Package ‘segmented’

May 10, 2012

Type Package

Title Segmented relationships in regression models with breakpoints/changepoints estimation

Version 0.2-9.1

Date 2012-05-09

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

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

Description Given a (generalized) linear model, segmented ‘updates’the model by adding one or more segmented relationships. Several variables with multiple breakpoints are allowed.

License GPL

Repository CRAN

Date/Publication 2012-05-10 04:14:56

R topics documented:

`segmented-package` *Segmented relationships in regression models with breakpoints/changepoints estimation*

Description

Estimation of Generalized Linear Models with piecewise linear relationships having a fixed number of break-points.

Details

Package:	segmented
Type:	Package
Version:	0.2-9.1
Date:	2012-05-09
License:	GPL

Package `segmented` is aimed to estimate linear and generalized linear models having one or more segmented relationships in the linear predictor. Estimates of the slopes and of the possibly multiple breakpoints are provided. 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.

A tentative approach to deal with unknown number of breakpoints is also provided, see option `stop.if.error` in `seg.control`.

Author(s)

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

References

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

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

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

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

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

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

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

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

Description

Given a segmented model (typically returned by a `segmented` method), `broken.line` computes the fitted values for each ‘segmented’ relationship.

Usage

```
broken.line(ogg, term = NULL, gap = FALSE, linkinv = FALSE,
    interc=TRUE)
```

Arguments

`ogg`	A fitted object of class segmented (returned by any `segmented` method).
`term`	A character meaning for which segmented variable prediction should be computed.
`gap`	Should the ‘gaps’ of the segmented relationships be plotted? Default to `FALSE`.
`linkinv`	Should the predictions be computed on the scale of the link function? Default to `FALSE`.
`interc`	Should the model intercept be added? (provided it exists).

Details

If `term=NULL` predictions for each segmented variable in the fitted model are computed. Argument `linkinv` is ignored whether `ogg` does not inherit from the class "glm".

Value

A matrix whose columns represent predictions for the segmented variables.

See Also

segmented, predict.glm

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=list(z=.5))
## Not run: plot(z,y)
## Not run: points(z,broken.line(o.seg,linkinv=TRUE),col=2,pch=20)
```

`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, rev.sgn=FALSE, var.diff=FALSE,
        digits=max(3, getOption("digits") - 3), ...)
```

Arguments

`object`	a fitted segmented object.
`parm`	the segmented variable of interest. If missing all the segmented variables are considered.
`level`	the confidence level required (default to 0.95).
`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 `var.diff=TRUE` and there is a single segmented variable, the standard error is based on sandwich-type formula of the covariance matrix. See Details in summary.segmented.
`digits`	controls the number of digits to print when printing the output.
`...`	additional parameters

Details

Currently `confint.segmented` computes confidence limits for the breakpoints using 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`.

Value

A list of matrices. Each matrix includes point estimate and confidence limits of the breakpoint(s) for each segmented variable in the model.

Author(s)

Vito M.R. Muggeo

See Also

`segmented`

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)
```

`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"), beta0=0, dispersion=NULL)
```

Arguments

- `obj` — a fitted model returned by `glm` or `lm`.
- `seg.Z` — a formula with no response variable, such as `seg.Z=~x1`, indicating the (continuous) segmented variable being tested. Only a single variable may be tested and a warning is printed when `seg.Z` includes two or more terms.
- `k` — number of points where the test should be evaluated. See details.
- `alternative` — a character string specifying the alternative hypothesis.
- `beta0` — the null value of the difference-in-slope; default to zero meaning no breakpoint, see details.
- `dispersion` — the dispersion parameter for the family to be used to compute the Wald 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 $H_0 : \beta = \beta_0$, where β is the difference-in-slope, i.e. the coefficient of the segmented function $\beta(x - \psi)_+$, and β_0 is the 'null' value specified via the argument `beta0`. Roughly speaking, the procedure computes `k` 'naive' (i.e. assuming fixed and known the breakpoint) Wald statistics for the difference-in-slope, seeks the 'best' value (according to the alternative hypothesis), and then corrects the selected (minimum) p-value. The `k` evaluation points are `k` equally spaced values between the 0.05 and 0.95 quantiles of the variable reported in `seg.Z`.

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 at which the maximum (or the minimum if `alternative="less"`) occurs
- `parameter` — number of evaluation points
- `p.value` — the adjusted p-value
- `process` — a two-column matrix including the evaluation points and corresponding values of the statistic

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.

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.

Examples

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

davies.test(o,~z)
davies.test(o,~x)

## 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`	the 'segmented' variable whose breakpoint values throughout iterations have to be displayed.
`...`	graphic parameters to be passed to `matplot()`.

Details

For a given `term` in a segmented fit, `draw.history()` displays the different breakpoint values obtained during the estimating process, since the starting values up to the final ones.

Value

None.

Author(s)

Vito M.R. Muggeo

Examples

```
data(stagnant)
os<-segmented(lm(y~x,data=stagnant),seg.Z=~x,psi=list(x=-.8))
draw.history(os)
```

intercept — *Intercept estimates from segmented relationships*

Description

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

Usage

```
intercept(ogg, parm, gap = TRUE, rev.sgn = FALSE, var.diff=FALSE)
```

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.
`gap`	logical. should the intercepts account for the (possible) gaps?
`rev.sgn`	vector of logicals. The length should be equal to the length of `parm`, but it is recycled otherwise. when `TRUE` it is assumed that the current `parm` is ‘minus’ the actual segmented variable, therefore the sign is reversed before printing. This is useful when a null-constraint has been set on the last slope.
`var.diff`	Currently ignored as only point estimates are computed.

Details

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

Value

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

Author(s)

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

See Also

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

Examples

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

## End(Not run)
```

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

Description

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

Usage

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

Arguments

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

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 for the selected segmented term.

Usage

```
## S3 method for class 'segmented'
plot(x, term, add=FALSE, res=FALSE, se=FALSE, show.gap=TRUE, linkinv = FALSE,
      res.col=1,  rev.sgn=FALSE, const=0, ...)
```

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, 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.
se	when TRUE pointwise confidence intervals are drawn. Currently unimplemented.
show.gap	when FALSE the (possible) gaps between the fitted lines at the estimated breakpoints are hidden.
linkinv	when TRUE, the fitted lines are tranformed on the inverse link (i.e. response) scale before plotting.
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.
...	other graphics parameters to pass on to plotting commands: 'col', 'lwd' and 'lty' (that can be vectors, see the example below) for the fitted piecewise lines; 'ylab', 'xlab', 'main', 'sub' when a new plot is produced (i.e. when add=FALSE); 'pch' and 'cex' for the partial residuals (when res=TRUE).

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. Due to the parameterization of the segmented terms, sometimes the fitted lines may not appear to join at the estimated breakpoints. If this is the case, the apparent 'gap' would indicate some lack-of-fit. However such 'gap' may be hidden by setting `show.gap=FALSE`: `show.gap=TRUE` is suggested only when the gap coefficients are nonsignificant, see `summary.segmented(..)$gap`. The partial residuals are computed as 'fitted + residuals', where 'fitted' are the fitted values of the segmented relationship. Notice that for GLMs the residuals are the response residuals if `linkinv=TRUE` and the working residuals weighted by the IWLS weights if `linkinv=FALSE`.

Value

None.

Author(s)

Vito M.R. Muggeo

See Also

`lines.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=list(z=.5))
par(mfrow=c(1,2))
plot(o.seg)
plot(z,y)
## add the fitted lines using different colors and styles..
plot(o.seg,add=TRUE,linkinv=TRUE,lwd=2,col=2:3, lty=c(1,3))
lines(o.seg,col=2,pch=19,bottom=FALSE,lwd=2)
```

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

Description

Printing the most important feautures of a segmented model.

Usage

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

Arguments

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

Author(s)

Vito M.R. Muggeo

See Also

`summary.segmented`, `print.summary.segmented`

`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` or `segmented.glm`).

Usage

```
seg.control(toll = 1e-04, it.max = 10, display = FALSE,
    stop.if.error = TRUE, K = 10, last = TRUE, maxit.glm = 25, h = 1,
    n.boot=20, size.boot=NULL, gap=FALSE, jt=FALSE, nonParam=TRUE,
    random=TRUE, powers=c(1,1), seed=NULL)
```

Arguments

`toll`	positive convergence tolerance.
`it.max`	integer giving the maximal number of iterations.
`display`	logical indicating if output should be produced for each iteration.
`stop.if.error`	logical indicating if non-admissible break-points should be removed during the estimating algorithm. Set it to `FALSE` if you want to perform a sort of 'automatic' breakpoint selection, provided that several starting values are provided for the breakpoints. See argument `psi` in `segmented.lm` or `segmented.glm`. The idea of removing 'non-admissible' break-points during the iterative process is discussed in Muggeo and Adelfio (2011) and it is not compatible with the bootstrap restart algorithm.
`K`	the number of quantiles to supply as starting values for the breakpoints when `NA` is specified in the `psi` argument of `segmented`. `K` is ignored when `psi` is different from `NA`.
`last`	logical indicating if output should include only the last fitted model.
`maxit.glm`	integer giving the maximum number of inner IWLS iterations (see details).

`h`	positive factor (from zero to one) modifying the increments in breakpoint updates during the estimation process (see details).
`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`.
`size.boot`	the size of the bootstrap samples. If `NULL`, it is taken equal to the actual sample size.
`gap`	logical, if `FALSE` the gap coefficients are *always* constrained to zero at the convergence.
`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.
`powers`	The powers of the pseudo covariates employed by the algorithm. These are possibly altered during the iterative process to stabilize the estimation procedure. Usually of no interest for the user.
`seed`	The seed to be passed on `set.seed()` when `n.boot>0`. Setting the seed can be useful to replicate the results when the bootstrap restart algorithm is employed.

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` quantiles are selected as starting values for the breakpoints. In this case, it may be useful to set also `stop.if.error=FALSE` to automize the procedure, see Muggeo and Adelfio (2011).

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.

Sometimes to stabilize the procedure, it can be useful to set `h<1` to reduce the increments in the breakpoint estimation. At each iteration the updated estimate is usually given by `psi.new=psi.old+increm`. By setting `h<1` (actually `min(abs(h),1)` is considered) causes the following alterations to the algorithm: (i) the actual maximum number of iterations is increased up to `it.max+round(it.max/2)` and (ii) the breakpoint update is computed via `psi.new=psi.old+h*increm` after the `it.max`-th iteration.

Since version 0.2-9.0 `segmented` implements the bootstrap restarting algorithm described in Wood (2011). 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 `toll` 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. 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)

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)
```

Arguments

`y`	vector of observations of length `n`.
`XREG`	design matrix for standard linear terms.
`Z`	appropriate matrix including the segmented variables whose breakpoints have to be estimated.
`PSI`	appropriate matrix including the starting values of the breakpoints to be estimated.
`w`	possibe weights vector.
`offs`	possibe offset vector.
`opz`	a list including information useful for model fitting.
`n.boot`	the number of bootstrap samples employed in the bootstrap restart algorithm.
`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.

Details

The functions call iteratively `lm.wfit` (or `glm.fit`) with proper design matrix depending on `XREG`, `Z` and `PSI`. `seg.lm.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, control = seg.control(),
    model = TRUE, ...)

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

## S3 method for class 'glm'
segmented(obj, seg.Z, psi, control = seg.control(),
    model = TRUE, ...)
```

Arguments

`obj`	standard 'linear' model of class "lm" or "glm".
`seg.Z`	a formula with no response variable, such as `seg.Z=~x1+x2`, indicating the (continuous) explanatory variables having segmented relationships with the response. Currently, formulas involving functions, such as `seg.Z=~log(x1)` or `seg.Z=~sqrt(x1)`, or selection operators, such as `seg.Z=~d[,"x1"]` or `seg.Z=~d$x1`, are *not* allowed.
`psi`	named list of vectors. The names have to match the variables of the `seg.Z` argument. Each vector includes starting values for the break-point(s) for the corresponding variable in `seg.Z`. If `seg.Z` includes only a variable, `psi` may be a numeric vector. A `NA` value means that 'K' quantiles are used as starting values, where `K` is fixed via the seg.control auxiliary function.
`control`	a list of parameters for controlling the fitting process. See the documentation for seg.control for details.
`model`	logical value indicating if the model.frame should be returned.
`...`	optional arguments.

Details

Given a linear regression model (of class "lm" or "glm"), segmented tries to estimate a new 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. 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.

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:

`psi`	estimated break-points and relevant (approximate) standard errors
`it`	number of iterations employed
`epsilon`	difference in the objective function when the algorithm stops
`model`	the model frame
`psi.history`	A list or a vector including the breakpoint estimates at each step

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. Moreover it is also advisable to look at the coefficients of the 'gap' variables. At the convergence they should be small or at least, non-significantly different from zero. `summary.segmented` and `print.summary.segmented` return information of the 'gap' coefficients. The bootstrap restarting option is to deal with poor clear cut relationships.

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 object, the name of the difference-in-slopes parameter is labelled with 'U.nameOfVariable'.
3. Methods specific to the class "segmented" are
   - print.segmented
   - summary.segmented
   - print.summary.segmented
   - plot.segmented
   - lines.segmented
   - confint.segmented
   - vcov.segmented

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

Author(s)

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

References

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

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

See Also

lm, glm

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)
o<-segmented(out.lm,seg.Z=~x,psi=list(x=c(30,60)),
    control=seg.control(display=FALSE))
slope(o)

out.lm<-lm(y~z,data=dati)
o1<-update(o,seg.Z=~x+z,psi=list(x=c(30,60),z=.3))

#automatic procedure to estimate breakpoints in the covariate x
# Notice: bootstrap restart is not allowed!
o<-segmented.lm(out.lm,seg.Z=~x+z,psi=list(x=NA,z=.3),
    control=seg.control(stop.if.error=FALSE,n.boot=0))
```

`segmented.default` *Default method for the generic segmented*

Description

`segmented` is a generic function, and `segmented.default` its default method.

Usage

```
## Default S3 method:
segmented(obj, seg.Z, psi, control = seg.control(),
    model = TRUE, ...)
```

Arguments

`obj`	see `segmented`
`seg.Z`	see `segmented`
`psi`	see `segmented`
`control`	see `segmented`
`model`	see `segmented`
`...`	see `segmented`

Details

Actually `segmented.default` makes nothing! Use the specific methods.

See Also

`segmented`

Examples

```
##Does not work!
# segmented.default(obj.glm,..)
```

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, var.diff=FALSE,
  APC=FALSE)
```

Arguments

`ogg`	an object of class "segmented", returned by any `segmented` method.
`parm`	the segmented variable whose slopes have to be computed. If missing all the segmented variables are considered.
`conf.level`	the confidence level required.
`rev.sgn`	vector of logicals. The length should be equal to the length of `parm`, but it is recycled otherwise. when `TRUE` it is assumed that the current `parm` is 'minus' the actual segmented variable, therefore the sign is reversed before printing. This is useful when a null-constraint has been set on the last slope.
`var.diff`	logical. If `var.diff=TRUE` and there is a single segmented variable, the computed standard errors are based on a sandwich-type formula of the covariance matrix. See Details in `summary.segmented`.
`APC`	logical. If `APC=TRUE` the 'annual percent changes', i.e. $100 \times (\exp(\beta) - 1)$, are computed for each interval (β is the slope). Only point estimates and confidence intervals are returned.

Details

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

Value

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

Note

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

Author(s)

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

References

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

See Also

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

Examples

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

---

stagnant *Stagnant band height data*

Description

The stagnant data frame has 28 rows and 2 columns.

Usage

```
data(stagnant)
```

Format

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

`x` log of flow rate in g/cm sec.

`y` log of band height in cm

Details

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

Source

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

Originally from the PhD thesis by R.A. Cook

Examples

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

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

Description

summary method for class `segmented`.

Usage

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

## 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.

`x`	a `summary.segmented` object produced by `summary.segmented()`.
`digits`	controls number of digits printed in output.
`signif.stars`	logical, should stars be printed on summary tables of coefficients?
`...`	further arguments.

Details

If short=TRUE only coefficients of the segmented relationships are printed. If var.diff=TRUE and there is only one segmented variable, different error variances are computed in the intervals defined by the estimated breakpoints of the segmented variable. For the jth interval with nj observations the error variance is estimated via $RSS_j/(n_j-p)$, where RSS_j is the residual sum of squares in interval jth, and p are the model parameters. Note `var.diff=TRUE` does *not* affect the parameter estimation which is performed via ordinary (and not weighted) least squares. However 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 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
```

---

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, ...)
```

Arguments

`object`	a fitted model object of class "segmented", returned by any `segmented` method.
`var.diff`	logical. If `var.diff=TRUE` and there is a single segmented variable, the covariance matrix is computed using a sandwich-type formula. See Details in `summary.segmented`.
`...`	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.

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)
```

Index