Package ‘MonoPoly’
April 5, 2016

Type Package
Title Functions to Fit Monotone Polynomials
Version 0.3-8
Date 2016-04-04
Description Functions for fitting monotone polynomials to data.
License GPL (>= 2)
Depends R (>= 3.1.0), quadprog
LazyData yes
Encoding UTF-8
NeedsCompilation yes
Author Berwin A. Turlach [aut, cre],
Kevin Murray [ctb]
Maintainer Berwin A. Turlach <berwin.Turlach@gmail.com>
Repository CRAN
Date/Publication 2016-04-05 08:14:19

R topics documented:

  coef.monpol                          ........................................ 2
  curvPol                             ........................................ 3
  evalPol                             ........................................ 4
  fitted.monpol                       ........................................ 5
  hawkins                             ........................................ 5
  ismonotone                          ........................................ 6
  model.matrix.monpol                 ...................................... 7
  monpol                              ........................................ 8
  monpol.control                      .................................... 10
  monpol.fit                          ........................................ 11
  predict.monpol                      ...................................... 13
  print.monpol                        ...................................... 13
  residuals.monpol                    .................................... 14
  w0                                  .......................................... 15
  w2                                  .......................................... 15
 coef.monpol

Index 17

---

**coef.monpol**  
*Extract Model Coefficients*

---

**Description**

coef method for ‘monpol’ objects.

**Usage**

```r  
## S3 method for class 'monpol'
coef(object, scale = c("original", "fitted"), type = c("beta", "monpar"), ...)
```

**Arguments**

- `object`: A ‘monpol’ object.
- `scale`: Extract coefficients on the original scale of the data or on the scale used during fitting.
- `type`: Extract coefficients in the ‘beta’ parameterisation of the polynomial or for the monotone parameterisation used in the algorithm.
- `...`: Additional optionals arguments. At present no optional arguments are used.

**Details**

This is the coef method for objects inheriting from class "monpol".

**Value**

Coefficients extracted from the model object `object`.

**Author(s)**

Berwin A Turlach
curvPol

Evaluating the Curvature of Polynomials

Description

Function to evaluate the curvature of polynomials

Usage

curvPol(x, beta)

Arguments

x numerical values at which to evaluate the curvature of polynomials, can be provided in a vector, matrix, array or data frame
beta numerical vector containing the coefficient of the polynomial

Value

The result of evaluating the curvature of the polynomial at the values in x, returned in the same dimension as x has.

Author(s)

Berwin A Turlach

Examples

beta <- c(1,2,1)
x <- 0:10
curvPol(x, beta)
str(curvPol(x, beta))

x <- cbind(0:10, 10:0)
curvPol(x, beta)
str(curvPol(x, beta))

x <- data.frame(x=0:10, y=10:0)
curvPol(x, beta)
str(curvPol(x, beta))
Description
Function to evaluate polynomials in a numerical robust way using the Horner scheme

Usage
\texttt{evalPol(x, beta)}

Arguments
\begin{itemize}
  \item \texttt{x} numerical values at which to evaluate polynomials, can be provided in a vector, matrix, array or data frame
  \item \texttt{beta} numerical vector containing the coefficient of the polynomial
\end{itemize}

Value
The result of evaluating the polynomial at the values in \texttt{x}, returned in the same dimension as \texttt{x} has.

Author(s)
Berwin A Turlach

Examples
\begin{verbatim}
beta <- c(1,2,1)
x <- 0:10
evalPol(x, beta)
str(evalPol(x, beta))

x <- cbind(0:10, 10:0)
evalPol(x, beta)
str(evalPol(x, beta))

x <- data.frame(x=0:10, y=10:0)
evalPol(x, beta)
str(evalPol(x, beta))
\end{verbatim}
Description

fitted method for 'monpol' objects.

Usage

```r
## S3 method for class 'monpol'
fitted(object, scale = c("original", "fitted"), ...)
```

Arguments

- `object`: A 'monpol' object.
- `scale`: Extract fitted values on the original scale of the data or on the scale used during fitting.
- `...`: Additional optionals arguments. At present no optional arguments are used.

Details

This is the fitted method for objects inheriting from class "monpol".

Value

Fitted values extracted from the model object `object`.

Author(s)

Berwin A Turlach

hawkins

Description

This data gives x and y variables for the data published in Hawkins' 1994 article. This data was originally simulated from a standard cubic polynomial with equally spaced x values between -1 and 1.

Format

A data frame with 50 simulated observations on the following 2 variables.

- `y`: a numeric vector
- `x`: a numeric vector
ismonotone

References


Examples

data(hawkins)

---

ismonotone

*Check whether a polynomial is monotone*

Description

Function to check whether a polynomial is monotone over a given interval.

Usage

```r
ismonotone(object, ...)  
```

## S3 method for class 'monpol'
```
ismonotone(object, a = -Inf, b = Inf, EPS = 1e-06, ...)
```

## Default S3 method:
```
ismonotone(object, a = -Inf, b = Inf, EPS = 1e-06, ...)
```

Arguments

- **object**: Either an object of class ‘monpol’ or a numeric vector containing the coefficient of the polynomial.
- **a**: Lower limit of the interval over which the polynomial should be monotone.
- **b**: Upper limit of the interval over which the polynomial should be monotone.
- **EPS**: Numerical precision, values with absolute value smaller than EPS are treated as zero.
- **...**: Further arguments passed to or from other methods.

Value

TRUE or FALSE depending on whether the polynomial is monotone over (a,b) or not.

Note that due to numerical precision issues it is possible that a polynomial that should be monotone is declared to be not monotone.

Author(s)

Kevin Murray and Berwin A Turlach
Examples

```r
fit <- monpol(y~x, w0)
ismonotone(fit)

beta <- c(1,0,2)  ## the polynomial 1 + 2*x^2
ismonotone(beta)
ismonotone(beta, a=0)
ismonotone(beta, b=0)
```

Description

`model.matrix` creates a design (or model) matrix for ‘monpol’ objects.

Usage

```r
## S3 method for class 'monpol'
model.matrix(object, scale = c("original", "fitted"), ...)
```

Arguments

- `object` A `monpol` object.
- `scale` Create design matrix on the original scale of the data or on the scale used during fitting.
- `...` Additional optionals arguments. At present no optional arguments are used.

Details

This is the `model.matrix` method for objects inheriting from class "monpol".

Value

Design matrix created from the model object object.

Author(s)

Berwin A Turlach
Monotone Polynomials

Description

Determine the least-squares estimates of the parameters of a monotone polynomial

Usage

monpol(formula, data, subset, weights, na.action,
  degree = 3, K, start,
  a = -Inf, b=Inf,
  trace = FALSE, plot.it = FALSE,
  control = monpol.control(),
  algorithm = c("Full", "Hawkins", "BCD", "CD1", "CD2"),
  ptype = c("SOS", "Elphinstone", "EHH", "Penttila"),
  ctype = c("cge0", "c2"),
  monotone,
  model=FALSE, x=FALSE, y=FALSE)

Arguments

formula an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted.
data an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which monpol is called.
subset an optional vector specifying a subset of observations to be used in the fitting process.
weights an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector.
na.action a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The ‘factory-fresh’ default is na.omit. Another possible value is na.exclude which can be useful.
degree positive integer, a polynomial with highest power equal to degree will be fitted to the data.
K non-negative integer, a polynomial with highest power \(2K + 1\) will be fitted to the data.
start optional starting value for the iterative fitting.
a,b polynomial should be monotone on the interval from a to b. If either parameter is finite, parameterisation “SOS” has to be used.
trace print out information about the progress of the iterative fitting at the start and then every trace iterations.
plot.it

control

algorithm

ptype

ctype

monotone

model, x, y

Details

A `monpol` object is a type of fitted model object. It has methods for the generic function `coef, fitted, formula, logLik, model.matrix, predict, print, residuals`.

The parameterisation type “SOS” with the “Full” algorithm is currently the recommended fitting procedure and is discussed in the 2016 paper in ‘References’. For this parameterisation the argument `ctype` is ignored.

The “Hawkins” algorithm is also recommended and discussed in both papers in the ‘References’.

The parameterisations “Elphinstone”, “EHH” and “Pentilla”, for which the argument “ctype” defines a further variation of parameterisation, work together with algorithms “Full”, “BCD”, “CD1” and “CD2”. These parameterisations and algorithms are discussed in the 2013 paper in ‘References’.

Value

`monpol` returns an object of class `"monpol"`

Author(s)

Berwin A Turlach

References


Examples

```r
monpol(y~x, w=0)
```
monpol.control  Control the Iterations in monpol

Description
Allow the user to set some characteristics of the monpol monotone polynomial fitting algorithm.

Usage
monpol.control(maxiter = 1000, tol = 1e-05,
               tol1=1e-10, tol2=1e-07, tolqr=1e-07)

Arguments
maxiter A positive integer specifying the maximum number of iterations allowed, used in all algorithms.
tol  A positive numeric value specifying an absolute tolerance for determining whether entries in the gradient are zero for algorithms ‘Full’, ‘BCD’, ‘CD1’ and ‘CD2’.
tol1 A positive numeric value, used in algorithm ‘Hawkins’. Any number not smaller than -tol1 is deemed to be non-negative.
tol2 A positive numeric value, used in algorithm ‘Hawkins’. Any number whose absolute value is smaller than tol2 is taken to be zero.
tolqr A positive numeric value, used in algorithm ‘Hawkins’ as tolerance for the QR factorisation of the design matrix.

Value
A list with exactly five components:
maxiter
tol
tol1
tol2
tolqr
with meanings as explained under ‘Arguments’.

Author(s)
Berwin A Turlach

See Also
monpol, monpol.fit, qr
monpol.fit

Examples

.monpol.control(maxiter = 2000)
.monpol.control(tolqr = 1e-10)

monpol.fit  Monotone Polynomials

Description

This is the basic computing engine called by monpol used to fit monotonic polynomials. These should usually not be used directly unless by experienced users.

Usage

monpol.fit(x, y, w, K=1, start, trace = FALSE, plot.it = FALSE,
control = monpol.control(),
algorithm = c("Full", "Hawkins", "BCD", "CD1", "CD2"),
ptype = c("Elphinstone", "EHH", "Penttila"),
cctype = c("cge0", "c2")
SOSpol.fit(x, y, w = NULL, deg.is.odd, K, start, a, b,
monotone = c("increasing", "decreasing"),
trace = FALSE, plot.it = FALSE, type,
control = monpol.control())

Arguments

x            vector containing the observed values for the regressor variable.
y            vector containing the observed values for the response variable; should be of same length as x.
w            optional vector of weights; should be of the same length as x if specified.
deg.is.odd, K  "deg.is.odd" is a logical, "K" is a non negative integer. If "deg.is.odd" is TRUE then a polynomial with highest power $2K + 1$ will be fitted to the data, otherwise the highest order will be $2K$.
start        optional starting value for the iterative fitting.
a, b, type   polynomial should be monotone on the interval from a to b; "type" should be 0 if neither of the boundaries is finite, 1 if a if finite but not b and 2 if both boundaries are finite.
monotone     force the desired monotonicity in case the default choice is wrong.
trace        print out information about the progress of the iterative fitting at the start and then every trace iterations.
plot.it      plot the data and initial fit, then plot current fit every plot.it iterations.
control      settings that control the iterative fit; see monpol.control for details.
algorithm    algorithm to be used; see monpol for details.
ptype        parameterisation to be used; see monpol for details.
cctype       parameterisation to be used; see monpol for details.
Value

a list with components

*par*  the fitted parameters.
*grad* the gradient of the objective function at the fitted parameters.
*beta* the coefficients of the fitted polynomial in the ‘beta’ parameterisation; on the fitted scale.
*RSS* the value of the objective function; on the fitted scale.
*niter* number of iterations.
*converged* indicates whether algorithm has converged.
*pctype* input parameter ptype.
*cptype* input parameter cptype.
*beta.raw* the coefficients of the fitted polynomial in the ‘beta’ parameterisation; on the original scale.
*fitted.values* the fitted values; on the fitted scale.
*residuals* the residuals; on the fitted scale.
*K* input parameter K.
*minx* the minimum value in the vector x.
*sclx* the difference between the maximum and minimum values in the vector x.
*miny* the minimum value in the vector y.
*scly* the difference between the maximum and minimum values in the vector y.
*algorithm* input parameter algorithm.

Author(s)

Berwin A Turlach

References


See Also

*monpol* which you should use for fitting monotonic polynomials unless you know better.
predict.monpol

Predicting from Monotone Polynomial Fits

Description

predict.monpol produces predicted values, obtained by evaluating the monotone polynomial in the frame newdata.

Usage

## S3 method for class 'monpol'
predict(object, newdata, scale = c("original", "fitted"), ...)

Arguments

- object: A 'monpol' object.
- newdata: A named list or data frame in which to look for variables with which to predict. If newdata is missing the fitted values at the original data points are returned.
- scale: Predict values on the original scale of the data or on the scale used during fitting. Data in newdata is assumed to be on the indicated scale.
- ...: Additional optionals arguments. At present no optional arguments are used.

Details

This is the predict method for objects inheriting from class "monpol".

Value

predict.monpol produces a vector of predictions.

Author(s)

Berwin A Turlach

print.monpol

Printing Monotone Polynomials

Description

print method for 'monpol' objects.

Usage

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

...
Arguments

x  A 'monpol' object.
digits  minimal number of significant digits, see \texttt{print.default}.
...  Additional optionals arguments. At present only those addition arguments for \texttt{coef.monpol} are used.

Details

This is the \texttt{print} method for objects inheriting from class "monpol".

Value

x returned invisibly.

Author(s)

Berwin A Turlach

---

\texttt{residuals.monpol} \quad \textit{Extract Model Residuals}

Description

\texttt{residuals} method for 'monpol' objects.

Usage

\begin{verbatim}
## S3 method for class 'monpol'
residuals(object, scale = c("original", "fitted"), ...)
\end{verbatim}

Arguments

object  A 'monpol' object.
scale  Extract residuals on the original scale of the data or on the scale used during fitting.
...  Additional optionals arguments. At present no optional arguments are used.

Details

This is the \texttt{residuals} method for objects inheriting from class "monpol".

Value

Residuals extracted from the model object object.

Author(s)

Berwin A Turlach
Simulated \( w_0 \) data used in Murray et al. (2013)

**Description**

This data set gives simulated data from the function

\[
y = 0.1x^3 + e
\]

for \( e \sim N(0, 0.01^2) \) and \( x \) evenly spaced between -1 and 1.

**Format**

A data frame with 21 observations on the following 2 variables.

- **y** a numeric vector
- **x** a numeric vector

**Source**


**Examples**

```r
str(w0)
plot(y~x, w0)
monpol(y~x, w0)
```

Simulated \( w_2 \) data used in Murray et al. (2013)

**Description**

Simulated data from the function

\[
y_{ij} = 4\pi - x_i + \cos(x_i - \frac{\pi}{2}) + e_{ij}
\]

for \( x_i = 0, 1, \ldots, 12 \); \( n_i = 5 \) for \( i = 0 \) and \( n_i = 3 \) otherwise; \( e_{ij} \sim N(0, 0.5^2) \)

**Format**

A data frame with 41 observations on the following 2 variables.

- **y** a numeric vector
- **x** a numeric vector
Source

Examples
```r
str(w2)
plot(y~x, w2)
monpol(y~x, w2)
monpol(y~x, w2, K=2)
```
Index

*Topic datasets
  hawkins, 5
  w0, 15
  w2, 15

*Topic models
  coef.monpol, 2
  fitted.monpol, 5
  model.matrix.monpol, 7
  monpol, 8
  monpol.control, 10
  monpol.fit, 11
  predict.monpol, 13
  residuals.monpol, 14

*Topic nonlinear
  monpol, 8
  monpol.fit, 11
  predict.monpol, 13

*Topic print
  print.monpol, 13

*Topic regression
  coef.monpol, 2
  curvPol, 3
  evalPol, 4
  fitted.monpol, 5
  monpol, 8
  monpol.control, 10
  monpol.fit, 11
  predict.monpol, 13
  residuals.monpol, 14

*Topic utilities
  curvPol, 3
  evalPol, 4

*Topic utilities
  ismonotone, 6

as.data.frame, 8

class, 9
coef, 2, 9
coef.monpol, 2, 14

curvPol, 3
evalPol, 4
fitted, 5, 9
fitted.monpol, 5
fitted.values.monpol(fitted.monpol), 5
formula, 8, 9
hawkins, 5
ismonotone, 6
logLik, 9
model.matrix, 7, 9
model.matrix.monpol, 7
monpol, 8, 10–12
monpol.control, 9, 10, 11
monpol.fit, 10, 11
na.exclude, 8
na.fail, 8
na.omit, 8
options, 8
predict, 9, 13
predict.monpol, 13
print, 9, 14
print.default, 14
print.monpol, 13
qr, 10
resid.monpol(residuals.monpol), 14
residuals, 9, 14
residuals.monpol, 14
SOSpol.fit(monpol.fit), 11
w0, 15
w2, 15