Package 'stlplus'

October 14, 2022

Type Package

Title Enhanced Seasonal Decomposition of Time Series by Loess

Version 0.5.1

Date 2016-01-05

Maintainer Ryan Hafen <rhafen@gmail.com>

Description Decompose a time series into seasonal, trend, and remainder components using an implementation of Seasonal Decomposition of Time Series by Loess (STL) that provides several enhancements over the STL method in the stats package. These enhancements include handling missing values, providing higher order (quadratic) loess smoothing with automated parameter choices, frequency component smoothing beyond the seasonal and trend components, and some basic plot methods for diagnostics.

LazyLoad yes

LazyData yes

NeedsCompilation yes

Imports lattice, yaImpute, stats, Rcpp

Suggests testthat

LinkingTo Rcpp

License BSD_3_clause + file LICENSE

URL https://github.com/hafen/stlplus

BugReports https://github.com/hafen/stlplus/issues

Note This is experimental software distributed free of charge and comes with no warranty! Exercise professional caution.

RoxygenNote 5.0.1

Author Ryan Hafen [aut, cre]

Repository CRAN

Date/Publication 2016-01-06 10:42:19

R topics documented:

`plot.stlplus` *Lattice plot of the raw, seasonal, trend, and remainder components*

Description

Lattice plot of the raw, seasonal, trend, and remainder components. If post-trend smoothing was done, these components will be plotted instead of the trend component.

Usage

```
## S3 method for class 'stlplus'
plot(x, scales = list(y = list(relation = "sliced")),
  type = "l", as.table = TRUE, strip = FALSE, strip.left = TRUE,
  between = list(y = 0.5), layout = NULL, ...)
```

Arguments

`x`	object of class `"stlplus"`.
`scales, type, as.table, strip, strip.left, between, layout, ...`	parameters to be passed to xyplot.

Value

object of class `"trellis"`.

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

`stlplus`

plot_cycle *Cycle-Subseries Plot for an stlplus Object*

Description

Plots the seasonal component by cycle-subseries, with lines emanating from the midmean of the values within each cycle-subseries.

Usage

```
plot_cycle(x, layout = c(x$pars$n.p, 1), col = "#0080ff", xlab = "Time",
  ylab = "Seasonal", panel = function(x, y, ...) {
  lattice::panel.segments(x, rep(.midmean(y), length(x)), x, y, col = col) },
  ...)
```

Arguments

`x`	object of class `"stlplus"`.
`layout, col, xlab, ylab, panel, ...`	parameters to be passed to `xyplot()`.

Value

object of class `"trellis"`.

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

`stlplus`

plot_rembycycle *Plot of Remainder Component by Cycle-Subseries*

Description

Plots the remainder component by cycle-subseries with a loess line.

Usage

```
plot_rembycycle(x, col = "darkgray", locol = "black", lolwd = 2,
  xlab = "Time", ylab = "Remainder", ...)
```

Arguments

`x`	object of class `"stlplus"`.
`col, locol, lolwd, xlab, ylab, ...`	parameters to be passed to `xyplot()`. `locol` and `lolwd` are the line color and width for the loess line.

Value

object of class `"trellis"`.

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

`stlplus`

`plot_seasonal` *Seasonal Diagnostic Plot for an stlplus Object*

Description

Plots each cycle-subseries of the detrended data (or equivalently, seasonal plus remainder), with the mean subtracted. The fitted seasonal component is plotted as a line through the points.

Usage

```
plot_seasonal(x, col = c("darkgray", "black"), lwd = 2, xlab = "Time",
  ylab = "Centered Seasonal + Remainder", ...)
```

Arguments

`x`	object of class `"stlplus"`.
`col, lwd, xlab, ylab, ...`	parameters to be passed to `xyplot()`.

Details

Helps decide how much of the variation in the data other than the trend should go into the seasonal component, and how much in the remainder.

Value

object of class `"trellis"`.

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

stlplus

`plot_trend` *Trend Diagnostic Plot for an stlplus Object*

Description

Plots the trend+remainder with the trend component overlaid, and the remainder component in a separate panel.

Usage

```
plot_trend(x, xlab = "Time", ylab = "Trend", span = 0.3, type = c("p",
  "l"), scales = list(y = list(relation = "free")), lwd = c(1, 1),
  col = c("darkgray", "black", "darkgray"), layout = c(1, 2),
  between = list(y = 0.5), strip = FALSE, strip.left = TRUE,
  as.table = TRUE, ...)
```

Arguments

`x`	object of class `"stlplus"`.
`xlab, ylab, span, type, scales, lwd, col, layout`	parameters to be passed to xyplot.
`between, strip, strip.left, as.table, ...`	parameters to be passed to xyplot.

Value

object of class `"trellis"`.

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

stlplus

seasonal *Accessor functions for elements of an stl and stlplus object*

Description

Retrieves the raw, seasonal, trend, remainder, or time components from an stlplus object. The methods `seasonal.stl`, ... also exist as a convenience for extracting components from R's `stl()`.

Usage

```
seasonal(object)

trend(object)

remainder(object)

getraw(object)

## S3 method for class 'stlplus'
remainder(object)

## S3 method for class 'stlplus'
fitted(object, ...)

## S3 method for class 'stlplus'
predict(object, ...)

## S3 method for class 'stlplus'
seasonal(object)

## S3 method for class 'stlplus'
trend(object)

fc(object, fcnum = 1)

## S3 method for class 'stlplus'
time(x, ...)

## S3 method for class 'stl'
remainder(object)

## S3 method for class 'stl'
seasonal(object)

## S3 method for class 'stl'
trend(object)
```

```
## S3 method for class 'stl'
time(x, ...)

## S3 method for class 'stl'
predict(object, ...)

## S3 method for class 'stl'
fitted(object, ...)
```

Arguments

fcnum	number of post-trend smoothing frequency component.
x, object	object of class "stl" or "stlplus".
...	additional parameters

Value

Returns a vector of either the getraw time series, the seasonal, trend, or remainder components, or the time values of the time series. If times are requested but were not supplied in the initial stlplus call, the 1:n vector is returned, where n is the number of data points. The fitted method returns the sum of the seasonal and trend.

Note

The fitted and predict methods are equivalent. For objects of class "stlplus", these functions return the sum of all components but the remainder, including post-trend smoothing components. Note also that the trend method for objects of class "stlplus" only returns the trend component from the STL iterations, even when post-trend smoothing is done.

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

stlplus

Examples

```
co2.stl <- stlplus(co2, t = as.vector(stats::time(co2)), n.p=12, l.window=13,
t.window=19, s.window=35, s.degree=1, sub.labels = substr(month.name, 1, 3))

plot(seasonal(co2.stl))
```

stlplus *Seasonal Decomposition of Time Series by Loess*

Description

Decompose a time series into seasonal, trend and irregular components using `loess`, acronym STL. A new implementation of STL. Allows for NA values, local quadratic smoothing, post-trend smoothing, and endpoint blending. The usage is very similar to that of R's built-in `stl()`.

Usage

```
stlplus(x, t = NULL, n.p, s.window, s.degree = 1, t.window = NULL,
  t.degree = 1, fc.window = NULL, fc.degree = NULL, fc.name = NULL,
  l.window = NULL, l.degree = t.degree, s.jump = ceiling(s.window/10),
  t.jump = ceiling(t.window/10), l.jump = ceiling(l.window/10),
  fc.jump = NULL, critfreq = 0.05, s.blend = 0, t.blend = 0,
  l.blend = t.blend, fc.blend = NULL, inner = 2, outer = 1,
  sub.labels = NULL, sub.start = 1, zero.weight = 0.000001,
  details = FALSE, ...)

## S3 method for class 'ts'
stlplus(x, t = as.numeric(stats::time(x)), n.p = frequency(x),
  s.window, s.degree = 1, t.window = NULL, t.degree = 1,
  fc.window = NULL, fc.degree = NULL, fc.name = NULL, l.window = NULL,
  l.degree = t.degree, s.jump = ceiling(s.window/10),
  t.jump = ceiling(t.window/10), l.jump = ceiling(l.window/10),
  fc.jump = NULL, critfreq = 0.05, s.blend = 0, t.blend = 0,
  l.blend = t.blend, fc.blend = NULL, inner = 2, outer = 1,
  sub.labels = NULL, sub.start = 1, zero.weight = 0.000001,
  details = FALSE, ...)

## S3 method for class 'zoo'
stlplus(...)
```

Arguments

`x`	vector of time series values, in order of time. If `x` is a time series object, then `t` and `n.p` do not need to be specified, although they still can be.
`t`	times at which the time series values were observed. Not required.
`n.p`	periodicity of the seasonal component. In R's `stl` function, this is the frequency of the time series.
`s.window`	either the character string `"periodic"` or the span (in lags) of the loess window for seasonal extraction, which should be odd. This has no default.
`s.degree`	degree of locally-fitted polynomial in seasonal extraction. Should be 0, 1, or 2.

`t.window` the span (in lags) of the loess window for trend extraction, which should be odd. If `NULL`, the default, `nextodd(ceiling((1.5*period) / (1-(1.5/s.window))))`, is taken.

`t.degree` degree of locally-fitted polynomial in trend extraction. Should be 0, 1, or 2.

`fc.window` vector of lengths of windows for loess smoothings for other trend frequency components after the original STL decomposition has been obtained. The smoothing is applied to the data with the STL seasonal component removed. A frequency component is computed by a loess fit with the window length equal to the first element of fc.window, the component is removed, another component is computed with the window length equal to the second element of fc.window, and so forth. In most cases, the values of the argument should be decreasing, that is, the frequency bands of the fitted components should increase. The robustness weights from original STL are used as weights in the loess fitting if specified.

`fc.degree` vector of degrees of locally-fitted polynomial in the loess smoothings for the frequency components specified in fc.window. Values of 0, 1 and 2 are allowed. If the length of fc.degree is less than that of fc.window, the former is expanded to the length of the latter using rep; thus, giving the value 1 specifies a degree of 1 for all components.

`fc.name` vector of names of the post-trend smoothing operations specified by `fc.window` and `fc.degree` (optional).

`l.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 `n.p` which is recommended since it prevents competition between the trend and seasonal components. If not an odd integer its given value is increased to the next odd one.

`l.degree` degree of locally-fitted polynomial for the subseries low-pass filter. Should be 0, 1, or 2.

`s.jump, t.jump, l.jump, fc.jump`
integers at least one to increase speed of the respective smoother. Linear interpolation happens between every `*.jump`th value.

`critfreq` the critical frequency to use for automatic calculation of smoothing windows for the trend and high-pass filter.

`s.blend, t.blend, l.blend, fc.blend`
vectors of proportion of blending to degree 0 polynomials at the endpoints of the series.

`inner` integer; the number of 'inner' (backfitting) iterations; usually very few (2) iterations suffice.

`outer` integer; the number of 'outer' robustness iterations. Default is 0, but Recommended if outliers are present.

`sub.labels` optional vector of length n.p that contains the labels of the subseries in their natural order (such as month name, day of week, etc.), used for strip labels when plotting. All entries must be unique.

`sub.start` which element of sub.labels does the series begin with. See details.

`zero.weight` value to use as zero for zero weighting

details	if TRUE, returns a list of the results of all the intermediate iterations.
...	additional parameters

Details

The seasonal component is found by *loess* smoothing the seasonal sub-series (the series of all January values, ...); if s.window = "periodic" smoothing is effectively replaced by taking the mean. The seasonal values are removed, and the remainder smoothed to find the trend. The overall level is removed from the seasonal component and added to the trend component. This process is iterated a few times. The remainder component is the residuals from the seasonal plus trend fit.

Cycle-subseries labels are useful for plotting and can be specified through the sub.labels argument. Here is an example for how the sub.labels and sub.start parameters might be set for one situation. Suppose we have a daily series with n.p=7 (fitting a day-of-week component). Here, sub.labels could be set to c("Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"). Now, if the series starts with a Wednesday value, then one would specify sub.labels=4, since Wednesday is the fourth element of sub.labels. This ensures that the labels in the plots to start the plotting with Sunday cycle-subseries instead of Wednesday.

Value

returns an object of class "stlplus", containing

data	data frame containing all of the components: raw, seasonal, trend, remainder, weights.
pars	list of parameters used in the procedure.
fc.number	number of post-trend frequency components fitted.
fc	data frame of the post-trend frequency components.
time	vector of time values corresponding to the raw values, if specified.
n	the number of observations.
sub.labels	the cycle-subseries labels.

Note

This is a complete re-implementation of the STL algorithm, with the loess part in C and the rest in R. Moving a lot of the code to R makes it easier to experiment with the method at a very minimal speed cost. Recoding in C instead of using R's built-in loess results in better performance, especially for larger series.

Author(s)

Ryan Hafen

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

plot.stlplus for plotting the components, getraw, seasonal, trend, remainder for accessing the components.

Examples

```
co2_stl <- stlplus(co2, t = as.vector(time(co2)), n.p = 12,
  l.window = 13, t.window = 19, s.window = 35, s.degree = 1,
  sub.labels = substr(month.name, 1, 3))

plot(co2_stl, ylab = "CO2 Concentration (ppm)", xlab = "Time (years)")
plot_seasonal(co2_stl)
plot_trend(co2_stl)
plot_cycle(co2_stl)
plot_rembycycle(co2_stl)

# post-trend smoothing

co2_stl_pt <- stlplus(co2, t = as.vector(time(co2)), n.p = 12,
  l.window = 13, t.window = 19, s.window = 35, s.degree = 1,
  sub.labels = substr(month.name, 1, 3),
  fc.degree = c(1, 2), fc.window = c(201, 35),
  fc.name = c("long-term", "so. osc."))

plot(co2_stl_pt, scales = list(y = list(relation = "free")),
  ylab = "CO2 Concentration (ppm)", xlab = "Time (years)",
  aspect = 0.25, type = c("l", "g"))

# with NAs

y <- co2
y[201:224] <- NA

y_stl <- stlplus(y, l.window = 13, t.window = 19, s.window = 35,
  s.degree = 1, sub.labels = substr(month.name, 1, 3))

plot(y_stl, ylab = "CO2 Concentration (ppm)", xlab = "Time (years)", type = c("l", "g"))
plot_seasonal(y_stl)
plot_trend(y_stl)
plot_cycle(y_stl)
plot_rembycycle(y_stl)

# if you don't want to use a time series object:
y_stl <- stlplus(y, t = as.vector(time(y)), n.p = 12,
  l.window = 13, t.window = 19, s.window = 35, s.degree = 1,
  sub.labels = substr(month.name, 1, 3))

# with an outlier
y2 <- co2
y2[200] <- 300

y2_stl <- stlplus(y2, t = as.vector(time(y2)), n.p = 12,
```

```
  l.window = 13, t.window = 19, s.window = 35, s.degree = 1,
  sub.labels = substr(month.name, 1, 3), outer = 10)

plot(y2_stl, ylab = "CO2 Concentration (ppm)", xlab = "Time (years)")
plot_seasonal(y2_stl)
plot_trend(y2_stl)
plot_cycle(y2_stl)
plot_rembycycle(y2_stl)

# compare to R's stl

x1 <- stlplus(co2, t = as.vector(time(co2)), n.p = 12,
  l.window = 13, t.window = 19, s.window = 11, s.degree = 1,
  sub.labels = substr(month.name, 1, 3))

x2 <- stl(co2, l.window = 13, t.window = 19, s.window = 11, s.degree = 1)

# will be different due to interpolation differences
plot(seasonal(x1) - seasonal(x2))

# but not if all jump parameters are 1
x1 <- stlplus(co2, t = as.vector(time(co2)), n.p = 12,
  l.window = 13, t.window = 19, s.window = 11, s.degree = 1,
  sub.labels = substr(month.name, 1, 3),
  s.jump = 1, t.jump = 1, l.jump = 1)

x2 <- stl(co2, l.window = 13, t.window = 19, s.window = 11, s.degree = 1,
  s.jump = 1, t.jump = 1, l.jump = 1)

plot(seasonal(x1) - seasonal(x2))
```

Index