Temporal Disaggregation of Time Series

Description

Perform temporal disaggregation or interpolation of low frequency to high frequency time series. `td` can be used with objects of class `"ts"` as well as with basic vectors.

Usage

```
td(formula, conversion = "sum", to = "quarterly",
  method = "chow-lin-maxlog", truncated.rho = 0, fixed.rho = 0.5,
  criterion = "proportional", h = 1, start = NULL, end = NULL, ...)
```

Arguments

`formula`	an object of class `"formula"`: a symbolic description of the the temporal disaggregation model. The details of model specification are given under 'Details'.
`conversion`	type of conversion: `"sum"`, `"average"`, `"first"` or `"last"`.
`to`	high-frequency destination frequency as a character string (`"quarterly"` or `"monthly"`) or as a scalar (e.g. `2`, `4`, `7`, `12`). If the input series are `ts` objects, the argument is necessary if no indicator is given. If the input series are vectors, `to` must be a scalar indicating the frequency ratio.
`method`	method of temporal disaggregation: `"chow-lin-maxlog"`, `"chow-lin-minrss-ecotrim"`, `"chow-lin-minrss-quilis"`, `"chow-lin-fixed"`, `"dynamic-maxlog"` (experimental), `"dynamic-minrss"` (experimental), `"dynamic-fixed"` (experimental), `"fernandez"`, `"litterman-maxlog"`, `"litterman-minrss"`, `"litterman-fixed"`, `"denton-cholette"`, `"denton"`, `"uniform"` or `"ols"`. See 'Details'.
`truncated.rho`	lower bound for the autoregressive parameter ρ. If set to `0` (default), no negative values are allowed. If set to `-1`, truncation is disabled.
`fixed.rho`	set a predefined autoregressive parameter ρ. Only works with the methods `"chow-lin-fixed"` and `"litterman-fixed"`.
`criterion`	minimzation criterion for Denton methods: `"proportional"` or `"additive"`. See 'Details'.
`h`	degree of differencing for Denton methods. See 'Details'.
`start`	(optional) start date. Similar to pre-processing the input series with `window`.
`end`	(optional) end date. Similar to pre-processing the input series with `window`.
`...`	additional arguments to be passed to the low level subfunctions.

Details

`td` is used to disaggregate or interpolate a low frequency to a higher frequency time series, while either the sum, the average, the first or the last value of the resulting high-frequency series is consistent with the low frequency series. Disaggregation can be performed with or without the help of one or more indicator series. It can deal with all situations where the high frequency is an integer multiple of the low frequency (e.g. weeks to days), but not with irregular frequencies (e.g. weeks to months).

If the high-frequency indicator(s) cover(s) a longer time span than the low-frequency series, an extrapolation or retropolation (Wei, 1994, p. 138) is performed, using the same model as for interpolation.

The selection of a temporal disaggregation model is similar to the selection of a linear regression model. Thus, `td` closely mirrors the working of the `lm` function. The left hand side of the `formula` denotes the low-frequency series, the right hand side the indicators. If no indicator is specified, the right hand side must be set equal to `1` (see examples).
Unlike `lm`, `td` handles `ts` and `mts` time-series objects, as a typical application involves the use of these objects. Alternatively, If used with basic vectors, the `to` argument specifies the ratio between the high and the low frequency series.

For the generalized least squares (GLS) methods `"chow-lin-maxlog"`, `"chow-lin-minrss-ecotrim"`, `"chow-lin-minrss-quilis"`,`"litterman-maxlog"` and `"litterman-minrss"`, an autoregressive parameter *ρ* is estimated. Default (and recommended) method is `chow-lin-maxlog`. With `truncated.rho = 0` (default), it produces good results for a wide range of applications.

There are two variants of the `chow-lin-minrss` approach that lead to different results: Ecotrim by Barcellan (2003) uses a correlation matrix instead of the variance covariance matrix (implemented in `"chow-lin-minrss-ecotrim"`), the Matlab library by Quilis (2009) multiplies the correlation matrix with *1/(1-ρ^2)* (implemented in `"chow-lin-minrss-quilis"`).

The methods `"dynamic-maxlog"`, `"dynamic-minrss"` and `"dynamic-fixed"` are dynamic extensions of Chow-Lin (Santos Silva and Cardoso, 2001). If the autoregressive parameter *ρ* is equal to 0, no truncation remainder is added.

The Denton methods `"denton"` and `"denton-cholette"` can be specified with one or without an indicator. The parameter `h` can be set equal to `0`, `1`, or `2`. Depending on the value, the `denton` procedure minimizes the sum of squares of the deviations between the levels (`0`), the first differences (`1`) or the second differences (`2`) of the indicator and the resulting series. Additionally, `criterion` can be set equal to `"proportional"` or `"additive"`, depending on whether the proportional or the absolute deviations should be considered for minimzation. `"denton-cholette"` removes the transient movement of the original `"denton"` method at the beginning of the resulting series.

`"uniform"` is a special case of the `"denton"` approach, with `h` equals `0` and `criterion` equals `"proportional"`. It distributes the residuals uniformly. If no indicator is used, this leads to a step-shaped series.

`"ols"` performs an ordinary least squares regression (OLS) and distributes the residuals uniformly. It is especially useful for comparing the estimators of GLS and OLS regressions.

Value

`td` returns an object of class `"td"`.

The function `predict` computes the interpolated high frequency series. If the high-frequency indicator series are longer than the low-frequency series, the resulting series will be extrapolated. The function `coefficients` extracts the coefficients. The function `residuals` extracts the low frequency residuals. The function `summary` prints a summary of the estimation.

An object of class `"td"` is a list containing the following components:

`values`	disaggregated or interpolated (and extrapolated) high frequency series
`fitted.values`	low frequency fitted values of the regression; low frequency indicator for the Denton methods.
`p`	preliminary high frequency series
`residuals`	low-frequency residuals
`rho`	autoregressive parameter, ρ
`truncated`	logical, whether ρ has been truncated
`coefficients`	a named vector of coefficients
`se`	standard errors of the coefficients
`s_2`	ML-estimator of the variance of the high-frequency residuals
`s_2_gls`	GLS-estimator of the variance of the high-frequency residuals
`tss`	weighted (low frequency) total sum of squares
`rss`	weighted (low frequency) residual sum of squares
`r.squared`	R squared
`adj.r.squared`	adjusted R squared
`logl`	log-likelihood
`aic`	Akaike information criterion
`bic`	Schwarz information criterion
`rank`	number of right hand variables (including intercept)
`df`	degrees of freedom
`method`	method of temporal disaggregation
`call`	function call
`name`	name of the low frequency variable
`fr`	the ratio of high to low-frequency series
`conversion`	type of temporal conversion
`actual`	actual values of the low frequeny series

`model`	a matrix containing the indicators (and a constant if present)
`criterion`	minimization criterion in Denton methods
`h`	order of differencing in Denton methods

References

Chow, G. C., & Lin, A. L. (1971). Best linear unbiased interpolation, distribution, and extrapolation of time series by related series. *The review of Economics and Statistics*, 372-375.

Denton, F. T. (1971). Adjustment of monthly or quarterly series to annual totals: an approach based on quadratic minimization. *Journal of the American Statistical Association*, 66(333), 99-102.

Santos Silva, J. M. C. & Cardoso, F. N. (2001). The Chow-Lin method using dynamic models. *Economomic Modelling*, 18, 269-280.

Wei, W. W. S. (1994). Time series analysis. Addison-Wesley publ.

Sax, C. und Steiner, P. (2013). Temporal Disaggregation of Time Series. *The R Journal*, 5(2), 80-88. http://journal.r-project.org/archive/2013-2/sax-steiner.pdf

See Also

`ta` for temporal aggregation, the inverse function of `td`.

`summary` is used to obtain and print a summary of the results.

`predict` is used to extract the disaggregated or interpolated high frequency series.

`plot` is used to plot the fitted and actual low frequency series, as well as the residuals.

Examples

```
data(swisspharma)

# one indicator, no intercept
mod1 <- td(sales.a ~ 0 + exports.q)
summary(mod1)  # summary statistics
plot(mod1)  # residual plot of regression
plot(predict(mod1))

# interpolated quarterly series

# temporally aggregated series is equal to the annual value
all.equal(window(
  ta(predict(mod1), conversion = "sum", to = "annual"),
  start = 1975), sales.a)

# several indicators, including an intercept
mod2 <- td(sales.a ~ imports.q + exports.q)
```

```
# no indicator (Denton-Cholette)
mod3 <- td(sales.a ~ 1, to = "quarterly", method = "denton-cholette")

# no indicator (uniform)
mod4 <- td(sales.a ~ 1, to = "quarterly", method = "uniform")

# Dynamic Chow-Lin (Santos Silva and Cardoso, 2001)
# (no truncation parameter added, because rho = 0)
mod5 <- td(sales.a ~ exports.q, method = "dynamic-maxlog")

# Example from Denton (1971), see references.
d.q <- ts(rep(c(50, 100, 150, 100), 5), frequency = 4)
d.a <- ts(c(500, 400, 300, 400, 500))

a1 <- predict(td(d.a ~ 0 + d.q, method = "denton",
                 criterion = "additive", h = 0))
a2 <- predict(td(d.a ~ 0 + d.q, method = "denton",
                 criterion = "additive", h = 1))
a3 <- predict(td(d.a ~ 0 + d.q, method = "denton",
                 criterion = "additive", h = 2))
a4 <- predict(td(d.a ~ 0 + d.q, method = "denton",
                 criterion = "additive", h = 3))

p1 <- predict(td(d.a ~ 0 + d.q, method = "denton",
                 criterion = "proportional", h = 0))
p2 <- predict(td(d.a ~ 0 + d.q, method = "denton",
                 criterion = "proportional", h = 1))
p3 <- predict(td(d.a ~ 0 + d.q, method = "denton",
                 criterion = "proportional", h = 2))
p4 <- predict(td(d.a ~ 0 + d.q, method = "denton",
                 criterion = "proportional", h = 3))

# Table in Denton (1971), page 101:
round(cbind(d.q, a1, a2, a3, a4, p1, p2, p3, p4))
```

Temporal Aggregation of Time Series

Description

Performs temporal aggregation of high to low frequency time series. Currently, `ta` only works with `ts` or `mts` time series objects.

Usage

```
ta(x, ...)

## S3 method for class 'ts'
ta(x, conversion = "sum", to = "annual", ...)
```

Arguments

`x`	a time series object of class `"ts"` or `"mts"`.
`...`	additional arguments, passed to the methods.
`conversion`	type of conversion: `"sum"`, `"average"`, `"first"` or `"last"`.
`to`	(low-frequency) destination frequency as a character string (`"annual"` or `"quarterly"`) or as a scalar (e.g. `1`, `2`, `4`).

Details

`ta` is used to aggregate a high frequency time series into a low frequency series, while the latter is either the sum, the average, the first or the last value of the high-frequency series. `ta` is the inverse function of `td`. If applied to an output series of `td`, `ta` yields the original series.

Value

`ta` returns an object of class `"ts"` or `"mts"`, depending on the class of the input series.

See Also

`td` for the main function for temporal disaggregation.

Examples

```
data(swisspharma)

sales.q.a <- ta(sales.q, conversion = "sum", to = "annual")
all.equal(sales.a, sales.q.a)
```

Time-Series Objects

Description

The function `ts` is used to create time-series objects.

`as.ts` and `is.ts` coerce an object to a time-series and test whether an object is a time series.

Usage

```
ts(data = NA, start = 1, end = numeric(), frequency = 1,
   deltat = 1, ts.eps = getOption("ts.eps"), class = , names = )
as.ts(x, ...)
is.ts(x)
```

Arguments

`data`	a vector or matrix of the observed time-series values. A data frame will be coerced to a numeric matrix via `data.matrix`. (See also ‘Details’.)
`start`	the time of the first observation. Either a single number or a vector of two integers, which specify a natural time unit and a (1-based) number of samples into the time unit. See the examples for the use of the second form.
`end`	the time of the last observation, specified in the same way as `start`.
`frequency`	the number of observations per unit of time.
`deltat`	the fraction of the sampling period between successive observations; e.g., 1/12 for monthly data. Only one of `frequency` or `deltat` should be provided.
`ts.eps`	time series comparison tolerance. Frequencies are considered equal if their absolute difference is less than `ts.eps`.
`class`	class to be given to the result, or none if `NULL` or `"none"`. The default is `"ts"` for a single series, `c("mts", "ts", "matrix")` for multiple series.
`names`	a character vector of names for the series in a multiple series: defaults to the colnames of `data`, or `Series 1`, `Series 2`, ...
`x`	an arbitrary **R** object.
`...`	arguments passed to methods (unused for the default method).

Details

The function `ts` is used to create time-series objects. These are vectors or matrices with class of `"ts"` (and additional attributes) which represent data which has been sampled at equispaced points in time. In the matrix case, each column of the matrix `data` is assumed to contain a single (univariate) time series. Time series must have at least one observation, and although they need not be numeric there is very limited support for non-numeric series.

Class `"ts"` has a number of methods. In particular arithmetic will attempt to align time axes, and subsetting to extract subsets of series can be used (e.g., `EuStockMarkets[, "DAX"]`). However, subsetting the first (or only) dimension will return a matrix or vector, as will matrix subsetting. Subassignment can be used to replace values but not to extend a series (see `window`). There is a method for `t` that transposes the series as a matrix (a one-column matrix if a vector) and hence returns a result that does not inherit from class `"ts"`.

The value of argument `frequency` is used when the series is sampled an integral number of times in each unit time interval. For example, one could use a value of `7` for `frequency` when the data are sampled daily, and the natural time period is a week, or `12` when the data are sampled monthly and the natural time period is a year. Values of `4` and `12` are assumed in (e.g.) `print` methods to imply a quarterly and monthly series respectively.

`as.ts` is generic. Its default method will use the `tsp` attribute of the object if it has one to set the start and end times and frequency.

`is.ts` tests if an object is a time series. It is generic: you can write methods to handle specific classes of objects, see InternalMethods.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

`tsp`, `frequency`, `start`, `end`, `time`, `window`; `print.ts`, the print method for time series objects; `plot.ts`, the plot method for time series objects.

For other definitions of ‘time series’ (e.g., time-ordered observations) see the CRAN task view at https://CRAN.R-project.org/view=TimeSeries.

Examples

```
require(graphics)

ts(1:10, frequency = 4, start = c(1959, 2)) # 2nd Quarter of 1959
print( ts(1:10, frequency = 7, start = c(12, 2)), calendar = TRUE)
# print.ts(.)
## Using July 1954 as start date:
gnp <- ts(cumsum(1 + round(rnorm(100), 2)),
          start = c(1954, 7), frequency = 12)
plot(gnp) # using 'plot.ts' for time-series plot

## Multivariate
z <- ts(matrix(rnorm(300), 100, 3), start = c(1961, 1), frequency = 12)
class(z)
head(z) # as "matrix"
plot(z)
plot(z, plot.type = "single", lty = 1:3)

## A phase plot:
plot(nhtemp, lag(nhtemp, 1), cex = .8, col = "blue",
     main = "Lag plot of New Haven temperatures")
```