Package ‘segmented’

May 14, 2024

Type Package

Title Regression Models with Break-Points / Change-Points Estimation (with Possibly Random Effects)

Version 2.1-0

Date 2024-05-14

Maintainer Vito M. R. Muggeo <vito.muggeo@unipa.it>

Description Fitting regression models where, in addition to possible linear terms, one or more covariates have segmented (i.e., broken-line
or piece-wise linear) or stepmented (i.e. piece-wise constant) effects.
Multiple breakpoints for the same variable are allowed. The estimation method is discussed in Muggeo (2003, <doi:10.1002/sim.1545>) and
illustrated in Muggeo (2008, <https://www.r-project.org/doc/Rnews/Rnews_2008-1.pdf>). An approach for hypothesis testing is presented
in Muggeo (2016, <doi:10.1080/00949655.2016.1149855>), and interval estimation for the breakpoint is discussed in Muggeo (2017, <doi:10.1111/anzs.12200>).
Segmented mixed models, i.e. random effects in the change point, are discussed in Muggeo (2014, <doi:10.1177/1471082X13504721>).
Estimation of piecewise-constant relationships and changepoints (mean-shift models) is discussed in Fasola et al. (2018, <doi:10.1007/s00180-017-0740-4>).

Depends R (>= 3.5.0), MASS, nlme

License GPL

NeedsCompilation no

Author Vito M. R. Muggeo [aut, cre] (<https://orcid.org/0000-0002-3386-4054>)

RoxygenNote 7.3.1

Repository CRAN

Date/Publication 2024-05-14 11:10:02 UTC

R topics documented:

segmented-package *Segmented Relationships in Regression Models with Breakpoints / Changepoints Estimation (with Possibly Random Effects)*

Description

Estimation and inference of regression models with piecewise linear relationships, also known as segmented regression models, with a number of break-points fixed or to be 'selected'. Random effects changepoints are also allowed since version 1.6-0, and since version 2.0-0 it is also possible to fit regression models with piecewise constant (or 'stepmented') relationships.

Details

Package:	segmented
Type:	Package
Version:	2.1-0
Date:	2024-05-14
License:	GPL

Package `segmented` aims to estimate linear and generalized linear models (and virtually any regression model) having one or more segmented or stepmented relationships in the linear predictor. Estimates of the slopes and breakpoints are provided along with standard errors. The package includes testing/estimating functions and methods to print, summarize and plot the results.

The algorithms used by `segmented` are *not* grid-search. They are iterative procedures (Muggeo, 2003; Fasola et al., 2018) that need starting values *only* for the breakpoint parameters and therefore they are quite efficient even with several breakpoints to be estimated. Moreover since version 0.2-9.0, `segmented` implements the bootstrap restarting (Wood, 2001) to make the algorithms less sensitive to the starting values (which can be also omitted by the user) .

Since version 0.5-0.0 a default method `segmented.default` has been added. It may be employed to include segmented relationships in *general* regression models where specific methods do not exist. Examples include quantile, Cox, and lme regressions where the random effects do not refer to the breakpoints; see `segmented.lme` to include random changepoints. `segmented.default` includes some examples.

Since version 1.0-0 the estimating algorithm has been slight modified and it appears to be much stabler (in examples with noisy segmented relationhips and flat log likelihoods) then previous versions.

Hypothesis testing (about the existence of the breakpoint) and confidence intervals are performed via appropriate methods and functions.

A tentative approach to deal with unknown number of breakpoints is also provided, see option `fix.npsi` in `seg.control`. Also, as version 1.3-0, the `selgmented` function has been introduced to select the number of breakpoints via the BIC or sequential hypothesis testing.

Since version 1.6-0, estimation of segmented mixed models has been introduced, see `segmented.lme` and related function. Since version 2.0-0, it is possible to fit segmented relationships with constraints on the slopes, see `segreg`.

Finally, since 2.0-0, it is possible to fit (G)LM wherein one or more covariates have a stepmented (i.e. a step-function like) relationship, see `stepmented`.

Author(s)

Vito M.R. Muggeo <vito.muggeo@unipa.it>

References

Muggeo V.M.R., Atkins D.C., Gallop R.J., Dimidjian S. (2014) Segmented mixed models with random changepoints: a maximum likelihood approach with application to treatment for depression study. *Statistical Modelling*, **14**, 293-313.

Muggeo, V.M.R. (2017) Interval estimation for the breakpoint in segmented regression: a smoothed score-based approach. *Australian & New Zealand Journal of Statistics*, **59**, 311–322.

Fasola S, Muggeo V.M.R., Kuchenhoff, H. (2018) A heuristic, iterative algorithm for change-point detection in abrupt change models, *Computational Statistics*, **2**, 997–1015.

Muggeo, V.M.R. (2016) Testing with a nuisance parameter present only under the alternative: a score-based approach with application to segmented modelling. *J of Statistical Computation and Simulation* **86**, 3059–3067.

Davies, R.B. (1987) Hypothesis testing when a nuisance parameter is present only under the alternative. *Biometrika* **74**, 33–43.

Seber, G.A.F. and Wild, C.J. (1989) *Nonlinear Regression*. Wiley, New York.

Bacon D.W., Watts D.G. (1971) Estimating the transistion between two intersecting straight lines. *Biometrika* **58**: 525 – 534.

Muggeo, V.M.R. (2003) Estimating regression models with unknown break-points. *Statistics in Medicine* **22**, 3055–3071.

Muggeo, V.M.R. (2008) Segmented: an R package to fit regression models with broken-line relationships. *R News* **8/1**, 20–25.

Muggeo, V.M.R., Adelfio, G. (2011) Efficient change point detection in genomic sequences of continuous measurements. *Bioinformatics* **27**, 161–166.

Wood, S. N. (2001) Minimizing model fitting objectives that contain spurious local minima by bootstrap restarting. *Biometrics* **57**, 240–244.

Muggeo, V.M.R. (2010) Comment on ‘Estimating average annual per cent change in trend analysis’ by Clegg et al., Statistics in Medicine; 28, 3670-3682. *Statistics in Medicine*, **29**, 1958–1960.

aapc	*Average annual per cent change in segmented trend analysis*

Description

Computes the average annual per cent change to summarize piecewise linear relationships in segmented regression models.

Usage

```
aapc(ogg, parm, exp.it = FALSE, conf.level = 0.95, wrong.se = TRUE,
    .vcov=NULL, .coef=NULL, ...)
```

Arguments

`ogg`	the fitted model returned by `segmented`.
`parm`	the *single* segmented variable of interest. It can be missing if the model includes a single segmented covariate. If missing and `ogg` includes several segmented variables, the first one is considered.
`exp.it`	logical. If `TRUE`, the per cent change is computed, namely $\exp(\hat{\mu}) - 1$ where $\mu = \sum_j \beta_j w_j$, see 'Details'.
`conf.level`	the confidence level desidered.
`wrong.se`	logical, if `TRUE`, the 'wrong" standard error (as discussed in Clegg et al. (2009)) ignoring uncertainty in the breakpoint estimate is returned as an attribute `"wrong.se"`.
`.vcov`	The *full* covariance matrix of estimates. If unspecified (i.e. `NULL`), the covariance matrix is computed internally by `vcov(ogg,...)`.
`.coef`	The regression parameter estimates. If unspecified (i.e. `NULL`), it is computed internally by `coef(ogg)`.
`...`	further arguments to be passed on to `vcov.segmented()`, such as `var.diff` or `is`.

Details

To summarize the fitted piecewise linear relationship, Clegg et al. (2009) proposed the 'average annual per cent change' (AAPC) computed as the sum of the slopes (β_j) weighted by corresponding covariate sub-interval width (w_j), namely $\mu = \sum_j \beta_j w_j$. Since the weights are the breakpoint differences, the standard error of the AAPC should account for uncertainty in the breakpoint estimate, as discussed in Muggeo (2010) and implemented by `aapc()`.

Value

`aapc` returns a numeric vector including point estimate, standard error and confidence interval for the AAPC relevant to variable specified in `parm`.

Note

`exp.it=TRUE` would be appropriate only if the response variable is the log of (any) counts.

Author(s)

Vito M. R. Muggeo, `<vito.muggeo@unipa.it>`

References

Clegg LX, Hankey BF, Tiwari R, Feuer EJ, Edwards BK (2009) Estimating average annual per cent change in trend analysis. *Statistics in Medicine*, **28**; 3670-3682.

Muggeo, V.M.R. (2010) Comment on 'Estimating average annual per cent change in trend analysis' by Clegg et al., Statistics in Medicine; 28, 3670-3682. *Statistics in Medicine*, **29**, 1958–1960.

Examples

```
set.seed(12)
x<-1:20
y<-2-.5*x+.7*pmax(x-9,0)-.8*pmax(x-15,0)+rnorm(20)*.3
o<-lm(y~x)
os<-segmented(o, psi=c(5,12))
aapc(os)
```

`broken.line` *Fitted values for segmented relationships*

Description

Given a segmented model (typically returned by a `segmented` method), `broken.line` computes the fitted values (and relevant standard errors) for the specified 'segmented' relationship.

Usage

```
broken.line(ogg, term = NULL, link = TRUE, interc=TRUE, se.fit=TRUE, isV=FALSE,
  .vcov=NULL, .coef=NULL, ...)
```

Arguments

`ogg`	A fitted object of class segmented (returned by any `segmented` method).
`term`	Three options. i) A named list (whose name should be one of the segmented covariates in the model ogg) including the covariate values for which segmented predictions should be computed; ii) a character meaning the name of any segmented covariate in the model (and predictions corresponding to the observed covariate values are returned); iii) It can be `NULL` if the model includes a single segmented covariate (and predictions corresponding to the observed covariate values are returned).

`link`	Should the predictions be computed on the scale of the link function if ogg is a segmented glm fit? Default to `TRUE`.
`interc`	Should the model intercept be added? (provided it exists).
`se.fit`	If `TRUE` also standard errors for predictions are returned.
`isV`	A couple of logicals indicating if the segmented terms $(x-\psi)_+$ and $I(x>\psi)$ in the model matrix should be replaced by their smoothed counterparts when computing the standard errors. If a single logical is provided, it is applied to both terms.
`.vcov`	Optional. The *full* covariance matrix of estimates. If `NULL` (and `se.fit=TRUE`), the matrix is computed internally via `vcov.segmented()`.
`.coef`	The regression parameter estimates. If unspecified (i.e. `NULL`), it is computed internally by `coef(ogg)`.
`...`	Additional arguments to be passed on to `vcov.segmented()` when computing the standard errors for the predictions, namely `is`, `var.diff`, `p.df`. See `summary.segmented` and `vcov.segmented`.

Details

If `term=NULL` or `term` is a valid segmented covariate name, predictions for that segmented variable are the relevant fitted values from the model. If `term` is a (correctly named) list with numerical values, predictions corresponding to such specified values are computed. If `link=FALSE` and `ogg` inherits from the class "glm", predictions and possible standard errors are returned on the response scale. The standard errors come from the Delta method. Argument `link` is ignored whether `ogg` does not inherit from the class "glm".

Value

A list having one component if (if `se.fit=FALSE`), and two components (if `se.fit=TRUE`) list representing predictions and standard errors for the segmented covariate values.

Note

This function was written when there was not `predict.segmented` (which is more general).

Author(s)

Vito M. R. Muggeo

See Also

`segmented`, `predict.segmented`, `plot.segmented`, `vcov.segmented`

Examples

```
set.seed(1234)
z<-runif(100)
y<-rpois(100,exp(2+1.8*pmax(z-.6,0)))
o<-glm(y~z,family=poisson)
```

```
o.seg<-segmented(o,seg.Z=~z)
## Not run: plot(z,y)
## Not run: points(z,broken.line(o.seg,link=FALSE)$fit,col=2) #ok, but use plot.segmented()!
```

---

confint.segmented *Confidence intervals for breakpoints*

Description

Computes confidence intervals for the breakpoints in a fitted 'segmented' model.

Usage

```
## S3 method for class 'segmented'
confint(object, parm, level=0.95, method=c("delta", "score", "gradient"),
   rev.sgn=FALSE, var.diff=FALSE, is=FALSE, digits=max(4, getOption("digits") - 1),
     .coef=NULL, .vcov=NULL, ...)
```

Arguments

object	a fitted segmented object.
parm	the segmented variable of interest. If missing the first segmented variable in object is considered.
level	the confidence level required, default to 0.95.
method	which confidence interval should be computed. One of "delta", "score", or "gradient". Can be abbreviated.
rev.sgn	vector of logicals. The length should be equal to the length of parm; recycled otherwise. when TRUE it is assumed that the current parm is 'minus' the actual segmented variable, therefore the sign is reversed before printing. This is useful when a null-constraint has been set on the last slope.
var.diff	logical. If method="delta", and there is a single segmented variable, var.diff=TRUE leads to standard errors based on sandwich-type formula of the covariance matrix. See Details in summary.segmented.
is	logical. If method="delta", is=TRUE means that the full covariance matrix is computed via vcov(.., is=TRUE)
digits	controls the number of digits to print when returning the output.
.coef	The regression parameter estimates. If unspecified (i.e. NULL), it is computed internally by coef(object).
.vcov	The *full* covariance matrix of estimates. If unspecified (i.e. NULL), the covariance matrix is computed internally by vcov(object).
...	additional parameters referring to Score-based confidence intervals, such as "h", "d.h", "bw", "msgWarn", and "n.values" specifying the number of points used to profile the Score (or Gradient) statistic.

Details

`confint.segmented` computes confidence limits for the breakpoints. Currently there are three options, see argument `method`. `method="delta"` uses the standard error coming from the Delta method for the ratio of two random variables. This value is an approximation (slightly) better than the one reported in the 'psi' component of the list returned by any `segmented` method. The resulting confidence intervals are based on the asymptotic Normal distribution of the breakpoint estimator which is reliable just for clear-cut kink relationships. See Details in `segmented`.
`method="score"` or `method="gradient"` compute the confidence interval via profiling the Score or the Gradient statistics smoothed out by the induced smoothing paradigm, as discussed in the reference below.

Value

A matrix including point estimate and confidence limits of the breakpoint(s) for the segmented variable possibly specified in `parm`.

Note

Currently `method="score"` or `method="gradient"` only works for segmented *linear* model. For segmented *generalized linear* model, currently only `method="delta"` is available.

Author(s)

Vito M.R. Muggeo

References

Muggeo, V.M.R. (2017) Interval estimation for the breakpoint in segmented regression: a smoothed score-based approach. *Australian & New Zealand Journal of Statistics* **59**, 311–322.

See Also

`segmented` and `lines.segmented` to plot the estimated breakpoints with corresponding confidence intervals.

Examples

```
set.seed(10)
x<-1:100
z<-runif(100)
y<-2+1.5*pmax(x-35,0)-1.5*pmax(x-70,0)+10*pmax(z-.5,0)+rnorm(100,0,2)
out.lm<-lm(y~x)
o<-segmented(out.lm,seg.Z=~x+z,psi=list(x=c(30,60),z=.4))
confint(o) #delta CI for the 1st variable
confint(o, "x", method="score") #also method="g"
```

`confint.segmented.lme` *Confidence intervals in segmented mixed models*

Description

Computes confidence intervals for all regression parameters, including the the breakpoint, in a fitted 'segmented mixed' model.

Usage

```
## S3 method for class 'segmented.lme'
confint(object, parm, level = 0.95, obj.boot, ...)
```

Arguments

`object`	A fit object returned by `segmented.lme`.
`parm`	A vector of numbers indicating which parameters should be considered. If missing all parameters.
`level`	The confidence level.
`obj.boot`	The possible list including the bootstrap distributions of the regression coefficients. Such list is returned by `vcov.segmented.lme(.., ret.b=TRUE)`
`...`	if `obj.boot` is missing and bootstrap CIs are requested, additional optional arguments, such as `B`, `seed`, and `it.max.b`, to be used in computations of the boot distributions.

Details

If `obj.boot` is provided or `...` includes the argument `B>0`, confidence intervals are computed by exploiting the bootstrap distributions.

Value

A matrix (or a list of matrices if bootstrap ci are requested) including the confidence intervals for the model parameters.

Warning

All the functions for segmented mixed models (*.segmented.lme) are still at an experimental stage

Author(s)

Vito Muggeo

See Also

`vcov.segmented.lme`

Examples

```
## Not run:
confint(os) #asymptotic CI

confint(os, B=50) #boot CIs

#it is possible to obtain the boot distribution beforehand
ob <-vcov(os, B=50, ret.b=TRUE)
confint(os, obj.boot=ob) #boot CI


## End(Not run)
```

`confint.stepmented` *Confidence intervals for jumpoints in stepmented regression*

Description

Computes confidence intervals for the changepoints (or jumpoints) in a fitted 'stepmented' model.

Usage

```
## S3 method for class 'stepmented'
confint(object, parm, level=0.95, method=c("delta", "score", "gradient"),
    round=TRUE, cheb=FALSE, digits=max(4, getOption("digits") - 1),
    .coef=NULL, .vcov=NULL, ...)
```

Arguments

`object`	a fitted `stepmented` object.
`parm`	the stepmented variable of interest. If missing the first stepmented variable in `object` is considered.
`level`	the confidence level required, default to 0.95.
`method`	which confidence interval should be computed. One of `"delta"`, `"score"`, or `"gradient"`. Can be abbreviated. Currently only `"delta"` allowed.
`round`	logical. Should the values (estimates and lower/upper limits) rounded to the smallest observed value?
`cheb`	logical. If `TRUE`, the confidence limits are computed using the Chebyshev inequality which yields conservative confidence intervals but it is 'robust' to the non-normality of the changepoint sampling distribution.
`digits`	controls the number of digits to print when returning the output.
`.coef`	The regression parameter estimates. If unspecified (i.e. `NULL`), it is computed internally by `coef(object)`.
`.vcov`	The *full* covariance matrix of estimates. If unspecified (i.e. `NULL`), the covariance matrix is computed internally by `vcov(object)`.
`...`	additional arguments passed to `vcov.stepmented`, namely `k`.

Details

`confint.stepmented` computes confidence limits for the changepoints. Currently the only option is `'delta'`, i.e. to compute the approximate covariance matrix via a smoothing approximation (see `vcov.stepmented`) and to build the limits using the standard Normal quantiles. Note that, the limits are rounded to the lowest observed value, thus the resulting confidence interval might not be symmetric if the stepmented covariate has not equispaced values.

Value

A matrix including point estimate and confidence limits of the breakpoint(s) for the stepmented variable possibly specified in `parm`.

Note

Currently only method='delta' is allowed.

Author(s)

Vito M.R. Muggeo

See Also

`stepmented` and `lines.segmented` to plot the estimated breakpoints with corresponding confidence intervals.

Examples

```
set.seed(10)
x<-1:100
z<-runif(100)
y<-2+2.5*(x>45)-1.5*(x>70)+z+rnorm(100)
o<-stepmented(y, npsi=2)

confint(o) #round=TRUE is default
confint(o, round=FALSE)
```

`davies.test` *Testing for a change in the slope*

Description

Given a generalized linear model, the Davies' test can be employed to test for a non-constant regression parameter in the linear predictor.

Usage

```
davies.test(obj, seg.Z, k = 10, alternative = c("two.sided", "less", "greater"),
    type=c("lrt","wald"), values=NULL, dispersion=NULL)
```

Arguments

obj
: a fitted model typically returned by `glm` or `lm`. Even an object returned by `segmented` can be set (e.g. if interest lies in testing for an additional breakpoint).

seg.Z
: a formula with no response variable, such as `seg.Z=~x1`, indicating the (continuous) segmented variable being tested. Only a single variable may be tested and an error is printed when `seg.Z` includes two or more terms. `seg.Z` can be omitted if i)`obj` is a segmented fit with a single segmented covariate (and that variable is taken), or ii)if it is a "lm" or "glm" fit with a single covariate (and that variable is taken)

k
: number of points where the test should be evaluated. See Details.

alternative
: a character string specifying the alternative hypothesis (relevant to the slope difference parameter).

type
: the test statistic to be used (only for GLM, default to lrt). Ignored if `obj` is a simple linear model.

values
: optional. The evaluation points where the Davies approximation is computed. See Details for default values.

dispersion
: the dispersion parameter for the family to be used to compute the test statistic. When `NULL` (the default), it is inferred from `obj`. Namely it is taken as `1` for the Binomial and Poisson families, and otherwise estimated by the residual Chi-squared statistic (calculated from cases with non-zero weights) divided by the residual degrees of freedom.

Details

`davies.test` tests for a non-zero difference-in-slope parameter of a segmented relationship. Namely, the null hypothesis is $H_0 : \beta = 0$, where β is the difference-in-slopes, i.e. the coefficient of the segmented function $\beta(x - \psi)_+$. The hypothesis of interest $\beta = 0$ means no breakpoint. Roughly speaking, the procedure computes `k` 'naive' (i.e. assuming fixed and known the breakpoint) test statistics for the difference-in-slope, seeks the 'best' value and corresponding naive p-value (according to the alternative hypothesis), and then corrects the selected (minimum) p-value by means of the `k` values of the test statistic. If `obj` is a LM, the Davies (2002) test is implemented. This approach works even for small samples. If `obj` represents a GLM fit, relevant methods are described in Davies (1987), and the Wald or the Likelihood ratio test statistics can be used, see argument `type`. This is an asymptotic test. The `k` evaluation points are `k` equally spaced values between the second and the second-last values of the variable reported in `seg.Z`. `k` should not be small; I find no important difference for `k` larger than 10, so default is `k=10`.

Value

A list with class '`htest`' containing the following components:

method
: title (character)

data.name
: the regression model and the segmented variable being tested

statistic
: the point within the range of the covariate in `seg.Z` at which the maximum (or the minimum if `alternative="less"`) occurs

`parameter`	number of evaluation points
`p.value`	the adjusted p-value
`process`	a two-column matrix including the evaluation points and corresponding values of the test statistic

Warning

The Davies test is *not* aimed at obtaining the estimate of the breakpoint. The Davies test is based on `k` evaluation points, thus the value returned in the `statistic` component (and printed as "'best' at") is the best among the `k` points, and typically it will differ from the maximum likelihood estimate returned by `segmented`. Use `segmented` if you are interested in the point estimate.

To test for a breakpoint in *linear* models with small samples, it is suggested to use `davies.test()` with objects of class "lm". If `obj` is a `"glm"` object with gaussian family, `davies.test()` will use an approximate test resulting in smaller p-values when the sample is small. However if the sample size is large (n>300), the exact Davies (2002) upper bound cannot be computed (as it relies on `gamma()` function) and the *approximate* upper bound of Davies (1987) is returned.

Note

Strictly speaking, the Davies test is not confined to the segmented regression; the procedure can be applied when a nuisance parameter vanishes under the null hypothesis. The test is slightly conservative, as the computed p-value is actually an upper bound.

Results should change slightly with respect to previous versions where the evaluation points were computed as `k` equally spaced values between the second and the second last observed values of the segmented variable.

Author(s)

Vito M.R. Muggeo

References

Davies, R.B. (1987) Hypothesis testing when a nuisance parameter is present only under the alternative. *Biometrika* **74**, 33–43.

Davies, R.B. (2002) Hypothesis testing when a nuisance parameter is present only under the alternative: linear model case. *Biometrika* **89**, 484–489.

See Also

See also `pscore.test` which is more powerful, especially when the signal-to-noise ratio is low.

Examples

```
## Not run:
set.seed(20)
z<-runif(100)
x<-rnorm(100,2)
y<-2+10*pmax(z-.5,0)+rnorm(100,0,3)
```

```
o<-lm(y~z+x)
davies.test(o,~z)
davies.test(o,~x)

o<-glm(y~z+x)
davies.test(o,~z) #it works but the p-value is too small..

## End(Not run)
```

---

down *Down syndrome in babies*

Description

The down data frame has 30 rows and 3 columns. Variable `cases` means the number of babies with Down syndrome out of total number of births `births` for mothers with mean age `age`.

Usage

```
data(down)
```

Format

A data frame with 30 observations on the following 3 variables.

`age` the mothers' mean age.

`births` count of total births.

`cases` count of babies with Down syndrome.

Source

Davison, A.C. and Hinkley, D. V. (1997) *Bootstrap Methods and their Application*. Cambridge University Press.

References

Geyer, C. J. (1991) Constrained maximum likelihood exemplified by isotonic convex logistic regression. *Journal of the American Statistical Association* **86**, 717–724.

Examples

```
data(down)
```

draw.history — *History for the breakpoint estimates*

Description

Displays breakpoint iteration values for segmented fits.

Usage

```
draw.history(obj, term, ...)
```

Arguments

`obj`	a segmented fit returned by any "segmented" method.
`term`	a character to mean the 'segmented' variable whose breakpoint values throughout iterations have to be displayed.
`...`	graphic parameters to be passed to `matplot()`.

Details

For a given `term` in a segmented fit, `draw.history()` produces two plots. On the left panel it displays the different breakpoint values obtained during the estimating process, since the starting values up to the final ones, while on the right panel the objective values at different iterations. When bootstrap restarting is employed, `draw.history()` produces two plots, the values of objective function and the number of distinct solutions against the bootstrap replicates.

Value

None.

Author(s)

Vito M.R. Muggeo

Examples

```
data(stagnant)
os<-segmented(lm(y~x,data=stagnant),seg.Z=~x,psi=-.8)
# draw.history(os) #diagnostics with boot restarting

os<-segmented(lm(y~x,data=stagnant),seg.Z=~x,psi=-.8, control=seg.control(n.boot=0))
# draw.history(os) #diagnostics without boot restarting
```

`fitted.segmented.lme` *Fitted values for segmented mixed fits*

Description

Computes fitted values at different levels of nesting for segmented mixed objects

Usage

```
## S3 method for class 'segmented.lme'
fitted(object, level = 1, sort=TRUE, ...)
```

Arguments

`object`	Object of class `"segmented.lme"`
`level`	the level to be considered. Currently only levels 0 or 1 are allowed.
`sort`	If `TRUE`, the fitted values are sorted by the names of the 'id' levels.
`...`	Ignored

Details

Currently it works only if `level=1`

Value

A numeric object including the fitted values at the specified level of nesting.

Warning

All the functions for segmented mixed models (*.segmented.lme) are still at an experimental stage

Author(s)

Vito Muggeo

See Also

`summary.segmented.lme`

`globTempAnom` *Global temperature anomalies 1850-2023*

Description

The `globTempAnom` data frame includes the global surface temperature anomalies from 1850 to 2023.

Usage

```
data(globTempAnom)
```

Format

The included variables are (clearly).

`Year` the calendar year.

`Anomaly` the temperature anomalies computed as differences of the annual (average) measurement with respect to the 20th century average (1901-2000).

Details

Data refer to averages measurements referring to land and ocean surface of Northern and Southern hemisphere.

Source

https://www.ncei.noaa.gov/access/monitoring/global-temperature-anomalies/anomalies

References

There are several references using such dataset, e.g.

Cahill, N., Rahmstorf, S., and Parnell, A. C. (2015). Change points of global temperature. *Environmental Research Letters*, 10: 1-6.

Examples

```
data(globTempAnom)
```

intercept *Intercept estimates from segmented relationships*

Description

Computes the intercepts of each ‘segmented’ relationship in the fitted model.

Usage

```
intercept(ogg, parm, rev.sgn = FALSE, var.diff=FALSE,
    .vcov=NULL, .coef=NULL, digits = max(4, getOption("digits") - 2),...)
```

Arguments

`ogg`	an object of class "segmented", returned by any `segmented` method.
`parm`	the segmented variable whose intercepts have to be computed. If missing all the segmented variables in the model are considered.
`rev.sgn`	vector of logicals. The length should be equal to the length of `parm`, but it is recycled otherwise. When `TRUE` it is assumed that the current `parm` is ‘minus’ the actual segmented variable, therefore the order is reversed before printing. This is useful when a null-constraint has been set on the last slope.
`var.diff`	Currently ignored as only point estimates are computed.
`.vcov`	The *full* covariance matrix of estimates. If unspecified (i.e. `NULL`), the covariance matrix is computed internally by `vcov(ogg)`.
`.coef`	The regression parameter estimates. If unspecified (i.e. `NULL`), it is computed internally by `coef(ogg)`.
`digits`	controls number of digits in the returned output.
`...`	Further arguments to be passed on to `vcov.segmented`, such as `var.diff` and `is`. See Details in `vcov.segmented`.

Details

A broken-line relationship means that a regression equation exists in the intervals ‘$min(x)$ to ψ_1’, ‘ψ_1 to ψ_2’, and so on. `intercept` computes point estimates of the intercepts of the different regression equations for each segmented relationship in the fitted model.

Value

`intercept` returns a list of one-column matrices. Each matrix represents a segmented relationship.

Author(s)

Vito M. R. Muggeo, <vito.muggeo@unipa.it>

See Also

See also slope to compute the slopes of the different regression equations for each segmented relationship in the fitted model.

Examples

```
## see ?slope
## Not run:
intercept(out.seg)

## End(Not run)
```

`lines.segmented` *Bars for interval estimate of the breakpoints*

Description

Draws bars relevant to breakpoint estimates (point estimate and confidence limits) on the current device

Usage

```
## S3 method for class 'segmented'
lines(x, term, bottom = TRUE, shift=FALSE, conf.level = 0.95, k = 50,
    pch = 18, rev.sgn = FALSE, .vcov=NULL, .coef=NULL, ...)
```

Arguments

`x`	an object of class `segmented`.
`term`	the segmented variable of the breakpoints being drawn. It may be unspecified when there is a single segmented variable.
`bottom`	logical, indicating if the bars should be plotted at the bottom (`TRUE`) or at the top (`FALSE`).
`shift`	logical, indicating if the bars should be 'shifted' on the y-axis before plotting. Useful for multiple breakpoints with overlapped confidence intervals.
`conf.level`	the confidence level of the confidence intervals for the breakpoints.
`k`	a positive integer regulating the vertical position of the drawn bars. See Details.
`pch`	either an integer specifying a symbol or a single character to be used in plotting the point estimates of the breakpoints. See `points`.
`rev.sgn`	should the signs of the breakpoint estimates be changed before plotting? see Details.
`.vcov`	The *full* covariance matrix of estimates. If unspecified (i.e. `NULL`), the covariance matrix is computed internally by `vcov(x)`.
`.coef`	The regression parameter estimates. If unspecified (i.e. `NULL`), it is computed internally by `coef(x)`.
`...`	further arguments passed to `arrows`, for instance 'col' that can be a vector.

Details

`lines.segmented` simply draws on the current device the point estimates and relevant confidence limits of the estimated breakpoints from a "segmented" object. The y coordinate where the bars are drawn is computed as `usr[3]+h` if `bottom=TRUE` or `usr[4]-h` when `bottom=FALSE`, where `h=(usr[4]-usr[3])/abs(k)` and `usr` are the extremes of the user coordinates of the plotting region. Therefore for larger values of `k` the bars are plotted on the edges. The argument `rev.sgn` allows to change the sign of the breakpoints before plotting. This may be useful when a null-right-slope constraint is set.

See Also

`plot.segmented` to plot the fitted segmented lines, and `points.segmented` to add the fitted joinpoints.

Examples

```
## See ?plot.segmented
```

`lines.stepmented` *Bars for interval estimate of the breakpoints*

Description

Draws bars relevant to breakpoint estimates (point estimate and confidence limits) on the current device

Usage

```
## S3 method for class 'stepmented'
lines(x, term, bottom = TRUE, shift=FALSE, conf.level = 0.95, k = 50,
    pch = 18, .vcov=NULL, .coef=NULL, ...)
```

Arguments

`x`	an object of class `stepmented`.
`term`	the stepmented variable of the breakpoints being drawn. It may be unspecified when there is a single stepmented variable.
`bottom`	logical, indicating if the bars should be plotted at the bottom (`TRUE`) or at the top (`FALSE`).
`shift`	logical, indicating if the bars should be 'shifted' on the y-axis before plotting. Useful for multiple breakpoints with overlapped confidence intervals.
`conf.level`	the confidence level of the confidence intervals for the breakpoints.
`k`	a positive integer regulating the vertical position of the drawn bars. See Details.
`pch`	either an integer specifying a symbol or a single character to be used in plotting the point estimates of the breakpoints. See `points`.

`.vcov`	The *full* covariance matrix of estimates. If unspecified (i.e. `NULL`), the covariance matrix is computed internally by `vcov(x)`.
`.coef`	The regression parameter estimates. If unspecified (i.e. `NULL`), it is computed internally by `coef(x)`.
`...`	further arguments passed to `arrows`, for instance 'col' that can be a vector.

Details

`lines.stepmented` simply draws on the current device the point estimates and relevant confidence limits of the estimated breakpoints from a "stepmented" object. The `y` coordinates where the bars are drawn is computed as `usr[3]+h` if `bottom=TRUE` or `usr[4]-h` when `bottom=FALSE`, where `h=(usr[4]-usr[3])/abs(k)` and `usr` are the extremes of the user coordinates of the plotting region. Therefore for larger values of `k` the bars are plotted on the edges.

See Also

`plot.stepmented` to plot the fitted stepmented lines

Examples

```
## See ?plot.stepmented
```

`model.matrix.segmented` *Design matrix for segmented fits*

Description

This function builds the model matrix for `segmented` fits.

Usage

```
## S3 method for class 'segmented'
model.matrix(object, ...)
```

Arguments

`object`	A segmented fit
`...`	additional arguments

Details

`model.matrix.segmented`

Value

The design matrix for a segmented regression model with the specified formula and data

Author(s)

Vito Muggeo

See Also

See Also as `model.matrix`

---

model.matrix.stepmented *Design matrix for stepmented fits*

Description

This function builds the model matrix for `stepmented` fits.

Usage

```
## S3 method for class 'stepmented'
model.matrix(object, type=c("cdf","abs","none"), k=NULL, ...)
```

Arguments

`object`	A stepmented fit
`k`	The (negative) exponent of the sample size to approximate the absolute value; see `vcov.stepmented` for details.
`type`	The approximation for the indicator function/absolute value. If `"none"`, the simple matrix with the original indicator values is returned. `type='abs'` is not yet allowed.
`...`	additional arguments

Details

If `type="none"`, `model.matrix.stepmented` return the design matrix including the indicator function values and ignoring the psi terms.

Value

The design matrix for a stepmented regression model with the specified formula and data

Author(s)

Vito Muggeo

See Also

See Also as `model.matrix`, `vcov.stepmented` plant *Plan organ dataset*

Description

The `plant` data frame has 103 rows and 3 columns.

Usage

```
data(plant)
```

Format

A data frame with 103 observations on the following 3 variables:

`y` measurements of the plant organ.

`time` times where measurements took place.

`group` three attributes of the plant organ, `RKV`, `RKW`, `RWC`.

Details

Three attributes of a plant organ measured over time where biological reasoning indicates likelihood of multiple breakpoints. The data are scaled to the maximum value for each attribute and all attributes are measured at each time.

Source

The data have been kindly provided by Dr Zongjian Yang at School of Land, Crop and Food Sciences, The University of Queensland, Brisbane, Australia.

Examples

```
## Not run:
data(plant)

lattice::xyplot(y~time,groups=group,auto.key=list(space="right"), data=plant)

o<-segreg(y~ 0+group+seg(time, by=group, npsi=2), data=plant)
summary(o)

par(mfrow=c(1,2))
plot(y~time, data=plant)
plot(o, term=1:3, add=TRUE, leg=NA, psi.lines=TRUE) #add the lines to the current plot

plot(o, term=1:3, col=3:5, res.col=3:5, res=TRUE, leg="bottomright")


## End(Not run)
```

plot.segmented *Plot method for segmented objects*

Description

Takes a fitted segmented object returned by segmented() and plots (or adds) the fitted broken-line relationship for the selected segmented term.

Usage

```
## S3 method for class 'segmented'
plot(x, term, add=FALSE, res=FALSE, conf.level=0, interc=TRUE, link=TRUE,
   res.col=grey(.15, alpha = .4), rev.sgn=FALSE, const=NULL, shade=FALSE, rug=!add,
   dens.rug=FALSE, dens.col = grey(0.8), transf=I, isV=FALSE, is=FALSE, var.diff=FALSE,
   p.df="p", .vcov=NULL, .coef=NULL, prev.trend=FALSE, smoos=NULL, hide.zeros=FALSE,
    leg="topleft", psi.lines=FALSE, ...)
```

Arguments

- `x` a fitted segmented object.
- `term` Numerical or character to indicate the segmented variable having the piece-wise relationship to be plotted. If there is a single segmented variable in the fitted model x, term can be omitted. If vector, multiple segmented lines will be drawn on the same plot.
- `add` when TRUE the fitted lines are added to the current device.
- `res` when TRUE the fitted lines are plotted along with corresponding partial residuals. See Details.
- `conf.level` If greater than zero, it means the confidence level at which the pointwise confidence itervals have to be plotted.
- `interc` If TRUE the computed segmented components include the model intercept (if it exists).
- `link` when TRUE (default), the fitted lines are plotted on the link scale, otherwise they are tranformed on the response scale before plotting. Ignored for linear segmented fits.
- `res.col` when res=TRUE it means the color of the points representing the partial residuals.
- `rev.sgn` when TRUE it is assumed that current term is 'minus' the actual segmented variable, therefore the sign is reversed before plotting. This is useful when a null-constraint has been set on the last slope.
- `const` constant to add to each fitted segmented relationship (on the scale of the linear predictor) before plotting. If const=NULL and the fit includes a segmented interaction term (obtained via seg(..,by) in the formula), the group-specific intercept is included.

`shade`	if `TRUE` and `conf.level>0` it produces shaded regions (in grey color) for the pointwise confidence intervals embracing the fitted segmented line.
`rug`	when `TRUE` the covariate values are displayed as a rug plot at the foot of the plot. Default is to `!add`.
`dens.rug`	when `TRUE` then smooth covariate distribution is plotted on the x-axis.
`dens.col`	if `dens.rug=TRUE`, it means the colour to be used to plot the density.
`transf`	A possible function to convert the fitted values before plotting. It is only effective if `res=FALSE`. If `res=TRUE` any transformation is ignored.
`isV`	logical value (to be passed to `broken.line`). Ignored if `conf.level=0`
`is`	logical value (to be passed to `broken.line`) indicating if the covariance matrix based on the induced smoothing should be used. Ignored if `conf.level=0`
`var.diff`	logical value to be passed to `summary.segmented` to compute dthe standard errors of fitted values (if `conf.level>0`).
`p.df`	degrees of freedom when `var.diff=TRUE`, see `summary.segmented`
`.vcov`	The *full* covariance matrix of estimates to be used when `conf.level>0`. If unspecified (i.e. `NULL`), the covariance matrix is computed internally by the function `vcov.segmented`.
`.coef`	The regression parameter estimates. If unspecified (i.e. `NULL`), it is computed internally by `coef()`.
`prev.trend`	logical. If `TRUE` dashed lines corresponding to the 'previous' trends (i.e. the trends if the breakpoints would not have occurred) are also drawn.
`smoos`	logical, indicating if the residuals (provided that `res=TRUE`) will be drawn using a *smoothed* scatterplot. If `NULL` (default) the smoothed scatterplot will be employed when the number of observation is larger than 10000.
`hide.zeros`	logical, indicating if the residuals (provided that `res=TRUE`) corresponding to the covariate zero values should be deleted. Useful when the fit includes an interaction term in the formula, such as `seg(.., by=..)`, and the zeros in covariates indicate units in other groups.
`leg`	If the plot refers to segmented relationships in groups, i.e. `term` has been specified as a vector, a legend is placed at the specified `leg` position. Put `NA` not to draw the legend.
`psi.lines`	if `TRUE` vertical lines corresponding to the estimated breakpoints are also drawn. Ignored if `term` is not a vector.
`...`	other graphics parameters to pass to plotting commands: 'col', 'lwd' and 'lty' (that can be vectors and are recycled if necessary, see the example below) for the fitted piecewise lines; 'ylab', 'xlab', 'main', 'sub', 'cex.axis', 'cex.lab', 'xlim' and 'ylim' when a new plot is produced (i.e. when `add=FALSE`); 'pch' and 'cex' for the partial residuals (when `res=TRUE`, `res.col` is for the color); `col.shade` for the shaded regions (provided that `shade=TRUE` and `conf.level>0`).

Details

Produces (or adds to the current device) the fitted segmented relationship between the response and the selected `term`. If the fitted model includes just a single 'segmented' variable, `term` may be

omitted.
The partial residuals are computed as ‘fitted + residuals’, where ‘fitted’ are the fitted values of the segmented relationship relevant to the covariate specified in `term`. Notice that for GLMs the residuals are the response residuals if `link=FALSE` and the working residuals if `link=TRUE`.

Value

None.

Note

For models with offset, partial residuals on the response scale are not defined. Thus `plot.segmented` does not work when `link=FALSE`, `res=TRUE`, and the fitted model includes an offset.
When `term` is a vector and multiple segmented relationships are being drawn on the same plot, `col` and `res.col` can be vectors. Also `pch`, `cex`, `lty`, and `lwd` can be vectors, if specified.

Author(s)

Vito M. R. Muggeo

See Also

`segmented` to fit the model, `lines.segmented` to add the estimated breakpoints on the current plot. `points.segmented` to add the joinpoints of the segmented relationship. `predict.segmented` to compute standard errors and confidence intervals for predictions from a "segmented" fit.

Examples

```
set.seed(1234)
z<-runif(100)
y<-rpois(100,exp(2+1.8*pmax(z-.6,0)))
o<-glm(y~z,family=poisson)
o.seg<-segmented(o) #single segmented covariate and one breakpoint: 'seg.Z' and 'psi' can be omitted
par(mfrow=c(1,2))
plot(o.seg, conf.level=0.95, shade=TRUE)
points(o.seg, link=TRUE, col=2)
## new plot
plot(z,y)
## add the fitted lines using different colors and styles..
plot(o.seg,add=TRUE,link=FALSE,lwd=2,col=2:3, lty=c(1,3))
lines(o.seg,col=2,pch=19,bottom=FALSE,lwd=2) #for the CI for the breakpoint
points(o.seg,col=4, link=FALSE)
## using the options 'is', 'isV', 'shade' and 'col.shade'.
par(mfrow=c(1,2))
plot(o.seg, conf.level=.9, is=TRUE, isV=TRUE, col=1, shade = TRUE, col.shade=2)
plot(o.seg, conf.level=.9, is=TRUE, isV=FALSE, col=2, shade = TRUE, res=TRUE, res.col=4, pch=3)
```

plot.segmented.lme *Plot method for segmented mixed objects*

Description

Takes a fitted segmented.lme object returned by segmented.lme() and plots (or adds) the fitted broken-line relationship for the segmented term.

Usage

```
## S3 method for class 'segmented.lme'
plot(x, level=1, id = NULL, res = TRUE, pop = FALSE, yscale = 1, xscale = 1,
    n.plot, pos.leg = "topright", vline = FALSE, lines = TRUE,
    by=NULL, add=FALSE, conf.level=0, withI=TRUE, vcov.=NULL, shade=FALSE,
    drop.var=NULL, text.leg=NULL, id.name=TRUE, ...)
```

Arguments

x	Object of class "segmented.lme"
level	An integer giving the level of grouping to be used when computing the segmented relationship(s). level=0 means depending on fixed effects estimates only (such estimates are also said, to some extend, 'population' or 'marginal' estimates), otherewise the segmented lines will also depend on the random effects predictions.
id	A scalar or vector meaning which subjects profiles have to be drawn. If NULL (default) all profiles are drawn. Ignored if level=0.
res	If TRUE, the data points are also drawn. Ignored if level=0.
pop	if TRUE, the fitted segmented relationships based on fixed-effects only is also portrayed. Ignored if level=0.
yscale	If >= 0, the same and common y-scale is used for all 'subjects' (panels); otherwise the y-scale will depend on the actual (observed and fitted) values for each 'subject'.
xscale	If >= 0, the same and common x-scale is used for all 'subjects' (panels); otherwise the x-scale will depend on the actual observed values of the segmented covariate for each 'subject'.
n.plot	a vector to be passed to par(mfrow=c(..)) for plotting multiple panels (should be coherent with length(id)). If missing, it is computed automatically depending on length(id). Type n.plot=1 to draw all the segmented profiles on the same panel.
pos.leg	a character ('topright', 'topleft',...) meaning the location of the legend. Set NULL for no legend.
vline	logical, if TRUE a vertical dashed segment is added to emphasize the breakpoint location.

`lines`	logical, if `FALSE` points, rather than lines, are used to portray the segmented relationships.
`by`	A named list indicating covariate names and corresponding values affecting the fitted segmented relationship. For instance: `by=list(sex="male",z=.2)`, provided that the variables `sex` and `z` affect the segmented relationship. Effective only if `level=0`.
`add`	If `TRUE` the (fixed-effect) fitted segmented relationship is added to the current plot. Effective only if `level=0`.
`conf.level`	The confidence level for pointwise confidence intervals. Effective only if `level=0`.
`withI`	If `TRUE`, the level 0 segmented relationship is computed with the model intercept. Effective only if `level=0`.
`vcov.`	The fixed effects covariance matrix. If `NULL`, it is computed by `vcov.segmented.lme()`. Effective only if `level=0`.
`shade`	If `TRUE` (and `conf.level>0`) the area within the pointiwise CIs is shaded. Effective only if `level=0`.
`drop.var`	Possible coefficient names to be removed before computing the segmented relationship (E.g. the group-specific intercept.).
`text.leg`	If specified (and `pos.leg` has been also specified), it is the legend text. Effective only if `level=0`.
`id.name`	If `pos.leg` is different from `NULL`, `id.name=TRUE` will portray the cluster variable name along the value. Namely `id.name=TRUE` leads to 'country = italy' on each panel, while `id.name=FALSE` to 'italy'.
`...`	additional arguments, such as `ylab`,`xlab`, `ylim` and `xlim`; `l.col,l.lwd,l.lty` (for the fitted individual lines - can be vectors and will be recycled); `p.col, p.lwd, p.lty` for the population line (if `pop=TRUE`); `col, cex, pch` for the data points (if `res=TRUE`); `t.col` for the legend color, if `pos.leg` is not `NULL`. If `level=0` and `conf.level>0`, `lty` and `lwd` can be vectors.

Details

The function plots the 'subject'-specific segmented profiles for the 'subjects' specificed in `id` or, if `level=0`, the fitted segmented relationship based on fixed effects only. The number of panels to drawn is actually the minimum between `length(id)` and `prod(n.plot)`, but if `n.plot=c(1,1)` (or also simply `n.plot=1`), the 'individual' profiles will be pictured on the same panel.

Value

A single or multiple (depending on `level` and `id`) plot showing the fitted segmented profiles.

Warning

All the functions for segmented mixed models (*.segmented.lme) are still at an experimental stage

Note

If by is specified (and `level=0` is set), a legend is also added in the plot reporting covariate(s) name and value affecting the segmented relationship. Set `pos.leg=TRUE` to have no legend. On the other hand, use `text.leg` to add legend reporting the covariate baseline values.

Author(s)

Vito Muggeo

See Also

segmented.lme

Examples

```
## Not run:
#continues example from segmented.lme

plot(os, yscale=-1) #different y-scales

plot(os2, n.plot=1, l.col=2:6, l.lwd=2) #all segmented profiles on the same plot


## End(Not run)
```

---

plot.stepmented *Plot method for stepmented objects*

Description

Takes a fitted `stepmented` object returned by `stepmented()` and plots (or adds) the fitted piecewise constant lines for the selected stepmented term.

Usage

```
## S3 method for class 'stepmented'
plot(x, term, add = FALSE, res = TRUE, conf.level=0, interc = TRUE, add.fx = FALSE,
       psi.lines = TRUE, link=FALSE, const=NULL, res.col=grey(.15, alpha = .4),
    surf=FALSE, zero.cor=TRUE, heurs=TRUE, shade=FALSE, se.type=c("cdf","abs","none"),
      k=NULL, .vcov=NULL, leg="topleft", ...)
```

Arguments

`x`	a fitted `stepmented` object.
`term`	the stepmented variable having the piece-wise constant relationship to be plotted. If there is a single stepmented variable in the fitted model `x`, `term` can be omitted.
`add`	when `TRUE` the fitted lines are added to the current device.
`res`	when `TRUE` the fitted lines are plotted along with corresponding partial residuals.
`conf.level`	the confidence level for the pointwise confidence intervals for the expected values.
`interc`	if `TRUE` the computed components include the model intercept (if it exists).

add.fx
: logical. If TRUE and the object fit also includes an additional term for the same stepmented variable, the plot also portrays such 'additional' term.

psi.lines
: if TRUE vertical lines corresponding to the estimated changepoints are also drawn

link
: if FALSE the fitted lines (and possibily the residuals) are reported on the response scale. Ignored if the fit object x is not a glm-like fit.

const
: constant to add to each fitted segmented relationship (on the scale of the linear predictor) before plotting. If const=NULL and the fit includes a segmented interaction term (obtained via seg(..,by) in the formula), the group-specific intercept is included.

res.col
: when res=TRUE it means the color of the points representing the partial residuals.

surf
: if the object fit x includes 2 stepmented covariates (x1 and x2, say) with relevant estimated breakpoints, surf=TRUE will draw on the plane x1-x2 the areas splitted according to the estimated breakpoints with corresponding estimated means superimposed.

zero.cor
: see zero.cor in vcov.stepmented; effective only if conf.level>0.

heurs
: logical; if TRUE, heuristic (usually somewhat conservative) confidence intervals are computed and plotted; effective only if conf.level>0.

shade
: if TRUE the pointwise confidence intervals are portrayed via shaded area; effective only if conf.level>0.

se.type
: which standard errors should be computed? see type in vcov.stepmented; effective only if conf.level>0.

k
: The value to be passed to vcov.stepmented to computed the standard errors.

.vcov
: The estimate var-covariance matrix; if NULL, it is computed internally by vcov.stepmented.

leg
: If the plot refers to stepmented relationships in groups, i.e. term has been specified as a vector, a legend is placed at the specified leg position. Put NA not to draw the legend.

...
: other graphics parameters to pass to plotting commands: 'col', 'lwd' and 'lty' (that can be vectors and are recycled if necessary, see the example below) for the fitted piecewise constant lines; 'ylab', 'xlab', 'main', 'sub', 'cex.axis', 'cex.lab', 'xlim' and 'ylim' when a new plot is produced (i.e. when add=FALSE); 'pch' and 'cex' for the partial residuals (when res=TRUE, res.col is for the color).

Details

Produces (or adds to the current device) the fitted step-function like relationship between the response and the selected term. If the fitted model includes just a single 'stepmented' variable, term may be omitted. If surf=TRUE, and res=TRUE the point widths are proportional to the partial residual values.

Value

None.

Note

Implementation of confidence intervals for the conditional means in stepmented regression is under development; `conf.level>0` should be used with care, especially with multiple jumpoints.

Author(s)

Vito M. R. Muggeo

See Also

See Also as `stepmented`

Examples

```
#Following code in stepmented..
## Not run:

par(mfrow=c(1,3))

plot(os,"x")
plot(os,"z")
plot(os,"z", add.fx=TRUE, psi.lines=FALSE )
lines(os, "z")

#display the 'surface'
par(mfrow=c(1,3))
plot(os, surf=TRUE, col=1, res.col=2)
plot(os, surf=TRUE, lty=2)
plot(x,z)
plot(os, surf=TRUE, add=TRUE, col=4, res=FALSE)


## End(Not run)
```

`points.segmented` *Points method for segmented objects*

Description

Takes a fitted `segmented` object returned by `segmented()` and adds on the current plot the joinpoints of the fitted broken-line relationships.

Usage

```
  ## S3 method for class 'segmented'
points(x, term, interc = TRUE, link = TRUE, rev.sgn=FALSE,
    transf=I, .vcov=NULL, .coef=NULL, const=0, v=TRUE, ...)
```

Arguments

- `x` — an object of class `segmented`.
- `term` — the segmented variable of interest. It may be unspecified when there is a single segmented variable.
- `interc` — If `TRUE` the computed joinpoints include the model intercept (if it exists).
- `link` — when `TRUE` (default), the fitted joinpoints are plotted on the link scale
- `rev.sgn` — when `TRUE`, the fitted joinpoints are plotted on the 'minus' scale of the current `term` variable. This is useful when a null-constraint has been set on the last slope.
- `transf` — A possible function to convert the fitted values before plotting.
- `.vcov` — The *full* covariance matrix of estimates. If unspecified (i.e. `NULL`), the covariance matrix is computed internally by `vcov()`.
- `.coef` — The regression parameter estimates. If unspecified (i.e. `NULL`), it is computed internally by `coef(x)`.
- `const` — A constant to be added (on the y-scale) to the values before transforming and plotting.
- `v` — logical. If `TRUE`, vertical lines at the breakpoints are also drawn.
- `...` — other graphics parameters to pass on to `points()` and `segments()` (if `v=TRUE`).

Details

We call 'joinpoint' the plane point having as coordinates the breakpoints (on the x scale) and the fitted values of the segmented relationship at that breakpoints (on the y scale). `points.segmented()` simply adds the fitted joinpoints on the current plot. This could be useful to emphasize the changes of the piecewise linear relationship.

See Also

`plot.segmented` to plot the fitted segmented lines.

Examples

```
## Not run:
#see examples in ?plot.segmented

## End(Not run)
```

predict.segmented *Predict method for segmented model fits*

Description

Returns predictions and optionally associated quantities (standard errors or confidence intervals) from a fitted segmented model object.

Usage

```
## S3 method for class 'segmented'
predict(object, newdata, se.fit=FALSE, interval=c("none","confidence", "prediction"),
        type = c("link", "response"), na.action=na.omit, level=0.95, .coef=NULL, ...)
```

Arguments

`object`	a fitted segmented model coming from `segmented.(g)lm` or `segreg`.
`newdata`	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
`se.fit`	Logical. Should the standard errors be returned?
`interval`	Which interval? See `predict.lm`
`type`	Predictions on the link or response scale? Only if `object` is a segmented glm.
`na.action`	How to deal with missing data, *if* `newdata` include them.
`level`	The confidence level.
`.coef`	The regression parameter estimates. If unspecified (i.e. `NULL`), it is computed internally by `coef()`.
`...`	further arguments.

Details

Basically `predict.segmented` builds the right design matrix accounting for breakpoint and passes it to `predict.lm` or `predict.glm` depending on the actual model fit `object`.

Value

`predict.segmented` produces a vector of predictions with possibly associated standard errors or confidence intervals. See `predict.lm` or `predict.glm`.

Warning

For segmented glm fits with offset obtained starting from the model `glm(.., offset=..)`, `predict.segmented` returns the fitted values without considering the offset.

Author(s)

Vito Muggeo

See Also

segmented, plot.segmented, broken.line, predict.lm, predict.glm

Examples

```
n=10
x=seq(-3,3,l=n)
set.seed(1515)
y <- (x<0)*x/2 + 1 + rnorm(x,sd=0.15)
segm <- segmented(lm(y ~ x), ~ x, psi=0.5)
predict(segm,se.fit = TRUE)$se.fit

#wrong (smaller) st.errors (assuming known the breakpoint)
olm<-lm(y~x+pmax(x-segm$psi[,2],0))
predict(olm,se.fit = TRUE)$se.fit
```

---

predict.stepmented *Predict method for stepmented model fits*

Description

Returns predictions and optionally associated quantities (standard errors or confidence intervals) from a fitted stepmented model object.

Usage

```
## S3 method for class 'stepmented'
predict(object, newdata, se.fit=FALSE, interval=c("none","confidence", "prediction"),
         type = c("link", "response"), na.action=na.omit, level=0.95, .coef=NULL,
          .vcov=NULL, apprx.fit=c("none","cdf"), apprx.se=c("cdf","none"), ...)
```

Arguments

`object`	a fitted stepmented model coming from `stepmented.lm` or `stepmented.glm`.
`newdata`	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
`se.fit`	Logical. Should the standard errors be returned?
`interval`	Which interval? See predict.lm
`type`	Predictions on the link or response scale? Only if `object` is a stepmented glm.
`na.action`	How to deal with missing data, *if* `newdata` include them.
`level`	The confidence level.
`.coef`	The regression parameter estimates. If unspecified (i.e. `NULL`), it is computed internally by `coef()`.
`.vcov`	The estimate covariance matrix. If unspecified (i.e. `NULL`), it is computed internally by `vcov.stepmented()`.

`apprx.fit`	The approximation of the $(x > \hat{\psi})$ used to compute the predictions/fitted values of the piece-wise relationships.
`apprx.se`	The same abovementioned approximation to compute the standard error.
`...`	further arguments, for instance `k` to be passed to `vcov.stepmented`.

Details

Basically `predict.stepmented` builds the right design matrix accounting for breakpoint and passes it to `predict.lm` or `predict.glm` depending on the actual model fit `object`.

Value

`predict.stepmented` produces a vector of predictions with possibly associated standard errors or confidence intervals. See `predict.lm`, `predict.glm`, or `predict.segmented`.

Warning

For stepmented glm fits with offset obtained starting from the model `glm(.., offset=..)`, `predict.stepmented` returns the fitted values without considering the offset.

Author(s)

Vito Muggeo

See Also

`stepmented`, `plot.stepmented`, `predict.lm`, `predict.glm`

Examples

```
n=10
x=seq(-3,3,l=n)
set.seed(1515)
y <- (x<0)*x/2 + 1 + rnorm(x,sd=0.15)
segm <- segmented(lm(y ~ x), ~ x, psi=0.5)
predict(segm,se.fit = TRUE)$se.fit
```

`print.segmented` *Print method for the segmented class*

Description

Printing the most important features and coefficients (including the breakpoints) of a segmented model.

Usage

```
## S3 method for class 'segmented'
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'segmented'
coef(object, include.psi=FALSE, ...)
```

Arguments

x	object of class segmented
digits	number of digits to be printed
object	object of class segmented
include.psi	logical. If TRUE, the breakpoints are returned along with the regression coefficients
...	arguments passed to other functions

Author(s)

Vito M.R. Muggeo

See Also

summary.segmented, print.summary.segmented

---

print.segmented.lme *Print method for the segmented.lme class*

Description

Printing and extracting the most important features of a segmented mixed model.

Usage

```
## S3 method for class 'segmented.lme'
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'segmented.lme'
fixef(object, ...)

## S3 method for class 'segmented.lme'
logLik(object, ...)
```

Arguments

`x`	object of class `segmented.lme`
`digits`	number of digits to be printed
`object`	object of class `segmented`
`...`	arguments passed to other functions

Author(s)

Vito M.R. Muggeo

See Also

`segmented.lme`, `summary.segmented.lme`

`pscore.test` *Testing for existence of one breakpoint*

Description

Given a (generalized) linear model, the (pseudo) Score statistic tests for the existence of one breakpoint.

Usage

```
pscore.test(obj, seg.Z, k = 10, alternative = c("two.sided", "less", "greater"),
    values=NULL, dispersion=NULL, df.t=NULL, more.break=FALSE, n.break=1,
    only.term=FALSE, break.type=c("break","jump"))
```

Arguments

`obj`	a fitted model typically returned by `glm` or `lm`. Even an object returned by `segmented` can be set. Offset and weights are allowed.
`seg.Z`	a formula with no response variable, such as `seg.Z=~x1`, indicating the (continuous) segmented variable being tested. Only a single variable may be tested and an error is printed when `seg.Z` includes two or more terms. `seg.Z` can be omitted if i)`obj` is a segmented fit with a single segmented covariate (and that variable is taken), or ii)if it is a "lm" or "glm" fit with a single covariate (and that variable is taken).
`k`	optional. Number of points (equi-spaced from the min to max) used to compute the pseudo Score statistic. See Details.
`alternative`	a character string specifying the alternative hypothesis (relevant to the slope difference parameter).
`values`	optional. The evaluation points where the Score test is computed. See Details for default values.

`dispersion`	optional. the dispersion parameter for the family to be used to compute the test statistic. When `NULL` (the default), it is inferred from `obj`. Namely it is taken as `1` for the Binomial and Poisson families, and otherwise estimated by the residual Chi-squared statistic in the model `obj` (calculated from cases with non-zero weights divided by the residual degrees of freedom).
`df.t`	optional. The degress-of-freedom used to compute the p-value. When `NULL`, the df extracted from `obj` are used.
`more.break`	optional, logical. If `obj` is a 'segmented' fit, `more.break=FALSE` tests for the actual breakpoint for the variable 'seg.Z', while `more.break=TRUE` tests for an *additional* breakpoint(s) for the variable 'seg.Z'. Ignored when `obj` is not a segmented fit.
`n.break`	optional. Number of breakpoints postuled under the alternative hypothesis.
`only.term`	logical. If `TRUE`, only the pseudo covariate(s) relevant to the testing for the breakpoint is returned, and no test is computed.
`break.type`	The kind of breakpoint being tested. `"break"` is for piecewise-linear relationships, `"jump"` means piecewise-constant, i.e. a step-function, relationships.

Details

`pscore.test` tests for a non-zero difference-in-slope parameter of a segmented relationship. Namely, the null hypothesis is $H_0 : \beta = 0$, where β is the difference-in-slopes, i.e. the coefficient of the segmented function $\beta(x - \psi)_+$. The hypothesis of interest $\beta = 0$ means no breakpoint. Simulation studies have shown that such Score test is more powerful than the Davies test (see reference) when the alternative hypothesis is 'one changepoint'. If there are two or more breakpoints (for instance, a sinusoidal-like relationships), `pscore.test` can have lower power, and `davies.test` can perform better.

The `dispersion` value, if unspecified, is taken from `obj`. If `obj` represents the fit under the null hypothesis (no changepoint), the dispersion parameter estimate will be usually larger, leading to a (potentially severe) loss of power.

The `k` evaluation points are `k` equally spaced values in the range of the segmented covariate. `k` should not be small. Specific values can be set via `values`, although I have found no important difference due to number and location of the evaluation points, thus default is `k=10` equally-spaced points. However, when the possible breakpoint is believed to lie into a specified narrower range, the user can specify `k` values in that range leading to higher power in detecting it, i.e. typically lower p-value.

If `obj` is a (segmented) *lm* object, the returned p-value comes from the t-distribution with appropriate degrees of freedom. Otherwise, namely if `obj` is a (segmented) *glm* object, the p-value is computed wrt the Normal distribution.

Value

A list with class '`htest`' containing the following components:

`method`	title (character)
`data.name`	the regression model and the segmented variable being tested
`statistic`	the empirical value of the statistic

parameter	number of evaluation points
p.value	the p-value
process	the alternative hypothesis set

Author(s)

Vito M.R. Muggeo

References

Muggeo, V.M.R. (2016) Testing with a nuisance parameter present only under the alternative: a score-based approach with application to segmented modelling. *J of Statistical Computation and Simulation*, **86**, 3059–3067.

See Also

See also davies.test.

Examples

```
## Not run:
set.seed(20)
z<-runif(100)
x<-rnorm(100,2)
y<-2+10*pmax(z-.5,0)+rnorm(100,0,3)

o<-lm(y~z+x)

#testing for one changepoint
#use the simple null fit
pscore.test(o,~z) #compare with davies.test(o,~z)..

#use the segmented fit
os<-segmented(o, ~z)
pscore.test(os,~z) #smaller p-value, as it uses the dispersion under the alternative (from 'os')

#test for the 2nd breakpoint in the variable z
pscore.test(os,~z, more.break=TRUE)


## End(Not run)
```

---

pwr.seg *Power Analysis in segmented regression*

Description

Given the appropriate input values, the function computes the power (sample size) corresponding to the specifed sample size (power). If a segmented fit object is provided, the power is computed taking the parameter estimates as input values.

Usage

```
pwr.seg(oseg, pow, n, z = "1:n/n", psi, d, s, n.range = c(10,300),
    X = NULL, break.type=c("break","jump"), alpha = 0.01, round.n = TRUE,
    alternative = c("two.sided", "greater", "less"), msg = TRUE, ci.pow=0)
```

Arguments

`oseg`	The fitted segmented object. If provided, the power is computed at the model parameter estimates, and all the remaining arguments but `alternative` and `alpha` are ignored.
`pow`	The desired power level. If provided `n` has to be missing
`n`	The fixed sample size. If provided `pow` has to be missing
`z`	The covariate understood to have a segmented effect. Default is `"1:n/n"`, i.e. equispaced values in (0,1). More generally a string indicating the quantile function having `p` and possible other numerical values as arguments. For istance `"qunif(p,0,1)"`, `"qnorm(p,2,5)"`, or `"qexp(p)"`. `"qunif(p,1,n)"` can be also specified, but attention should be paid to guarantee `psi` within the covariate range. Finally, it could be also a numerical vector meaning the actual covariate, but `pow` has to be missing. Namely if the covariate is supplied (and `n` is known), only the relevant power can be estimated.
`psi`	The breakpoint value within the covariate range
`d`	The slope difference
`s`	The response standard deviation
`n.range`	When `pow` is provided and the relevant sample size estimate has to be returned, the function evaluates 50 sample sizes equally spaced in `n.range`. However the function can also compute, via spline interpolation, sample sizes outside the specified range.
`X`	The design matrix including additional linear variables in the regression equation. Default to `NULL` which means intercept and linear term for the segmented covariate.
`break.type`	Type of breakpoint. `break.type='break'` means piecewise linear (segmented), `break.type='jump'` refers to piecewise constant.
`alpha`	The type-I error probability. Default to 0.01.
`round.n`	logical. If `TRUE` the (possible) returned sample size value is rounded.
`alternative`	a character string specifying the alternative hypothesis, must be one of "two.sided", "greater" or "less". Note, this refers to the sign of the slope difference.
`msg`	logical. If `TRUE` the output is returned along with a simple message, otherwise only the values are returned
`ci.pow`	Numerical. If `oseg` has been supplied, `ci.pow` replicates are drawn to build a 95% confidence interval for the power.

Details

The function exploits the sampling distribution of the pseudo Score statistic under the alternative hypothesis of one breakpoint.

Value

The computed power *or* sample size, with or without message (depending on `msg`)

Note

Currently the function assumes just 1 breakpoint in one covariate

Author(s)

Nicoletta D'Angelo and Vito Muggeo

References

D'Angelo N, Muggeo V.M.R. (2021) Power analysis in segmented regression, working paper https://www.researchgate.net/publication/355885747.

Muggeo, V.M.R. (2016) Testing with a nuisance parameter present only under the alternative: a score-based approach with application to segmented modelling. *J of Statistical Computation and Simulation*, **86**, 3059–3067.

See Also

`pscore.test`

Examples

```
## pwr.seg(pow=.7, psi=.5, d=1.5, s=.5) #returns the sample size
## pwr.seg(n=219, psi=.5, d=1.5, s=.5) #returns the power
## pwr.seg(n=20,z="qnorm(p, 2,5)", psi=3, d=.5, s=2) #the covariate is N(2,5)
## pwr.seg(n=20,z="qexp(p)", psi=.1, d=.5, s=.1) #the covariate is Exp(1)
```

---

seg — *Specifying a segmented/stepmented term in the segreg/stepreg formula*

Description

Function used to define a segmented (stepmented) term within the segreg (stepreg) formula. The function simply passes relevant information to proper fitter functions.

Usage

```
seg(x, npsi = 1, psi = NA, est = NA, R = NA, fixed.psi = NULL, by = NULL, f.x = I)
```

Arguments

- `x` — The segmented/stepmented (numeric) covariate
- `npsi` — The number of breakpoints/jumpoints to estimate. Default to `npsi=1`. If `by` has been specified, the same `npsi` applies to all categories of the factor `by`; otherwise it can be vector, wherein the entries represent the number of breakpoints of the segmented relationships within the categories of `by`. If `npsi` is specified as a vector, the corresponding `est` should be a list. The `npsi` starting values are computed according the specification of `quant` in `seg.control`.
- `psi` — Numerical vector indicating possible starting value(s) for the breakpoint(s). When provided, `psi` overwrites `npsi`. If `by` has been specified, `psi` can be a list, wherein the components represent the starting values of the segmented/stepmented relationships within the categories of `by`.
- `est` — Possible vector (of length equal to `npsi+1`) of 0's and 1's to indicate whether the slopes have to be estimated or fixed to zero. `NA`, the default, means all ones, namely every slope is estimated. Consecutive zeroes are not allowed. If `by` has been specified, the same `est` applies to all categories of the factor `by`; otherwise `est` can be a list, wherein the components represent the slope constraints of the segmented relationships within the categories of `by`. Note that, when a grouping variable has been specified in `by`, `npsi` can be specified as a vector, and `est` should be a list.
- `R` — Matrix to constrain the slopes. If provided, it overwrites the matrix (which is built internally) coming from the specification of `est`.
- `fixed.psi` — Possible *fixed* breakpoint values to be accounted for *in addition* to the estimated ones; `slope` and `plot.segmented` will account for them correctly.
- `by` — A possible *factor* meaning an interaction with the segmented term `x`. Hence, if specified, a different segmented relationship is fitted within each category of `by`.
- `f.x` — an optional function meaning a function to apply to the covariate before fitting

Details

The function is used within `segreg` and `stepreg` to 'build' information about the segmented relationships to fit.

Value

The function simply returns the covariate with added attributes relevant to segmented term

Note

If any value is provided in `fix.psi`, the corresponding slope difference coefficient will be labelled by `*.fixed.*`. The `slope` function will compute the 'right' slopes also accounting for the fixed breakpoints.

Author(s)

Vito Muggeo

See Also

segreg

Examples

```
##see ?segreg
```

---

seg.control *Auxiliary for controlling segmented/stepmented model fitting*

Description

Auxiliary function as user interface for 'segmented' and 'stepmented' fitting. Typically only used when calling any 'segmented' or 'stepmented' method.

Usage

```
seg.control(n.boot=10, display = FALSE, tol = 1e-05, it.max = 30, fix.npsi=TRUE,
   K = 10, quant = FALSE, maxit.glm = NULL, h = 1.25, break.boot=5, size.boot=NULL,
    jt=FALSE, nonParam=TRUE, random=TRUE, seed=NULL, fn.obj=NULL, digits=NULL,
    alpha = NULL, fc=.95, check.next=TRUE, tol.opt=NULL, fit.psi0=NULL, eta=NULL,
    min.nj=2)
```

Arguments

`n.boot`	number of bootstrap samples used in the bootstrap restarting algorithm. If 0 the standard algorithm, i.e. without bootstrap restart, is used. Default to 10 that appears to be sufficient in most of problems. However when multiple breakpoints have to be estimated it is suggested to increase `n.boot`, e.g. `n.boot=50`.
`display`	logical indicating if the value of the objective function should be printed along with current breakpoint estimates at each iteration or at each bootstrap resample (but no more than 5 breakpoints are printed). If bootstrap restarting is employed, the values of objective and breakpoint estimates should not change at the last runs.
`tol`	positive convergence tolerance.
`it.max`	integer giving the maximal number of iterations.
`fix.npsi`	logical (it replaces previous argument `stop.if.error`) If `TRUE` (default) the *number* (and not location) of breakpoints is held fixed throughout iterations. Otherwise a sort of 'automatic' breakpoint selection is carried out, provided that several starting values are supplied for the breakpoints, see argument `psi` in `segmented.lm` or `segmented.glm`. The idea, relying on removing the 'non-admissible' breakpoint estimates at each iteration, is discussed in Muggeo and Adelfio (2011) and it is not compatible with the bootstrap restart algorithm. `fix.npsi=FALSE`, indeed, should be considered as a preliminary and tentative approach to deal with an unknown number of breakpoints. Alternatively, see `selgmented`.

- `K` — the number of quantiles (or equally-spaced values) to supply as starting values for the breakpoints when the `psi` argument of `segmented` is set to `NA`. `K` is ignored when `psi` is different from `NA`.
- `quant` — logical, indicating how the starting values should be selected. If `FALSE` equally-spaced values are used, otherwise the quantiles. Ignored when `psi` is different from `NA`.
- `maxit.glm` — integer giving the maximum number of inner IWLS iterations (see details). If `NULL`, the number is low in the first iterations and then increases as th eprocess goes on. Ignored for segmented lm fits
- `h` — positive factor modifying the increments in breakpoint updates during the estimation process (see details).
- `break.boot` — Integer, less than `n.boot`. If `break.boot` consecutive bootstrap samples lead to the same objective function during the estimation process, the algorithm stops without performing all `n.boot` 'trials'. This can save computational time considerably. Default is `5` for the `segmented` and `5+2` for the `stepmented` functions.
- `size.boot` — the size of the bootstrap samples. If `NULL`, it is taken equal to the actual sample size. If the sample is very large, the idea is to run bootstrap restarting using smaller bootstrap samples.
- `jt` — logical. If `TRUE` the values of the segmented variable(s) are jittered before fitting the model to the bootstrap resamples.
- `nonParam` — if `TRUE` nonparametric bootstrap (i.e. case-resampling) is used, otherwise residual-based. Currently working only for LM fits. It is not clear what residuals should be used for GLMs.
- `random` — if `TRUE`, when the algorithm fails to obtain a solution, random values are employed to obtain candidate values.
- `seed` — The seed to be passed on to `set.seed()` when `n.boot>0`. If `NULL`, a seed depending on the response values is generated and used. Otherwise it can be a numerical value or, if `NA`, a random value is generated. Fixing the seed can be useful to replicate *exactly* the results when the bootstrap restart algorithm is employed. Whichever choice, the segmented fit includes the component `seed` representing the value saved just before the bootstrap resampling. Re-use it if you want to replicate the bootstrap restarting algorithm with the *same* re-samples.
- `fn.obj` — A *character string* to be used (optionally) only when `segmented.default` is used. It represents the function (with argument `'x'`) to be applied to the fit object to extract the objective function to be *minimized*. Thus for `"lm"` fits (although unnecessary) it should be `fn.obj="sum(x$residuals^2)"`, for `"coxph"` fits it should be `fn.obj="-x$loglik[2]"`. If `NULL` the 'minus log likelihood' extracted from the object, namely `"-logLik(x)"`, is used. See `segmented.default`.
- `digits` — optional. If specified it means the desidered number of decimal points of the breakpoint to be used during the iterative algorithm.
- `alpha` — optional numerical values. The breakpoints are estimated within the quantiles `alpha[1]` and `alpha[2]` of the relevant covariate. If a single value is provided, it is assumed `alpha` and `1-alpha`. Defaults to `NULL` which means `alpha=max(.05, 1/n)`. Note: Providing `alpha=c(mean(x<=a),mean(x<=b))` means to constrain the breakpoint estimates within $[a, b]$.

`fc` A proportionality factor (≤ 1) to adjust the breakpoint estimates *if* these come close to the boundary or too close each other. For instance, if `psi` turns up close to the maximum, it will be changed to `psi*fc` or to `psi/fc` if close to the minimum. This is useful to get finite point estimate and standard errors for each slope paramete.

`check.next` logical, effective only for stepmented fit. If `TRUE` the solutions next to the current one are also investigated.

`tol.opt` Numerical value to be passed to `tol` in `optimize`.

`fit.psi0` Possible list including preliminary values.

`eta` Only for segmented/stepmented fits: starting values to be passed to `etastart` in `glm.fit`.

`min.nj` How many observations (at least) should be in the covariate intervals induced by the breakpoints?

Details

Fitting a 'segmented' GLM model is attained via fitting iteratively standard GLMs. The number of (outer) iterations is governed by `it.max`, while the (maximum) number of (inner) iterations to fit the GLM at each fixed value of psi is fixed via `maxit.glm`. Usually three-four inner iterations may be sufficient.

When the starting value for the breakpoints is set to `NA` for any segmented variable specified in `seg.Z`, `K` values (quantiles or equally-spaced) are selected as starting values for the breakpoints.

Since version 0.2-9.0 `segmented` implements the bootstrap restarting algorithm described in Wood (2001). The bootstrap restarting is expected to escape the local optima of the objective function when the segmented relationship is noisy and the loglikelihood can be flat. Notice bootstrap restart runs `n.boot` iterations regardless of `tol` that only affects convergence within the inner loop.

Value

A list with the arguments as components.

Author(s)

Vito Muggeo

References

Muggeo, V.M.R., Adelfio, G. (2011) Efficient change point detection in genomic sequences of continuous measurements. *Bioinformatics* **27**, 161–166.

Wood, S. N. (2001) Minimizing model fitting objectives that contain spurious local minima by bootstrap restarting. *Biometrics* **57**, 240–244.

Examples

```
#decrease the maximum number inner iterations and display the
#evolution of the (outer) iterations
```

```
#seg.control(display = TRUE, maxit.glm=4)
```

seg.lm.fit *Fitter Functions for Segmented Linear Models*

Description

seg.lm.fit is called by segmented.lm to fit segmented linear (gaussian) models. Likewise, seg.glm.fit is called by segmented.glm to fit generalized segmented linear models, and seg.def.fit is called by segmented.default to fit segmented relationships in general regression models (e.g., quantile regression and Cox regression). seg.lm.fit.boot, seg.glm.fit.boot, and seg.def.fit.boot are employed to perform bootstrap restart. The functions segConstr.* are called by segreg() when some contraints are set on the slopes of the segmented relationships.

These functions should usually not be used directly by the user.

Usage

```
seg.lm.fit(y, XREG, Z, PSI, w, offs, opz, return.all.sol=FALSE)

seg.lm.fit.boot(y, XREG, Z, PSI, w, offs, opz, n.boot=10,
    size.boot=NULL, jt=FALSE, nonParam=TRUE, random=FALSE, break.boot=n.boot)

seg.glm.fit(y, XREG, Z, PSI, w, offs, opz, return.all.sol=FALSE)

seg.glm.fit.boot(y, XREG, Z, PSI, w, offs, opz, n.boot=10,
    size.boot=NULL, jt=FALSE, nonParam=TRUE, random=FALSE, break.boot=n.boot)

seg.def.fit(obj, Z, PSI, mfExt, opz, return.all.sol=FALSE)

seg.def.fit.boot(obj, Z, PSI, mfExt, opz, n.boot=10, size.boot=NULL,
    jt=FALSE, nonParam=TRUE, random=FALSE, break.boot=n.boot)

seg.Ar.fit(obj, XREG, Z, PSI, opz, return.all.sol=FALSE)

seg.Ar.fit.boot(obj, XREG, Z, PSI, opz, n.boot=10, size.boot=NULL, jt=FALSE,
    nonParam=TRUE, random=FALSE, break.boot=n.boot)

seg.num.fit(y, XREG, Z, PSI, w, opz, return.all.sol=FALSE)

seg.num.fit.boot(y, XREG, Z, PSI, w, opz, n.boot=10, size.boot=NULL, jt=FALSE,
    nonParam=TRUE, random=FALSE, break.boot=n.boot)

segConstr.lm.fit(y, XREG, Z, PSI, w, offs, opz, return.all.sol = FALSE)

segConstr.lm.fit.boot(y, XREG, Z, PSI, w, offs, opz, n.boot=10, size.boot=NULL,
    jt=FALSE, nonParam=TRUE, random=FALSE, break.boot=n.boot)
```

```
segConstr.glm.fit(y, XREG, Z, PSI, w, offs, opz, return.all.sol = FALSE)

segConstr.glm.fit.boot(y, XREG, Z, PSI, w, offs, opz, n.boot=10, size.boot=NULL,
    jt=FALSE, nonParam=TRUE, random=FALSE, break.boot=n.boot)
```

Arguments

`y`	vector of observations of length `n`.
`XREG`	design matrix for standard linear terms.
`Z`	appropriate matrix including the segmented variables whose breakpoints have to be estimated.
`PSI`	appropriate matrix including the starting values of the breakpoints to be estimated.
`w`	possibe weights vector.
`offs`	possibe offset vector.
`opz`	a list including information useful for model fitting.
`n.boot`	the number of bootstrap samples employed in the bootstrap restart algorithm.
`break.boot`	Integer, less than `n.boot`. If `break.boot` consecutive bootstrap samples lead to the same objective function, the algorithm stops without performing all `n.boot` 'trials'. This can save computational time considerably.
`size.boot`	the size of the bootstrap resamples. If `NULL` (default), it is taken equal to the sample size. values smaller than the sample size are expected to increase perturbation in the bootstrap resamples.
`jt`	logical. If `TRUE` the values of the segmented variable(s) are jittered before fitting the model to the bootstrap resamples.
`nonParam`	if `TRUE` nonparametric bootstrap (i.e. case-resampling) is used, otherwise residual-based.
`random`	if `TRUE`, when the algorithm fails to obtain a solution, random values are used as candidate values.
`return.all.sol`	if `TRUE`, when the algorithm fails to obtain a solution, the values visited by the algorithm with corresponding deviances are returned.
`obj`	the starting regression model where the segmented relationships have to be added.
`mfExt`	the model frame.

Details

The functions call iteratively `lm.wfit` (or `glm.fit`) with proper design matrix depending on `XREG`, `Z` and `PSI`. `seg.lm.fit.boot` (and `seg.glm.fit.boot`) implements the bootstrap restarting idea discussed in Wood (2001).

Value

A list of fit information.

Note

These functions should usually not be used directly by the user.

Author(s)

Vito Muggeo

References

Wood, S. N. (2001) Minimizing model fitting objectives that contain spurious local minima by bootstrap restarting. *Biometrics* **57**, 240–244.

See Also

segmented.lm, segmented.glm

Examples

```
##See ?segmented
```

`segmented` *Segmented relationships in regression models*

Description

Fits regression models with segmented relationships between the response and one or more explanatory variables. Break-point estimates are provided.

Usage

```
segmented(obj, seg.Z, psi, npsi, fixed.psi=NULL, control = seg.control(),
    model = TRUE, ...)

## Default S3 method:
segmented(obj, seg.Z, psi, npsi, fixed.psi=NULL, control = seg.control(),
    model = TRUE, keep.class=FALSE, ...)

## S3 method for class 'lm'
segmented(obj, seg.Z, psi, npsi, fixed.psi=NULL, control = seg.control(),
    model = TRUE, keep.class=FALSE, ...)

## S3 method for class 'glm'
segmented(obj, seg.Z, psi, npsi, fixed.psi=NULL, control = seg.control(),
    model = TRUE, keep.class=FALSE, ...)

## S3 method for class 'Arima'
segmented(obj, seg.Z, psi, npsi, fixed.psi=NULL, control = seg.control(),
```

```
    model = TRUE, keep.class=FALSE, ...)

## S3 method for class 'numeric'
segmented(obj, seg.Z, psi, npsi, fixed.psi=NULL, control = seg.control(),
    model = TRUE, keep.class=FALSE, adjX=FALSE, weights=NULL, ...)
```

Arguments

`obj` standard 'linear' model of class "lm", "glm" or "Arima", or potentially any regression fit may be supplied since version 0.5-0 (see 'Details'). `obj` can include any covariate understood to have a linear (i.e. no break-points) effect on the response. If `obj` also includes the segmented covariate specified in `seg.Z`, then all the slopes of the fitted segmented relationship will be estimated. On the other hand, if `obj` misses the segmented variable, then the 1st (the leftmost) slope is assumed to be zero. Since version 1.5.0, `obj` can be a simple numeric or `ts` object but with only a single segmented variable (`segmented.numeric`) see examples below.

`seg.Z` the segmented variable(s), i.e. the continuous covariate(s) understood to have a piecewise-linear relationship with response. It is a formula with no response variable, such as `seg.Z=~x` or `seg.Z=~x1+x2`. It can be missing when `obj` includes only one covariate which is taken as segmented variable. Currently, formulas involving functions, such as `seg.Z=~log(x1)`, or selection operators, such as `seg.Z=~d[,"x1"]` or `seg.Z=~d$x1`, are *not* allowed. Also, variable names formed by `U` or `V` only (with or without numbers) are not permitted.

`psi` starting values for the breakpoints to be estimated. If there is a single segmented variable specified in `seg.Z`, `psi` is a numeric vector, and it can be missing when 1 breakpoint has to be estimated (and the median of the segmented variable is used as a starting value). If `seg.Z` includes several covariates, `psi` has be specified as a *named* list of vectors whose names have to match the variables in the `seg.Z` argument. Each vector of such list includes starting values for the break-point(s) for the corresponding variable in `seg.Z`. A `NA` value means that '`K`' quantiles (or equally spaced values) are used as starting values; `K` is fixed via the `seg.control` auxiliary function.

`npsi` A named vector or list meaning the *number* (and not locations) of breakpoints to be estimated. The starting values will be internally computed via the quantiles or equally spaced values, as specified in argument `quant` in `seg.control`. `npsi` can be missing and `npsi=1` is assumed for all variables specified in `seg.Z`. If `psi` is provided, `npsi` is ignored.

`fixed.psi` An optional named list meaning the breakpoints to be kept fixed during the estimation procedure. The names should be a subset of (or even the same) variables specified in `seg.Z`. If there is a single variable in `seg.Z`, a simple numeric vector can be specified. Note that, in addition to the values specified here, `segmented` will estimate additional breakpoints. To keep fixed all breakpoints (to be specified in `psi`) use `it.max=0` in `seg.control`

`control` a list of parameters for controlling the fitting process. See the documentation for `seg.control` for details.

`model` logical value indicating if the model.frame should be returned.

keep.class — logical value indicating if the final fit returned by `segmented.default` should keep the class '`segmented`' (along with the class of the original fit `obj`). Ignored by the segmented methods.

... — optional arguments (to be ignored safely). Notice specific arguments relevant to the original call (via `lm` or `glm` for instance), such as `weights` or `offet`, have to be included in the starting model `obj`

adjX — if `obj` is a `ts`, the segmented variable (if not specified in `seg.Z`) is computed by taking information from the time series (e.g., years starting from 2000, say). If `adjX=TRUE`, the segmented variable is shifted such that its min equals zero. Default is using the unshifted values, but if there are several breakpoints to be estimated , it is strongly suggested to set `adjX=TRUE`.

weights — the weights if `obj` is a vector or a ts object, otherwise the weights should be specified in the starting fit `obj`.

Details

Given a linear regression model usually of class "lm" or "glm" (or even a simple numeric/ts vector), segmented tries to estimate a new regression model having broken-line relationships with the variables specified in `seg.Z`. A segmented (or broken-line) relationship is defined by the slope parameters and the break-points where the linear relation changes. The number of breakpoints of each segmented relationship is fixed via the `psi` argument, where initial values for the break-points must be specified. The model is estimated simultaneously yielding point estimates and relevant approximate standard errors of all the model parameters, including the break-points.

Since version 0.2-9.0 `segmented` implements the bootstrap restarting algorithm described in Wood (2001). The bootstrap restarting is expected to escape the local optima of the objective function when the segmented relationship is flat and the log likelihood can have multiple local optima.

Since version 0.5-0.0 the default method `segmented.default` has been added to estimate segmented relationships in general (besides "lm" and "glm" fits) regression models, such as Cox regression or quantile regression (for a single percentile). The objective function to be minimized is the (minus) value extracted by the `logLik` function or it may be passed on via the `fn.obj` argument in `seg.control`. See example below. While the default method is expected to work with any regression fit (where the usual `coef()`, `update()`, and `logLik()` returns appropriate results), it is not recommended for "lm" or "glm" fits (as `segmented.default` is slower than the specific methods `segmented.lm` and `segmented.glm`), although final results are the same. However the object returned by `segmented.default` is *not* of class "segmented", as currently the segmented methods are not guaranteed to work for 'generic' (i.e., besides "lm" and "glm") regression fits. The user could try each "segmented" method on the returned object by calling it explicitly (e.g. via `plot.segmented()` or `confint.segmented()` wherein the regression coefficients and relevant covariance matrix have to be specified, see `.coef` and `.vcov` in `plot.segmented()`, `confint.segmented()`, `slope()`).

Value

segmented returns an object of class "segmented" which inherits from the class of `obj`, for instance "lm" or "glm".

An object of class "segmented" is a list containing the components of the original object `obj` with additionally the followings:

psi	estimated break-points and relevant (approximate) standard errors
it	number of iterations employed
epsilon	difference in the objective function when the algorithm stops
model	the model frame
psi.history	a list or a vector including the breakpoint estimates at each step
seed	the integer vector containing the seed just before the bootstrap resampling. Returned only if bootstrap restart is employed
..	Other components are not of direct interest of the user

Warning

At convergence, if the estimated breakpoints are too close each other or at the boundaries, the parameter point estimate could be returned, but without finite standard errors. To avoid that, `segmented` revises the final breakpoint estimates to allow that at least `min.nj` are within each interval of the segmented covariate. A warning message is printed if such adjustment is made. See `min.nj` in `seg.control`.

Note

1. The algorithm will start if the `it.max` argument returned by `seg.control` is greater than zero. If `it.max=0` `segmented` will estimate a new linear model with break-point(s) fixed at the values reported in `psi`.Alternatively, it is also possible to set `h=0` in `seg.control()`. In this case, bootstrap restarting is unncessary, then to have breakpoints at `mypsi` type

   ```
   segmented(.., psi=mypsi, control=seg.control(h=0, n.boot=0, it.max=1))
   ```
2. In the returned fit object, 'U.' is put before the name of the segmented variable to mean the difference-in-slopes coefficient.
3. Methods specific to the class `"segmented"` are
   - `print.segmented`
   - `summary.segmented`
   - `print.summary.segmented`
   - `plot.segmented`
   - `lines.segmented`
   - `confint.segmented`
   - `vcov.segmented`
   - `predict.segmented`
   - `points.segmented`
   - `coef.segmented`

   Others are inherited from the class `"lm"` or `"glm"` depending on the class of `obj`.

Author(s)

Vito M. R. Muggeo, <vito.muggeo@unipa.it>

References

Muggeo, V.M.R. (2003) Estimating regression models with unknown break-points. *Statistics in Medicine* **22**, 3055–3071.

Muggeo, V.M.R. (2008) Segmented: an R package to fit regression models with broken-line relationships. *R News* **8/1**, 20–25.

See Also

`segmented.glm` for segmented GLM and `segreg` to fit the models via a formula interface. `segmented.lme` fits random changepoints (segmented mixed) models.

Examples

```
set.seed(12)
xx<-1:100
zz<-runif(100)
yy<-2+1.5*pmax(xx-35,0)-1.5*pmax(xx-70,0)+15*pmax(zz-.5,0)+rnorm(100,0,2)
dati<-data.frame(x=xx,y=yy,z=zz)
out.lm<-lm(y~x,data=dati)

#the simplest example: the starting model includes just 1 covariate
#.. and 1 breakpoint has to be estimated for that
o<-segmented(out.lm) #1 breakpoint for x

#the single segmented variable is not in the starting model, and thus..
#... you need to specify it via seg.Z, but no starting value for psi
o<-segmented(out.lm, seg.Z=~z)
#note the leftmost slope is constrained to be zero (since out.lm does not include z)

#2 segmented variables, 1 breakpoint each (again no need to specify npsi or psi)
o<-segmented(out.lm,seg.Z=~z+x)


#1 segmented variable, but 2 breakpoints: you have to specify starting values (vector) for psi:
o<-segmented(out.lm,seg.Z=~x,psi=c(30,60), control=seg.control(display=FALSE))

#.. or you can specify just the *number* of breakpoints
#o<-segmented(out.lm,seg.Z=~x, npsi=2, control=seg.control(display=FALSE))

slope(o) #the slopes of the segmented relationship


#2 segmented variables: starting values requested via a named list
out.lm<-lm(y~z,data=dati)
o1<-update(o,seg.Z=~x+z,psi=list(x=c(30,60),z=.3))
#..or by specifying just the *number* of breakpoints
#o1<-update(o,seg.Z=~x+z, npsi=c(x=2,z=1))
```

```
#the default method leads to the same results (but it is slower)
#o1<-segmented.default(out.lm,seg.Z=~x+z,psi=list(x=c(30,60),z=.3))
#o1<-segmented.default(out.lm,seg.Z=~x+z,psi=list(x=c(30,60),z=.3),
#    control=seg.control(fn.obj="sum(x$residuals^2)"))


#automatic procedure to estimate breakpoints in the covariate x (starting from K quantiles)
# Hint: increases number of iterations. Notice: bootstrap restart is not allowed!
# However see ?selgmented for a better approach
#o<-segmented.lm(out.lm,seg.Z=~x+z,psi=list(x=NA,z=.3),
#   control=seg.control(fix.npsi=FALSE, n.boot=0, tol=1e-7, it.max = 50, K=5, display=TRUE))

#assess the progress of the breakpoint estimates throughout the iterations
## Not run:
par(mfrow=c(1,2))
draw.history(o, "x")
draw.history(o, "z")

## End(Not run)
#try to increase the number of iterations and re-assess the
#convergence diagnostics


# A simple segmented model with continuous responses and no linear covariates
# No need to fit the starting lm model:
segmented(yy, npsi=2) #NOTE: subsetting the vector works ( segmented(yy[-1],..) )
#only a single segmented covariate is allowed in seg.Z, and if seg.Z is unspecified,
#   the segmented variable is taken as 1:n/n


# An example using the Arima method:
## Not run:
n<-50
idt <-1:n #the time index

mu<-50-idt +1.5*pmax(idt-30,0)
set.seed(6969)
y<-mu+arima.sim(list(ar=.5),n)*3.5

o<-arima(y, c(1,0,0), xreg=idt)
os1<-segmented(o, ~idt, control=seg.control(display=TRUE))

#note using the .coef argument is mandatory!
slope(os1, .coef=os1$coef)
plot(y)
plot(os1, add=TRUE, .coef=os1$coef, col=2)


## End(Not run)

#################################################################
#################################################################
######Four examples using the default method:
```

```
###############################################################
###############################################################


###############################################################
#==> 1. Cox regression with a segmented relationship
###############################################################
## Not run:
library(survival)
data(stanford2)

o<-coxph(Surv(time, status)~age, data=stanford2)
os<-segmented(o, ~age, psi=40) #estimate the breakpoint in the age effect
summary(os) #actually it means summary.coxph(os)
plot(os) #it does not work
plot.segmented(os) #call explicitly plot.segmented() to plot the fitted piecewise lines


###############################################################
# ==> 2. Linear mixed model via the nlme package
###############################################################

dati$g<-gl(10,10) #the cluster 'id' variable
library(nlme)
o<-lme(y~x+z, random=~1|g, data=dati)
os<-segmented.default(o, ~x+z, npsi=list(x=2, z=1))

#summarizing results (note the '.coef' argument)
slope(os, .coef=fixef(os))
plot.segmented(os, "x", .coef=fixef(os), conf.level=.95)
confint.segmented(os, "x", .coef=fixef(os))
dd<-data.frame(x=c(20,50),z=c(.2,.6), g=1:2)
predict.segmented(os, newdata=dd, .coef=fixef(os))


###############################################################
# ==> 3. segmented quantile regression via the quantreg  package
###############################################################

library(quantreg)
data(Mammals)
y<-with(Mammals, log(speed))
x<-with(Mammals, log(weight))
o<-rq(y~x, tau=.9)
os<-segmented.default(o, ~x) #it does NOT work. It cannot compute the vcov matrix..

#Let's define the vcov.rq function.. (I don't know if it is the best option..)
vcov.rq<-function(x,...) {
  V<-summary(x,cov=TRUE,se="nid",...)$cov
  rownames(V)<-colnames(V)<-names(x$coef)
V}

os<-segmented.default(o, ~x) #now it does work
```

```
 plot.segmented(os, res=TRUE, col=2, conf.level=.95)


#################################################################
# ==> 4. segmented regression with the svyglm() (survey  package)
#################################################################

library(survey)
data(api)
dstrat<-svydesign(id=~1,strata=~stype, weights=~pw, data=apistrat, fpc=~fpc)

o<-svyglm(api00~ell, design=dstrat)

#specify as a string the objective function to be minimized. It can be obtained via svyvar()

fn.x<- 'as.numeric(svyvar(resid(x, "pearson"), x$survey.design, na.rm = TRUE))'
os<-segmented.default(o, ~ell, control=seg.control(fn.obj=fn.x, display=TRUE))
slope(os)
plot.segmented(os, res=TRUE, conf.level=.9, shade=TRUE)

## End(Not run)
```

`segmented.lme` *Segmented relationships in linear mixed models*

Description

Fits linear mixed models with a segmented relationship between the response and a numeric covariate. Random effects are allowed in each model parameter, including the breakpoint.

Usage

```
## S3 method for class 'lme'
segmented(obj, seg.Z, psi, npsi = 1, fixed.psi = NULL,
    control = seg.control(), model = TRUE,
    z.psi = ~1, x.diff = ~1, random = NULL,
    random.noG = NULL, start.pd = NULL, psi.link = c("identity", "logit"),
    start = NULL, data, fixed.parms = NULL,...)
```

Arguments

`obj` A 'lme' fit returned by `lme` or simply its call. See example below. This represents the linear mixed model where the segmented relationship is added.

`seg.Z` A one-sided formula indicating the segmented variable, i.e. the quantitative variable having a segmented relationship with the response. In longitudinal studies typically it is the time.

`psi`	An optional starting value for the breakpoint. If missing a starting value is obtained via the nadir estimate of a quadratic fit. When provided it may be a single numeric value or a vector of length equal to the number of clusters (i.e. subjects).
`z.psi`	Optional. A one-sided formula meaning the covariates in the sub-regression model for the changepoint parameter. Default to `~1`.
`x.diff`	Optional. A one-sided formula meaning the covariates in the sub-regression model for the difference-in-slopes parameter. Default to `~1` for no covariate for the difference-in-slope parameter.
`npsi`	Ignored. Currently only `npsi=1` is allowed.
`fixed.psi`	Ignored.
`control`	A list returned by `seg.control`, in particular `display`, `n.boot` for the bootstrap restarting.
`model`	Ignored.
`random`	A list, as the one supplied in `random` of `lme()` including the random effects. Default to `NULL`, meaning that the same random effect structure of the initial lme fit supplied in `obj` should be used. When specified, this list could include the variables '`G0`' and '`U`'. `G0` means random effects in the breakpoints and `U` means random effects in the slope-difference parameter. Assuming `id` is the the cluster variable and `x` the segmented variable, some examples are `random = list(id = pdDiag(~1 + x + U))` #ind. random eff. (changepoint fixed) `random = list(id = pdDiag(~1 + x + U + G0))` #ind. random eff. (in the changepoint too) `random = list(id=pdBlocked(list(pdSymm(~1+x), pdSymm(~U+G0-1))))` #block diagonal
`random.noG`	Ignored.
`start.pd`	An optional starting value for the variances of the random effects. It should be coherent with the specification in `random`.
`psi.link`	The link function used to specify the sub-regression model for the breakpoint psi. The identity (default) assumes $$\psi_i = \eta_i$$ while the logit link is $$\psi_i = (m + M * exp(\eta_i))/(1 + exp(\eta_i))$$ where m and M are the observed minimum and maximum of the segmented variable in `seg.Z`. In each case the 'linear predictor' is $\eta_i = \kappa_0 + z_i^T \kappa_1 + k_i$, where z^T includes the covariates specified in `z.psi` and the k_is are the changepoint random effects included by means of `G0` in the `random` argument.
`start`	An optional list including the *starting values* for the difference-in-slopes parameter, delta0 and delta, and the changepoint parameter, kappa and kappa0. When provided, 'kappa0' overwrites 'psi'. If provided, the components 'delta' and 'kappa' should be *named* vectors with length and names matching length and names in `x.diff` and `z.psi` respectively. The component `delta0` can be a scalar or a vector with length equal to the number of clusters (subjects).

`data` the dataframe where the variables are stored. If missing, the dataframe of the `"lme"` fit `obj` is assumed.

`fixed.parms` An optional *named* vector representing the coefficients *of the changepoint* to be maintained *fixed* during the estimation process. Allowed names are "G0" or any variable (in the dataframe) supposed to affect the location of breakpoints. For instance `fixed.parms=c(G0=.3)` implies a fixed value for the changepoint. Notice if you use the same variable in `fixed.parms` and in `z.psi`, for instance `fixed.parms=c(x2=.3)` and `z.psi=~x2`, a warning is printed and the coefficient "G.x2" is estimated to maximize the log likelihood *given* that fixed value. As an example, suppose the unconstrained estimated coefficient for x2, say, in `z.psi` is 0.5; if in a new call both `fixed.parms=c(x2=.4)` and `z.psi=~x2` are included, the estimate of "G.x2" will be (approximately) 0.1. Essentially, if you really want to fix the parameters in `fixed.parms`, then do not include the same covariates in `z.psi`.

`...` Ignored

Details

The function fits segmented mixed regression models, i.e. segmented models with random effects also in the slope-difference and change-point parameters.

Value

A list of class `segmented.lme` with several components. The most relevant are

`lme.fit` The fitted lme object at convergence

`lme.fit.noG` The fitted lme object at convergence assuming known the breakpoints

`psi.i` The subject/cluster-specific change points (fixed + random). It includes 2 attributes: `attr(,"ni")` for the number of measurements in each 'cluster', and `attr(,"is.break")` a vector of logicals indicating if the breakpoint for each subject i can be reliable (`TRUE`) or not (`FALSE`). Here 'reliable' simply means within the covariate range (for subject i). See also argument `nq`.

`fixed.eta.psi` The fixed-effect linear predictor for the change points regression equation. These values will different among 'clusters' only if at least one covariate has been specified in `z.psi`.

`fixed.eta.delta`
The fixed-effect linear predictor of the slope difference regression equation. These values will different among 'clusters' only if at least one covariate has been specified in `x.diff`.

Warning

The function deals with estimation with a *single* breakpoint only.

Note

Currently only one breakpoint (with or without random effects) can be estimated. If `fit` is the segmented.lme fit, use `VarCorr(fit$lme.fit)` to extract the random effect covariance matrix.

Author(s)

Vito M.R. Muggeo <vito.muggeo@unipa.it>

References

Muggeo V., Atkins D.C., Gallop R.J., Dimidjian S. (2014) Segmented mixed models with random changepoints: a maximum likelihood approach with application to treatment for depression study. Statistical Modelling, 14, 293-313.

Muggeo V. (2016) Segmented mixed models with random changepoints in R. Working paper available on RG. doi: 10.13140/RG.2.1.4180.8402

See Also

`plot.segmented.lme` for the plotting method and `segmented.default` (example 2) for segmented models with no random effects in breakpoints or slope difference.

Examples

```
## Not run:
library(nlme)
data(Cefamandole)
Cefamandole$lTime <-log(Cefamandole$Time)
Cefamandole$lconc <-log(Cefamandole$conc)

o<-lme(lconc ~ lTime, random=~1|Subject, data=Cefamandole)

os<-segmented.lme(o, ~lTime, random=list(Subject=pdDiag(~1+lTime+U+G0)),
  control=seg.control(n.boot=0, display=TRUE))
slope(os)


#####################################################
# covariate effect on the changepoint and slope diff


#let's assume a new subject-specific covariates..
set.seed(69)
Cefamandole$z <- rep(runif(6), rep(14,6))
Cefamandole$group <- gl(2,42,labels=c('a','b'))

#Here 'group' affects the slopes and 'z' affects the changepoint

o1 <-lme(lconc ~ lTime*group, random=~1|Subject, data=Cefamandole)
os1 <- segmented(o1, ~lTime, x.diff=~group, z.psi=~z,
  random=list(Subject=pdDiag(~1+lTime+U+G0)))

slope(os1, by=list(group="a")) #the slope estimates in group="a" (baseline level)
slope(os1, by=list(group="b")) #the slope estimates in group="b"
```

```
####################################################
# A somewhat "complicated" example:
#     i)  strong heterogeneity in the changepoints
#     ii) No changepoint for the Subject #7 (added)

d<-Cefamandole
d$x<- d$lTime
d$x[d$Subject==1]<- d$lTime[d$Subject==1]+3
d$x[d$Subject==5]<- d$lTime[d$Subject==5]+5
d$x[d$Subject==3]<- d$lTime[d$Subject==3]-5
d<-rbind(d, d[71:76,])
d$Subject <- factor(d$Subject, levels=c(levels(d$Subject),"7"))
d$Subject[85:90] <- rep("7",6)

o<-lme(lconc ~ x, random=~1|Subject, data=d)
os2<-segmented.lme(o, ~x, random=list(Subject=pdDiag(~1+x+U+G0)),
  control=seg.control(n.boot=5, display=TRUE))

#plots with common x- and y- scales (to note heterogeneity in the changepoints)
plot(os2, n.plot = c(3,3))
os2$psi.i
attr(os2$psi.i, "is.break") #it is FALSE for Subject #7

#plots with subject-specific scales
plot(os2, n.plot = c(3,3), xscale=-1, yscale = -1)

## End(Not run)
```

---

segreg *Fitting segmented regression*

Description

segreg (stepreg) fits linear or generalized linear segmented (stepmented) regression via a symbolic description of the linear predictor. This is an alternative, introduced since version 2.0-0 (segreg) and 2.1-0 (stepreg), to `segmented.lm` and `segmented.glm`, or (`stepmented.lm` or `stepmented.glm`).

Usage

```
segreg(formula, data, subset, weights, na.action, family = lm, control = seg.control(),
     transf = NULL, contrasts = NULL, model = TRUE, x = FALSE, var.psi = TRUE, ...)

stepreg(formula, data, subset, weights, na.action, family = lm, control = seg.control(),
     transf = NULL, contrasts = NULL, model = TRUE, x = FALSE, var.psi = FALSE, ...)
```

Arguments

`formula` A standard model formula also including one or more 'segmented'/'stepmented' terms via the function `seg`

`data`	The possible dataframe where the variables are stored
`subset`	Possible subset, as in `lm` or `glm`
`weights`	Possible weight vector, see `weights` in `lm` or `glm`
`na.action`	a function which indicates what happen when the data contain NA values. See `na.action` in `lm` or `glm`.
`family`	The family specification, similar to `family` in `glm`. Default to `'lm'` for segmented/stepmented linear models.
`control`	See `seg.control`
`transf`	an optional character string (with "y" as argument) meaning a function to apply to the response variable before fitting
`contrasts`	see `contrasts` in `glm`
`model`	If `TRUE`, the model frame is returned.
`x`	If `TRUE`, the model matrix is returned.
`var.psi`	logical, meaning if the standard errors for the breakpoint estimates should be returned in the object fit. If `FALSE`, the standard errors will be computed by `vcov.segmented` or `summary.segmented`. Setting `var.psi=FALSE` could speed up model estimation for very large datasets. Default to `TRUE` for `segreg` and `FALSE` for `stepreg`.
`...`	Ignored

Details

The function allows to fit segmented/stepmented (G)LM regression models using a formula interface. Results will be the same of those coming from the traditional `segmented.lm` and `segmented.glm` (or `stepmented.lm` or `stepmented.glm`), but there are some additional facilities: i) it is possible to estimate strightforwardly the segmented/stepmented relationships in each level of a categorical variable, see argument `by` in `seg`; ii) it is possible to constrain some slopes of the segmented relationship, see argument `est` or `R` in `seg`.

Value

An object of class "segmented" (or "stepmented") which inherits from the class "lm" or "glm" depending on `family` specification. See `segmented.lm`.

Warning

Currently for fits returned by `segreg`, `confint.segmented` only works if `method="delta"`. Constraints on the mean levels (possibly via argument 'est' of seg) are not yet allowed when calling `stepreg`.

Note

When the formula includes even a single segmented term with constraints (specified via the argument `est` in `seg()`), the relevant coefficients returned do not represent the slope differences as in `segmented.lm` or `segmented.glm`. The values depend on the constraints and are not usually interpretable. Use `slope` the recover the actual slopes of the segmented relationships.

Author(s)

Vito Muggeo

References

Muggeo, V.M.R. (2003) Estimating regression models with unknown break-points. Statistics in Medicine 22, 3055-3071.

See Also

seg, segmented, stepmented

Examples

```
############################
#An example using segreg()
############################

set.seed(10)
x<-1:100
z<-runif(100)
w<-runif(100,-10,-5)
y<-2+1.5*pmax(x-35,0)-1.5*pmax(x-70,0)+10*pmax(z-.5,0)+rnorm(100,0,2)

##the traditional approach
out.lm<-lm(y~x+z+w)
o<-segmented(out.lm, seg.Z=~x+z, psi=list(x=c(30,60),z=.4))

o1<-segreg(y ~ w+seg(x,npsi=2)+seg(z))
all.equal(fitted(o), fitted(o1))

#put some constraints on the slopes
o2<-segreg(y ~ w+seg(x,npsi=2, est=c(0,1,0))+seg(z))
o3<-segreg(y ~ w+seg(x,npsi=2, est=c(0,1,0))+seg(z, est=c(0,1)))

slope(o2)
slope(o3)

##see ?plant for an additional example


############################
#An example using stepreg()
############################

### Two stepmented covariates (with 1 and 2 breakpoints)
n=100
x<-1:n/n
z<-runif(n,2,5)
w<-rnorm(n)
mu<- 2+ 1*(x>.6)-2*(z>3)+3*(z>4)
y<- mu + rnorm(n)*.8
```

```
os <-stepreg(y~seg(x)+seg(z,2)+w) #also includes 'w' as a possible linear term
os
summary(os)
plot(os, "z", col=2:4) #plot the effect of z
```

---

selgmented *Selecting the number of breakpoints in segmented regression*

Description

This function selects (and estimates) the number of breakpoints of the segmented relationship according to the BIC/AIC criterion or sequential hypothesis testing.

Usage

```
selgmented(olm, seg.Z, Kmax=2, type = c("score", "bic", "davies", "aic"),
  alpha = 0.05, control = seg.control(), refit = FALSE, stop.if = 5,
  return.fit = TRUE, bonferroni = FALSE, msg = TRUE, plot.ic = FALSE, th = NULL,
  G = 1, check.dslope = TRUE)
```

Arguments

`olm`	A starting `lm` or `glm` object, or a simple numerical vector meaning the response variable.
`seg.Z`	A one-side formula for the segmented variable. Only one term can be included, and it can be omitted if `olm` is a (g)lm fit including just one covariate.
`Kmax`	The maximum number of breakpoints being tested. If `type='bic'` or `type='aic'`, any integer value can be specified; otherwise at most `Kmax=2` breakpoints can be tested via the Score or Davies statistics.
`type`	Which criterion should be used? Options `"score"` and `"davies"` allow to carry out sequential hypothesis testing with no more than 2 breakpoints (`Kmax=2`). Alternatively, the number of breakpoints can be selected via the BIC (or AIC) with virtually no upper bound for `Kmax`.
`alpha`	The fixed type I error probability when sequential hypothesis testing is carried out (i.e. `type='score'` or `'davies'`). It is also used when `type='bic'` (or `type='aic'`) and `check.dslope=TRUE` to remove the breakpoints based on the slope diffence t-value.
`control`	See `seg.control`.
`refit`	If `TRUE`, the final selected model is re-fitted using arguments in `control`, typically with bootstrap restarting. Set `refit=FALSE` to speed up computation (and possibly accepting near-optimal estimates). It is always `TRUE` if `type='score'` or `type='davies'`.

stop.if
: An integer. If, when trying models with an increasing (when `G=1`) or decreasing (when `G>1`) number of breakpoints, `stop.if` consecutive fits exhibit higher AIC/BIC values, the search is interrupted. Set a large number, larger then `Kmax` say, if you want to assess the fits for all breakpoints `0, 1, 2, ..., Kmax`. Ignored if `type='score'` or `type='davies'`.

return.fit
: If `TRUE`, the fitted model (with the number of breakpoints selected according to `type`) is returned.

bonferroni
: If `TRUE`, the Bonferroni correction is employed, i.e. `alpha/Kmax` (rather than `alpha`) is always taken as threshold value to reject or not. If `FALSE`, `alpha` is used in the second level of hypothesis testing. It is also effective when `type="bic"` (or `'aic'`) and `check.dslope=TRUE`, see Details.

msg
: If `FALSE` the final fit is returned silently with the selected number of breakpoints, otherwise the messages about the selection procedure (i.e. the BIC values), and possible warnings are also printed.

plot.ic
: If `TRUE` the information criterion values with respect to the number of breakpoints are plotted. Ignored if `type='score'` or `type='davies'` or `G>1`. Note that if `check.dslope=TRUE`, the final number of breakpoints could differ from the one selected by the BIC/AIC leading to an inconsistent plot of the information criterion, see Note below.

th
: When a large number of breakpoints is being tested, it could happen that 2 estimated breakpoints are too close each other, and only one can be retained. Thus if the difference between two breakpoints is less or equal to `th`, one (the first) breakpoint is removed. Of course, `th` depends on the `x` scale: Integers, like 5 or 10, are appropriate if the covariate is the observation index. Default (`NULL`) means `th=diff(range(x))/100`. Set `th=0` if you are willing to consider even breakpoints very clode each other. Ignored if `type='score'` or `type='davies'`.

G
: Number of sub-intervals to consider to search for the breakpoints when `type='bic'` or `'aic'`. See Details.

check.dslope
: Logical. Effective only if `type='bic'` or `'aic'`. After the optimal number of breakpoints has been selected (via AIC/BIC), should the t values of the slope differences be checked? If `TRUE`, the breakpoints corresponding to slope differences with a 'low' t values will be removed. Note the model is re-fitted at each removal and a new check is performed. Simulation evidence suggests that such strategy leads to better results. See Details.

Details

The function uses properly the functions `segmented`, `pscore.test` or `davies.test` to select the 'optimal' number of breakpoints `0,1,...,Kmax`. If `type='bic'` or `'aic'`, the procedure stops if the last `stop.if` fits have increasing values of the information criterion. Moreover, a breakpoint is removed if too close to other, actually if the difference between two consecutive estimates is less then `th`. Finally, if `check.dslope=TRUE`, breakpoints whose corresponding slope difference estimate is 'small' (i.e. p-value larger then `alpha` or `alpha/Kmax`) are also removed.

When $G > 1$ the dataset is split into G groups, and the search is carried out separately within each group. This approach is fruitful when there are many breakpoints not evenly spaced in the covariate range and/or concentrated in some sub-intervals. `G=3` or `4` is recommended based on simulation evidence.

Note `Kmax` is always tacitely reduced in order to have at least 1 residual df in the model with `Kmax` changepoints. Namely, if $n = 20$, the maximal segmented model has `2*(Kmax + 1)` parameters, and therefore the largest `Kmax` allowed is 8.

When `type='score'` or `'davies'`, the function also returns the 'overall p-value' coming from combing the single p-values using the Fisher method. The pooled p-value, however, does not affect the final result which depends on the single p-values only.

Value

The returned object depends on argument `return.fit`. If `FALSE`, the returned object is a list with some information on the compared models (i.e. the BIC values), otherwise a classical `'segmented'` object (see `segmented` for details) with the component `selection.psi` including the A/BIC values and
- if `refit=TRUE`, `psi.no.refit` that represents the breakpoint values before the last fit (with boot restarting)
- if `G>1`, `cutvalues` including the cutoffs values used to split the data.

Note

If `check.dslope=TRUE`, there is no guarantee that the final model has the lowest AIC/BIC. Namely the model with the best A/BIC could have 'non-significant' slope differences which will be removed (with the corresponding breakpoints) by the final model. Hence the possible plot (obtained via `plot.ic=TRUE`) could be misleading. See Example 1 below.

Author(s)

Vito M. R. Muggeo

References

Muggeo V (2020) Selecting number of breakpoints in segmented regression: implementation in the R package segmented https://www.researchgate.net/publication/343737604

See Also

`segmented`, `pscore.test`, `davies.test`

Examples

```
set.seed(12)
xx<-1:100
zz<-runif(100)
yy<-2+1.5*pmax(xx-35,0)-1.5*pmax(xx-70,0)+15*pmax(zz-.5,0)+rnorm(100,0,2)
dati<-data.frame(x=xx,y=yy,z=zz)
out.lm<-lm(y~x,data=dati)

os <-selgmented(out.lm) #selection (Kmax=2) via the Score test (default)

os <-selgmented(out.lm, type="bic", Kmax=3) #BIC-based selection
```

```
## Not run:
########################################
#Example 1: selecting a large number of breakpoints

b <- c(-1,rep(c(1.5,-1.5),l=15))
psi <- seq(.1,.9,l=15)
n <- 2000
x <- 1:n/n
X <- cbind(x, outer(x,psi,function(x,y)pmax(x-y,0)))
mu <- drop(tcrossprod(X,t(b)))
set.seed(113)
y<- mu + rnorm(n)*.022
par(mfrow=c(1,2))

#select number of breakpoints via the BIC (and plot it)
o<-selgmented(y, Kmax=20, type='bic', plot.ic=TRUE, check.dslope = FALSE)
plot(o, res=TRUE, col=2, lwd=3)
points(o)
# select via the BIC + check on the slope differences (default)
o1 <-selgmented(y, Kmax=20, type='bic', plot.ic=TRUE) #check.dslope = TRUE by default
#note the plot of BIC is misleading.. But the number of psi is correct
plot(o1, add=TRUE, col=3)
points(o1, col=3, pch=3)

##################################################
#Example 2: a large number of breakpoints not evenly spaced.

b <- c(-1,rep(c(2,-2),l=10))
psi <- seq(.5,.9,l=10)
n <- 2000
x <- 1:n/n
X <- cbind(x, outer(x,psi,function(x,y)pmax(x-y,0)))
mu <- drop(tcrossprod(X,t(b)))
y<- mu + rnorm(n)*.02

#run selgmented with G>1. G=3 or 4 recommended.
#note G=1 does not return the right number of breaks
o1 <-selgmented(y, type="bic", Kmax=20, G=4)

## End(Not run)
```

---

slope *Slope estimates from segmented/stepmented relationships*

Description

Computes the slopes of each 'segmented' (or even 'stepmented') relationship in the fitted model.

Usage

```
slope(ogg, parm, conf.level = 0.95, rev.sgn=FALSE,
    APC=FALSE, .vcov=NULL, .coef=NULL,
  use.t=NULL, by=NULL, interc=TRUE, ..., digits = max(4, getOption("digits") - 2))
```

Arguments

`ogg`	an object of class "segmented", returned by any `segmented` method or a list of two segmented fits to compare the estimates of corresponding slopes.
`parm`	the segmented variable whose slopes have to be computed. If missing all the segmented variables are considered.
`conf.level`	the confidence level required.
`rev.sgn`	vector of logicals. The length should be equal to the length of `parm`, but it is recycled otherwise. When `TRUE` it is assumed that the current `parm` is 'minus' the actual segmented variable, therefore the sign is reversed before printing. This is useful when a null-constraint has been set on the last slope.
`APC`	logical. If `APC=TRUE` the 'annual percent changes', i.e. $100 \times (\exp(\beta) - 1)$, are computed for each interval (β is the slope). Only point estimates and confidence intervals are returned.
`.vcov`	The *full* covariance matrix of estimates. If unspecified (i.e. `NULL`), the covariance matrix is computed internally by `vcov(ogg)`.
`.coef`	The regression parameter estimates. If unspecified (i.e. `NULL`), it is computed internally by `coef(ogg)`.
`use.t`	Which quantiles should be used to compute the confidence intervals? If `NULL` (default) the t distribution is used only for objects obtained by `segmented.lm`.
`by`	Only for `segmented.lme` objects. It is a named list indicating covariate names and corresponding values affecting the fitted segmented relationship. For instance, `by=list(group="2",z2=.2)`, provided that the model has been fitted by specifying `group` and `z2` in `x.diff` (or as interaction with the segmented variable). Note that if the provided variables or values are irrelevant for changing the slopes, a warning message is printed.
`interc`	logical, only for `'stepmented'` fits. If `TRUE`, the mean levels also account for the intercept; otherwise the first level is assumed to be zero.
`...`	Further arguments to be passed on to `vcov.segmented`, such as `var.diff` and `is`. See Details in `vcov.segmented` and `summary.segmented`.
`digits`	controls number of digits in the returned output.

Details

To fit broken-line relationships, `segmented` uses a parameterization whose coefficients are not the slopes. Therefore given an object `"segmented"`, `slope` computes point estimates, standard errors, t-values and confidence intervals of the slopes of each segmented relationship in the fitted model.

Value

`slope` returns a list of matrices. Each matrix represents a segmented relationship and its number of rows equal to the number of segments, while five columns summarize the results.

Note

The returned summary is based on limiting Gaussian distribution for the model parameters involved in the computations. Sometimes, even with large sample sizes such approximations are questionable (e.g., with small difference-in-slope parameters) and the results returned by `slope` might be unreliable. Therefore is responsability of the user to gauge the applicability of such asymptotic approximations. Anyway, the t values may be not assumed for testing purposes and they should be used just as guidelines to assess the estimate uncertainty.

Author(s)

Vito M. R. Muggeo, <vito.muggeo@unipa.it>

References

Muggeo, V.M.R. (2003) Estimating regression models with unknown break-points. *Statistics in Medicine* **22**, 3055–3071.

See Also

See also `davies.test` and `pscore.test` to test for a nonzero difference-in-slope parameter.

Examples

```
set.seed(16)
x<-1:100
y<-2+1.5*pmax(x-35,0)-1.5*pmax(x-70,0)+rnorm(100,0,3)
out<-glm(y~1)
out.seg<-segmented(out,seg.Z=~x,psi=list(x=c(20,80)))
## the slopes of the three segments....
slope(out.seg)
rm(x,y,out,out.seg)
#
## an heteroscedastic example..
set.seed(123)
n<-100
x<-1:n/n
y<- -x+1.5*pmax(x-.5,0)+rnorm(n,0,1)*ifelse(x<=.5,.4,.1)
o<-lm(y~x)
oseg<-segmented(o,seg.Z=~x,psi=.6)
slope(oseg)
slope(oseg,var.diff=TRUE) #better CI
```

`stagnant` *Stagnant band height data*

Description

The `stagnant` data frame has 28 rows and 2 columns.

Usage

```
data(stagnant)
```

Format

A data frame with 28 observations on the following 2 variables.

- `x` log of flow rate in g/cm sec.
- `y` log of band height in cm

Details

Bacon and Watts report that such data were obtained by R.A. Cook during his investigation of the behaviour of stagnant surface layer height in a controlled flow of water.

Source

Bacon D.W., Watts D.G. (1971) Estimating the transistion between two intersecting straight lines. *Biometrika* **58**: 525 – 534.

Originally from the PhD thesis by R.A. Cook

Examples

```
data(stagnant)
## plot(stagnant)
```

---

step.lm.fit   *Fitter Functions for stepmented Linear Models*

Description

`step.lm.fit` is called by `stepmented.lm` to fit stepmented linear (gaussian) models. Likewise, `step.glm.fit` is called by `stepmented.glm` to fit generalized stepmented linear models.

The `step.*.fit.boot` functions are employed to perform bootstrap restarting. These functions should usually not be used directly by the user.

Usage

```
step.lm.fit(y, x.lin, Xtrue, PSI, ww, offs, opz, return.all.sol=FALSE)

step.lm.fit.boot(y, XREG, Z, PSI, w, offs, opz, n.boot=10, size.boot=NULL,
        jt=FALSE, nonParam=TRUE, random=FALSE, break.boot=n.boot)

step.glm.fit(y, x.lin, Xtrue, PSI, ww, offs, opz, return.all.sol=FALSE)

step.glm.fit.boot(y, XREG, Z, PSI, w, offs, opz, n.boot=10, size.boot=NULL,
        jt=FALSE, nonParam=TRUE, random=FALSE, break.boot=n.boot)
```

Arguments

`y`	vector of observations of length `n`.
`x.lin, XREG`	design matrix for standard linear terms.
`Xtrue, Z`	appropriate matrix including the stepmented variables whose breakpoints have to be estimated.
`PSI`	appropriate matrix including the starting values of the breakpoints to be estimated.
`ww,w`	possibe weights vector.
`offs`	possibe offset vector.
`opz`	a list including information useful for model fitting.
`n.boot`	the number of bootstrap samples employed in the bootstrap restart algorithm.
`break.boot`	Integer, less than `n.boot`. If `break.boot` consecutive bootstrap samples lead to the same objective function, the algorithm stops without performing all `n.boot` 'trials'. This can save computational time considerably.
`size.boot`	the size of the bootstrap resamples. If `NULL` (default), it is taken equal to the sample size. values smaller than the sample size are expected to increase perturbation in the bootstrap resamples.
`jt`	logical. If `TRUE` the values of the stepmented variable(s) are jittered before fitting the model to the bootstrap resamples.
`nonParam`	if `TRUE` nonparametric bootstrap (i.e. case-resampling) is used, otherwise residual-based.
`random`	if `TRUE`, when the algorithm fails to obtain a solution, random values are used as candidate values.
`return.all.sol`	if `TRUE`, when the algorithm fails to obtain a solution, the values visited by the algorithm with corresponding deviances are returned.

Details

The functions call iteratively `lm.wfit` (or `glm.fit`) with proper design matrix depending on `XREG`, `Z` and `PSI`. `step.lm.fit.boot` (and `step.glm.fit.boot`) implements the bootstrap restarting idea discussed in Wood (2001).

Value

A list of fit information.

Note

These functions should usually not be used directly by the user.

Author(s)

Vito Muggeo

References

Wood, S. N. (2001) Minimizing model fitting objectives that contain spurious local minima by bootstrap restarting. *Biometrics* **57**, 240–244.

See Also

`stepmented.lm` or `stepmented.glm`

Examples

```
##See ?stepmented
```

`stepmented` *stepmented relationships in regression models*

Description

Fits regression models with stepmented (i.e. piecewise-constant) relationships between the response and one or more explanatory variables. Break-point estimates are provided.

Usage

```
stepmented(obj, seg.Z, psi, npsi, fixed.psi=NULL, control = seg.control(),
   keep.class=FALSE, var.psi=FALSE, ...)

## S3 method for class 'lm'
stepmented(obj, seg.Z, psi, npsi, fixed.psi=NULL, control = seg.control(),
   keep.class=FALSE, var.psi=FALSE, ...)

## S3 method for class 'glm'
stepmented(obj, seg.Z, psi, npsi, fixed.psi=NULL, control = seg.control(),
    keep.class=FALSE, var.psi=FALSE, ...)

## S3 method for class 'numeric'
stepmented(obj, seg.Z, psi, npsi, fixed.psi=NULL, control = seg.control(),
    keep.class=FALSE, var.psi=FALSE, ...,
    pertV=0, centerX=FALSE, adjX=NULL, weights=NULL)

## S3 method for class 'ts'
```

```
stepmented(obj, seg.Z, psi, npsi, fixed.psi=NULL, control = seg.control(),
    keep.class=FALSE, var.psi=FALSE, ...,
    pertV=0, centerX=FALSE, adjX=NULL)
```

Arguments

`obj` A standard 'linear' regression model of class "lm" or "glm". Alternatively, a simple "ts" object or a simple data vector may be supplied.

`seg.Z` the stepmented variables(s), i.e. the numeric covariate(s) understood to have a piecewise-constant relationship with response. It is a formula with no response variable, such as `seg.Z=~x` or `seg.Z=~x1+x2`. Currently, formulas involving functions, such as `seg.Z=~log(x1)`, or selection operators, such as `seg.Z=~d[,"x1"]` or `seg.Z=~d$x1`, are *not* allowed. Also, variable names formed by `U` or `V` only (with or without numbers) are not permitted. If missing, the index variable `id=1,2,..,n` is used. For `stepmented.ts`, `seg.Z` is usually unspecified as the (time) covariate is obtained by the `ts` object itself.

`psi` starting values for the breakpoints to be estimated. If there is a single stepmented variable specified in `seg.Z`, `psi` can be a numeric vector, and it can be missing when 1 breakpoint has to be estimated (and the median of the stepmented variable is used as a starting value). If `seg.Z` includes several covariates, `psi` has to be specified as a *named* list of vectors whose names have to match the variables in the `seg.Z` argument. Each vector of such list includes starting values for the break-point(s) for the corresponding variable in `seg.Z`. A `NA` value means that '`K`' quantiles (or equally spaced values) are used as starting values; `K` is fixed via the `seg.control` auxiliary function.

`npsi` A named vector or list meaning the *number* (and not locations) of breakpoints to be estimated. The starting values will be internally computed via the quantiles or equally spaced values, as specified in argument `quant` in `seg.control`. `npsi` can be missing and `npsi=1` is assumed for all variables specified in `seg.Z`. If `psi` is provided, `npsi` is ignored.

`fixed.psi` An optional named list including the breakpoint values to be kept fixed during the estimation procedure. The names should be a subset of (or even the same) variables specified in `seg.Z`. If there is a single variable in `seg.Z`, a simple numeric vector can be specified. Note that, in addition to the values specified here, `stepmented` will estimate additional breakpoints. To keep fixed all breakpoints (to be specified in `psi`) use `it.max=0` in `seg.control`

`control` a list of parameters for controlling the fitting process. See the documentation for `seg.control` for details.

`keep.class` logical value indicating if the final fit returned by `stepmented.default` should keep the class '`stepmented`' (along with the class of the original fit `obj`). Ignored by the stepmented methods.

`...` optional arguments (to be ignored safely). Notice specific arguments relevant to the original call (via `lm` or `glm` for instance), such as `weights` or `offet`, have to

	be included in the starting model `obj`.
`pertV`	Only for `stepmented.ts` and `stepmented.numeric`.
`centerX`	Only for `stepmented.ts` and `stepmented.numeric`. If `TRUE`, the covariate is centered before fitting.
`adjX`	Only for `stepmented.ts` and `stepmented.numeric`. If the response vector leads to covariate with large values (such as years for ts objects), `adjX=TRUE` will shift the covariate to have a zero origin. Default is `NULL` which means `TRUE` if the minimum of covariate is 1000 or larger.
`var.psi`	logical. If `TRUE`, the estimate covariance matrix is also computed via `vcov.stepmented`, thus the breakpoint standard errors are also included in the `psi` component of the returned object. Default is `FALSE`, as computing the estimate covariance matrix is somewhat time-consuming when the sample size is large.
`weights`	possible weights to include in the estimation process (only for `stepmented.numeric`).

Details

Given a linear regression model (usually of class "lm" or "glm"), stepmented tries to estimate a new regression model having piecewise-constant (i.e. step-function like) relationships with the variables specified in `seg.Z`. A *stepmented* relationship is defined by the mean level parameters and the break-points where the mean level changes. The number of breakpoints of each stepmented relationship depends on the `psi` argument, where initial values for the break-points must be specified. The model is estimated simultaneously yielding point estimates and relevant approximate standard errors of all the model parameters, including the break-points.

`stepmented` implements the algorithm described in Fasola et al. (2018) along with bootstrap restarting (Wood, 2001) to escape local optima. The procedure turns out to be particularly appealing and probably efficient when there are two or more covariates exhibiting different change points to be estimated.

Value

The returned object is of class "stepmented" which inherits from the class "lm" or "glm" depending on the class of `obj`. When `only.mean=FALSE`, it is a list having two 'stepmented' fits (for the mean and for the dispersion submodels).

An object of class "stepmented" is a list containing the components of the original object `obj` with additionally the followings:

`psi`	estimated break-points and relevant (approximate) standard errors (on the continuum)
`psi.rounded`	the rounded estimated break-points (see Note, below)
`it`	number of iterations employed
`epsilon`	difference in the objective function when the algorithm stops
`model`	the model frame
`psi.history`	a list or a vector including the breakpoint estimates at each step
`seed`	the integer vector containing the seed just before the bootstrap resampling. Returned only if bootstrap restart is employed
`..`	Other components are not of direct interest of the user

Note

The component psi.rounded of the fit object includes the rounded changepoint values which are usually taken as the final estimates. More specifically, each column of psi.rounded represents a changepoint and the corresponding rows are the range of the 'optimal' interval. The first row, i.e. the lower bound of the interval, is taken as point estimate. print.stepmented, print.summary.stepmented, and confint.stepmented return the rounded (lower) value of the interval.

Also:

1. The algorithm will start if the it.max argument returned by seg.control is greater than zero. If it.max=0 stepmented will estimate a new linear model with break-point(s) fixed at the starting values reported in psi. Alternatively, it is also possible to set h=0 in seg.control(). In this case, bootstrap restarting is unncessary, then to have changepoints at mypsi type

   ```
   stepmented(.., psi=mypsi, control=seg.control(h=0, n.boot=0, it.max=1))
   ```

2. In the returned fit object, 'U.' is put before the name of the stepmented variable to indicate the difference in the mean levels. slope can be used to compute the actual mean levels corresponding to the different intervals.
3. Currently methods specific to the class "stepmented" are
   - print.stepmented
   - summary.stepmented
   - print.summary.stepmented
   - plot.stepmented
   - confint.stepmented
   - vcov.stepmented
   - lines.stepmented

   Others are inherited from the class "lm" or "glm" depending on the class of obj.

Author(s)

Vito M. R. Muggeo, <vito.muggeo@unipa.it> (based on original code by Salvatore Fasola)

References

Fasola S, Muggeo VMR, Kuchenhoff H (2018) A heuristic, iterative algorithm for change-point detection in abrupt change models, *Computational Statistics* **33**, 997–1015

See Also

segmented for segmented regression, lm, glm

Examples

```
n=20
x<-1:n/n
mu<- 2+ 1*(x>.6)
```

```
y<- mu + rnorm(n)*.8

#fitting via regression model
os <-stepmented(lm(y~1),~x)

y<-ts(y)
os1<- stepmented(y)  #the 'ts' method
os2<- stepmented(y, npsi=2)
#plot(y)
#plot(os1, add=TRUE)
#plot(os2, add=TRUE, col=3:5)


### Example with (poisson) GLM
y<- rpois(n,exp(mu))
o<-stepmented(glm(y~1,family=poisson))
plot(o, res=TRUE)

## Not run:

## Example using the (well-known) Nile dataset
data(Nile)
plot(Nile)
os<- stepmented(Nile)
plot(os, add=TRUE)


### Example with (binary) GLM (example from the package stepR)
set.seed(1234)
y <- rbinom(200, 1, rep(c(0.1, 0.7, 0.3, 0.9), each=50))
o<-stepmented(glm(y~1,family=binomial), npsi=3)
plot(o, res=TRUE)

### Two stepmented covariates (with 1 and 2 breakpoints); z has also an additional linear effect
n=100
x<-1:n/n
z<-runif(n,2,5)
mu<- 2+ 1*(x>.6)-2*(z>3)+3*(z>4)+z
y<- mu + rnorm(n)*.8

os <-stepmented(lm(y~z),~x+z, npsi=c(x=1,z=2))
os
summary(os)

## see ?plot.stepmented

## End(Not run)
```

summary.segmented *Summarizing model fits for segmented regression*

Description

summary method for class `segmented`.

Usage

```
## S3 method for class 'segmented'
summary(object, short = FALSE, var.diff = FALSE, p.df="p", .vcov=NULL, ...)

## S3 method for class 'summary.segmented'
print(x, short=x$short, var.diff=x$var.diff,
    digits = max(3, getOption("digits") - 3),
    signif.stars = getOption("show.signif.stars"),...)
```

Arguments

`object`	Object of class "segmented".
`short`	logical indicating if the 'short' summary should be printed.
`var.diff`	logical indicating if different error variances should be computed in each interval of the segmented variable, see Details. If `.vcov` is provided, `var.diff` is set to `FALSE`.
`p.df`	A character as a function of `'p'` (number of parameters) and `'K'` (number of groups or segments) affecting computations of the group-specific variance (and the standard errors) if `var.diff=TRUE`, see Details.
`.vcov`	Optional. The full covariance matrix for the parameter estimates. If provided, standard errors are computed (and displayed) according to this matrix.
`x`	a `summary.segmented` object produced by `summary.segmented()`.
`digits`	controls number of digits printed in output.
`signif.stars`	logical, should stars be printed on summary tables of coefficients?
`...`	further arguments.

Details

If `short=TRUE` only coefficients of the segmented relationships are printed. If `var.diff=TRUE` and there is only one segmented variable, different error variances are computed in the intervals defined by the estimated breakpoints of the segmented variable. For the jth interval with n_j observations, the error variance is estimated via $RSS_j/(n_j - p)$, where RSS_j is the residual sum of squares in interval j, and p is the number of model parameters. This number to be subtracted from n_j can be changed via argument `p.df`. For instance `p.df="0"` uses $RSS_j/(n_j)$, and `p.df="p/K"` leads to $RSS_j/(n_j - p/K)$, where K is the number of groups (segments), and p/K can be interpreted as the average number of model parameter in that group.

Note `var.diff=TRUE` only affects the estimates covariance matrix. It does *not* affect the parameter estimates, neither the log likelihood and relevant measures, such as AIC or BIC. In other words, `var.diff=TRUE` just provides 'alternative' standard errors, probably appropriate when the error variances are different before/after the estimated breakpoints. Also $p - values$ are computed using the t-distribution with 'naive' degrees of freedom (as reported in `object$df.residual`).

If `var.diff=TRUE` the variance-covariance matrix of the estimates is computed via the sandwich formula,

$$(X^T X)^{-1} X^T V X (X^T X)^{-1}$$

where V is the diagonal matrix including the different group-specific error variance estimates. Standard errors are the square root of the main diagonal of this matrix.

Value

A list (similar to one returned by `segmented.lm` or `segmented.glm`) with additional components:

- `psi` estimated break-points and relevant (approximate) standard errors
- `Ttable` estimates and standard errors of the model parameters. This is similar to the matrix `coefficients` returned by `summary.lm` or `summary.glm`, but without the rows corresponding to the breakpoints. Even the p-values relevant to the difference-in-slope parameters have been replaced by NA, since they are meaningless in this case, see `davies.test`.
- `gap` estimated coefficients, standard errors and t-values for the 'gap' variables
- `cov.var.diff` if `var.diff=TRUE`, the covaraince matrix accounting for heteroscedastic errors.
- `sigma.new` if `var.diff=TRUE`, the square root of the estimated error variances in each interval.
- `df.new` if `var.diff=TRUE`, the residual degrees of freedom in each interval.

Author(s)

Vito M.R. Muggeo

See Also

`print.segmented`, `davies.test`

Examples

```
##continues example from segmented()
# summary(segmented.model,short=TRUE)

## an heteroscedastic example..
# set.seed(123)
# n<-100
# x<-1:n/n
# y<- -x+1.5*pmax(x-.5,0)+rnorm(n,0,1)*ifelse(x<=.5,.4,.1)
# o<-lm(y~x)
# oseg<-segmented(o,seg.Z=~x,psi=.6)
# summary(oseg,var.diff=TRUE)$sigma.new
```

summary.segmented.lme *Summarizing model fits for segmented mixed-effects regression*

Description

summary method for class `segmented.lme`.

Usage

```
## S3 method for class 'segmented.lme'
summary(object, .vcov=NULL, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`object`	Object of class "segmented.lme".
`.vcov`	Optional. The full covariance matrix for the parameter estimates. If provided, standard errors are computed (and displayed) according to this matrix.
`digits`	controls number of digits printed in output.
`...`	further arguments.

Details

The function summarizes and prints the most relevant information on the segmented mixed fit. The output is similar to that returned by `print.summary.lme`

Value

A list (similar to one returned by `segmented.lm`) with estimates of the variance components, and point estimates, standard errors, DF, t-value and p-value for the fixed effects. p-values for the variables `U` and `G0` are omitted as pointless.

Author(s)

Vito M.R. Muggeo

See Also

`print.segmented.lme`

Examples

```
##continues example from segmented.lme()
# summary(os)
```

`summary.stepmented` *Summarizing model fits for stepmented regression*

Description

summary/print method for class `stepmented`.

Usage

```
## S3 method for class 'stepmented'
summary(object, short = FALSE, var.diff = FALSE, p.df="p", .vcov=NULL, ...)

## S3 method for class 'summary.stepmented'
print(x, short=x$short, var.diff=x$var.diff,
    digits = max(3, getOption("digits") - 3),
    signif.stars = getOption("show.signif.stars"),...)

## S3 method for class 'stepmented'
print(x, digits = max(3, getOption("digits") - 3),
    ...)
```

Arguments

`object, x`	Object of class "stepmented" or a `summary.stepmented` object produced by `summary.stepmented()`.
`short`	logical indicating if the 'short' summary should be printed.
`var.diff`	logical indicating if different error variances should be computed in each interval of the stepmented variable, see Details. If `.vcov` is provided, `var.diff` is set to `FALSE`.
`p.df`	A character as a function of `'p'` (number of parameters) and `'K'` (number of groups or segments) affecting computations of the group-specific variance (and the standard errors) if `var.diff=TRUE`, see Details.
`.vcov`	Optional. The full covariance matrix for the parameter estimates. If provided, standard errors are computed (and displayed) according to this matrix.
`digits`	controls number of digits printed in output.
`signif.stars`	logical, should stars be printed on summary tables of coefficients?
`...`	further arguments, notably `type` to be passed to `vcov.stepmented` to compute the standard errors. See `vcov.stepmented`.

Details

If `short=TRUE` only coefficients of the stepmented relationships are printed. If `var.diff=TRUE` and there is only one stepmented variable, different error variances are computed in the intervals defined by the estimated breakpoints of the stepmented variable. For the jth interval with n_j observations, the error variance is estimated via $RSS_j/(n_j - p)$, where RSS_j is the residual sum of squares in interval j, and p is the number of model parameters. This number to be subtracted from n_j can be changed via argument `p.df`. For instance `p.df="0"` uses $RSS_j/(n_j)$, and `p.df="p/K"` leads to $RSS_j/(n_j - p/K)$, where K is the number of groups (segments), and p/K can be interpreted as the average number of model parameter in that group.

Note `var.diff=TRUE` only affects the estimates covariance matrix. It does *not* affect the parameter estimates, neither the log likelihood and relevant measures, such as AIC or BIC. In other words, `var.diff=TRUE` just provides 'alternative' standard errors, probably appropriate when the error variances are different before/after the estimated breakpoints. Also $p - values$ are computed using the t-distribution with 'naive' degrees of freedom (as reported in `object$df.residual`).

If `var.diff=TRUE` the variance-covariance matrix of the estimates is computed via the sandwich formula,

$$(X^T X)^{-1} X^T V X (X^T X)^{-1}$$

where V is the diagonal matrix including the different group-specific error variance estimates. Standard errors are the square root of the main diagonal of this matrix.

Value

A list (similar to one returned by `stepmented.lm` or `stepmented.glm`) with additional components:

- `psi` — estimated break-points and relevant (approximate) standard errors
- `Ttable` — estimates and standard errors of the model parameters. This is similar to the matrix `coefficients` returned by `summary.lm` or `summary.glm`, but without the rows corresponding to the breakpoints. Even the p-values relevant to the difference-in-slope parameters have been replaced by NA, since they are meaningless in this case, see `davies.test`.
- `cov.var.diff` — if `var.diff=TRUE`, the covaraince matrix accounting for heteroscedastic errors.
- `sigma.new` — if `var.diff=TRUE`, the square root of the estimated error variances in each interval.
- `df.new` — if `var.diff=TRUE`, the residual degrees of freedom in each interval.

Warning

If `type` is not specified in `...` (which means `type="standard"`), no standard error will be computed (and returned) for the jumpoint.

Author(s)

Vito M.R. Muggeo

See Also

`pscore.test`

Examples

```
##continues example from stepmented()
# summary(stepmented.model,short=TRUE)

## an heteroscedastic example..
# set.seed(123)
# n<-100
# x<-1:n/n
# y<- -x+1.5*pmax(x-.5,0)+rnorm(n,0,1)*ifelse(x<=.5,.4,.1)
# o<-lm(y~x)
# oseg<-stepmented(o,seg.Z=~x,psi=.6)
# summary(oseg,var.diff=TRUE)$sigma.new
```

---

vcov.segmented — *Variance-Covariance Matrix for a Fitted Segmented Model*

Description

Returns the variance-covariance matrix of the parameters (including breakpoints) of a fitted segmented model object.

Usage

```
## S3 method for class 'segmented'
vcov(object, var.diff = FALSE, is = FALSE, ...)
```

Arguments

`object`	a fitted model object of class "segmented", returned by any `segmented` method or `segreg`.
`var.diff`	logical. If `var.diff=TRUE` and there is a single segmented variable, the covariance matrix is computed using a sandwich-type formula. See Details in `summary.segmented`.
`is`	logical. If `TRUE`, the *asymptotic* covariance matrix based on the idea of induced smoothing is returned. If `is=TRUE`, `var.diff=FALSE` is set. `is=TRUE` only works with segmented (g)lm fits.
`...`	additional arguments.

Details

The returned covariance matrix is based on an approximation of the nonlinear segmented term. Therefore covariances corresponding to breakpoints are reliable only in large samples and/or clear cut segmented relationships. If `is=TRUE`, the returned covariance matrix depends on the design matrix having the term $I(x > \psi)$ replaced by its smooth counterpart.

Value

The full matrix of the estimated covariances between the parameter estimates, including the breakpoints.

Note

`var.diff=TRUE` works when there is a single segmented variable.

Author(s)

Vito M. R. Muggeo, `<vito.muggeo@unipa.it>`

See Also

`summary.segmented`

Examples

```
##continues example from summary.segmented()
# vcov(oseg)
# vcov(oseg, var.diff=TRUE)
# vcov(oseg, is=TRUE)
```

---

vcov.segmented.lme *Variance-Covariance Matrix for a Fitted Segmented Mixed Model*

Description

Returns the variance-covariance matrix of the parameters (including breakpoints) of a fitted segmented mixed model object.

Usage

```
## S3 method for class 'segmented.lme'
vcov(object, B=0, ret.b=FALSE, ...)
```

Arguments

`object`	a fitted model object of class "segmented.lme", returned by `segmented.lme` method.
`B`	number of bootstrap replicates, if a bootstrap-based covariance matrix is requested.
`ret.b`	logical. If `FALSE` the full covariance matrix (for the fixed effect estimates) based on `B` case-resampling bootstrap samples is returned; otherwise a list with information on the bootstrap sampling distributions.
`...`	optional arguments, i.e. `seed` and `it.max.b`, used when implementing the bootstrap.

Details

The returned covariance matrix is based on an approximation of the nonlinear segmented term. Therefore covariances corresponding to breakpoints are reliable only in large samples and/or clear cut segmented relationships. If `B>0` is set, case resampling bootstrap (on the outermost nesting level) is carried out. Moreover, if `ret.b=TRUE`, the bootstrap distributions are returned, rather than the covariance matrix.

Value

The full matrix of the estimated covariances of the fixed effects estimates, including the breakpoint.

Warning

All the functions for segmented mixed models (*.segmented.lme) are still at an experimental stage

Author(s)

Vito M. R. Muggeo, `<vito.muggeo@unipa.it>`

See Also

`summary.segmented.lme`

Examples

```
##continues example from segmented.lme()
# vcov(os)
# vcov(os, B=50)
# vcov(os, B=50, ret.b=TRUE)
```

`vcov.stepmented` *Variance-Covariance Matrix for a Fitted Stepmented Model*

Description

Returns the variance-covariance matrix of the parameters estimates (including breakpoints) of a fitted stepmented model object.

Usage

```
## S3 method for class 'stepmented'
vcov(object, k=NULL, zero.cor=TRUE, type=c("cdf", "none", "abs"), ...)
```

Arguments

`object`	a fitted model object of class "stepmented", returned by any `stepmented` method
`k`	The power of `n` for the smooth approximation. Simulation evidence suggests `k` in $[-1, -1/2]$; with $k = -1/2$ providing somewhat 'conservative' standard errors especially at small sample sizes. In general, the larger k, the smaller n^{-k}, and the smaller the jumpoint standard error.
`zero.cor`	If `TRUE`, the covariances between the jumpoints and the remaining linear coefficients are set to zero (as theory states).
`type`	How the covariance matrix should be computed. If `"none"`, the usual asymptotic covariance matrix for the linear coefficients only (under homoskedasticity and assuming known the jumpoints) is returned; if `"cdf"`, the standard normal cdf is used to approximate the indicator function (see details); `"abs"` is yet another approximation (currently unimplemented).
`...`	additional arguments.

Details

The full covariance matrix is based on the smooth approximation

$$I(x > \psi) \approx \Phi((x - \psi)/n^k)$$

via the sandwich formula using the empirical information matrix and assuming $x \in [0, 1]$. $\Phi(\cdot)$ is the standard Normal cdf, and k is the argument `k`. When `k=NULL` (default), it is computed via

$$k = -(0.6 + 0.5\ \log(snr)/\sqrt{s}nr - (|\hat{\psi} - 0.5|/n)^{1/2})$$

where snr is the signal-to-noise ratio corresponding to the estimated changepoint $\hat{\psi}$ (in the range (0,1)). The above formula comes from extensive simulation studies under different scenarios: Seo and Linton (2007) discuss using the normal cdf to smooth out the indicator function by suggesting $\log(n)/n^{1/2}$ as bandwidth; we found such suggestion does not perform well in practice.

Value

The full matrix of the estimated covariances between the parameter estimates, including the breakpoints.

Warning

The function, including the value of k, must be considered at preliminary stage. Currently the value of k appears to overestimate slightly the true $\hat{\psi}$ variability.

Note

If the fit `object` has been called by `stepmented(.., var.psi=TRUE)`, then `vcov.stepmented` will return `object$vcov`, unless the power `k` differs from `-2/3`.

Author(s)

Vito Muggeo

References

Seo MH, Linton O (2007) A smoothed least squares estimator for threshold regression models, J of Econometrics, 141: 704-735

See Also

stepmented

Examples

```
##see ?stepmented
```

Index