# Package ‘disaggR’

November 4, 2025

**Type** Package

**Title** Two-Steps Benchmarks for Time Series Disaggregation

**Version** 1.0.5.4

**Description** The twoStepsBenchmark() and threeRuleSmooth() functions allow you to disaggregate a low-frequency time series with higher frequency time series, using the French National Accounts methodology. The aggregated sum of the resulting time series is strictly equal to the low-frequency time series within the benchmarking window. Typically, the low-frequency time series is an annual one, unknown for the last year, and the high frequency one is either quarterly or monthly. See ``Methodology of quarterly national accounts'', Insee Méthodes N°126, by Insee (2012, ISBN:978-2-11-068613-8, <https://www.insee.fr/en/information/2579410>).

**Imports** graphics, grDevices, methods, RColorBrewer (>= 1.1-2), stats, utils

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.1

**Suggests** knitr, ggplot2 (>= 3.0.0), rmarkdown (>= 2.0.0), shiny (>= 1.5.0), shinytest2 (>= 0.1.0), testthat (>= 3.0.0), vdiffr (>= 1.0.0)

**Depends** R (>= 3.6.0)

**BugReports** https://github.com/InseeFr/disaggR/issues

**LazyData** yes

**Collate** 'bflSmooth.R' 'data.R' 'disaggR.R' 'utils.R' 'in.R' 's4register.R' 'twoStepsBenchmark.R' 'methods.R' 'plot.R' 'praislm.R' 'reView.R' 'threeRuleSmooth.R'

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**URL** https://inseefr.github.io/disaggR/

**NeedsCompilation** no

**Author** Arnaud Feldmann [aut] (ORCID: <https://orcid.org/0000-0003-0109-7505>,
Author and maintener of the package until the version 1.0.1),
Pauline Meinzel [cre],
Thomas Laurent [ctb] (Maintener of the package from 1.0.2 to 1.0.5.2),
Franck Arnaud [ctb] (barplot base graphics method for the mts class),
Institut national de la statistique et des études économiques [cph]
(https://www.insee.fr/)

**Maintainer** Pauline Meinzel <pauline.meinzel@insee.fr>

**Repository** CRAN

**Date/Publication** 2025-11-04 00:00:02 UTC

## Contents

---

`bflSmooth` *Smooth a time series*

---

### Description

bflSmooth smoothes a time series into a time series of a higher frequency that exactly aggregates 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 series to be smoothed |
| `nfrequency` | the new high frequency. It must be a multiple of the low frequency. |
| `weights` | NULL or a time series of the same size than the expected high-frequency serie. |
| `lfserie.is.rate` | TRUE or FALSE. Only taken into account 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 the input lfserie.
- if `TRUE` the input lfserie is the rate to be smoothed, with the constraint that the low-frequency weighted means of the output are equal to lfserie.

## Value

A time series of frequency nfrequency

---

`distance` *Distance computation for disaggregations*

---

## Description

This function `distance` computes the Minkowski distance of exponent p, related to a tscomparison object, produced with `in_sample`, `in_disaggr` or `in_revisions`

## Usage

```
distance(x, p = 2)
```

## Arguments

| | |
|---|---|
| `x` | an object of class `tscomparison` |
| `p` | an integer greater than 1L, or Inf. |

## Details

The meaning depends on the tscomparison function :

- `in_sample` will produce the low-frequency distance between the predicted value and the response, on the coefficient calculation window.
- `in_disaggr` will produce the high-frequency distance between the inputs (eventually, the sum of its contributions) and the benchmarked series.
- `in_revisions` will produce the high-frequency distance between the two benchmarked series (contributions distance isn't permitted).

### Value

a numeric of length 1, the distance.

### See Also

in_sample in_disaggr in_revisions

### Examples

```
benchmark <- twoStepsBenchmark(turnover,construction,include.rho = TRUE)
distance(in_sample(benchmark,type="changes"))
distance(in_disaggr(benchmark,type="contributions"),p=1L)
distance(in_disaggr(benchmark,type="changes"),p=Inf)
```

---

## `in_disaggr` *Comparing a disaggregation with the high-frequency input*

---

### Description

The function `in_disaggr` takes a twoStepsBenchmark or a threeRuleSmooth object as an input. It produces a comparison between the benchmarked time series and the high-frequency input.

### Usage

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

### Arguments

| | |
|---|---|
| `object` | an object of class `"twoStepsBenchmark"` or `"threeRuleSmooth"`. |
| `type` | `"levels"`,`"levels-rebased"`, `"changes"` or `"contributions"`. This defines the type of output. |

### Details

The functions `plot` and `autoplot` can be used on this object to produce graphics.

### Value

a named matrix time series of two columns, one for the response and the other for the input. A `tscomparison` class is added to the object.

### See Also

in_sample in_revisions in_scatter plot.tscomparison

### Examples

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

`in_revisions` *Comparing two disaggregations together*

## Description

The function `in_revisions`takes two inputs, twoStepsBenchmark or a threeRuleSmooth, and produces a comparison between those.

## Usage

```
in_revisions(object, object_old, type = "changes")
```

## Arguments

| | |
|---|---|
| `object` | an object of class `"twoStepsBenchmark"` or `"threeRuleSmooth"`. |
| `object_old` | an object of class `"twoStepsBenchmark"` or `"threeRuleSmooth"`. |
| `type` | `"levels"`,`"levels-rebased"`, `"changes"` or `"contributions"`. This defines the type of output. |

## Details

The functions `plot` and `autoplot` can be used on this object to produce graphics.

## Value

a named matrix time series of two columns, one for the response and the other for the predicted value. A `tscomparison` class is added to the object.

## See Also

in_sample in_disaggr in_scatter plot.tscomparison

## Examples

```
benchmark <- twoStepsBenchmark(turnover,construction,include.rho = TRUE)
benchmark2 <- twoStepsBenchmark(turnover,construction,include.differenciation = TRUE)
plot(in_revisions(benchmark,benchmark2))
```

`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 functions `plot` and `autoplot` can be used on this object to produce graphics.

The predicted values are different from the fitted values :

- they are eventually reintegrated.
- they contain the autocorrelated part of the residuals.

Besides, changes are relative to the latest benchmark value, not the latest predicted value.

## Value

a named matrix time series of two columns, one for the response and the other for the predicted value. A `"tscomparison"` class is added to the object.

## See Also

in_disaggr in_revisions in_scatter plot.tscomparison

## Examples

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

`in_scatter` *Comparing the inputs of a praislm regression*

## Description

The function `in_scatter` returns low-frequency comparisons of the inputs from a praislm, a twoStepsBenchmark or threeRuleSmooth.

## Usage

```
in_scatter(
  object,
  type = if (model.list(object)$include.differenciation) "changes" else "levels"
)
```

## Arguments

| | |
|---|---|
| `object` | an object of class `"praislm"`, `"twoStepsBenchmark"` or `"threeRuleSmooth"`. |
| `type` | `"levels"` or `"changes"`. This defines the type of output. A differencied model can't have a scatterplot in levels. |

## Details

The functions `plot` and `autoplot` can be used on this object to produce graphics.

## Value

a named matrix time series of two or three columns, one for the low-frequency serie and the others for the high-frequency series (eventually differentiated if `include.differenciation` is `TRUE`). A `tscomparison` class is added to the object. For a `twoStepsBenchmark` object, this matrix has three columns, for the low-frequency series, the high-frequency on the regression span and the high-frequency series on the benchmark span.

If outlier effects are estimated, the contributions of the outliers are substracted from the low-frequency series.

## See Also

in_sample in_disaggr in_revisions plot.tscomparison

## Examples

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

## plot.twoStepsBenchmark *Plotting disaggR objects*

### Description

Plot methods for objects of class "tscomparison", threeRuleSmooth and twoStepsBenchmark. :

- plot draws a plot with base graphics
- autoplot produces a ggplot object

Objects of class tscomparison can be produced with the functions in_sample, in_scatter, in_revisions, in_disaggr.

### Usage

```
## S3 method for class 'twoStepsBenchmark'
plot(
  x,
  xlab = NULL,
  ylab = NULL,
  start = NULL,
  end = NULL,
  col = default_col_pal(x),
  lty = default_lty_pal(x),
  show.legend = TRUE,
  main = NULL,
  mar = default_margins(main, xlab, ylab),
  ...
)

## S3 method for class 'threeRuleSmooth'
plot(
  x,
  xlab = NULL,
  ylab = NULL,
  start = NULL,
  end = NULL,
  col = default_col_pal(x),
  lty = default_lty_pal(x),
  show.legend = TRUE,
  main = NULL,
  mar = default_margins(main, xlab, ylab),
  ...
)

## S3 method for class 'tscomparison'
```

```
plot(
  x,
  xlab = NULL,
  ylab = NULL,
  start = NULL,
  end = NULL,
  col = default_col_pal(x),
  lty = default_lty_pal(x),
  show.legend = TRUE,
  main = NULL,
  mar = default_margins(main, xlab, ylab),
  ...
)

## S3 method for class 'twoStepsBenchmark'
autoplot(
  object,
  xlab = NULL,
  ylab = NULL,
  start = NULL,
  end = NULL,
  col = default_col_pal(object),
  lty = default_lty_pal(object),
  show.legend = TRUE,
  main = NULL,
  mar = NULL,
  theme = default_theme_ggplot(object, start, end, show.legend, xlab, ylab, mar),
  ...
)

## S3 method for class 'threeRuleSmooth'
autoplot(
  object,
  xlab = NULL,
  ylab = NULL,
  start = NULL,
  end = NULL,
  col = default_col_pal(object),
  lty = default_lty_pal(object),
  show.legend = TRUE,
  main = NULL,
  mar = NULL,
  theme = default_theme_ggplot(object, start, end, show.legend, xlab, ylab, mar),
  ...
)

## S3 method for class 'tscomparison'
autoplot(
  object,
  xlab = NULL,
  ylab = NULL,
  start = NULL,
  end = NULL,
  col = default_col_pal(object),
  lty = default_lty_pal(object),
  show.legend = TRUE,
  main = NULL,
  mar = NULL,
 theme = default_theme_ggplot(object, start, end, show.legend, xlab, ylab, mar),
  ...
)
```

## Arguments

| | |
|---|---|
| `x` | (for the plot method) a tscomparison, a twoStepsBenchmark or a threeRuleSmooth. |
| `xlab` | the title for the x axis |
| `ylab` | the title for the y axis |
| `start` | a numeric of length 1 or 2. The start of the plot. |
| `end` | a numeric of length 1 or 2. The end of the plot. |
| `col` | the color scale applied on the plot. Could be a vector of colors, or a function from n to a color vector of size n. |
| `lty` | the linetype scales applied on the plot. Could be a vector of linetypes, or a function from n to a linetypes vector of size n. |
| `show.legend` | `TRUE` or `FALSE`. Should an automatic legend be added to the plot. |
| `main` | a character of length 1, the title of the plot |
| `mar` | a numeric of length 4, the margins of the plot specified in the form `c(bottom, left, top, right)`. |
| `...` | other arguments passed either to ggplot or plot |
| `object` | (for the autoplot method) a tscomparison, a twoStepsBenchmark or a threeRuleSmooth. |
| `theme` | a ggplot theme object to replace the default one (only for autoplot methods) |

## Value

`NULL` for the plot methods, the ggplot object for the autoplot methods

## Examples

```
benchmark <- twoStepsBenchmark(turnover,construction,include.rho = TRUE)
plot(benchmark)
plot(in_sample(benchmark))
if(require("ggplot2")) {
  autoplot(in_disaggr(benchmark,type="changes"),
           start=c(2015,1),
          end=c(2020,12))
}
plot(in_scatter(benchmark),xlab="title x",ylab="title y")
```

---

`rePort` *Producing a report*

---

## Description

This function takes an output of the reView **shiny** application and produces an html report with the same outputs than in shiny.

## Usage

```
rePort(
  object,
  output_file = NULL,
  launch.browser = if (is.null(output_file)) TRUE else FALSE,
  hfserie_name = NULL,
  lfserie_name = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `object` | a twoStepsBenchmark with an univariate hfserie, a reViewOutput, or a character of length 1 with the path of their RDS file. If a reViewOutput is chosen, the former new benchmark is taken as the old one. |
| `output_file` | The file in which the html should be saved. If `NULL` the file is temporary, and opened in a tab of the default browser. |
| `launch.browser` | `TRUE` or `FALSE`. If TRUE, the output is opened in the browser. Defaults to TRUE if output_file is NULL. |
| `hfserie_name` | a language object or a character of length 1. The name of the hfserie, eventually its expression. |
| `lfserie_name` | a language object or a character of length 1. The name of the lfserie, eventually its expression. |
| `...` | other arguments passed to rmarkdown::render |

## Details

It can also directly take a twoStepsBenchmark as an input.

## See Also

reView

## `reUseBenchmark` *Using an estimated benchmark model on another time series*

### Description

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

### Usage

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

### Arguments

| | |
|---|---|
| `hfserie` | the bended time series. If it is a matrix time series, 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 series is equal to the low-frequency series. |

### Details

`reUseBenchmark` is primarily meant to be used on a series 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 series isn't equal to the low-frequency series.

### Value

`reUseBenchmark` 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)
```

## reView *A shiny app to reView and modify twoStepsBenchmarks*

### Description

reView allows the user to easily access diverse outputs in order to review a benchmark object, made with twoStepsBenchmark.

The `hfserie_name` and `lfserie_name` define :

### Usage

```
reView(object, hfserie_name = NULL, lfserie_name = NULL, compare = TRUE)
```

### Arguments

| | |
|---|---|
| `object` | a twoStepsBenchmark with an univariate hfserie, a reViewOutput, or a character of length 1 with the path of their RDS file. If a reViewOutput is chosen, the former new benchmark is taken as the old one. |
| `hfserie_name` | a language object or a character of length 1. The name of the hfserie, eventually its expression. |
| `lfserie_name` | a language object or a character of length 1. The name of the lfserie, eventually its expression. |
| `compare` | a boolean of length 1, that tells if the outputs of the old benchmark should be displayed. |

### Details

- the default file name of the RDS file
- the names of the series in the output `call` element

By default, these are set as defined in their `call` element.

The app is made of **shiny** modules in order to make it easy to integrate it into a wider application. In the module part, every input are defined as reactive variables.

### Value

a list, of class reViewOutput, containing the new benchmark, the old one, the names of the series and the boolean compare. This object can also be saved in RDS format through the app. The reViewOutput object can be displayed as a html report with the same informations than in shiny, with the rePort method.

### See Also

rePort

## Examples

```
## Not run:
reView(twoStepsBenchmark(turnover,construction))

## End(Not run)
```

---

## threeRuleSmooth *Bends a time series with a lower frequency one by smoothing their rate*

---

### Description

threeRuleSmooth bends a time series with a time series of a lower frequency. The procedure involved is a proportional Denton benchmark.

Therefore, the resulting time series is the product of the high frequency input with a smoothed rate. This latter is extrapolated through an arithmetic sequence.

The resulting time series is equal to the low-frequency series after aggregation within the benchmark window.

### Usage

```
threeRuleSmooth(
  hfserie,
  lfserie,
  start.benchmark = NULL,
  end.benchmark = NULL,
  start.domain = NULL,
  end.domain = NULL,
  start.delta.rate = NULL,
  end.delta.rate = NULL,
  set.delta.rate = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| `hfserie` | the bended time series. It can be a matrix time series. |
| `lfserie` | a time series whose frequency divides the frequency of `hfserie`. |
| `start.benchmark` | an optional start for `lfserie` to bend `hfserie`. Should be a numeric of length 1 or 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 numeric of length 1 or 2, like a window for `lfserie`. If NULL, the start is defined by lfserie's window. |

| | |
|---|---|
| `start.domain` | an optional start of the output high-frequency series. 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 numeric of length 1 or 2, like a window for `hfserie`. If NULL, the start is defined by hfserie's window. |
| `end.domain` | an optional end of the output high-frequency series. 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. |
| `start.delta.rate` | an optional start for the mean of the rate difference. It is required as a common difference for the arithmetical extrapolation of the rate. Should be a numeric of length 1 or 2, like a window for `lfserie`. If NULL, the start is defined by lfserie's window. |
| `end.delta.rate` | an optional end for the mean of the rate difference. It is required as a common difference for the arithmetical extrapolation of the rate. Should be a numeric of length 1 or 2, like a window for `lfserie`. If NULL, the end is defined by lfserie's window. |
| `set.delta.rate` | an optional double, that allows the user to set the delta mean instead of using a mean. |
| `...` | if the dots contain a cl item, its value overwrites the value of the returned call. This feature allows to build wrappers. |

## Details

In order to smooth the rate, threeRuleSmooth calls bflSmooth and uses a modified and extrapolated version of hfserie as weights :

- only the full cycles are kept
- the first and last full cycles are replicated respectively backwards and forwards to fill the domain window.

## Value

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

The functions `plot` and `autoplot` (the generic from **ggplot2**) produce graphics of the benchmarked series and the bending series. The functions in_disaggr, in_revisions, in_scatter produce various comparisons on which plot and autoplot can also be used.

The generic accessor functions `as.ts`, `model.list`, `smoothed.rate` extract various useful features of the returned value.

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

| | |
|---|---|
| `benchmarked.serie` | a time series, that is the result of the benchmark. |
| `lfrate` | a time series, that is the low-frequency rate of the threeRuleSmooth. |
| `smoothed.rate` | the smoothed rate of the threeRuleSmooth. |

| | |
|---|---|
| `hfserie.as.weights` | the modified and extrapolated hfserie (see details). |
| `delta.rate` | the low-frequency delta of the rate, used to extrapolate the low-frequenccy rate time series. It is estimated as the mean value in the specified window. |
| `model.list` | a list containing all the arguments submitted to the function. |
| `call` | the matched call. |

## Examples

```
## How to use threeRuleSmooth

smooth <- threeRuleSmooth(hfserie = turnover,
                          lfserie = construction)
as.ts(smooth)
coef(smooth)
summary(smooth)
library(ggplot2)
autoplot(in_disaggr(smooth))
```

---

`twoStepsBenchmark` *Regress and bends a time series with a lower frequency one*

---

## Description

twoStepsBenchmark bends a time series with a time series of a lower frequency. The procedure involved is a Prais-Winsten regression, then an additive Denton benchmark.

Therefore, the resulting time series is the sum of a regression fit and of a smoothed part. The smoothed part minimizes the sum of squares of its differences.

The resulting time series is equal to the low-frequency series after aggregation within the benchmark window.

## 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,outliers=NULL,
                  ...)


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

## Arguments

| | |
|---|---|
| `hfserie` | the bended time series. It can be a matrix time series. |
| `lfserie` | a time series whose frequency divides the frequency of `hfserie`. |
| `include.differenciation` | a boolean of length 1. If `TRUE`, `lfserie` and `hfserie` are differentiated 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 numeric, that allows the user to set the regression coefficients instead of evaluating them. If hfserie is not a matrix, set.coeff can be an unnamed numeric of length 1. Otherwise, `set.coeff` has to be a named numeric, which will set the corresponding coefficients instead of evaluating them. Each column name of hfserie and each outlier set with the `outlier` arg initialize a coefficient with the same name, that can be set through set.coeff. The default name for a non-matrix time series is then `"hfserie"`, By example, a LS2003 and the time series can be set using `set.coeff=c(hfserie=3,LS2003=1)`. |
| `set.const` | an optional numeric 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 numeric of length 1 or 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 numeric of length 1 or 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 numeric of length 1 or 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 numeric of length 1 or 2, like a window for `lfserie`. If NULL, the start is defined by lfserie's window. |
| `start.domain` | an optional for the output high-frequency series. 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 numeric of length 1 or 2, like a window for `hfserie`. If NULL, the start is defined by hfserie's window. |

`end.domain` an optional end for the output high-frequency series. 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 numeric of length 1 or 2, like a window for `hfserie`. If NULL, the start is defined by hfserie's window.

`outliers` an optional named list of numeric vectors, whose pattern is like `list(AO2008T2=c(0,0,3,2),LS2002=c(` where :

- `"AO"` stands for additive outlier or `"LS"` for level shift
- The integer that follows stands for the outlier starting year
- an optional integer, preceded by the letter T, stands for the low-frequency cycle of the outlier start.
- The numeric vector values stands for the disaggregated value of the outlier and its length must be a multiple of hf / lf

The outliers coefficients are evaluated though the regression process, like any coefficient. Therefore, if any outlier is outside of the coefficient calculation window, it should be fixed using `set.coeff`.

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

## Details

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.

## 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 generic from **ggplot2**) produce graphics of the benchmarked serie and the bending serie. The functions in_disaggr, in_revisions, in_scatter produce comparisons on which plot and autoplot can also be used.

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 series, that is the result of the benchmark. It is equal to `fitted.values + smoothed.part`.

`fitted.values` a time series, that is the high-frequency series as it is after having applied the regression coefficients. Compared to the fitted values of the regression, which can be retrieved inside the regression component, it has a high-frequency time series and can eventually be integrated if `include.differenciation` is `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. It is the smoothed difference between the `fitted.values` and lfserie. |
| `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 <- twoStepsBenchmark(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 <- twoStepsBenchmark(hfserie = turnover,
                                lfserie = construction,
                                include.differenciation = TRUE,
                                set.coeff = 0.1)
coef(benchmark2)
```

# Index