Package 'disaggR'

October 12, 2020

Type Package

Title Two-Steps Benchmarks for Time Series Disaggregation

Version 0.1.9

Date 2020-10-09

Author Arnaud Feldmann

Maintainer Arnaud Feldmann <arnaud.feldmann@gmail.com>

Description The twoStepsBenchmark() function and its wrappers allow you to disaggregate a low frequency time serie with time series of higher frequency, using the French National Accounts methodology.
The aggregated sum of the resulting time-serie is strictly equal to the low-frequency serie within the benchmarking window.
Typically, the low frequency serie is an annual one, unknown for the last year, and the high frequency is either quarterly or mensual.
See ``Methodology of quarterly national accounts", Insee Méthodes N°126, by Insee (2012, ISBN:978-2-11-068613-8).

Imports ggplot2, Rcpp

LinkingTo Rcpp

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.1.1

Suggests testthat (>= 2.1.0), vdiffr

Depends R (>= 2.10)

BugReports https://github.com/arnaud-feldmann/disaggR/issues

LazyData yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-10-11 23:30:02 UTC

R topics documented:

bflSmooth 2
in_sample 3
model.list 4
prais 4
reUseBenchmark 5
rho 6
se 7
smoothed.part 7
twoStepsBenchmark 8

Index **11**

`bflSmooth` *Smooth a time serie*

Description

bflSmooth smoothes a time-serie into a time serie of a higher frequency that exactly aggregate into the higher one. The process followed is Boot, Feibes and Lisman, which minimizes the squares of the variations.

Usage

```
bflSmooth(lfserie, nfrequency, weights = NULL, lfserie.is.rate = FALSE)
```

Arguments

`lfserie`	a time-serie to be smoothed
`nfrequency`	the new high frequency. It must be a multiple of the low frequency.
`weights`	NULL or a time-serie of the same size than the expected high-frequency serie.
`lfserie.is.rate`	TRUE or FALSE. only means a thing if weights isn't NULL.

Details

If `weights` isn't `NULL` the results depends of `lfserie.is.rate` :

- if `FALSE` the rate output/weights is smoothed with the constraint that the aggregated output is equal to lfserie.
- if `TRUE` the output is the rate to be smoothed, and the output is the aggregated rate.

Value

A time serie of frequency nfrequency

Author(s)

Arnaud Feldmann

`in_sample` *Producing the in sample predictions of a prais-lm regression*

Description

The function `in_sample` returns in-sample predictions from a praislm or a twoStepsBenchmark object.

Usage

```
in_sample(object, type = "changes")
```

Arguments

`object`	an object of class `praislm` or `twoStepsBenchmark`
`type`	"changes" or "levels". The results are either returned in changes or in levels.

Details

The predicted values are different from the fitted values :

- they are eventually reintegrated
- the autocorrelated part of the residuals is added Besides, changes are relative to the latest response value, not the latest predicted value.

Value

a named matrix time-serie of two columns, one for the response and the other for the predicted value. A `insample` class is added to the object. Then, the functions `plot` and `autoplot` (the latter requires to load **ggplot2**) can be used to produce graphics.

Examples

```
benchmark <- twoStepsBenchmark(turnover,construction,include.rho = TRUE)
in_sample(benchmark)
```

model.list *Extracting all the arguments submitted to generate an object*

Description

The function `model.list` returns the arguments submitted to the function praislm or a twoStepsBenchmark.

Usage

```
model.list(object)
```

Arguments

`object`	a praislm or twoStepsBenchmark object.

Details

These are returned as they are after evaluation, model.list doesn't return a call.

Value

a list containing every evaluated arguments

Examples

```
benchmark <- twoStepsBenchmark(turnover,construction); model.list(benchmark)
```

prais *Extracting the regression of a twoStepsBenchmark*

Description

prais is a function which extracts the regression, a praislm object, of a twoStepsBenchmark.

Usage

```
prais(x)
```

Arguments

`x`	a twoStepsBenchmark

Value

prais returns an object of class "praislm".

The functions that can be used on that class are almost the same than for the class twoStepsBenchmark. summary, coefficients, residuals will return the same values. However, as for fitted.values, the accessor returns the fitted values of the regression, not the high-frequency, eventually integrated, time-serie contained in a twoStepsBenchmark.

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

coefficients	a named vector of coefficients.
residuals	the residuals, that is response minus fitted values.
fitted.values	a time-serie, the fitted mean values
se	a named vector of standard errors.
df.residuals	the residual degrees of freedom.
rho	the autocorrelation coefficients of the residuals. It is equal to zero if twoStepsBenchmark was called with include.rho=FALSE
residuals.decorrelated	the residuals of the model after having been transformed by rho in a least square model.
fitted.values.decorrelated	the fitted values of the model after having been transformed by rho in a least square model.

Examples

```
benchmark <- twoStepsBenchmark(turnover,construction); prais(benchmark)
```

---

reUseBenchmark *Using the same estimated benchmark model on another time-serie*

Description

This function reapplies the coefficients and parameters of a benchmark on new time-serie.

Usage

```
reUseBenchmark(hfserie,benchmark,reeval.smoothed.part=FALSE)
```

Arguments

hfserie	the bended time-serie. If it is a matrix time-serie, it has to have the same column names than the hfserie used for the benchmark.
benchmark	a twoStepsBenchmark object, from which the parameters and coefficients are taken
reeval.smoothed.part	a boolean of length 1. If TRUE, the smoothed part is reevaluated, hence the aggregated benchmarked serie is equal to the low-frequency serie.

Details

reUseBenchmark is primarily meant to be used on a serie that is derived from the previous one, after some modifications that would bias the estimation otherwise. Working-day adjustment is a good example. Hence, by default, the smoothed part of the first model isn't reevaluated ; the aggregated benchmarked serie isn't equal to the low-frequency serie.

Value

twoStepsBenchark returns an object of class twoStepsBenchmark.

Examples

```
benchmark <- twoStepsBenchmark(turnover,construction)
turnover_modif <- turnover
turnover_modif[2] <- turnover[2]+2
benchmark2 <- reUseBenchmark(turnover_modif,benchmark)
```

---

rho *Extracting the autocorrelation parameter*

Description

The function `rho` returns the autocorrelation parameter from either a praislm or a twoStepsBenchmark object. If `include.rho` is `FALSE`, `rho` returns zero.

Usage

```
rho(object)
```

Arguments

`object`	a praislm or twoStepsBenchmark object.

Value

a double of length 1.

Examples

```
benchmark <- twoStepsBenchmark(turnover,construction,include.rho = TRUE); rho(benchmark)
```

se *Extracting the standard error*

Description

The function se returns the standard error the coefficients from either a praislm or a twoStepsBenchmark object.

Usage

```
se(object)
```

Arguments

object	a praislm or twoStepsBenchmark object.

Value

a double, that is named the same way that the coefficients are. If some coefficients are set by the user, they return NA as for their standard error.

smoothed.part *Extracting the smoothed part of a twoStepsBenchmark*

Description

The function smoothed.part returns the smoothed part of a twoStepsBenchmark. It derives from the residuals of the aggregated regression, with some differences :

- it is eventually integrated if include.differenciation=TRUE.
- it is extrapolated to match the domain window.
- it is smoothed with an additive Denton benchmark.

Usage

```
smoothed.part(object)
```

Arguments

object	a twoStepsBenchmark object.

Value

a time-serie

Examples

```
benchmark <- twoStepsBenchmark(turnover,construction); smoothed.part(benchmark)
```

`twoStepsBenchmark` *Bends a time-serie with a lower frequency one*

Description

twoStepsBenchmark bends a time-serie with a time-serie of a lower frequency. The procedure involved is a Prais-Winsten regression, then an additive Denton benchmark. annualBenchmark is a wrapper of the main function, that applies more specifically to annual series, and changes the default window parameters to the ones that are commonly used by quarterly national accounts.

Usage

```
twoStepsBenchmark(hfserie,lfserie,include.differenciation=FALSE,include.rho=FALSE,
                  set.coeff=NULL,set.const=NULL,
                  start.coeff.calc=NULL,end.coeff.calc=NULL,
                  start.benchmark=NULL,end.benchmark=NULL,
                  start.domain=NULL,end.domain=NULL,...)

annualBenchmark(hfserie,lfserie,include.differenciation=FALSE,include.rho=FALSE,
                set.coeff=NULL,set.const=NULL,
              start.coeff.calc=start(lfserie)[1],end.coeff.calc=end(lfserie)[1],
               start.benchmark=start(lfserie)[1],end.benchmark=end.coeff.calc+1,
                start.domain=start(hfserie),
                end.domain=c(end.benchmark+2,frequency(hfserie)))
```

Arguments

`hfserie`	the bended time-serie. It can be a matrix time-serie.
`lfserie`	a time-serie whose frequency divides the frequency of `hfserie`.
`include.differenciation`	a boolean of length 1. If `TRUE`, `lfserie` and `hfserie` are differenced before the estimation of the regression.
`include.rho`	a boolean of length 1. If `TRUE`, the regression includes an autocorrelation parameter for the residuals. The applied procedure is a Prais-Winsten estimation.
`set.coeff`	an optional double, that allows the user to set the regression coefficients instead of evaluating them. If `hfserie` is a matrix, each column initializes a coefficient with the same name as the column name. Hence, `set.coeff` has to be a named double, which will optionally set some coefficients instead of evaluating them.
`set.const`	an optional double of length 1, that sets the regression constant. The constant is actually an automatically added column to `hfserie`. Using `set.constant=3` is equivalent to using `set.coeff=c(constant=3)`.

`start.coeff.calc`
: an optional start for the estimation of the coefficients of the regression. Should be a double or a numeric of length 2, like a window for `lfserie`. If NULL, the start is defined by lfserie's window.

`end.coeff.calc`
: an optional end for the estimation of the coefficients of the regression. Should be a double or a numeric of length 2, like a window for `lfserie`. If NULL, the end is defined by lfserie's window.

`start.benchmark`
: an optional start for `lfserie` to bend `hfserie`. Should be a double or a numeric of length 2, like a window for `lfserie`. If NULL, the start is defined by lfserie's window.

`end.benchmark`
: an optional end for `lfserie` to bend `hfserie`. Should be a double or a numeric of length 2, like a window for `lfserie`. If NULL, the start is defined by lfserie's window.

`start.domain`
: the start of the output high-frequency serie. It also defines the smoothing window : The low-frequency residuals will be extrapolated until they contain the smallest low-frequency window that is around the high-frequency domain window. Should be a double or a numeric of length 2, like a window for `hfserie`. If NULL, the start is defined by hfserie's window.

`end.domain`
: the end of the output high-frequency serie. It also defines the smoothing window : The low-frequency residuals will be extrapolated until they contain the smallest low-frequency window that is around the high-frequency domain window. Should be a double or a numeric of length 2, like a window for `hfserie`. If NULL, the start is defined by hfserie's window.

`...`
: if the dots contain a cl item, its value overwrites the value of the returned call. This feature allows to build wrappers.

Value

twoStepsBenchark returns an object of class "`twoStepsBenchmark`".

The function `summary` can be used to obtain and print a summary of the regression used by the benchmark. The functions `plot` and `autoplot` (the latter requires to load **ggplot2**) produces graphics of the benchmarked serie and the bending serie. The function in_sample produces in-sample predictions with the inner regression. The generic accessor functions `as.ts`, `prais`, `coefficients`, `residuals`, `fitted.values`, `model.list`, `se`, `rho` extract various useful features of the returned value.

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

`benchmarked.serie`
: a time-serie, that is the result of the benchmark.

`fitted.values`
: a time-serie, that is the high-frequency serie as it is after having applied the regression coefficients. The difference `benchmarked.serie` - `fitted.values` is then a smoothed residual, eventually integrated if `include.differenciation=TRUE`.

`regression`
: an object of class praislm, it is the regression on which relies the benchmark. It can be extracted with the function prais

`smoothed.part`
: the smoothed part of the two-steps benchmark.

model.list	a list containing all the arguments submitted to the function.
call	the matched call (either of twoStepsBenchmark or annualBenchmark)

Examples

```
## How to use annualBenchmark or twoStepsBenchark

benchmark <- annualBenchmark(hfserie = turnover,
                             lfserie = construction,
                             include.differenciation = TRUE)
as.ts(benchmark)
coef(benchmark)
summary(benchmark)
library(ggplot2)
autoplot(in_sample(benchmark))

## How to manually set the coefficient

benchmark2 <- annualBenchmark(hfserie = turnover,
                              lfserie = construction,
                              include.differenciation = TRUE,
                              set.coeff = 0.1)
coef(benchmark2)
```

Index

annualBenchmark *(*twoStepsBenchmark*)*, 8

bflSmooth, 2

in_sample, 3, *9*

model.list, 4

prais, 4, *9*
praislm, *3, 4*, *6, 7*
praislm *(*prais*)*, 4

reUseBenchmark, 5
rho, 6

se, 7
smoothed.part, 7

twoStepsBenchmark, *3, 4*, *6, 7*, 8