Package ‘GeDS’

June 30, 2025

Type Package

Title Geometrically Designed Spline Regression

Version 0.3.3

Date 2025-06-30

Maintainer Emilio L. Sáenz Guillén
<Emilio.Saenz-Guillen@citystgeorges.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 model 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. (2025).

License GPL-3

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

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

Depends R (>= 4.4.0)

Imports doFuture, doParallel, doRNG, foreach, future, graphics, grDevices, MASS, Matrix, mboost, parallel, plot3D, Rcpp, splines, stats, utils

Suggests knitr, R.rsp, rmarkdown, testthat (>= 3.0.0), TH.data

LinkingTo Rcpp

VignetteBuilder R.rsp

Config/testthat/edition 3

Encoding UTF-8

LazyData TRUE

RoxygenNote 7.3.2

NeedsCompilation yes

Author Dimitrina S. Dimitrova [aut],
Vladimir K. Kaishev [aut],
Andrea Lattuada [aut],
Emilio L. Sáenz Guillén [aut, cre],
Richard J. Verrall [aut]

Repository CRAN

Date/Publication 2025-06-30 07:10:06 UTC

Contents

`GeDS-package` *Geometrically Designed Spline Regression*

Description

Geometrically designed spline (GeDS) regression is a non-parametric method for fitting spline regression models with variable knots. The GeDS technique is inspired by geometric principles and 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

GeDS provides a novel solution to the spline regression problem and, in particular, to the problem of estimating the number and position of the knots. The GeDS estimation method is based on two stages: first, in stage A, a piecewise linear fit (spline fit of order 2) capturing the underlying functional shape determined by the data is constructed; second, in stage B, the latter fit is approximated through shape preserving (variation diminishing) spline fits of higher orders ($n = 3$, $n = 4$,. . . , i.e., degrees 2, 3,. . .). As a result, GeDS simultaneously produces a linear, a quadratic and a cubic spline fit.

The GeDS method was originally developed by Kaishev et al. (2016) for the univariate case, 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 provides an enhanced implementation of the original normal GeDS *Mathematica* code, through the `NGeDS` function; it also includes a generalization, `GGeDS`, which extends the method to any distribution in 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. (2025) have recently made significant enhancements to the **GeDS** methodology, by incorporating generalized additive models (GAM-GeDS) and functional gradient boosting (FGB-GeDS). 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 through the function `NGeDSgam`. On the other hand, the **GeDS** package incorporates functional gradient descent algorithm by utilizing normal GeD splines (i.e., `NGeDS`) as base learners within the boosting iterations. Unlike typical boosting methods, the final FGB-GeDS model is expressed as a single spline model rather than as a sum of base-learner fits. For this, `NGeDSboost` leverages the piecewise polynomial representation of B-splines, and, at each boosting iteration, performs a piecewise update of the corresponding polynomial coefficients.

The outputs of both `NGeDS` and `GGeDS` functions are `"GeDS"` class objects, while the outputs of `NGeDSgam` and `NGeDSboost` functions are`"GeDSgam"` class and `"GeDSboost"` class objects, respectively. `"GeDS"` class, `"GeDSgam"` class and `"GeDSboost"` class objects contain second, third and fourth order spline fits. As described in Kaishev et al. (2016), Dimitrova et al. (2023) and Dimitrova et al. (2025), the "final" GeDS fit is the one minimizing the empirical deviance. Nevertheless, the user can choose to use any of the available fits.

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 `NGeDS`/`GGeDS`, `NGeDSgam` and `NGeDSboost`, 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@citystgeorges.ac.uk`>.

Author(s)

Dimitrina S. Dimitrova <D.Dimitrova@citystgeorges.ac.uk>,
Vladimir K. Kaishev <V.Kaishev@citystgeorges.ac.uk>,
Andrea Lattuada <Andrea.Lattuada@hotmail.com>,
Emilio L. Sáenz Guillén <Emilio.Saenz-Guillen@citystgeorges.ac.uk> and
Richard J. Verrall <R.J.Verrall@citystgeorges.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

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

Dimitrova, D. S., Kaishev, V. K. and Saenz Guillen, E. L. (2025). **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/emilioluissaenzguillen/GeDS
- Report bugs at https://github.com/emilioluissaenzguillen/GeDS/issues

---

`BaFe2As2` *Barium-Ferrum-Arsenide Powder Diffraction Data*

Description

This dataset contains the results of a neutron diffraction experiment on Barium-Ferrum-Arsenide ($BaFe_2As_2$) 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 $BaFe_2As_2$. *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

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)
```

`BivariateFitters` *Fitter Function for GeD Spline Regression for Bivariate Data*

Description

These are computing engines called by NGeDS and GGeDS, 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 = 0L,
  max.intknots = 300L,
  q = 2L,
  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 = 0L,
  max.intknots = 300L,
  q = 2L,
  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`	A contingency table (i.e., frequency of observations) for the independent variables `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 specifying the minimum number of internal knots required in Stage A's fit. Default is `0L`.
`max.intknots`	Optional parameter allowing the user to set a maximum number of internal knots to be added in Stage A by the GeDS estimation algorithm. Default equals `300L`.
`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 fitting information at each step. Default is `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 defining whether to compute the higher order fits (quadratic and cubic) after stage A is run. Default is TRUE.

`Xintknots` A vector of starting internal knots in the X direction. Allows the user to begin Stage A's GeDS algorithm with a linear (least-squares) spline fit using a predefined vector of internal X knots, instead of starting with a straight line fit (i.e., with zero internal knots). Note that this is not available for GenBivariateFitter. Default is NULL.

`Yintknots` A vector of starting internal knots in the Y direction. Allows the user to begin Stage A's GeDS algorithm with a linear (least-squares) spline fit using a predefined vector of internal Y knots, instead of starting with a straight line fit (i.e., with zero internal knots). Note that this is not available for GenBivariateFitter. 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. Note that this argument applies only to GenBivariateFitter.

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

See Also

NGeDS, GGeDS and UnivariateFitters.

`bl_imp.GeDSboost` *Base Learner Importance for GeDSboost Objects*

Description

S3 method for "GeDSboost" class objects that calculates the in-bag risk reduction ascribable to each base-learner of an FGB-GeDS model. Essentially, it measures and aggregates the decrease in the empirical risk attributable to each base-learner for every time it is selected across the boosting iterations. 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 the empiral risk (loss function). 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 a potential initial GeDS 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)

data("bodyfat", package = "TH.data")
N <- nrow(bodyfat); ratio <- 0.8
set.seed(123)
trainIndex <- sample(1:N, size = floor(ratio * N))
# Subset the data into training and test sets
train <- bodyfat[trainIndex, ]
test <- bodyfat[-trainIndex, ]

Gmodboost <- NGeDSboost(formula = DEXfat ~ f(hipcirc) + f(kneebreadth) + f(anthro3a),
                        data = train, phi = 0.7, initial_learner = FALSE)
MSE_Gmodboost_linear <- mean((test$DEXfat - predict(Gmodboost, newdata = test, n = 2))^2)
MSE_Gmodboost_quadratic <- mean((test$DEXfat - predict(Gmodboost, newdata = test, n = 3))^2)
MSE_Gmodboost_cubic <- mean((test$DEXfat - predict(Gmodboost, newdata = test, n = 4))^2)

# Print MSE
cat("\n", "TEST 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 the 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

Method for the function `coef` that allows to extract the estimated coefficients of a fitted GeDS regression model from a "GeDS" class object.

Usage

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

Arguments

`object`	The "GeDS" class object from which the coefficients of the selected GeDS regression model should be extracted.
`n`	Integer value (2, 3 or 4) specifying the order (= degree +1) of the "GeDS", "GeDSgam" or "GeDSboost" fit whose coefficients should be extracted. By default equal to 3L; non-integer values will be passed to the function `as.integer`.
`onlySpline`	Logical variable specifying whether only the coefficients for the GeDS component of a fitted multivariate regression model should be extracted or whether the coefficients of both the GeDS and the parametric components should be returned.
`...`	Potentially further arguments (required by the definition of the generic function). These will be ignored, but with a warning.

Details

Simple method for the function `coef`.

As "GeDS" class objects contain three different fits (linear, quadratic and cubic), the argument `n` can be used to specify the order of the GeDS fit for which regression coefficients are required.

As mentioned in the Details of `formula`, the predictor model may be multivariate and it may include a (univariate or bivariate) GeD spline component, plus a parametric component involving the remaining variables. 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 univariate or 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 more 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.GeDSgam,boost *Coef Method for GeDSgam, GeDSboost*

Description

Method for the function coef that allows to extract the estimated coefficients of a "GeDSgam" class or "GeDSboost" class object.

Usage

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

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

Arguments

`object` The "GeDSgam" class or "GeDSboost" class object from which the coefficients should be extracted.

`n` Integer value (2, 3 or 4) specifying the order (= degree +1) of the FGB-GeDS/GAM-GeDS fit whose coefficients should be extracted.

- If `n = 2L`: the piecewise polynomial coefficients of the univariate GeDS base-learners and linear base-learners are provided (the B-spline coefficients are accessible through `x$final_model$linear.fit$theta`); for bivariate GeDS base-learners and `class(object) == "GeDSboost"`, the B-spline coefficients for each iteration where the corresponding bivariate base-learner was selected are provided; for bivariate base-learners and `class(object) == "GeDSgam"`, the final local-scoring B-spline coefficients are provided.
- If `n = 3L` or `n = 4L`: B-spline coefficients are provided for both univariate and bivariate GeDS learners.

By default `n` is 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). These 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; `NGeDSgam` and `NGeDSboost` for examples.

Examples

```
data(mtcars)
# Convert specified variables to factors
categorical_vars <- c("cyl", "vs", "am", "gear", "carb")
mtcars[categorical_vars] <- lapply(mtcars[categorical_vars], factor)

Gmodgam <- NGeDSgam(mpg ~ f(disp, hp) + f(qsec) + carb,
                    data = mtcars, family = gaussian, phi = 0.7)

# Piecewise polynomial coefficients of the (univariate) GeDS and linear learners
coef(Gmodgam, n = 2)$`f(qsec)`
coef(Gmodgam, n = 2)$carb
# B-spline coefficients of the bivariate learner
coef(Gmodgam, n = 2)$`f(disp, hp)`

Gmodboost <- NGeDSboost(mpg ~ f(disp, hp) + f(qsec) + carb,
                        data = mtcars, family = mboost::Gaussian(), shrinkage = 0.7)

# Piecewise polynomial coefficients of the (univariate) GeDS and linear learners
coef(Gmodboost, n = 2)$`f(qsec)`
coef(Gmodboost, n = 2)$carb
```

```
# B-spline coefficients of the bivariate learner at each boosting iteration
# this was selected
coef(Gmodboost, n = 2)$`f(disp, hp)`
```

`confint.GeDS` *Confidence Intervals for GeDS Models Coefficients*

Description

Method for `confint.default` to compute confidence intervals for the coefficients of a fitted GeDS model stored in a "GeDS", "GeDSgam" or "GeDSboost" class object.

Usage

```
## S3 method for class 'GeDS'
confint(object, parm, level = 0.95, n = 3L, ...)

## S3 method for class 'GeDSgam'
confint(object, parm, level = 0.95, n = 3L, ...)

## S3 method for class 'GeDSboost'
confint(object, parm, level = 0.95, n = 3L, ...)
```

Arguments

`object`	The "GeDS"/"GeDSgam"/"GeDSboost" class object from which the confidence intervals for the selected order `n` should be extracted.
`parm`	A specification of which parameters are to be given confidence intervals, either a vector of numbers or names; defaults to all parameters.
`level`	The confidence level required (default is 0.95).
`n`	Integer value (2, 3 or 4) specifying the order (= degree +1) of the "GeDS", "GeDSgam" or "GeDSboost" fit for which to compute confidence intervals. By default equal to `3L`; non-integer values will be passed to the function `as.integer`.
`...`	Additional arguments passed to `confint.default`.

Value

A matrix with columns giving lower and upper confidence limits for each spline coefficient of the selected GeDS model (by default 2.5% and 97.5%).

See Also

`confint.default`, `NGeDS`, `GGeDS`, `NGeDSgam`, `NGeDSboost` crossv_GeDS *K-Fold Cross-Validation*

Description

crossv_GeDS performs k-fold cross-validation for tuning the relevant parameters of NGeDS, GGeDS, NGeDSgam, and NGeDSboost functions.

Arguments

formula
: A description of the structure of the model structure, including the dependent and independent variables.

data
: A data.frame containing the variables referenced in the formula.

model_fun
: The GeDS model to cross-validate, that is, NGeDS, GGeDS, NGeDSgam or NGeDSboost.

parameters
: A set of parameters to be tuned via cross-validation. These are: beta, phi and q in the case of NGeDS, GGeDS and NGeDSgam. In addition, for NGeDSboost, int.knots_init and 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) and
  - 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 cross-validated MSE and the average number of internal knots across the folds for each possible combination of parameters, given the input parameters. In the case of model_fun = NGeDSboost, it also provides the cross-validated number of boosting iterations.

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)
data = data.frame(X = X, Y = Y)
```

```
## Not run:
## NGeDS
# Define different combinations of parameters to cross-validate
param = list(beta_grid = c(0.5),
             phi_grid = c(0.9, 0.95),
             q_grid = c(2))

cv_NGeDS <- crossv_GeDS(Y ~ f(X), data = data, NGeDS, n = 3,
                        parameters = param)

print(cv_NGeDS$best_params)
View(cv_NGeDS$results)

## NGeDSboost
param = list(int.knots_init_grid = c(1, 2),
             shrinkage_grid = 1,
             beta_grid = c(0.3, 0.5),
             phi_grid = c(0.95, 0.99),
             q_grid = 2)

cv_NGeDSboost <- crossv_GeDS(Y ~ f(X), data = data, NGeDSboost, n = 2L,
                             n_folds = 2L, parameters = param)

print(cv_NGeDSboost$best_params)
View(cv_NGeDSboost$results)

## End(Not run)
```

`CrystalData` *Crystallographic Scattering Data*

Description

This dataset contains crystallographic measurements obtained from a particle accelerator experiment. The measurements correspond to the function $F(Q)$ at various Q values, which are used to analyze the scattering properties of an unknown crystalline material. The dataset is available in two versions based on the precision of the measurements:

- `CrystalData10k` (lower precision);
- `CrystalData300k` (higher precision);

The goal of the experiment is to estimate $F(Q)$ from noisy data using a GeDS model, and subsequently compute its Fourier transform to gain some valuable insights into the structure of the material.

Usage

```
data(CrystalData10k)

data(CrystalData300k)
```

Format

A `data.frame` with 1721 observations and 2 variables:

- `Q`: The scattering vector, measured in inverse angstroms, $\AA^{-1}$;
- `FQ`: The measured function $F(Q)$, given in arbitrary units (a.u.).

Source

Unpublished data from a controlled particle accelerator experiment.

Examples

```
## Not run:
# Load the dataset (choose 10k or 300k version)
data('CrystalData10k')

# Fit a GeDS/GeDSboost model and compare how well the intensity peaks are captured
Gmod <- NGeDS(F_Q ~ f(Q), data = CrystalData10k, phi = 0.999, q = 3)
# for CrystalData300k set int.knots_init = 1, phi = 0.999, q = 4, instead
Gmodboost <- NGeDSboost(F_Q ~ f(Q), data = CrystalData10k, phi = 0.9975, q = 4)

par(mfrow = c(1,2))
plot(Gmod, n = 2)
plot(Gmodboost, n = 2)

## End(Not run)
```

Derive *Derivative of GeDS Objects*

Description

This function computes derivatives of a fitted GeDS regression model.

Usage

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

Arguments

- `object` — An object of class "GeDS" 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

This function relies on the `splineDesign` function to compute the exact derivatives of the GeDS fit. Specifically, it leverages the well-known property that the m-th derivative of a spline (for $m = 1, 2, \ldots$) can be obtained by differentiating its B-spline basis functions. This property is detailed, e.g., in De Boor (2001, Chapter X, formula (15)).

Note that the GeDS fit is a B-spline representation of the predictor. Consequently, the derivative is computed with respect to the predictor scale and not the response scale. This implies that, in the GNM(GLM) framework, the function does not return derivatives of the conditional mean on the response scale, but rather of the underlying linear predictor scale.

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, GeDSgam, GeDSboost*

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 typically returned by NGeDS/GGeDS, NGeDSgam or NGeDSboost.

Usage

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

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

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

Arguments

object	The "GeDS", "GeDSgam" or "GeDSboost" class object from which the deviance should be extracted.
n	Integer value (2, 3 or 4) specifying the order (= degree +1) of the "GeDS", "GeDSgam" or "GeDSboost" 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). These will be ignored, but with a warning.

Details

This is a method for the function deviance in the **stats** package. As "GeDS", "GeDSgam" and "GeDSboost" 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", "GeDSgam" or "GeDSboost" fit.

See Also

deviance for the standard definition; NGeDS, GGeDS, NGeDSgam, NGeDSboost 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 one or two independent variables and a parametric component in which the remaining covariates may enter as additive terms. GAM-GeDS and FGB-GeDS models may include more than one GeD spline regression component.

The function `f` is to be used in the `formula` argument of `NGeDS`, `GGeDS`, `NGeDSgam` or `NGeDSboost` 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

`x`	Numeric vector containing N sample values of the covariate chosen to enter the spline regression component of the predictor model.
`xx`	Numeric vector containing N sample values for the second covariate (in case `NGeDS`/`GGeDS` is run for two dimensions). It has to be either `NULL` (the default) or a vector of size N, same as `x`.
`...`	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 model via NGeDS, GGeDS, NGeDSgam or NGeDSboost and not to be called in other cases by the user.

See Also

formula; NGeDS; GGeDS; NGeDSgam; NGeDSboost

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)))
```

---

family.GeDS *Extract Family from a GeDS, GeDSgam, GeDSboost Object*

Description

Method for family that returns the error distribution family used in the fitted GeDS model.

Usage

```
## S3 method for class 'GeDS'
family(object, ...)
```

```
## S3 method for class 'GeDSgam'
family(object, ...)

## S3 method for class 'GeDSboost'
family(object, ...)
```

Arguments

`object`	A "GeDS", "GeDSgam" or "GeDSboost" class object.
`...`	Further arguments (ignored).

Value

An object of class `family` describing the distribution and link function used in the GeDS fit.

See Also

`family`

---

formula.GeDS *Formula for the Predictor Model*

Description

A description of the structure of the predictor model fitted using `NGeDS`, `GGeDS`, `NGeDSgam` or `NGeDSboost`.

Usage

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

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

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

Arguments

`x`	Fitted "GeDS", "GeDSgam" or "GeDSboost" class object, produced by `NGeDS`, `GGeDS`, `NGeDSgam` or `NGeDSboost` from which the predictor model `formula` should be extracted.
`...`	Unused in this case.

Details

In GeDS GNM (GLM) regression (implemented through NGeDS and GGeDS) the mean of the response variable, correspondingly transformed through an appropriate link function, is modeled using a potentially multivariate predictor model. The latter comprises two components: a GeD variable-knot spline regression involving up to two of the independent variables and a parametric component for the remaining independent variables. The formula defines the structure of this potentially multivariate predictor.

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 mean of y to x1, the predictor has only a GeD spline component and the formula should be in the form y ~ f(x1).

As noted, there may be additional independent variables x2, x3, ... 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 ~ f(x1) + x2 + x3 which assumes a spline regression only between the transformed mean of y and x1, while x2 and x3 enter the predictor model linearly.

Both NGeDS and GGeDS functions, generate bivariate GeDS regression models. Therefore, if the functional dependence of the mean of the response variable y on x1 and x2 needs to be jointly modeled and there are no other covariates, the formula for the corresponding two dimensional predictor model should be specified as y ~ f(x1,x2).

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.

For NGeDSgam and NGeDSboost, more than one GeD spline component can be included in the formula, e.g., y ~ f(x1) + f(x2,x3) + x4, where f() denotes GeD spline-based (univariate or bivariate) regression smoothing functions/base-learners, and x4 is included as a linear term in the predictor model. Offset terms are not supported by NGeDSboost and will be ignored if included in the formula. Known additive components can instead be manually incorporated into the response variable prior to fitting the model.

GGeDS *Generalized Geometrically Designed Spline Regression Estimation*

Description

GGeDS constructs a geometrically designed (univariate or bivariate) variable knots spline regression model for the predictor in the context of generalized (non-)linear models. This is referred to as a GeDS model for a response with a 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. If the formula variables are not found in data, they are taken from environment(formula), typically the environment from which GGeDS is called.

weights
: An optional vector of "prior weights" to be put on the observations during 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 stoptype and Details below.

min.intknots
: Optional parameter allowing the user to set a minimum number of internal knots to be fit in stage A. By default equal to zero.

max.intknots
: Optional parameter allowing the user to set a maximum number of internal knots to be added by stage A's GeDS estimation algorithm. By default equal to the number of knots κ for the saturated GeDS model (i.e., $\kappa = N - 2$, 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 2L. 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 bivariate GeDS is run). See Details.

show.iters — Logical variable indicating whether or not to print information of the fit at each GeDS iteration. 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 — Logical; if TRUE, the function proceeds to Stage B, fitting higher-order models (quadratic and cubic), after completing Stage A. Default is TRUE.

Details

The GGeDS function extends the GeDS methodology, 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 that 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 (an object of class "GeDS"). 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 or two independent variables are 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. At the beginning of each GeDS iteration, a second-order spline is fitted to the data. As follows, the "working residuals" (see IRLSfit) are computed and grouped by their sign. A new knot is then placed within the cluster that maximizes a certain measure. This measure is defined as a weighted linear combination of the range of the independent variable at the cluster and the mean of the absolute residuals within it. The parameter beta determines the weights in this measure correspondingly: beta and 1 - beta. The higher beta 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: "RD", that stands for *Ratio of Deviances*; "SR", that stands for *Smoothed Ratio* of deviances; and "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 fine-tunes the stopping rule in stage A. It specifies the number of consecutive iterations over which the deviance must exhibit stable convergence to terminate knot placement in stage A. Specifically, 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 introducing the last q knots.

Value

An object of class "GeDS" (a named list) with similar components described under NGeDS's @return plus the following slots:

type Character string indicating the type of regression performed. This can be "GLM - Univ"/"GLM - Biv", respectively corresponding to generalized (GNM-GLM) univariate/bivariate GeDS (implemented by GGeDS).

guesses Initial values for the estimation of the spline coefficients at each iteration of stage A. Since the initial values are used only in the IRLS procedure, this slot applies only to objects created by GGeDS, GenUnivariateFitter or GenBivariateFitter 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, GenUnivariateFitter or GenBivariateFitter, this slot is empty if the object is not created by one of these functions.

extcall call to the GGeDS function.

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

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

See Also

NGeDS; S3 methods such as coef.GeDS, confint.GeDS, deviance.GeDS, family, formula, knots.GeDS, lines.GeDS, logLik, plot.GeDS, predict.GeDS, print.GeDS, summary.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 = 0.99, q = 2, 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(X, sapply(X, f_1), lwd = 2)
lines(Gmod, n = 3, col = "red")
lines(Gmod, n = 4, col = "blue", lty = 2)
legend("topleft",
       legend = expression(f[1](x), "Quadratic", "Cubic"),
       col = c("black", "red", "blue"),
       lty = c(1, 1, 2),
       lwd = c(2, 1, 1),
       bty = "n")

# 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 = 0.99, q = 2, 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 = 0.99, family = "quasibinomial",
              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 = quasipoisson(), 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 = 0.99, family = "poisson")

# Poisson mean deviance w.r.t data
deviance(BivGeDS, n = 2) # or sum(poisson()$dev.resids(Z, BivGeDS$Linear.Fit$Predicted, wt = 1))
deviance(BivGeDS, n = 3)
deviance(BivGeDS, n = 4)

# Poisson mean deviance w.r.t true function
f_XY <- apply(cbind(X, Y), 1, function(row) doublesin(matrix(row, ncol = 2)))
mean(poisson()$dev.resids(f_XY, BivGeDS$Linear.Fit$Predicted, wt = 1))
mean(poisson()$dev.resids(f_XY, BivGeDS$Quadratic.Fit$Predicted, wt = 1))
mean(poisson()$dev.resids(f_XY, BivGeDS$Cubic.Fit$Predicted, wt = 1))
```

```
# 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 = NULL, knots = NULL, coef = NULL, from, to, n = 3L)
```

Arguments

- `object` An object of class "GeDS" containing the GeDS fit which should be integrated. It should be the result of fitting a univariate GeDS regression via `NGeDS` or `GGeDS`. If this is provided, the `knots` and `coef` parameters will be automatically extracted from the `GeDS` object. If `object` is `NULL`, the user must provide the `knots` and `coef` vectors explicitly.
- `knots` A numeric vector of knots. This is required if `object` is `NULL`. If a `GeDS` object is provided, this parameter is ignored.
- `coef` A numeric vector of coefficients. This is required if `object` is `NULL`. If a `GeDS` object is provided, this parameter is ignored
- `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.
- `to` Numeric vector containing the upper limit(s) of integration.
- `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 relies 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 (i.e., the standard representation of a GeDS regression model) is equal to the sum of its corresponding coefficients.

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 same 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, from = 1, to = c(-1,1), 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, from = c(1,-1), to = c(-1,1), n = 3)

# Compute $\int_{-\infty}^{x} f(s)ds$
Integrate(Gmod, from = rep(-Inf, N), to = X, n = 3)
```

IRLSfit *IRLS Estimation*

Description

This function is an implementation of the IRLS estimation algorithm adjusted to the specific usage within the function `SplineReg_GLM`.

Usage

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

Arguments

x	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).
y	A vector of size N containing the observed values of the response variable y.
weights	An optional vector of prior weights for the observations, used when weighted IRLS fitting is required. By default, this is a vector of 1s.
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 the case that 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. `gaussian`) or the result of a call to a family function (e.g., `gaussian()`). 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` function 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:

coefficients A named vector containing the estimated regression coefficients.

residuals The working residuals, which are the residuals from the final iteration of the IRLS fit. Cases with zero weights are omitted, and their working residuals are `NA`.

res2 The working residuals after the final IRLS iteration. They are used within the knot placement steps of stage A of GeDS.

fitted.values The fitted mean values, obtained by transforming the predictor through the inverse of the link function.

rank The numeric rank of the fitted linear model.

family The `family` object used.

linear.predictors The fitted predictor.

deviance_iters A vector containing the deviances obtained at each IRLS iteration.

deviance The deviance at the last IRLS iteration.

null.deviance The deviance for the null model (see `glm` documentation).

iter The number of IRLS iterations performed.

weights The working weights after the last IRLS iteration.

prior.weights The "prior weights" (see the `weights` argument).

df.residual The residual degrees of freedom.

df.null The residual degrees of freedom for the null model.

y The vector of values of the response variable used in the fitting.

z The transformed responses computed after the last IRLS iteration.

converged Logical. Was the IRLS algorithm judged to have converged?

boundary 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 `glm.fit` documentation.

See Also

`glm.fit`

`knots.GeDS` *Knots Method for GeDS, GeDSgam, GeDSboost*

Description

Method for the generic function `knots` that allows the user to extract the vector of knots of a GeDS, GAM-GeDS or FGB-GeDS fit of a specified order contained in a `"GeDS"`, `"GeDSgam"` or `"GeDSboost"` class, respectively.

Usage

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

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

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

Arguments

`Fn`	The "GeDS", "GeDSgam" or "GeDSboost" class object from which the vector of knots for the specified GeDS, GAM-GeDS or FGB-GeDS fit should be extracted.
`n`	Integer value (2, 3 or 4) specifying the order (= degree +1) of the "GeDS", "GeDSgam" or "GeDSboost" 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, `NGeDSgam` and `NGeDSboost` 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/GAM-GeDS/FGB-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 *Lines Method for GeDS Objects*

Description

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

Usage

```
## S3 method for class 'GeDS'
lines(
  x,
  n = 3L,
  transform = function(x) x,
  onlySpline = TRUE,
  data = data.frame(),
  ...
)
```

Arguments

x	A "GeDS" class object, as returned by NGeDS() or GGeDS().
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. It should contain 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.
...	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)
```

`logLik.GeDS` *Extract Log-Likelihood from a GeDS Object*

Description

Method for `logLik` that returns the log-likelihood of the selected GeDS, GeDS-GAM or FGB-GeDS model.

Usage

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

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

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

Arguments

`object`	A `"GeDS"`, `"GeDSgam"` or `"GeDSboost"` class object.
`n`	Integer value (2, 3 or 4) specifying the order (= degree +1) of the `"GeDS"`, `"GeDSgam"` or `"GeDSboost"` fit whose loglikelihood should be extracted. By default equal to `3L`; non-integer values will be passed to the function `as.integer`.

`...` Additional arguments passed to `logLik`.

Value

An object of class `logLik`.

See Also

`logLik`

`N.boost.iter` *Extract Number of Boosting Iterations from a GeDSboost Object*

Description

Method for `N.boost.iter` that returns the number of boosting iterations used in fitting a `"GeDSboost"` class object.

Usage

```
## S3 method for class 'GeDSboost'
N.boost.iter(object, ...)
```

Arguments

`object`	A `"GeDSboost"` class object.
`...`	Further arguments (ignored).

Value

An integer indicating the total number of boosting iterations.

`NGeDS` *Geometrically Designed Spline Regression Estimation*

Description

`NGeDS` constructs a geometrically designed variable knot 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,
  max.intknots,
  q = 2L,
  Xextr = NULL,
  Yextr = NULL,
  show.iters = FALSE,
  stoptype = "RD",
  higher_order = TRUE,
  intknots_init = NULL,
  fit_init = NULL,
  only_pred = 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 data, the variables are taken from environment(formula), 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 NULL 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 stoptype 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., $\kappa = N - 2$).
q	Numeric parameter which allows to fine-tune the stopping rule of stage A of GeDS, by default equal to 2L. See Details.
Xextr	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.

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 bivariate GeDS is run). See Details.
show.iters	Logical variable indicating whether or not to print information about the fitting at each step.
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.
higher_order	Logical. If TRUE, after completing stage A, the function proceeds to stage B and fits higher-order models (quadratic and cubic). Default is TRUE.
intknots_init	A vector of initial internal knots for stage A. The default is NULL, in which case stage A starts with a straight-line fit (i.e., no internal knots).
fit_init	A list containing fitted values, pred, along with corresponding internal knots, intknots, and coefficients, coef, representing the initial fit from which to begin stage A GeDS iteration (i.e. departing from step 2). See Details.
only_pred	Logical. If TRUE only predictions are computed.

Details

The NGeDS function implements the GeDS methodology, 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 (an object of class "GeDS").

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 the residual cluster for which a certain measure attains its maximum. The latter measure is defined as a weighted linear combination of the range of each cluster 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. The default value of beta is 0.5.

The argument stoptype allows to choose between three alternative stopping rules for the knot selection in stage A of GeDS: "RD", that stands for *Ratio of Deviances*, "SR", that stands for *Smoothed*

Ratio of deviances and "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.

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 that 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

An object of class "GeDS" (a named list) with components:

type Character string indicating the type of regression performed. This can be "LM - Univ"/"LM - Biv" respectively corresponding to normal univariate/bivariate GeDS (implemented by NGeDS).

linear.intknots Vector containing the locations of the internal knots of the second order GeD spline fit produced at stage A.

quadratic.intknots Vector containing the locations of the internal knots of the third order GeD spline fit produced in stage B.

cubic.intknots Vector containing the locations of the internal knots of the fourth order GeD spline fit produced in stage B.

dev.linear Deviance of the second order GeD spline fit, produced in stage A.

dev.quadratic Deviance of the third order GeD spline fit, produced in stage B.

dev.cubic Deviance of the fourth order GeD spline fit, produced in stage B.

rss Vector containing the deviances of the second-order spline fits computed at each iteration of stage A of GeDS.

linear.fit List containing the results from running SplineReg function to fit the second order spline fit of stage A.

quadratic.fit List containing the results from running SplineReg function to fit the third order spline fit of stage B.

cubic.fit List containing the results from running SplineReg function to fit the fourth order spline fit of stage B.

stored Matrix containing the knot locations estimated at each iteration 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 produced in stage A.

iters Number of iterations performed during stage A of the GeDS fitting procedure.

coefficients Matrix containing the fitted coefficients of the GeD spline regression component and the parametric component at each iteration of stage A.

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. These respectively contain the values at each iteration of stage A of $\hat{\phi}_\kappa$, $\hat{\gamma}_0$ and $\hat{\gamma}_1$. See Dimitrova et al. (2023) for further details on these parameters.

formula The model `formula`.

extcall `call` to the `NGeDS` functions.

terms `terms` object containing information on the model frame.

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

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

See Also

GGeDS; S3 methods such as `coef.GeDS`, `confint.GeDS`, `deviance.GeDS`, `family`, `formula`, `knots.GeDS`, `lines.GeDS`, `logLik`, `plot.GeDS`, `predict.GeDS`, `print.GeDS`, `summary.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)
confint(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), phi = 0.9))

# Extract quadratic coefficients/knots/deviance
coef(BivGeDS, n = 3)
confint(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 implements 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. (2025)). Differing from standard component-wise boosting, this approach performs a piecewise polynomial update of the coefficients at each iteration, yielding a final fit in the form of a single spline model.

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,
  phi_boost_exit = 0.99,
  q_boost = 2L,
  beta = 0.5,
  phi = 0.99,
  int.knots_boost,
  q = 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 ~ f(X1) + f(X2) + X3).
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 FGB-GeDS 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. Note that 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 initial_learner = FALSE it also determines the corresponding empirical risk minimizer to be used as offset initial learner. By default, it is set to mboost::Gaussian(). Users can specify any of the Family objects from the **mboost** package that are listed below.

link A character string specifying the link function to be used, in case the Family object does not include the desired one. Must correspond to a valid link in family.

initial_learner A logical value. If set to TRUE, the model's initial learner will be a GeD spline. If set to FALSE, then the initial predictor will consist of the empirical risk minimizer corresponding to the specified family.

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 initial_learner = TRUE. 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 500L. This setting serves as a fallback when the stopping rule, based on consecutive deviances and tuned by phi_boost_exit and q_boost, does not trigger an earlier termination (see Dimitrova et al. (2025)). In general, adjusting the number of boosting iterations by increasing or decreasing phi_boost_exit and/or q_boost should suffice.

shrinkage Numeric parameter in the interval $(0, 1]$ defining the step size or shrinkage parameter. This helps control 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.

phi_boost_exit Numeric parameter in the interval $(0, 1)$ specifying the threshold for the boosting iterations stopping rule. Default is equal to 0.99.

q_boost Numeric parameter which allows to fine-tune the boosting iterations stopping rule. Default is equal to 2L.

beta Numeric parameter in the interval $[0, 1]$ tuning the knot placement within the GeDS base-learner at each boosting iteration. 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 the GeDS base-learner. Default is equal to 0.99. See Details in NGeDS.

int.knots_boost
: The maximum number of internal knots that each GeDS base-learner may have at each boosting iteration, effectively setting the value of max.intknots in NGeDS. By default equal to the number of knots for the saturated GeDS model (i.e., $\kappa = N - 2$). Increasing or decreasing phi and/or q should suffice to regulate the base-learner strength (i.e., the number of internal knots).

q
: Numeric parameter which allows to fine-tune the stopping rule of stage A of the GeDS base-learner. Default is 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 FGB algorithm is run. Default is TRUE.

boosting_with_memory
: Logical value. If TRUE, boosting is performed taking into account previously fitted knots when fitting a GeDS learner at each new boosting iteration. If boosting_with_memory is TRUE, we recommend setting int.knots_init = 1 and int.knots_boost = 1.

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 NGeDS/GGeDS. In the same way phi and q tune the stopping rule of NGeDSGGeDS, 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

An object of class "GeDSboost" (a named list) with components:

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 includes:

response data.frame containing the response variable observations.

predictors data.frame containing the observations corresponding to the predictor variables included in the model.

base_learners Description of the model's base learners.

family The statistical family. The possible options are:

- mboost::Binomial(type = c("adaboost", "glm"), link = c("logit", "probit", "cloglog", "cauchit", "log"), ...),
- mboost::Gaussian(),
- mboost::Poisson() and
- mboost::GammaReg(nuirange = c(0, 100)).

Other mboost families may be suitable; however, these have not yet been thoroughly tested and are therefore not recommended for use.

initial_learner If TRUE a NGeDS or GGeDS fit was used as the initial learner; otherwise, the empirical risk minimizer corresponding to the selected family was employed.

int.knots_init If initial_learner = TRUE, this corresponds to the maximum number of internal knots set in the NGeDS/GGeDS function for 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, otherwise this is NULL).

X_sd Standard deviation of the predictors (only if normalize_data = TRUE, otherwise this is NULL).

Y_mean Mean of the response variable (only if normalize_data = TRUE, otherwise this is NULL).

Y_sd Standard deviation of the response variable (only if normalize_data = TRUE, otherwise this is NULL).

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 in a `NGeDS`/`GGeDS` object (see `SplineReg` for details).

predictions A list containing the predicted values obtained for each of the fits (linear, quadratic and cubic).

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

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

Bühlmann P., Yu B. (2003). Boosting With the L2 Loss. *Journal of the American Statistical Association*, **98(462)**, 324–339. doi: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

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

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

Dimitrova, D. S., Kaishev, V. K. and Saenz Guillen, E. L. (2025). **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`; S3 methods such as `coef`, `confint`, `deviance.GeDSboost`, `family`, `formula`, `knots`, `logLik`, `predict`, `print`, `summary`. Also variable importance measures (`bl_imp`) and sequential plotting facilities (`visualize_boosting`).

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

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); summary(Gmodboost)
# Knots
knots(Gmodboost, n = 2)
knots(Gmodboost, n = 3)
knots(Gmodboost, n = 4)
# Coefficients
coef(Gmodboost, n = 2)
coef(Gmodboost, n = 3)
coef(Gmodboost, n = 4)
# Wald-type confidence intervals
confint(Gmodboost, n = 2)
confint(Gmodboost, n = 3)
confint(Gmodboost, n = 4)
# Deviances
deviance(Gmodboost, n = 2)
deviance(Gmodboost, n = 3)
deviance(Gmodboost, n = 4)

# Plot
plot(Gmodboost, n = 3)
```

```
############################ Example 2 - Bodyfat ############################
library(TH.data)
data("bodyfat", package = "TH.data")

Gmodboost <- NGeDSboost(formula = DEXfat ~ age + f(hipcirc, waistcirc) + f(kneebreadth),
data = bodyfat, phi_boost_exit = 0.9, q_boost = 1, phi = 0.9, q = 1)

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 linear GeD splines (i.e., `NGeDS` function) to fit the targets within each backfitting iteration. Higher order fits are computed by pursuing stage B of GeDS after the local-scoring algorithm is run.

Usage

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

Arguments

`formula` A description of the model structure to be fitted, specifying both the dependent and independent variables. Unlike `NGeDS` and `GGeDS`, this formula supports multiple additive (normal) GeD spline regression components as well as linear components. For example, setting `formula = Y ~ f(X1) + f(X2) + X3` implies using a normal linear GeD spline as the smoother for `X1` and for `X2`, while for `X3` a linear model would be used.

`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.

`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 local-scoring iterations to be run. If not specified, it defaults to 0L.

`max_iterations` — Optional parameter to manually set the maximum number of local-scoring 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 `phi_gam_exit` and `q_gam`, does not trigger an earlier termination (see Dimitrova et al. (2025)). Therefore, users can increase/decrease the number of local-scoring iterations, by increasing/decreasing the value `phi_gam_exit` and/or `q_gam`, or directly specify `max_iterations`.

`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 `0.99`.

`q_gam` — Numeric parameter which allows to fine-tune the stopping rule of the local-scoring and backfitting iterations. By default equal to `2L`.

`beta` — Numeric parameter in the interval $[0, 1]$ tuning the knot placement in stage A of GeDS, for each of the GeD spline components of the model. 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, for each of the GeD spline components of the model. 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 smoothers at each backfitting iteration, effectively setting the value of `max.intknots` 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, for each of the GeD spline components of the model. 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 function 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 NGeDS/GGeDS. In the same way phi and q tune the stopping rule of NGeDS/GGeDS, phi_gam_exit and q_gam tune the stopping rule of NGeDSgam. The user can also manually control the number of local-scoring iterations through min_iterations and max_iterations.

A model term wrapped in offset() is treated as a known (fixed) component and added directly to the linear predictor when fitting the model. 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. See formula.

Value

An object of class "GeDSgam" (a named list) with components:

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 the response variable observations.
- predictors data.frame containing the corresponding observations of the 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") and
  - quasipoisson(link = "log", "identity", "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, otherwise this is NULL).

Y_mean Mean of the response variable (only if normalize_data = TRUE, otherwise this is NULL).

Y_sd Standard deviation of the response variable (only if normalize_data = TRUE, otherwise this is NULL).

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, when family = "gaussian", the algorithm is tantamount to directly implementing backfitting.

dev Deviance of the final model. For family = "gaussian" this coincides with the Residual Sum of Squares.

Y_hat Fitted values, including: - eta: the additive predictor, - mu: the vector of means, - z: the adjusted dependent variable.

base_learners A list containing, for each base-learner, the corresponding linear fit piece-wise 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 linear fit in B-spline form (see SplineReg).

quadratic.fit Quadratic fit obtained via Schoenberg variation diminishing approximation (see SplineReg).

cubic.fit Cubic fit obtained via via Schoenberg variation diminishing approximation (see SplineReg).

predictions A list containing the predicted values obtained for each of the fits (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

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

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

Dimitrova, D. S., Kaishev, V. K. and Saenz Guillen, E. L. (2025). **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; S3 methods such as coef, confint, deviance.GeDSgam, family, formula, knots, logLik, predict, print, summary.

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 = 0.8)
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 'GeDSgam'
# Print
print(Gmodgam); summary(Gmodgam)
# Knots
knots(Gmodgam, n = 2)
knots(Gmodgam, n = 3)
knots(Gmodgam, n = 4)
# Coefficients
coef(Gmodgam, n = 2)
coef(Gmodgam, n = 3)
coef(Gmodgam, n = 4)
# Wald-type confidence intervals
confint(Gmodgam, n = 2)
confint(Gmodgam, n = 3)
confint(Gmodgam, n = 4)
# Deviances
deviance(Gmodgam, n = 2)
deviance(Gmodgam, n = 3)
deviance(Gmodgam, n = 4)
```

---

plot.GeDS *Plot Method for GeDS Objects*

Description

Plot method for GeDS objects. Plots GeDS fits.

Usage

```
## S3 method for class 'GeDS'
plot(
  x,
  f = NULL,
  which,
  dev = FALSE,
  ask = FALSE,
  main,
  legend.pos = "topright",
  legend.text = NULL,
  new.window = FALSE,
  wait = 0.5,
  n = 3L,
  type = c("none", "polygon", "nci", "aci"),
  ...
)
```

Arguments

`x`	An object of class "GeDS", as returned by `NGeDS()` or `GGeDS()`.
`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`	An optional character string used as the plot title. If set to "detail", the knots vector will be displayed on the plot.
`legend.pos`	The position of the legend within the panel. See legend for details.
`legend.text`	A character vector specifying the legend text.
`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, main = "detail")
plot(Gmod, type = "nci")
plot(Gmod, type = "aci")

# 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:(Gmod$Nintknots + Gmod$args$q + 1))

## 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:
iters <- (Gmod2$Nintknots + Gmod2$args$q + 1)
plot(Gmod2, which = 1:iters, n = 2)
plot(Gmod2, which = 1:iters, n = 2, ask = T)

## End(Not run)
```

---

plot.GeDSboost *Plot Method for GeDSboost Objects*

Description

Plots the component functions of a GeDSboost object fitted using NGeDSboost. If the model has a single base-learner, the plot will be returned on the response scale. Otherwise, plots are produced on the linear predictor scale. Note that only univariate base-learner plots are returned, as the representation of the boosted model as a single spline model is available only for univariate base-learners (see Dimitrova et al. (2025)). Additionally, since component-wise gradient boosting inherently performs base-learner selection, plots will only be generated for the base-learners selected during the boosting iterations.

Usage

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

Arguments

x	A GeDSboost object, as returned by `NGeDSboost()`.
n	Integer value (2, 3 or 4) specifying the order (= degree +1) of the FGB-GeDS fit to be extracted.
...	Further arguments to be passed to the `plot.default` function.

References

Dimitrova, D. S., Kaishev, V. K. and Saenz Guillen, E. L. (2025). **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

NGeDSboost

Examples

```
data(mtcars)
# Convert specified variables to factors
categorical_vars <- c("cyl", "vs", "am", "gear", "carb")
mtcars[categorical_vars] <- lapply(mtcars[categorical_vars], factor)
N <- nrow(mtcars); ratio <- 0.8
set.seed(123)
trainIndex <- sample(1:N, size = floor(ratio * N))
# Subset the data into training and test sets
train <- mtcars[trainIndex, ]
test <- mtcars[-trainIndex, ]
Gmodboost <- NGeDSboost(mpg ~ cyl + f(drat) + f(wt) + f(hp) + vs + am,
                        data = train, phi = 0.7, shrinkage = 0.9, initial_learner = FALSE)

par(mfrow = c(2,3))
plot(Gmodboost, n = 2)
```

plot.GeDSgam *Plot Method for GeDSgam Objects*

Description

Plots on the linear predictor scale the component functions of a GeDSgam object fitted using NGeDSgam.

Usage

```
## S3 method for class 'GeDSgam'
plot(x, base_learners = NULL, f = NULL, n = 3L, ...)
```

Arguments

x	A GeDSgam object, as returned by NGeDSgam().
base_learners	Either NULL or a vector of character string specifying the base-learners of the model for which predictions should be plotted. Note that single base-learner predictions are provided on the linear predictor scale.
f	(Optional) specifies the underlying component 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.
n	Integer value (2, 3 or 4) specifying the order (= degree +1) of the GAM-GeDS fit.
...	Further arguments to be passed to the plot.default function.

See Also

NGeDSgam

Examples

```
## Gu and Wahba 4 univariate term example ##
# Generate a data sample for the response variable
# y and the covariates x0, x1 and x2; include a noise predictor x3
set.seed(123)
N <- 400
f_x0x1x2 <- function(x0,x1,x2) {
  f0 <- function(x0) 2 * sin(pi * x0)
  f1 <- function(x1) exp(2 * x1)
  f2 <- function(x2) 0.2 * x2^11 * (10 * (1 - x2))^6 + 10 * (10 * x2)^3 * (1 - x2)^10
  f <- f0(x0) + f1(x1) + f2(x2)
  return(f)
}
x0 <- runif(N, 0, 1)
x1 <- runif(N, 0, 1)
x2 <- runif(N, 0, 1)
```

```
x3 <- runif(N, 0, 1)
# Specify a model for the mean of y
f <- f_x0x1x2(x0 = x0, x1 = x1, x2 = x2)
# Add (Normal) noise to the mean of y
y <- rnorm(N, mean = f, sd = 0.2)
data <- data.frame(y = y, x0 = x0, x1 = x1, x2 = x2, x3 = x3)

# Fit GAM-GeDS model
Gmodgam <- NGeDSgam(y ~ f(x0) + f(x1) + f(x2) + f(x3), data = data)

f0 <- function(x0) 2 * sin(pi * x0)
f1 <- function(x1) exp(2 * x1)
f2 <- function(x2) 0.2 * x2^11 * (10 * (1 - x2))^6 + 10 * (10 * x2)^3 * (1 - x2)^10
fs <- list(f0, f1, f2)
main_f0 <- expression(
  f[0](x[0]) == 2 * sin(pi * x[0])
  )
main_f1 <- expression(
  f[1](x[1]) == e^(2 * x[1])
  )
main_f2 <- expression(
  f[2](x[2]) == 0.2 * x[2]^11 * (10 * (1 - x[2]))^6 +
    10 * (10 * x[2])^3 * (1 - x[2])^10
  )
mains <- list(main_f0, main_f1, main_f2)

# Create and display the plot
par(mfrow = c(1,3), mar = c(5.1, 5.1, 4.1, 2.1))
for (i in 1:3) {
  # Plot the base learner
  plot(Gmodgam, n = 3, base_learners = paste0("f(x", i-1, ")"), f = fs[[i]],
  main = mains[[i]], col = "seagreen",
  cex.lab = 2, cex.axis = 2, cex.main = 1.5)
# Add legend
if (i == 2) {
  position <- "topleft"
} else if (i == 3) {
  position <- "topright"
} else {
  position <- "bottom"
}
  legend(position, legend = c("NGeDSgam quad.", paste0("f(x", i-1, ")")),
         col = c("seagreen", "darkgray"),
         lwd = c(2, 2),
         bty = "n",
         cex = 1.5)
}
```

PPolyRep *Piecewise Polynomial Spline Representation*

Description

This function converts a univariate GeDS fit from its B-spline representation to a piecewise polynomial form.

Usage

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

Arguments

object
: The "GeDS" class object of type "LM - Univ" or "GLM - Univ" from which the GeDS fit to be converted should be extracted.

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—stored as an object of class "GeDS" and represented using B-splines—into an equivalent representation using piecewise polynomials.

It wraps the function `polySpline`, enabling it to handle "GeDS" objects as input. This provides a convenient bridge between the **GeDS** and **splines** packages, allowing users to leverage the functionality available in **splines**.

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.pos = 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 data.frame, list 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 x itself.
type	Character string specifying the type of prediction to return. The default is "response", which gives predictions on the scale of the response variable. See Details for other available options.
n	Integer value (2, 3 or 4) specifying the order (= degree +1) of the "GeDS", "GeDSgam" or "GeDSboost" 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 in the **stats** package, 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.GeDSgam,boost *Predict Method for GeDSgam, GeDSboost*

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 'GeDSgam'
predict(
  object,
  newdata,
  n = 3L,
  base_learner = NULL,
  type = c("response", "link"),
  ...
)

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

Arguments

object	The "GeDSgam" class or "GeDSboost" class object.
newdata	An optional data.frame containing values of the independent variables at which predictions are to be computed. If omitted, the fitted values are extracted from the object itself.
n	The order of the GeDS fit (2L for linear, 3L for quadratic, and 4L for cubic). Alternatively, "all" can be used to return a list with the predictions from all three fits. Default is 3L.
base_learner	Either NULL or a character string specifying the base-learner of the model for which predictions should be computed. Note that single base-learner predictions are provided on the linear predictor scale.
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. Alternatively if one wants the predictions to be on the predictor scale, it is necessary to set type = "link".
...	Potentially further arguments.

Value

Numeric vector (or list) of predictions.

References

Gu, C. and Wahba, G. (1991). Minimizing GCV/GML Scores with Multiple Smoothing Parameters via the Newton Method. *SIAM J. Sci. Comput.*, **12**, 383–398.

Examples

```
## Gu and Wahba 4 univariate term example ##
# Generate a data sample for the response variable
# y and the covariates x0, x1 and x2; include a noise predictor x3
set.seed(123)
N <- 400
f_x0x1x2 <- function(x0,x1,x2) {
  f0 <- function(x0) 2 * sin(pi * x0)
  f1 <- function(x1) exp(2 * x1)
  f2 <- function(x2) 0.2 * x2^11 * (10 * (1 - x2))^6 + 10 * (10 * x2)^3 * (1 - x2)^10
  f <- f0(x0) + f1(x1) + f2(x2)
  return(f)
}
x0 <- runif(N, 0, 1)
x1 <- runif(N, 0, 1)
x2 <- runif(N, 0, 1)
x3 <- runif(N, 0, 1)
# Specify a model for the mean of y
f <- f_x0x1x2(x0 = x0, x1 = x1, x2 = x2)
# Add (Normal) noise to the mean of y
y <- rnorm(N, mean = f, sd = 0.2)
data <- data.frame(y = y, x0 = x0, x1 = x1, x2 = x2, x3 = x3)

# Fit GAM-GeDS model
Gmodgam <- NGeDSgam(y ~ f(x0) + f(x1) + f(x2) + f(x3), data = data)
# Check that the sum of the individual base-learner predictions equals the final
# model prediction

pred0 <- predict(Gmodgam, n = 2, newdata = data, base_learner = "f(x0)")
pred1 <- predict(Gmodgam, n = 2, newdata = data, base_learner = "f(x2)")
pred2 <- predict(Gmodgam, n = 2, newdata = data, base_learner = "f(x1)")
pred3 <- predict(Gmodgam, n = 2, newdata = data, base_learner = "f(x3)")
round(predict(Gmodgam, n = 2, newdata = data) -
(mean(predict(Gmodgam, n = 2, newdata = data)) + pred0 + pred1 + pred2 + pred3), 12)

pred0 <- predict(Gmodgam, n = 3, newdata = data, base_learner = "f(x0)")
pred1 <- predict(Gmodgam, n = 3, newdata = data, base_learner = "f(x2)")
pred2 <- predict(Gmodgam, n = 3, newdata = data, base_learner = "f(x1)")
pred3 <- predict(Gmodgam, n = 3, newdata = data, base_learner = "f(x3)")

round(predict(Gmodgam, n = 3, newdata = data) - (pred0 + pred1 + pred2 + pred3), 12)

pred0 <- predict(Gmodgam, n = 4, newdata = data, base_learner = "f(x0)")
pred1 <- predict(Gmodgam, n = 4, newdata = data, base_learner = "f(x2)")
pred2 <- predict(Gmodgam, n = 4, newdata = data, base_learner = "f(x1)")
pred3 <- predict(Gmodgam, n = 4, newdata = data, base_learner = "f(x3)")

round(predict(Gmodgam, n = 4, newdata = data) - (pred0 + pred1 + pred2 + pred3), 12)

# Plot GeDSgam partial fits to f(x0), f(x1), f(x2)
par(mfrow = c(1,3))
for (i in 1:3) {
```

```
  # Plot the base learner
  plot(Gmodgam, n = 3, base_learners = paste0("f(x", i-1, ")"), col = "seagreen",
       cex.lab = 1.5, cex.axis = 1.5)
  # Add legend
  if (i == 2) {
    position <- "topleft"
    } else if (i == 3) {
      position <- "topright"
      } else {
        position <- "bottom"
      }
  legend(position, legend = c("GAM-GeDS Quadratic", paste0("f(x", i-1, ")")),
         col = c("seagreen", "darkgray"),
         lwd = c(2, 2),
         bty = "n",
         cex = 1.5)
}
```

`print.GeDS` *Print Method for GeDS, GeDSgam, GeDSboost*

Description

Method for the generic function `print` that allows to print on screen the main information related to a fitted `"GeDS"`, `"GeDSgam"`, `"GeDSboost"` class model.

Usage

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

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

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

Arguments

`x`	The `"GeDS"`, `"GeDSgam"` or `"GeDSboost"` 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 `"GeDS"`, `"GeDSgam"` or `"GeDSboost"` object.

Value

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

Nintknots 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` *Estimation for Models with Spline and Parametric Components*

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, estimation.

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_pred = 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 `offset` is particularly useful in `Splinereg_GLM` 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 fitted in stage A's iterations.
`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 to the range 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_pred`	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 `length(InterKnots) + n + 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 order, vector of knots of the spline and distribution of the response variable (from the exponential family). The functions `SplineReg_LM` and `SplineReg_GLM` are based on LS and IRLS, respectively, and are used in `NGeDS` and `GGeDS` to estimate the coefficients of the final GeDS fits in stage B. These fits use knots positioned at the Greville abscissae 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). If the variance matrix is singular the Moore-Penrose pseudo-inverse is computed instead.

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 `SplineReg_LM` is called or the residuals described in Details if `SplineReg_GLM` is called.

rss The deviance for the fitted predictor model, defined as in Dimitrova et al. (2023), which for `SplineReg_LM` coincides with the residual sum of squares.

basis The matrix of B-spline regression functions and the covariates of the parametric part evaluated at the sample values of the covariate(s).

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.

aci A list containing the lower (`Low`) and upper (`Upp`) limits of the asymptotic confidence intervals computed 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).

temporary The result of the function `lm` if `SplineReg_LM` is used, or the output of the function `glm` if `SplineReg_GLM` is used.

References

Kaishev, V. K., Dimitrova, D. S., Haberman, S. & Verrall, R. J. (2006). Geometrically designed, variable know 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

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

See Also

`NGeDS`, `GGeDS`, `Fitters`, `IRLSfit`, `lm` and `glm.fit`.

`summary.GeDS` *Summary Method for GeDS, GeDSgam, GeDSboost*

Description

Method for the generic function `summary` that allows you to print on screen the main information related to a fitted `"GeDS"`, `"GeDSgam"` or `"GeDSboost"` model. Similar to `print.GeDS` but with some extra detail.

Usage

```
## S3 method for class 'GeDS'
summary(object, ...)

## S3 method for class 'GeDSgam'
summary(object, ...)

## S3 method for class 'GeDSboost'
summary(object, ...)
```

Arguments

`object`	The "GeDS", "GeDSgam" or "GeDSboost" object for which the main information should be printed on screen.
`...`	Potentially further arguments (required by the definition of the generic function).

See Also

print.GeDS

UnivariateFitters — *Functions Used to Fit GeDS Objects with a Univariate Spline Regression*

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_init = NULL,
  fit_init = NULL,
  only_pred = 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 GenUnivariateFitter 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 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., $\kappa = N - 2$, 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 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_init 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.

fit_init A list containing fitted values pred, along with corresponding intknots and coef, representing the initial fit from which to begin Stage A GeDS iteration (i.e. departing from step 2).

only_pred 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. gaussian) or the result of a call to a family function (e.g. gaussian()). 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 $|\delta^* - \delta_i| <$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

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

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 negative gradient. Note that this is only applicable to NGeDSboost models with a single univariate base-learner.

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