Package 'bentcableAR'

July 2, 2014

Type Package

Title Bent-Cable Regression for Independent Data or Autoregressive Time Series

Version 0.2.3

Date 2010-09-20

Author Grace Chiu <bentcable@gmail.com>, CSIRO Mathematics,Informatics and Statistics, GPO Box 664, Canberra, ACT 2601,Australia

Maintainer author <bentcable@gmail.com>

Description This package contains two main interfaces for fitting and diagnosing bent-cable regressions for autoregressive time-series data or independent data (time series or otherwise). The interfaces are 'bentcable.ar()' and 'bentcable.dev.plot()'. Some components in the package can also be used as stand-alone functions. The bent cable (linear-quadratic-linear) generalizes the broken stick (linear-linear), which is also handled by this package. Version 0.2 corrects a glitch in the computation of confidence intervals for the CTP. References updated from Versions 0.2.1 and 0.2.2 appear in Version 0.2.3.

License GPL (>= 3)

Repository CRAN

Date/Publication 2012-10-29 08:58:16

NeedsCompilation no

R topics documented:

`bentcable.ar` *Bent-Cable Regression for Independent and Autoregressive Data*

Description

These two functions are the main interfaces in the `bentcableAR` package. They perform bent-cable (including broken-stick) regression to AR(p) time-series data or independent data (time-series or otherwise) and produce diagnostic plots. Confidence intervals for the *critical time point* (CTP) are included in some cases.

Usage

```
bentcable.ar(y.vect, tgdev = NULL, p = 0, stick = FALSE, t.vect = NULL,
init.cable = NULL, init.phi = NULL, tol = 1e-04,
method0 = "css", method1 = "yw", ci.level = 0.95,
main = NULL)
bentcable.dev.plot(tau.vect, gamma.vect = NULL, y.vect, t.vect = NULL,
stick = FALSE, p = 0)
```

Arguments

`y.vect`	A numeric vector of response data.
`t.vect`	A numeric vector of design points, which **MUST** be equidistant with unit increments if p>0 is assumed. They need not be equidistant for independent data. Specifying `t.vect=NULL` is equivalent to specifying the default time points `c(0,1,2,...)`. Also see **Warnings** below.
`tau.vect, gamma.vect`	Numeric vectors specifying a (τ, γ)-grid over which the bent-cable profile deviance surface / function is to be evaluated. If `stick=TRUE`, then `gamma.vect` is overwritten by `c(0)` in `bentcable.dev.plot`.
`tgdev`	A `bentcable.dev.plot` object. An error results if this is supplied together with `init.cable` or `init.phi`.
`p`	The autoregressive order (non-negative integer). `p=0` specifies independent data that may or may not be from a time series context.

`stick` A logical value; if `TRUE` then a broken stick (i.e. bent cable with γ=0.) is fitted. Also see `gamma.vect` above.

`init.cable` A numeric vector of initial values for the bent-cable parameters. If `stick=FALSE`, then `init.cable` should have the form `c(b0,b1,b2,tau,gamma,...)`. If `stick=TRUE`, then `init.cable` should have the form `c(b0,b1,b2,tau,...)`. In either case, `...` will be ignored. An error results if this is supplied together with `tgdev`.

`init.phi` A numeric vector of initial values for the AR coefficients. If not provided, then a default value is assigned, consisting of the first `p` elements of the vector `c(0.5,-0.5,0.5,-0.5,...)`. When provided and its dimension does not match `p`, then the function determines which to reject depending on the situation, and reports its decision in the screen output. An error results if this is supplied together with `tgdev`.

`tol` Tolerance for determining convergence.

`method0, method1`
The fitting method when p>0. `"css"` stands for *conditional sum-of-squares* and corresponds to conditional maximum likelihood. `"yw"` stands for *Yule-Walker*, and `"mle"` for (full) *maximum likelihood estimation*. If `method0` fails to converge, then `method1` is attempted.

`ci.level` A numeric value between 0 and 1, exclusive. Used to compute the CTP confidence interval when `p` is greater than 0. See `cable.change.conf` and **Warnings** below.

`main` A title for the set of diagnostic plots.

Details

`bentcable.dev.plot` involves bent-cable regression assuming a known transition. It plots a profile deviance surface over a fixed grid (see **References**). It also returns the grid and the profile deviance surface matrix, which can be used to generate initial values for an overall bent-cable regression (no known parameters).

`bentcable.ar` is used mainly for overall bent-cable regression, with one exception. Different scenarios determine the behaviour of `bentcable.ar`, as follows.

(1) Independent data and `tgdev` is supplied. In this case, `bentcable.ar` calls `cable.ar.0.fit` which identifies the best grid-based fit from `tgdev`, then feeds it through an internal engine `cable.ar.p.iter` or `stick.ar.0` that performs overall bent-cable regression. This best fit is returned but not plotted, and the autocorrelation is diagnosed (even for non-time-series data) by a PACF plot and a suggested value of `p` based on the AIC (see `ar`). As stated in the screen output, these diagnostics should be used only for time-series data, where the returned best AR(0) estimates are intended to be supplied as `init.cable` in a subsequent call of `bentcable.ar` for an AR(p>0) fit. To produce a plot of the returned best AR(0) fit and/or the corresponding CTP confidence interval, the user can supply the returned parameter estimates as `init.cable` in another call of `bentcable.ar` with `p=0` (see Scenario (3)).

(2) AR(p>0) data and `tgdev` is supplied. In this case, no graphics are produced; `bentcable.ar` simply locates the highest point on the grid-based profile deviance surface and returns the corresponding (crude) parameter estimates to be used as `init.cable` and `init.phi` in subsequent overall bent-cable fits. If multiple peaks exist (such as along a ridge), then only that at the smallest τ and smallest γ is used.

(3) Independent data (time series or otherwise) and `init.cable` are supplied. In this case, `bentcable.ar` performs overall bent-cable regression and produces a scatterplot of the data superimposed with the best fit and estimated transition. For time series data where the CTP is applicable (see **Warnings**), the CTP confidence interval is additionally computed and superimposed in blue. No other plots are produced. Since `init.cable` is supposed to have come from a reasonable source (such as grid-based), this fit is not intended to be fed to another round of `bentcable.ar`, except when the user wishes to explore using a positive p (but this should be performed in conjunction with another round of grid-based approach in Scenario (2)).

(4) AR(p>0) data and `init.cable` are supplied. In this case, `bentcacble.ar` computes the overall bent-cable fit and CTP confidence interval (see `cable.change.conf`). Also included are the following diagnostics: a scatterplot of the data superimposed with the best fit and estimated transition $(\tau-\gamma, \tau, \tau+\gamma)$ (in red) and the CTP confidence interval (in blue, if it exists - see **Warnings**), and ACF and PACF plots for the fitted residuals and innovations (see `cable.ar.p.resid` for their difference). Since `init.cable` is supposed to have come from a reasonable source (such as grid-based), this fit is not intended to be fed to another round of `bentcable.ar`, except when the user wishes to explore using an alternative p (but this should be performed in conjunction with another round of grid-based approach in Scenario (1) or (2)), or when the `"css"` algorithm fails to converge but the SSE value is desired (see **Details**).

Below is a summary of the bent-cable regression methodology, and how one may apply it by using the `bentcableAR` package.

The *bent cable* is a linear-quadratic-linear function, where the quadratic bend is regarded as the transition from the incoming linear phase to the outgoing linear phase. A bent cable has the form $f(t) = b_0 + b_1 t + b_2 q(t)$, where $q(t)$ is the *basic bent cable* with incoming slope 0 and outgoing slope 1, and a quadratic bend that is centred at τ with half-width $\gamma \geq 0$:

$$q(t) = \frac{(t-\tau+\gamma)^2}{4\gamma} I\{|t-\tau| \leq \gamma\} + (t-\tau) I\{t > \tau+\gamma\}.$$

The *broken stick* is a special bent cable with no quadratic bend (i.e. γ=0). The term *bent-cable regression* implicitly includes *broken-stick regression*.

For independent data (time series or otherwise), bent-cable regression by maximum likelihood is performed via nonlinear least-squares estimation of $\theta = (b_0, b_1, b_2, \tau, \gamma)$. For AR(p) data, the AR coefficients are $\phi = (\phi_1, \phi_2, \ldots, \phi_p)$, and conditional maximum likelihood (CML) estimation of (θ, ϕ) (conditioned on the first p data points) is performed by nonlinear conditional least squares (i.e. minimizing the conditional sum-of-squares error (SSE)). In this time-series context, time points are assumed to be equidistant with unit increments.

Minimization of the (conditional) SSE is specified as `"css"` by default for `method0`. However, `"css"` sometimes fails to converge, or the resulting ϕ estimate sometimes corresponds to non-stationarity. In this case, the alternative estimation approach specified for `method1` is attempted. `"mle"` specifies the *CML-ML hybrid* algorithm, and `"yw"` the *CML-ML-MM hybrid* algorithm (*MM* stands for *method of moments*; see **References**.) Both `"yw"` and `"mle"` guarantee stationarity, but often take much longer than `"css"` to converge.

Due to nonlinearity, initial values must be supplied for proper parameter estimation. Also, bent-cable regression is a notoriously irregular estimation problem (due to low-order differentiability), and the estimation algorithms (mainly the built-in R functions `nls` and `optim`) may fail to converge from initial values that are unrefined guesses of the parameters. When this happens, the user is advised to generate an initial value from a grid-based procedure.

The grid-based procedure involves specifying a (τ, γ)-grid over which the bent-cable profile deviance surface is evaluated and plotted, such as by `bentcable.dev.plot`. At each grid point, the transition is fixed, and bent-cable regression involves only linear parameters b_0, b_1, b_2 and AR coefficients ϕ, all of which can be estimated using standard time-series algorithms (mainly the built-in R functions `ar` and `arima`). Regression at each grid point yields a point on the profile deviance surface. The grid point at which the profile deviance is maximum corresponds to a bent-cable fit (given a known transition) that is best among the specified grid points. Thus, for a high-resolution grid, this *best grid point* together with the corresponding estimates of b_0, b_1, b_2 and ϕ may be regarded as the ML or CML estimate for the model. However, high-resolution grid-based estimation may be computationally infeasible. Instead, the *best grid point* on a coarser grid can give good initial values for the true ML or CML estimate that is trapped between grid points.

However, the *true* ML or CML estimate may not easily come by even with good initial values. Irregularity of bent-cable regression often manifests itself in the form of multiple peaks on the deviance surface. Thus, the user should be aware of different local maxima on which the optimization algorithm can converge despite initial values for θ that are very similar. The user is advised to combine several exploratory analyses as well as model diagnoses before settling on a *best* fit.

For example, one may first fix p=0 as the AR order, then use `bentcable.dev.plot` to conduct a visual inspection of the profile deviance surface over a fine (τ, γ)-grid. This is to identify the neighbourhood of the global maximum for p=0. If necessary, one can *zoom in* to this neighbourhood by placing over it an even finer grid to hone the grid-based approximation. The resulting `bentcable.dev.plot` object can then be fed to `bentcable.ar` to produce a best overall fit for the AR(0) assumption in that neighbourhood. If p=0 is deemed inadequate based on the `bentcable.ar` diagnostics, then the regression must now be repeated for a newly chosen p. Since the bent-cable parameter estimates will differ for different values of p, the earlier AR(0) estimates may or may not be good initial values for this new AR(p) fit. The user is advised to try several additional initial values, possibly repeating the grid-based procedure, but this time using the new p. To further screen out local maxima, the SSE values for these AR(p) fits (with common p) should be compared. For a `"css"` fit, the SSE is stored in `$cable$ar.p.fit$value` of the returned object. The SSE is not directly retrievable for a `"yw"` or `"mle"` fit, but the user can apply the estimates returned in `$cable$est` as the initial values to a subsequent `"yw"` fit, and the SSE will appear in the screen output as *initial value* while the `"css"` algorithm iterates.

As with any numerical optimization procedure, there is no guarantee that the fit observed to have the smallest SSE value indeed corresponds to the global maximum.

Value

`cable` An object that is compatible with a `cable.ar.p.iter` object. Returned by `bentcable.ar` in Scenarios (3) and (4). Note the different components of `cable` depending on the scenario. See `cable.ar.p.iter` and `stick.ar.0`.

`ctp` A `cable.change.conf` object, if the CTP is successfully estimated; returned by `bentcable.ar` in Scenarios (3) and (4). This object has three components: the CTP estimate, its estimated asymptotic variance, and the corresponding Wald confidence interval.

`fit` Returned by `bentcable.ar` in Scenarios (1) and (2). In (1), the returned `bentcable.ar` object is a `cable.ar.0.fit` object (largely compatible with `cable.ar.p.iter` objects); thus, `fit` is an `nls` object containing the overall independent-data bent-cable fit. In (2), the returned `bentcable.ar` object is a `cable.fit.known.change`

object; thus, `fit` is an `arima` object containing the AR(p>0) bent-cable fit at the known transition grid point. In either scenario, `fit` is intended to be fed through another round of `bentcable.ar` for subsequent overall AR(p>0) fits.

`init` Returned by `bentcable.ar` in Scenario (2). It is the vector of parameter estimates extracted from `fit` and intended to be used as initial values in subsequent calls to `bentcable.ar` for overall bent-cable regression.

`y, t, n, p, stick` Returned by `bentcable.ar` explicitly in Scenario (1) (but embedded in `cable` of Scenarios (3) and (4)). They are `y.vect`, `t.vect`, `n`, `p`, and `stick` as supplied by the user.

`dev, tau, gamma` Returned by `bentcable.dev.plot`. Note that `dev` is a `cable.dev` object, i.e. a matrix of profile deviance values evaluated at the grid specified by `tau` and `gamma`.

Warnings

For time-series data, `t.vect` *MUST* be equidistant with unit increments; otherwise, these functions will return meaningless values. (For independent data, `t.vect` can be non-equidistant.)

Computations for the CTP estimate and confidence interval are based on a time vector of the form `c(0,1,2,...)`. For any other form for the time vector, the CTP will not be computed, and on-screen warnings will appear. To ensure compatibility between the model fit and CTP estimates, the user is advised to fit the model using the default time vector. Then, if necessary, the user may transform the results to the preferred time scale after the model and CTP estimates have been produced.

The above computational issue implies that the CTP cannot be computed for non-time-series data. **Rationale**: In a non-time-series context design points are often non-equidistant, and the cable's slope often never changes sign; even with a sign change, the point at which this takes place may be less interpretable. In such a context, the user is advised to rely on confidence regions for (τ, γ) (see **References**).

Note

The major engines for `bentcable.dev.plot` are `cable.dev` and `cable.fit.known.change`. The computational engines for `bentcable.ar` are `cable.ar.p.iter`, `cable.ar.0.fit`, `stick.ar.0`, and `cable.change.conf`, while the plotting engine is `cable.ar.p.diag`. Although these and other *lesser* functions are called internally by the two main interfaces described here, they can be used as stand-alone functions, and the user is advised to refer to their documentation. Type `library(help="bentcableAR")` for a full list of available functions.

Author(s)

Grace Chiu

References

Chiu, G.S. and Lockhart, R.A. (2010), Bent-Cable Regression with Autoregressive Noise, *Canadian Journal of Statistics*, **38**, 386–407. http://faculty.washington.edu/gchiu/Articles/bentcable-cjs.pdf

Chiu, G., Lockhart, R. and Routledge, R. (2006), Bent-Cable Regression Theory and Applications, *Journal of the American Statistical Association*, **101**, 542–553. http://faculty.washington.edu/gchiu/Articles/bentcable-jasa.pdf

See Also

cable.lines, lm, nls, optim, ar, arima, plot, par, contour, persp

Examples

```
data(stagnant)
data(sockeye)

# Scenario (1)
###############

# independent non-time-series cable:
bentcable.dev.plot( seq(-1,1,length=20),
seq(.1,1,length=20), stagnant$loght, stagnant$logflow )

# zoom in to global max
dev0 <- bentcable.dev.plot( seq(-.04,.16,length=20),
seq(.2,.65,length=20), stagnant$loght, stagnant$logflow )
# locally smooth deviance surface

cable <- bentcable.ar( stagnant$loght, tgdev=dev0, t.vect=stagnant$logflow )
# ignore time-series diagnostics
# local regularity - expect to be true best fit
# SSE=0.005
# feed 'cable' in Scenario (3) to get fitted plot:
# bentcable.ar( cable$y, init.cable=coef(cable$fit),
# t.vect=cable$t )


# AR(0) stick, start time at 80:
dev0 <- bentcable.dev.plot( seq(85,97,length=15), 0,
sockeye$logReturns, sockeye$year, TRUE )  # obvious global max
stick0 <- bentcable.ar( sockeye$logReturns, tgdev=dev0, stick=TRUE,
t.vect=sockeye$year )
# local regularity - should be true best fit
# SSE=8.85
# diagnostics: take p=0 to 4 ??

# AR(0) cable, start at time 0:
bentcable.dev.plot( seq(1,20,length=25),
seq(.1,15,length=25), sockeye$logReturns )

# zoom in to global max
dev0 <- bentcable.dev.plot( seq(10,15,length=25),
seq(2,10,length=20), sockeye$logReturns )
# surface has ridge - expect some trouble locating true peak
```

```
cable0 <- bentcable.ar( sockeye$logReturns, tgdev=dev0 )
# apparent best AR(0) fit: SSE=8.68
# diagnostics: take p=2 to 6

# compare to this:
# dev1 <- bentcable.dev.plot( seq(10,15,length=25),
# seq(2,10,length=15), sockeye$logReturns )
# bentcable.ar( sockeye$logReturns, tgdev=dev1 ) # SSE=8.683
# # not an obvious local max!

# feed 'cable0' in Scenario (3) to get fitted plot:
# bentcable.ar( cable0$y, init.cable=coef(cable0$fit) )




# Scenario (2)
###############

# AR(2) cable, start time at 0:
bentcable.dev.plot( seq(6,18,length=15),
seq(.01,12,length=15), sockeye$logReturns, p=2 )

# zoom in to global max
dev2 <- bentcable.dev.plot( seq(10,12,length=15),
seq(1,5,length=15), sockeye$logReturns, p=2 )

# best grid-based fit
gr.cable2 <- bentcable.ar( sockeye$logReturns, tgdev=dev2, p=2 )
# to be used in Scenario (4)
# local regularity - expect little trouble

# AR(2) stick, start time at 80:
bentcable.dev.plot( seq(86,98,length=15), y.vect=sockeye$logReturns,
p=2, stick=TRUE, t.vect=sockeye$year )

# zoom in to global max
dev3 <- bentcable.dev.plot( seq(88.5,93,length=25),
y.vect=sockeye$logReturns,
p=2, stick=TRUE, t.vect=sockeye$year )
# camel hump - double peaks!

# best grid-based fit
gr.stick2 <- bentcable.ar( sockeye$logReturns, tgdev=dev3, p=2, stick=TRUE,
t.vect=sockeye$year )
# irregularity - expect some trouble if used in Scenario (4)

# AR(4) cable, start time at 0:
bentcable.dev.plot( seq(6,18,length=15), seq(.01,12,length=15),
sockeye$logReturns, p=4 )

# zoom in to global max
dev4 <- bentcable.dev.plot( seq(10,12,length=15),
```

```
seq(1,7,length=25), sockeye$logReturns, p=4 )
# slight ridge

# best grid-based fit
gr.cable4 <- bentcable.ar( sockeye$logReturns, tgdev=dev4, p=4 )
# to be used in Scenario (4)
# will ridge be problem???



# Scenario (3)
###############

# independent non-time-series cable:
bentcable.ar( stagnant$loght, t.vect=stagnant$logflow,
init.cable=c(.6,-.4,-.7,0,.5) )   # SSE=0.005
# identical to 'cable' in Scenario (1)
# no irregularity, no ambiguity!



# AR(0) stick, start time at 80:
bentcable.ar( sockeye$logReturns, init.cable=c(10,.1,-.5,90),
stick=TRUE, t.vect=sockeye$year )
# identical to 'stick0' in Scenario (1)
# local regularity, no trouble


# AR(0) stick, start time at 0:
bentcable.ar( sockeye$logReturns, init.cable=coef(cable0$fit)[1:5],
stick=TRUE )
# identical to 'cable0' in Scenario (1)
# here you get plot of fit and CTP confidence interval

# Scenario (4)
###############

# AR(2) cable, start time at 0:
# use 'gr.cable2' from Scenario (2)
cable2 <- bentcable.ar( sockeye$logReturns,
init.cable=gr.cable2$init[1:5], init.phi=gr.cable2$init[-c(1:5)] )
# "css" successful
# best AR(2) fit, SSE=4.868

# compare to this:
# bentcable.ar( sockeye$logReturns,
# init.cable=c(13,.1,-.5,11,4), p=2 )
# "css" successful, same SSE, virtually same fit
# recall local regularity from 'dev2'

# AR(2) stick, start time at 80:
# use 'gr.stick2' from Scenario (2)
stick2 <- bentcable.ar( sockeye$logReturns, init.cable=gr.stick2$init[1:4],
```

```
init.phi=gr.stick2$init[-c(1:4)], stick=TRUE, t.vect=sockeye$year )
# "css" successful, best AR(2) fit, SSE=5.0

# compare this to the other peak shown in 'dev3'
# bentcable.ar( sockeye$logReturns,
# init.cable=c(10,0,-.5,91.5), p=2, stick=TRUE,
# t.vect=sockeye$year )
# "css" successful, SSE=5.1, not best fit!



# AR(4) cable, start time at 0:
cable4 <- bentcable.ar( sockeye$logReturns,
init.cable=gr.cable4$init[1:5], init.phi=gr.cable4$init[-c(1:5)] )
# "css" unsuccessful, switched to "yw"
# feed 'cable4' in Scenario (4) to get SSE from screen output:

bentcable.ar( cable4$cable$y, init.cable=cable4$cable$est[1:5],
init.phi=cable4$cable$est[-c(1:5)] )
# SSE=2.47 from screen output
```

`bentcableAR` *The Bent-Cable Regression Package*

Description

Perform bent-cable (including broken-stick) regression to independent data or autoregressive time series.

Details

Package:	bentcableAR
Type:	Package
Version:	0.2.3
Date:	2010-09-20
License:	GPL (>=3)

There are two main interfaces in this package: `bentcable.dev.plot` for plotting profile deviance surfaces, and `bentcable.ar` for fitting and diagnosing the regression. In some cases, confidence intervals for the *change point* are also computed..

Detailed documentation and examples are available on the function help pages.

The major engines for `bentcable.dev.plot` are `cable.dev` and `cable.fit.known.change`. The computational engines for `bentcable.ar` are `cable.ar.p.iter`, `cable.ar.0.fit`, `stick.ar.0`, and `cable.change.conf`, while the plotting engine is `cable.ar.p.diag`. Although these and other *lesser* functions are called internally by the two main interfaces described above, they can be used as stand-alone functions, and the user is advised to refer to their documentation. Type

`library(help="bentcableAR")` for a full list of available functions.

Disclaimer: The package functions and examples have been thoroughly tested in R 2.6.2 installed on the author's two Mac machines running OS X. Results are known to vary depending on machine and platform.

Author(s)

Grace Chiu <bentcable at gmail.com>

Maintainer: Grace Chiu <bentcable at gmail.com>

References

Chiu, G.S. and Lockhart, R.A. (2010), Bent-Cable Regression with Autoregressive Noise, *Canadian Journal of Statistics*, **38**, 386–407. http://faculty.washington.edu/gchiu/Articles/bentcable-cjs.pdf

Chiu, G., Lockhart, R. and Routledge, R. (2006), Bent-Cable Regression Theory and Applications, *Journal of the American Statistical Association*, **101**, 542–553. http://faculty.washington.edu/gchiu/Articles/bentcable-jasa.pdf

---

cable.ar.p.diag *Bent-Cable AR(p>0) Diagnostics*

Description

ACF, PACF, and other plots are produced for diagnosing an AR(p) bent-cable fit when p>0.

Usage

```
cable.ar.p.diag(ar.p.fit, resid.type = "p", xlab = "time", ylab = "",
main = NULL, main.all = NULL, ctp.ci = NULL)
```

Arguments

`ar.p.fit`	A `cable.ar.p.iter` object for AR(p) data, p>0.
`resid.type`	A `type` argument for the `plot` function; used to control the way fitted residuals and innovations are displayed. Default is `"p"` for *points*.
`xlab`	Character string: x-axis label.
`ylab`	Character string: y-axis label.
`main`	Character string: title for the time series plot.
`main.all`	Character string: title for the entire set of plots.
`ctp.ci`	A `cable.change.conf` object.

Details

This function splits the plotting canvas into several panels. For one panel, `ar.p.fit` is fed to `cable.ar.p.plot` that produces a scatterplot of the data and overlays on it the fitted bent cable with the estimated transition. The optioinal `ctp.ci` is also fed to `cable.ar.p.plot` to add the CTP confidence interval to the same panel. Additionally, `ar.p.fit` is fed to `cable.ar.p.resid` to extract the fitted residuals and innovations, which are then plotted in separate panels that again show the estimated transition and confidence interval. Finally, four panels show ACF and PACF diagnostics for the fitted residuals and innovations, via the built-in R functions `acf` and `pacf`.

Warning

See the warnings from `cable.ar.p.plot` and `cable.ar.p.resid`.

Note

This function is intended for internal use by `bentcable.ar`.

Author(s)

Grace Chiu

References

See the `bentcableAR` package references.

See Also

`cable.lines`, `plot`, `par`, `acf`, `pacf`

Examples

```
data(sockeye)

# AR(2) cable fit
fit.ar2 <- cable.ar.p.iter( c(13,.1,-.5,11,4,.5,-.5),
sockeye$logReturns, tol=1e-4 )
cable.ar.p.diag( fit.ar2, main="bent cable", main.all="Sockeye",
ctp.ci=cable.change.conf( fit.ar2, .9 ) )
# compare to this:
# fit.ar2 <- bentcable.ar( sockeye$logReturns,
# init.cable=c(13,.1,-.5,11,4), p=2, main="Sockeye bent cable",
# ci.level=.9 )

# AR(4) stick fit
fit.ar4 <- cable.ar.p.iter( c(13,.1,-.5,11,.5,-.5,.5,-.5),
sockeye$logReturns, tol=1e-4, stick=TRUE )
cable.ar.p.diag( fit.ar4, ctp.ci=cable.change.conf( fit.ar4, .95 ) )
# compare to this:
# fit.ar4 <- bentcable.ar( sockeye$logReturns,
# init.cable=c(13,.1,-.5,11), p=4, stick=TRUE )
```

cable.ar.p.iter *Bent-Cable Regression for Independent or AR Data, With Exception*

Description

This function is the main engine for bentcable.ar. It performs bent-cable (including broken-stick) regression to AR(p) time-series data or independent data (time-series or otherwise). However, it **cannot fit broken sticks to independent data** (see stick.ar.0).

Usage

```
cable.ar.p.iter(init, y.vect, t.vect = NULL, n = NA, tol,
method0 = "css", method1 = "yw", stick = FALSE)
```

Arguments

init	A numeric vector of initial values, in the form of c(b0,b1,b2,tau,gamma,phi.1,...,phi.p) when stick=FALSE, and c(b0,b1,b2,tau,phi.1,...,phi.p) when stick=TRUE. phi values correspond to AR(p) coefficients - if not included, then independent data are assumed.
y.vect	A numeric vector of response data.
t.vect	A numeric vector of design points, which **MUST** be equidistant with unit increments if AR(p) is assumed. They need not be equidistant for independent data. Specifying t.vect=NULL is equivalent to specifying the default time points c(0,1,2,...).
n	Length of response vector (optional).
tol	Tolerance for determining convergence.
method0, method1	The fitting method when p>0. "css" stands for *conditional sum-of-squares* and corresponds to conditional maximum likelihood. "yw" stands for *Yule-Walker*, and "mle" for (full) *maximum likelihood estimation*. If method0 fails to converge, then method1 is attempted.
stick	A logical value; if TRUE, a broken-stick regression is performed.

Details

The *bent cable* has the form $f(t) = b_0 + b_1 t + b_2 q(t)$, where $q(t)$ is the *basic bent cable*

$$q(t) = \frac{(t-\tau+\gamma)^2}{4\gamma} I\{|t-\tau| \le \gamma\} + (t-\tau) I\{t > \tau+\gamma\}$$

for $\gamma \ge 0$.

For independent data (time series or otherwise), bent-cable regression by maximum likelihood is performed via nonlinear least-squares estimation of $\theta = (b_0, b_1, b_2, \tau, \gamma)$ through the built-in R function nls. For AR(p) data, conditional maximum likelihood (CML) estimation of (θ, ϕ) (conditioned on the first p data points) is performed through the built-in R function optim with the "BFGS"

algorithm, where $\phi = (\phi_1, \ldots, \phi_p)$ are the AR coefficients. In either case, the estimation relies on the user-supplied initial values in `init`. A Gaussian model is assumed, so that CML estimation is equivalent to minimizing the conditional sum-of-squares error, specified as `"css"` by default for `method0`. However, `"css"` sometimes fails to converge, or the resulting ϕ estimate sometimes corresponds to non-stationarity. In this case, the alternative estimation approach specified for `method1` is attempted. `"mle"` specifies the *CML-ML hybrid* algorithm, and `"yw"` the *CML-ML-MM hybrid* algorithm (*MM* stands for *method of moments*; see **References**.) Both `"yw"` and `"mle"` guarantee stationarity, but often take much longer than `"css"` to converge.

The bent-cable likelihood / deviance often has multiple peaks. Thus, the user should be aware of different local maxima on which the optimization algorithm can converge despite initial values for θ that are very similar. The user is advised to combine several exploratory analyses as well as model diagnoses before settling on a *best* fit. See **Details** on the `bentcable.ar` help page for a detailed description.

Value

`fit`	An `nls` object, returned if independent data are assumed. It is the maximum likelihood bent-cable fit.
`estimate`	A numeric vector, returned if AR(p>0) is assumed. It is the estimated value of (θ, ϕ).
`ar.p.fit`	Returned if AR(p>0) is assumed. If `"css"` is used, converges, and yields a ϕ estimate that corresponds to stationarity, then `$ar.p.fit` is an `optim` object containing the CML fit. If `"yw"` or `"mle"` is used and converges, then `$ar.p.fit` is an `ar` object containing the CML-ML(-MM) fit.
`y, t, n, p, stick`	As supplied by the user; always returned.
`method`	A character string, returned if AR(p>0) is assumed. It indicates the method that yielded the returned fit.

Note

This function is intended for internal use by `bentcable.ar`.

For several fits that assume a common p, their (conditional) likelihood values should be compared to screen out those that result from local maxima. Equivalently, the (conditional) sum-of-squares error (SSE) can be compared and only the smallest kept. See **Examples** below. Also see **Details** on the `bentcable.ar` help page.

Author(s)

Grace Chiu

References

See the `bentcableAR` package references.

See Also

`stick.ar.0`, `fullcable.t`, `bentcable.dev.plot`, `nls`, `optim`, `ar`.

Examples

```
data(stagnant)
data(sockeye)

# 'stagnant': independent data cable fit
fit0 <- cable.ar.p.iter( c(.6,-.4,-.7,0,.5),
stagnant$loght, stagnant$logflow )    # 'nls' fit
# compare to this:
# bentcable.ar( stagnant$loght, t.vect=stagnant$logflow,
# init.cable=c(.6,-.4,-.7,0,.5) )

fit0$fit   # 'fit0' SSE=0.005


# 'sockeye': AR(2) cable fit
fit1 <- cable.ar.p.iter( c(13,.1,-.5,11,4,.5,-.5),
sockeye$logReturns, tol=1e-4 )    # "css" successful
# compare to this:
# fit1 <- bentcable.ar( sockeye$logReturns,
# init.cable=c(13,.1,-.5,11,4), p=2 )

fit1$ar.p.fit$value     # 'fit1' SSE=4.9


# 'sockeye': AR(2) cable fit
fit2 <- cable.ar.p.iter( c(10,0,0,5,.1,.5,-.5), sockeye$logReturns,
tol=1e-4 )    # "css" unsuccessful, switched to "yw"
# compare to this:
# fit2 <- bentcable.ar(sockeye$logReturns,
# init.cable=c(10,0,0,5,.1), p=2 )

cable.ar.p.iter( fit2$est, sockeye$logReturns,
tol=1e-4 )   # 'fit2' SSE=13.8 (from first line of screen output)


# 'sockeye': AR(4) stick fit
cable.ar.p.iter( c(13,.1,-.5,11,.5,-.5,.5,-.5),
sockeye$logReturns, tol=1e-4, stick=TRUE )
# compare to this:
# bentcable.ar( sockeye$logReturns,
# init.cable=c(13,.1,-.5,11), p=4, stick=TRUE )
```

`cable.ar.p.plot` *Plot Bent Cable AR(p) Fit Over Data*

Description

Plot the bent-cable AR(p) regression.

Usage

```
cable.ar.p.plot(ar.p.fit, xlab = "time", ylab = "", main = NULL, ctp.ci = NULL)
```

Arguments

ar.p.fit	A cable.ar.p.iter object for AR(p) data, p>0.
xlab	Character string: x-axis label.
ylab	Character string: y-axis label.
main	Character string: plot title.
ctp.ci	A cable.change.conf object.

Details

The time series data and bent-cable / broken-stick fit are extracted from the argument ar.p.fit. These data are then plotted, with the fitted regression superimposed in red. The estimated transition τ and $\tau \pm \gamma$ are also marked in red. The optional ctp.ci, if provided, adds to the plot in blue the confidence interval for the CTP (unique point at which the cable's slope changes sign).

Warning

This function fails if ar.p.fit is from a non-AR(p>0) fit. For fits with independent data, use cable.lines.

Note

This function is intended for internal use by bentcable.ar.

Author(s)

Grace Chiu

References

See the bentcableAR package references.

See Also

cable.lines, plot, par

Examples

```
data(sockeye)

# AR(2) cable fit
fit.ar2 <- cable.ar.p.iter( c(13,.1,-.5,11,4,.5,-.5),
sockeye$logReturns, tol=1e-4 )
cable.ar.p.plot( fit.ar2, ctp.ci=cable.change.conf( fit.ar2, .9 ) )

# compare to this:
```

```
# fit.ar2 <- bentcable.ar( sockeye$logReturns,
# init.cable=c(13,.1,-.5,11,4), p=2, ci.level=.9 )
# cable.ar.p.plot( fit.ar2$cable, ctp.ci=fit.ar2$ctp )

# AR(4) stick fit
fit.ar4 <- cable.ar.p.iter( c(13,.1,-.5,11,.5,-.5,.5,-.5),
sockeye$logReturns, tol=1e-4, stick=TRUE )
cable.ar.p.plot( fit.ar4, ctp.ci=cable.change.conf( fit.ar4, .9 ) )

# compare to this:
# fit.ar4 <- bentcable.ar( sockeye$logReturns,
# init.cable=c(13,.1,-.5,11), p=4, stick=TRUE, ci.level=.9 )
# cable.ar.p.plot( fit.ar4$cable, ctp.ci=fit.ar4$ctp )
```

`cable.ar.p.resid` *Fitted Residuals and Innovations for AR(p>0) Bent Cable*

Description

This function computes the fitted residuals and fitted innovations from an AR(p>0) bent-cable regression.

Usage

```
cable.ar.p.resid(ar.p.fit)
```

Arguments

`ar.p.fit`	A `cable.ar.p.iter` object for AR(p) data, p>0.

Details

Fitted residuals correspond to the detrended time series, where the fitted bent cable is subtracted from the data. They retain the autocorrelation structure of the response time series.

Fitted innovations are the *estimated* residual noise after adjusting for the AR(p) structure, and should be approximately independent if the AR(p) fit successfully captures the actual autocorrelation in the data.

Both types of errors may be used for model diagnosis.

Value

`resid`	A numeric vector of fitted residuals; it has the same length as the response data.
`innov`	A numeric vector of fitted innovations; the first value in the vector corresponds to the (p+1)st time point.

Warnings

This function fails if `ar.p.fit` is from a non-AR(p>0) fit.

The fitted innovations are only meaningful if `ar.p.fit` is associated with equidistant time points with unit increments.

Note

This function is intended for internal use by `bentcable.ar`.

Author(s)

Grace Chiu

References

See the bentcableAR package references.

See Also

`cable.ar.p.diag`

Examples

```
data(sockeye)

fit.ar2 <- cable.ar.p.iter( c(13,.1,-.5,11,4,.5,-.5),
sockeye$logReturns, tol=1e-4 )
cable.ar.p.resid( fit.ar2 )
```

---

cable.change.conf *Confidence Interval for CTP of AR(p) Bent Cable*

Description

The *critical time point* (CTP) is estimated and accompanied by a Wald confidence interval.

Usage

```
cable.change.conf(ar.p.fit, level)
```

Arguments

`ar.p.fit`	A `cable.ar.p.iter` object for AR(p$\geq$0) data.
`level`	A numeric value between 0 and 1, exclusive.

Details

The CTP is the unique time point at which the cable's slope changes sign. If this exists, then it must happen inside the transition $\tau \pm \gamma$, and is estimated by this function based on the bent-cable regression supplied as `ar.p.fit`. Additionally, an approximate confidence interval using the Wald method is obtained by estimating the asymptotic variance of the CTP estimator. Variance estimation involves inverting an approximate Fisher information matrix by calling the built-in R function `solve`.

`cable.change.conf` returns an error if the CTP (almost) does not exist, e.g. when the estimated bent cable slope (almost) does not change signs, or when the fit from `ar.p.fit` is obtained with a time vector that is not `c(0,1,2,...)`. See **Warnings** below.

Value

`change.hat`	The estimated CTP.
`var`	The estimated asymptotic variance of the CTP estimator.
`interval`	The 100*`level` percent Wald confidence interval for the CTP.

Warnings

Computations for the CTP estimate and confidence interval are based on a time vector of the form `c(0,1,2,...)`. For any other form for the time vector, the CTP will not be computed, and on-screen warnings will appear. To ensure compatibility between the model fit and CTP estimates, the user is advised to fit the model using the default time vector. Then, if necessary, the user may transform the results to the preferred time scale after the model and CTP estimates have been produced.

The above computational issue implies that the function cannot handle non-time-series data. **Rationale**: In a non-time-series context design points are often non-equidistant, and the cable's slope often never changes sign; even with a sign change, the point at which this takes place may be less interpretable. In such a context, the user is advised to rely on confidence regions for (τ, γ) (see **References**).

Note

This function is intended for internal use by `bentcable.ar`.

Author(s)

Grace Chiu

References

See the `bentcableAR` package references.

See Also

`solve`, `cable.ar.p.plot`

Examples

```
data(sockeye)

# AR(2) cable fit
fit.ar2 <- cable.ar.p.iter( c(13,.1,-.5,11,4,.5,-.5),
sockeye$logReturns, tol=1e-4 )
cable.change.conf( fit.ar2, .9 )

# compare to this:
# fit.ar2 <- bentcable.ar( sockeye$logReturns,
# init.cable=c(13,.1,-.5,11,4), p=2, ci.level=.9 )
# cable.change.conf( fit.ar2$cable, .9 )

# AR(2) stick fit
stick.ar2 <- cable.ar.p.iter( c(13,.1,-.5,11,.5,-.5),
sockeye$logReturns, tol=1e-4, stick=TRUE)
cable.change.conf( stick.ar2, .9)
# compare to this:
# stick.ar2 <- bentcable.ar( sockeye$logReturns,
# init.cable=c(13,.1,-.5,11), p=2, stick=TRUE, ci.level=.9 )
# cable.change.conf( stick.ar2$cable, .9 )

# AR(4) stick fit
fit.ar4 <- cable.ar.p.iter( c(13,.1,-.5,11,.5,-.5,.5,-.5),
sockeye$logReturns, tol=1e-4, stick=TRUE )
cable.change.conf( fit.ar4, .9 )

# compare to this:
# fit.ar4 <- bentcable.ar( sockeye$logReturns,
# init.cable=c(13,.1,-.5,11), p=4, stick=TRUE, ci.level=.9 )
# cable.change.conf( fit.ar4$cable, .9 )
```

---

cable.fit.known.change

Grid-based Bent-Cable Regression for Independent or AR Data

Description

These functions compute the profile deviance over a (τ, γ)-grid to either fit a bent-cable regression with known transition, or to generate initial values for a bent-cable regression with unknown transition. `cable.dev` and `cable.fit.known.change` form the main engine of `bentcable.dev.plot`.

Usage

```
cable.ar.0.fit(y.vect, t.vect = NULL, tau.vect, gamma.vect, dev.mat,
stick = FALSE)
cable.dev(tau.vect, gamma.vect, y.vect, t.vect = NULL, p = 0)
cable.fit.known.change(y.vect, t.vect = NULL, n = NA,
tau.vect, gamma.vect, dev.mat, p = 0)
```

Arguments

y.vect	A numeric vector of response values.
t.vect	A numeric vector of design points. Specifying t.vect=NULL is equivalent to specifying the default time points c(0,1,2,...). Also see **Warnings** below.
n	Length of response vector (optional).
tau.vect	A numeric vector of τ-coordinates of the grid points.
gamma.vect	A numeric vector of γ-coordinates of the grid points.
dev.mat	A numeric matrix (can be single column) corresponding to the bent-cable profile deviance surface / function over the (τ, γ)-grid.
p	The autoregressive order (non-negative integer). p=0 specifies independent data that may or may not be from a time series context.
stick	A logical value; if TRUE then a broken stick (i.e. bent cable with γ=0.) is fitted.

Details

Given the response data in y.vect and design points in t.vect, cable.dev evaluates the bent-cable profile deviance surface / function over the user-specified (τ, γ)-grid. The returned values are intended to be used in conjunction with contour or persp, in which case tau.vect and gamma.vect should have length greater than 1 so that the returned object is a matrix with at least two columns. If such a plot is not required, then tau.vect and/or gamma.vect can be scalar. This function is internal to the main plotting interface bentcable.dev.plot.

The grid point at which the profile deviance is maximum corresponds to a bent-cable fit given a known transition that is best among the specified grid points. cable.fit.known.change locates this peak and computes this fit. If multiple peaks exist (such as along a ridge), then only that at the smallest τ and smallest γ is used.

For both functions, p=0 should be specified to indicate independent data (time series or otherwise). For time-series data, a positive integer p should be specified as the autoregressive order. Fitting is done by internally calling the built-in R function lm for p=0 and arima for non-zero p; this procedure is appropriate since bent-cable regression with a known transition is linear.

Note that the grid-based cable.fit.known.change does not locate the true peak of the continuous profile deviance surface / function. However, for a grid that traps the true peak between grid points, the returned fit is approximately the overall best fit (with all parameters unknown), and thus can be fed into bentcable.ar as initial values for computing the actual best fit. A special case is p=0 for independent data (time-series or otherwise), which can be handled by cable.ar.0.fit (called internally by bentcable.ar). cable.ar.0.fit calls cable.ar.p.iter when stick=FALSE but calls stick.ar.0 when stick=TRUE; in both cases, the built-in R function nls is utilized to perform maximum likelihood.

For all three functions, to fit a broken stick with a known break point, gamma.vect should be the single value 0, and thus dev.mat is a column matrix (see bentcable.dev.plot).

Value

fit	Returned by cable.fit.known.change and cable.ar.0.fit. For cable.fit.known.change, $fit is the AR(p) bent-cable regression at the known transition grid point; if p=0, it is an lm object, otherwise it is an arima object.

	For `cable.ar.0.fit`, `$fit` is an `nls` object that is the maximum likelihood bent-cable fit.
`init`	Returned by `cable.fit.known.change`, containing the coefficients from `$fit` that can be used as initial values in bent-cable regression with unknown transition.
`y`	Same as `y.vect`: returned by `cable.ar.0.fit`.
`t`	Same as `t.vect`: returned by `cable.ar.0.fit`.
`n, p, stick`	As supplied by the user: returned by `cable.ar.0.fit`.

`cable.dev` returns the evaluated profile deviance surface / function as a matrix.

Warnings

For time-series data, `t.vect` *MUST* be equidistant with unit increments; otherwise, these functions will return meaningless values. (For independent data, `t.vect` can be non-equidistant.)

Grid-based bent-cable regression and its use in subsequent overall fits rely on locating the region in which the continous deviance surface truly peaks. The user should be aware of possible local maxima and/or coarse grids that result in less-than-best fits.

Note

These functions are intended for internal use by `bentcable.dev.plot` and `bentcable.ar`.

Author(s)

Grace Chiu

References

See the `bentcableAR` package references.

See Also

`bentcable.dev.plot`, `bentcable.ar`, `nls`, `lm`, `arima`

Examples

```
data(stagnant)
data(sockeye)

# non-time-series data: compute grid-based profile deviance
cable.dev( seq(-.04,.16,length=10), seq(.2,.65,length=10),
stagnant$loght, stagnant$logflow )
# compare to this:
# bentcable.dev.plot( seq(-.04,.16,length=10),
# seq(.2,.65,length=10), stagnant$loght, stagnant$logflow )$dev

# AR(2) bent cable, start time at 0: find best grid-based fit
dev <- cable.dev( seq(6,18,length=15), seq(.01,12,length=15),
sockeye$logReturns, p=2 )
```

```
contour( seq(6,18,length=15), seq(.01,12,length=15), dev )
cable.fit.known.change( sockeye$logReturns, tau.v=seq(6,18,length=15),
gamma.v=seq(.01,12,length=15), dev.mat=dev, p=2 )

# AR(0) broken stick, start time at 80: find best overall fit
dev <- cable.dev ( seq(85,97,length=15), 0, sockeye$logReturns,
sockeye$year)
plot( seq(85,97,length=15), dev, type="l" )
cable.ar.0.fit( sockeye$logReturns, sockeye$year,
tau.v=seq(85,97,length=15), gamma.v=0, dev.mat=dev,
stick=TRUE )
# compare to this:
# bentcable.ar( sockeye$logReturns, bentcable.dev.plot(
# seq(85,97,length=15), 0, sockeye$logReturns, sockeye$year, TRUE
# ), stick=TRUE, t.vect=sockeye$year )
```

`cable.lines` *Overlay Bent Cable On Existing Plot*

Description

A user-specified bent cable is added to an existing plot. Its intended use is for superimposing a bent-cable regression fit to a scatterplot of the data. The transition is marked by vertical lines at τ and $\tau \pm \gamma$.

Usage

```
cable.lines(x, theta, col = "black", lwd = 1, lty = 2, fit.lty = 1)
```

Arguments

`x`	A numeric vector of design points or the range of these design points on the existing scatterplot.
`theta`	A vector of bent-cable coefficients, in the form of `c(b0,b1,b2,tau,gamma)`.
`col, lwd`	Graphical parameters for plotting the bent-cable function and transition.
`lty`	Graphical parameter for marking the transition.
`fit.lty`	Graphical parameter of type `lty` for plotting the bent-cable function.

Note

This function is intended for internal use by `bentcable.ar`.

Author(s)

Grace Chiu

References

See the bentcableAR package references.

See Also

fullcable.t, par, lines

Examples

```
data(sockeye)

plot(sockeye)
cable.lines( sockeye$year, c(6.6,.08,-.68,92,6.01) )
```

`fullcable.t` *Evaluate Bent Cable Function*

Description

The bent-cable response is evaluated at a provided design point `t`.

Usage

```
fullcable.t(t, b0, b1, b2, tau, gamma)
```

Arguments

`t`	A design point at which the bent cable is to be evaluated.
`b0`	Intercept.
`b1`	Incoming slope.
`b2`	Coefficient of the *basic bent cable*.
`tau`	Centre of the quadratic bend (transition).
`gamma`	Non-negative half-width of the quadratic bend.

Details

All arguments must be numeric, and at most one can be a vector.

The *full bent cable* has the form $f(t) = b_0 + b_1 t + b_2 q(t)$, where $q(t)$ is the *basic bent cable* function with intercept and slope 0 and outgoing slope 1:

$$q(t) = \frac{(t-\tau+\gamma)^2}{4\gamma} I\{|t-\tau| \le \gamma\} + (t-\tau) I\{t > \tau+\gamma\}$$

for $\gamma \ge 0$.

Note

This function is intended for internal use by `bentcable.ar` and `bentcable.dev.plot`.

Author(s)

Grace Chiu

References

See the bentcableAR package references.

Examples

```
# basic broken stick, kink at 0:
plot( seq(-10,10), fullcable.t(seq(-10,10),0,0,1,0,0) )

# full bent cable, bend centred at 0 with half-width 3:
plot( seq(-10,10), fullcable.t(seq(-10,10),1,.1,-.5,0,3) )
```

`is.stationary` *Stationarity Check of AR Time Series*

Description

Check if AR coefficients correspond to stationarity.

Usage

```
is.stationary(phi.vect)
```

Arguments

`phi.vect`	A vector of at least one AR coefficient.

Details

Stationarity check is performed via the simulation of an AR time series using the built-in R function `arima.sim`.

Value

logical

Note

This function is intended for internal use by `bentcable.ar`.

Author(s)

Grace Chiu

Examples

```
is.stationary(1) # F
is.stationary(c(-.5,.2)) # T
```

`sockeye` *Rivers Inlet Sockeye Abundance*

Description

This dataset contains the figures for the returns of Rivers Inlet sockeye salmon (*Oncorhynchus nerka*) recorded annually from 1980 to 2000.

Usage

```
data(sockeye)
```

Format

A data frame with two columns:

- `year` The year minus 1900. E.g. `80` is 1980 and `100` is the year 2000.
- `logReturns` The figure for the returning number of salmon, converted to the natural logarithmic scale.

Source

Fisheries and Oceans Canada, Pacific Region.

References

Chiu, G., Lockhart, R. and Routledge, R. (2006), Bent-Cable Regression Theory and Applications, *Journal of the American Statistical Association*, **101**, 542–553. http://faculty.washington.edu/gchiu/Articles/bentcable-jasa.pdf

stagnant *Stagnant Band Height Data*

Description

This is the well-studied data from R.A. Cook's Ph.D. thesis (Queen's University) on the relationship between the flow rate and band height of water on an incline.

Usage

```
data(stagnant)
```

Format

A data frame with two columns:

- `logflow` The flow rate in g/cm-s, log-transformed
- `loght` The band height in cm, log-transformed.

Source

Seber, G. A. F. and Wild, C. J. (2003), *Nonlinear Regression*, Hoboken: Wiley.

References

Chiu, G., Lockhart, R. and Routledge, R. (2006), Bent-Cable Regression Theory and Applications, *Journal of the American Statistical Association*, **101**, 542–553. http://faculty.washington.edu/gchiu/Articles/bentcable-jasa.pdf

stick.ar.0 *Broken-Stick Regression for Independent Data*

Description

This function is the main engine for `bentcable.ar` when a broken stick (i.e. γ=0 for bent cable) model is assumed for independent data. For AR(p) time-series data, this function is intended for determining an appropriate p and initial values for the stick parameters.

Usage

```
stick.ar.0(init.vect, y.vect, t.vect = NULL, n = NA)
```

Arguments

`init.vect`	A numeric vector of initial values, in the form of `c(b0,b1,b2,tau)`.
`y.vect`	A numeric vector of response data.
`t.vect`	A numeric vector of design points, which need not be equidistant. Specifying `t.vect=NULL` is equivalent to specifying the default time points `c(0,1,2,...)`.
`n`	Length of response vector (optional).

Details

The returned object is compatible with a `cable.ar.p.iter` object for independent data.

The broken stick as a special case of the bent cable has form $f(t) = b_0 + b_1 t + b_2(t-\tau)I\{t > \tau\}$.

Broken-stick regression by maximum likelihood for independent data is performed via nonlinear least-squares estimation of $\theta = (b_0, b_1, b_2, \tau)$ through the built-in R function `nls`. The estimation relies on the user-supplied initial values in `init`.

Value

`fit`	An `nls` object that is the maximum likelihood fit.
`y, t, n`	As supplied by the user.
`p, stick`	The values `0` and `TRUE`, respectively; used internally by `bentcable.ar` and `cable.ar.0.fit`.

Note

This function is intended for internal use by `bentcable.ar`.

Author(s)

Grace Chiu

References

See the bentcableAR package references.

See Also

`cable.ar.p.iter`, `fullcable.t`, `cable.fit.known.change`.

Examples

```
data(sockeye)

stick.ar.0( c(13,.1,-.7,12), sockeye$logReturns )
```

Index