Package ‘uniReg’

October 12, 2022

Type Package

Title Unimodal Penalized Spline Regression using B-Splines

Version 1.1

Date 2016-06-05

Author Claudia Koellmann

Maintainer Claudia Koellmann <koellmann@statistik.tu-dortmund.de>

Imports DoseFinding, graphics, MASS, parallel, stats, mvtnorm,
quadprog, SEL, splines

Description Univariate spline regression. It is possible to add the shape constraint of unimodality and predefined or
self-defined penalties on the B-spline coefficients.

License GPL (>= 3)

NeedsCompilation no

Repository CRAN

Date/Publication 2016-06-06 06:39:59

R topics documented:

equiknots 2
plot.unireg 3
points.unireg 4
predict.unireg 5
print.unireg 6
unimat 7
unireg 8

Index **13**

equiknots — *Determine the knot sequence.*

Description

Determines $g+2k+2$ knots for the spline basis of degree k on the interval $[a, b]$. The g inner knots lie equidistant in $[a, b]$. If `coinc=TRUE`, the outer knots (k on each side of the interval) are placed coincident with a and b, otherwise the outer knots are also equidistant beyond $[a, b]$.

Usage

```
equiknots(a, b, g, k, coinc)
```

Arguments

`a`	Left numeric boundary of the spline interval.
`b`	Right numeric boundary of the spline interval.
`g`	A non-negative integer giving the number of inner knots.
`k`	A non-negative integer specifying the degree of the spline basis.
`coinc`	Logical indicating, if the outer knots should be coincident with the boundary knots or not. If `coinc=TRUE`, there are k coincident outer knots at a as well as at b.

Value

A numeric vector of length $g+2k+2$ with knot locations.

Author(s)

Claudia Koellmann

See Also

`unimat`, `unireg`

Examples

```
equiknots(0,5,3,3,TRUE)
equiknots(0,5,3,3,FALSE)
```

`plot.unireg` *Plot method for* `unireg` *objects.*

Description

Plotting a unimodal regression object.

Usage

```
## S3 method for class 'unireg'
plot(x, onlySpline=FALSE, type="l", xlab="x", ylab=NULL, col="black", ...)
```

Arguments

`x`	Object of class `"unireg"`, a result of `unireg`.
`onlySpline`	Logical indicating whether only the fitted spline or also the original data points should be plotted. Defaults to FALSE (plotting both).
`type`	Per default plotting type `"l"` is used for the fitted spline.
`xlab`	Per default the x-axis is labelled with "x".
`ylab`	If the user does not specify a label for the y-axis, that is when `ylab=NULL`(default), prespecified labels like "Fitted unimodal spline function" (depending on the constraint) are used.
`col`	Colour of the spline function to be plotted (default: black).
`...`	other parameters to be passed through to the generic `plot` function.

Details

This is a plot method for unimodal regression objects. The spline function is plotted using a grid of x-values equally spaced across the interval on which the spline is defined. The distance between the grid values is given by the minimal distance of the observed x-values (used for fitting) divided by 10.

Author(s)

Claudia Koellmann

See Also

`unireg`,`predict.unireg`,`points.unireg`

Examples

```
x <- sort(rep(0:5,20))
n <- length(x)
set.seed(41333)
func <- function(mu){rnorm(1,mu,0.05)}
y <- sapply(dchisq(x,3),func)

# fit with default settings
fit <- unireg(x, y, g=5)
# short overview of the fitted spline
fit

# plot of fitted spline with and without data
plot(fit, col="orange")
plot(fit, onlySpline=TRUE)
```

---

points.unireg *Points method for* unireg *objects.*

Description

Plotting a unimodal regression object into an existing plot.

Usage

```
## S3 method for class 'unireg'
points(x, type="l", ...)
```

Arguments

x	Object of class "unireg", a result of unireg.
type	Per default plotting type "l" is used for the fitted spline.
...	other parameters to be passed through to the generic points functions.

Details

This is a points method for unimodal regression objects. The spline function is plotted using a grid of x-values equally spaced across the interval on which the spline is defined. The distance between the grid values is given by the minimal distance of the observed x-values (used for fitting) divided by 10.

Author(s)

Claudia Koellmann

See Also

unireg,predict.unireg,plot.unireg

Examples

```
x <- sort(rep(0:5,20))
n <- length(x)
set.seed(41333)
func <- function(mu){rnorm(1,mu,0.05)}
y <- sapply(dchisq(x,3),func)

# plot of data
plot(jitter(x), y, xlab="x (jittered)")

# fit with default settings
fit <- unireg(x, y, g=5)
# short overview of the fitted spline
fit

# plot of true and fitted functions
plot(jitter(x), y, xlab="x (jittered)")
curve(dchisq(x,3), 0, 5, type="l", col="grey", lwd=2, add=TRUE)
points(fit, lwd=2, col="orange")
legend("bottomright", legend = c("true mean function",
        "difference penalized unimodal fit"),
    col=c("grey","orange"),lwd=c(2,2))
```

`predict.unireg` *Predict method for* `unireg` *objects.*

Description

Predicted values based on a unimodal regression object.

Usage

```
## S3 method for class 'unireg'
predict(object, newdata, ...)
```

Arguments

`object`	Object of class `'unireg'`, a result of `unireg`.
`newdata`	A numeric vector of values at which to evaluate the fitted spline function.
`...`	Further arguments (currently not used).

Details

`predict.unireg` produces predicted values by evaluating the fitted regression spline function at the values in `newdata`.

Value

Returns a numeric vector of predicted function values.

Author(s)

Claudia Koellmann

See Also

`unireg`,`plot.unireg`,`points.unireg`

Examples

```
x <- sort(rep(0:5,20))
n <- length(x)
set.seed(41333)
func <- function(mu){rnorm(1,mu,0.05)}
y <- sapply(dchisq(x,3),func)

# plot of data
plot(jitter(x), y, xlab="x (jittered)")

# fit with default settings
fit <- unireg(x, y, g=5)
# short overview of the fitted spline
fit

# prediction at interim values
predict(fit, c(1.5,2.5,3.5,4.5))
```

`print.unireg` *Print method for* `unireg` *objects.*

Description

Prints unimodal regression objects.

Usage

```
## S3 method for class 'unireg'
print(x, ...)
```

Arguments

`x`	Object of class `'unireg'`, a result of function `unireg`.
`...`	Further arguments (currently not used).

Details

Prints a short overview of a fitted unimodal regression object to the console, namely, the type of the fitted model (including degree of the spline and type of constraint and penalty), the coefficients and their mode location, the tuning parameter and the variance estimate.

Value

Invisibly returns the input x.

Author(s)

Claudia Koellmann

See Also

unireg,plot.unireg,points.unireg

Examples

```
x <- sort(rep(0:5,20))
n <- length(x)
set.seed(41333)
func <- function(mu){rnorm(1,mu,0.05)}
y <- sapply(dchisq(x,3),func)

# plot of data
plot(jitter(x), y, xlab="x (jittered)")

# fit with default settings
fit <- unireg(x, y, g=5)
# short overview of the fitted spline
fit
```

---

unimat *Create the matrix of unimodality constraints.*

Description

Returns a matrix C that can be used to specify linear constraints $Cb \geq 0$ to impose unimodality with mode at the m-th element on a numeric vector b of length p.

Usage

```
unimat(p, m)
```

Arguments

p	Integer (>=2) giving the length of the vector b.
m	Location of the mode within the vector b. Should be in integer between 1 and p.

Value

Matrix C with coefficients for the linear constraints.

Author(s)

Claudia Koellmann

See Also

`equiknots`, `unireg`

Examples

```
unimat(4,2)
unimat(5,3)
```

`unireg` *Fitting a unimodal penalized spline regression.*

Description

Function for fitting spline regressions to data. The fit can be constrained to be unimodal, inverse-unimodal, isotonic or antitonic and an arbitrary penalty on the B-spline coefficients can be used.

Usage

```
unireg(x, y, w=NULL, sigmasq=NULL, a=min(x), b=max(x), g=10, k=3,
constr=c("unimodal","none","invuni","isotonic","antitonic"),
penalty=c("diff", "none", "sigEmax", "self", "diag"), Om=NULL,
beta0=NULL, coinc=NULL, tuning=TRUE, abstol=0.01,vari=5,ordpen=2,
m=1:(g+k+1), allfits=FALSE, nCores=1)
```

Arguments

`x`	A numeric vector of x-values, length n. Contains at least $d = g + k + 1 \leq n$ distinct values.
`y`	A numeric vector of observed y-values of length n.
`w`	A positive numeric weight vector of length n. The weights do not have to sum to n, but will be transformed to do so internally. If `sigmasq` is given, `w` should be `NULL` (default).
`sigmasq`	Estimate(s) of the residual (co-)variance(s). Can be a positive numeric vector of length n, giving estimates for the variance at each of the x-values. If it is a vector of length 1, equal varainces across all x-values are assumed. If `sigmasq=NULL`, each x-value has to be appear at least twice and a global variance (same for all x-values) is estimated internally. If `sigmasq` is given, `w` should be `NULL`.
`a`	The left numeric boundary of the interval, on which the spline is defined. If `coinc=TRUE`, the spline is zero to the left of this value. By default `a` is equal to the minimal x-value.

`b`	The right numieric boundary of the interval, on which the spline is defined. If `coinc=TRUE`, the spline is zero to the right of this value. By default `b` is equal to the maximal *x*-value.
`g`	A non-negative integer giving the number of inner knots of the spline (default: 10).
`k`	A non-negative integer specifying the degree of the spline. By default a cubic spline (k = 3) is fitted.
`constr`	A character string specifying the shape constraint for the fit. Can be one of "unimodal" (default), "none", "invuni" (inverse-unimodal), "isotonic" or "antitonic".
`penalty`	A character string specifying, which penalty on the B-spline coefficients should be used. Possible choices are `"diff"` (default) for the differences penalty of order `ordpen`, `"none"` for no penalty, `"sigEmax"` for the sigmoid Emax penalty, `"self"` for a self-defined penalty and `"diag"` for a ridge penalty. For a self-defined penalty, `Om` and `beta0` have to be provided.
`Om`	If a self-defined penalty on the B-spline coefficients should be used, `Om` is the penalty matrix of dimension $d \times d$ and full rank d. Otherwise, `Om` should be `NULL` (default).
`beta0`	If a self-defined penalty on the B-spline coefficients should be used, `beta0` is the penalty vector of length *d*. Otherwise, `beta0` should be `NULL` (default).
`coinc`	Logical indicating, if the outer knots of the knot sequence should be coincident with the boundary knots or not? Default is `NULL` and altering has no effect, if a pre-defined penalty is used. If `penalty="self"`, it has to be specified.
`tuning`	Logical indicating, if the tuning parameter lambda should be optimized with (`tuning=TRUE`, default, computationally expensive) or without (`tuning=FALSE`) consideration of the shape constraint. Changing `tuning` has no effect, when `constr="none"` or `penalty="none"`.
`abstol`	The iterative estimation of the residual variance σ^2 and the coefficient vector stops after iteration ς, when $\lvert\hat{\sigma}^{(\varsigma)} - \hat{\sigma}^{(\varsigma-1)}\rvert$ is less than a positive numeric value `abstol` (default: 0.01) or when $\varsigma = 10$. If `sigmasq` is not `NULL`, the supplied value is used as starting value in this iteration scheme. There is no iterative estimation, if `abstol` is set to `NULL`.
`vari`	Variance parameter $sigma_v^2 > 0$ in the full-rank precision matrix of the prior for beta. By default 5.
`ordpen`	Order of the difference penalty (integer ≥ 0, default 2). Only effective, if `penalty="diff"`.
`m`	An integer vector specifying the modes of the coefficient vector which should be used for fitting, in explicit, a subset of *{1,...,d}*. This argument only has an effect if `constr="unimodal"` or `"invuni"`.
`allfits`	Logical indicating if the estimated coefficient vectors for all modes in `m` should be returned (`TRUE`) or only the one with minimal residual sum of squares (`FALSE`).
`nCores`	The integer number of cores used for parallelization. If `nCores=1`, there is no parallelization (default).

Details

This function combines implementations of the spline methods described in Koellmann et al. Given paired data $(x_1, y_1), ..., (x_n, y_n)$ it is possible to fit regression splines using the B-spline basis and the maximum likelihood approach. If the spline is unrestricted, the problem reduces to a simple linear regression problem. If the spline is restricted to be unimodal, inverse unimodal, isotonic or antitonic, this leads to a quadratic programming problem. If a penalty is used on the spline coefficients, the tuning parameter is chosen via restricted maximum likelihood (REML).

The data should contain repeated measurements at certain points on the *x*-axis (at least 2 for each point), so that a start estimate of the residual variance can be calculated. Then the function iterates between estimation of the spline coefficients and of the variance. Both estimates will be weighted, if weights are given. If there is only one measurement per *x*-value, the function expects an input in `sigmasq`, an estimate of the variance(s) used for weighted estimation (no further weights can be used).

If no penalty is used, the number of estimable B-spline coefficients, which is *d=g+k+1*, equals the number of distinct *x*-values. *g* and *k* have to be chosen accordingly.

Value

Returns an object of class "unireg", that is, a list containing the following components:

`x`	The (sorted) vector of *x*-values.
`y`	The input vector of *y*-values (sorted according to *x*).
`w`	The vector of weights used for fitting (sorted according to *x*).
`a`	The left boundary of the domain [a,b].
`b`	The right boundary of the domain [a,b].
`g`	The number `g` of inner knots.
`degree`	The degree `k` of the spline.
`knotsequence`	The sequence of knots (length *g+2k+2*) used for spline fitting.
`constr`	The constraint on the coefficients.
`penalty`	The type of penalty used.
`Om`	The penalty matrix.
`beta0`	The penalty vector.
`coinc`	The input parameter `coinc`.
`tuning`	The input parameter `tuning`.
`abstol`	The input value of `abstol`.
`vari`	The input variance parameter `vari`.
`ordpen`	The order of the difference penalty.
`coef`	The vector of estimated B-spline coefficients (corresponding to the mode with minimal RSS).
`fitted.values`	The fitted values at each *x*-value (corresponding to the mode with minimal RSS).
`lambdaopt`	The optimal tuning parameter found via REML (corresponding to the mode with minimal RSS).

`sigmasq`	The estimated residual variance. If the input for `abstol` was `NULL`, `sigmasq` equals its input value.
`variter`	The number ς of iterations used to estimate the spline coefficients and the variance.
`ed`	The effective degrees of freedom (corresponding to the mode with minimal RSS).
`modes`	The input vector `m` of modes.
`allcoefs`	A matrix of coefficient vectors (corresponding to the modes specified in `m`) or `NULL` (if `allfits=FALSE`)

Author(s)

Claudia Koellmann

References

Koellmann, C., Bornkamp, B., and Ickstadt, K. (2104), *Unimodal regression using Bernstein-Schoenberg splines and penalties*, Biometrics 70 (4), 783-793.

See Also

`unimat`, `equiknots`, `plot.unireg`, `points.unireg`, `print.unireg`, `predict.unireg`,

Examples

```
x <- sort(rep(0:5,20))
n <- length(x)
set.seed(41333)
func <- function(mu){rnorm(1,mu,0.05)}
y <- sapply(dchisq(x,3),func)

# plot of data
plot(jitter(x), y, xlab="x (jittered)")

# fit with default settings
fit <- unireg(x, y, g=5)
# short overview of the fitted spline
fit

# prediction at interim values
predict(fit, c(1.5,2.5,3.5,4.5))

# fit without penalty (we can use at most g=2 inner knots if k=3)
fit2 <- unireg(x, y, penalty="none", g=2)

# plot of fitted spline with or without data
plot(fit2)
plot(fit2, onlySpline=TRUE)

# fit without penalty and without constraint
# (does not differ from fit2 with constraint in this case)
fit3 <- unireg(x, y, penalty="none", g=2, constr="none")

# plot of true and fitted functions
plot(jitter(x), y, xlab="x (jittered)")
curve(dchisq(x,3), 0, 5, type="l", col="grey", lwd=2, add=TRUE)
points(fit, lwd=2)
points(fit2, col="blue", lwd=2)
points(fit3, col="red", lwd=2)
legend("bottomright", legend = c("true mean function",
        "difference penalized unimodal fit",
        "unpenalized fit (with and without constraint)"),
    col=c("grey","black","red"),lwd=c(2,2,2))

# estimated variance
fit$sigmasq
fit2$sigmasq

## Not run:
# fit with isotonic, antitonic and inverse-unimodal constraint (just for completeness)
fit4 <- unireg(x,y,constr="antitonic",g=5)
fit5 <- unireg(x,y,constr="isotonic",g=5)
fit6 <- unireg(x,y,constr="invuni",g=5)

points(fit4,col="orange",lwd=2)
points(fit5,col="brown",lwd=2)
points(fit6,col="yellow",lwd=2)

# suppose only aggregated data had been given
means <- c(mean(y[1:20]), mean(y[21:40]), mean(y[41:60]), mean(y[61:80]),
           mean(y[81:100]), mean(y[101:120]))
sigmasq <- c(sd(y[1:20]),sd(y[21:40]),sd(y[41:60]),sd(y[61:80]),sd(y[81:100]),sd(y[101:120]))^2

# unimodal fit with differences penalty
fit7 <- unireg(x=unique(x), y=means, g=5, w=NULL, sigmasq=sigmasq, abstol=NULL)
plot(unique(x), means, pch=19, ylim=range(y))
curve(dchisq(x,3), 0, 5, type="l", col="grey", lwd=2, add=TRUE)
points(fit7, type="l", col="green", lwd=2)
legend("bottomright", legend = c("true mean function", "observed mean values",
    "diff. penalized unimodal fit for means"),
  col=c("grey","black","green"), lty=c(1,NA,1), lwd=c(2,0,2), pch=c(NA,19,NA))

## End(Not run)
```

Index

∗ **models**
  plot.unireg, 3
  points.unireg, 4
  predict.unireg, 5
  print.unireg, 6
  unireg, 8
∗ **nonparametric**
  plot.unireg, 3
  points.unireg, 4
  predict.unireg, 5
  print.unireg, 6
  unireg, 8
∗ **regression**
  plot.unireg, 3
  points.unireg, 4
  predict.unireg, 5
  print.unireg, 6
  unireg, 8

equiknots, 2, *8*, *11*

plot.unireg, 3, *4*, *6*, *7*, *11*
points.unireg, *3*, 4, *6*, *7*, *11*
predict.unireg, *3*, *4*, 5, *11*
print.unireg, 6, *11*

unimat, *2*, 7, *11*
unireg, *2–8*, 8