box), [17](#)

CCF (ACF), [3](#)

ccf, [4](#)

CCF(), [5](#)

classical\_decomposition, [5](#)

coef\_hurst, [6](#)

fabletools::dable(), [6](#), [21](#)

feasts (feasts-package), [2](#)

feasts-package, [2](#)

feat\_acf, [7](#)

feat\_pacf, [8](#)

feat\_spectral, [8](#)

feat\_stl, [9](#)

ForeCA::spectral\_entropy(), [8](#)

gg\_arma, [10](#)

gg\_lag, [11](#)

gg\_season, [12](#)

gg\_season(), [13](#)

gg\_subseries, [13](#)

gg\_tsdisplay, [14](#)

gg\_tsdisplay(), [15](#), [16](#)

gg\_tsresiduals, [15](#)

grid::arrow(), [11](#)

guerrero, [16](#)

ljung\_box, [17](#)

n\_crossing\_points, [18](#)

n\_flat\_spots, [18](#)

PACF (ACF), [3](#)

pacf, [4](#)

PACF(), [5](#)

plot.ts, [15](#)

portmanteau\_tests (ljung\_box), [17](#)

scale\_cf\_lag, [19](#)

scale\_x\_cf\_lag (scale\_cf\_lag), [19](#)

shift\_kl\_max (shift\_level\_max), [19](#)

shift\_level\_max, [19](#)

shift\_var\_max (shift\_level\_max), [19](#)

spec.ar, [15](#)

stat\_arch\_lm, [20](#)

stats::acf(), [7](#), [8](#)

stats::Box.test(), [17](#)

stats::pacf(), [7](#), [8](#)

stats::stl(), [9](#), [21](#)

STL, [21](#)

stl, [21](#), [22](#)

supsmu, [21](#), [22](#)

unitroot\_kpss, [22](#)

unitroot\_ndiffs, [23](#)

unitroot\_nsdiffs (unitroot\_ndiffs), [23](#)

unitroot\_pp (unitroot\_kpss), [22](#)

urca::ur.kpss(), [23](#)

urca::ur.pp(), [23](#)

var\_tiled\_mean (var\_tiled\_var), [24](#)

var\_tiled\_var, [24](#)