

Package ‘segmented’

April 19, 2009

Type Package

Title Segmented relationships in regression models

Version 0.2-4

Date 2008-01-21

Author Vito M.R. Muggeo <vmuggeo@dssm.unipa.it>

Maintainer Vito M.R. Muggeo <vmuggeo@dssm.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 2008-01-21 19:41:54

R topics documented:

segmented-package	2
broken.line	2
confint.segmented	3
davies.test	5
down	6
draw.history	7
lines.segmented	8
plant	9
plot.segmented	10
print.segmented	11
seg.control	12
segmented	13
segmented.default	15
slope	16
stagnant	18
summary.segmented	18

Index**20**

 segmented-package *Segmented relationships in regression models*

Description

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

Details

Package: segmented
 Type: Package
 Version: 0.2-4
 Date: 2008-01-21
 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.

Author(s)

Vito M.R. Muggeo <vmuggeo@dssm.unipa.it> (thanks to Marco Enea for his partial support)

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.

 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)
```

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. Currently unimplemented.
linkinv	Should the predictions be computed on the scale of the link function? Default to FALSE.

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))
plot(z,y)
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,
        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.
digits	controls the number of digits to print when printing the output.
...	additional parameters

Details

Currently `confint.segmented` computes confidence limits using standard errors coming from the Delta method for the ratio of two random variables of the estimated. This value is a better approximation of 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	<i>Testing for a change in the slope</i>
-------------	--

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(ogg, term, k = 10, alternative = c("two.sided", "less", "greater"))
```

Arguments

<code>ogg</code>	a fitted model returned by <code>glm</code> or <code>lm</code> .
<code>term</code>	a character string to mean the segmented variable being tested.
<code>k</code>	number of points where the test should be evaluated. See details.
<code>alternative</code>	a character string specifying the alternative hypothesis.

Details

`davies.test` tests for a non zero difference-in-slope parameter of a segmented relationship. Roughly speaking, the procedure computes `k` 'naive' (i.e. assuming fixed and known the break-point) 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 the quantiles of the variable `term`.

Value

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

<code>method</code>	title (character)
<code>data.name</code>	the regression model and the segmented variable being tested
<code>statistic</code>	the point at which the maximum (or the minimum if <code>alternative="less"</code>) occurs
<code>parameter</code>	number of evaluation points
<code>p.value</code>	the adjusted p-value

Warning

Currently `davies.test` does not work if the fitted model `ogg` has been built without the argument `data`.

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

```
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")
```

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	<i>History for the breakpoint estimates</i>
--------------	---

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 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()` 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)
```

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, 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)
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

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 $h = (usr[4] - usr[3]) / \text{abs}(k)$ 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

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

plot.segmented *Plot method for segmented objects*

Description

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

Usage

```
## S3 method for class 'segmented':
plot(x, term = NULL, se = FALSE, const = coef(x)[ "(Intercept) " ],
     add = FALSE, linkinv = FALSE, show.gap=TRUE, rev.sgn=FALSE, n.points = 10, ...)
```

Arguments

<code>x</code>	a fitted segmented object
<code>term</code>	the segmented variable whose piece-wise relationship has to be plotted. If there is a single segmented variable, <code>term</code> can be omitted
<code>se</code>	when TRUE pointwise confidence intervals are drawn. Currently unimplemented.
<code>const</code>	constant to add to each fitted segmented relationship (on the scale of the linear predictor) before plotting
<code>add</code>	when TRUE the fitted lines are added to the current device
<code>linkinv</code>	when TRUE, the fitted lines are (possibly) transformed on the inverse link scale before plotting; in this case it could be useful to increase <code>n.points</code> to get smooth curves
<code>show.gap</code>	should the gap between the fitted lines at the estimated breakpoints to be shown?
<code>rev.sgn</code>	when TRUE it is assumed that current <code>term</code> 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
<code>n.points</code>	number of points where the fitted lines have to be computed
<code>...</code>	other graphics parameters to pass on to plotting commands

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`: in this case the new fitted values are re-computed by means of a linear model fit. Note that, as hiding could lead to a wrong impression of the fit, `show.gap=TRUE` is suggested only when the gap coefficients are nonsignificant.

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)
plot(o.seg,add=TRUE,linkinv=TRUE,lwd=2,col=2)
lines(o.seg,col=2,pch=19,bottom=FALSE,lwd=2)
```

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

Description

Printing the most important features of a segmented model.

Usage

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

Arguments

<code>x</code>	object of class <code>segmented</code>
<code>digits</code>	number of digits to be printed
<code>...</code>	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 = 20, display = FALSE, last = TRUE,
           maxit.glm = 25, h = 1)
```

Arguments

<code>toll</code>	positive convergence tolerance.
<code>it.max</code>	integer giving the maximal number of iterations.
<code>display</code>	logical indicating if output should be produced for each iteration.
<code>last</code>	logical indicating if output should include only the last fitted model.
<code>maxit.glm</code>	integer giving the maximum number of inner IWLS iterations (see details).
<code>h</code>	positive factor (from zero to one) modifying the increments in breakpoint estimation (see details).

Details

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

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+increment`. 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*increment` after the `it.max`th iteration.

Value

A list with the arguments as components.

Examples

```
#decrease the maximum number inner iterations and display the
#evolution of the (outer) iterations
seg.control(display = TRUE, maxit.glm=4)
```

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.frame = TRUE, ...)

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

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

Arguments

<code>obj</code>	standard 'linear' model of class "lm" or "glm".
<code>seg.Z</code>	a formula with no response variable, indicating the (continuous) explanatory variables having segmented relationships with the response
<code>psi</code>	named list of vectors. The names have to match the variables of the <code>seg.Z</code> argument. Each vector includes starting values for the break-point(s) for the corresponding variable in <code>seg.Z</code> . If <code>seg.Z</code> includes only a variable, <code>psi</code> may be a numeric vector.
<code>control</code>	a list of parameters for controlling the fitting process. See the documentation for seg.control for details.
<code>model.frame</code>	logical value indicating if the model.frame should be returned.
<code>...</code>	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.

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>mframe</code>	the model frame
<code>psi.history</code>	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.

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.name_of_variable'.
3. Methods specific to the class "segmented" are
 - `print.segmented`
 - `summary.segmented`
 - `print.summary.segmented`

- `plot.segmented`
- `lines.segmented`
- `confint.segmented`

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

Author(s)

Vito M. R. Muggeo, vmuggeo@dssm.unipa.it

References

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

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<-## S3 method for class 'lm':
segmented(out.lm,seg.Z=~x,psi=list(x=c(30,60)),
          control=seg.control(display=FALSE))
slope(o)

out.lm1<-lm(y~z,data=dati)
o1<-update(o,seg.Z=~x+z,psi=list(x=c(30,60),z=.3))
```

`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.frame = TRUE, ...)
```

Arguments

obj see [segmented](#)
 seg.z see [segmented](#)
 psi see [segmented](#)
 control see [segmented](#)
 model.frame 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 slopes of each 'segmented' relationship in the fitted model.

Usage

```
slope(ogg, parm, conf.level = 0.95, rev.sgn=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.

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, vmuggeo@dssm.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)
```

stagnant	<i>Stagnant band height data</i>
----------	----------------------------------

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, ...)
```

Arguments

object	Object of class "segmented"
short	logical indicating if the 'short' summary should be printed
...	further arguments

Details

If short=TRUE only coefficients of the segmented relationships are printed.

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 <code>coefficients</code> returned by <code>summary.lm</code> or <code>summary.glm</code> , 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

Author(s)

Vito M.R. Muggeo

See Also

[print.segmented](#), [davies.test](#)

Examples

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

Index

*Topic **datasets**

- down, [6](#)
- plant, [8](#)
- stagnant, [17](#)

*Topic **htest**

- davies.test, [4](#)
- slope, [16](#)

*Topic **models**

- print.segmented, [10](#)

*Topic **nonlinear**

- broken.line, [2](#)
- confint.segmented, [3](#)
- draw.history, [6](#)
- lines.segmented, [7](#)
- plot.segmented, [9](#)
- segmented, [12](#)
- segmented-package, [1](#)

*Topic **regression**

- broken.line, [2](#)
- confint.segmented, [3](#)
- draw.history, [6](#)
- lines.segmented, [7](#)
- plot.segmented, [9](#)
- seg.control, [11](#)
- segmented, [12](#)
- segmented-package, [1](#)
- segmented.default, [15](#)
- slope, [16](#)
- summary.segmented, [18](#)

broken.line, [2](#)

confint.segmented, [3](#)

davies.test, [4](#), [17](#), [18](#)

down, [6](#)

draw.history, [6](#)

glm, [14](#)

lines.segmented, [7](#), [10](#)

lm, [14](#)

plant, [8](#)

plot.segmented, [8](#), [9](#)

points, [8](#)

predict.glm, [3](#)

print.segmented, [10](#), [18](#)

print.summary.segmented, [11](#), [14](#)

print.summary.segmented

(*summary.segmented*), [18](#)

seg.control, [11](#), [13](#)

segmented, [3](#), [4](#), [12](#), [15](#)

segmented-package, [1](#)

segmented.default, [15](#)

segments, [8](#)

slope, [16](#)

stagnant, [17](#)

summary.segmented, [11](#), [14](#), [18](#)