Package: rjd3bench (via r-universe)

October 29, 2024

Type Package

Title Interface to 'JDemetra+ 3.x' time series analysis software

Version 2.1.0.9000

Description R Interface to 'JDemetra+ 3.x'
(<https://github.com/jdemetra>) time series analysis software.

Depends R (>= 4.1.0)

Imports rJava (>= 1.0-6), rjd3toolkit (>= 3.2.4), RProtoBuf (>=
0.4.20)

Remotes github::rjdverse/rjd3toolkit

SystemRequirements Java (>= 17)

License EUPL | file LICENSE

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

LazyData TRUE

Suggests knitr, rmarkdown

RoxygenNote 7.3.2

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

Encoding UTF-8

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

NeedsCompilation no

VignetteBuilder knitr

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

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

RemoteRef HEAD

RemoteSha 16037ba1614de8ea14580590ff3c03ef1aee9e0d

Contents

`adl_disaggregation` *Temporal disaggregation of a time series with ADL models*

Description

Temporal disaggregation of a time series with ADL models

Usage

```
adl_disaggregation(
  series,
  constant = TRUE,
  trend = FALSE,
  indicators = NULL,
  conversion = c("Sum", "Average", "Last", "First", "UserDefined"),
  conversion.obsposition = 1,
  phi = 0,
  phi.fixed = FALSE,
  phi.truncated = 0,
  xar = c("FREE", "SAME", "NONE")
)
```

Arguments

xar

Examples

```
# qna data, fernandez with/without 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<-rjd3bench::adl_disaggregation(Y, indicators=x, xar="FREE")
td2<-rjd3bench::adl_disaggregation(Y, indicators=x, xar="SAME")
```

---

calendarization *Calendarization*

Description

Based on "Calendarization with splines and state space models" B. Quenneville, F.Picard and S.Fortier Appl. Statistics (2013) 62, part 3, pp 371-399. State space implementation.

Usage

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

Arguments

calendarobs	Observations (list of start, end, value). See the example.
freq	Annual frequency. If 0, only the daily series are computed
start	Starting day of the calendarization. Could be before the calendar obs (extrapolation)
end	Final day of the calendarization. Could be after the calendar obs (extrapolation)
dailyweights	Daily weights. Should have the same length as the requested series
stde	

Examples

```
obs<-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<-calendarization(obs, 4, end="2003-12-31", stde=TRUE)
Q<-cal$rslt
eQ<-cal$erslt
```

`cholette` *Cholette method*

Description

Benchmarking by means of the Cholette method.

Usage

```
cholette(
  s,
  t,
  rho = 1,
  lambda = 1,
  bias = "None",
  conversion = "Sum",
  obsposition = 1
)
```

Arguments

`s`	Disaggregated series. Mandatory
`t`	Aggregation constraint. Mandatory
`obsposition`	Postion of the observation in the aggregated period (only used with "UserDefined" conversion)

Details

$$\sum_{i,t}\left(\left(\frac{x_{i,t}-z_{i,t}}{|z_{i,t}|^{\lambda}}\right)-\rho\left(\frac{x_{i,t-1}-z_{i,t-1}}{|z_{i,t-1}|^{\lambda}}\right)\right)^2$$

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 sub-period indicator (or disaggregated series) is used, cubic splines are no longer drawn based on the low frequency data 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 indicator.

Usage

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

Arguments

s	Disaggregated series. If not NULL, it must be the same class as t.
t	Aggregation constraint. Mandatory. it must be either an object of class ts or a numeric vector.
nfreq	Annual frequency of the disaggregated variable. Used if no disaggregated series is provided.
conversion	Conversion rule. Usually "Sum" or "Average". Sum by default.
obsposition	Postion of the observation in the aggregated period (only used with "UserDefined" conversion)

Value

The benchmarked series is returned

Examples

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

# cubic spline without disaggregated series
y1<-rjd3bench::cubicspline(t=Y, nfreq=4)

# cubic spline with disaggregated series
x1<-y1+rnorm(n=length(y1), mean=0, sd=10)
```

```
y2<-rjd3bench::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))
y3<-rjd3bench::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 = 1,
  mul = TRUE,
  nfreq = 4,
  modified = TRUE,
  conversion = c("Sum", "Average", "Last", "First", "UserDefined"),
  obsposition = 1
)
```

Arguments

`s`	Preliminary series. If not NULL, it must be the same class as t.
`t`	Aggregation constraint. Mandatory. it must be either an object of class ts or a numeric vector.
`d`	Differencing order. 1 by default.
`mul`	Multiplicative or additive benchmarking. Multiplicative by default.
`nfreq`	Annual frequency of the disaggregated variable. Used if no disaggregated series is provided.
`modified`	Modified (TRUE) or unmodified (FALSE) Denton. Modified by default.
`conversion`	Conversion rule. Usually "Sum" or "Average". Sum by default.
`obsposition`	Position of the observation in the aggregated period (only used with "UserDefined" conversion).

Value

The benchmarked series is returned

Examples

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

# denton PFD without high frequency series
y1 <- rjd3bench::denton(t=Y, nfreq=4)

# denton PFD with high frequency series
x <- y1 + rnorm(n=length(y1), mean=0, sd=10)
y2 <- rjd3bench::denton(s=x, t=Y)

# denton ASD
y3 <- rjd3bench::denton(s=x, t=Y, d=2, mul=FALSE)
```

---

denton_modelbased *Temporal disaggregation of a time series by model-based Denton proportional method*

Description

Denton proportional method can be expressed as a statistical model in a State space representation (see documentation for the definition of states). This approach is interesting as it allows more flexibility in the model such as the inclusion of outliers (level shift in the Benchmark to Indicator ratio) that could otherwise induce unintended wave effects with standard Denton method. Outliers and their intensity are defined by changing the value of the 'innovation variances'.

Usage

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

Arguments

series	Aggregation constraint. Mandatory. It must be either an object of class ts or a numeric vector.
indicator	High-frequency indicator. Mandatory. It must be of same class as series
differencing	Not implemented yet. Keep it equals to 1 (Denton PFD method).
conversion	Conversion rule. Usually "Sum" or "Average". Sum by default.

conversion.obsposition
: Position of the observation in the aggregated period (only used with "UserDefined" conversion)

outliers
: a list of structured definition of the outlier periods and their intensity. The period must be submitted first in the format YYYY-MM-DD and enclosed in quotation marks. This must be followed by an equal sign and the intensity of the outlier, defined as the relative value of the 'innovation variances' (1= normal situation)

fixedBIratios
: a list of structured definition of the periods where the BI ratios must be fixed. The period must be submitted first in the format YYYY-MM-DD and enclosed in quotation marks. This must be followed by an equal sign and the value of the BI ratio.

Value

an object of class 'JD3MBDenton'

Examples

```
# retail data, monthly indicator
Y<-rjd3toolkit::aggregate(rjd3toolkit::retail$RetailSalesTotal, 1)
x<-rjd3toolkit::aggregate(rjd3toolkit::retail$FoodAndBeverageStores, 4)
td<-rjd3bench::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<-rjd3bench::denton_modelbased(Y, x)
td2<-rjd3bench::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
## Not run:
ts.plot(bi1,bi2,gpars=list(col=c("red","blue")))
ts.plot(y1,y2,gpars=list(col=c("red","blue")))

## End(Not run)
```

---

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,
  conversion = c("Sum", "Average", "Last", "First", "UserDefined"),
  obsposition = 1,
  eps = 1e-12,
  iter = 500,
  dentoninitialization = TRUE
)
```

Arguments

s	Preliminary series. Mandatory. It must be a ts object.
t	Aggregation constraint. Mandatory. It must be a ts object.
conversion	Conversion rule. "Sum" by default.
obsposition	Position of the observation in the aggregated period (only used with "UserDefined" conversion)
eps	Numeric. Defines the convergence precision. BFGS algorithm is run until the reduction in the objective is within this eps value (1e-12 is the default) or until the maximum number of iterations is hit.
iter	Integer. Maximum number of iterations in BFGS algorithm (500 is the default).
dentoninitialization	indicate whether the series benchmarked via modified Denton PFD is used as starting values of the GRP optimization procedure (TRUE/FALSE, TRUE by default). If FALSE, the average benchmark is used for flow variables (e.g. t/4 for quarterly series with annual constraints and conversion = 'Sum'), or the benchmark for stock variables.

Value

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*.

Examples

```
data("qna_data")
Y <- ts(qna_data$B1G_Y_data[,"B1G_FF"], frequency=1, start=c(2009,1))
x <- rjd3bench::denton(t=Y, nfreq=4) + rnorm(n=length(Y)*4, mean=0, sd=10)
y_grp <- rjd3bench::grp(s=x, t=Y)
```

---

multivariatecholette *Multi-variate Cholette*

Description

Multi-variate Cholette

Usage

```
multivariatecholette(
  xlist,
  tcvector = NULL,
  ccvector = NULL,
  rho = 1,
  lambda = 1
)
```

Arguments

lambda

---

plot.JD3AdlDisagg *Plot function for object of class JD3AdlDisagg*

Description

Plot function for object of class JD3AdlDisagg

Usage

```
## S3 method for class 'JD3AdlDisagg'
plot(x, ...)
```

Arguments

x	an object of class JD3AdlDisagg
...	further arguments to pass to ts.plot.

Examples

```
Y<-rjd3toolkit::aggregate(rjd3toolkit::retail$RetailSalesTotal, 1)
x<-rjd3toolkit::retail$FoodAndBeverageStores
td<-rjd3bench::adl_disaggregation(Y, indicator=x, xar="FREE")
plot(td)
```

---

plot.JD3MBDenton *Plot function for object of class JD3MBDenton*

Description

Plot function for object of class JD3MBDenton

Usage

```
## S3 method for class 'JD3MBDenton'
plot(x, ...)
```

Arguments

x	an object of class JD3MBDenton
...	further arguments to pass to ts.plot.

Examples

```
Y<-rjd3toolkit::aggregate(rjd3toolkit::retail$RetailSalesTotal, 1)
x<-rjd3toolkit::retail$FoodAndBeverageStores
td<-rjd3bench::temporaldisaggregationI(Y, indicator=x)
plot(td)
```

---

plot.JD3TempDisagg *Plot function for object of class JD3TempDisagg*

Description

Plot function for object of class JD3TempDisagg

Usage

```
## S3 method for class 'JD3TempDisagg'
plot(x, ...)
```

Arguments

x	an object of class JD3TempDisagg
...	further arguments to pass to ts.plot.

Examples

```
Y<-rjd3toolkit::aggregate(rjd3toolkit::retail$RetailSalesTotal, 1)
x<-rjd3toolkit::retail$FoodAndBeverageStores
td<-rjd3bench::temporaldisaggregation(Y, indicator=x)
plot(td)
```

---

plot.JD3TempDisaggI *Plot function for object of class JD3TempDisaggI*

Description

Plot function for object of class JD3TempDisaggI

Usage

```
## S3 method for class 'JD3TempDisaggI'
plot(x, ...)
```

Arguments

x	an object of class JD3TempDisaggI
...	further arguments to pass to ts.plot.

Examples

```
Y<-rjd3toolkit::aggregate(rjd3toolkit::retail$RetailSalesTotal, 1)
x<-rjd3toolkit::retail$FoodAndBeverageStores
td<-rjd3bench::temporaldisaggregationI(Y, indicator=x)
plot(td)
```

print.JD3AdlDisagg *Print function for object of class JD3AdlDisagg*

Description

Print function for object of class JD3AdlDisagg

Usage

```
## S3 method for class 'JD3AdlDisagg'
print(x, ...)
```

Arguments

x	an object of class JD3AdlDisagg

Examples

```
Y<-rjd3toolkit::aggregate(rjd3toolkit::retail$RetailSalesTotal, 1)
x<-rjd3toolkit::retail$FoodAndBeverageStores
td<-rjd3bench::adl_disaggregation(Y, indicator=x, xar="FREE")
print(td)
```

print.JD3MBDenton *Print function for object of class JD3MBDenton*

Description

Print function for object of class JD3MBDenton

Usage

```
## S3 method for class 'JD3MBDenton'
print(x, ...)
```

Arguments

x	an object of class JD3MBDenton
...	further arguments passed to or from other methods.

Examples

```
Y<-rjd3toolkit::aggregate(rjd3toolkit::retail$RetailSalesTotal, 1)
x<-rjd3toolkit::aggregate(rjd3toolkit::retail$FoodAndBeverageStores, 4)
td<-rjd3bench::denton_modelbased(Y, x,
                                 outliers = list("2000-01-01"=100, "2005-07-01"=100))
print(td)
```

print.JD3TempDisagg *Print function for object of class JD3TempDisagg*

Description

Print function for object of class JD3TempDisagg

Usage

```
## S3 method for class 'JD3TempDisagg'
print(x, ...)
```

Arguments

x	an object of class JD3TempDisagg
...	further arguments passed to or from other methods.

Examples

```
Y<-rjd3toolkit::aggregate(rjd3toolkit::retail$RetailSalesTotal, 1)
x<-rjd3toolkit::retail$FoodAndBeverageStores
td<-rjd3bench::temporaldisaggregation(Y, indicator=x)
print(td)
```

print.JD3TempDisaggI *Print function for object of class JD3TempDisaggI*

Description

Print function for object of class JD3TempDisaggI

Usage

```
## S3 method for class 'JD3TempDisaggI'
print(x, ...)
```

Arguments

x	an object of class JD3TempDisaggI
...	further arguments passed to or from other methods.

Examples

```
Y<-rjd3toolkit::aggregate(rjd3toolkit::retail$RetailSalesTotal, 1)
x<-rjd3toolkit::retail$FoodAndBeverageStores
td<-rjd3bench::temporaldisaggregationI(Y, indicator=x)
print(td)
```

---

summary.JD3AdlDisagg *Summary function for object of class JD3AdlDisagg*

Description

Summary function for object of class JD3AdlDisagg

Usage

```
## S3 method for class 'JD3AdlDisagg'
summary(object, ...)
```

Arguments

object	an object of class JD3AdlDisagg
...	further arguments passed to or from other methods.

Examples

```
Y<-rjd3toolkit::aggregate(rjd3toolkit::retail$RetailSalesTotal, 1)
x<-rjd3toolkit::retail$FoodAndBeverageStores
td<-rjd3bench::adl_disaggregation(Y, indicator=x)
summary(td)
```

---

summary.JD3MBDenton *Summary function for object of class JD3MBDenton*

Description

Summary function for object of class JD3MBDenton

Usage

```
## S3 method for class 'JD3MBDenton'
summary(object, ...)
```

Arguments

object	an object of class JD3MBDenton
...	further arguments passed to or from other methods.

Examples

```
Y<-rjd3toolkit::aggregate(rjd3toolkit::retail$RetailSalesTotal, 1)
x<-rjd3toolkit::aggregate(rjd3toolkit::retail$FoodAndBeverageStores, 4)
td<-rjd3bench::denton_modelbased(Y, x,
                                 outliers = list("2000-01-01"=100, "2005-07-01"=100))
summary(td)
```

---

summary.JD3TempDisagg *Summary function for object of class JD3TempDisagg*

Description

Summary function for object of class JD3TempDisagg

Usage

```
## S3 method for class 'JD3TempDisagg'
summary(object, ...)
```

Arguments

object	an object of class JD3TempDisagg
...	further arguments passed to or from other methods.

Examples

```
Y<-rjd3toolkit::aggregate(rjd3toolkit::retail$RetailSalesTotal, 1)
x<-rjd3toolkit::retail$FoodAndBeverageStores
td<-rjd3bench::temporaldisaggregation(Y, indicator=x)
summary(td)
```

summary.JD3TempDisaggI

Summary function for object of class JD3TempDisaggI

Description

Summary function for object of class JD3TempDisaggI

Usage

```
## S3 method for class 'JD3TempDisaggI'
summary(object, ...)
```

Arguments

object	an object of class JD3TempDisaggI
...	further arguments passed to or from other methods.

Examples

```
Y<-rjd3toolkit::aggregate(rjd3toolkit::retail$RetailSalesTotal, 1)
x<-rjd3toolkit::retail$FoodAndBeverageStores
td<-rjd3bench::temporaldisaggregationI(Y, indicator=x)
summary(td)
```

temporaldisaggregation

Temporal disaggregation of a time series by regression models.

Description

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

Usage

```
temporaldisaggregation(
  series,
  constant = TRUE,
  trend = FALSE,
  indicators = NULL,
  model = c("Ar1", "Rw", "RwAr1"),
  freq = 4,
  conversion = c("Sum", "Average", "Last", "First", "UserDefined"),
```

```
  conversion.obsposition = 1,
  rho = 0,
  rho.fixed = FALSE,
  rho.truncated = 0,
  zeroinitialization = FALSE,
  diffuse.algorithm = c("SqrtDiffuse", "Diffuse", "Augmented"),
  diffuse.regressors = FALSE
)
```

Arguments

series	The time series that will be disaggregated. It must be a ts object.
constant	Constant term (T/F). Only used with Ar1 model when zeroinitialization=F
trend	Linear trend (T/F)
indicators	High-frequency indicator(s) used in the temporal disaggregation. It must be a (list of) ts object(s).
model	Model of the error term (at the disaggregated level). "Ar1" = Chow-Lin, "Rw" = Fernandez, "RwAr1" = Litterman
freq	Annual frequency of the disaggregated variable. Used if no indicator is provided
conversion	Conversion mode (Usually "Sum" or "Average")
conversion.obsposition	Only used with "UserDefined" mode. Position of the observed indicator in the aggregated periods (for instance 7th month of the year)
rho	Only used with Ar1/RwAr1 models. (Initial) value of the parameter
rho.fixed	Fixed rho (T/F, F by default)
rho.truncated	Range for Rho evaluation (in [rho.truncated, 1[)
zeroinitialization	The initial values of an auto-regressive model are fixed to 0 (T/F, F by default)
diffuse.algorithm	Algorithm used for diffuse initialization. "SqrtDiffuse" by default
diffuse.regressors	Indicates if the coefficients of the regression model are diffuse (T) or fixed unknown (F, default)

Value

An object of class "JD3TempDisagg"

Examples

```
# retail data, chow-lin with monthly indicator
Y<-rjd3toolkit::aggregate(rjd3toolkit::retail$RetailSalesTotal, 1)
x<-rjd3toolkit::retail$FoodAndBeverageStores
td<-rjd3bench::temporaldisaggregation(Y, indicators=x)
y<-td$estimation$disagg
```

```
# qna data, fernandez with/without 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<-rjd3bench::temporaldisaggregation(Y, indicators=x, model = "Rw")
td2<-rjd3bench::temporaldisaggregation(Y, model = "Rw")
mod1<- td1$regression$model
```

`temporaldisaggregationI` *Temporal disaggregation using the model: x(t) = a + b y(t), where x(t) is the indicator, y(t) is the unknown target series, with low-frequency constraints on y.*

Description

Temporal disaggregation using the model: x(t) = a + b y(t), where x(t) is the indicator, y(t) is the unknown target series, with low-frequency constraints on y.

Usage

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

Arguments

`series`	The time series that will be disaggregated. It must be a ts object.
`indicator`	High-frequency indicator used in the temporal disaggregation. It must be a ts object.
`conversion`	Conversion mode (Usually "Sum" or "Average")
`conversion.obsposition`	Only used with "UserDefined" mode. Position of the observed indicator in the aggregated periods (for instance 7th month of the year)
`rho`	Only used with Ar1/RwAr1 models. (Initial) value of the parameter
`rho.fixed`	Fixed rho (T/F, F by default)
`rho.truncated`	Range for Rho evaluation (in [rho.truncated, 1[)

Value

An object of class "JD3TempDisaggI"

Examples

```
# retail data, monthly indicator
Y<-rjd3toolkit::aggregate(rjd3toolkit::retail$RetailSalesTotal, 1)
x<-rjd3toolkit::retail$FoodAndBeverageStores
td<-rjd3bench::temporaldisaggregationI(Y, indicator=x)
y<-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<-rjd3bench::temporaldisaggregationI(Y, indicator=x)
a<-td$regression$a
b<-td$regression$b
```

Index