

Package ‘GeDS’

July 19, 2024

Type Package

Title Geometrically Designed Spline Regression

Version 0.2.3

Date 2024-07-19

Author Dimitrina S. Dimitrova <D.Dimitrova@city.ac.uk>,
Emilio S. Guillen <Emilio.Saenz-Guillen@bayes.city.ac.uk>,
Vladimir K. Kaishev <V.Kaishev@city.ac.uk>,
Andrea Lattuada <andrea.lattuada@hotmail.com> and
Richard J. Verrall <R.J.Verrall@city.ac.uk>

Maintainer Emilio S. Guillen <Emilio.Saenz-Guillen@bayes.city.ac.uk>

Description Spline Regression, Generalized Additive Models, and Component-wise Gradient Boosting, utilizing Geometrically Designed (GeD) Splines. GeDS regression is a non-parametric method inspired by geometric principles, for fitting spline regression models with variable knots in one or two independent variables. It efficiently estimates the number of knots and their positions, as well as the spline order, assuming the response variable follows a distribution from the exponential family. GeDS models integrate the broader category of Generalized (Non-)Linear Models, offering a flexible approach to modeling complex relationships. A description of the method can be found in Kaishev et al. (2016) <doi:10.1007/s00180-015-0621-7> and Dimitrova et al. (2023) <doi:10.1016/j.amc.2022.127493>. Further extending its capabilities, GeDS's implementation includes Generalized Additive Models (GAM) and Functional Gradient Boosting (FGB), enabling versatile multivariate predictor modeling, as discussed in the forthcoming work of Dimitrova et al. (2024).

License GPL-3

URL <https://github.com/emilioluissaenzguillen/GeDS>

BugReports <https://github.com/emilioluissaenzguillen/GeDS/issues>

LazyData TRUE

Depends R (>= 3.0.1), Rcpp (>= 0.12.1), splines, stats, utils, Matrix,
methods, Rmpfr

LinkingTo Rcpp

RoxygenNote 7.3.2

Encoding UTF-8

Imports doFuture, doParallel, doRNG, foreach, future, MASS, mboost,
parallel, plot3D, TH.data

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-07-19 14:30:02 UTC

Contents

GeDS-package	3
BaFe2As2	5
BivariateFitters	6
bl_imp.GeDSboost	8
coalMining	9
coef.GeDS	10
coef.GeDSboost,gam	12
crossv_GeDS	13
Derive	13
deviance.GeDS	14
EWmortality	15
f	16
formula.GeDS	17
GeDS-class	18
GeDSboost-class	20
GeDSgam-class	22
GGeDS	23
Integrate	29
IRLSfit	31
knots.GeDS	33
lines,GeDS-method	34
NGeDS	36
NGeDSboost	40
NGeDSgam	45
plot,GeDS-method	48
plot.GeDSboost	51
PPolyRep	52
predict.GeDS	53
predict.GeDSboost,gam	55
print.GeDS	55
SplineReg	56
UnivariateFitters	60
visualize_boosting	63

Index

65

GeDS-package

GeDS

Description

Geometrically Designed Splines (GeDS) regression is a non-parametric method inspired by geometric principles, which is designed for fitting spline predictor models with variable knots. This method falls within the domain of generalized non-linear models (GNM), which include generalized linear models (GLM) as a special case. GeDS regression is fitted based on a sample of N observations of a response variable y , dependent on a set of (currently up to two) covariates, assuming y has a distribution from the exponential family. In addition, GeDS methodology is implemented both in the context of Generalized Additive Models (GAM) and Functional Gradient Boosting (FGB). On the one hand, GAM consist of an additive modeling technique where the impact of the predictor variables is captured through smooth (GeDS, in this case) functions. On the other hand, GeDS incorporates gradient boosting machine learning technique by implementing functional gradient descent algorithm to optimize general risk functions utilizing component-wise GeDS estimates.

Details

The GeDS estimation method is based on: first constructing a piecewise linear fit (spline fit of order 2) at stage A which captures the shape of the data and; second approximating this fit with shape preserving (variation diminishing) spline fits of higher orders 3, 4, . . . (i.e., degrees 2, 3, . . .) at stage B. As a result of this, GeDS estimates the number and location of the knots and the order of the spline fit in a fast and efficient way.

The GeDS method was originally developed by Kaishev et al. (2016) assuming the response variable y to be normally distributed and a corresponding *Mathematica* code was provided.

The GeDS method was extended by Dimitrova et al. (2023) to cover any distribution from the exponential family. The **GeDS** R package presented here includes an enhanced R implementation of the original Normal GeDS *Mathematica* code due to Kaishev et al. (2016), implemented as the [NGeDS](#) function and a generalization of it in the function [GGeDS](#) which covers the case of any distribution from the exponential family.

The **GeDS** package allows also to fit two dimensional response surfaces and to construct multivariate (predictor) models with a GeD spline component and a parametric component (see the functions [f](#), [formula](#), [NGeDS](#) and [GGeDS](#) for details).

Dimitrova et al. (2024) have recently made significant enhancements to the **GeDS** methodology, to incorporate generalized additive models (GAM) and functional gradient boosting (FGB). On the one hand, generalized additive models are encompassed by implementing the *local-scoring* algorithm using normal GeD splines (i.e., [NGeDS](#)) as function smoothers within the *backfitting* iterations. This is implemented via the function [NGeDSgam](#). On the other hand, the **GeDS** package incorporates the functional gradient descent algorithm by utilizing normal GeD splines (i.e., [NGeDS](#)) as base learners. This is implemented via the function [NGeDSboost](#).

The outputs of both [NGeDS](#) and [GGeDS](#) functions are [GeDS-class](#) objects, while the outputs of [NGeDSgam](#) and [NGeDSboost](#) are [GeDSgam-class](#) and [GeDSboost-class](#) objects, respectively. As described in Kaishev et al. (2016), Dimitrova et al. (2023) and Dimitrova et al. (2024), the final

GeDS fit is the one whose order is chosen according to a strategy described in stage B of the algorithm. However, `GeDS-class`, `GeDSgam-class` and `GeDSboost-class` objects contain second, third and fourth order spline fits and the user has the possibility to choose among them.

The **GeDS** package also includes some datasets where GeDS regression proves to be very efficient and some user friendly functions that are designed to easily extract required information. Several methods are also provided to handle GeDS, GAM-GeDS and FGB-GeDS output results (see `GeDS-class`, `GeDSgam-class` and `GeDSboost-class`, respectively).

Throughout this document, we use the terms GeDS predictor model, GeDS regression and GeDS fit interchangeably.

Please report any issue arising or bug in the code to <emilio.saenz-guillen@bayes.city.ac.uk>.

Package: GeDS
Version: 0.2.0
Date: 2024-01-28
License: GPL-3

Author(s)

Dimitrina S. Dimitrova <D.Dimitrova@city.ac.uk>, Emilio S. Guillen <emilio.saenz-guillen@bayes.city.ac.uk>, Vladimir K. Kaishev <V.Kaishev@city.ac.uk>, Andrea Lattuada <Andrea.Lattuada@unicatt.it> and Richard J. Verrall <R.J.Verrall@city.ac.uk>

References

Kaishev, V.K., Dimitrova, D.S., Haberman, S., & Verrall, R.J. (2016). Geometrically designed, variable knot regression splines. *Computational Statistics*, **31**, 1079–1105.

DOI: [doi:10.1007/s0018001506217](https://doi.org/10.1007/s0018001506217)

Dimitrova, D. S., Kaishev, V. K., Lattuada, A. and Verrall, R. J. (2023). Geometrically designed variable knot splines in generalized (non-)linear models. *Applied Mathematics and Computation*, **436**.

DOI: [doi:10.1016/j.amc.2022.127493](https://doi.org/10.1016/j.amc.2022.127493)

Dimitrova, D. S., Guillen, E. S. and Kaishev, V. K. (2024). **GeDS**: An R Package for Regression, Generalized Additive Models and Functional Gradient Boosting, based on Geometrically Designed (GeD) Splines. *Manuscript submitted for publication*.

See Also

Useful links:

- <https://github.com/emilioluissaezguillen/GeDS>
- Report bugs at <https://github.com/emilioluissaezguillen/GeDS/issues>

BaFe₂As₂*Barium-Ferrum-Arsenide Powder Diffraction Data*

Description

This dataset contains the results of a neutron diffraction experiment on Barium-Ferrum-Arsenide (BaFe₂As₂) powder carried out by Kimber et al. (2009) and used in Kaishev et al. (2016). The neutron diffraction intensity was measured at 1,151 different dispersion angles in order to model the diffraction profile.

Usage

```
data(BaFe2As2)
```

Format

A data frame with 1151 cases and 2 variables:

angle the dispersion angle, viewed as the independent variable.

intensity the neutron diffraction intensity, viewed as the response variable.

Source

openaccess.city.ac.uk

References

Kimber, S.A.J., Kreyssig, A., Zhang, Y.Z., Jeschke, H.O., Valenti, R., Yokaichiya, F., Colombier, E., Yan, J., Hansen, T.C., Chatterji, T., McQueeney, R.J., Canfield, P.C., Goldman, A.I. and Argyriou, D.N. (2009). Similarities between structural distortions under pressure and chemical doping in superconducting BaFe₂As₂. *Nat Mater*, **8**, 471–475.

Kaishev, V.K., Dimitrova, D.S., Haberman, S. and Verrall, R.J. (2016). Geometrically designed, variable knot regression splines. *Computational Statistics*, **31**, 1079–1105.
DOI: [doi:10.1007/s0018001506217](https://doi.org/10.1007/s0018001506217)

Examples

```
## Not run:  
# to load the data  
data('BaFe2As2')  
  
# fit a GeDS regression and produce a simple plot of the result. See ?NGeDS  
# c.f. Kaishev et al. (2016), section 4.2  
(Gmod <- NGeDS(intensity ~ f(angle), data = BaFe2As2, beta = 0.6, phi = 0.99,  
               q = 3, show.iters = T))  
plot(Gmod)  
  
## End(Not run)
```

Description

These are computing engines called by [NGeDS](#), needed for the underlying fitting procedures.

Usage

```
BivariateFitter(  
  X,  
  Y,  
  Z,  
  W,  
  weights = rep(1, length(X)),  
  Indicator,  
  beta = 0.5,  
  phi = 0.99,  
  min.intknots = 0,  
  max.intknots = 300,  
  q = 2,  
  Xextr = range(X),  
  Yextr = range(Y),  
  show.iters = TRUE,  
  tol = as.double(1e-12),  
  stoptype = c("SR", "RD", "LR"),  
  higher_order = TRUE,  
  Xintknots = NULL,  
  Yintknots = NULL  
)
```

```
GenBivariateFitter(  
  X,  
  Y,  
  Z,  
  W,  
  family = family,  
  weights = rep(1, length(X)),  
  Indicator,  
  beta = 0.5,  
  phi = 0.5,  
  min.intknots = 0,  
  max.intknots = 300,  
  q = 2,  
  Xextr = range(X),  
  Yextr = range(Y),  
  show.iters = TRUE,
```

```

    tol = as.double(1e-12),
    stoptype = c("SR", "RD", "LR"),
    higher_order = TRUE
)

```

Arguments

X	a numeric vector containing N sample values of the first independent variable chosen to enter the spline regression component of the predictor model.
Y	a numeric vector containing N sample values of the second independent variable chosen to enter the spline regression component of the predictor model.
Z	a vector of size N containing the observed values of the response variable.
W	a design matrix with N rows containing other covariates selected to enter the parametric component of the predictor model (see formula). If no such covariates are selected, it is set to NULL by default.
weights	an optional vector of size N of ‘prior weights’ to be put on the observations in the fitting process in case the user requires weighted GeDS fitting. It is NULL by default.
Indicator	contingency table of X and Y.
beta	numeric parameter in the interval $[0, 1]$ tuning the knot placement in stage A of GeDS. See the description of NGeDS or GGeDS .
phi	numeric parameter in the interval $[0, 1]$ specifying the threshold for the stopping rule (model selector) in stage A of GeDS. See also stoptype and details in the description of NGeDS or GGeDS .
min.intknots	optional parameter allowing the user to set a minimum number of internal knots required. By default equal to zero.
max.intknots	optional parameter allowing the user to set a maximum number of internal knots to be added by the GeDS estimation algorithm. By default equal to the number of internal knots κ for the saturated GeDS model (i.e. $\kappa = N - 2$).
q	numeric parameter which allows to fine-tune the stopping rule of stage A of GeDS, by default equal to 2. See details in the description of NGeDS or GGeDS .
Xextr	boundary knots in the X direction. By default equal to the range of X.
Yextr	boundary knots in the Y direction. By default equal to the range of Y.
show.iters	logical variable indicating whether or not to print information at each step. By default equal to FALSE.
tol	numeric value indicating the tolerance to be used in checking whether two knots should be considered different during the knot placement steps in stage A.
stoptype	a character string indicating the type of GeDS stopping rule to be used. It should be either "SR", "RD" or "LR", partial match allowed. See details of NGeDS or GGeDS .
higher_order	a logical that defines whether to compute the higher order fits (quadratic and cubic) after stage A is run. Default is TRUE.
Xintknots	vector of starting internal knots in the X direction. Default is NULL.

`Yintknots` vector of starting internal knots in the Y direction. Default is NULL.

`family` a description of the error distribution and link function to be used in the model. This can be a character string naming a family function (e.g. "gaussian"), the family function itself (e.g. `gaussian`) or the result of a call to a family function (e.g. `gaussian()`). See [family](#) for details on family functions.

Value

A [GeDS-Class](#) object, but without the `Formula`, `extcall`, `terms` and `znames` slots.

References

Dimitrova, D. S., Kaishev, V. K., Lattuada, A. and Verrall, R. J. (2023). Geometrically designed variable knot splines in generalized (non-)linear models. *Applied Mathematics and Computation*, **436**.
DOI: [doi:10.1016/j.amc.2022.127493](https://doi.org/10.1016/j.amc.2022.127493)

See Also

[NGeDS](#) and [UnivariateFitters](#).

bl_imp.GeDSboost *Base Learner Importance for GeDSboost objects*

Description

This function calculates the in-bag mean squared error (MSE) reduction ascribable to each of the base-learners with regards to the final prediction of the component-wise gradient boosted model encapsulated in a [GeDSboost-Class](#) object. Essentially, it measures the decrease in MSE attributable to each base-learner for every time it is selected across the boosting iterations, and aggregates them. This provides a measure on how much each base-learner contributes to the overall improvement in the model's accuracy, as reflected by the decrease in MSE. This function is adapted from [varimp](#) and is compatible with the available [mboost-package](#) methods for [varimp](#), including `plot`, `print` and `as.data.frame`.

Usage

```
## S3 method for class 'GeDSboost'
bl_imp(object, boosting_iter_only = FALSE, ...)
```

Arguments

`object` an object of class [GeDSboost-Class](#).

`boosting_iter_only` logical value, if TRUE then base-learner in-bag risk reduction is only computed across boosting iterations, i.e., without taking into account the initial learner.

... potentially further arguments.

Details

See `varimp` for details.

Value

An object of class `varimp` with available `plot`, `print` and `as.data.frame` methods.

References

Hothorn T., Buehlmann P., Kneib T., Schmid M. and Hofner B. (2022). `mboost`: Model-Based Boosting. R package version 2.9-7, <https://CRAN.R-project.org/package=mboost>.

Examples

```
library(GeDS)
library(TH.data)
set.seed(290875)
data("bodyfat", package = "TH.data")
data = bodyfat
Gmodboost <- NGeDSboost(formula = DEXfat ~ f(hipcirc) + f(kneebreadth) + f(anthro3a),
                        data = data, initial_learner = FALSE)
MSE_Gmodboost_linear <- mean((data$DEXfat - Gmodboost$predictions$pred_linear)^2)
MSE_Gmodboost_quadratic <- mean((data$DEXfat - Gmodboost$predictions$pred_quadratic)^2)
MSE_Gmodboost_cubic <- mean((data$DEXfat - Gmodboost$predictions$pred_cubic)^2)

# Print MSE
cat("\n", "MEAN SQUARED ERROR", "\n",
    "Linear NGeDSboost:", MSE_Gmodboost_linear, "\n",
    "Quadratic NGeDSboost:", MSE_Gmodboost_quadratic, "\n",
    "Cubic NGeDSboost:", MSE_Gmodboost_cubic, "\n")

# Base Learner Importance
bl_imp <- bl_imp(Gmodboost)
print(bl_imp)
plot(bl_imp)
```

 coalMining

Coal Mining Disasters data

Description

A dataset with 112 entries containing annual numbers of accidents due to disasters in British coal mines for years from 1850 to 1962, considered in Carlin et al. (1992) and also in Eilers and Marx (1996).

Usage

```
data(coalMining)
```

Format

A data.frame with 112 entries, corresponding to the years from 1850 to 1962. Each entry has:

accidents number of severe accidents that have occurred each year.

years year during which accidents occurred.

Source

<https://people.reed.edu/~jones/141/Coal.html>

References

Carlin, B.P., Gelfand, A.E. and Smith, A.F.M. (1992). Hierarchical Bayesian analysis of change-point problems. *Applied Statistics*, **41**(2), 389–405.

Eilers, P.H.C. and Marx, B.D. (1996). Flexible Smoothing with B-splines and Penalties. *Statistical Science*, **11**(2), 89–121.

coef.GeDS

Coef method for GeDS objects

Description

Methods for the functions `coef` and `coefficients` that allow to extract the estimated coefficients of a fitted GeDS regression from a `GeDS-Class` object.

Usage

```
## S3 method for class 'GeDS'
coef(object, n = 3L, onlySpline = TRUE, ...)

## S3 method for class 'GeDS'
coefficients(object, n = 3L, onlySpline = TRUE, ...)
```

Arguments

<code>object</code>	the <code>GeDS-class</code> object from which the coefficients of the selected GeDS regression should be extracted.
<code>n</code>	integer value (2, 3 or 4) specifying the order (= degree +1) of the GeDS fit whose coefficients should be extracted. By default equal to 3L. Non-integer values will be passed to the function <code>as.integer</code> .
<code>onlySpline</code>	logical variable specifying whether only the coefficients for the GeDS component of the fitted multivariate regression model should be extracted or alternatively also the coefficients of the parametric component should also be extracted.
<code>...</code>	potentially further arguments (required by the definition of the generic function). They will be ignored, but with a warning.

Details

These are simple methods for the functions `coef` and `coefficients`.

As `GeDS-class` objects contain three different fits (linear, quadratic and cubic), it is possible to specify the order of the fit for which GeDS regression coefficients are required via the input argument `n`.

As mentioned in the details of `formula`, the predictor model may be multivariate and it may include a GeD spline component whereas the remaining variables may be part of a parametric component. If the `onlySpline` argument is set to `TRUE` (the default value), only the coefficients corresponding to the GeD spline component of order `n` of the multivariate predictor model are extracted.

Value

A named vector containing the required coefficients of the fitted multivariate predictor model. The coefficients corresponding to the variables that enter the parametric component of the fitted multivariate predictor model are named as the variables themselves. The coefficients of the GeDS component are coded as "N" followed by the index of the corresponding B-spline.

See Also

`coef` for the standard definition; `NGeDS` for examples.

Examples

```
# Generate a data sample for the response variable
# and the covariates
set.seed(123)
N <- 500
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
X <- sort(runif(N ,min = -2, max = 2))
Z <- runif(N)
# Specify a model for the mean of the response Y to be a superposition of
# a non-linear component f_1(X), a linear component 2*Z and a
# free term 1, i.e.
means <- f_1(X) + 2*Z + 1
# Add normal noise to the mean of y
Y <- rnorm(N, means, sd = 0.1)

# Fit to this sample a predictor model of the form f(X) + Z, where
# f(X) is the GeDS component and Z is the linear (additive) component
# see ?formula.GeDS for details
(Gmod <- NGeDS(Y ~ f(X) + Z, beta = 0.6, phi = 0.995, Xextr = c(-2,2)))

# Extract the GeD spline regression coefficients
coef(Gmod, n = 3)

# Extract all the coefficients, including the one for the linear component
coef(Gmod, onlySpline = FALSE, n = 3)
```

coef.GeDSboost,gam *Coef method for GeDSboost, GeDSgam*

Description

Methods for the functions `coef` and `coefficients` that allow to extract the estimated coefficients of `GeDSboost-Class` or `GeDSgam-Class` object.

Usage

```
## S3 method for class 'GeDSboost'
coef(object, n = 3L, ...)

## S3 method for class 'GeDSboost'
coefficients(object, n = 3L, ...)

## S3 method for class 'GeDSgam'
coef(object, n = 3L, ...)

## S3 method for class 'GeDSgam'
coefficients(object, n = 3L, ...)
```

Arguments

<code>object</code>	the <code>GeDSboost-class</code> or <code>GeDSgam-Class</code> object from which the coefficients should be extracted.
<code>n</code>	integer value (2, 3 or 4) specifying the order (= degree +1) of the FGB-GeDS/GAM-GeDS fit whose coefficients should be extracted. If <code>n = 2L</code> piecewise polynomial coefficients of the univariate GeDS base-learners are provided. In the case of bivariate GeDS base learners and <code>class(object) == "GeDSboost"</code> , the B-spline coefficients for each iteration where a particular base-learner was selected are provided. In the case of bivariate base learners and <code>class(object) == "GeDSgam"</code> , the final local-scoring B-spline coefficients for each base-learner are provided. If <code>n = 3L</code> or <code>n = 4L</code> B-spline coefficients are provided. By default <code>n</code> is equal to <code>3L</code> . Non-integer values will be passed to the function <code>as.integer</code> .
<code>...</code>	potentially further arguments (required by the definition of the generic function). They will be ignored, but with a warning.

Value

A named vector containing the required coefficients of the fitted multivariate predictor model.

See Also

`coef` for the standard definition; `NGeDSboost` and `NGeDSgam` for examples.

crossv_GeDS	<i>k-fold cross-validation</i>
-------------	--------------------------------

Description

crossv_GeDS implements k-fold cross-validation for the relevant parameters of NGeDS, GGeDS, NGeDSgam and NGeDSboost

Arguments

formula	a description of the structure of the model to be fitted, including the dependent and independent variables.
data	a data frame containing the variables referenced in the formula.
model_fun	the GeDS model to be fitted, that is, NGeDS, GGeDS, NGeDSgam or NGeDSboost.
parameters	to cross-validate. beta, phi and q in the case of NGeDS, GGeDS and NGeDSgam. In addition, for NGeDSboost, int.knots_init, shrinkage can also be tuned. Default values are int.knots_init_grid = c(0, 1, 2), shrinkage_grid = c(0.1, 0.5, 1), beta_grid = c(0.3, 0.5, 0.7), phi_grid = c(0.9, 0.95, 0.99), q_grid = c(2, 3)).

Value

Two data frames, best_params and results. best_params contains the best combination of parameters according to the cross-validated MSE. results presents the results for each of the possible combinations of parameters, given the input parameters.

Derive	<i>Derivative of GeDS objects</i>
--------	-----------------------------------

Description

This function computes derivatives of a fitted GeDS regression model.

Usage

```
Derive(object, order = 1L, x, n = 3L)
```

Arguments

object	the GeDS-Class object containing the GeDS fit which should be differentiated. It should be the result of fitting a univariate GeDS regression via NGeDS or GGeDS .
order	integer value indicating the order of differentiation required (e.g. first, second or higher derivatives). Note that order should be lower than n and that non-integer values will be passed to the function as.integer .

- x numeric vector containing values of the independent variable at which the derivatives of order order should be computed.
- n integer value (2, 3 or 4) specifying the order (= degree +1) of the GeDS fit to be differentiated. By default equal to 3L.

Details

The function is based on [splineDesign](#) and it computes the exact derivative of the fitted GeDS regression.

The function uses the property that the m -th derivative of a spline, $m = 1, 2, \dots$, expressed in terms of B-splines can be computed by differentiating the corresponding B-spline coefficients (see e.g. De Boor, 2001, Chapter X, formula (15)). Since the GeDS fit is a B-spline representation of the predictor, it cannot work on the response scale in the GNM (GLM) framework.

References

De Boor, C. (2001). *A Practical Guide to Splines (Revised Edition)*. Springer, New York.

Examples

```
# Generate a data sample for the response variable
# Y and the covariate X
set.seed(123)
N <- 500
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
X <- sort(runif(N, min = -2, max = 2))
# Specify a model for the mean of Y to include only
# a component non-linear in X, defined by the function f_1
means <- f_1(X)
# Add (Normal) noise to the mean of Y
Y <- rnorm(N, means, sd = 0.1)

# Fit GeDS regression with only a spline component in the predictor model
Gmod <- NGeDS(Y ~ f(X), beta = 0.6, phi = 0.995, Xextr = c(-2,2))

# Compute the second derivative of the cubic GeDS fit
# at the points 0, -1 and 1
Derive(Gmod, x = c(0, -1, 1), order = 2, n = 4)
```

deviance.GeDS

Deviance method for GeDS, GeDSboost, GeDSgam

Description

Method for the function [deviance](#) that allows the user to extract the value of the deviance corresponding to a selected GeDS, GeDSboost or GeDSgam fit from a [GeDS-Class](#), [GeDSboost-Class](#) or [GeDSgam-Class](#) object.

Usage

```
## S3 method for class 'GeDS'
deviance(object, n = 3L, ...)

## S3 method for class 'GeDSboost'
deviance(object, n = 3L, ...)

## S3 method for class 'GeDSgam'
deviance(object, n = 3L, ...)
```

Arguments

object	the GeDS-class , GeDSboost-class or GeDSgam-class object from which the deviance should be extracted.
n	integer value (2, 3 or 4) specifying the order (= degree +1) of the GeDS/GeDSboost/GeDSgam fit whose deviance should be extracted. By default equal to 3L. Non-integer values will be passed to the function as.integer .
...	potentially further arguments (required by the definition of the generic function). They will be ignored, but with a warning.

Details

This is a method for the function [deviance](#). As [GeDS-class](#), [GeDSboost-class](#) and [GeDSgam-class](#) objects contain three different fits (linear, quadratic and cubic), it is possible to specify the order of the GeDS fit for which the deviance is required via the input argument n.

Value

A numeric value corresponding to the deviance of the selected GeDS/GeDSboost/GeDSgam fit. #'

See Also

[deviance](#) for the standard definition; [GGeDS](#) for examples.

EWmortality

Death counts in England and Wales

Description

The dataset consists of information about the mortality of the English and Welsh male population aggregated over the years 2000, 2001 and 2002.

Usage

```
data(EWmortality)
```

Format

A data.frame with 109 entries and 3 variables: Age, Deaths and Exposure. Exposure is a mid-year estimate of the population exposed to risk.

f

Defining the covariates for the spline component in a GeDS formula.

Description

In general the GeDS predictor model may include a GeD spline regression component with respect to part of the independent variables and a parametric component in which the remaining covariates may enter as additive terms.

The function `f` is to be used in the `formula` argument of `NGeDS` or `GGeDS` in order to specify which independent variables (covariates) should be included in the GeD spline regression component of the predictor model.

Usage

```
f(x, xx = NULL, ...)
```

Arguments

<code>x</code>	numeric vector containing N sample values of the covariate chosen to enter the spline regression component of the predictor model.
<code>xx</code>	numeric vector containing N sample values for the second covariate (in case <code>NGeDS</code> is run for two dimensions). It has to be either <code>NULL</code> (the default) or a vector of size N , same as <code>x</code> .
<code>...</code>	further arguments. As GeDS currently allows for up to two covariates, specification of further arguments will return an error.

Note

This function is intended to be used only as part of the `formula` in a GeDS regression via `NGeDS` or `GGeDS` and not to be called in other cases by the user.

See Also

[NGeDS](#); [GGeDS](#).

Examples

```
# Generate a data sample for the response variable Y and
# the covariates X, reg1, reg2 and off
set.seed(123)
N <- 500
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
X <- sort(runif(N ,min = -2, max = 2))
```

```

reg1 <- runif(500, min = -0.1, max = 0.1)
reg2 <- runif(500, min = -0.2, max = 0.2)
off <- runif(500, min = -1, max = 1)
# Specify a model for the mean of Y to include a component non linear
# in X defined by the function f_1 and a linear one in the other covariates
means <- f_1(X) + 2*reg1 + 0.5*reg2 + off
# Add Normal noise to the mean of Y
Y <- rnorm(N, means, sd = 0.1)

# Specify a formula that will be used to model Y as a
# function of X, reg1, reg2 and off.
# The covariate X is for the spline component modeled as GeDS,
# reg1 and reg2 enter linearly, off is an offset, i.e. no coefficient
# will be estimated for it
formula <- Y ~ f(X) + reg1 + reg2 + offset(off)

# Fit a GeDS model specified in formula using NGeDS
(Gmod <- NGeDS(formula, beta = 0.6, phi = 0.995, Xextr = c(-2,2)))

```

formula.GeDS

Formula for the predictor model

Description

A description of the structure of a predictor model to be fitted using [NGeDS](#) and/or [GGeDS](#) and how this information can be extracted from a [GeDS-class](#) object.

Usage

```
## S3 method for class 'GeDS'
formula(x, ...)
```

Arguments

x	Fitted GeDS-class object, typically produced by NGeDS or GGeDS from which the predictor model formula should be extracted.
...	Unused in this case.

Details

In the GeDS GNM (GLM) regression, implemented in [NGeDS](#) and [GGeDS](#), it is assumed that the mean of the response variable transformed using an appropriate link function is modelled through a possibly multivariate predictor model involving two components: a GeD variable knot spline regression component involving up to two of the independent variables and a parametric component with respect to the remaining independent variables. The formula is used to specify the structure of such a possibly multivariate predictor model.

The formulae that are input in `NGeDS` and `GGeDS` are similar to those input in `lm` or `glm` except that the function `f` should be specified in order to identify which of the covariates enter the GeD spline regression part of the predictor model. For example, if the predictor model is univariate and it links the transformed means of y to x_1 , the predictor has only a GeD spline component and the `formula` should be in the form $y \sim f(x_1)$.

As noted, there may be additional independent variables, x_2, x_3, \dots which may enter linearly into the parametric component of the predictor model and not be part of the GeD spline regression component. For example one may use the formula $y \sim f(x_1) + x_2 + x_3$ which assumes a spline regression only between the transformed mean of y and x_1 , while x_2 and x_3 enter the predictor model just linearly.

In the current version of the package, `GGeDS` is univariate, therefore only one covariate which enters the spline regression component can be specified.

In contrast, the function `NGeDS`, generates also bivariate GeDS regression models. Therefore, if the functional dependence of the mean of the response variable y on x_1 and x_2 needs to be jointly modelled and there are no other covariates, the formula for the corresponding two dimensional predictor model should be specified as $y \sim f(x_1, x_2)$.

Within the argument `formula`, similarly as in other R functions, it is possible to specify one or more offset variables, i.e. known terms with fixed regression coefficients equal to 1. These terms should be identified via the function `offset`.

GeDS-class

GeDS Class

Description

A fitted GeDS object returned by the functions `NGeDS` or `GGeDS` inheriting the methods from class "GeDS". Methods for functions `coef`, `knots`, `print`, `predict`, `plot`, and `lines` are available.

Slots

`Type` Character string indicating the type of the regression performed. One of "LM - Univ", "LM - Biv" or "GLM - Univ" corresponding to the Normal univariate GeDS, the Normal bivariate GeDS performed by `NGeDS` and the generalized (GNM-GLM) univariate GeDS performed by `GGeDS`.

`Linear.Knots` Vector containing the locations of the knots of the second order GeDS spline fit generated at stage A.

`Quadratic.Knots` Vector containing the locations of the knots of the third order GeDS spline fitted in stage B.

`Cubic.knots` Vector containing the locations of the knots of the fourth order GeDS spline fitted in stage B.

`Dev.Linear` Deviance of the second order GeD spline fit of stage A.

`Dev.Quadratic` Deviance of the third order GeD spline fit of stage B.

`Dev.Cubic` Deviance of the fourth order GeD spline fit of stage B.

- RSS** Vector containing the deviances of the second order spline fits computed at each GeDS iteration in stage A.
- Linear** List containing the results from running a `SplineReg` function used to fit the second order spline of stage A.
- Quadratic** List containing the results from running `SplineReg` function used to fit the third order spline in stage B.
- Cubic** List containing the results from a `SplineReg` function used to fit the fourth order spline in stage B.
- Stored** Matrix containing the knot locations estimated at each step of stage A.
- Args** List containing the input arguments passed on the `Fitters` functions.
- Call** call to the `Fitters` functions.
- Nintknots** The final number of internal knots of the second order GeD spline fit of stage A.
- iters** Number of iterations performed in stage A of the GeDS fitting procedure.
- Guesses** Initial values for the coefficients used at each iteration of stage A in order to estimate the spline coefficients. Since the initial values are used only in the IRLS procedure, this slot is empty if the object is not created by `GGeDS` or `GenUnivariateFitter` functions.
- Coefficients** Matrix containing the fitted coefficients of the GeD spline regression component and the parametric component at each iteration of stage A.
- deviance** Vector containing the deviances of the second order spline fits computed at each IRLS iteration in stage A. Since the IRLS procedure is used only in `GGeDS` or `GenUnivariateFitter`, this slot is empty if the object is not created by one of these functions.
- iterIrls** Vector containing the numbers of IRLS iterations for all iterations of stage A cumulatively. Since the IRLS procedure is used only in `GGeDS` or `GenUnivariateFitter`, this slot is empty if the object is not created by one of these functions.
- stopinfo** List of values providing information related to the stopping rule of stage A of GeDS. The sub-slots of `stopinfo` are `phis`, `phis_star`, `oldintc` and `oldslp`. The sub-slot `phis` is a vector containing the values of the ratios of deviances (or the difference of deviances if the LR stopping rule was chosen). The sub-slots `phis_star`, `oldintc` and `oldslp` are non-empty slots if the SR stopping rule was chosen. They contain respectively $\hat{\phi}_\kappa$, $\hat{\gamma}_0$ and $\hat{\gamma}_1$ computed at each iteration of stage A, see Dimitrova et al. (2023).
- Formula** The model `formula`.
- extcall** call to the `NGeDS` or `GGeDS` functions.
- terms** `terms` object containing information on the model frame.

References

- Dimitrova, D. S., Kaishev, V. K., Lattuada, A. and Verrall, R. J. (2023). Geometrically designed variable knot splines in generalized (non-)linear models. *Applied Mathematics and Computation*, **436**.
 DOI: [doi:10.1016/j.amc.2022.127493](https://doi.org/10.1016/j.amc.2022.127493)

 GeDSboost-class

GeDSboost Class

Description

A fitted GeDSboost object returned by the function `NGeDSboost` inheriting the methods from class "GeDSboost". Methods for functions `coef`, `knots`, `print`, `predict`, `visualize_boosting`, and `bl_imp` are available.

Slots

`extcall` call to the `NGeDSboost` function.

`formula` A formula object representing the model to be fitted.

`args` A list containing the arguments passed to the `NGeDSboost` function. This list includes:

- `response`: data.frame containing observations of the response variable.
- `predictors`: data.frame containing observations of the vector of predictor variables included in the model.
- `base_learners`: description of model's base learners.
- `family`: the statistical family. The possible options are
 - `mboost::AdaExp()`
 - `mboost::AUC()`
 - `mboost::Binomial(type = c("adaboost", "glm"), link = c("logit", "probit", "cloglog", "cauchit", "log"), ...)`
 - `mboost::Gaussian()`
 - `mboost::Huber(d = NULL)`
 - `mboost::Laplace()`
 - `mboost::Poisson()`
 - `mboost::GammaReg(nuirange = c(0, 100))`
 - `mboost::CoxPH()`
 - `mboost::QuantReg(tau = 0.5, qoffset = 0.5)`
 - `mboost::ExpectReg(tau = 0.5)`
 - `mboost::NBinomial(nuirange = c(0, 100))`
 - `mboost::PropOdds(nuirange = c(-0.5, -1), offrange = c(-5, 5))`
 - `mboost::Weibull(nuirange = c(0, 100))`
 - `mboost::Loglog(nuirange = c(0, 100))`
 - `mboost::Lognormal(nuirange = c(0, 100))`
 - `mboost::Gehan()`
 - `mboost::Hurdle(nuirange = c(0, 100))`
 - `mboost::Multinomial()`
 - `mboost::Cindex(sigma = 0.1, ipcw = 1)`
 - `mboost::RCG(nuirange = c(0, 1), offrange = c(-5, 5))`

- `initial_learner`: if TRUE a `NGeDS` fit was used as initial learner; otherwise, the empirical risk minimizer corresponding to the selected family was employed.
- `int.knots_init`: if `initial_learner = TRUE` the maximum number of internal knots set to the `NGeDS` function before the initial learner fit.
- `shrinkage`: shrinkage/step-length/learning rate utilized throughout the boosting iterations.
- `normalize_data`: if TRUE, then response and predictors were standardized before running the FGB algorithm.
- `X_mean`: mean of the predictor variables (only if `normalize_data = TRUE`).
- `X_sd`: standard deviation of the predictors (only if `normalize_data = TRUE`).
- `Y_mean`: mean of the response variable (only if `normalize_data = TRUE`).
- `Y_sd`: standard deviation of the response variable (only if `normalize_data = TRUE`).

`models` A list containing the 'model' generated at each boosting iteration. Each of these models includes:

- `best_bl`: fit of the base learner that minimized the residual sum of squares (RSS) in fitting the gradient at the i -th boosting iteration.
- `Y_hat`: model fitted values at the i -th boosting iteration.
- `base_learners`: knots and polynomial coefficients for each of the base-learners at the i -th boosting iteration.

`final_model` A list detailing the final GeDSboost model after the gradient descent algorithm is run:

- `model_name`: the boosting iteration corresponding to the final model.
- `DEV`: deviance of the final model.
- `Y_hat`: fitted values.
- `base_learners`: a list containing, for each base-learner, the intervals defined by the piecewise linear fit and its corresponding polynomial coefficients. It also includes the knots corresponding to each order fit, which result from computing the corresponding averaging knot location. See Kaishev et al. (2016) for details. If the number of internal knots of the final linear fit is less than $n-1$, the averaging knot location is not computed.
- `Linear.Fit/Quadratic.Fit/Cubic.Fit`: final linear, quadratic and cubic fits in B-spline form. These include the same elements as `Linear`, `Quadratic` and `Cubic` in a `GeDS-class` object (see `SplineReg` for details).

`predictions` A list containing the predicted values obtained (linear, quadratic, and cubic).

`internal_knots` A list detailing the internal knots obtained for the fits of different order (linear, quadratic, and cubic).

References

Dimitrova, D. S., Kaishev, V. K., Lattuada, A. and Verrall, R. J. (2023). Geometrically designed variable knot splines in generalized (non-)linear models. *Applied Mathematics and Computation*, **436**.

DOI: [doi:10.1016/j.amc.2022.127493](https://doi.org/10.1016/j.amc.2022.127493)

Dimitrova, D. S., Guillen, E. S. and Kaishev, V. K. (2024). **GeDS**: An R Package for Regression, Generalized Additive Models and Functional Gradient Boosting, based on Geometrically Designed (GeD) Splines. *Manuscript submitted for publication*.

 GeDSgam-class

GeDSgam Class

Description

A fitted GeDSgam object returned by the function `NGeDSgam` inheriting the methods from class "GeDSgam". Methods for functions `coef`, `knots`, `print` and `predict`.

Slots

`extcall` call to the `NGeDSgam` function.

`formula` A formula object representing the model to be fitted.

`args` A list containing the arguments passed to the `NGeDSgam` function. This list includes:

- `response`: `data.frame` containing observations of the response variable.
- `predictors`: `data.frame` containing observations of the vector of predictor variables included in the model.
- `base_learners`: description of the model's base learners ('smooth functions').
- `family`: the statistical family. The possible options are
 - `binomial(link = "logit", "probit", "cauchit", "log", "cloglog")`
 - `gaussian(link = "identity", "log", "inverse")`
 - `Gamma(link = "inverse", "identity", "log")`
 - `inverse.gaussian(link = "1/mu^2", "inverse", "identity", "log")`
 - `poisson(link = "log", "identity", "sqrt")`
 - `quasi(link = "identity", variance = "constant")`
 - `quasibinomial(link = "logit", "probit", "cloglog", "identity", "inverse", "log", "1/mu^2", "sqrt")`
 - `quasipoisson(link = "logit", "probit", "cloglog", "identity", "inverse", "log", "1/mu^2", "sqrt")`
- `normalize_data`: if TRUE, then response and predictors were standardized before running the local-scoring algorithm.
- `X_mean`: mean of the predictor variables (only if `normalize_data = TRUE`).
- `X_sd`: standard deviation of the predictors (only if `normalize_data = TRUE`).
- `Y_mean`: mean of the response variable (only if `normalize_data = TRUE`).
- `Y_sd`: standard deviation of the response variable (only if `normalize_data = TRUE`).

`final_model` A list detailing the final GeDSgam model selected after running the local scoring algorithm. The chosen model minimizes deviance across all models generated by each local-scoring iteration. This list includes:

- `model_name`: local-scoring iteration that yielded the best model. Note that when `family = "gaussian"`, it will always correspond to `iter1`, as only one local-scoring iteration is conducted in this scenario. This occurs because, with `family = "gaussian"`, the algorithm is equivalent to simply backfitting.
- `DEV`: the deviance for the fitted predictor model, defined as in Dimitrova et al. (2023), which for `family = "gaussian"` coincides with the Residual Sum of Squares.

- `Y_hat`: fitted values.
 - `eta`: additive predictor.
 - `mu`: vector of means.
 - `z`: adjusted dependent variable.
- `base_learners`: a list containing, for each base-learner, the corresponding piecewise linear fit polynomial coefficients. It includes the knots for each order fit, resulting from computing the averaging knot location. Although if the number of internal knots of the final linear fit is less than $n-1$, the averaging knot location is not computed.
- `Linear.Fit`: final model linear fit in B-spline form. See for details [SplineReg](#).
- `Quadratic.Fit`: quadratic fit obtained via Schoenberg variation diminishing spline approximation. See for details [SplineReg](#).
- `Cubic.Fit`: cubic fit obtained via Schoenberg variation diminishing spline approximation. See for details [SplineReg](#).

`predictions` A list containing the predicted values obtained (linear, quadratic, and cubic). Each of the predictions contains both the additive predictor `eta` and the vector of means `mu`.

`internal_knots` A list detailing the internal knots obtained for the fits of different order (linear, quadratic, and cubic).

References

Dimitrova, D. S., Kaishev, V. K., Lattuada, A. and Verrall, R. J. (2023). Geometrically designed variable knot splines in generalized (non-)linear models. *Applied Mathematics and Computation*, **436**.

DOI: [doi:10.1016/j.amc.2022.127493](https://doi.org/10.1016/j.amc.2022.127493)

Dimitrova, D. S., Guillen, E. S. and Kaishev, V. K. (2024). **GeDS**: An R Package for Regression, Generalized Additive Models and Functional Gradient Boosting, based on Geometrically Designed (GeD) Splines. *Manuscript submitted for publication*.

GGeDS

Generalized Geometrically Designed Spline regression estimation

Description

GGeDS constructs a Geometrically Designed (univariate) variable knots spline regression model for the predictor in the context of Generalized (Non-)Linear Models, referred to as a GeDS model, for a response with a pre-specified distribution from the Exponential Family.

Usage

```
GGeDS(
  formula,
  family = gaussian(),
  data,
  weights,
  beta,
```

```

phi = 0.99,
min.intknots,
max.intknots,
q = 2L,
Xextr = NULL,
Yextr = NULL,
show.iters = FALSE,
stoptype = "SR",
higher_order = TRUE
)

```

Arguments

formula	a description of the structure of the predictor model to be fitted, including the dependent and independent variables. See formula for details.
family	a description of the error distribution and link function to be used in the model. This can be a character string naming a family function (e.g. "gaussian"), the family function itself (e.g. gaussian) or the result of a call to a family function (e.g. gaussian()). See family for details on family functions.
data	an optional data frame, list or environment containing the variables of the predictor model. In case the variables are not found in data, they are taken from <code>environment(formula)</code> , typically the environment from which GGeDS is called.
weights	an optional vector of 'prior weights' to be put on the observations in the fitting process in case the user requires weighted GeDS fitting. It is NULL by default.
beta	numeric parameter in the interval $[0, 1]$ tuning the knot placement in stage A of GeDS. See details below.
phi	numeric parameter in the interval $[0, 1]$ specifying the threshold for the stopping rule (model selector) in stage A of GeDS. See also <code>stoptype</code> and details below.
min.intknots	optional parameter allowing the user to set a minimum number of internal knots required. By default equal to zero.
max.intknots	optional parameter allowing the user to set a maximum number of internal knots to be added by the GeDS estimation algorithm. By default equal to the number of knots for the saturated GeDS model (i.e. $N - 2$, where N is the number of observations).
q	numeric parameter which allows to fine-tune the stopping rule of stage A of GeDS, by default equal to 2. See details below.
Xextr	numeric vector of 2 elements representing the left-most and right-most limits of the interval embedding the observations of the independent variable. See details.
Yextr	numeric vector of 2 elements representing the left-most and right-most limits of the interval embedding the observations of the second independent variable (if the bivariate GeDS is run). See details.
show.iters	logical variable indicating whether or not to print information at each step. By default equal to FALSE.
stoptype	a character string indicating the type of GeDS stopping rule to be used. It should be either one of "SR", "RD" or "LR", partial match allowed. See details below.

`higher_order` a logical that defines whether to compute the higher order fits (quadratic and cubic) after stage A is run. Default is TRUE.

Details

The GGeDS function extends the GeDS methodology, recently developed by Kaishev et al. (2016) and implemented in the NGeDS function for the Normal case, to the more general GNM (GLM) context, allowing for the response to have any distribution from the Exponential Family. Under the GeDS-GNM approach the (non-)linear predictor is viewed as a spline with variable knots which are estimated along with the regression coefficients and the order of the spline, using a two stage procedure. In stage A, a linear variable-knot spline is fitted to the data applying iteratively re-weighted least squares (see IRLSfit function). In stage B, a Schoenberg variation diminishing spline approximation to the fit from stage A is constructed, thus simultaneously producing spline fits of order 2, 3 and 4, all of which are included in the output, a GeDS-Class object. A detailed description of the underlying algorithm can be found in Dimitrova et al. (2023).

As noted in formula, the argument formula allows the user to specify predictor models with two components, a spline regression (non-parametric) component involving part of the independent variables identified through the function f and an optional parametric component involving the remaining independent variables. For GGeDS only one independent variable is allowed for the spline component and arbitrary many independent variables for the parametric component of the predictor. Failure to specify the independent variable for the spline regression component through the function f will return an error. See formula.

Within the argument formula, similarly as in other R functions, it is possible to specify one or more offset variables, i.e. known terms with fixed regression coefficients equal to 1. These terms should be identified via the function offset.

The parameter beta tunes the placement of a new knot in stage A of the algorithm. Once a current second-order spline is fitted to the data the 'working' residuals (see IRLSfit) are computed and grouped by their sign. A new knot is placed at a location defined by the group for which a certain measure attains its maximum. The latter measure is defined as a weighted linear combination of the range of each group and the mean of the absolute residuals within it. The parameter beta determines the weights in this measure correspondingly as beta and 1 - beta. The higher it is, the more weight is put to the mean of the residuals and the less to the range of their corresponding x-values (see Kaishev et al., 2016, for further details).

The default values of beta are beta = 0.5 if the response is assumed to be Gaussian, beta = 0.2 if it is Poisson (or Quasipoisson), while if it is Binomial, Quasibinomial or Gamma beta = 0.1, which reflect our experience of running GeDS for different underlying functional dependencies.

The argument stoptype allows to choose between three alternative stopping rules for the knot selection in stage A of GeDS, the "RD", that stands for *Ratio of Deviances*, the "SR", that stands for *Smoothed Ratio of deviances* and the "LR", that stands for *Likelihood Ratio*. The latter is based on the difference of deviances rather than on their ratio as in the case of "RD" and "SR". Therefore "LR" can be viewed as a log likelihood ratio test performed at each iteration of the knot placement. In each of these cases the corresponding stopping criterion is compared with a threshold value phi (see below).

The argument phi provides a threshold value required for the stopping rule to exit the knot placement in stage A of GeDS. The higher the value of phi, the more knots are added under the "RD" and "SR" stopping rules contrary to the case of the stopping rule "LR" where the lower phi is, more

knots are included in the spline regression. Further details for each of the three alternative stopping rules can be found in Dimitrova et al. (2023).

The argument `q` is an input parameter that allows to fine-tune the stopping rule in stage A. It identifies the number of consecutive iterations over which the deviance should exhibit stable convergence so as the knot placement in stage A is terminated. More precisely, under any of the rules "RD", "SR" or "LR" the deviance at the current iteration is compared to the deviance computed `q` iterations before, i.e. before selecting the last `q` knots. Setting a higher `q` will lead to more knots being added before exiting stage A of GeDS.

Value

A `GeDS-Class` object, i.e. a list of items that summarizes the main details of the fitted GeDS regression. See `GeDS-Class` for details. Some S3 methods are available in order to make these objects tractable, such as `coef`, `deviance`, `knots`, `predict` and `print` as well as S4 methods for `lines` and `plot`.

References

Kaishev, V.K., Dimitrova, D.S., Haberman, S. and Verrall, R.J. (2016). Geometrically designed, variable knot regression splines. *Computational Statistics*, **31**, 1079–1105.

DOI: [doi:10.1007/s0018001506217](https://doi.org/10.1007/s0018001506217)

Dimitrova, D. S., Kaishev, V. K., Lattuada, A. and Verrall, R. J. (2023). Geometrically designed variable knot splines in generalized (non-)linear models. *Applied Mathematics and Computation*, **436**.

DOI: [doi:10.1016/j.amc.2022.127493](https://doi.org/10.1016/j.amc.2022.127493)

See Also

`NGeDS`; `GeDS-Class`; S3 methods such as `coef.GeDS`, `deviance.GeDS`, `knots.GeDS`, `print.GeDS` and `predict.GeDS`; `Integrate` and `Derive`; `PPolyRep`.

Examples

```
#####
# Generate a data sample for the response variable Y and the covariate X
# assuming Poisson distributed error and log link function
# See section 4.1 in Dimitrova et al. (2023)
set.seed(123)
N <- 500
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
X <- sort(runif(N, min = -2, max = 2))
# Specify a model for the mean of Y to include only a component
# non-linear in X, defined by the function f_1
means <- exp(f_1(X))

#####
## POISSON ##
#####
# Generate Poisson distributed Y according to the mean model
Y <- rpois(N, means)
```

```

# Fit a Poisson GeDS regression using GGeDS
(Gmod <- GGeDS(Y ~ f(X), beta = 0.2, phi = 0.995, family = poisson(),
              Xextr = c(-2,2)))

# Plot the quadratic and cubic GeDS fits
plot(X,log(Y),xlab = "x", ylab = expression(f[1](x)))
lines(Gmod, n = 3, col = "red")
lines(Gmod, n = 4, col = "blue", lty = 2)
legend("topleft", c("Quadratic", "Cubic"),
      col = c("red", "blue"), lty = c(1,2))

# Generate GeDS prediction at X=0, first on the response scale and then on
# the predictor scale
predict(Gmod, n = 3, newdata = data.frame(X = 0))
predict(Gmod, n = 3, newdata = data.frame(X = 0), type = "link")

# Apply some of the other available methods, e.g.
# knots, coefficients and deviance extractions for the
# quadratic GeDS fit
knots(Gmod)
coef(Gmod)
deviance(Gmod)

# the same but for the cubic GeDS fit
knots(Gmod, n = 4)
coef(Gmod, n = 4)
deviance(Gmod, n = 4)

#####
## GAMMA ##
#####
# Generate Gamma distributed Y according to the mean model
Y <- rgamma(N, shape = means, rate = 0.1)
# Fit a Gamma GeDS regression using GGeDS
Gmod <- GGeDS(Y ~ f(X), beta = 0.1, phi = 0.995, family = Gamma(log),
              Xextr = c(-2,2))
plot(Gmod, f = function(x) exp(f_1(x))/0.1)

#####
## BINOMIAL ##
#####
# Generate Binomial distributed Y according to the mean model
eta <- f_1(X) - 4
means <- exp(eta)/(1+exp(eta))
Y <- rbinom(N, size = 50, prob = means) / 50
# Fit a Binomial GeDS regression using GGeDS
Gmod <- GGeDS(Y ~ f(X), beta = 0.1, phi = 0.995, family = "binomial",
              Xextr = c(-2,2))
plot(Gmod, f = function(x) exp(f_1(x) - 4)/(1 + exp(f_1(x) - 4)))

#####

```

```

# A real data example
# See Dimitrova et al. (2023), Section 4.2

data("coalMining")
(Gmod2 <- GGeDS(formula = accidents ~ f(years), beta = 0.1, phi = 0.98,
                family = poisson(), data = coalMining))
(Gmod3 <- GGeDS(formula = accidents ~ f(years), beta = 0.1, phi = 0.985,
                family = poisson(), data = coalMining))
plot(coalMining$years, coalMining$accidents, type = "h", xlab = "Years",
     ylab = "Accidents")
lines(Gmod2, tr = exp, n = 4, col = "red")
lines(Gmod3, tr = exp, n = 4, col = "blue", lty = 2)
legend("topright", c("phi = 0.98", "phi = 0.985"), col = c("red", "blue"),
      lty=c(1, 2))

## Not run:
#####
# The same regression in the example of GeDS
# but assuming Gamma and Poisson responses
# See Dimitrova et al. (2023), Section 4.2

data('BaFe2As2')
(Gmod4 <- GGeDS(intensity ~ f(angle), data = BaFe2As2, beta = 0.6, phi = 0.995, q = 3,
                family = Gamma(log), stoptype = "RD"))
plot(Gmod4)

(Gmod5 <- GGeDS(intensity ~ f(angle), data = BaFe2As2, beta = 0.1, phi = 0.995, q = 3,
                family = poisson(), stoptype = "SR"))
plot(Gmod5)

## End(Not run)

#####
# Life tables
# See Dimitrova et al. (2023), Section 4.2

data(EWmortality)
attach(EWmortality)
(M1 <- GGeDS(formula = Deaths ~ f(Age) + offset(log(Exposure)),
             family = poisson(), phi = 0.99, beta = 0.1, q = 3,
             stoptype = "LR"))

Exposure_init <- Exposure + 0.5 * Deaths
Rate <- Deaths / Exposure_init
(M2 <- GGeDS(formula = Rate ~ f(Age), weights = Exposure_init,
             family = quasibinomial(), phi = 0.99, beta = 0.1,
             q = 3, stoptype = "LR"))

op <- par(mfrow=c(2,2))
plot(Age, Deaths/Exposure, ylab = expression(mu[x]), xlab = "Age")
lines(M1, n = 3, tr = exp, lwd = 1, col = "red")

```

```

plot(Age, Rate, ylab = expression(q[x]), xlab = "Age")
lines(M2, n = 3, tr = quasibinomial()$linkinv, lwd = 1, col = "red")
plot(Age, log(Deaths/Exposure), ylab = expression(log(mu[x])), xlab = "Age")
lines(M1, n = 3, lwd = 1, col = "red")
plot(Age, quasibinomial()$linkfun(Rate), ylab = expression(logit(q[x])), xlab = "Age")
lines(M2, n = 3, lwd = 1, col = "red")
par(op)

#####
# bivariate example
set.seed(123)
doublesin <- function(x) {
# Adjusting the output to ensure it's positive
exp(sin(2*x[,1]) + sin(2*x[,2]))
}
X <- round(runif(400, min = 0, max = 3), 2)
Y <- round(runif(400, min = 0, max = 3), 2)
# Calculate lambda for Poisson distribution
lambda <- doublesin(cbind(X,Y))
# Generate Z from Poisson distribution
Z <- rpois(400, lambda)
data <- data.frame(X, Y, Z)

# Fit a Poisson GeDS regression using GGeDS
BivGeDS <- GGeDS(Z ~ f(X,Y), beta = 0.2, phi = 0.995, family = "poisson",
Xextr = c(0, 3), Yextr = c(0, 3))

# MSEs w.r.t data
mean((Z-BivGeDS$Linear$Predicted)^2)
mean((Z-BivGeDS$Quadratic$Predicted)^2)
mean((Z-BivGeDS$Cubic$Predicted)^2)

# MSEs w.r.t true function
f_XY <- apply(cbind(X, Y), 1, function(row) doublesin(matrix(row, ncol = 2)))
mean((f_XY - BivGeDS$Linear$Predicted)^2)
mean((f_XY - BivGeDS$Quadratic$Predicted)^2)
mean((f_XY - BivGeDS$Cubic$Predicted)^2)

# Surface plot of the generating function (doublesin)
plot(BivGeDS, f = doublesin)
# Surface plot of the fitted model
plot(BivGeDS)

```

Integrate

Defined integral of GeDS objects

Description

This function computes defined integrals of a fitted GeDS regression model.

Usage

```
Integrate(object, to, from, n = 3L)
```

Arguments

object	the GeDS-class object containing the GeDS fit which should be integrated. It should be the result of fitting a univariate GeDS regression via NGeDS or GGeDS .
to	numeric vector containing the upper limit(s) of integration.
from	optional numeric vector containing the lower limit(s) of integration. It should be either of size one or of the same size as the argument to. If left unspecified, by default it is set to the left-most limit of the interval embedding the observations of the independent variable.
n	integer value (2, 3 or 4) specifying the order (= degree +1) of the GeDS fit to be integrated. By default equal to 3L. Non-integer values will be passed to the function as.integer .

Details

The function is based on the well known property (c.f. De Boor, 2001, Chapter X, formula (33)) that the integral of a linear combination of appropriately normalized B-splines is equal to the sum of its corresponding coefficients, noting that the GeDS regression is in fact such a linear combination.

Since the function is based on this property, it is designed to work only on the predictor scale in the GNM (GLM) framework.

If the argument `from` is a single value, then it is taken as the lower limit of integration for all the defined integrals required, whereas the upper limits of integration are the values contained in the argument `to`. If the arguments `from` and `to` are of similar size, the integrals (as many as the size) are computed by sequentially taking the pairs of values in the `from` and `to` vectors as limits of integration.

References

De Boor, C. (2001). *A Practical Guide to Splines (Revised Edition)*. Springer, New York.

Examples

```
# Generate a data sample for the response variable
# Y and the single covariate X
# see Dimitrova et al. (2023), section 4.1
set.seed(123)
N <- 500
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
X <- sort(runif(N, min = -2, max = 2))
# Specify a model for the mean of Y to include only
# a component non-linear in X, defined by the function f_1
means <- f_1(X)
# Add (Normal) noise to the mean of Y
Y <- rnorm(N, means, sd = 0.1)
```

```

# Fit GeDS regression using NGeDS
Gmod <- NGeDS(Y ~ f(X), beta = 0.6, phi = .995, Xextr = c(-2,2))

# Compute defined integrals (in TeX style)  $\int_{-1}^1 f(x)dx$ 
# and  $\int_{-1}^1 f(x)dx$ 
#  $f$  being the quadratic fit
Integrate(Gmod, to = c(-1,1), from = 1, n = 3)

# Compute defined integrals (in TeX style)  $\int_{-1}^1 f(x)dx$ 
# and  $\int_{-1}^1 f(x)dx$ 
#  $f$  being the quadratic fit
Integrate(Gmod, to = c(-1,1), from = c(1,-1), n = 3)

## Not run:
## This gives an error
Integrate(Gmod, to = 1, from = c(1,-1), n = 3)

## End(Not run)

```

IRLSfit

IRLS Estimation

Description

This function is an implementation of the IRLS estimation algorithm adjusted to the specific usage in the function [SplineReg_GLM](#).

Usage

```

IRLSfit(
  x,
  y,
  weights = rep(1, nobs),
  mustart = NULL,
  offset = rep(0, nobs),
  family = gaussian(),
  control = list()
)

```

Arguments

<code>x</code>	a matrix of regression functions (e.g. B-splines and/or terms of the parametric part) evaluated at the sample values of the covariate(s).
<code>y</code>	a vector of size N containing the observed values of the response variable y .
<code>weights</code>	an optional vector of ‘prior weights’ to be put on the observations in case the user requires weighted IRLS fitting. It is a vector of 1s by default.

mustart	initial values for the vector of means of the response variable in the IRLS regression estimation. Must be a vector of length N .
offset	a vector of size N that can be used to specify a fixed covariate to be included in the predictor model avoiding the estimation of its corresponding regression coefficient. In case more than one covariate is fixed, the user should sum the corresponding coordinates of the fixed covariates to produce one common N -vector of coordinates.
family	a description of the error distribution and link function to be used in the model. This can be a character string naming a family function (e.g. "gaussian"), the family function itself (e.g. <code>gaussian</code>) or the result of a call to a family function (e.g. <code>gaussian()</code>). See family for details on family functions.
control	a list of parameters for controlling the IRLS fitting process to be passed on to glm.control . See glm.fit for further details.

Details

This function is a slightly modified version of the [glm.fit](#) from the package **stats** to which we refer for further details. The difference in the inputs of `IRLSfit` and [glm.fit](#) is that the former admits initial values only for the vector of means.

The output from `IRLSfit` has some additional slots compared to [glm.fit](#). We note that the slots `weights`, `res2` and `z` contain values of the IRLS weights, "working residuals" and transformed responses computed *after* the last IRLS iteration, i.e. they are based on the estimated coefficients that are returned by `IRLSfit`.

The source code of `IRLSfit` contains also some commented lines that produce useful plots at each IRLS iteration. Normally, printing these plots is time consuming, but they could be run for inspection purposes.

Value

A list containing:

<code>coefficients</code>	a named vector containing the estimated regression coefficients;
<code>residuals</code>	the 'working' residuals, that are the residuals in the final iteration of the IRLS fit. Since cases with zero weights are omitted, their working residuals are NA;
<code>res2</code>	the working residuals after the final IRLS iteration. They are used within the knot placement steps of stage A of GeDS;
<code>fitted.values</code>	the fitted mean values, obtained by transforming the predictor by the inverse of the link function;
<code>rank</code>	the numeric rank of the fitted linear model;
<code>family</code>	the family object used;
<code>linear.predictors</code>	the fitted predictor;
<code>deviance</code>	a vector containing the deviances obtained at each IRLS iteration;
<code>lastdeviance</code>	the deviance at the last IRLS iteration;
<code>null.deviance</code>	The deviance for the null model (see glm documentation);

<code>iter</code>	the number of IRLS iterations performed;
<code>weights</code>	the working weights after the last IRLS iteration;
<code>prior.weights</code>	the “prior weights” (see the <code>weights</code> argument);
<code>df.residual</code>	the residual degrees of freedom;
<code>df.null</code>	the residual degrees of freedom for the null model;
<code>y</code>	the vector of values of the response variable used in the fitting;
<code>z</code>	the transformed responses computed after the last IRLS iteration;
<code>converged</code>	logical. Was the IRLS algorithm judged to have converged?
<code>boundary</code>	logical. Is the fitted value on the boundary of the attainable values?

In addition, non-empty fits will have components `qr`, `R` and `effects` relating to the final weighted linear fit, see [lm.fit](#) documentation.

See Also

[glm.fit](#)

knots.GeDS

Knots method for GeDS, GeDSboost, GeDSgam

Description

Method for the generic function [knots](#) that allows the user to extract the vector of knots of a GeDS, GeDSboost or GeDSgam fit of a specified order contained in a [GeDS-class](#), [GeDSboost-class](#) or [GeDSgam-class](#) object, respectively.

Usage

```
## S3 method for class 'GeDS'
knots(Fn, n = 3L, options = c("all", "internal"), ...)

## S3 method for class 'GeDSboost'
knots(Fn, n = 3L, options = c("all", "internal"), ...)

## S3 method for class 'GeDSgam'
knots(Fn, n = 3L, options = c("all", "internal"), ...)
```

Arguments

<code>Fn</code>	the GeDS-class , GeDSboost-class or GeDSgam-class object from which the vector of knots for the specified GeDS, FGB-GeDS or GAM-GeDS fit should be extracted.
<code>n</code>	integer value (2, 3 or 4) specifying the order (= degree +1) of the GeDS, FGB-GeDS or GAM-GeDS fit whose knots should be extracted. By default equal to 3L. Non-integer values will be passed to the function as.integer .

options a character string specifying whether "all" knots, including the left-most and the right-most limits of the interval embedding the observations (the default) or only the "internal" knots should be extracted.

... potentially further arguments (required for compatibility with the definition of the generic function). Currently ignored, but with a warning.

Details

This is a method for the function `knots` in the `stats` package.

As `GeDS-class`, `GeDSboost-class` and `GeDSgam-class` objects contain three different fits (linear, quadratic and cubic), it is possible to specify the order of the GeDS fit whose knots are required via the input argument `n`.

Value

A vector in which each element represents a knot of the GeDS/FGB-GeDS/GAM-GeDS fit of the required order.

See Also

`knots` for the definition of the generic function; `NGeDS`, `GGeDS`, `NGeDSboost` and `NGeDSgam` for examples.

`lines, GeDS-method` *Lines method for GeDS objects.*

Description

Lines method for GeDS objects. Adds a GeDS curve to an existing plot.

Usage

```
## S4 method for signature 'GeDS'
lines(
  x,
  n = 3L,
  transform = function(x) x,
  onlySpline = TRUE,
  data = data.frame(),
  ...
)
```

Arguments

x	a GeDS-Class object from which the GeDS fit should be extracted.
n	integer value (2, 3 or 4) specifying the order (= degree +1) of the GeDS fit that should be plotted. By default equal to 3L. Non-integer values will be passed to the function as.integer .
transform	a function that can be used to transform the scale of the Y axis. Typically it can be the inverse of the link function if the plot is on the scale of the response variable.
onlySpline	logical variable specifying whether only the spline component of the fitted GeDS predictor model should be plotted or alternatively also the parametric component (see formula) should be plotted.
data	an optional data.frame, list or environment containing values of the independent variables for which the GeDS predicted values should be plotted. If left empty the values are extracted from the object x itself.
...	further arguments to be passed to the default lines function.

Details

This method can be used to add a curve corresponding to a particular GeDS fit to an active plot.

As GeDS objects contain three different fits (linear, quadratic and cubic), it is possible to specify the order of the GeDS regression to be plotted via the input argument n.

See Also

[lines](#) for the definition of the generic function; [NGeDS](#) and [GGeDS](#) for examples.

Examples

```
# Generate a data sample for the response variable
# Y and the single covariate X
set.seed(123)
N <- 500
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
X <- sort(runif(N, min = -2, max = 2))
# Specify a model for the mean of Y to include only a component
# non-linear in X, defined by the function f_1
means <- f_1(X)
# Add (Normal) noise to the mean of Y
Y <- rnorm(N, means, sd = 0.1)

# Fit a GeDS regression model using NGeDS
(Gmod <- NGeDS(Y ~ f(X), beta = 0.6, phi = 0.995, Xextr = c(-2,2)))

# Plot the GeDS third order fit (the quadratic one)
# without its corresponding Polygon
plot(Gmod, type = "none")

# Add a curve corresponding to the second order fit (the linear one)
```

```
lines(Gmod, n = 2, col = "green", lwd = 2, lty = 3)
```

 NGeDS

Geometrically Designed Spline regression estimation

Description

NGeDS constructs a Geometrically Designed variable knots spline regression model referred to as a GeDS model, for a response having a Normal distribution.

Usage

```
NGeDS(
  formula,
  data,
  weights,
  beta = 0.5,
  phi = 0.99,
  min.intknots = 0,
  max.intknots = 500,
  q = 2,
  Xextr = NULL,
  Yextr = NULL,
  show.iters = FALSE,
  stoptype = "RD",
  higher_order = TRUE,
  intknots = NULL,
  only_predictions = FALSE
)
```

Arguments

formula	a description of the structure of the model to be fitted, including the dependent and independent variables. See formula for details.
data	an optional data frame, list or environment containing the variables of the model. If not found in <code>data</code> , the variables are taken from <code>environment(formula)</code> , typically the environment from which NGeDS is called.
weights	an optional vector of ‘prior weights’ to be put on the observations in the fitting process in case the user requires weighted GeDS fitting. It should be <code>NULL</code> or a numeric vector of the same length as the response variable in the argument formula .
beta	numeric parameter in the interval $[0, 1]$ tuning the knot placement in stage A of GeDS. See details.
phi	numeric parameter in the interval $[0, 1]$ specifying the threshold for the stopping rule (model selector) in stage A of GeDS. See also <code>stoptype</code> and details below.

<code>min.intknots</code>	optional parameter allowing the user to set a minimum number of internal knots required. By default equal to zero.
<code>max.intknots</code>	optional parameter allowing the user to set a maximum number of internal knots to be added by the GeDS estimation algorithm. By default equal to the number of knots for the saturated GeDS model.
<code>q</code>	numeric parameter which allows to fine-tune the stopping rule of stage A of GeDS, by default equal to 2. See details.
<code>Xextr</code>	numeric vector of 2 elements representing the left-most and right-most limits of the interval embedding the observations of the first independent variable. See details.
<code>Yextr</code>	numeric vector of 2 elements representing the left-most and right-most limits of the interval embedding the observations of the second independent variable (if the bivariate GeDS is run). See details.
<code>show.iters</code>	logical variable indicating whether or not to print information at each step.
<code>stoptype</code>	a character string indicating the type of GeDS stopping rule to be used. It should be either one of "SR", "RD" or "LR", partial match allowed. See details.
<code>higher_order</code>	a logical that defines whether to compute the higher order fits (quadratic and cubic) after stage A is run. Default is TRUE.
<code>intknots</code>	vector of starting internal knots. Default is NULL.
<code>only_predictions</code>	logical, if TRUE only predictions are computed.

Details

The NGeDS function implements the GeDS methodology, recently developed by Kaishev et al. (2016) and extended in the GGeDS function for the more general GNM, (GLM) context, allowing for the response to have any distribution from the Exponential Family. Under the GeDS approach the (non-)linear predictor is viewed as a spline with variable knots which are estimated along with the regression coefficients and the order of the spline, using a two stage algorithm. In stage A, a linear variable-knot spline is fitted to the data applying iteratively least squares regression (see `lm` function). In stage B, a Schoenberg variation diminishing spline approximation to the fit from stage A is constructed, thus simultaneously producing spline fits of order 2, 3 and 4, all of which are included in the output, a `GeDS-Class` object.

As noted in `formula`, the argument `formula` allows the user to specify models with two components, a spline regression (non-parametric) component involving part of the independent variables identified through the function `f` and an optional parametric component involving the remaining independent variables. For NGeDS one or two independent variables are allowed for the spline component and arbitrary many independent variables for the parametric component. Failure to specify the independent variable for the spline regression component through the function `f` will return an error. See `formula`.

Within the argument `formula`, similarly as in other R functions, it is possible to specify one or more offset variables, i.e. known terms with fixed regression coefficients equal to 1. These terms should be identified via the function `offset`.

The parameter `beta` tunes the placement of a new knot in stage A of the algorithm. Once a current second-order spline is fitted to the data the regression residuals are computed and grouped by their

sign. A new knot is placed at a location defined by the group for which a certain measure attains its maximum. The latter measure is defined as a weighted linear combination of the range of each group and the mean of the absolute residuals within it. The parameter β determines the weights in this measure correspondingly as β and $1 - \beta$. The higher it is, the more weight is put to the mean of the residuals and the less to the range of their corresponding x -values. The default value of β is 0.5.

The argument `stoptype` allows to choose between three alternative stopping rules for the knot selection in stage A of GeDS, the "RD", that stands for *Ratio of Deviances*, the "SR", that stands for *Smoothed Ratio of deviances* and the "LR", that stands for *Likelihood Ratio*. The latter is based on the difference of deviances rather than on their ratio as in the case of "RD" and "SR". Therefore "LR" can be viewed as a log likelihood ratio test performed at each iteration of the knot placement. In each of these cases the corresponding stopping criterion is compared with a threshold value ϕ (see below).

The argument `phi` provides a threshold value required for the stopping rule to exit the knot placement in stage A of GeDS. The higher the value of ϕ , the more knots are added under the "RD" and "SR" stopping rules contrary to the case of the stopping rule "LR" where the lower ϕ is, more knots are included in the spline regression. Further details for each of the three alternative stopping rules can be found in Dimitrova et al. (2023).

The argument `q` is an input parameter that allows to fine-tune the stopping rule in stage A. It identifies the number of consecutive iterations over which the deviance should exhibit stable convergence so as the knot placement in stage A is terminated. More precisely, under any of the rules "RD", "SR", or "LR", the deviance at the current iteration is compared to the deviance computed q iterations before, i.e., before selecting the last q knots. Setting a higher q will lead to more knots being added before exiting stage A of GeDS.

Value

`GeDS-Class` object, i.e. a list of items that summarizes the main details of the fitted GeDS regression. See `GeDS-Class` for details. Some S3 methods are available in order to make these objects tractable, such as `coef`, `deviance`, `knots`, `predict` and `print` as well as S4 methods for `lines` and `plot`.

References

- Kaishev, V.K., Dimitrova, D.S., Haberman, S. and Verrall, R.J. (2016). Geometrically designed, variable knot regression splines. *Computational Statistics*, **31**, 1079–1105.
DOI: [doi:10.1007/s0018001506217](https://doi.org/10.1007/s0018001506217)
- Dimitrova, D. S., Kaishev, V. K., Lattuada, A. and Verrall, R. J. (2023). Geometrically designed variable knot splines in generalized (non-)linear models. *Applied Mathematics and Computation*, **436**.
DOI: [doi:10.1016/j.amc.2022.127493](https://doi.org/10.1016/j.amc.2022.127493)

See Also

`GGeDS`; `GeDS-Class`; S3 methods such as `coef.GeDS`, `deviance.GeDS`, `knots.GeDS`, `print.GeDS` and `predict.GeDS`; `Integrate` and `Derive`; `PPolyRep`.

Examples

```
#####
# Generate a data sample for the response variable
# Y and the single covariate X
set.seed(123)
N <- 500
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
X <- sort(runif(N, min = -2, max = 2))
# Specify a model for the mean of Y to include only a component
# non-linear in X, defined by the function f_1
means <- f_1(X)
# Add (Normal) noise to the mean of Y
Y <- rnorm(N, means, sd = 0.1)

# Fit a Normal GeDS regression using NGeDS
(Gmod <- NGeDS(Y ~ f(X), beta = 0.6, phi = 0.995, Xextr = c(-2,2)))

# Apply some of the available methods, e.g.
# coefficients, knots and deviance extractions for the
# quadratic GeDS fit
# Note that the first call to the function knots returns
# also the left and right limits of the interval containing
# the data
coef(Gmod, n = 3)
knots(Gmod, n = 3)
knots(Gmod, n = 3, options = "internal")
deviance(Gmod, n = 3)

# Add a covariate, Z, that enters linearly
Z <- runif(N)
Y2 <- Y + 2*Z + 1
# Re-fit the data using NGeDS
(Gmod2 <- NGeDS(Y2 ~ f(X) + Z, beta = 0.6, phi = 0.995, Xextr = c(-2,2)))
coef(Gmod2, n = 3)
coef(Gmod2, onlySpline = FALSE, n = 3)

## Not run:
#####
# Real data example
# See Kaishev et al. (2016), section 4.2
data('BaFe2As2')
(Gmod2 <- NGeDS(intensity ~ f(angle), data = BaFe2As2, beta = 0.6, phi = 0.99, q = 3))
plot(Gmod2)

## End(Not run)

#####
# bivariate example
# See Dimitrova et al. (2023), section 5

# Generate a data sample for the response variable
# Z and the covariates X and Y assuming Normal noise
```

```

set.seed(123)
doublesin <- function(x){
  sin(2*x[,1])*sin(2*x[,2])
}

X <- (round(runif(400, min = 0, max = 3),2))
Y <- (round(runif(400, min = 0, max = 3),2))
Z <- doublesin(cbind(X,Y))
Z <- Z+rnorm(400, 0, sd = 0.1)
# Fit a two dimensional GeDS model using NGeDS
(BivGeDS <- NGeDS(Z ~ f(X, Y), Xextr = c(0, 3), Yextr = c(0, 3)))

# Extract quadratic coefficients/knots/deviance
coef(BivGeDS, n = 3)
knots(BivGeDS, n = 3)
deviance(BivGeDS, n = 3)

# Surface plot of the generating function (doublesin)
plot(BivGeDS, f = doublesin)
# Surface plot of the fitted model
plot(BivGeDS)

```

 NGeDSboost

Component-wise gradient boosting with NGeDS base-learners

Description

NGeDSboost performs component-wise gradient boosting (Bühlmann and Yu (2003), Bühlmann and Hothorn (2007)) using normal GeD splines (i.e., fitted with [NGeDS](#) function) as base-learners (see Dimitrova et al. (2024)).

Usage

```

NGeDSboost(
  formula,
  data,
  weights = NULL,
  normalize_data = FALSE,
  family = mboost::Gaussian(),
  link = NULL,
  initial_learner = TRUE,
  int.knots_init = 2L,
  min_iterations,
  max_iterations,
  shrinkage = 1,
  beta = 0.5,
  phi = 0.99,
  int.knots_boost = 500L,

```

```

q = 2L,
phi_boost_exit = 0.99,
q_boost = 2L,
higher_order = TRUE,
boosting_with_memory = FALSE
)

```

Arguments

formula	a description of the structure of the model to be fitted, including the dependent and independent variables. Unlike NGeDS and GGeDS , the formula specified allows for multiple additive GeD spline regression components (as well as linear components) to be included (e.g., $Y \sim f(X1) + f(X2) + X3$). See formula for further details.
data	a data frame containing the variables referenced in the formula.
weights	an optional vector of ‘prior weights’ to be put on the observations during the fitting process. It should be NULL or a numeric vector of the same length as the response variable defined in the formula.
normalize_data	a logical that defines whether the data should be normalized (standardized) before fitting the baseline linear model, i.e., before running the FGB algorithm. Normalizing the data involves scaling the predictor variables to have a mean of 0 and a standard deviation of 1. This process alters the scale and interpretation of the knots and coefficients estimated. Default is equal to FALSE.
family	determines the loss function to be optimized by the boosting algorithm. In case <code>initial_learner = FALSE</code> it also determines the corresponding empirical risk minimizer to be used as offset initial learner. By default, it is set to <code>mboost::Gaussian()</code> . Users can specify any Family object from the mboost package.
link	in case the Family object has not the desired link function you can specify it here.
initial_learner	a logical value. If set to TRUE, the model’s initial learner will be a normal GeD spline. If set to FALSE, then the initial predictor will consist of the empirical risk minimizer corresponding to the specified family. Note that if <code>initial_learner = TRUE</code> , family must be <code>mboost::Gaussian()</code> .
int.knots_init	optional parameter allowing the user to set a maximum number of internal knots to be added by the initial GeDS learner in case <code>initial_learner = TRUE</code> . Default is equal to 2L.
min_iterations	optional parameter to manually set a minimum number of boosting iterations to be run. If not specified, it defaults to 0L.
max_iterations	optional parameter to manually set the maximum number of boosting iterations to be run. If not specified, it defaults to 100L. This setting serves as a fallback when the stopping rule, based on consecutive deviances and tuned by <code>phi_boost_exit</code> and <code>q_boost</code> , does not trigger an earlier termination (see Dimitrova et al. (2024)). Therefore, users can increase/decrease the number of boosting iterations, by increasing/decreasing the value <code>phi_boost_exit</code> and/or <code>q_boost</code> , or directly specify <code>max_iterations</code> .

<code>shrinkage</code>	numeric parameter in the interval $[0, 1]$ defining the step size or shrinkage parameter. This controls the size of the steps taken in the direction of the gradient of the loss function. In other words, the magnitude of the update each new iteration contributes to the final model. Default is equal to 1.
<code>beta</code>	numeric parameter in the interval $[0, 1]$ tuning the knot placement in stage A of GeDS. Default is equal to 0.5. See details in NGeDS .
<code>phi</code>	numeric parameter in the interval $[0, 1]$ specifying the threshold for the stopping rule (model selector) in stage A of GeDS. Default is equal to 0.99. See details in NGeDS .
<code>int.knots_boost</code>	The maximum number of internal knots that can be added by the GeDS base-learners in each boosting iteration, effectively setting the value of <code>max.intknots</code> in NGeDS at each boosting iteration. Default is 500L.
<code>q</code>	numeric parameter which allows to fine-tune the stopping rule of stage A of GeDS, by default equal to 2L. See details in NGeDS .
<code>phi_boost_exit</code>	numeric parameter in the interval $[0, 1]$ specifying the threshold for the boosting iterations stopping rule. Default is equal to 0.995.
<code>q_boost</code>	numeric parameter which allows to fine-tune the boosting iterations stopping rule, by default equal to 2L.
<code>higher_order</code>	a logical that defines whether to compute the higher order fits (quadratic and cubic) after the FGB algorithm is run. Default is TRUE.
<code>boosting_with_memory</code>	logical value. If TRUE, boosting is performed taking into account previously fitted knots when fitting a GeDS learner at each new boosting iteration. If <code>boosting_with_memory</code> is TRUE, we recommend setting <code>int.knots_init = 1</code> and <code>int.knots_boost = 1</code> .

Details

The `NGeDSboost` function implements functional gradient boosting algorithm for some pre-defined loss function, using linear GeD splines as base learners. At each boosting iteration, the negative gradient vector is fitted through the base procedure encapsulated within the [NGeDS](#) function. The latter constructs a Geometrically Designed variable knots spline regression model for a response having a Normal distribution. The FGB algorithm yields a final linear fit. Higher order fits (quadratic and cubic) are then computed by calculating the Schoenberg's variation diminishing spline (VDS) approximation of the linear fit.

On the one hand, `NGeDSboost` includes all the parameters of [NGeDS](#), which in this case tune the base-learner fit at each boosting iteration. On the other hand, `NGeDSboost` includes some additional parameters proper to the FGB procedure. We describe the main ones as follows.

First, `family` allows to specify the loss function and corresponding risk function to be optimized by the boosting algorithm. If `initial_learner = FALSE`, the initial learner employed will be the empirical risk minimizer corresponding to the family chosen. If `initial_learner = TRUE` then the initial learner will be an [NGeDS](#) fit with maximum number of internal knots equal to `int.knots_init`.

`shrinkage` tunes the step length/shrinkage parameter which helps to control the learning rate of the model. In other words, when a new base learner is added to the ensemble, its contribution

to the final prediction is multiplied by the shrinkage parameter. The smaller shrinkage is, the slower/more gradual the learning process will be, and viceversa.

The number of boosting iterations is controlled by a *Ratio of Deviances* stopping rule similar to the one presented for GGeDS. In the same way phi and q tune the stopping rule of GGeDS, phi_boost_exit and q_boost tune the stopping rule of NGeDSboost. The user can also manually control the number of boosting iterations through min_iterations and max_iterations.

Value

GeDSboost-Class object, i.e. a list of items that summarizes the main details of the fitted FGB-GeDS model. See GeDSboost-Class for details. Some S3 methods are available in order to make these objects tractable, such as coef, knots, print and predict. Also variable importance measures (bl_imp) and improved plotting facilities (visualize_boosting).

References

Friedman, J.H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, **29** (5), 1189–1232.

DOI: [doi:10.1214/aos/1013203451](https://doi.org/10.1214/aos/1013203451)

Bühlmann P., Yu B. (2003). Boosting With the L2 Loss. *Journal of the American Statistical Association*, **98**(462), 324–339. [doi:10.1198/016214503000125](https://doi.org/10.1198/016214503000125)

Bühlmann P., Hothorn T. (2007). Boosting Algorithms: Regularization, Prediction and Model Fitting. *Statistical Science*, **22**(4), 477 – 505.

DOI: [doi:10.1214/07STS242](https://doi.org/10.1214/07STS242)

Kaishev, V.K., Dimitrova, D.S., Haberman, S. and Verrall, R.J. (2016). Geometrically designed, variable knot regression splines. *Computational Statistics*, **31**, 1079–1105.

DOI: [doi:10.1007/s0018001506217](https://doi.org/10.1007/s0018001506217)

Dimitrova, D. S., Kaishev, V. K., Lattuada, A. and Verrall, R. J. (2023). Geometrically designed variable knot splines in generalized (non-)linear models. *Applied Mathematics and Computation*, **436**.

DOI: [doi:10.1016/j.amc.2022.127493](https://doi.org/10.1016/j.amc.2022.127493)

Dimitrova, D. S., Guillen, E. S. and Kaishev, V. K. (2024). **GeDS**: An R Package for Regression, Generalized Additive Models and Functional Gradient Boosting, based on Geometrically Designed (GeD) Splines. *Manuscript submitted for publication*.

See Also

NGeDS; GGeDS; GeDSboost-Class; S3 methods such as knots.GeDSboost; coef.GeDSboost; deviance.GeDSboost; predict.GeDSboost

Examples

```
##### Example 1 #####
# Generate a data sample for the response variable
# Y and the single covariate X
set.seed(123)
N <- 500
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
```

```

X <- sort(runif(N, min = -2, max = 2))
# Specify a model for the mean of Y to include only a component
# non-linear in X, defined by the function f_1
means <- f_1(X)
# Add (Normal) noise to the mean of Y
Y <- rnorm(N, means, sd = 0.2)
data = data.frame(X, Y)

# Fit a Normal FGB-GeDS regression using NGeDSboost

Gmodboost <- NGeDSboost(Y ~ f(X), data = data)
MSE_Gmodboost_linear <- mean((sapply(X, f_1) - Gmodboost$predictions$pred_linear)^2)
MSE_Gmodboost_quadratic <- mean((sapply(X, f_1) - Gmodboost$predictions$pred_quadratic)^2)
MSE_Gmodboost_cubic <- mean((sapply(X, f_1) - Gmodboost$predictions$pred_cubic)^2)

cat("\n", "MEAN SQUARED ERROR", "\n",
    "Linear NGeDSboost:", MSE_Gmodboost_linear, "\n",
    "Quadratic NGeDSboost:", MSE_Gmodboost_quadratic, "\n",
    "Cubic NGeDSboost:", MSE_Gmodboost_cubic, "\n")

# Compute predictions on new randomly generated data
X <- sort(runif(100, min = -2, max = 2))

pred_linear <- predict(Gmodboost, newdata = data.frame(X), n = 2L)
pred_quadratic <- predict(Gmodboost, newdata = data.frame(X), n = 3L)
pred_cubic <- predict(Gmodboost, newdata = data.frame(X), n = 4L)

MSE_Gmodboost_linear <- mean((sapply(X, f_1) - pred_linear)^2)
MSE_Gmodboost_quadratic <- mean((sapply(X, f_1) - pred_quadratic)^2)
MSE_Gmodboost_cubic <- mean((sapply(X, f_1) - pred_cubic)^2)
cat("\n", "MEAN SQUARED ERROR", "\n",
    "Linear NGeDSboost:", MSE_Gmodboost_linear, "\n",
    "Quadratic NGeDSboost:", MSE_Gmodboost_quadratic, "\n",
    "Cubic NGeDSboost:", MSE_Gmodboost_cubic, "\n")

## S3 methods for class 'GeDSboost'
# Print
print(Gmodboost)
# Knots
knots(Gmodboost, n = 2L)
knots(Gmodboost, n = 3L)
knots(Gmodboost, n = 4L)
# Coefficients
coef(Gmodboost, n = 2L)
coef(Gmodboost, n = 3L)
coef(Gmodboost, n = 4L)
# Deviances
deviance(Gmodboost, n = 2L)
deviance(Gmodboost, n = 3L)
deviance(Gmodboost, n = 4L)

##### Example 2 - Bodyfat #####
library(TH.data)

```

```

data("bodyfat", package = "TH.data")

Gmodboost <- NGeDSboost(formula = DEXfat ~ age + f(hipcirc, waistcirc) + f(kneebreadth),
data = bodyfat)

MSE_Gmodboost_linear <- mean((bodyfat$DEXfat - Gmodboost$predictions$pred_linear)^2)
MSE_Gmodboost_quadratic <- mean((bodyfat$DEXfat - Gmodboost$predictions$pred_quadratic)^2)
MSE_Gmodboost_cubic <- mean((bodyfat$DEXfat - Gmodboost$predictions$pred_cubic)^2)
# Comparison
cat("\n", "MSE", "\n",
    "Linear NGeDSboost:", MSE_Gmodboost_linear, "\n",
    "Quadratic NGeDSboost:", MSE_Gmodboost_quadratic, "\n",
    "Cubic NGeDSboost:", MSE_Gmodboost_cubic, "\n")

```

NGeDSgam

NGeDSgam: Local Scoring Algorithm with GeD Splines in Backfitting

Description

Implements the Local Scoring Algorithm (Hastie and Tibshirani (1986)), applying normal GeD splines (i.e., [NGeDS](#) function) to fit the targets within the backfitting iterations.

Usage

```

NGeDSgam(
  formula,
  family = "gaussian",
  data,
  weights = NULL,
  offset = NULL,
  normalize_data = FALSE,
  min_iterations,
  max_iterations,
  phi_gam_exit = 0.995,
  q_gam = 2,
  beta = 0.5,
  phi = 0.99,
  internal_knots = 500,
  q = 2,
  higher_order = TRUE
)

```

Arguments

formula	a description of the structure of the model to be fitted, including the dependent and independent variables. Unlike NGeDS and GGeDS , the formula specified allows for multiple additive GeD spline regression components (as well as linear
---------	--

	components) to be included (e.g., $Y \sim f(X1) + f(X2) + X3$). See formula for further details.
family	a character string indicating the response variable distribution and link function to be used. Default is "gaussian". This should be a character or a family object.
data	a data frame containing the variables referenced in the formula.
weights	an optional vector of 'prior weights' to be put on the observations during the fitting process. It should be NULL or a numeric vector of the same length as the response variable defined in the formula.
offset	a vector of size N that can be used to specify a fixed component to be included in the linear predictor during fitting. In case more than one covariate is fixed, the user should sum the corresponding coordinates of the fixed covariates to produce one common N -vector of coordinates.
normalize_data	a logical that defines whether the data should be normalized (standardized) before fitting the baseline linear model, i.e., before running the local-scoring algorithm. Normalizing the data involves scaling the predictor variables to have a mean of 0 and a standard deviation of 1. This process alters the scale and interpretation of the knots and coefficients estimated. Default is equal to FALSE.
min_iterations	optional parameter to manually set a minimum number of boosting iterations to be run. If not specified, it defaults to 0L.
max_iterations	optional parameter to manually set the maximum number of boosting iterations to be run. If not specified, it defaults to 100L. This setting serves as a fallback when the stopping rule, based on consecutive deviances and tuned by <code>phi_gam_exit</code> and <code>q_gam</code> , does not trigger an earlier termination (see Dimitrova et al. (2024)). Therefore, users can increase/decrease the number of boosting iterations, by increasing/decreasing the value <code>phi_gam_exit</code> and/or <code>q_gam</code> , or directly specify <code>max_iterations</code> .
phi_gam_exit	Convergence threshold for local-scoring and backfitting. Both algorithms stop when the relative change in the deviance is below this threshold. Default is 0.995.
q_gam	numeric parameter which allows to fine-tune the stopping rule of local-scoring/backfitting, by default equal to 2L.
beta	numeric parameter in the interval $[0, 1]$ tuning the knot placement in stage A of GeDS. Default is equal to 0.5. See details in NGeDS .
phi	numeric parameter in the interval $[0, 1]$ specifying the threshold for the stopping rule (model selector) in stage A of GeDS. Default is equal to 0.99. See details in NGeDS .
internal_knots	The maximum number of internal knots that can be added by the GeDS base-learners in each boosting iteration, effectively setting the value of <code>max.intknots</code> in NGeDS at each backfitting iteration. Default is 500L.
q	numeric parameter which allows to fine-tune the stopping rule of stage A of GeDS, by default equal to 2L. See details in NGeDS .
higher_order	a logical that defines whether to compute the higher order fits (quadratic and cubic) after the local-scoring algorithm is run. Default is TRUE.

Details

The NGeDSgam function employs the local scoring algorithm to fit a Generalized Additive Model (GAM). This algorithm iteratively fits weighted additive models by backfitting. Normal linear GeD splines, as well as linear learners, are supported as function smoothers within the backfitting algorithm. The local-scoring algorithm ultimately produces a linear fit. Higher order fits (quadratic and cubic) are then computed by calculating the Schoenberg's variation diminishing spline (VDS) approximation of the linear fit.

On the one hand, NGeDSgam includes all the parameters of [NGeDS](#), which in this case tune the smoother fit at each backfitting iteration. On the other hand, NGeDSgam includes some additional parameters proper to the local-scoring procedure. We describe the main ones as follows.

The family chosen determines the link function, adjusted dependent variable and weights to be used in the local-scoring algorithm. The number of local-scoring and backfitting iterations is controlled by a *Ratio of Deviances* stopping rule similar to the one presented for [GGeDS](#). In the same way `phi` and `q` tune the stopping rule of [GGeDS](#), `phi_boost_exit` and `q_boost` tune the stopping rule of NGeDSgam. The user can also manually control the number of local-scoring iterations through `min_iterations` and `max_iterations`.

Value

[GeDSgam-Class](#) object, i.e. a list of items that summarizes the main details of the fitted GAM-GeDS model. See [GeDSgam-Class](#) for details. Some S3 methods are available in order to make these objects tractable, such as [coef](#), [knots](#), [print](#) and [predict](#).

References

Hastie, T. and Tibshirani, R. (1986). Generalized Additive Models. *Statistical Science* **1** (3) 297 - 310.

DOI: [doi:10.1214/ss/1177013604](https://doi.org/10.1214/ss/1177013604)

Kaishev, V.K., Dimitrova, D.S., Haberman, S. and Verrall, R.J. (2016). Geometrically designed, variable knot regression splines. *Computational Statistics*, **31**, 1079–1105.

DOI: [doi:10.1007/s0018001506217](https://doi.org/10.1007/s0018001506217)

Dimitrova, D. S., Kaishev, V. K., Lattuada, A. and Verrall, R. J. (2023). Geometrically designed variable knot splines in generalized (non-)linear models. *Applied Mathematics and Computation*, **436**.

DOI: [doi:10.1016/j.amc.2022.127493](https://doi.org/10.1016/j.amc.2022.127493)

Dimitrova, D. S., Guillen, E. S. and Kaishev, V. K. (2024). **GeDS**: An R Package for Regression, Generalized Additive Models and Functional Gradient Boosting, based on Geometrically Designed (GeD) Splines. *Manuscript submitted for publication*.

See Also

[NGeDS](#); [GGeDS](#); [GeDSgam-Class](#); S3 methods such as [knots.GeDSgam](#); [coef.GeDSgam](#); [deviance.GeDSgam](#); [predict.GeDSgam](#)

Examples

```
# Load package
```

```

library(GeDS)

data(airquality)
data = na.omit(airquality)
data$Ozone <- data$Ozone^(1/3)

formula = Ozone ~ f(Solar.R) + f(Wind, Temp)
Gmodgam <- NGeDSgam(formula = formula, data = data,
  phi_gam_exit = 0.995, phi = 0.995, q = 2)
MSE_Gmodgam_linear <- mean((data$Ozone - Gmodgam$predictions$pred_linear)^2)
MSE_Gmodgam_quadratic <- mean((data$Ozone - Gmodgam$predictions$pred_quadratic)^2)
MSE_Gmodgam_cubic <- mean((data$Ozone - Gmodgam$predictions$pred_cubic)^2)

cat("\n", "MEAN SQUARED ERROR", "\n",
  "Linear NGeDSgam:", MSE_Gmodgam_linear, "\n",
  "Quadratic NGeDSgam:", MSE_Gmodgam_quadratic, "\n",
  "Cubic NGeDSgam:", MSE_Gmodgam_cubic, "\n")

## S3 methods for class 'GeDSboost'
# Print
print(Gmodgam)
# Knots
knots(Gmodgam, n = 2L)
knots(Gmodgam, n = 3L)
knots(Gmodgam, n = 4L)
# Coefficients
coef(Gmodgam, n = 2L)
coef(Gmodgam, n = 3L)
coef(Gmodgam, n = 4L)
# Deviances
deviance(Gmodgam, n = 2L)
deviance(Gmodgam, n = 3L)
deviance(Gmodgam, n = 4L)

```

plot, GeDS-method

Plot method for GeDS objects.

Description

Plot method for GeDS objects. Plots GeDS fits.

Usage

```

## S4 method for signature 'GeDS'
plot(
  x,
  f = NULL,
  which,
  DEV = FALSE,

```

```

ask = FALSE,
main,
legend.pos = "topright",
new.window = FALSE,
wait = 0.5,
n = 3L,
type = c("none", "Polygon", "NCI", "ACI"),
...
)

```

Arguments

x	a GeDS-Class object from which the GeDS fit(s) should be extracted.
f	(optional) specifies the underlying function or generating process to which the model was fit. This parameter is useful if the user wishes to plot the specified function/process alongside the model fit and the data
which	a numeric vector specifying the iterations of stage A for which the corresponding GeDS fits should be plotted. It has to be a subset of 1:nrow(x\$stored). See details.
DEV	logical variable specifying whether a plot representing the deviance at each iteration of stage A should be produced or not.
ask	logical variable specifying whether the user should be prompted before changing the plot page.
main	optional character string to be used as a title of the plot.
legend.pos	the position of the legend within the panel. See legend for details.
new.window	logical variable specifying whether the plot should be shown in a new window or in the active one.
wait	time, in seconds, the system should wait before plotting a new page. Ignored if ask = TRUE.
n	integer value (2, 3 or 4) specifying the order (= degree + 1) of the GeDS fit that should be plotted. By default equal to 3L. Non-integer values will be passed to the function as.integer .
type	character string specifying the type of plot required. Should be set either to "Polygon" if the user wants to get also the control polygon of the GeDS fit, "NCI" or "ACI" if 95% confidence bands for the predictions should be plotted (see details) or "none" if only the fitted GeDS curve should be plotted. Applies only when plotting a univariate spline regression.
...	further arguments to be passed to the plot.default function.

Details

This method is provided in order to allow the user to plot the GeDS fits contained in the [GeDS-Class](#) objects.

Since in Stage A of the GeDS algorithm the knots of a linear spline fit are sequentially located, one at a time, the user may wish to visually inspect this process using the argument `which`. The latter

specifies a particular iteration number (or a vector of such numbers) for which the corresponding linear fit(s) should be plotted. The `ask` and `wait` arguments can help the user to manage these pages.

By means of `ask` the user can determine for how long each page should appear on the screen. Pages are sequentially replaced by pressing the enter button.

Note that, in order to ensure stability, if the object was produced by the function `GGeDS`, plotting intermediate fits of stage A is allowed only if $n = 2$, in contrast to objects produced by `NGeDS` for which plotting intermediate results is allowed also for $n = 2$ or 3 results.

The confidence intervals obtained by setting `type = "NCI"` are approximate local bands obtained considering the knots as fixed constants. Hence the columns of the design matrix are seen as covariates and standard methodology relying on the `se.fit` option of `predict.lm` or `predict.glm` is applied.

Setting `type = "ACI"`, asymptotic confidence intervals are plotted. This option is applicable only if the canonical link function has been used in the fitting procedure.

See Also

[NGeDS](#) and [GGeDS](#); [plot](#).

Examples

```
#####
# Generate a data sample for the response variable
# Y and the single covariate X, assuming Normal noise
set.seed(123)
N <- 500
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
X <- sort(runif(N, min = -2, max = 2))
# Specify a model for the mean of Y to include only a component
# non-linear in X, defined by the function f_1
means <- f_1(X)
# Add (Normal) noise to the mean of Y
Y <- rnorm(N, means, sd = 0.1)

# Fit a Normal GeDS regression using NGeDS
(Gmod <- NGeDS(Y ~ f(X), beta = 0.6, phi = 0.995, Xextr = c(-2,2)))

# Plot the final quadratic GeDS fit (red solid line)
# with its control polygon (blue dashed line)
plot(Gmod)

# Plot the quadratic fit obtained from the linear fit at the 10th
# iteration of stage A i.e. after 9 internal knots have been inserted
# by the GeDS procedure
plot(Gmod, which=10)

# Generate plots of all the intermediate fits obtained
# by running the GeDS procedure
## Not run:
plot(Gmod, which=1:16)
```

```

## End(Not run)

#####
# Generate a data sample for the response variable Y and the covariate
# X assuming Poisson distributed error and a log link function

set.seed(123)
N <- 500
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
X <- sort(runif(N ,min = -2, max = 2))
# Specify a model for the mean of Y to include only a component
# non-linear in X, defined by the function f_1
means <- exp(f_1(X))
# Generate Poisson distributed Y according to the mean model
Y <- rpois(N,means)

# Fit a Poisson GeDS regression model using GGeDS
(Gmod2 <- GGeDS(Y ~ f(X), beta = 0.2, phi = 0.995, family = poisson(),
               Xextr = c(-2,2)))

# similar plots as before, but for the linear fit
plot(Gmod2, n = 2)
plot(Gmod2, which = 10, n = 2)
## Not run:
plot(Gmod2, which = 1:16, n = 2)
plot(Gmod2, which = 1:16, n = 2, ask = T)

## End(Not run)

```

plot.GeDSboost

Plot method for non-additive GeDSboost

Description

Plot method for GeDSboost objects. Plots GeDSboost fits.

Usage

```

## S3 method for class 'GeDSboost'
plot(object, n = 3L, ...)

```

Arguments

object	the <code>GeDSboost-class</code>
n	integer value (2, 3 or 4) specifying the order (= degree +1) of the FGB-GeDS fit whose coefficients should be extracted.
...	potentially further arguments (required by the definition of the generic function). They will be ignored, but with a warning.

 PPolyRep

Piecewise Polynomial Spline Representation

Description

The function converts a GeDS fit which has a B-spline representation to a piecewise polynomial form.

Usage

```
PPolyRep(object, n = 3)
```

Arguments

object	the GeDS-class where the GeDS fit to be converted is found.
n	integer value (2, 3 or 4) specifying the order (= degree +1) of the GeDS fit which should be converted to a piecewise polynomial form. By default equal to 3L. Non-integer values will be passed to the function as.integer .

Details

This function converts a selected GeDS fit from a [GeDS-class](#) object represented in terms of B-splines into an object where the fit is represented in terms of piecewise polynomials.

The function wraps [polySpline](#) in order to let it accept [GeDS-class](#) objects as input. Hence the function provides a useful link between the package **GeDS** and the package **splines**, allowing the user to take advantage of the functions provided in the **splines** package.

Value

An object that inherits from classes "spline" and "polySpline". It is a list whose arguments are:

knots	a vector of size $k + 2$ containing the complete set of knots (internal knots plus the limits of the interval) of the GeDS fit.
coefficients	a $(k+2) \times n$ matrix containing the coefficients of the polynomials in the required piecewise polynomial representation.

Note

Let us note that the first $k + 1$ rows of the matrix contain the n coefficients of the $k + 1$ consecutive pieces of the piecewise polynomial representation. The last $(k + 2)$ -th row is extraneous and it appears as a result of the use of the function [polySpline](#).

Examples

```

# Generate a data sample for the response variable
# Y and the single covariate X
set.seed(123)
N <- 500
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
X <- sort(runif(N, min = -2, max = 2))
# Specify a model for the mean of Y to include only
# a component non-linear in X, defined by the function f_1
means <- f_1(X)
# Add (Normal) noise to the mean of Y
Y <- rnorm(N, means, sd = 0.1)

# Fit a Normal GeDS regression using NGeDS
Gmod <- NGeDS(Y ~ f(X), beta = 0.6, phi = 0.995, Xextr = c(-2,2))

# construct the PP representation of the cubic GeDS fit
# and apply some functions of the package splines
Polymod <- PPolyRep(Gmod, 4)
require(splines)
class(Polymod)
splineKnots(Polymod)
knots(Gmod, n = 4)
plot(Polymod)

# Generate a plot showing the PP representation
# based on the same example
knt <- splineKnots(Polymod)
coeffs <- coef(Polymod)
plot(Gmod, n = 4, legend = FALSE, main = "Cubic Curves")
cols <- sample(heat.colors(length(knt)), length(knt))
for(i in 1:(length(knt))){
  curve(coeffs[i,1] + coeffs[i,2]*(x - knt[i])+
        coeffs[i,3]*(x - knt[i])^2+
        coeffs[i,4]*(x - knt[i])^3,
        add = TRUE, col = cols[i])
  abline(v = knt[i])
}

```

predict.GeDS

Predict method for GeDS objects

Description

This is a user friendly method to compute predictions from GeDS objects.

Usage

```
## S3 method for class 'GeDS'
predict(object, newdata, type = c("response", "link", "terms"), n = 3L, ...)
```

Arguments

object	the GeDS-class object for which the computation of the predicted values is required.
newdata	an optional <code>data.frame</code> , <code>list</code> or environment containing values of the independent variables for which predicted values of the predictor model (including the GeDS and the parametric components) should be computed. If left empty the values are extracted from the object <code>x</code> itself.
type	character string indicating the type of prediction required. By default it is equal to "response", i.e. the result is on the scale of the response variable. See details for the other options.
n	integer value (2, 3 or 4) specifying the order (= degree +1) of the GeDS fit whose predicted values should be computed. By default equal to 3L. Non-integer values will be passed to the function as.integer .
...	potentially further arguments (required by the definition of the generic function). They are ignored, but with a warning.

Details

This is a method for the function [predict](#) that allows the user to handle [GeDS-Class](#) objects.

In analogy with the function [predict.glm](#) in the `stats` package, the user can specify the scale on which the predictions should be computed through the argument `type`. If the predictions are required to be on the scale of the response variable, the user should set `type = "response"`, which is the default. Alternatively if one wants the predictions to be on the predictor scale, it is necessary to set `type = "link"`. By specifying `type = "terms"`, it is possible to inspect the predicted values separately for each single independent variable which enter either the GeD spline component or the parametric component of the predictor model. In this case the returned result is a matrix whose columns correspond to the terms supplied via `newdata` or extracted from the object.

As GeDS objects contain three different fits (linear, quadratic and cubic), it is possible to specify the order for which GeDS predictions are required via the input argument `n`.

Value

A numeric vector corresponding to the predicted values (if `type = "link"` or `type = "response"`). If `type = "terms"` a numeric matrix with a column per term.

See Also

[predict](#) for the standard definition; [GGeDS](#) for examples.

predict.GeDSboost,gam *Predict method for GeDSboost, GeDSgam*

Description

This method computes predictions from GeDSboost and GeDSgam objects. It is designed to be user-friendly and accommodate different orders of the GeDSboost or GeDSgam fits.

Usage

```
## S3 method for class 'GeDSboost'
predict(object, newdata, n = 2L, ...)
```

```
## S3 method for class 'GeDSgam'
predict(object, newdata, n = 2L, ...)
```

Arguments

object	The GeDSboost-class or GeDSgam-class object.
newdata	An optional data frame for prediction.
n	The order of the GeDS fit (2 for linear, 3 for quadratic, and 4 for cubic). Default is 2.
...	potentially further arguments.

Value

Numeric vector of predictions (vector of means).

print.GeDS *Print method for GeDS, GeDSboost, GeDSgam*

Description

Method for the generic function [print](#) that allows to print on screen the main information related to the fitted predictor model that can be extracted from a [GeDS-class](#), [GeDSboost-class](#) or [GeDSgam-class](#) object.

Usage

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

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

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

Arguments

x	the GeDS-class , GeDSboost-class or GeDSgam-class object for which the main information should be printed on screen.
digits	number of digits to be printed.
...	potentially further arguments (required by the definition of the generic function).

Details

This method allows to print on screen basic information related to the fitted predictor model such as the function call, the number of internal knots for the linear GeDS/FGB-GeDS/GAM-GeDS fit and the deviances for the three (linear, quadratic and cubic) fitted predictor models embedded in the [GeDSboost-class](#) or [GeDSgam-class](#) object.

Value

This function returns (invisibly) the same input object, but adding the slot `Print` that contains the three sub-slots:

Nknots	the number of internal knots of the linear GeDS/FGB-GeDS/GAM-GeDS fit
Deviances	the deviances of the three (linear, quadratic and cubic) GeDS/FGB-GeDS/GAM-GeDS fits
Call	the call to the function that produced the x object

See Also

[print](#) for the standard definition.

SplineReg	<i>Estimation of the coefficients of a predictor model with spline and possibly parametric components.</i>
-----------	--

Description

Functions that estimate the coefficients of a predictor model involving a spline component and possibly a parametric component applying (Iteratively Re-weighted) Least Squares (IR)LS iteration.

Usage

```
SplineReg_LM(
  X,
  Y,
  Z = NULL,
  offset = rep(0, length(X)),
  weights = rep(1, length(X)),
  InterKnots,
  n,
```

```

    extr = range(X),
    prob = 0.95,
    coefficients = NULL,
    only_predictions = FALSE
  )

SplineReg_GLM(
  X,
  Y,
  Z,
  offset = rep(0, nobs),
  weights = rep(1, length(X)),
  InterKnots,
  n,
  extr = range(X),
  family,
  mustart,
  inits = NULL,
  etastart = NULL
)

```

Arguments

X	a numeric vector containing N sample values of the covariate chosen to enter the spline regression component of the predictor model.
Y	a vector of size N containing the observed values of the response variable y .
Z	a design matrix with N rows containing other covariates selected to enter the parametric component of the predictor model (see formula). If no such covariates are selected, it is set to NULL by default.
offset	a vector of size N that can be used to specify a fixed covariate to be included in the predictor model avoiding the estimation of its corresponding regression coefficient. In case more than one covariate is fixed, the user should sum the corresponding coordinates of the fixed covariates to produce one common N -vector of coordinates. The argument <code>offset</code> is particularly useful in <code>Splinerreg_GLM</code> if the link function used is not the identity.
weights	an optional vector of ‘prior weights’ to be put on the observations in the fitting process in case the user requires weighted fitting. It is a vector of 1s by default.
InterKnots	a numeric vector containing the locations of the internal knots necessary to compute the B-splines. In GeDS these are the internal knots in a current iteration of stage A.
n	integer value specifying the order of the spline to be evaluated. It should be 2 (linear spline), 3 (quadratic spline) or 4 (cubic spline). Non-integer values will be passed to the function as.integer .
extr	optional numeric vector of 2 elements representing the left-most and right-most limits of the interval embedding the sample values of X . By default equal correspondingly to the smallest and largest values of X .

prob	the confidence level to be used for the confidence bands in the SplineReg_LM fit. See details below.
coefficients	optional vector of spline coefficients. If provided, SplineReg computes only the corresponding predicted values.
only_predictions	logical, if TRUE only Theta, Predicted, Residuals and RSS will be computed.
family	a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. See family for details of family functions.
mustart	initial values for the vector of means in the IRLS estimation. Must be a vector of length N .
inits	a numeric vector of length $\text{length}(\text{InterKnots}) + n + \text{NCOL}(Z)$ providing initial values for the coefficients, to be used in the IRLS estimation (alternative to providing the mustart vector).
etastart	initial values for the predictor in the IRLS estimation (alternative to providing either inits or mustart). Must be a vector of length N .

Details

The functions estimate the coefficients of a predictor model with a spline component (and possibly a parametric component) for a given, fixed order and vector of knots of the spline and a specified distribution of the response variable (from the Exponential Family). The functions `SplineReg_LM` and `SplineReg_GLM` are based correspondingly on LS and IRLS and used correspondingly in `NGeDS` and `GGeDS`, to estimate the coefficients of the final GeDS fits of stage B, after their knots have been positioned to coincide with the Greville abscissas of the knots of the linear fit from stage A (see Dimitrova et al. 2023). Additional inference related quantities are also computed (see Value below). The function `SplineReg_GLM` is also used to estimate the coefficients of the linear GeDS fit of stage A within `GGeDS`, whereas in `NGeDS` this estimation is performed internally leading to faster R code.

In addition `SplineReg_LM` computes some useful quantities among which confidence intervals and the Control Polygon (see Section 2 of Kaishev et al. 2016).

The confidence intervals contained in the output slot `NCI` are approximate local bands obtained considering the knots as fixed constants. Hence the columns of the design matrix are seen as covariates and standard methodology relying on the `se.fit` option of `predict.lm` or `predict.glm` is used. In the `ACI` slot, asymptotic confidence intervals are provided, following Kaishev et al (2006).

As mentioned, `SplineReg_GLM` is intensively used in Stage A of the GeDS algorithm implemented in `GGeDS` and in order to make it as fast as possible input data validation is mild. Hence it is expected that the user checks carefully the input parameters before using `SplineReg_GLM`. The "Residuals" in the output of this function are similar to the so called "working residuals" in the `glm` function. "Residuals" are the residuals r_i used in the knot placement procedure, i.e.

$$r_i = (y_i - \hat{\mu}_i) \frac{d\mu_i}{d\eta_i},$$

but in contrast to `glm` "working residuals", they are computed using the final IRLS fitted $\hat{\mu}_i$. "Residuals" are then used in locating the knots of the linear spline fit of Stage A.

In `SplineReg_GLM` confidence intervals are not computed.

Value

A list containing:

Theta	a vector containing the fitted coefficients of the spline regression component and the parametric component of the predictor model.
Predicted	a vector of N predicted mean values of the response variable computed at the sample values of the covariate(s).
Residuals	a vector containing the normal regression residuals if <code>SplineReg_LM</code> is called or the residuals described in Details if <code>SplineReg_GLM</code> is called.
RSS	the deviance for the fitted predictor model, defined as in Dimitrova et al. (2023), which for <code>SplineReg_LM</code> coincides with the Residual Sum of Squares.
NCI	a list containing the lower (Low) and upper (Upp) limits of the approximate confidence intervals computed at the sample values of the covariate(s). See details above.
Basis	the matrix of B-spline regression functions and the covariates of the parametric part evaluated at the sample values of the covariate(s).
Polygon	a list containing x-y coordinates ("Kn" and "Thetas") of the vertices of the Control Polygon, see Dimitrova et al. (2023).
deviance	a vector containing deviances computed at each IRLS step (computed only with the <code>SplineReg_GLM</code>).
temporary	the result of the function <code>lm</code> if <code>SplineReg_LM</code> is used or the output of the function <code>IRLSfit</code> (which is similar to the output from <code>glm.fit</code>), if <code>SplineReg_GLM</code> is used.

References

- Kaishev, V. K., Dimitrova, D. S., Haberman, S. & Verrall, R. J. (2006). Geometrically designed, variable knot regression splines: asymptotics and inference (*Statistical Research Paper No. 28*). London, UK: Faculty of Actuarial Science & Insurance, City University London.
URL: openaccess.city.ac.uk
- Kaishev, V.K., Dimitrova, D.S., Haberman, S., & Verrall, R.J. (2016). Geometrically designed, variable knot regression splines. *Computational Statistics*, **31**, 1079–1105.
DOI: [doi:10.1007/s0018001506217](https://doi.org/10.1007/s0018001506217)
- Dimitrova, D. S., Kaishev, V. K., Lattuada, A. and Verrall, R. J. (2023). Geometrically designed variable knot splines in generalized (non-)linear models. *Applied Mathematics and Computation*, **436**.
DOI: [doi:10.1016/j.amc.2022.127493](https://doi.org/10.1016/j.amc.2022.127493)

See Also

[NGeDS](#), [GGeDS](#), [Fitters](#), [IRLSfit](#), [lm](#) and [glm.fit](#).

UnivariateFitters	<i>Functions used to fit GeDS objects w/univariate spline regression component</i>
-------------------	--

Description

These are computing engines called by [NGeDS](#) and [GGeDS](#), needed for the underlying fitting procedures.

Usage

```
UnivariateFitter(
  X,
  Y,
  Z = NULL,
  offset = rep(0, NROW(Y)),
  weights = rep(1, length(X)),
  beta = 0.5,
  phi = 0.5,
  min.intknots = 0,
  max.intknots = 300,
  q = 2,
  extr = range(X),
  show.iters = FALSE,
  tol = as.double(1e-12),
  stoptype = c("SR", "RD", "LR"),
  higher_order = TRUE,
  intknots = NULL,
  only_predictions = FALSE
)
```

```
GenUnivariateFitter(
  X,
  Y,
  Z = NULL,
  offset = rep(0, NROW(Y)),
  weights = rep(1, length(X)),
  family = gaussian(),
  beta = 0.5,
  phi = 0.5,
  min.intknots = 0,
  max.intknots = 300,
  q = 2,
  extr = range(X),
  show.iters = F,
  tol = as.double(1e-12),
  stoptype = c("SR", "RD", "LR"),
```

```

    higher_order = TRUE
)

```

Arguments

X	a numeric vector containing N sample values of the covariate chosen to enter the spline regression component of the predictor model.
Y	a vector of size N containing the observed values of the response variable y .
Z	a design matrix with N rows containing other covariates selected to enter the parametric component of the predictor model (see formula). If no such covariates are selected, it is set to NULL by default.
offset	a vector of size N that can be used to specify a fixed covariate to be included in the predictor model avoiding the estimation of its corresponding regression coefficient. In case more than one covariate is fixed, the user should sum the corresponding coordinates of the fixed covariates to produce one common N -vector of coordinates. The offset argument is particularly useful when using <code>GenUnivariateFitter</code> if the link function used is not the identity.
weights	an optional vector of size N of ‘prior weights’ to be put on the observations in the fitting process in case the user requires weighted GeDS fitting. It is NULL by default.
beta	numeric parameter in the interval $[0, 1]$ tuning the knot placement in stage A of GeDS. See the description of NGeDS or GGeDS .
phi	numeric parameter in the interval $[0, 1]$ specifying the threshold for the stopping rule (model selector) in stage A of GeDS. See also <code>stoptype</code> and details in the description of NGeDS or GGeDS .
min.intknots	optional parameter allowing the user to set a minimum number of internal knots required. By default equal to zero.
max.intknots	optional parameter allowing the user to set a maximum number of internal knots to be added by the GeDS estimation algorithm. By default equal to the number of internal knots κ for the saturated GeDS model (i.e. $\kappa = N - 2$).
q	numeric parameter which allows to fine-tune the stopping rule of stage A of GeDS, by default equal to 2. See details in the description of NGeDS or GGeDS .
extr	numeric vector of 2 elements representing the left-most and right-most limits of the interval embedding the sample values of X. By default equal correspondingly to the smallest and largest values of X.
show.iters	logical variable indicating whether or not to print information at each step. By default equal to FALSE.
tol	numeric value indicating the tolerance to be used in the knot placement steps in stage A. By default equal to 1E-12. See details below.
stoptype	a character string indicating the type of GeDS stopping rule to be used. It should be either "SR", "RD" or "LR", partial match allowed. See details of NGeDS or GGeDS .
higher_order	a logical that defines whether to compute the higher order fits (quadratic and cubic) after stage A is run. Default is TRUE.

intknots	vector of initial internal knots from which to start the GeDS Stage A iterations. See Section 3 of Kaishev et al. (2016). Default is NULL.
only_predictions	logical, if TRUE only predictions are computed.
family	a description of the error distribution and link function to be used in the model. This can be a character string naming a family function (e.g. "gaussian"), the family function itself (e.g. <code>gaussian</code>) or the result of a call to a family function (e.g. <code>gaussian()</code>). See family for details on family functions.

Details

The functions `UnivariateFitter` and `GenUnivariateFitter` are in general not intended to be used directly, they should be called through `NGeDS` and `GGeDS`. However, in case there is a need for multiple GeDS fitting (as may be the case e.g. in Monte Carlo simulations) it may be efficient to use the fitters outside the main functions.

The argument `tol` is used in the knot placement procedure of stage A of the GeDS algorithm in order to check whether the current knot δ^* is set at an acceptable location or not. If there exists a knot δ_i such that $|\delta^* - \delta_i| < \text{tol}$, δ^* , then the new knot is considered to be coalescent with an existing one, it is discarded and the algorithm seeks alternative knot locations. By default it is equal to $1e-12$.

See `NGeDS` and `GGeDS`, Kaishev et al. (2016) and Dimitrova et al. (2023) for further details.

Value

A `GeDS-Class` object, but without the `Formula`, `extcall`, `terms` and `znames` slots.

References

Kaishev, V.K., Dimitrova, D.S., Haberman, S., & Verrall, R.J. (2016). Geometrically designed, variable knot regression splines. *Computational Statistics*, **31**, 1079–1105.

DOI: [doi:10.1007/s0018001506217](https://doi.org/10.1007/s0018001506217)

Dimitrova, D. S., Kaishev, V. K., Lattuada, A. and Verrall, R. J. (2023). Geometrically designed variable knot splines in generalized (non-)linear models. *Applied Mathematics and Computation*, **436**.

DOI: [doi:10.1016/j.amc.2022.127493](https://doi.org/10.1016/j.amc.2022.127493)

See Also

[NGeDS](#) and [GGeDS](#).

Examples

```
# Examples similar to the ones
# presented in NGeDS and in GGeDS

# Generate a data sample for the response variable
# Y and the covariate X
set.seed(123)
N <- 500
```

```

f_1 <- function(x) (10*x/(1+100*x^2))*4+4
X <- sort(runif(N ,min = -2, max = 2))
# Specify a model for the mean of Y to include only
# a component non-linear in X, defined by the function f_1
means <- f_1(X)
# Add (Normal) noise to the mean of Y
Y <- rnorm(N, means, sd = 0.1)

# Fit a Normal GeDS regression model using the fitter function
(Gmod <- UnivariateFitter(X, Y, beta = 0.6, phi = 0.995,
  extr = c(-2,2)))

#####
# second: very similar example, but based on Poisson data
set.seed(123)
X <- sort(runif(N , min = -2, max = 2))
means <- exp(f_1(X))
Y <- rpois(N,means)
(Gmod2 <- GenUnivariateFitter(X, Y, beta = 0.2,
  phi = 0.995, family = poisson(), extr = c(-2,2)))

# a plot showing quadratic and cubic fits,
# in the predictor scale
plot(X,log(Y), xlab = "x", ylab = expression(f[1](x)))
lines(Gmod2, n = 3, col = "red")
lines(Gmod2, n = 4, col = "blue", lty = 2)
legend("topleft", c("Quadratic","Cubic"),
  col = c("red","blue"), lty = c(1,2))

```

visualize_boosting *Visualize Boosting Iterations*

Description

This function plots the [NGeDSboost](#) fit to the data at the beginning of a given boosting iteration and then plots the subsequent [NGeDS](#) fit on the corresponding residual (negative gradient). Note: Applicable only for [NGeDSboost](#) models with one covariate and family = `mboost::Gaussian()`.

Usage

```

## S3 method for class 'GeDSboost'
visualize_boosting(object, iters = NULL, final_fits = FALSE)

```

Arguments

object	a GeDSboost-Class object.
iters	numeric, specifies the iteration(s) number.
final_fits	logical indicating whether the final linear, quadratic and cubic fits should be plotted.

Examples

```

# Load packages
library(GeDS)

# Generate a data sample for the response variable
# Y and the single covariate X
set.seed(123)
N <- 500
f_1 <- function(x) (10*x/(1+100*x^2))*4+4
X <- sort(runif(N, min = -2, max = 2))
# Specify a model for the mean of Y to include only a component
# non-linear in X, defined by the function f_1
means <- f_1(X)
# Add (Normal) noise to the mean of Y
Y <- rnorm(N, means, sd = 0.2)
data = data.frame(X, Y)
Gmodboost <- NGeDSboost(Y ~ f(X), data = data, normalize_data = TRUE)

# Plot
plot(X, Y, pch=20, col=c("darkgrey"))
lines(X, sapply(X, f_1), col = "black", lwd = 2)
lines(X, Gmodboost$predictions$pred_linear, col = "green4", lwd = 2)
lines(X, Gmodboost$predictions$pred_quadratic, col="red", lwd=2)
lines(X, Gmodboost$predictions$pred_cubic, col="purple", lwd=2)
legend("topright",
legend = c("Order 2 (degree=1)", "Order 3 (degree=2)", "Order 4 (degree=3)"),
col = c("green4", "red", "purple"),
lty = c(1, 1),
lwd = c(2, 2, 2),
cex = 0.75,
bty="n",
bg = "white")
# Visualize boosting iterations + final fits
par(mfrow=c(4,2))
visualize_boosting(Gmodboost, iters = 0:3, final_fits = TRUE)
par(mfrow=c(1,1))

```

Index

- * **package**
 - GeDS-package, 3
- as.integer, [10](#), [12](#), [13](#), [15](#), [30](#), [33](#), [35](#), [49](#), [52](#), [54](#), [57](#)
- BaFe2As2, [5](#)
- BivariateFitter (BivariateFitters), [6](#)
- BivariateFitters, [6](#)
- bl_imp, [43](#)
- bl_imp(bl_imp.GeDSboost), [8](#)
- bl_imp.GeDSboost, [8](#)
- coalMining, [9](#)
- coef, [10–12](#), [26](#), [38](#), [43](#), [47](#)
- coef.GeDS, [10](#), [26](#), [38](#)
- coef.GeDSboost, [43](#)
- coef.GeDSboost (coef.GeDSboost, gam), [12](#)
- coef.GeDSboost, (coef.GeDSboost, gam), [12](#)
- coef.GeDSboost, gam, [12](#)
- coef.GeDSgam, [47](#)
- coef.GeDSgam (coef.GeDSboost, gam), [12](#)
- coefficients, [10–12](#)
- coefficients.GeDS (coef.GeDS), [10](#)
- coefficients.GeDSboost
 - (coef.GeDSboost, gam), [12](#)
- coefficients.GeDSgam
 - (coef.GeDSboost, gam), [12](#)
- crossv_GeDS, [13](#)
- Derive, [13](#), [26](#), [38](#)
- deviance, [14](#), [15](#), [26](#), [38](#)
- deviance.GeDS, [14](#), [26](#), [38](#)
- deviance.GeDSboost, [43](#)
- deviance.GeDSboost (deviance.GeDS), [14](#)
- deviance.GeDSgam, [47](#)
- deviance.GeDSgam (deviance.GeDS), [14](#)
- EWmortality, [15](#)
- f, [3](#), [16](#), [18](#)
- Family, [41](#)
- family, [8](#), [24](#), [32](#), [58](#), [62](#)
- Fitters, [19](#), [59](#)
- Fitters (UnivariateFitters), [60](#)
- formula, [3](#), [7](#), [11](#), [16–19](#), [24](#), [25](#), [35–37](#), [41](#), [46](#), [57](#), [61](#)
- formula.GeDS, [17](#)
- gaussian, [8](#), [24](#), [32](#), [62](#)
- GeDS (GeDS-package), [3](#)
- GeDS-Class, [38](#)
- GeDS-Class (GeDS-class), [18](#)
- GeDS-class, [18](#)
- GeDS-package, [3](#)
- GeDSboost-Class (GeDSboost-class), [20](#)
- GeDSboost-class, [20](#)
- GeDSgam-Class (GeDSgam-class), [22](#)
- GeDSgam-class, [22](#)
- GenBivariateFitter (BivariateFitters), [6](#)
- GenUnivariateFitter, [19](#)
- GenUnivariateFitter
 - (UnivariateFitters), [60](#)
- GGeDS, [3](#), [7](#), [13](#), [15–19](#), [23](#), [30](#), [34](#), [35](#), [37](#), [38](#), [41](#), [43](#), [45](#), [47](#), [50](#), [54](#), [58–62](#)
- glm, [18](#), [32](#), [58](#)
- glm.control, [32](#)
- glm.fit, [32](#), [33](#), [59](#)
- Integrate, [26](#), [29](#), [38](#)
- IRLSfit, [25](#), [31](#), [59](#)
- knots, [26](#), [33](#), [34](#), [38](#), [43](#), [47](#)
- knots.GeDS, [26](#), [33](#), [38](#)
- knots.GeDSboost, [43](#)
- knots.GeDSboost (knots.GeDS), [33](#)
- knots.GeDSboost, (knots.GeDS), [33](#)
- knots.GeDSgam, [47](#)
- knots.GeDSgam (knots.GeDS), [33](#)
- Legend, [49](#)

lines, [26](#), [35](#), [38](#)
lines, GeDS-method, [34](#)
lines.GeDS (lines, GeDS-method), [34](#)
lm, [18](#), [37](#), [59](#)
lm.fit, [33](#)

NGeDS, [3](#), [6–8](#), [11](#), [13](#), [16–19](#), [21](#), [25](#), [26](#), [30](#),
[34](#), [35](#), [36](#), [40–43](#), [45–47](#), [50](#), [58–63](#)
NGeDSboost, [3](#), [12](#), [20](#), [34](#), [40](#), [63](#)
NGeDSgam, [3](#), [12](#), [22](#), [34](#), [45](#)

offset, [18](#), [25](#), [37](#)

plot, [26](#), [38](#), [50](#)
plot, GeDS, ANY-method
 (plot, GeDS-method), [48](#)
plot, GeDS-method, [48](#)
plot.default, [49](#)
plot.GeDS (plot, GeDS-method), [48](#)
plot.GeDSboost, [51](#)
polySpline, [52](#)
PPolyRep, [26](#), [38](#), [52](#)
predict, [26](#), [38](#), [43](#), [47](#), [54](#)
predict.GeDS, [26](#), [38](#), [53](#)
predict.GeDSboost, [43](#)
predict.GeDSboost
 (predict.GeDSboost, gam), [55](#)
predict.GeDSboost,
 (predict.GeDSboost, gam), [55](#)
predict.GeDSboost, gam, [55](#)
predict.GeDSgam, [47](#)
predict.GeDSgam
 (predict.GeDSboost, gam), [55](#)
predict.glm, [54](#)
print, [26](#), [38](#), [43](#), [47](#), [55](#), [56](#)
print.GeDS, [26](#), [38](#), [55](#)
print.GeDSboost (print.GeDS), [55](#)
print.GeDSgam (print.GeDS), [55](#)

splineDesign, [14](#)
SplineReg, [19](#), [21](#), [23](#), [56](#)
SplineReg_GLM, [31](#)
SplineReg_GLM (SplineReg), [56](#)
SplineReg_LM (SplineReg), [56](#)

UnivariateFitter (UnivariateFitters), [60](#)
UnivariateFitters, [8](#), [60](#)

varimp, [8](#), [9](#)
visualize_boosting, [43](#), [63](#)