Package ‘cobs’

January 24, 2020

Version 1.3-4
VersionNote Released 1.3-3 on 2017-03-29
Date 2020-01-20
Title Constrained B-Splines (Sparse Matrix Based)
Description Qualitatively Constrained (Regression) Smoothing Splines via Linear Programming and Sparse Matrices.
Author Pin T. Ng <Pin.Ng@nau.edu> and Martin Maechler
Maintainer Martin Maechler <maechler@stat.math.ethz.ch>
Imports SparseM (>= 1.6), quantreg (>= 4.65), grDevices, graphics, splines, stats, methods
Suggests Matrix
LazyData yes
BuildResaveData no
URL http://curves-etc.r-forge.r-project.org/
       http://r-forge.r-project.org/R/?group_id=846
License GPL (>= 2)
NeedsCompilation yes
Repository CRAN
Date/Publication 2020-01-24 14:20:07 UTC

R topics documented:
cobs ................................................................. 2
cobs-methods .................................................... 6
conreg ............................................................. 7
conreg-methods .................................................. 10
drqssbc2 .......................................................... 11
DublinWind ....................................................... 14
exHe ............................................................... 15
globtemp .......................................................... 16
Computes constrained quantile curves using linear or quadratic splines. The median spline ($L_1$ loss) is a robust (constrained) smoother.

**Usage**

```r
cobs(x, y, constraint = c("none", "increase", "decrease", "convex", "concave", "periodic"), w = rep(1,n), knots, nknots = if(lambda == 0) 6 else 20, method = "quantile", degree = 2, tau = 0.5, lambda = 0, ic = c("AIC", "SIC", "BIC", "aic", "sic", "bic"), knots.add = FALSE, repeat.delete.add = FALSE, pointwise = NULL, keep.data = TRUE, keep.x.ps = TRUE, print.warn = TRUE, print.mesg = TRUE, trace = print.mesg, lambdaSet = exp(seq(log(lambda.lo), log(lambda.hi), length = lambda.length)), lambda.lo = f.lambda*1e-4, lambda.hi = f.lambda*1e3, lambda.length = 25, maxiter = 100, rq.tol = 1e-8, toler.kn = 1e-6, tol.0res = 1e-6, nk.start = 2)
```

**Arguments**

- **x**: vector of covariate; missing values are omitted.
- **y**: vector of response variable. It must have the same length as x.
- **constraint**: character (string) specifying the kind of constraint; must be one of the values in the default list above; may be abbreviated. More flexible constraints can be specified via the `pointwise` argument (below).
- **w**: vector of weights the same length as x (y) assigned to both x and y; default to all weights being one.
- **knots**: vector of locations of the knot mesh; if missing, nknots number of knots will be created using the specified method and automatic knot selection will be carried out for regression B-spline (lambda=0); if not missing and length(knots)==nknots, the provided knot mesh will be used in the fit and no automatic knot selection will be performed; otherwise, automatic knots selection will be performed on the provided knots.
nknots  maximum number of knots; defaults to 6 for regression B-splines, 20 for smoothing B-splines.

method  character string specifying the method for generating nknots number of knots when knots is not provided; either "quantile" (equally spaced in percentile levels) or "uniform" (equally spaced knots); defaults to "quantile".

degree  degree of the splines; 1 for linear spline (equivalent to $L_1$-roughness) and 2 for quadratic spline (corresponding to $L_\infty$ roughness); defaults to 2.

tau  desired quantile level; defaults to 0.5 (median).

lambda  penalty parameter $\lambda$

$\lambda = 0$: no penalty (regression B-spline);
$\lambda > 0$: smoothing B-spline with the given $\lambda$;
$\lambda < 0$: smoothing B-spline with $\lambda$ chosen by a Schwarz-type information criterion.

ic  string indicating the information criterion used in knot deletion and addition for the regression B-spline method, i.e., when lambda == 0;
"AIC" (Akaike-type, equivalently "aic") or
"SIC" (Schwarz-type, equivalently "BIC", "sic" or "bic"). Defaults to "AIC".

knots.add  logical indicating if an additional step of stepwise knot addition should be performed for regression B-splines.

repeat.delete.add  logical indicating if an additional step of stepwise knot deletion should be performed for regression B-splines.

pointwise  an optional three-column matrix with each row specifies one of the following constraints:

$( 1, x_i, y_i)$: fitted value at $x_i$ will be $\geq y_i$;
$(-1, x_i, y_i)$: fitted value at $x_i$ will be $\leq y_i$;
$( 0, x_i, y_i)$: fitted value at $x_i$ will be $= y_i$;
$( 2, x_i, y_i)$: derivative of the fitted function at $x_i$ will be $y_i$.

keep.data  logical indicating if the x and y input vectors after removing NAs should be kept in the result.

keep.x.ps  logical indicating if the pseudo design matrix $\tilde{X}$ should be returned (as sparse matrix). That is needed for interval prediction, predict.cobs(*, interval=..).

print.warn  flag for printing of interactive warning messages; true by default; set to FALSE if performing simulation.

print.mesg  logical flag or integer for printing of intermediate messages; true by default. Probably needs to be set to FALSE in simulations.

trace  integer $\geq 0$ indicating how much diagnostics the low-level code in drqssbc2 should print; defaults to print.mesg.

lambdaSet  numeric vector of lambda values to use for grid search; in that case, defaults to a geometric sequence (a "grid in log scale") from lambda.lo to lambda.hi of length lambda.length.
lambda.lo, lambda.hi
lower and upper bound of the grid search for lambda (when lambda < 0). The
defaults are meant to keep everything scale-equivariant and are hence using the
factor
\[ f = \sigma \cdot d \cdot x \]
i.e., \( f \cdot \text{lambda} <- \text{sd(x)}^\text{degree} \).
Note however that currently the underlying algorithms in package \texttt{quantreg} are
not scale equivariant yet.

lambda.length
number of grid points in the grid search for optimal lambda.

maxiter
upper bound of the number of iterations; defaults to 100.

rq.tol
numeric convergence tolerance for the interior point algorithm called from \texttt{rq.fit.sfnc()} or
\texttt{rq.fit.sfn()}.

ampler.kn
numeric tolerance for shifting the boundary knots outside; defaults to \(10^{-6}\).

tol.0res
tolerance for testing \(|r_i| = 0\), passed to \texttt{qbsks2} and \texttt{drqssbc2}.

nk.start
number of starting knots used in automatic knot selection. Defaults to the minimum
of 2 knots.

Details
cobs() computes the constraint quantile smoothing B-spline with penalty when lambda is not zero.
If lambda < 0, an optimal lambda will be chosen using Schwarz type information criterion.
If lambda > 0, the supplied lambda will be used.
If lambda = 0, cobs computes the constraint quantile regression B-spline with no penalty using the
provided knots or those selected by Akaike or Schwarz information criterion.

Value
an object of class cobs, a list with components

call
the \texttt{cobs(.)} call used for creation.

tau, degree
same as input

constraint
as input (but no more abbreviated).

pointwise
as input.

coef
B-spline coefficients.

knots
the final set of knots used in the computation.

ifl
exit code := 1 + ierr and ierr is the error from \texttt{rq.fit.sfnc} (package \texttt{quantreg});
consequently, ifl = 1 means “success”; all other values point to algorithmic
problems or failures.

icyc
length 2: number of cycles taken to achieve convergence for final lambda, and
total number of cycles for all lambdas.

k
the effective dimensionality of the final fit.

k0
(usually the same)

x.ps
the pseudo design matrix X (as returned by \texttt{qbsks2}).

resid
vector of residuals from the fit.
fitted  vector of fitted values from the fit.
SSy     the sum of squares around centered y (e.g. for computation of $R^2$.)
lambda  the penalty parameter used in the final fit.
pp.lambda vector of all lambdas used for lambda search when lambda < 0 on input.
pp.sic   vector of Schwarz information criteria evaluated at pp.lambda; note that it is not quite sure how good these are for determining an optimal lambda.

References


A postscript version of the paper that describes the details of COBS can be downloaded from \url{http://www.cba.nau.edu/pin-ng/cobs.html}.

See Also

\texttt{smooth.spline} for unconstrained smoothing splines; \texttt{bs} for unconstrained (regression) B-splines.

Examples

```r
x <- seq(-1,3,,150)
y <- (f.true <- pnorm(2*x)) + rnorm(150)/10
## specify pointwise constraints (boundary conditions)
con <- rbind(c( 1,min(x),0), # f(min(x)) >= 0
        c(-1,max(x),1), # f(max(x)) <= 1
        c(0, 0, 0.5))# f(0) = 0.5

## obtain the median REGRESSION B-spline using automatically selected knots
Rbs <- cobs(x,y, constraint= "increase", pointwise = con)
Rbs
plot(Rbs, lwd = 2.5)
lines(spline(x, f.true), col = "gray40")
lines(predict(cobs(x,y)), col = "blue")
mtext("cobs(x,y) # completely unconstrained", 3, col= "blue")

## compute the median SMOOTHING B-spline using automatically chosen lambda
Sbs <- cobs(x,y, constraint="increase", pointwise= con, lambda= -1)
```
cobs-methods

Methods for COBS Objects

Description
Print, summary and other methods for cobs objects.

Usage

## S3 method for class 'cobs'
print(x, digits = getOption("digits"), ...)
## S3 method for class 'cobs'
summary(object, digits = getOption("digits"), ...)

## S3 method for class 'cobs'
coef(object, ...)
## S3 method for class 'cobs'
fitted(object, ...)
## S3 method for class 'cobs'
knots(Fn, ...)
## S3 method for class 'cobs'
residuals(object, ...)

Arguments

x, object, Fn  object of class cobs.
digits  number of digits to use for printing.
...  further arguments passed from and to methods.

Details
These are methods for fitted COBS objects, as computed by cobs.
Value

print.cobs() returns its argument invisibly. The coef(), fitted(), knots(), and residuals() methods return a numeric vector.

Author(s)

Martin Maechler

See Also

predict.cobs for the predict method, plot.cobs for the plot method, and cobs for examples.

Examples

example(cobs)
Sbs # uses print.*

summary(Sbs)
coef(Sbs)
knots(Sbs)

Description

Compute a univariate concave or convex regression, i.e., for given vectors, $x, y, w$ in $R^n$, where $x$ has to be strictly sorted ($x_1 < x_2 < \ldots < x_n$), compute an $n$-vector $m$ minimizing the weighted sum of squares $\sum_{i=1}^{n} w_i(y_i - m_i)^2$ under the constraints

$$(m_i - m_{i-1})/(x_i - x_{i-1}) \geq (m_{i+1} - m_i)/(x_{i+1} - x_i),$$

for $1 \leq i \leq n$ and $m_0 := m_{n+1} := -\infty$, for concavity. For convexity (convex=TRUE), replace $\geq$ by $\leq$ and $-\infty$ by $+\infty$.

Usage

conreg(x, y = NULL, w = NULL, convex = FALSE,
method = c("Duembgen06.R", "SR"),
tol = c(1e-10, 1e-7), maxit = c(500, 20),
adjTol = TRUE, verbose = FALSE)
Arguments

- **x, y** numeric vectors giving the values of the predictor and response variable. Alternatively a single “plotting” structure (two-column matrix / y-values only / list, etc) can be specified: see `xy.coords`.
- **w** for method "Duembgen06_R" only: optional vector of weights of the same length as x; defaults to all 1.
- **convex** logical indicating if convex or concave regression is desired.
- **method** a character string indicating the method used, "Duembgen06_R" is an active set method written by Lutz Duembgen (University of Berne, CH) in Matlab in 2006 and translated to R by Martin Maechler.
- **tol** convergence tolerance(s); do not make this too small!
- **maxit** maximal number of (outer and inner) iterations of knot selection.
- **adjTol** (for "Duembgen06_R" only:) logical indicating if the convergence test tolerance is to be adjusted (increased) in some cases.
- **verbose** logical or integer indicating if (and how much) knot placement and fitting iterations should be “reported”.

Details

Both algorithms need some numerical tolerances because of rounding errors in computation of finite difference ratios. The active-set "Duembgen06_R" method notably has two different such tolerances which were both $10^{-7}$ up to March 2016.

The two default tolerances (and the exact convergence checks) may change in the future, possibly to become more adaptive.

Value

an object of class `conreg` which is basically a list with components

- **x** sorted (and possibly aggregated) abscissa values x.
- **y** corresponding y values.
- **w** corresponding weights, only for "Duembgen06_R".
- **yf** corresponding fitted values.
- **convex** logical indicating if a convex or a concave fit has been computed.
- **iKnots** integer vector giving indices of the knots, i.e. locations where the fitted curve has kinks. Formally, these are exactly those indices where the constraint is fulfilled strictly, i.e., those $i$ where $(m_i - m_{i-1})/(x_i - x_{i-1}) > (m_{i+1} - m_i)/(x_{i+1} - x_i)$.
- **call** the call to `conreg()` used.
**iter**

integer (vector of length one or two) with the number of iterations used (in the outer and inner loop for "Duembgen06_R").

Note that there are several methods defined for conreg objects, see predict.conreg or methods(class = "conreg").

Notably print and plot; also predict, residuals, fitted, knots.

Also, interpSplineCon() to construct a more smooth (cubic) spline, and isIsplineCon() which checks if the int is strictly concave or convex the same as the conreg() result from which it was constructed.

**Author(s)**

Lutz Duembgen programmed the original Matlab code in July 2006; Martin Maechler ported it to R, tested, catch infinite loops, added more options, improved tolerance, etc; from April 27, 2007.

**See Also**

isoreg for isotone (monotone) regression; CRAN packages ftnonpar, cobs, logcondens.

**Examples**

```r
## Generated data :
N <- 100
f <- function(X) 4*X*(1 - X)
xx <- seq(0,1, length=501)# for plotting true f()
set.seed(1)# -> conreg does not give convex cubic
x <- sort(runif(N))
y <- f(x) + 0.2 * rnorm(N)
plot(x,y, cex = 0.6)
lines(xx, f(xx), col = "blue", lty=2)
rc <- conreg(x,y)
lines(rc, col = 2, force.iSpl = TRUE)
# 'force.iSpl': force the drawing of the cubic spline through the kinks
title("Concave Regression in R")
y2 <- y

## Trivial cases work too:
(r.1 <- conreg(1,7))
(r.2 <- conreg(1:2,7:6))
(r.3 <- conreg(1:3,c(4:5,1)))
```
Summary Methods for 'conreg' Objects

Description
Methods for conreg objects

Usage

## S3 method for class 'conreg'
fitted(object, ...)
## S3 method for class 'conreg'
residuals(object, ...)
## S3 method for class 'conreg'
knots(Fn, ...)
## S3 method for class 'conreg'
lines(x, type = "l", col = 2, lwd = 1.5, show.knots = TRUE,
       add.iSpline = TRUE, force.iSpl = FALSE, ...)
## S3 method for class 'conreg'
plot(x, type = "l", col = 2, lwd = 1.5, show.knots = TRUE,
      add.iSpline = TRUE, force.iSpl = FALSE,
      xlab = "x", ylab = expression(s[c](x)),
      sub = "simple concave regression", col.sub = col, ...)
## S3 method for class 'conreg'
predict(object, x, deriv = 0, ...)
## S3 method for class 'conreg'
print(x, digits = max(3, getOption("digits") - 3), ...)

Arguments

object, Fn, x  an R object of class conreg, i.e., typically the result of conreg(....). For predict(), x is a numeric vector of abscissa values at which to evaluate the concave/convex spline function.
type, col, lwd, xlab, ylab, sub, col.sub
plotting arguments as in plot.default.
show.knots  logical indicating the spline knots should be marked additionally.
add.iSpline  logical indicating if an interpolation spline should be considered for plotting. This is only used when it is itself concave/convex, unless force.iSpl is TRUE.
force.iSpl  logical indicating if an interpolating spline is drawn even when it is not concave/convex.
deriv  for predict, integer specifying the derivative to be computed; currently must be 0 or 1.
drqssbc2

Regression Quantile Smoothing Spline with Constraints

Description

Estimate the B-spline coefficients for a regression quantile smoothing spline with optional constraints, using Ng(1996)'s algorithm.

digits number of significant digits for printing.
...
further arguments, potentially passed to methods.

Author(s)

Martin Maechler

See Also

conreg,...

Examples

example(conreg, echo = FALSE)
class(rc) # "conreg"
rc # calls the print method
knots(rc)
plot(rc)
## and now _force_ the not-quite-concave cubic spline :
plot(rc, force.iSpl=TRUE)

xx <- seq(-0.1, 1.1, length=201) # slightly extrapolate
## Get function s(x) and first derivative s'(x) :
yx <- predict(rc, xx)
y1 <- predict(rc, xx, deriv = 1)

op <- par(las=1)
plot(xx, yx, type = "l",
     main="plot(xx, predict( conreg(.), xx))")
par(new=TRUE) # draw the first derivative "on top"
plot(xx, y1, type = "l", col = "blue",
     axes = FALSE, ann = FALSE)
abline(h = 0, lty="1A", col="blue")
axis(4, col="blue", col.axis="blue", col.ticks="blue")
mtext("first derivative s'(.)", col="blue")
par(op)
Usage

drqssbc2(x, y, w = rep.int(1,n), pw, knots, degree, Tlambda,  
        constraint, ptConstr, maxiter = 100, trace = 0,  
        nrq = length(x), nll, neqc, niqc, nvar,  
        tau = 0.5, select.lambda, give.pseudo.x = FALSE,  
        rq.tol = 1e-8 * sc.y, tol.0res = 1e-6,  
        print.warn = TRUE, rq.print.warn = FALSE)

Arguments

- **x**: numeric vector, sorted increasingly, the abscissa values
- **y**: numeric, same length as x, the observations.
- **w** numeric vector of weights, same length as x, as in cobs.
- **pw**: penalty weights vector passed to l1.design2 or loo.design2. FIXME: This is currently unused.
- **knots**: numeric vector of knots for the splines.
- **degree**: integer, must be 1 or 2.
- **Tlambda**: vector of smoothing parameter values $\lambda$; if it is longer than one, an “optimal” value will be selected from these.
- **constraint**: see cobs (but cannot be abbreviated here).
- **ptConstr**: list of pointwise constraints; notably equal, smaller, greater and gradient are 3-column matrices specifying the respective constraints. May have 0 rows if there are no constraints of the corresponding kind.
- **maxiter**: maximal number of iterations; defaults to 100.
- **trace**: integer or logical indicating the tracing level of the underlying algorithms; not much implemented (due to lack of trace in quantreg ...)
- **nrq**: integer, $= n$, the number of observations.
- **nll**: integer, number of observations in the l1 norm that correspond to roughness measure (may be zero).
- **neqc**: integer giving the number of equations.
- **niqc**: integer giving the number of inequality constraints; of the same length as constraint.
- **nvar**: integer giving the number of equations and constraints.
- **tau**: desired quantile level; defaults to 0.5 (median).
- **select.lambda**: logical indicating if an optimal lambda should be selected from the vector of Tlambda.
- **give.pseudo.x**: logical indicating if the pseudo design matrix $\tilde{X}$ should be returned (as sparse matrix).
- **rq.tol**: numeric convergence tolerance for the interior point algorithm called from rq.fit.sfnc() or rq.fit.sfn(). Note that (for scale invariance) this has to be in units of y, which the default makes use of.
- **tol.0res**: tolerance used to check for zero residuals, i.e., $|r_i| < tol \ast \text{mean}(|r_i|)$. 
drqssbc2

print.warn logical indicating if warnings should be printed, when the algorithm seems to have behaved somewhat unexpectedly.

rq.print.warn logical indicating if warnings should be printed from inside the rq.* function calls, see below.

Details

This is an auxiliary function for cobs, possibly interesting on its own. Depending on degree, either l1.design2 or loo.design2 are called for construction of the sparse design matrix.

Subsequently, either rq.fit.sfnc or rq.fit.sfn is called as the main “work horse”.

This documentation is currently sparse; read the source code!

Value

a list with components

comp1 Description of ‘comp1’
comp2 Description of ‘comp2’
...

Author(s)

Pin Ng; this help page: Martin Maechler.

References


See Also

The main function cobs and its auxiliary qbsks2 which calls drqssbc2() repeatedly.

l1.design2 and loo.design2; further rq.fit.sfnc and rq.fit.sfn from package quantreg.

Examples

set.seed(1243)
x <- 1:32
fx <- (x-5)*(x-15)^2*(x-21)
y <- fx + round(rnorm(x,s = 0.25),2)
DublinWind

Daily Wind Speeds in Dublin

Description

The DublinWind data frame is basically the time series of daily average wind speeds from 1961 to 1978, measured in Dublin, Ireland. These are 6574 observations (18 full years among which four leap years).

Usage

data(DublinWind)

Format

This data frame contains the following columns:

- `speed` numeric vector of average daily wind speed in knots
- `day` an integer vector giving the day number of the year, i.e., one of 1:366.

Details

The periodic pattern along the 18 years measured and the autocorrelation are to be taken into account for analysis, see the references. This is Example 3 of the COBS paper.

Source

From shar file available from [http://www.cba.nau.edu/pin-ng/cobs.html](http://www.cba.nau.edu/pin-ng/cobs.html)

Has also been available from Statlib; then, with more variables, e.g., in `help(wind, package = "gstat")` from CRAN package `gstat`.

References


Examples

data(DublinWind)
str(DublinWind)
plot(speed ~ day, data = DublinWind)# not so nice; want time scale

## transform 'day' to correct "Date" object; and then plot
Dday <- seq(from = as.Date("1961-01-01"), by = 1,
    length = nrow(DublinWind))
plot(speed ~ Dday, data = DublinWind, type = "l",
    main = paste("DublinWind speed daily data, n=",
        nrow(DublinWind)))

##--- ~ He & Ng "Example 3" %%% much more is in ../tests/wind.R
co.o50 <-
    with(DublinWind, ## use nknots > (default) 6 :
        cobs(day, speed, knots.add = TRUE, constraint= "periodic", nknots = 10,
            tau = .5, method = "uniform"))
summary(co.o50)
lines(Dday, fitted(co.o50), col=2, lwd = 2)

## the periodic "smooth" function - in one period
plot(predict(co.o50), type = "o", cex = 0.2, col=2,
    xlab = "day", ylim = c(0,20))
points(speed ~ day, data = DublinWind, pch = ".")

---

**exHe**  

*Small Dataset Example of He*

Description

The `exHe` data frame has 10 rows and 2 columns. It is an example for which `smooth.spline` cannot be used.

Usage

`data(exHe)`

Format

This data frame contains the following columns:

- **x** only values 0, 1, and 2.
- **y** 10 randomly generated values
Details

Xuming He wrote about this **JUST FOR FUN:**
I was testing COBS using the following "data". For comparison, I tried smooth.spline in S+. I never got an answer back! No warning messages either. The point is that even the well-tested algorithm like `smooth.spline` could leave you puzzled.

To tell you the truth, the response values here were generated by white noise. An ideal fitted curve would be a flat line. See for yourself what COBS would do in this case.

Source

Originally found at the bottom of http://ux6.cso.uiuc.edu/~x-he/ftp.html, the web resource directory of Xuming He at the time, say 2006.

See Also

cobs

Examples

data(exHe)
plot(exHe, main = "He's 10 point example and cobs() fits")
tm <- tapply(exHe$y, exHe$x, mean)
lines(unique(exHe$x), tm, lty = 2)

cH. <- with(exHe,
  cobs(x, y, degree=1, constraint = "increase"))
cH <- with(exHe,
  cobs(x, y, lambda=0.2, degree=1, constraint = "increase"))
plot(exHe)
lines(predict(cH.), type = "o", col="tomato3", pch = "i")# constant
lines(predict(cH), type = "o", col=2, pch = "i")

cHn <- cobs(exHe$x, exHe$y, degree=1, constraint = "none")
lines(predict(cHn), col= 3, type = "o", pch = "n")

cHd <- cobs(exHe$x, exHe$y, degree=1, constraint = "decrease")
lines(predict(cHd), col= 4, type = "o", pch = "d")

---

**globtemp**  
*Annual Average Global Surface Temperature*

Description

Time Series of length 113 of annual average global surface temperature deviations from 1880 to 1992.

Usage

data(globtemp)
Details

This is Example 1 of the COBS paper, where the hypothesis of a monotonely increasing trend is considered; Koenker and Schorfheide (1994) consider modeling the autocorrelations.

Source


References

He, X. and Ng, P. (1999) COBS: Qualitatively Constrained Smoothing via Linear Programming; Computational Statistics 14, 315–337.

Examples

data(globtemp)
plot(globtemp, main = "Annual Global Temperature Deviations")
str(globtemp)
## forget about time-series, just use numeric vectors:
year <- as.vector(time(globtemp))
temp <- as.vector(globtemp)

##---- Code for Figure 1a of He and Ng (1999) -----------
a50 <- cobs(year, temp, knots.add = TRUE, degree = 1, constraint = "increase")
summary(a50)
## As suggested in the warning message, we increase the number of knots to 9
a50 <- cobs(year, temp, nknots = 9, knots.add = TRUE, degree = 1,
            constraint = "increase")
summary(a50)
## Here, we use the same knots sequence chosen for the 50th percentile
a10 <- cobs(year, temp, nknots = length(a50$knots), knots = a50$knot,
             degree = 1, tau = 0.1, constraint = "increase")
summary(a10)
a90 <- cobs(year, temp, nknots = length(a50$knots), knots = a50$knot,
             degree = 1, tau = 0.9, constraint = "increase")
summary(a90)

which(hot.idx <- temp >= a90$fit)
which(cold.idx <- temp <= a10$fit)
normal.idx <- !hot.idx & !cold.idx

plot(year, temp, type = "n", ylab = "Temperature (C)", ylim = c(-.7,.6))
lines(predict(a50, year, interval = "both"), col = 2)
lines(predict(a10, year, interval = "both"), col = 3)
lines(predict(a90, year, interval = "both"), col = 3)
points(year, temp, pch = c(1,3)[2 - normal.idx])
## interpSplineCon

### Description

From a "conreg" object representing a linear spline,

- `interpSplineCon()` produces the corresponding (cubic) spline (via package `splines' `interpSpline()`)
  by interpolating at the knots, thus "smoothing the kinks".
- `isIsplineCon()` determines if the spline fulfills the same convexity / concavity constraints as the underlying "conreg" object.

### Usage

```r
interpSplineCon(object, ...)
isIsplineCon(object, isp, ...)
```

### Arguments

- **object**: an R object as resulting from `conreg()`.
- **isp**: optionally, the result of `interpSplineCon(object,...)`; useful if that is already available in the caller.
- **...**: optional further arguments passed to `interpSpline()`.

### Value

- `interpSplineCon()` returns the `interpSpline()` interpolation spline object.
- `isIsplineCon()` is TRUE (or FALSE), indicating if the convexity/concavity constraints are fulfilled (in knot intervals).

### Author(s)

Martin Maechler

### See Also

`conreg, interpSpline`
Examples

cc <- conreg(cars[,"speed"], cars[,"dist"], convex=TRUE)
iS <- interpSplineCon(cc)
(isC <- isIsplineCon(cc)) # FALSE: not strictly convex
## Passing the interpolation spline --- if you have it anyway ---
## is more efficient (faster):
stopifnot(identical(isC, isIsplineCon(cc, isp = iS)))

## the interpolation spline is not quite convex:
plot(cc)
with(cars, points(dist ~ speed, col = adjustcolor(1, 1/2)))
lines(predict(iS, seq(1,28, by=1/4)),
      col = adjustcolor("forest green", 3/4), lwd=2)

Description

COBS (cobs) auxiliary function for constructing the pointwise constraint specification list from the pointwise 3-column matrix (as used as argument in cobs).

Usage

mk.pt.constr(pointwise)

Arguments

pointwise numeric 3-column matrix, see pointwise in cobs.

Value

A list with components

n.equal number of equality constraints
n.greater number of "greater" constraints
n.smaller number of "smaller" constraints
n.gradient number of gradient constraints

Unless the input pointwise was NULL, the result also has corresponding components:

equal 3-column matrix of equality constraints
greater 3-column matrix of "greater" constraints
smaller 3-column matrix of "smaller" constraints
gradient 3-column matrix of gradient constraints
Author(s)

Martin Maechler

Examples

```r
## from ?cobs:
x <- seq(-1.3, 1.5, 150)
con <- rbind(c(1, min(x), 0),
            c(-1, max(x), 1),
            c(0, 0, 0.5))  # f(0) = 0.5
r <- mk.pt.constr(con)
str(r)
```

plot.cobs

Plot Method for COBS Objects

Description

The `plot` method for `cobs` objects. If there was lambda selection, it provides two plots by default.

Usage

```r
## S3 method for class 'cobs'
plot(x, which = if(x$select.lambda) 1:2 else 2,
     show.k = TRUE,
     col = par("col"), l.col = c("red","pink"), k.col = gray(c(0.6, 0.8)),
     lwd = 2, cex = 0.4, ylim = NULL,
     xlab = deparse(x$call[[2]]),
     ylab = deparse(x$call[[3]]),
     main = paste(deparse(x$call, width.cutoff = 100), collapse="\n"),
     subtit = c("choosing lambda", "data & spline curve"),
     ...)
```

Arguments

- `x` object of class `cobs`.
- `which` integer vector specifying which plots should be drawn.
- `show.k` logical indicating if the “effective dimensionality” `k` should also be shown. Only applicable when `which` contains 1.
- `col`, `l.col`, `k.col` colors for plotting; `k.col` only applies when `show.k` is true in the first plot (which == 1) where `l.col[2]` and `k.col[2]` are only used as well, for denoting “doubtful” points; `col` is only used for the 2nd plot (which == 2).
- `lwd`, `cex` line width and point size for the 2nd plot (i.e. which == 2).
- `ylim` y-axis limits, see `axis`, with a smart default.
- `xlab`, `ylab`, `main` axis annotation; see also `axis`.
- `subtit` a vector of length 2, specifying a sub title for each plot (according to which).
- `...` further arguments passed to internal `plot` methods.
Details
plot(.) produces two side-by-side plots in case there was a search for the optimal lambda (which = 1:2), and only the (second) data plus spline curve plot otherwise (which = 2).

Author(s)
Martin Maechler

See Also
There are several other methods for COBS objects, see, e.g. summary.cobs or predict.cobs. cobs for examples.

Examples

example(cobs)

plot(Sbs)
plot(fitted(Sbs), resid(Sbs),
   main = "Tukey-Anscombe plot for cobs()",
   sub = deparse(Sbs$call))

---

predict.cobs  Predict method for COBS Fits

Description
Compute predicted values and simultaneous or pointwise confidence bounds for cobs objects.

Usage

## S3 method for class 'cobs'
predict(object, z, deriv = 0L,
   minz = knots[1], maxz = knots[nknots], nz = 100,
   interval = c("none", "confidence", "simultaneous", "both"),
   level = 0.95, ...)

Arguments

object  object of class cobs.
z  vector of grid points at which the fitted values are evaluated; defaults to an equally spaced grid with nz grid points between minz and maxz. Note that now z may lie outside of the knots interval which was not allowed originally.
deriv  scalar integer specifying (the order of) the derivative that should be computed.
minz  numeric needed if z is not specified; defaults to min(x) or the first knot if knots are given.
maxz analogous to minz; defaults to max(x) or the last knot if knots are given.
nz number of grid points in z if that is not given; defaults to 100.
interval type of interval calculation, see below
level confidence level
... further arguments passed to and from methods.

Value

a matrix of predictions and bounds if interval is set (not "none"). The columns are named z, fit, further cb.lo and cb.up for the simultaneous confidence band, and ci.lo and ci.up the pointwise confidence intervals according to specified level.

If z has been specified, it is unchanged in the result.

Author(s)

Martin Maechler, based on He and Ng’s code in cobs().

See Also

cobs the model fitting function.

Examples

eexample(cobs) # continuing :
(pRbs <- predict(Rbs))
#str(pSbs <- predict(Sbs, xx, interval = "both"))
str(pSbs <- predict(Sbs, xx, interval = "none"))

plot(x,y, xlim = range(xx), ylim = range(y, pSbs[,2], finite = TRUE),
     main = "COBS Median smoothing spline, automatical lambda")
lines(pSbs, col = "red")
lines(spline(x,f.true), col = "gray40")
#matlines(pSbs[,1], pSbs[,-(1:2)],
#         col= rep(c("green","blue"),c(2,2)), lty=2)

qbsks2 Quantile B-Spline with Fixed Knots

Description

Compute B-spline coefficients for regression quantile B-spline with stepwise knots selection and quantile B-spline with fixed knots regression spline, using Ng (1996)’s algorithm.
Usage

qbsks2(x,y,w,pw, knots,nknots, degree,Tlambda, constraint, ptConstr, maxiter, trace, nrq,nl1, neqc, tau, select.lambda, ks, do.select, knots.add, repeat.delete.add, ic, print.mesg, give.pseudo.x = TRUE, rq.tol = 1e-8, tol.kn = 1e-6, tol.0res = 1e-6, print.warn, nk.start)

Arguments

x numeric vector, sorted increasingly, the abscissa values
y numeric, same length as x, the observations.
w numeric vector of weights, same length as x, as in cobs.
pw penalty weights vector ...
knots numeric vector of knots of which nknots will be used.
nknots number of knots to be used.
degree integer specifying polynomial degree; must be 1 or 2.
Tlambda (vector of) smoothing parameter(s) λ, see drqssbc2.
constraint string (or empty) specifying the global constraints; see cobs.
ptConstr list of pointwise constraints.
maxiter non-negative integer: maximal number of iterations, passed to drqssbc2.
trace integer or logical indicating the tracing level of the underlying algorithms; not implemented (due to lack of trace in quantreg ...)
nrq,nl1,neqc integers specifying dimensionalities, directly passed to drqssbc2, see there.
tau desired quantile level (in interval (0,1)).
select.lambda passed to drqssbc2, see there.
ks number used as offset in SIC/AIC/BIC.
do.select logical indicating if knots shall be selected (instead of used as specified).
ks, do.select, knots.add, repeat.delete.add
give.pseudo.x logicals, see cobs.
print.mesg an integer indicating how qbsks2() should print message about its current stages.
give.pseudo.x logical indicating if the pseudo design matrix $\tilde{X}$ should be returned (as sparse matrix).
rq.tol numeric convergence tolerance for the interior point algorithm called from rq.fit.sfnc() or rq.fit.sfn().
tol.kn “tolerance” for shifting the outer knots.
tol.0res tolerance passed to drqssbc2.
print.warn flag indicating if and how much warnings and information is to be printed; currently just passed to drqssbc2.
nk.start number of starting knots used in automatic knot selection.
Details

This is an auxiliary function for \texttt{cobs(*,lambda = \emptyset)}, possibly interesting on its own. This documentation is currently sparse; read the source code!

Value

a list with components

\begin{itemize}
  \item \texttt{coef} ..
  \item \texttt{fidel} ..
  \item \texttt{k} dimensionality of model fit.
  \item \texttt{if1} integer “flag”; the return code.
  \item \texttt{icyc} integer of length 2, see \texttt{cobs}.
  \item \texttt{knots} the vector of inner knots.
  \item \texttt{nknots} the number of inner knots.
  \item \texttt{nvar} the number of “variables”, i.e. unknowns including constraints.
  \item \texttt{lambda} the penalty factor, chosen or given.
  \item \texttt{pseudo.x} the pseudo design matrix \( X \), as returned from \texttt{drqssbc2}.
\end{itemize}

Author(s)

Pin Ng; this help page: Martin Maechler.

References


See also the \textit{references} in \texttt{cobs}.

See Also

the main function \texttt{cobs}; further \texttt{drqssbc2} which is called from \texttt{qbsks2}().

\begin{center}
\begin{tabular}{lc}
\texttt{USArmyRoofs} & \textit{Roof Quality in US Army Bases} \\
\end{tabular}
\end{center}

Description

The \texttt{USArmyRoofs} data frame has 153 observations of roof sections of US Army bases and 2 columns, age and \texttt{fci}. This is Example 2 of He \\& Ng (1999).

Usage

\texttt{data(USArmyRoofs)}
**Format**

This data frame contains the following columns:

- **age** numeric vector giving the roof’s age in years.
- **fci** numeric, giving the FCI, the flash condition index, i.e., the percentage of flashing which is in good condition.

**Source**

From shar file available from [http://www.cba.nau.edu/pin-ng/cobs.html](http://www.cba.nau.edu/pin-ng/cobs.html)

**References**


**Examples**

```r
data(USArmyRoofs)
plot(fci ~ age, data = USArmyRoofs, main = "US Army Roofs data")
```
Index

*Topic datasets
  DublinWind, 14
  exHe, 15
  globtemp, 16
  USArmyRoofs, 24

*Topic misc
  mk.pt.constr, 19

*Topic models
  conreg-methods, 10

*Topic print
  cobs-methods, 6
  conreg-methods, 10
  plot.cobs, 20

*Topic regression
  cobs, 2
  conreg, 7
  predict.cobs, 21

*Topic smooth
  cobs, 2
  conreg, 7
  drqssbc2, 11
  interpSplineCon, 18
  qbsks2, 22

*Topic utilities
  drqssbc2, 11
  interpSplineCon, 18
  qbsks2, 22

axis, 20
bs, 5
call, 8
cobs, 2, 6, 7, 12, 13, 16, 19–24
  cobs-methods, 6
  coef.cobs(cobs-methods), 6
  conreg, 7, 10, 11, 18
  conreg-methods, 10
  drqssbc2, 3, 4, 11, 23, 24

DublinWind, 14
exHe, 15
fitted, 9
fitted.cobs(cobs-methods), 6
fitted.conreg(conreg-methods), 10
globtemp, 16
interpSpline, 18
interpSplineCon, 9, 18
isIsplineCon, 9
isIsplineCon(interpSplineCon), 18
isoreg, 9
knots, 9
knots.cobs(cobs-methods), 6
knots.conreg(conreg-methods), 10
l1.design2, 12, 13
lines.conreg(conreg-methods), 10
list, 12, 23
loo.design2, 12, 13
methods, 9
mk.pt.constr, 19
NA, 3
plot, 7, 20
plot.cobs, 7, 20
plot.conreg(conreg-methods), 10
plot.default, 10
predict, 7, 9
predict.cobs, 3, 7, 21, 21
predict.conreg, 9
predict.conreg(conreg-methods), 10
print.cobs(cobs-methods), 6
print.conreg(conreg-methods), 10
qbsks2, 4, 13, 22
residuals, 9
residuals.cobs (cobs-methods), 6
residuals.conreg (conreg-methods), 10
rq.fit.sfn, 4, 12, 13, 23
rq.fit.sfnc, 4, 12, 13, 23

smooth.spline, 5, 15, 16
summary.cobs, 21
summary.cobs (cobs-methods), 6

USArmyRoofs, 24

xy.coords, 8