stl {stats} R Documentation

Seasonal Decomposition of Time Series by Loess

Description

Decompose a time series into seasonal, trend and irregular components using `loess`, acronym STL.

Usage

```
stl(x, s.window, s.degree = 0,
    t.window = NULL, t.degree = 1,
    l.window = nextodd(period), l.degree = t.degree,
    s.jump = ceiling(s.window/10),
    t.jump = ceiling(t.window/10),
    l.jump = ceiling(l.window/10),
    robust = FALSE,
    inner = if(robust)  1 else 2,
    outer = if(robust) 15 else 0,
    na.action = na.fail)
```

Arguments

`x`
: univariate time series to be decomposed. This should be an object of class `"ts"` with a frequency greater than one.

`s.window`
: either the character string `"periodic"` or the span (in lags) of the loess window for seasonal extraction, which should be odd and at least 7, according to Cleveland et al. This has no default.

`s.degree`
: degree of locally-fitted polynomial in seasonal extraction. Should be zero or one.

`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 zero or one.

`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 `frequency(x)` 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. Must be 0 or 1.

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

`robust`
: logical indicating if robust fitting be used in the `loess` procedure.

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

`outer`
: integer; the number of ‘outer’ robustness iterations.

`na.action`
: action on missing values.

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.

Several methods for the resulting class `"stl"` objects, see, `plot.stl`.

Value

`stl` returns an object of class `"stl"` with components

`time.series`
: a multiple time series with columns `seasonal`, `trend` and `remainder`.

`weights`	the final robust weights (all one if fitting is not done robustly).
`call`	the matched call.
`win`	integer (length 3 vector) with the spans used for the `"s"`, `"t"`, and `"l"` smoothers.
`deg`	integer (length 3) vector with the polynomial degrees for these smoothers.
`jump`	integer (length 3) vector with the ‘jumps’ (skips) used for these smoothers.
`ni`	number of **i**nner iterations
`no`	number of **o**uter robustness iterations

Note

This is similar to but not identical to the `stl` function in S-PLUS. The `remainder` component given by S-PLUS is the sum of the `trend` and `remainder` series from this function.

Author(s)

B.D. Ripley; Fortran code by Cleveland *et al* (1990) from ‘`netlib`’.

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.stl` for `stl` methods; `loess` in package **stats** (which is not actually used in `stl`).

`StructTS` for different kind of decomposition.

Examples

```
require(graphics)

plot(stl(nottem, "per"))
plot(stl(nottem, s.window = 7, t.window = 50, t.jump = 1))

plot(stllc <- stl(log(co2), s.window = 21))
summary(stllc)
## linear trend, strict period.
plot(stl(log(co2), s.window = "per", t.window = 1000))

## Two STL plotted side by side :
        stmd <- stl(mdeaths, s.window = "per") # non-robust
summary(stmR <- stl(mdeaths, s.window = "per", robust = TRUE))
op <- par(mar = c(0, 4, 0, 3), oma = c(5, 0, 4, 0), mfcol = c(4, 2))
plot(stmd, set.pars = NULL, labels  =  NULL,
     main = "stl(mdeaths, s.w = \"per\",  robust = FALSE / TRUE )")
plot(stmR, set.pars = NULL)
# mark the 'outliers' :
(iO <- which(stmR $ weights  < 1e-8)) # 10 were considered outliers
sts <- stmR$time.series
points(time(sts)[iO], 0.8* sts[,"remainder"][iO], pch = 4, col = "red")
par(op)   # reset
```

[Package *stats* version 3.4.0 Index]