

# Package ‘segmented’

December 9, 2022

**Type** Package

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

**Version** 1.6-2

**Date** 2022-12-09

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

**Description** Given a regression model, segmented `updates' it by adding one or more segmented (i.e., piece-wise linear) relationships. Several variables with multiple breakpoints 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](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 in Muggeo (2014, <doi:10.1177/1471082X13504721>).

**Depends** MASS, nlme

**License** GPL

**NeedsCompilation** no

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

**Repository** CRAN

**Date/Publication** 2022-12-09 18:50:02 UTC

## R topics documented:

segmented-package . . . . .	2
aapc . . . . .	4
broken.line . . . . .	5
confint.segmented . . . . .	7
confint.segmented.lme . . . . .	9
davies.test . . . . .	10
down . . . . .	12
draw.history . . . . .	13

fitted.segmented.lme . . . . .	14
intercept . . . . .	15
lines.segmented . . . . .	16
plant . . . . .	17
plot.segmented . . . . .	18
plot.segmented.lme . . . . .	21
points.segmented . . . . .	23
predict.segmented . . . . .	24
print.segmented . . . . .	25
print.segmented.lme . . . . .	26
pscore.test . . . . .	27
pwr.seg . . . . .	29
seg.control . . . . .	31
seg.lm.fit . . . . .	34
segmented . . . . .	36
segmented.lme . . . . .	43
selgmented . . . . .	47
slope . . . . .	49
stagnant . . . . .	51
summary.segmented . . . . .	52
summary.segmented.lme . . . . .	54
vcov.segmented . . . . .	55
vcov.segmented.lme . . . . .	57

## Index 59

---

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

---

## Description

Estimation and inference of regression models with piecewise linear relationships, also known as segmented regression models, having a fixed number of break-points. Random effects changepoints are also allowed since version 1.6-0.

## Details

Package: segmented  
 Type: Package  
 Version: 1.6-2  
 Date: 2022-12-09  
 License: GPL

Package segmented is aimed to estimate linear and generalized linear models (and virtually any regression model) having one or more segmented 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 algorithm used by `segmented` is *not* grid-search. It is an iterative procedure (Muggeo, 2003) that needs starting values *only* for the breakpoint parameters and therefore it is 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 algorithm less sensitive to starting values.

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 examples in `segmented.default`.

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

### 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.
- 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 transition 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 <i>single</i> 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 desired.
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 <i>full</i> covariance matrix of estimates. If unspecified (i.e. NULL), the covariance matrix is computed internally by <code>vcov(ogg, ...)</code> .
.coef	The regression parameter estimates. If unspecified (i.e. NULL), it is computed internally by <code>coef(ogg)</code> .
...	further arguments to be passed on to <code>vcov.segmented()</code> , such as <code>var.diff</code> or <code>is</code> .

## 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 ( $\beta_j$ ) 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 each 'segmented' relationship.

**Usage**

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

**Arguments**

<code>ogg</code>	A fitted object of class <code>segmented</code> (returned by any <code>segmented</code> method).
<code>term</code>	Three options. i) A named list (whose name should be one of the segmented covariates in the model <code>ogg</code> ) 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 <code>NULL</code> if the model includes a single segmented covariate, and predictions corresponding to the observed covariate values are returned;
<code>link</code>	Should the predictions be computed on the scale of the link function? Default to <code>TRUE</code> .
<code>interc</code>	Should the model intercept be added? (provided it exists).
<code>se.fit</code>	If <code>TRUE</code> also standard errors for predictions are returned.
<code>isV</code>	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. If a single logical is provided, it is applied to both terms.
<code>.vcov</code>	Optional. The <i>full</i> covariance matrix of estimates. If <code>NULL</code> (and <code>se.fit=TRUE</code> ), the matrix is computed internally via <code>vcov.segmented()</code> .
<code>.coef</code>	The regression parameter estimates. If unspecified (i.e. <code>NULL</code> ), it is computed internally by <code>coef(ogg)</code> .
<code>...</code>	Additional arguments to be passed on to <code>vcov.segmented()</code> when computing the standard errors for the predictions, namely <code>is</code> , <code>var.diff</code> , <code>p.df</code> . See <a href="#">summary.segmented</a> and <a href="#">vcov.segmented</a> .

**Details**

If `term=NULL` or `term` is a valid segmented covariate name, predictions for each 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.

**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,psi=.5)
## 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 <a href="#">summary.segmented</a> .
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 <a href="#">segmented.lme</a> .
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 <code>vcov.segmented.lme(..., ret.b=TRUE)</code>
...	if <code>obj.boot</code> is missing and bootstrap CIs are requested, additional optional arguments, such as <code>B</code> , <code>seed</code> , and <code>it.max.b</code> , 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)
```

---

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 <code>glm</code> or <code>lm</code> . Even an object returned by segmented can be set (e.g. if interest lies in testing for an additional break-point).
seg.Z	a formula with no response variable, such as <code>seg.Z=~x1</code> , indicating the (continuous) segmented variable being tested. Only a single variable may be tested and an error is printed when <code>seg.Z</code> includes two or more terms. <code>seg.Z</code> can be omitted if i) <code>obj</code> 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 <code>obj</code> 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  $\beta$  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 <code>seg.Z</code> at which the maximum (or the minimum if <code>alternative="less"</code> ) 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 <code>matplot()</code> .

**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**

<code>object</code>	Object of class "segmented.lme"
<code>level</code>	the level to be considered. Currently only levels 0 or 1 are allowed.
<code>sort</code>	If TRUE, the fitted values are sorted by the names of the 'id' levels.
<code>...</code>	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](#)

---

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 <i>full</i> covariance matrix of estimates. If unspecified (i.e. NULL), the covariance matrix is computed internally by <code>vcov(ogg)</code> .
.coef	The regression parameter estimates. If unspecified (i.e. NULL), it is computed internally by <code>coef(ogg)</code> .
digits	controls number of digits in the returned output.
...	Further arguments to be passed on to <code>vcov.segmented</code> , such as <code>var.diff</code> and <code>is</code> . See Details in <a href="#">vcov.segmented</a> .

**Details**

A broken-line relationship means that a regression equation exists in the intervals ' $\min(x)$  to  $\psi_1$ ', ' $\psi_1$  to  $\psi_2$ ', 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**

<code>x</code>	an object of class <code>segmented</code> .
<code>term</code>	the segmented variable of the breakpoints being drawn. It may be unspecified when there is a single segmented variable.
<code>bottom</code>	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 <a href="#">Details</a> .
pch	either an integer specifying a symbol or a single character to be used in plotting the point estimates of the breakpoints. See <a href="#">points</a> .
rev.sgn	should the signs of the breakpoint estimates be changed before plotting? see <a href="#">Details</a> .
.vcov	The <i>full</i> covariance matrix of estimates. If unspecified (i.e. NULL), the covariance matrix is computed internally by <code>vcov(x)</code> .
.coef	The regression parameter estimates. If unspecified (i.e. NULL), it is computed internally by <code>coef(x)</code> .
...	further arguments passed to <a href="#">segments</a> , 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 join-points.

### Examples

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

---

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)
attach(plant)

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

## 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=1, rev.sgn=FALSE, const=0, 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, ...)
```

**Arguments**

x	a fitted segmented object.
term	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.
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 intervals 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 transformed 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.
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 <a href="#">broken.line</a> ). Ignored if conf.level=0
is	logical value (to be passed to <a href="#">broken.line</a> ) 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 <a href="#">summary.segmented</a> to compute the standard errors of fitted values (if conf.level>0).
p.df	degrees of freedom when var.diff=TRUE, see <a href="#">summary.segmented</a>
.vcov	The <i>full</i> 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 <code>vcov.segmented</code> .
.coef	The regression parameter estimates. If unspecified (i.e. NULL), it is computed internally by <code>coef()</code> .
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.

... 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.

### 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, ...)
```

## 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 extent, 'population' or 'marginal' estimates), otherwise 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 <code>par(mfrow=c(. . .))</code> for plotting multiple panels (should be coherent with <code>length(id)</code> ). If missing, it is computed automatically depending on <code>length(id)</code> .
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: <code>by=list(sex="male", z=.2)</code> , provided that the variables <code>sex</code> and <code>z</code> affect the segmented relationship. Effective only if <code>level=0</code> .
add	If TRUE the (fixed-effect) fitted segmented relationship is added to the current plot. Effective only if <code>level=0</code> .
conf.level	The confidence level for pointwise confidence intervals. Effective only if <code>level=0</code> .
withI	If TRUE, the level 0 segmented relationship is computed with the model intercept. Effective only if <code>level=0</code> .
vcov.	The fixed effects covariance matrix. If NULL, it is computed by <code>vcov.segmented.lme()</code> . Effective only if <code>level=0</code> .
shade	If TRUE (and <code>conf.level&gt;0</code> ) the area within the pointwise CIs is shaded. Effective only if <code>level=0</code> .
drop.var	Possible coefficient names to be removed before computing the segmented relationship (E.g. the group-specific intercept.).
text.leg	If specified (and <code>pos.leg</code> has been also specified), it is the legend text. Effective only if <code>level=0</code> .
...	additional arguments, such as 'ylab', 'xlab', 'col.l' (for the fitted line) and 'col' (for the data points, provided that <code>res=TRUE</code> has been set). If <code>level=0</code> and <code>conf.level&gt;0</code> , <code>lty</code> and <code>lwd</code> can be vectors.

### Details

The function plots the 'subject'-specific segmented profiles for the 'subjects' specified in `id` or, if `level=0`, the fitted segmented relationship based on fixed effects only.

### 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

## 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, ...)
```

**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 <i>full</i> covariance matrix of estimates. If unspecified (i.e. NULL), the covariance matrix is computed internally by <code>vcov()</code> .
.coef	The regression parameter estimates. If unspecified (i.e. NULL), it is computed internally by <code>coef(x)</code> .
const	A constant to be added (on the y-scale) to the values before transforming and plotting.
...	other graphics parameters to pass on to <code>points()</code> function.

### 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, .coef=NULL, ...)
```

### Arguments

object	a fitted segmented model coming from <code>segmented.lm</code> or <code>segmented.glm</code> .
newdata	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
.coef	The regression parameter estimates. If unspecified (i.e. NULL), it is computed internally by <code>coef()</code> .
...	further arguments passed to <code>predict.lm</code> or <code>predict.glm</code> . Usually these are <code>se.fit</code> , or <code>interval</code> or <code>type</code> .



**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`.

**Note**

If `type="terms"`, `predict.segmented` returns predictions for each component of the segmented term. Namely if `'my.x'` is the segmented variable, predictions for `'my.x'`, `'U1.my.x'` and `'psi1.my.x'` are returned. These are meaningless individually, however their sum provides the predictions for the segmented term.

**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
```

---

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=1)
```

**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 <code>seg.Z=~x1</code> , indicating the (continuous) segmented variable being tested. Only a single variable may be tested and an error is printed when <code>seg.Z</code> includes two or more terms. <code>seg.Z</code> can be omitted if i) <code>obj</code> 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 degrees-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 <i>additional</i> breakpoint(s) for the variable 'seg.Z'. Ignored when obj is not a segmented fit.
n.break	optional. Number of breakpoints postulated 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. 1 is for piecewise-linear relationships, 2 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  $\beta$  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)
```

**Description**

Given the appropriate input values, the function computes the power (sample size) corresponding to the specified 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=1, 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 <code>alternative</code> and <code>alpha</code> are ignored.
pow	The desired power level. If provided <code>n</code> has to be missing
n	The fixed sample size. If provided <code>pow</code> 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 <code>p</code> and possible other numerical values as arguments. For instance "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 <code>psi</code> within the covariate range. Finally, it could be also a numerical vector meaning the actual covariate, but <code>pow</code> has to be missing. Namely if the covariate is supplied (and <code>n</code> 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 <code>pow</code> is provided and the relevant sample size estimate has to be returned, the function evaluates 50 sample sizes equally spaced in <code>n.range</code> . 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	Numerical, meaning the type of breakpoint. <code>break.type=1</code> means piecewise linear (segmented), <code>break.type=2</code> 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 <code>oseg</code> has been supplied, <code>ci.pow</code> 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.control

*Auxiliary for controlling segmented model fitting*

---

**Description**

Auxiliary function as user interface for 'segmented' fitting. Typically only used when calling any 'segmented' method (segmented.lm, segmented.glm, segmented.Arima or segmented.default).

**Usage**

```
seg.control(n.boot=10, display = FALSE, tol = 1e-05, it.max = 30, fix.npsi=TRUE,
  K = 10, quant = TRUE, maxit.glm = 25, h = 1, break.boot=5, size.boot=NULL,
  jt=FALSE, nonParam=TRUE, random=TRUE, seed=12345, fn.obj=NULL, digits=NULL,
  conv.psi=FALSE, alpha = NULL, min.step=.0001,
  powers=c(1,1), last = TRUE, stop.if.error = NULL, gap=FALSE, fc=.95)
```

**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. 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 <i>number</i> (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 <a href="#">segmented.lm</a> or <a href="#">segmented.glm</a> . 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 <a href="#">selgmented</a> .
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).
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, the algorithm stops without performing all n.boot ‘trials’. This can save computational time considerably.
size.boot	the size of the bootstrap samples. If NULL, it is taken equal to the actual sample size.
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. Set to NULL to use a random seed. Fixing the seed can be useful to replicate the results when the



	bootstrap restart algorithm is employed. The segmented fit includes seed representing the integer vector saved just before the bootstrap resampling. Re-use it if you want to replicate the bootstrap restarting algorithm with the <i>same</i> samples.
fn.obj	A <i>character string</i> 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 <i>minimized</i> . 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 <a href="#">segmented.default</a> .
digits	optional. If specified it means the desired number of decimal points of the breakpoint to be used during the iterative algorithm.
conv.psi	optional. Should convergence of iterative procedure to be assessed on changes of breakpoint estimates or changes in the objective? Default to FALSE.
alpha	optional numerical value. The breakpoints are estimated within the quantiles alpha and 1-alpha of the relevant covariate. Defaults to NULL which means max(.05, 1/n).
min.step	optional. The minimum step size to break the iterative algorithm. Default to 0.0001.
powers	The powers of the pseudo covariates employed by the algorithm. These can be altered during the iterative process to stabilize the estimation procedure. Usually of no interest for the user. <i>This argument will be removed in next releases.</i>
last	logical indicating if output should include only the last fitted model. <i>This argument will be removed in next releases</i>
stop.if.error	same than fix.npsi. <i>This argument will be removed in next releases</i> , and replaced by fix.npsi. If provided, and different from NULL, it overwrites fix.npsi
gap	logical, if FALSE the gap coefficients are <i>always</i> constrained to zero at the convergence. <i>This argument will be removed in next releases.</i>
fc	A proportionality factor ( $\leq 1$ ) to adjust the breakpoint estimates <i>if</i> 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 parameter.

## 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. In this case, it may be useful to set also `fix.npsi=FALSE` to automate the procedure, see Muggeo and Adelfio (2011). The maximum number of iterations (`it.max`) should be also increased when the 'automatic' procedure is used.

If `last=TRUE`, the object resulting from `segmented.lm` (or `segmented.glm`) is a list of fitted GLM; the *i*-th model is the segmented model with the values of the breakpoints at the *i*-th iteration.

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. 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. 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)

```

### 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	possible weights vector.
offs	possible 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**

<code>obj</code>	standard 'linear' model of class "lm", "glm" or "Arima". Since version 0.5.0-0 any regression fit may be supplied (see 'Details'). <code>obj</code> could include any covariate, and if there is also the variable specified in <code>seg.Z</code> , then all the slopes of the fitted segmented relationship will be estimated. If <code>obj</code> misses the segmented variable, then the 1st (the leftmost) slope is assumed to be zero. Since version 1.5.0, <code>obj</code> can be a simple numeric or <code>ts</code> object, see examples below.
<code>seg.Z</code>	the segmented variables(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 <code>seg.Z=~x</code> or <code>seg.Z=~x1+x2</code> . It can be missing when <code>obj</code> includes only one covariate which is taken as segmented variable. Currently, formulas involving functions, such as <code>seg.Z=~log(x1)</code> , or selection operators, such as <code>seg.Z=~d["x1"]</code> or <code>seg.Z=~d\$x1</code> , are <i>not</i> allowed. Also, variable names formed by U or V only (with or without numbers) are not permitted.
<code>psi</code>	starting values for the breakpoints to be estimated. If there is a single segmented variable specified in <code>seg.Z</code> , <code>psi</code> 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 <code>seg.Z</code> includes several covariates, <code>psi</code> has to be specified as a <i>named</i> list of vectors whose names have to match the variables in the <code>seg.Z</code> argument. Each vector of such list includes starting values for the break-point(s) for the corresponding variable in <code>seg.Z</code> . A NA value means that 'K' quantiles (or equally spaced values) are used as starting values; K is fixed via the <code>seg.control</code> auxiliary function.

<code>npsi</code>	A named vector or list meaning the <i>number</i> (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 <code>quant</code> in <code>seg.control</code> . <code>npsi</code> can be missing and <code>npsi=1</code> is assumed for all variables specified in <code>seg.Z</code> . If <code>psi</code> is provided, <code>npsi</code> is ignored.
<code>fixed.psi</code>	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 <code>seg.Z</code> . If there is a single variable in <code>seg.Z</code> , a simple numeric vector can be specified. Note that, in addition to the values specified here, <code>segmented</code> will estimate additional breakpoints. To keep fixed all breakpoints (to be specified in <code>psi</code> ) use <code>it.max=0</code> in <code>seg.control</code>
<code>control</code>	a list of parameters for controlling the fitting process. See the documentation for <code>seg.control</code> for details.
<code>model</code>	logical value indicating if the <code>model.frame</code> should be returned.
<code>keep.class</code>	logical value indicating if the final fit returned by <code>segmented.default</code> should keep the class 'segmented' (along with the class of the original fit obj). Ignored by the <code>segmented</code> methods.
<code>...</code>	optional arguments (to be ignored safely). Notice specific arguments relevant to the original call (via <code>lm</code> or <code>glm</code> for instance), such as <code>weights</code> or <code>offset</code> , have to be included in the starting model obj
<code>adjX</code>	if obj is a <code>ts</code> , the segmented variable (if not specified in <code>seg.Z</code> ) is computed by taking information from the time series (e.g., years starting from 2000, say). If <code>adjX=TRUE</code> , 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 <code>adjX=TRUE</code> .
<code>weights</code>	the weights if obj is a vector or a <code>ts</code> 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

The returned object depends on the `last` component returned by `seg.control`. If `last=TRUE`, the default, `segmented` returns an object of class "segmented" which inherits from the class "lm" or "glm" depending on the class of `obj`. Otherwise a list is returned, where the last component is the fitted model at the final iteration, see [seg.control](#).

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

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

### Warning

It is well-known that the log-likelihood function for the break-point may be not concave, especially for poor clear-cut kink-relationships. In these circumstances the initial guess for the break-point, i.e. the `psi` argument, must be provided with care. For instance visual inspection of a, possibly smoothed, scatter-plot is usually a good way to obtain some idea on breakpoint location. However bootstrap restarting, implemented since version 0.2-9.0, is relatively more robust to starting values specified in `psi`. Alternatively an automatic procedure may be implemented by specifying `psi=NA` and `fix.npsi=FALSE` in [seg.control](#): experience suggests to increase the number of iterations via `it.max` in `seg.control()`. This automatic procedure, however, is expected to overestimate the number of breakpoints.

### 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`.
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](mailto: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.lme](#) for 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 1 breakpoint for that:
#... you need to specify the variable 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, 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 by specifying 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!
#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: segmented(yy[-1],..) also works
#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)

#####Three 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)

## 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"),
  nq = 0, adjust = 0, 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 sub-regression model for the change-point parameter. Default to ~1.
x.diff	Optional. A one-sided formula meaning 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 <a href="#">seg.control</a> , 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. (changept fixed)
random = list(id = pdDiag(~1 + x + U + G0)) #ind. random eff. (in the changept 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 `Z`. In each case the 'linear predictor' is  $\eta_i = \kappa_0 + \kappa_1 * z_i + k_i$ , where `z` is the covariate specified in `z.psi` and the  $k_i$ s are the changepoint random effects included by means of `G0` in the `random` argument.

nq Integer value to assess if the estimated breakpoint for the subject `i` can be considered 'reliable' or not. For each subject,  $\hat{\psi}$  is not considered reliable if

$$\hat{\psi} \leq z_{(1+nq)} \text{ and } \hat{\psi} \geq z_{(n_i-nq)}$$

where  $z_{(\cdot)}$  means the sorted values of the segmented variable. Note `nq` just affects the logical vector `attr(ψ.i,"is.break")` where `ψ.i` is the component of the model fit including the subject-specific breakpoints. Default is `nq=0`.

adjust A numerical (0 or 1) value. If `adjust=1` at convergence and for each subject `i` the estimated breakpoint assessed as unreliable (see argument `nq`) is moved to the maximum of the covariate range (for the subject `i`). Thus for the subjects with 'unreliable' breakpoint, the corresponding fitted profile will be truly linear.

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).

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

<code>lme.fit</code>	The fitted lme object at convergence
<code>lme.fit.noG</code>	The fitted lme object at convergence assuming known the breakpoints
<code>psi.i</code>	The subject/cluster-specific change points (fixed + random). It includes 2 attributes: <code>attr("ni")</code> for the number of measurements in each 'cluster', and <code>attr("is.break")</code> a vector of logicals indicating if the breakpoint for each subject <code>i</code> can be reliable (TRUE) or not (FALSE). Here 'reliable' simply means within the covariate range (for subject <code>i</code> ). See also argument <code>nq</code> .
<code>fixed.eta.psi</code>	The fixed-effect linear predictor for the change points regression equation. These values will differ among 'clusters' only if at least one covariate has been specified in <code>z.psi</code> .
<code>fixed.eta.delta</code>	The fixed-effect linear predictor of the slope difference regression equation. These values will differ among 'clusters' only if at least one covariate has been specified in <code>x.diff</code> .

### Warning

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

### Note

Currently only one breakpoint (with or without random effects) can be estimated.

**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(z=.1)) #the slope estimates when z=.1

#####
```

```

# 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))

plot(os2, n.plot = c(3,3))
os2$psi.i
attr(os2$psi.i, "is.break") #it is FALSE for Subject #7

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

## End(Not run)

```

---

selgmented

*Selecting number of breakpoints in segmented regression*


---

## Description

This function selects the number of breakpoints of the segmented relationship according to the BIC criterion or sequential hypothesis testing.

## Usage

```

selgmented(olm, seg.Z, alpha = 0.05, type = c("score", "davies", "bic", "aic"),
  control = seg.control(), refit=TRUE, stop.if=6, return.fit = TRUE,
  bonferroni = FALSE, Kmax = 2, msg = TRUE, plot.ic = FALSE, th = NULL)

```

## Arguments

olm	A starting <code>lm</code> or <code>glm</code> 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 <code>olm</code> includes just one covariate.
alpha	The fixed type I error probability.

<code>type</code>	Which criterion should be used? Options <code>score</code> and <code>davies</code> allow to carry out sequential hypothesis testing with no more than 2 breakpoints ( $K_{\max}=2$ ). Alternatively, the number of breakpoints can be selected via the BIC (or AIC) with virtually no upper bound for $K_{\max}$ .
<code>control</code>	See <a href="#">seg.control</a> .
<code>refit</code>	If TRUE, the final selected model is re-fitted using arguments in <code>control</code> , typically with bootstrap restarting. Set <code>refit=FALSE</code> to speed up computation (and possibly accepting near-optimal estimates). Ignored if <code>type='score'</code> or <code>type='davies'</code> .
<code>stop.if</code>	An integer. If <code>stop.if</code> fits provide higher AIC/BIC values the search is interrupted. Ignored if <code>type='score'</code> or <code>type='davies'</code> .
<code>return.fit</code>	If TRUE, the fitted model (with the number of breakpoints selected according to <code>type</code> ) is returned.
<code>bonferroni</code>	If TRUE, the Bonferroni correction is employed, i.e. $\alpha/K_{\max}$ is always taken as threshold value to reject or not. If FALSE, $\alpha$ is used in the second level of hypothesis testing.
<code>Kmax</code>	The maximum number of breakpoints being tested. If <code>type='bic'</code> or <code>type='aic'</code> , any integer value can be specified, otherwise at most $K_{\max}=2$ breakpoints can be tested via the Score or Davies statistics.
<code>msg</code>	If FALSE the final fit is returned silently with the selected number of breakpoints, otherwise the message including information about the selection procedure (i.e. the BIC values) is printed.
<code>plot.ic</code>	If TRUE the information criterion values with respect to the number of breakpoints are plotted. Ignored if <code>type='score'</code> or <code>type='davies'</code> .
<code>th</code>	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 <code>th</code> , one (the first) is deleted. Ignored if <code>type='score'</code> or <code>type='davies'</code> . Of course, <code>th</code> depends on the $x$ scale: Integers, like 5 or 10, are appropriate if the covariate is the observation index. Default (NULL) means $th=\text{diff}(\text{range}(x))/100$ .

### Details

The function uses properly the functions `segmented`, `pscore.test` or `davies.test` to select the 'optimal' number of breakpoints  $0, 1, \dots, K_{\max}$ . If `type='bic'` or `'aic'`, the procedure stops if the last `stop.if` fits have increasing values of the information criterion.

### 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 with the component `selection.psi` including the aforementioned information. See [segmented](#) for details.

### Note

This is a sperimental function. Please use that with caution.



**Author(s)**

Vito 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**

```
## from ?segmented
## Not run:
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) ## selects number of breakpoints via the Score test

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

#####
#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(X)
y<-mu + rnorm(n)*.02
par(mfrow=c(1,2))
o<-selgmented(y, Kmax=20, type='bic', plot.ic=TRUE, refit=FALSE)
plot(o, res=T, col=2, lwd=3)

## End(Not run)
```

---

slope

*Slope estimates from segmented relationships*


---

**Description**

Computes the slopes of each 'segmented' 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, ..., digits = max(4, getOption("digits") - 2))
```

**Arguments**

<code>ogg</code>	an object of class "segmented", returned by any segmented method or a list of two segmented fits to compare the estimates of corresponding slopes.
<code>parm</code>	the segmented variable whose slopes have to be computed. If missing all the segmented variables are considered.
<code>conf.level</code>	the confidence level required.
<code>rev.sgn</code>	vector of logicals. The length should be equal to the length of <code>parm</code> , but it is recycled otherwise. When TRUE it is assumed that the current <code>parm</code> 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.
<code>APC</code>	logical. If APC=TRUE the 'annual percent changes', i.e. $100 \times (\exp(\beta) - 1)$ , are computed for each interval ( $\beta$ is the slope). Only point estimates and confidence intervals are returned.
<code>.vcov</code>	The <i>full</i> covariance matrix of estimates. If unspecified (i.e. NULL), the covariance matrix is computed internally by <code>vcov(ogg)</code> .
<code>.coef</code>	The regression parameter estimates. If unspecified (i.e. NULL), it is computed internally by <code>coef(ogg)</code> .
<code>use.t</code>	Which quantiles should be used to compute the confidence intervals? If NULL (default) the <i>t</i> distribution is used only for objects obtained by <code>segmented.lm</code> .
<code>by</code>	Only for <code>segmented.lme</code> objects. It is a named list indicating covariate names and corresponding values affecting the fitted segmented relationship. For instance, <code>by=list(group="2", z2=.2)</code> , provided that the model has been fitted by specifying <code>group</code> and <code>z2</code> in <code>x.diff</code> and/or <code>z.psi</code>
<code>...</code>	Further arguments to be passed on to <code>vcov.segmented</code> , such as <code>var.diff</code> and <code>is</code> . See Details in <a href="#">vcov.segmented</a> and <a href="#">summary.segmented</a> .
<code>digits</code>	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 responsibility 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 transition between two intersecting straight lines. *Biometrika* **58**: 525 – 534.

Originally from the PhD thesis by R.A. Cook

**Examples**

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

---

summary.segmented	<i>Summarizing model fits for segmented regression</i>
-------------------	--

---

**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  $j$ th 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**

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

---

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

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

# Index

- \* **changepoint**
  - plot.segmented.lme, 21
- \* **datasets**
  - down, 12
  - plant, 17
  - stagnant, 51
- \* **hplot**
  - plot.segmented, 18
- \* **htest**
  - davies.test, 10
  - pscore.test, 27
  - slope, 49
- \* **models**
  - predict.segmented, 24
  - print.segmented, 25
  - print.segmented.lme, 26
- \* **nonlinear**
  - broken.line, 5
  - confint.segmented, 7
  - draw.history, 13
  - lines.segmented, 16
  - plot.segmented, 18
  - plot.segmented.lme, 21
  - points.segmented, 23
  - seg.lm.fit, 34
  - segmented, 36
  - segmented-package, 2
- \* **regression**
  - aapc, 4
  - broken.line, 5
  - confint.segmented, 7
  - draw.history, 13
  - intercept, 15
  - lines.segmented, 16
  - plot.segmented, 18
  - plot.segmented.lme, 21
  - points.segmented, 23
  - predict.segmented, 24
  - seg.control, 31
  - seg.lm.fit, 34
  - segmented, 36
  - segmented-package, 2
  - slope, 49
  - summary.segmented, 52
  - summary.segmented.lme, 54
  - vcov.segmented, 55
  - vcov.segmented.lme, 57
- aapc, 4
- broken.line, 5, 19, 25
- coef.segmented, 40
- coef.segmented(print.segmented), 25
- confint.segmented, 7, 40
- confint.segmented.lme, 9
- davies.test, 10, 28, 29, 49, 51, 54
- down, 12
- draw.history, 13
- fitted.segmented.lme, 14
- fixef.segmented.lme  
(print.segmented.lme), 26
- intercept, 15
- lines.segmented, 8, 16, 20, 40
- logLik.segmented.lme  
(print.segmented.lme), 26
- plant, 17
- plot.segmented, 6, 17, 18, 24, 25, 40
- plot.segmented.lme, 21, 46
- points, 17
- points.segmented, 17, 20, 23, 40
- predict.glm, 25
- predict.lm, 25
- predict.segmented, 6, 20, 24, 40
- print.segmented, 25, 40, 54

print.segmented.lme, 26, 55  
print.summary.segmented, 26, 40  
print.summary.segmented  
    (summary.segmented), 52  
pscore.test, 12, 27, 31, 49, 51  
pwr.seg, 29

seg.Ar.fit(seg.lm.fit), 34  
seg.control, 3, 31, 37–39, 43, 48  
seg.def.fit(seg.lm.fit), 34  
seg.glm.fit(seg.lm.fit), 34  
seg.lm.fit, 34  
seg.num.fit(seg.lm.fit), 34  
segmented, 6, 8, 11, 20, 25, 36, 48, 49  
segmented-package, 2  
segmented.default, 3, 33, 46  
segmented.glm, 32, 36  
segmented.lm, 32, 36  
segmented.lme, 3, 9, 23, 27, 40, 43  
segments, 17  
selgmented, 3, 32, 47  
slope, 16, 49  
stagnant, 51  
summary.segmented, 6, 7, 19, 26, 40, 50, 52,  
    56  
summary.segmented.lme, 15, 27, 54, 57

vcov.segmented, 6, 15, 40, 50, 55  
vcov.segmented.lme, 10, 57