Package: rjd3bench (via r-universe)

October 22, 2026

Type Package

Title Temporal Disaggregation and Benchmarking in 'JDemetra+' 3.x

Version 3.1.3

Description Interface to 'JDemetra+' 3.x
(<https://github.com/jdemetra>) time series analysis software.
It provides a variety of methods for temporal disaggregation &
interpolation, benchmarking, reconciliation and
calendarization. It incorporates statistical methods described
in the latest European Statistical System (ESS) guidelines on
temporal disaggregation, benchmarking, and reconciliation (2018
edition). The package implements highly efficient algorithms
for fast and reliable computation.

License EUPL

URL https://github.com/rjdverse/rjd3bench,
https://rjdverse.github.io/rjd3bench/

BugReports https://github.com/rjdverse/rjd3bench/issues

Depends R (>= 4.1.0)

Imports rJava (>= 1.0-6), rjd3jars (>= 0.0.5), rjd3toolkit (>= 3.8.0),
RProtoBuf (>= 0.4.25), stats, graphics

Encoding UTF-8

RoxygenNote 7.3.3

SystemRequirements Java (>= 21)

Collate 'utils.R' 'adl.R' 'benchmark.R' 'calendarization.R'
'deprecated.R' 'mbdenton.R' 'print.R' 'tempdisagg.R' 'zzz.R'

LazyData TRUE

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Jean Palate [aut], Corentin Lemasson [aut, cre], Tanguy
Barthelemy [ctb, art], Nikolina Rizanovska [ctb]

Maintainer Corentin Lemasson <corentin.lemasson@nbb.be>

Config/pak/sysreqs default-jdk

Repository https://cran.r-universe.dev

Date/Publication 2026-07-16 08:40:02 UTC

RemoteUrl https://github.com/cran/rjd3bench

RemoteRef HEAD

RemoteSha b6e8bd4d0e7d5a1fb970829e1e88dcd493292675

Contents

adl_disaggregation . . . 2
calendarization . . . 4
cholette . . . 6
cubicspline . . . 8
denton . . . 9
denton_modelbased . . . 11
denton_raw . . . 13
deprecated-rjd3bench . . . 15
grp . . . 15
multivariatecholette . . . 17
qna_data . . . 20
temporal_disaggregation . . . 21
temporal_disaggregation_raw . . . 23
temporal_interpolation . . . 26
temporal_interpolation_raw . . . 28
temporaldisaggregationI . . . 31

Index **33**

`adl_disaggregation` *Temporal Disaggregation of a Time Series by ADL Model*

Description

Perform temporal disaggregation of low-frequency to high-frequency time series using an Autoregressive Distributed Lag regression model.

Usage

```
adl_disaggregation(
  series,
  constant = TRUE,
  trend = FALSE,
  indicators = NULL,
  average = FALSE,
```

```
  phi = 0,
  phi.fixed = FALSE,
  phi.truncated = 0,
  xar = c("FREE", "SAME", "NONE"),
  diffuse = FALSE
)
```

Arguments

series	A low-frequency time series to be disaggregated. It must be "ts" object.
constant	Boolean. Indicates whether a constant term is included in the model. The default is TRUE.
trend	Boolean. Indicates whether a linear trend is included in the model. The default is FALSE.
indicators	One or more high-frequency indicator series. If not NULL (the default), this must be a "ts" object or a list of "ts" objects.
average	Boolean. Indicates whether an average conversion should be considered. The default is FALSE, corresponding to additive conversion.
phi	A numeric value giving the (initial) value of the phi parameter
phi.fixed	Boolean. Specifies whether the supplied value of phi is fixed. The default is FALSE, which indicates that phi is estimated.
phi.truncated	A numeric value defining the lower bound of the admissible range for phi. The evaluation range is [phi.truncated, 1[.
xar	A character string specifying the constraints imposed on the coefficients of the lagged regression variables. The default is "FREE", which indicates that no constraints are applied. Other options are: "SAME"and "NONE". For additional information, see the package vignette.
diffuse	Boolean. Indicates whether the coefficients of the regression model are treated as diffuse (TRUE) or as fixed unknown (FALSE, the default).

Value

An object of class "JD3_ADLDISAGG_RSLTS" is returned. The following are returned invisibly as a list:

- regression [[1]] regression coefficients;
- estimation [[2]] disaggregated Time-Series and standard deviation, parameter and residuals;
- likelihood [[3]] likelihood statistics.

References

Proietti, P. (2005). Temporal Disaggregation by State Space Methods: Dynamic Regression Methods Revisited. Working papers and Studies, European Commission, ISSN 1725-4825.

See Also

For more information, see the vignette:

`utils::browseVignettes()`, e.g. `browseVignettes(package = "rjd3bench")`

Examples

```
# ADL model
data("qna_data")
Y <- ts(qna_data$B1G_Y_data[,"B1G_FF"], frequency = 1, start = c(2009,1))
x <- ts(qna_data$TURN_Q_data[,"TURN_INDEX_FF"], frequency = 4, start = c(2009,1))
td1 <- adl_disaggregation(Y, indicators = x, xar = "FREE")
td1$estimation$disagg

# ADL models with constraints
td2 <- adl_disaggregation(Y, indicators = x, xar = "SAME") # ~ Chow-Lin
td3 <- adl_disaggregation(Y, constant = FALSE, indicators = x,
                          xar = "SAME", phi = 1, phi.fixed = TRUE) # ~ Fernandez
td4 <- adl_disaggregation(Y, indicators = x, xar = "NONE") # ~ Santos Silva-Cardoso
```

`calendarization` *Calendarization*

Description

Time series data do not always coincide with calendar periods (e.g., fiscal years starting in March-April or retail data collected in non-monthly intervals). Calendarization is the process of transforming the values of a flow time series observed over varying time intervals into values that cover given calendar intervals such as month, quarter or year. The process involves two steps. At first, a state-space representation of the Denton proportional first difference (PFD) method is considered to perform a temporal disaggregation of the observed data into daily values. After that, the resulting daily values are aggregated into the desired calendar reference periods.

Usage

```
calendarization(
  calendarobs,
  freq,
  start = NULL,
  end = NULL,
  dailyweights = NULL,
  stde = FALSE
)
```

Arguments

calendarobs	A named list containing the observed data. The list must consist of three elements: start, end and value, where the first two indicate the starting and ending dates of the observation. See the example.
freq	An integer specifying the annual frequency. If set to 0, only the daily series is computed.
start	The starting day of the calendarization. This date may precede the first observed data (retropolation).
end	The ending day of the calendarization. This date may exceed the last observed data (extrapolation).
dailyweights	A numeric vector of daily indicator values (or weights). The vector must have the same length as the requested daily series. When available, these weights typically reflects daily levels of activity, which may vary due to seasonality, trading day effects, or other calendar effects such as public holidays.
stde	Boolean. If TRUE, the function also returns the standard errors associated with the results. The default is FALSE.

Value

A list containing the disaggregated daily values, the final aggregated series, and their associated standard errors if requested.

References

Quenneville, B., Picard F., Fortier S. (2012). Calendarization with interpolating splines and state space models. Statistics Canada, Appl. Statistics (2013) 62, part 3, pp 371-399.

See Also

For more information, see the vignette:

utils::browseVignettes(), e.g. browseVignettes(package = "rjd3bench")

Examples

```
# Example 1 (from Quenneville et al (2012))

## Observed data
obs_1 <- list(
    list(start = "2009-02-18", end = "2009-03-17", value = 9000),
    list(start = "2009-03-18", end = "2009-04-14", value = 5000),
    list(start = "2009-04-15", end = "2009-05-12", value = 9500),
    list(start = "2009-05-13", end = "2009-06-09", value = 7000))

## a) calendarization in absence of daily indicator values (or weights)
cal_1a <- calendarization(obs_1, 12, end = "2009-06-30", dailyweights = NULL, stde = TRUE)

ym_1a <- cal_1a$rslt
```

```
eym_1a <- cal_1a$erslt
yd_1a <- cal_1a$days
eyd_1a <- cal_1a$edays

## b) calendarization in presence of daily indicator values (or weights)
x <- rep(c(1.0, 1.2, 1.8 , 1.6, 0.0, 0.6, 0.8), 19)
cal_1b <- calendarization(obs_1, 12, end = "2009-06-30", dailyweights = x, stde = TRUE)

ym_1b <- cal_1b$rslt
eym_1b <- cal_1b$erslt
yd_1b <- cal_1b$days
eyd_1b <- cal_1b$edays

# Example 2 (incl. negative value)

obs_2 <- list(
    list(start = "1980-01-01", end = "1989-12-31", value = 100),
    list(start = "1990-01-01", end = "1999-12-31", value = -10),
    list(start = "2000-01-01", end = "2002-12-31", value = 50))

cal_2 <- calendarization(obs_2, 4, end = "2003-12-31")

yq_2 <- cal_2$rslt
```

---

cholette *Benchmarking by means of the Cholette Method*

Description

Cholette method is based on a benchmarking methodology developed at Statistics Canada. It is a generalized method relying on the principle of movement preservation that encompasses other benchmarking methods. The Denton method (both the AFD and PFD variants), as well as the naive pro-rating method, emerge as particular cases of the Cholette method. This method has been widely used for the purpose of benchmarking seasonally adjusted series among others.

Usage

```
cholette(
  s,
  t,
  rho = 1,
  lambda = 1,
  bias = c("None", "Additive", "Multiplicative"),
  conversion = c("Sum", "Average", "Last", "First", "UserDefined"),
  obsposition = 1L
)
```

Arguments

s	A preliminary series. It must be of the same class as t.
t	The low-frequency aggregation constraint. It must be either a "ts" object or a numeric vector.
rho	Numeric. A smoothing parameter whose value must lie between 0 and 1. See the package vignette for more information on the choice of the rho parameter.
lambda	Numeric. The adjustment model parameter. Typical choices include lambda = 1 for proportional benchmarking, lambda = 0 for additive benchmarking, and lambda = 0.5 with rho = 0 for the naive pro-rating method. See the package vignette for more information on the choice of the lambda parameter.
bias	Character. Specifies the bias-correction factor. By default, no systematic bias is considered. Other options are: "Additive" and "Multiplicative". See vignette for more details. See the package vignette for more information on the other alternatives.
conversion	A character string specifying the conversion mode, typically "Sum" (the default) or "Average". Other options are: "Last", "First" and "UserDefined".
obsposition	An integer specifying the position of the observations of the low-frequency constraint within the benchmarked series (e.g. the 7th month of the year). This argument is used only when conversion = "UserDefined".

Value

A "ts" object with the benchmarked series is returned.

References

Quenneville, B., Fortier S., Chen Z.-G., Latendresse E. (2006). Recent Developments in Benchmarking to Annual Totals in X12-ARIMA and at Statistics Canada. Statistics Canada, Working paper of the Time Series Research and Analysis Centre.

See Also

For more information, see the vignette:

utils::browseVignettes(), e.g. browseVignettes(package = "rjd3bench")

Examples

```
ym_true <- rjd3toolkit::Retail$RetailSalesTotal
yq_true <- rjd3toolkit::aggregate(ym_true, 4)
Y_full <- rjd3toolkit::aggregate(ym_true, 1)

Y <- window(Y_full, end = c(2009,1)) # say no benchmark yet for the year 2010
xm <- ym_true + rnorm(n = length(ym_true), mean = -5000, sd = 10000)
xq <- rjd3toolkit::aggregate(xm, 4)

# Proportional benchmarking with a bias and some recommended value of rho for
# monthly and quarterly series respectively
cholette(s = xm, t = Y, rho = 0.9, lambda = 1, bias = "Multiplicative")
```

```
cholette(s = xq, t = Y, rho = 0.729, lambda = 1, bias = "Multiplicative")

# Proportional benchmarking with no bias
xm_no_bias <- ym_true + rnorm(n = length(ym_true), mean = 0, sd = 10000)
cholette(s = xm_no_bias, t = Y, rho = 0.9, lambda = 1)

# Additive benchmarking
cholette(s = xm, t = Y, rho = 0.9, lambda = 0, bias = "Additive")

# Denton PFD
cholette(s = xm, t = Y, rho = 1, lambda = 1)

# Pro-rating
cholette(s = xm, t = Y, rho = 0, lambda = 0.5)
```

`cubicspline` *Benchmarking by means of Cubic Splines*

Description

Cubic splines are piecewise cubic functions that are linked together in a way to guarantee smoothness at data points. Additivity constraints are added for benchmarking purpose and sub-period estimates are derived from each spline. When a preliminary series is used, cubic splines are no longer drawn based on the low-frequency constraint but the Benchmark-to-Indicator (BI ratio) is the one being smoothed. Sub-period estimates are then simply the product between the smoothed high frequency BI ratio and the preliminary series.

Usage

```
cubicspline(
  s = NULL,
  t,
  nfreq = 4L,
  conversion = c("Sum", "Average", "Last", "First", "UserDefined"),
  obsposition = 1L
)
```

Arguments

`s`	A preliminary series. If not `NULL`, it must be of the same class as `t`.
`t`	The low-frequency aggregation constraint. It must be either an object of class `ts` or a numeric vector.
`nfreq`	An integer giving the annual frequency of the benchmarked series. This argument is used only when no preliminary series is provided.
`conversion`	A character string specifying the conversion mode, typically `"Sum"` (the default) or `"Average"`. Other options are: `"Last"`, `"First"` and `"UserDefined"`.

obsposition — An integer specifying the position of the observations of the low-frequency constraint within the benchmarked series (e.g. the 7th month of the year). This argument is used only when conversion = "UserDefined".

Value

A "ts" object with the benchmarked series is returned.

See Also

For more information, see the vignette:

utils::browseVignettes(), e.g. browseVignettes(package = "rjd3bench")

Examples

```
data("qna_data")
Y <- ts(qna_data$B1G_Y_data[,"B1G_FF"], frequency = 1, start = c(2009,1))

# Cubic spline without preliminary series
y1 <- cubicspline(t = Y, nfreq = 4L)

# Cubic spline with preliminary series
x1 <- y1 + rnorm(n = length(y1), mean = 0, sd = 10)
cubicspline(s = x1, t = Y)

# Cubic splines used for temporal disaggregation
x2 <- ts(qna_data$TURN_Q_data[,"TURN_INDEX_FF"], frequency = 4, start = c(2009,1))
cubicspline(s = x2, t = Y)
```

---

denton *Benchmarking by means of the Denton Method.*

Description

Denton method relies on the principle of movement preservation. There exist a few variants corresponding to different definitions of movement preservation: additive first difference (AFD), proportional first difference (PFD), additive second difference (ASD), proportional second difference (PSD), etc. The default and most widely used is the Denton PFD method.

Usage

```
denton(
  s = NULL,
  t,
  d = 1L,
  mul = TRUE,
  nfreq = 4L,
```

```
  modified = TRUE,
  conversion = c("Sum", "Average", "Last", "First", "UserDefined"),
  obsposition = 1L,
  nbcsts = 0L,
  nfcsts = 0L
)
```

Arguments

s	A preliminary series. If not NULL, it must be of the same class as t.
t	The low-frequency aggregation constraint. It must be either a "ts" object or a numeric vector.
d	An integer specifying the differencing order. The default is 1.
mul	Boolean. Indicates whether benchmarking is multiplicative (TRUE) or additive (FALSE). The default is multiplicative.
nfreq	An integer giving the annual frequency of the benchmarked series. This argument is used only when no preliminary series is provided.
modified	Boolean. Specifies whether the modified Denton method (TRUE) or the unmodified Denton method (FALSE) is applied. The default is TRUE.
conversion	A character string specifying the conversion mode, typically "Sum" (the default) or "Average". Other options are: "Last", "First" and "UserDefined".
obsposition	An integer specifying the position of the observations of the low-frequency constraint within the benchmarked series (e.g. the 7th month of the year). This argument is used only when conversion = "UserDefined".
nbcsts	An integer specifying the number of backcast periods. This argument is ignored when a preliminary series is provided. (Not yet implemented.)
nfcsts	An integer specifying the number of forecast periods. This argument is ignored when a preliminary series is provided. (Not yet implemented.)

Value

A "ts" object with the benchmarked series is returned.

See Also

For more information, see the vignette:

utils::browseVignettes(), e.g. browseVignettes(package = "rjd3bench")

Examples

```
Y <- rjd3toolkit::aggregate(rjd3toolkit::Retail$RetailSalesTotal, 1)

# Denton PFD without a preliminary series
y1 <- denton(t = Y, nfreq = 4)
print(y1)

# Denton PFD without a preliminary series and conversion = "Average"
```

```
denton(t = Y, nfreq = 4, conversion = "Average")

# Denton PFD with a preliminary series
x <- y1 + rnorm(n = length(y1), mean = 0, sd = 10000)
denton(s = x, t = Y)

# Denton AFD with a preliminary series
denton(s = x, t = Y, mul = FALSE)
```

`denton_modelbased` *Temporal Disaggregation and Interpolation of a Time Series using the Model-Based Denton Proportional Method*

Description

The Denton proportional first difference (PFD) method can be expressed as a statistical model in a state-space representation. This formulation provides increased flexibility, including the ability to incorporate outliers, which correspond to level shifts in the Benchmark-to-Indicator (BI) ratio, that would otherwise induce unintended wave effects under the standard Denton PFD method. In addition, the approach allows the disaggregated series to be constrained (or 'frozen') at specific periods or prior to a given date by fixing the corresponding high-frequency BI ratios.

Usage

```
denton_modelbased(
  series,
  indicator,
  differencing = 1L,
  conversion = c("Sum", "Average", "Last", "First", "UserDefined"),
  conversion.obsposition = 1L,
  outliers = NULL,
  fixedBIratios = NULL
)
```

Arguments

`series`	A low-frequency time series to be disaggregated or interpolated. It must be either a `"ts"` object or a numeric vector.
`indicator`	A high-frequency indicator series. It must be of the same class as `series`.
`differencing`	Not yet implemented. This should be left equal to 1 (corresponding to the Denton PFD method).
`conversion`	A character string specifying the conversion mode, typically `"Sum"` (the default) or `"Average"`. Other options are: `"Last"`, `"First"` and `"UserDefined"`.
`conversion.obsposition`	An integer specifying the position of the low-frequency observations within the interpolated series (e.g. the 7th month of the year). This argument is used only for interpolation when `conversion = "UserDefined"`.

`outliers` A list specifying the outlier periods and their magnitude. Each element must be provided as `"YYYY-MM-DD" = value`, where the date identifies the period. The numeric value specifies the intensity of the outlier and corresponds to the relative value of the innovation variance (with 1 indicating the normal situation).

`fixedBIratios` A list specifying the periods for which the Benchmark-to-Indicator (BI) ratios should be fixed. Each element must be provided as `"YYYY-MM-DD" = value`, where the date identifies the period and the numeric value specifies the fixed BI ratio.

Value

An object of class "JD3_MBDENTON_RSLTS" is returned. The following are returned invisibly as a list:

- `estimation [[1]]` disaggregated Time-Series, BI ratios and standard deviations;
- `likelihood [[2]]` likelihood statistics.

See Also

For more information, see the vignette:

`utils::browseVignettes()`, e.g. `browseVignettes(package = "rjd3bench")`

Examples

```
# Retail data, monthly indicator
Y <- rjd3toolkit::aggregate(rjd3toolkit::Retail$RetailSalesTotal, 1)
x <- rjd3toolkit::aggregate(rjd3toolkit::Retail$FoodAndBeverageStores, 4)
td <- denton_modelbased(Y, x, outliers = list("2000-01-01" = 100, "2005-07-01" = 100))
y <- td$estimation$edisagg

# qna data, quarterly indicator
data("qna_data")
Y <- ts(qna_data$B1G_Y_data[,"B1G_FF"], frequency = 1, start = c(2009,1))
x <- ts(qna_data$TURN_Q_data[,"TURN_INDEX_FF"], frequency = 4, start = c(2009,1))

td1 <- denton_modelbased(Y, x)
td2 <- denton_modelbased(Y, x, outliers = list("2020-04-01" = 100),
                         fixedBIratios = list("2021-04-01" = 39.0))
bi1 <- td1$estimation$biratio
bi2 <- td2$estimation$biratio
y1 <- td1$estimation$disagg
y2 <- td2$estimation$disagg

stats::ts.plot(bi2, bi1, main = "BI ratios",
               gpars = list(col = c("red", "black")))
graphics::legend("topright", lty = 1, col = c("black", "red"),
                 legend = c("td1", "td2"))
stats::ts.plot(y2, y1, main = "Disaggregated series",
               gpars = list(col = c("red", "black")))
graphics::legend("topleft", lty = 1, col = c("black", "red"),
                 legend = c("td1", "td2"))
```

denton_raw *Benchmarking of an Atypical Frequency Series by means of the Denton Method.*

Description

Denton method relies on the principle of movement preservation. There exist a few variants corresponding to different definitions of movement preservation: additive first difference (AFD), proportional first difference (PFD), additive second difference (ASD), proportional second difference (PSD), etc. The default and most widely used is the Denton PFD method. The `denton_raw()` function extends `denton()` by allowing benchmarking for any frequency ratio.

Usage

```
denton_raw(
  s = NULL,
  t,
  freqratio,
  d = 1L,
  mul = TRUE,
  modified = TRUE,
  conversion = c("Sum", "Average", "Last", "First", "UserDefined"),
  obsposition = 1L,
  startoffset = 0L,
  nbcsts = 0L,
  nfcsts = 0L
)
```

Arguments

`s`	A preliminary series. If not `NULL`, it must be a numeric vector.
`t`	The low-frequency aggregation constraint. It must be a numeric vector.
`freqratio`	An integer specifying the frequency ratio between the benchmarked series and the low-frequency constraint. This argument is mandatory and must be a positive integer.
`d`	An integer specifying the differencing order. The default is `1`.
`mul`	Boolean. Indicates whether benchmarking is multiplicative (`TRUE`) or additive (`FALSE`). The default is multiplicative.
`modified`	Boolean. Specifies whether the modified Denton method (`TRUE`) or the unmodified Denton method (`FALSE`) is applied. The default is `TRUE`.
`conversion`	A character string specifying the conversion mode, typically `"Sum"` (the default) or `"Average"`. Other options are: `"Last"`, `"First"` and `"UserDefined"`.

obsposition	An integer specifying the position of the observations of the low-frequency constraint within the benchmarked series (e.g. the 7th month of the year). This argument is used only when conversion = "UserDefined".
startoffset	The number of initial observations in the preliminary series that precede the start of the low-frequency constraint. The value must be either 0 or a positive integer (default is 0). This argument is ignored when no preliminary series is provided.
nbcsts	An integer specifying the number of backcast periods. This argument is ignored when a preliminary series is provided. (Not yet implemented.)
nfcsts	An integer specifying the number of forecast periods. This argument is ignored when a preliminary series is provided. (Not yet implemented.)

Value

A numeric vector with the benchmarked series is returned.

See Also

For more information, see the vignette:

utils::browseVignettes(), e.g. browseVignettes(package = "rjd3bench")

Examples

```
Y <- c(500,510,525,520)
x <- c(97, 98, 98.5, 99.5, 104,
       99, 100, 100.5, 101, 105.5,
       103, 104.5, 103.5, 104.5, 109,
       104, 107, 103, 108, 113,
       110)

# Denton PFD
# for example, x and Y could be annual and quinquennal series respectively
denton_raw(x, Y, freqratio = 5)

# Denton AFD
denton_raw(x, Y, freqratio = 5, mul = FALSE)

# Denton PFD without indicator
denton_raw(t = Y, freqratio = 2, conversion = "Average")

# Denton PFD with/without an offset and conversion = "Last"
x2 <- c(485,
        490, 492.5, 497.5, 520, 495,
        500, 502.5, 505, 527.5, 515,
        522.5, 517.5, 522.5, 545, 520,
        535, 515, 540, 565, 550)
denton_raw(x2, Y, freqratio = 5, conversion = "Last")
denton_raw(x2, Y, freqratio = 5, conversion = "Last", startoffset = 1)
```

deprecated-rjd3bench *Deprecated Functions*

Description

This function is deprecated. Use the function `temporal_disaggregation()` or `temporal_interpolation()` instead.

Usage

```
temporaldisaggregation(
  series,
  constant = TRUE,
  trend = FALSE,
  indicators = NULL,
  model = c("Ar1", "Rw", "RwAr1"),
  freq = 4L,
  conversion = c("Sum", "Average", "Last", "First", "UserDefined"),
  conversion.obsposition = 1L,
  rho = 0,
  rho.fixed = FALSE,
  rho.truncated = 0,
  zeroinitialization = FALSE,
  diffuse.algorithm = c("SqrtDiffuse", "Diffuse", "Augmented"),
  diffuse.regressors = FALSE
)
```

Arguments

`series`, `constant`, `trend`, `indicators`, `model`, `freq`, `conversion`, `conversion.obsposition`, `rho`, `rho.fixed`, `rho.truncated`, `zeroinitialization`, `diffuse.algorithm`, `diffuse.regressors`
Parameters.

Value

Return the same value as either function that replaces it.

grp *Benchmarking following the Growth Rate Preservation Principle.*

Description

GRP is a method which explicitly preserves the period-to-period growth rates of the preliminary series. It corresponds to the method of Cauley and Trager (1981), using the solution proposed by Di Fonzo and Marini (2011). BFGS is used as line-search algorithm for the reduced unconstrained minimization problem.

Usage

```
grp(
  s,
  t,
  objective = c("Forward", "Backward", "Symmetric", "Log"),
  conversion = c("Sum", "Average", "Last", "First", "UserDefined"),
  obsposition = 1L,
  eps = 1e-12,
  iter = 500L,
  dentoninitialization = TRUE
)
```

Arguments

s	A preliminary series. It must be a "ts" object.
t	The low-frequency aggregation constraint. It must be a "ts" object.
objective	A character string specifying the objective function. The default is "Forward". Other options are: "Backward", "Symmetric" and "Log". For additional information on this, see the package vignette.
conversion	A character string specifying the conversion mode, typically "Sum" (the default) or "Average". Other options are: "Last", "First" and "UserDefined".
obsposition	An integer specifying the position of the observations of the low-frequency constraint within the benchmarked series (e.g. the 7th month of the year). This argument is used only when conversion = "UserDefined".
eps	A numeric value specifying the convergence tolerance. The BFGS algorithm proceeds until the reduction in the objective function is within this tolerance (default is 1e-12) or until the maximum number of iterations is reached.
iter	An integer giving the maximum number of iterations allowed in the BFGS algorithm. The default is 500.
dentoninitialization	Boolean. Indicates whether the series obtained via the modified Denton PFD method is used as the starting values for the GRP optimization procedure. The default is TRUE. If FALSE, the starting values are derived directly from the aggregation constraint (e.g. t/4 for quarterly series with annual constraint and conversion = "Sum").

Value

A "ts" object with the benchmarked series is returned.

References

Causey, B., and Trager, M.L. (1981). Derivation of Solution to the Benchmarking Problem: Trend Revision. Unpublished research notes, U.S. Census Bureau, Washington D.C. Available as an appendix in Bozik and Otto (1988).

Di Fonzo, T., and Marini, M. (2011). A Newton's Method for Benchmarking Time Series according to a Growth Rates Preservation Principle. *IMF WP/11/179*.

Daalmans, J., Di Fonzo, T., Mushkudiani, N. and Bikker, R. (2018). Growth Rates Preservation (GRP) temporal benchmarking: Drawbacks and alternative solutions. *Statistics Canada*.

See Also

For more information, see the vignette:

`utils::browseVignettes()`, e.g. `browseVignettes(package = "rjd3bench")`

Examples

```
data("qna_data")

Y <- ts(qna_data$B1G_Y_data[, "B1G_FF"], frequency = 1, start = c(2009, 1))
x <- denton(t = Y, nfreq = 4) + rnorm(n = length(Y) * 4, mean = 0, sd = 10)
grp(s = x, t = Y)
```

---

multivariatecholette *Reconciliation by means of the Multivariate Cholette Method*

Description

This is a multivariate extension of the Cholette benchmarking method which can be used for the purpose of reconciliation. While standard benchmarking methods consider one target series at a time, reconciliation techniques aim to restore consistency in a system of time series with regards to both contemporaneous and temporal constraints. Reconciliation techniques are typically needed when the total and its components are estimated independently (the so-called direct approach). The multivariate Cholette method relies on the principle of movement preservation and encompasses other reconciliation methods such as the multivariate Denton method.

Usage

```
multivariatecholette(
  xlist,
  tcvector = NULL,
  ccvector = NULL,
  rho = 0.8,
  lambda = 0.5
)
```

Arguments

`xlist`	A named list of `ts` objects containing all input. Each element should correspond to one input series: a preliminary series, a low-frequency series representing a temporal aggregation constraint, or a high-frequency series representing a contemporaneous constraint.

`tcvector` A character vector defining the temporal constraints. Each element must be written in the form `"Y = sum(x)"`, where `"Y"` is the name of a low-frequency temporal constraint and `"x"` is the name of a high-frequency preliminary series. The names must match those provided in `xlist`. The default is `NULL`, indicating that no temporal constraints are considered.

`ccvector` NULL (default) or a character vector defining each contemporaneous constraints. If NULL, no contemporaneous constraint is considered.This is equivalent to applying the univariate Cholette method to each of the preliminary series separately. Otherwise, each element of the vector must be written in the form $z = w_1x_1 + \ldots + w_nx_n$ or $c = w_1x_1 + \ldots + w_nx_n$ where:

- z is the name of a high-frequency contemporaneous constraint,
- $(w_1, \ldots, w_n)$ are optional numeric weights,
- $(x_1, \ldots, x_n)$ are the names of the high-frequency preliminary series and
- c is a constant.

The $+$ operator can be replaced by $-$. The names of the contemporaneous constraint(s) and the preliminary series are the one given in the `xlist` argument.

Important: Any series placed on the left-hand side of a constraint cannot appear on the right-hand side of any other constraint. This is because quantities on the left-hand side are fixed, while those on the right-hand side are adjusted to satisfy the equality.

`rho` Numeric. The smoothing parameter whose value must lie between 0 and 1. The default is `0.8`. See the package vignette for more information on the choice of the `rho` parameter.

`lambda` Numeric. The adjustment model parameter. Typical values include `lambda = 0`, `lambda = 0.5` (the default) and `lambda = 1`. See the package vignette for more information on the choice of the `lambda` parameter.

Value

A named list containing the benchmarked series is returned.

See Also

For more information, see the vignette:

`utils::browseVignettes()`, e.g. `browseVignettes(package = "rjd3bench")`

Examples

```
# Example 1: one "standard" contemporaneous constraint: z=x1+x2+x3

x1 <- ts(c(7, 7.2, 8.1, 7.5, 8.5, 7.8, 8.1, 8.4), frequency = 4, start = c(2010, 1))
x2 <- ts(c(18, 19.5, 19.0, 19.7, 18.5, 19.0, 20.3, 20.0), frequency = 4, start = c(2010, 1))
x3 <- ts(c(1.5, 1.8, 2, 2.5, 2.0, 1.5, 1.7, 2.0), frequency = 4, start = c(2010, 1))

z <- ts(c(27.1, 29.8, 29.9, 31.2, 29.3, 27.9, 30.9, 31.8), frequency = 4, start = c(2010, 1))

Y1 <- ts(c(30.0, 30.6), frequency = 1, start = c(2010, 1))
Y2 <- ts(c(80.0, 81.2), frequency = 1, start = c(2010, 1))
```

```
Y3 <- ts(c(8.0, 8.1), frequency = 1, start = c(2010, 1))

## Check consistency between temporal and contemporaneous constraints
lfs <- cbind(Y1,Y2,Y3)
rowSums(lfs) - stats::aggregate.ts(z) # should all be 0

data_list <- list(x1 = x1, x2 = x2, x3 = x3, z = z, Y1 = Y1, Y2 = Y2, Y3 = Y3)
tc <- c("Y1 = sum(x1)", "Y2 = sum(x2)", "Y3 = sum(x3)") # temporal constraints
cc <- c("z = x1+x2+x3") # (binding) contemporaneous constraint
cc_nb <- c("0 = x1+x2+x3-z") # non-binding contemporaneous constraint

## Run function with default values for rho and lambda
multivariatecholette(xlist = data_list, tcvector = tc, ccvector = cc)

## Run function with some trade-off values for rho and lambda
multivariatecholette(xlist = data_list, tcvector = tc, ccvector = cc, rho = .5, lambda = .5)

## Run function with the value of rho corresponding to Denton or Cholette
multivariatecholette(xlist = data_list, tcvector = tc, ccvector = cc, rho = 1) # Denton
multivariatecholette(xlist = data_list, tcvector = tc, ccvector = cc, rho = 0.729) # Cholette

## Run function without temporal constraints
multivariatecholette(xlist = data_list, tcvector = NULL, ccvector = cc)

## Run function considering non-binding contemporaneous constraint
multivariatecholette(xlist = data_list, tcvector = tc, ccvector = cc_nb)

# Example 2: two contemporaneous constraints: x1+3*x2+0.5*x3+x4+x5 = z1 and x1+x2 = x4

x1 <- ts(c(7.0,7.3,8.1,7.5,8.5,7.8,8.1,8.4), frequency=4, start=c(2010,1))
x2 <- ts(c(1.5,1.8,2.0,2.5,2.0,1.5,1.7,2.0), frequency=4, start=c(2010,1))
x3 <- ts(c(18.0,19.5,19.0,19.7,18.5,19.0,20.3,20.0), frequency=4, start=c(2010,1))
x4 <- ts(c(8,9.5,9.0,10.7,8.5,10.0,10.3,9.0), frequency=4, start=c(2010,1))
x5 <- ts(c(5,9.6,7.2,7.1,4.3,4.6,5.3,5.9), frequency=4, start=c(2010,1))

z1 <- ts(c(38.1,41.8,41.9,43.2,38.8,39.1,41.9,43.7), frequency=4, start=c(2010,1))

Y1 <- ts(c(30.0,30.5), frequency=1, start=c(2010,1))
Y2 <- ts(c(10.0,10.5), frequency=1, start=c(2010,1))
Y3 <- ts(c(80.0,81.0), frequency=1, start=c(2010,1))
Y4 <- ts(c(40.0,41.0), frequency=1, start=c(2010,1))
Y5 <- ts(c(25.0,20.0), frequency=1, start=c(2010,1))

### check consistency between temporal and contemporaneous constraints
wlfs <- cbind(Y1,3*Y2,0.5*Y3,Y4,Y5)
rowSums(wlfs) - stats::aggregate.ts(z1) # cc1: should all be 0
Y1 + Y2 - Y4 # cc2: should all be 0

data.list <- list(x1=x1,x2=x2,x3=x3,x4=x4,x5=x5,z1=z1,Y1=Y1,Y2=Y2,Y3=Y3,Y4=Y4,Y5=Y5)
tc <- c("Y1=sum(x1)", "Y2=sum(x2)", "Y3=sum(x3)", "Y4=sum(x4)", "Y5=sum(x5)")
cc <- c("z1=x1+3*x2+0.5*x3+x4+x5", "0=x1+x2-x4")

multivariatecholette(xlist = data.list, tcvector = tc, ccvector = cc)
```

qna_data *Quarterly National Accounts data for temporal disaggregation*

Description

This dataset contains two data frames used for temporal disaggregation and benchmarking exercises. The first data frame, B1G_Y_data, includes three annual benchmark series corresponding to Belgian annual value added for the period 2009–2020 in three industries: chemical industry (CE), construction (FF), and transport services (HH). The second data frame, TURN_Q_data, contains the corresponding quarterly indicator series derived from VAT-based production indicators, covering the period 2009Q1–2021Q4.

Usage

```
qna_data
```

Format

A named list with two elements:

B1G_Y_data A data frame with columns:

- DATE Annual periods.
- B1G_CE Value added for chemical industry.
- B1G_FF Value added for construction.
- B1G_HH Value added for transport services.

TURN_Q_data A data frame with columns:

- DATE Quarterly periods.
- TURN_INDEX_CE Quarterly indicator for chemical industry.
- TURN_INDEX_FF Quarterly indicator for construction.
- TURN_INDEX_HH Quarterly indicator for transport services.

Source

Belgian Quarterly National Accounts

Examples

```
data(qna_data)
names(qna_data)
head(qna_data$B1G_Y_data)
head(qna_data$TURN_Q_data)
```

`temporal_disaggregation` *Temporal Disaggregation of a Time Series by Regression Models.*

Description

Perform temporal disaggregation of low-frequency to high-frequency time series by regression models. The implemented models include Chow-Lin, Fernandez, Litterman and some variants of those algorithms.

Usage

```
temporal_disaggregation(
  series,
  constant = TRUE,
  trend = FALSE,
  indicators = NULL,
  model = c("Ar1", "Rw", "RwAr1"),
  freq = 4L,
  average = FALSE,
  rho = 0,
  rho.fixed = FALSE,
  rho.truncated = 0,
  zeroinitialization = FALSE,
  diffuse.algorithm = c("SqrtDiffuse", "Diffuse", "Augmented"),
  diffuse.regressors = FALSE,
  nbcsts = 0L,
  nfcsts = 0L
)
```

Arguments

`series`	A low-frequency time series to be disaggregated. It must be a `"ts"` object.
`constant`	Boolean. Indicates whether a constant term is included in the model. The default is `TRUE`. Note that this argument is used only with `model = "Ar1"` when `zeroinitialization = FALSE`. For additional information, see the package vignette.
`trend`	Boolean. Indicates whether a linear trend is included in the model. The default is `FALSE`.
`indicators`	One or more high-frequency indicator series used in the temporal disaggregation. If `NULL` (the default), no indicator is used. When provided, the argument must be a `"ts"` object or a list of `"ts"` objects.
`model`	A character string specifying the model of the error term at the disaggregated level. The options are: `"Ar1"` (Chow Lin, the default), `"Rw"` (Fernandez), and `"RwAr1"` (Litterman).

`freq` An integer giving the annual frequency of the disaggregated series. This argument is ignored when one or more indicator series is provided.

`average` Boolean. Indicates whether an average conversion should be considered. The default is `FALSE`, corresponding to additive conversion.

`rho` A numeric value giving the (initial) value of the autoregressive parameter. This argument is used only for `"Ar1"` and `"RwAr1"` models.

`rho.fixed` Boolean. Specifies whether the supplied value of `rho` is fixed. The default is `FALSE`, which indicates that `rho` is estimated.

`rho.truncated` A numeric value defining the lower bound of the admissible range for `rho`. The evaluation range is `[rho.truncated, 1[`.

`zeroinitialization`
Boolean. If `TRUE`, the initial values of the autoregressive model are set to zero. The default is `FALSE`.

`diffuse.algorithm`
A character string specifying the algorithm used for diffuse initialization. The default is `"SqrtDiffuse"`. Other options are: `"Diffuse"` and `"Augmented"`.

`diffuse.regressors`
Boolean. Indicates whether the coefficients of the regression model are treated as diffuse (`TRUE`) or as fixed unknown (`FALSE`, the default).

`nbcsts` An integer specifying the number of backcast periods. This argument is ignored when one or more indicator series is provided.

`nfcsts` An integer specifying the number of forecast periods. This argument is ignored when one or more indicator series is provided.

Value

An object of class "JD3_TEMPDISAGG_RSLTS" is returned. The following are returned invisibly as a list:

- `regression [[1]]` regression coefficients;
- `estimation [[2]]` disaggregated Time-Series and standard deviation, regression effects, smoothing part, parameter and residuals;
- `likelihood [[3]]` likelihood statistics.

See Also

`temporal_interpolation()` for interpolation,

`temporal_disaggregation_raw()` for temporal disaggregation of atypical frequency series,

`temporal_interpolation_raw()` for interpolation of atypical frequency series

For more information, see the vignette:

`utils::browseVignettes()`, e.g. `browseVignettes(package = "rjd3bench")`

Examples

```
# Chow-lin with a monthly indicator
Y <- rjd3toolkit::aggregate(rjd3toolkit::Retail$RetailSalesTotal, 1)
x <- rjd3toolkit::Retail$FoodAndBeverageStores
td <- temporal_disaggregation(Y, indicators = x)
td$estimation$disagg

# Fernandez with and without a quarterly indicator
data("qna_data")
Y <- ts(qna_data$B1G_Y_data[,"B1G_FF"], frequency = 1, start = c(2009,1))
x <- ts(qna_data$TURN_Q_data[,"TURN_INDEX_FF"], frequency = 4, start = c(2009,1))
td1 <- temporal_disaggregation(Y, indicators = x, model = "Rw")
td1$estimation$disagg

td2 <- temporal_disaggregation(Y, model = "Rw", nfcsts = 6)
td2$estimation$disagg

# Chow-lin applied to index series
Y_index <- 100 * Y / Y[1]
x_index <- 100 * x / x[1]
td3 <- temporal_disaggregation(Y, indicators = x, average = TRUE)
td3$estimation$disagg
```

---

temporal_disaggregation_raw

Temporal Disaggregation of an Atypical Frequency Series by Regression Models.

Description

Perform temporal disaggregation of low-frequency to high-frequency time series by regression models. The implemented models include Chow-Lin, Fernandez, Litterman and some variants of those algorithms. The `temporal_disaggregation_raw()` function extends `temporal_disaggregation()` by allowing temporal disaggregation for any frequency ratio.

Usage

```
temporal_disaggregation_raw(
  series,
  constant = TRUE,
  trend = FALSE,
  indicators = NULL,
  startoffset = 0L,
  model = c("Ar1", "Rw", "RwAr1"),
  freqratio,
  average = FALSE,
  rho = 0,
```

```
  rho.fixed = FALSE,
  rho.truncated = 0,
  zeroinitialization = FALSE,
  diffuse.algorithm = c("SqrtDiffuse", "Diffuse", "Augmented"),
  diffuse.regressors = FALSE,
  nbcsts = 0L,
  nfcsts = 0L
)
```

Arguments

series	A low-frequency time series to be disaggregated. It must be a numeric vector.
constant	Boolean. Indicates whether a constant term is included in the model. The default is TRUE. Note that this argument is used only with model = "Ar1" when zeroinitialization = FALSE. For additional information on this, see the package vignette.
trend	Boolean. Indicates whether a linear trend is included in the model. The default is FALSE.
indicators	One or more high-frequency indicator series used in the temporal disaggregation. If NULL (the default), no indicator is used. When provided, the argument must be a numeric vector or a matrix.
startoffset	The number of initial observations in the indicator series that precede the start of the low-frequency series. The value must be either 0 or a positive integer (default is 0). This argument is ignored when no indicator is provided.
model	A character string specifying the model of the error term at the disaggregated level. The options are: "Ar1" (Chow Lin, the default), "Rw" (Fernandez), and "RwAr1" (Litterman).
freqratio	An integer specifying the frequency ratio between the disaggregated series and the low-frequency series. This argument is mandatory and must be a positive integer.
average	Boolean. Indicates whether an average conversion should be considered. The default is FALSE, corresponding to additive conversion.
rho	A numeric value giving the (initial) value of the autoregressive parameter. This argument is used only for "Ar1" and "RwAr1" models.
rho.fixed	Boolean. Specifies whether the supplied value of rho is fixed. The default is FALSE, which indicates that rho is estimated.
rho.truncated	A numeric value defining the lower bound of the admissible range for rho. The evaluation range is [rho.truncated, 1[.
zeroinitialization	Boolean. If TRUE, the initial values of the autoregressive model are set to zero. The default is FALSE.
diffuse.algorithm	A character string specifying the algorithm used for diffuse initialization. The default is "SqrtDiffuse". Other options are: "Diffuse" and "Augmented".

`diffuse.regressors`
: Boolean. Indicates whether the coefficients of the regression model are treated as diffuse (`TRUE`) or as fixed unknown (`FALSE`, the default).

`nbcsts`
: An integer specifying the number of backcast periods. This argument is ignored when one or more indicator series is provided.

`nfcsts`
: An integer specifying the number of forecast periods. This argument is ignored when one or more indicator series is provided.

Value

An object of class "JD3_TEMPDISAGGRAW_RSLTS" is returned. The following are returned invisibly as a list:

- `regression [[1]]` regression coefficients;
- `estimation [[2]]` disaggregated values and standard deviation, regression effects, smoothing part, parameter and residuals;
- `likelihood [[3]]` likelihood statistics.

See Also

`temporal_interpolation_raw()`

For more information, see the vignette:

`utils::browseVignettes()`, e.g. `browseVignettes(package = "rjd3bench")`

Examples

```
# Use of Chow-lin method to disaggregate a biennial series with an annual indicator
Y <- stats::aggregate(rjd3toolkit::Retail$RetailSalesTotal, 0.5)
x <- stats::aggregate(rjd3toolkit::Retail$FoodAndBeverageStores, 1)
td <- temporal_disaggregation_raw(as.numeric(Y), indicators = as.numeric(x), freqratio = 2)
td$estimation$disagg

# Use of Fernandez method to disaggregate a series without indicator
# considering a frequency ratio of 5 (for example, it could be a quinquennial
# series to disaggregate on an annual basis)
Y2 <- c(500,510,525,520)
td2 <- temporal_disaggregation_raw(Y2, model = "Rw", freqratio = 5, nfcsts = 2)
td2$estimation$disagg

# Same with an indicator, considering an offset in the latter
Y2 <- c(500,510,525,520)
x2 <- c(97,
        98, 98.5, 99.5, 104, 99,
        100, 100.5, 101, 105.5, 103,
        104.5, 103.5, 104.5, 109, 104,
        107, 103, 108, 113, 110)
td3 <- temporal_disaggregation_raw(Y2, indicators = x2, startoffset = 1,
                                   model = "Rw", freqratio = 5)
td3$estimation$disagg
```

`temporal_interpolation` *Interpolation of a Time Series by Regression Models.*

Description

Perform temporal interpolation of low-frequency to high-frequency time series by regression models. The implemented models include Chow-Lin, Fernandez, Litterman and some variants of those algorithms.

Usage

```
temporal_interpolation(
  series,
  constant = TRUE,
  trend = FALSE,
  indicators = NULL,
  model = c("Ar1", "Rw", "RwAr1"),
  freq = 4L,
  obsposition = -1L,
  rho = 0,
  rho.fixed = FALSE,
  rho.truncated = 0,
  zeroinitialization = FALSE,
  diffuse.algorithm = c("SqrtDiffuse", "Diffuse", "Augmented"),
  diffuse.regressors = FALSE,
  nbcsts = 0L,
  nfcsts = 0L
)
```

Arguments

`series`	A low-frequency time series to be interpolated. It must be a `"ts"` object.
`constant`	Boolean. Indicates whether a constant term is included in the model. The default is `TRUE`. Note that this argument is used only with `model = "Ar1"` when `zeroinitialization = FALSE`. For additional information, see the package vignette.
`trend`	Boolean. Indicates whether a linear trend is included in the model. The default is `FALSE`.
`indicators`	One or more high-frequency indicator series used in the temporal interpolation. If `NULL` (the default), no indicator is used. When provided, the argument must be a `"ts"` object or a list of `"ts"` objects.
`model`	A character string specifying the model of the error term at the interpolated level. The options are: `"Ar1"` (Chow Lin, the default), `"Rw"` (Fernandez), and `"RwAr1"` (Litterman).

freq An integer giving the annual frequency of the interpolated series. This argument is ignored when one or more indicator series is provided.

obsposition An integer specifying the position of the low-frequency observations within the interpolated series (e.g. the 1st month of the year, the 2d month, etc.). The value must be a positive integer or -1 (the default). The default value is equivalent to setting the value of the parameter equal to the frequency of the series, meaning that the last value of the interpolated series is consistent with the low-frequency series.

rho A numeric value giving the (initial) value of the autoregressive parameter. This argument is used only for "Ar1" and "RwAr1" models.

rho.fixed Boolean. Specifies whether the supplied value of rho is fixed. The default is FALSE, which indicates that rho is estimated.

rho.truncated A numeric value defining the lower bound of the admissible range for rho. The evaluation range is [rho.truncated, 1[.

zeroinitialization
Boolean. If TRUE, the initial values of the autoregressive model are set to zero. The default is FALSE.

diffuse.algorithm
A character string specifying the algorithm used for diffuse initialization. The default is "SqrtDiffuse". Other options are: "Diffuse" and "Augmented".

diffuse.regressors
Boolean. Indicates whether the coefficients of the regression model are treated as diffuse (TRUE) or as fixed unknown (FALSE, the default).

nbcsts An integer specifying the number of backcast periods. This argument is ignored when one or more indicator series is provided.

nfcsts An integer specifying the number of forecast periods. This argument is ignored when one or more indicator series is provided.

Value

An object of class "JD3_INTERP_RSLTS" is returned. The following are returned invisibly as a list:

- regression [[1]] regression coefficients;
- estimation [[2]] interpolated Time-Series and standard deviation, regression effects and smoothing part, parameter and residuals;
- likelihood [[3]] likelihood statistics.

See Also

temporal_disaggregation(),

temporal_interpolation_raw() for interpolation of atypical frequency series,

temporal_disaggregation_raw() for temporal disaggregation of atypical frequency series

For more information, see the vignette:

utils::browseVignettes(), e.g. browseVignettes(package = "rjd3bench")

Examples

```
# Chow-lin / Fernandez when the last value of the interpolated series is
# consistent with the low-frequency series
Y <- rjd3toolkit::aggregate(rjd3toolkit::Retail$RetailSalesTotal, 1)
x <- rjd3toolkit::Retail$FoodAndBeverageStores
ti1 <- temporal_interpolation(Y, indicators = x)
ti1$estimation$interp

ti2 <- temporal_interpolation(Y, indicators = x, model = "Rw")
ti2$estimation$interp

# Same without indicator
ti3 <- temporal_interpolation(Y, model = "Rw", freq = 12, nfcsts = 6)
ti3$estimation$interp

ti4 <- temporal_interpolation(Y, indicators = x, obsposition = 1)
ti4$estimation$interp
```

`temporal_interpolation_raw` *Interpolation of an Atypical Frequency Series by Regression Models.*

Description

Perform temporal interpolation of low-frequency to high-frequency time series by regression models. The implemented models include Chow-Lin, Fernandez, Litterman and some variants of those algorithms. The `temporal_interpolation_raw()` function extends `temporal_interpolation()` by allowing temporal interpolation for any frequency ratio.

Usage

```
temporal_interpolation_raw(
  series,
  constant = TRUE,
  trend = FALSE,
  indicators = NULL,
  startoffset = 0L,
  model = c("Ar1", "Rw", "RwAr1"),
  freqratio,
  obsposition = -1L,
  rho = 0,
  rho.fixed = FALSE,
  rho.truncated = 0,
  zeroinitialization = FALSE,
  diffuse.algorithm = c("SqrtDiffuse", "Diffuse", "Augmented"),
  diffuse.regressors = FALSE,
  nbcsts = 0L,
```

```
  nfcsts = 0L
)
```

Arguments

series	A low-frequency time series to be interpolated. It must be a numeric vector.
constant	Boolean. Indicates whether a constant term is included in the model. The default is TRUE. Note that this argument is used only with model = "Ar1" when zeroinitialization = FALSE. For additional information, see the package vignette.
trend	Boolean. Indicates whether a linear trend is included in the model. The default is FALSE.
indicators	One or more high frequency indicator series used in the interpolation. If NULL (the default), no indicator is used. When provided, the argument must be a numeric vector or a matrix.
startoffset	The number of initial observations in the indicator series that precede the start of the low-frequency series. The value must be either 0 or a positive integer (default is 0). This argument is ignored when no indicator is provided.
model	A character string specifying the model of the error term at the disaggregated level. The options are: "Ar1" (Chow Lin, the default), "Rw" (Fernandez), and "RwAr1" (Litterman).
freqratio	An integer specifying the frequency ratio between the interpolated series and the low-frequency series. This argument is mandatory and must be a positive integer.
obsposition	An integer specifying the position of the low-frequency observations within the interpolated series (e.g. the 1st month of the year, the 2d month, etc.). The value must be a positive integer or -1 (the default).The default value is equivalent to setting the value of the parameter equal to the frequency of the series, meaning that the last value of the interpolated series is consistent with the low-frequency series.
rho	A numeric value giving the (initial) value of the autoregressive parameter. This argument is used only for "Ar1" and "RwAr1" models.
rho.fixed	Boolean. Specifies whether the supplied value of rho is fixed. The default is FALSE, which indicates that rho is estimated.
rho.truncated	A numeric value defining the lower bound of the admissible range for rho. The evaluation range is [rho.truncated, 1[.
zeroinitialization	Boolean. If TRUE, the initial values of the autoregressive model are set to zero. The default is FALSE.
diffuse.algorithm	A character string specifying the algorithm used for diffuse initialization. The default is "SqrtDiffuse". Other options are: "Diffuse" and "Augmented".
diffuse.regressors	Boolean. Indicates whether the coefficients of the regression model are treated as diffuse (TRUE) or as fixed unknown (FALSE, the default).

nbcsts An integer specifying the number of backcast periods. This argument is ignored when one or more indicator series is provided.

nfcsts An integer specifying the number of forecast periods. This argument is ignored when one or more indicator series is provided.

Value

An object of class "JD3_INTERPRAW_RSLTS" is returned. The following are returned invisibly as a list:

- regression [[1]] regression coefficients;
- estimation [[2]] interpolated values and standard deviation, regression effects, smoothing part, parameter and residuals;
- likelihood [[3]] likelihood statistics.

See Also

temporal_disaggregation_raw()

For more information, see the vignette:

utils::browseVignettes(), e.g. browseVignettes(package = "rjd3bench")

Examples

```
# Use of Chow-lin method to interpolate a biennial series with an annual
# indicator (the low-frequency series is consistent with the last value of the
# interpolated series)
Y <- stats::aggregate(rjd3toolkit::Retail$RetailSalesTotal, 0.5)
x <- stats::aggregate(rjd3toolkit::Retail$FoodAndBeverageStores, 1)
ti <- temporal_interpolation_raw(as.numeric(Y), indicators = as.numeric(x), freqratio = 2)
ti$estimation$interp

# Use of Fernandez method to interpolate a series without indicator
# considering a frequency ratio of 5 (the low-frequency series is consistent
# with the last value of the interpolated series). For example, Y2 could be a
# quinquennial series to interpolate annually.
Y2 <- c(500,510,525,520)
ti2 <- temporal_interpolation_raw(Y2, model = "Rw", freqratio = 5, nbcsts = 1, nfcsts = 2)
ti2$estimation$interp

# Same with an indicator, considering an offset in the latter
Y2 <- c(500,510,525,520)
x2 <- c(485,
        490, 492.5, 497.5, 520, 495,
        500, 502.5, 505, 527.5, 515,
        522.5, 517.5, 522.5, 545, 520,
        535, 515, 540, 565, 550)
ti3 <- temporal_interpolation_raw(Y2, indicators = x2, startoffset = 1, model = "Rw", freqratio = 5)
ti3$estimation$interp

# Same considering that the first value of the interpolated series is the one
```

```
# consistent with the low-frequency series
ti4 <- temporal_interpolation_raw(Y2, indicators = x2, startoffset = 1,
                                  model = "Rw", freqratio = 5, obsposition = 1)
ti4$estimation$interp
```

`temporaldisaggregationI` *Temporal Disaggregation and Interpolation of a Time Series by means of a Reverse Regression Model.*

Description

Perform temporal disaggregation and interpolation of low-frequency to high frequency time series by means of a reverse regression model. Unlike the usual regression-based models, this approach treats a high-frequency indicator as the dependent variable and the unknown target series as the independent variable.

Usage

```
temporaldisaggregationI(
  series,
  indicator,
  conversion = c("Sum", "Average", "Last", "First", "UserDefined"),
  conversion.obsposition = 1L,
  rho = 0,
  rho.fixed = FALSE,
  rho.truncated = 0
)
```

Arguments

`series`	A low-frequency time series to be disaggregated or interpolated. It must be a `"ts"` object.
`indicator`	A high-frequency indicator series. It must be a `"ts"` object.
`conversion`	A character string specifying the conversion mode, typically `"Sum"`(the default) or `"Average"`. Other options are: `"Last"`, `"First"` and `"UserDefined"`.
`conversion.obsposition`	An integer specifying the position of the low-frequency observations within the interpolated series (e.g. the 7th month of the year). This argument is used only for interpolation when `conversion = "UserDefined"`.
`rho`	A numeric value giving the (initial) value of the autoregressive parameter.
`rho.fixed`	Boolean. Specifies whether the supplied value of `rho` is fixed. The default is `FALSE`, which indicates that `rho` is estimated.
`rho.truncated`	A numeric value defining the lower bound of the admissible range for `rho`. The evaluation range is `[rho.truncated, 1[`.

Value

An object of class "JD3_TEMPDISAGGI_RSLTS" is returned. The following are returned invisibly as a list:

- `regression [[1]]` regression coefficients;
- `estimation [[2]]` disaggregated Time-Series and parameter;
- `likelihood [[3]]` likelihood statistics.

References

Bournay J., Laroque G. (1979). Reflexions sur la methode d'elaboration des comptes trimestriels. Annales de l'Insee, n. 36, pp.3-30.

See Also

For more information, see the vignette:

`utils::browseVignettes()`, e.g. `browseVignettes(package = "rjd3bench")`

Examples

```
# Retail data, monthly indicator
Y <- rjd3toolkit::aggregate(rjd3toolkit::Retail$RetailSalesTotal, 1)
x <- rjd3toolkit::Retail$FoodAndBeverageStores
td <- temporaldisaggregationI(Y, indicator = x)
td$estimation$disagg

# qna data, quarterly indicator
data("qna_data")
Y <- ts(qna_data$B1G_Y_data[,"B1G_CE"], frequency = 1, start = c(2009,1))
x <- ts(qna_data$TURN_Q_data[,"TURN_INDEX_CE"], frequency = 4, start = c(2009,1))
td <- temporaldisaggregationI(Y, indicator = x)
td$regression$a
td$regression$b
```

Index

∗ **datasets**
qna_data, 20

adl_disaggregation, 2

calendarization, 4
cholette, 6
cubicspline, 8

denton, 9
denton_modelbased, 11
denton_raw, 13
deprecated-rjd3bench, 15

grp, 15

multivariatecholette, 17

qna_data, 20

temporal_disaggregation, 21
temporal_disaggregation_raw, 23
temporal_interpolation, 26
temporal_interpolation_raw, 28
temporaldisaggregation
*(*deprecated-rjd3bench*)*, 15
temporaldisaggregationI, 31