Package 'brokenstick'

Type  Package
Title  Broken Stick Model for Irregular Longitudinal Data
Version  2.0.0
Description  The broken stick model describes a set of individual curves by a linear mixed model using a second-order linear B-spline. The main use of the model is to align irregularly observed data to a user-specified grid of break ages. All fitting can done in the Z-score scale, so non-linearity and irregular data can be treated as separate problems. This package contains functions for fitting a broken stick model to data, for predicting broken stick curves in new data, and for plotting the broken stick estimates. For additional documentation on background, methodology and applications see <https://stefvanbuuren.name/publications/2021_brokenstick_JSS_manuscript.pdf>.
Depends  R (>= 3.5.0)
Imports  coda, dplyr, lme4, matrixsampling, methods, rlang, splines, stats, tidyR
Suggests  AGD, ggplot2, grDevices, gridExtra, knitr, lattice, MASS, Matrix, mice, mvtnorm, plyr, svglite, testthat, rmarkdown
BugReports  https://github.com/growthcharts/brokenstick/issues
Encoding  UTF-8
License  MIT + file LICENSE
LazyData  TRUE
VignetteBuilder  knitr
RoxygenNote  7.1.2
NeedsCompilation  no
Author  Stef van Buuren [aut, cre]
Maintainer  Stef van Buuren <stef.vanbuuren@tno.nl>
Repository  CRAN
Date/Publication  2021-11-11 16:30:08 UTC
**Description**

The `brokenstick()` function fits an irregularly observed series of measurements onto a user-specified grid of points (knots). The model codes the grid by a series of linear B-splines. Each modelled trajectory consists of straight lines that join at the chosen knots and look like a broken stick. Differences between observations are expressed by a random effect per knot.

**Usage**

```r
brokenstick(
  formula,
  data,
  knots = NULL,
  boundary = NULL,
  k = NULL,
  degree = 1L,
  method = c("kr", "lmer"),
  control = set_control(method = method, ...),
  na.action = na.exclude,
```
brokenstick

```r
light = FALSE,
...  
)
```

**Arguments**

- `formula`: A formula specifying the outcome, the predictor and the group variable in `data`. The generic shape is `formula = y ~ x | group`. The left-hand side is the outcome, the right-hand side the predictor, and the name of the grouping variable occurs after the `|` sign. Formula treatment is non-standard: 1) `y` and `x` should be numeric, 2) only one variable is allowed in each model term (additional variables will be ignored).

- `data`: A data frame or matrix containing the outcome (numeric), predictor (numeric) and group (numeric, factor, character) variable.

- `knots`: Optional, but recommended. Numerical vector with the locations of the internal knots to be placed on the values of the predictor.

- `boundary`: Optional, but recommended. Numerical vector of length 2 with the left and right boundary knot. The boundary setting is passed to `splines::bs()` as the `Boundary.knots` argument. If not specified, the function determines the boundary knots as `range(x)`. When specified, the boundary range is internally expanded to include at least `range(knots)`.

- `k`: Optional, a convenience parameter for the number of internal knots. If specified, then `k` internal knots are placed at equidense quantiles of the predictor. For example, specifying `k = 1` puts a knot at the 50th quantile (median), setting `k = 3` puts knots at the 25th, 50th and 75th quantiles, and so on. If the user specifies both `k` and `knots` arguments then `knots` takes precedence.

- `degree`: the degree of the spline. The broken stick model requires linear splines, so the default is `degree = 1`. Setting `degree = 0` yields (crisp) dummy coding, and one column less than for `degree = 1`. The brokenstick package supports only `degree = 0` and `degree = 1`.

- `method`: Estimation method. Either "kr" (for the Kasim-Raudenbush sampler) or "lmer" (for `lme4::lmer()`). Version 1.1.1.9000 changed the default to `method = "kr"`.

- `control`: List of control options returned by `set_control()` used to set algorithmic details. A list with parameters. When not specified, the functions sets to defaults for method "kr" by `control_kr()`, and for method "lmer" by `lme4::lmerControl()`. For ease of use, the user may set individual options to "kr" (e.g. `niter = 500`) via the ...arguments.

- `na.action`: A function that indicates what `lme4::lmer()` should do when the data contain NAs. Default set to `na.exclude`. Only used by method "lmer".

- `light`: Should the returned object be lighter? If `light = TRUE` the returned object will contain only the model settings and parameter estimates and not store the data, `imp` and `mod` elements. The light object can be used to predict broken stick estimates for new data, but does not disclose the training data and is very small (often <20 Kb).

... Forwards arguments to `control_kr()`.
Details

The choice between method = "kr" and method = "lmer" depends on the size of the data and the complexity of the model. In general, setting method = "lmer" can require substantial calculation time for more complex models (say > 8 internal knots) and may not converge. Method "kr" is less sensitive to model complexity and small samples, and has the added benefit that the variance-covariance matrix of the random effects can be constrained through the cormodel argument. On the other hand, "lmer" is the better-researched method, and is more efficient for simpler models and datasets with many rows.

The default algorithm since version 2.0 is the Bayesian Kasim-Raudenbush sampler (method = "kr"). The variance-covariance matrix of the broken stick estimates absorbs the relations over time. The "kr" method allows enforcing a simple structure on this variance-covariance matrix. Currently, there are three such correlation models: "none" (default), "argyle" and "cole". Specify the seed argument for reproducibility. See control_kr() for more details.

The alternative method = "lmer" fits the broken stick model by lme4::lmer(). With this method, the variance-covariance matrix can only be unstructured. This estimate may be unstable if the number of children is small relative to the number of specified knots. The default setting in lme4::lmerControl() is check.nobs.vs.nRE= "stop". The [set_control()] function changes this to check.nobs.vs.nRE= "warning" by default, since otherwise many broken stick models would not run at all. The method throws warnings that estimates are not stable. It can be time for models with many internal knots. Despite the warnings, the results often look reasonable.

Diagnostics with codas and lme4: The function returns an object of class brokenstick. For method = "kr" the list component named "mod" contains a list of mcmc objects that can be further analysed with coda::acfplot(), coda::autocorr(), coda::crosscorr(), coda::cumplot(), coda::densplot(), coda::effectiveSize(), coda::geweke.plot(), coda::raftery.diag(), coda::traceplot() and the usual plot() and summary() functions. For method = "lmer" the list component named "mod" contains an object of class lme4::merMod. These model objects are omitted in light brokenstick objects.

Value

A object of class brokenstick.

Note

Note that automatic knot specification is data-dependent, and may not reproduce on other data. Likewise, knots specified via k are data-dependent and do not transfer to other data sets. Fixing the model requires specifying both knots and boundary.

Examples

data <- smocc_200[1:1198, ]

# using kr method, default
fl <- brokenstick(hgt_z ~ age | id, data, knots = 0:3, seed = 123)
plot(fl, data, n_plot = 9)

# study sampling behaviour of the sigma2 parameter with coda
library(coda)
plot(f1$mod$sigma2)
acfplot(f1$mod$sigma2)

# using lmer method
f2 <- brokenstick(hgt_z ~ age | id, data, knots = 0:3, method = "lmer")
plot(f2, data, n_plot = 9)

# drill down into merMod object with standard diagnostics in lme4
library(lme4)
summary(f2$mod)
plot(f2$mod)

# a model with more knots
knots <- round(c(0, 1, 2, 3, 6, 9, 12, 15, 18, 24, 36) / 12, 4)

# method kr takes about 2 seconds
f3 <- brokenstick(hgt_z ~ age | id, data, knots, seed = 222)
plot(f3, data, n_plot = 9)

# method lmer takes about 40 seconds
f4 <- brokenstick(hgt_z ~ age | id, data, knots, method = "lmer")
plot(f4, data, n_plot = 9)

---

### brokenstick-class

**Class** brokenstick

**Description**

The main fitting function `brokenstick()` returns an object of class `brokenstick`. This object collects the fitted broken stick model.

**Details**

The package exports S3 methods for the brokenstick class for the following generic functions: `coef()`, `fitted()`, `model.frame()`, `model.matrix()`, `plot()`, `predict()`, `print()`, `residuals()` and `summary()`.

The package exports the following helper functions for brokenstick objects: `get_knots()`, `get_omega()` and `get_r2()`.

A brokenstick object is a list with the following named elements:

**Elements**

- **call**: Call that created the object
- **formula**: A formula with the model specification, e.g. `formula(y ~ x | group)`
- **names**: A named list with three elements ("x", "y", "g") providing the variable name for time, outcome and subject, respectively.
- **internal**: Numeric vector of with internal knots.
boundary  Numeric vector of length 2 with the boundary knots.
derg  The degree of the B-spline.  See splines::bs().  Support only the values of 0 (step model) or 1 (broken stick model).
method  String, either "kr" or "lmer", identifying the fitting model.
control  List of control options returned by set_control() used to set algorithmic details.
beta  Numeric vector with fixed effect estimates.
omega  Numeric matrix with variance-covariance estimates of the broken stick estimates.
sigma2j  Numeric vector with estimates of the residual variance per group. Only used by method "kr".
sigma2  Numeric vector with the mean residual variance.
sample  A numeric vector with descriptives of the training data.
light  Should the returned object be lighter? If light = TRUE the returned object will contain only the model settings and parameter estimates and not store the sample, data, imp and mod elements. The light object can be used to predict broken stick estimates for new data, but does not disclose the training data and is small.
data  The training data used to fit the model.
imp  The imputations generated for the missing outcome data. Only for method = "kr".
mod  For method = "kr": A named list with four components, each of class coda::mcmc. For method = "lmer": An object of class lme4::merMod.

Author(s)
Stef van Buuren, 2021

References

brokenstick-pkg  brokenstick: A package for irregular longitudinal data.

Description
The broken stick model describes a set of individual curves by a linear mixed model using second-order linear B-splines. The main use of the model is to align irregularly observed data to a user-specified grid of break ages.

Details
The brokenstick package contains functions for fitting a broken stick model to data, for predicting broken stick curves for new data, and for plotting the results.

brokenstick functions
The main functions are:
The following functions are user-oriented helpers:

- `coef()` Extract estimated parameters
- `fitted()` Calculate fitted values
- `get_knots()` Obtain the knots from a broken stick model
- `get_omega()` Extract variance-covariance of random effects
- `get_r2()` Obtain proportion of explained variance
- `model.frame()` Extract model frame
- `model.matrix()` Extract design matrix
- `residuals()` Extract residuals from broken stick model

The following functions perform calculations:

- `set_control()` Set controls to steer calculations
- `control_kr()` Set controls for the `kr` method

**Note**

Development of this package was kindly supported under the Healthy Birth, Growth and Development knowledge integration (HBGDki) program of the Bill & Melinda Gates Foundation.

**References**


**See Also**

- `brokenstick`, `EB`, `predict.brokenstick`

---

**control_kr**

Set controls for Kasim-Raudenbush sampler

**Description**

Set controls for Kasim-Raudenbush sampler
Usage

control_kr(
    niter = 200L,
    nimp = 0L,
    start = 101L,
    thin = 1L,
    seed = NA_integer_,
    cormodel = c("none", "argyle", "cole"),
    ...
)

Arguments

niter  Integer. Number of samples from posterior. Default: 200.
nimp  Integer. Number of multiple imputations. Default: 0.
start Integer. The iteration number of the first observation
thin  Integer. The thinning interval between consecutive observations
seed  Integer. Seed number for base::set.seed(). Use NA to bypass seed setting.
cormodel String indicating the correlation model: "none" (default), "argyle" or "cole"
...
 Allow for dot parameters

Value

A list with eight components. The function calculates parameters end (the iteration number of the last iteration) and thin_imp (thinning factor for multiple imputations) from the other inputs.

EB

Empirical Bayes predictor for random effects

Description

This function can estimate random effect for a given set of model estimates and new user data. The unit may be new to the model. The methods implements the EB estimate (also known as BLUP) as described in Skrondral and Rabe-Hasketh, 2009, p. 683. This function can also provide the broken stick estimate for a given level, the sum of the global (fixed) and individual (random) effects. The current implementation does not provide prediction errors.

Usage

EB(model, y, X, Z = X, BS = TRUE)
Arguments

model     An object of class brokenstick.
y         A vector of new measurements for unit j, scaled in the same metric as the fitted model.
X         A nj * p matrix with fixed effects for unit j, typically produced by \texttt{bs()}.
Z         A nj * q matrix with random effects for unit j. The default sets Z equal to X.
BS        A logical indicating whether broken stick estimates should be returned (BS = TRUE) or the random effects (BS = FALSE). The default is TRUE.

Value

A vector of length q containing the random effect or broken stick estimates for unit j.

Author(s)

Stef van Buuren, 2015/2020

References


Description

Calculate fitted values

Usage

## S3 method for class 'brokenstick'
fitted(object, newdata = NULL, ...)

Arguments

object     A brokenstick object.
newdata   Optional. A data frame in which to look for variables with which to predict. The training data are used if omitted and if \texttt{object$light} is FALSE.
...       Additional arguments. Ignored.

Value

See \texttt{predict.brokenstick()}.

See Also

Other brokenstick: \texttt{residuals.brokenstick()}
**Description**

Object `fit_200` has class `brokenstick` and contains the fitted broken stick model, including the training data and diagnostics.

**Format**

An object of class `brokenstick`, fitted by the `brokenstick()`.

**Details**

The dataset was constructed as

```r
knots <- round(c(0, 1, 2, 3, 6, 9, 12, 15, 18, 24)/12, 4)
fit_200 <- brokenstick(hgt_z ~ age | id, data = smocc_200,
                       knots = knots, boundary = c(0, 3), seed = 1)
```

**Description**

Object `fit_200_light` has class `brokenstick` and stores the model settings and parameter estimates.

**Format**

An object of class `brokenstick`, fitted by the `brokenstick()`.

**Details**

The datasets was constructed as

```r
knots <- round(c(0, 1, 2, 3, 6, 9, 12, 15, 18, 24)/12, 4)
fit_200_light <- brokenstick(hgt_z ~ age | id, data = smocc_200,
                              knots = knots, boundary = c(0, 3),
                              light = TRUE, seed = 1)
```
get_knots

Obtain the knots from a broken stick model

Description

Obtain the knots from a broken stick model

Usage

get_knots(
  object,
  what = c("all", "internal", "boundary", "dropfirst", "droplast")
)

Arguments

- **object**: An object of class brokenstick
- **what**: A character vector of length 1. Valid values are "all", "internal", "boundary", "dropfirst" or "droplast". The default is what = "all".

Value

A vector with knot locations, either both, internal only or boundary only. The result is NULL if object does not have proper class. Returns numeric(0) if there are no internal knots.

Examples

get_knots(fit_200, "internal")

get_omega

Extract Variance and Correlation Components

Description

Extracts variance-covariance or correlation matrix from a brokenstick object.

Usage

get_omega(x, what = c("cov", "cor"), names = NULL)

Arguments

- **x**: Object of class brokenstick
- **what**: Either "cov" (default) for the covariance matrix, or "cor" for the correlation matrix.
- **names**: A vector of column names of. If not specified, the function automatically drops the entries corresponding to the right boundary. Specify names = "all" to prevent dropping.
Value
A numeric matrix, possibly with zero rows and columns if no names match

Examples
f1 <- brokenstick(hgt_z ~ age | id, smocc_200[1:1000, ], knots = 0:3, seed = 1)
get_omega(f1, "cor", c("age_1", "age_2"))

get_r2

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obtain proportion of explained variance from a broken stick model</td>
</tr>
</tbody>
</table>

Usage
get_r2(object, newdata = NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>An object of class brokenstick</td>
</tr>
<tr>
<td>newdata</td>
<td>Data on which r.squared must be calculated</td>
</tr>
</tbody>
</table>

Value
Proportion of explained variance

Examples
get_r2(fit_200)

kr

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kasim-Raudenbush sampler for two-level normal model</td>
</tr>
</tbody>
</table>

Usage
kr(y, x, g, control = control_kr())
kr

Arguments

y Vector with outcome value
x Matrix with predictor value
g Vector with group values
control A list created by control.kr() that sets algorithmic options of the sampler and correlation model.

Details

The speed of the Kasim-Raudenbush sampler is almost independent of the number of random effect, and foremost depends on the total number of iterations.

The defaults start = 100, n = 200 and thin = 1 provide 200 parameter draws with a reasonable approximation to the variance-covariance matrix of the random effects.

For a closer approximations with 200 draws set control = control.kr(thin = 10) (better) or thin = 20 (best), at the expense of a linear increase in calculation time. Drawing fewer than 50 observations is not recommended, and such results are best treated as indicative.

It is possible to draw multiple imputations by setting the nimp parameter. For example, to draw five imputations for each missing outcome specify control = control.kr(nimp = 5).

Value

An object of class kr, basically a list with components:

* `beta` Fixed effects
* `omega` Variance-covariance of random effects
* `sigma2.j` Residual variance per group
* `sigma2` Average residual variance
* `sample` Descriptive statistics about the data
* `imp` Numeric matrix with `nimp` multiple imputations.
* `mod` A list of objects of class [coda::mcmc()]

The number of rows in imp is equal to the number of missing values in the outcome vector y. The number of columns equals nimp.

Author(s)

Stef van Buuren, based on mice::mice.impute.2l.norm()

References

**make_basis**

*Create linear splines basis*

**Description**

This function creates the basis function of a second-order (linear) splines at a user-specific set of break points.

**Usage**

```r
make_basis(
  x,
  xname = "x",
  internal = NULL,
  boundary = range(x),
  degree = 1L,
  warn = TRUE
)
```

**Arguments**

- **x** numeric vector
- **xname** predictor name. Default is "x"
- **internal** a vector of internal knots, excluding boundary knots
- **boundary** vector of external knots
- **degree** the degree of the spline. The broken stick model requires linear splines, so the default is degree = 1. Setting degree = 0 yields (crisp) dummy coding, and one column less than for degree = 1.
- **warn** a logical indicating whether warnings from `splines::bs()` should be given.

**Value**

A matrix with `length(x)` rows and `length(breaks)` columns, with some extra attributes described by `bs()`.

**Note**

Before version 0.54, it was standard practice that the knots array always included `boundary[1L]`.

**Author(s)**

Stef van Buuren, 2020
parse_formula

Parse formula for brokenstick model

Description
A bare bones formula parser to extract variables names from formulas of y ~ x | g. It return the name of the first variable mentioned in each formula component.

Usage
parse_formula(f)

Arguments
f                formula object

Value
A list with elements x, y and g. Each element has length 1.

Author(s)
Stef van Buuren, 2020

plot.brokenstick
Plot observed and fitted trajectories by group

Description
The plot method for a brokenstick object plots the observed and fitted trajectories of one or more groups.

Usage
## S3 method for class 'brokenstick'
plot(x, newdata = NULL, ...)

Arguments
x      An object of class brokenstick.
newdata Optional. A data frame in which to look for variables with which to predict. The training data are used if omitted and if object$light is FALSE.
...    Extra arguments passed down to predict.brokenstick() and plot_trajectory().
plot_trajectory

Plot observed and fitted trajectories from fitted brokenstick model

plot_trajectory(x, 
    newdata = NULL, 
    what = "droplast", 
    .x = NULL, 
    group = NULL, 
)

Details

By default, plot(fit) will plot the observed and fitted data for the first three groups in the data. The default setting drops the fitted value at the right boundary knot from the display.

Value

An object of class ggplot2::ggplot.

Author(s)

Stef van Buuren 2021

See Also

predict.brokenstick, plot_trajectory.

Examples

## Not run:
# fit model on raw hgt with knots at 0, 1, 2 and 3 years
fit1 <- brokenstick(hgt ~ age | id, smooc_200, knots = 0:3)
gp <- c(10001, 10005, 10022)
plot(fit1, group = gp, xlab = "Age (years)", ylab = "Length (cm)"

# fit model on standard deviation score
fit2 <- brokenstick(hgt_z ~ age | id, smooc_200, knots = 0:3)
plot(fit2, group = gp, xlab = "Age (years)", ylab = "Length (SDS)"

# built-in model with 11 knots
plot(fit_200, group = gp, xlab = "Age (years)", ylab = "Length (SDS)"

## End(Not run)
color_y = c(grDevices::hcl(240, 100, 40, 0.7), grDevices::hcl(240, 100, 40, 0.8)),
size_y = 2,
color_yhat = c(grDevices::hcl(0, 100, 40, 0.7), grDevices::hcl(0, 100, 40, 0.8)),
size_yhat = 2,
color_imp = c("grey80", "grey80"),
size_imp = 2,
ncol = 3L,
xlab = NULL,
ylab = NULL,
xlim = NULL,
ylim = NULL,
show = c(TRUE, TRUE, FALSE),
n_plot = 3L,
scales = "fixed",
theme = ggplot2::theme_light(),
...)

Arguments

x
A object of class brokenstick.

newdata
A data.frame or matrix

what
Which knots to plot? See get_knots(). The default, what = "droplast", does not plot the right boundary knot.

.x
The x argument of the predict.brokenstick() function.

group
A vector with group identifications

color_y
A character vector with two elements specifying the symbol and line color of the measured data points

size_y
Dot size of measured data points

color_yhat
A character vector with two elements specifying the symbol and line color of the predicted data points

size_yhat
Dot size of predicted data points

color_imp
A character vector with two elements specifying the symbol and line color of the imputed data

size_imp
Dot size of imputed data

ncol
Number of columns in plot

xlab
The label of the x-axis

ylab
The label of the y-axis

xlim
Vector of length 2 with range of x-axis

ylim
Vector of length 2 with range of y-axis

show
A logical vector of length 3. Element 1 specifies whether the observed data are plotted, element 2 specifies whether the broken stick are plotted, element 3 specifies whether imputations are plotted. The default is c(TRUE, TRUE, FALSE).
predict.brokenstick

n_plot A integer indicating the number of individual plots. The default is 3, which plots the trajectories of the first three groups. The n_plot is a safety measure to prevent unintended plots of the entire data set.
scales Axis scaling, e.g. "fixed", "free", and so on
tHEME Plotting theme
... Extra arguments passed down to predict.brokenstick().

Value
An object of class ggplot

See Also
plot.brokenstick

Description
The predictions from a broken stick model coincide with the group-conditional means of the random effects. This function takes an object of class brokenstick and returns predictions in one of several formats. The user can calculate predictions for new persons, i.e., for persons who are not part of the fitted model, through the x and y arguments.

Usage
## S3 method for class 'brokenstick'
predict(
oBJECT, 
newdata = NULL,
..., 
x = NULL,
y = NULL,
group = NULL,
strip_data = TRUE,
shape = c("long", "wide", "vector"),
what = NULL
)

Arguments
object A brokenstick object.
newdata Optional. A data frame in which to look for variables with which to predict. The training data are used if omitted and if object$light is FALSE.
... Not used, but required for extensibility.
predict.brokenstick

x Optional. A numeric vector with values of the predictor. It could also be the special keyword x = "knots" replaces x by the positions of the knots.
y Optional. A numeric vector with measurements.
group A vector with group identifications
strip_data A logical indicating whether the row with the observed data from newdata should be stripped from the return value. The default is TRUE. Set to FALSE to infer which data points are extracted from newdata. Works best for shape = "long".
shape A string: "long" (default), "wide" or "vector" specifying the shape of the return value. Note that use of "wide" with many unique values in x creates an unwieldy, large and sparse matrix.
what Which knots to predict when x = "knots"? See get_knots(). The default, NULL, calculates all knots.

Details

By default, predict() calculates predictions for every row in newdata. If the user specifies no newdata argument, then the function searches object for the training data (which are only available if object$light is FALSE). It is possible to tailor the behaviour of predict() through the x, y and group arguments. What exactly happens depends on which of these arguments is specified:

1. If the user specifies x, but no y and group, the function returns - for every group in newdata - predictions at the specified x values. This method will use the data from newdata.
2. If the user specifies x and y but no group, the function forms a hypothetical new group with the x and y values. This method uses no information from newdata, and also works for a light brokenstick object.
3. If the user specifies group, but no x or y, the function searches for the relevant data in newdata and limits its predictions to those groups. This is useful if the user needs a prediction for only one or a few groups. This does not work for a light brokenstick object.
4. If the user specifies x and group, but no y, the function will create new values for x in each group, search for the relevant data in newdata and provide predictions at values of x in those groups.
5. If the user specifies x, y and group, the function assumes that these vectors contain additional data on top on what is already available in newdata. The lengths of x, y and group must match. For a light brokenstick object, case effectively becomes case 6. See below.
6. As case 5, but now without newdata available. All data are specified through x, y and group and form a data frame. Matching to newdata is attempted, but as long as group id’s are different from the training sample effectively new cases will be made.

Value

If shape == "long" a long tibble of predictions. If x, y and group are not specified, the number of rows in the tibble is guaranteed to be the same as the number of rows in newdata.
If shape == "wide" a wide tibble of prediction, one record per group. Note that this format could be inefficient, depending on the data.
If shape == "vector" a vector of predicted values, of all x-values and groups.
Examples

```r
library(dplyr)

# -- Data
train <- smocc_200[1:1198,]
test <- smocc_200[1199:1940,]

# -- Fit model
fit <- brokenstick(hgt_z ~ age | id, data = train, knots = 0:3, seed = 1)
fit_light <- brokenstick(hgt_z ~ age | id, data = train, knots = 0:3, light = TRUE, seed = 1)

# -- Predict, standard cases
# Use train data, return column with predictions
pred <- predict(fit)
identical(nrow(train), nrow(pred))

# Predict without newdata, not possible for light object
## Not run:
predict(fit_light)
## End(Not run)

# Use test data
pred <- predict(fit, newdata = test)
identical(nrow(test), nrow(pred))

# Predict, same but using newdata with the light object
pred_light <- predict(fit_light, newdata = test)
identical(pred, pred_light)

# -- Predict, special cases

# -- Case 1: x, -y, -group

# Case 1: x as "knots", standard estimates, train sample (n = 124)
z <- predict(fit, x = "knots", shape = "wide")
head(z, 3)

# Case 1: x as values, linearly interpolated, train sample (n = 124)
z <- predict(fit, x = c(0.5, 1, 1.5), shape = "wide")
head(z, 3)

# Case 1: x as values, linearly interpolated, test sample (n = 76)
z <- predict(fit, test, x = c(0.5, 1, 1.5), shape = "wide")
```
```
head(z, 3)

# -- Case 2: x, y, -group
# Case 2: form one new group with id = 0
predict(fit, x = "knots", y = c(1, 1, 0.5, 0), shape = "wide")

# Case 2: works also for a light object
predict(fit_light, x = "knots", y = c(1, 1, 0.5, 0), shape = "wide")

# -- Case 3: -x, -y, group
# Case 3: Predict at observed age for subset of groups, training sample
pred <- predict(fit, group = c(10001, 10005, 10022))
head(pred, 3)

# Case 3: Of course, we cannot do this for light objects
## Not run:
pred_light <- predict(fit_light, group = c(10001, 10005, 10022))
## End(Not run)

# Case 3: We can use another sample. Note there is no child 999
pred <- predict(fit, test, group = c(11045, 11120, 999))
tail(pred, 3)

# Case 3: Works also for a light object
pred_light <- predict(fit_light, test, group = c(11045, 11120, 999))
identical(pred, pred_light)

# -- Case 4: x, -y, group
# Case 4: Predict at specified x, only in selected groups, train sample
pred <- predict(fit, x = c(0.5, 1, 1.25), group = c(10001, 10005, 10022))
pred

# Case 4: strip_data = FALSE provides access to the observed data
pred_all <- predict(fit, 
  x = c(0.5, 1, 1.25), group = c(10001, 10005, 10022),
  strip_data = FALSE
)
pred_all %>%
dplyr::filter(id == 10001) %>%
dplyr::arrange(age)

# Case 4: Applies also to test sample
pred <- predict(fit, test, x = c(0.5, 1, 1.25), group = c(11045, 11120, 999))
pred

# Case 4: Works also with light object
```
residuals.brokenstick

Extract residuals from brokenstick model

**Description**

Extract residuals from brokenstick model

```r
pred_light <- predict(fit_light, test,
  x = c(0.5, 1, 1.25),
  group = c(11045, 11120, 999)
)
identical(pred_light, pred)

# -- Case 5: x, y, group
# Case 5: Add new data to training sample, and refreshes broken stick
# estimate at age x.
# Note that novel child (not in train) 999 has one data point
predict(fit,
  x = c(0.9, 0.9, 0.9), y = c(1, 1, 1),
  group = c(10001, 10005, 999)
)

# Case 5: Same, but now for test sample. Novel child 899 has two data points
predict(fit, test,
  x = c(0.5, 0.9, 0.6, 0.9),
  y = c(0, 0.5, 0.5, 0.6), group = c(11045, 11120, 899, 899)
)

# Case 5: Also works for light object
predict(fit_light, test,
  x = c(0.5, 0.9, 0.6, 0.9),
  y = c(0, 0.5, 0.5, 0.6), group = c(11045, 11120, 899, 899)
)

# -- Case 6: As Case 5, but without previous data
# Case 6: Same call as last, but now without newdata = test
# All children are de facto novel as they do not occur in the training sample.
# Note: Predictions for 11045 and 11120 differ from prediction in Case 5.
predict(fit,
  x = c(0.5, 0.9, 0.6, 0.9),
  y = c(0, 0.5, 0.5, 0.6), group = c(11045, 11120, 899, 899)
)

# This also work for the light brokenstick object
predict(fit_light,
  x = c(0.5, 0.9, 0.6, 0.9),
  y = c(0, 0.5, 0.5, 0.6), group = c(11045, 11120, 899, 899)
)
```
set_control

Usage

```r
## S3 method for class 'brokenstick'
residuals(object, newdata = NULL, ...)
```

Arguments

- `object`: A brokenstick object.
- `newdata`: Optional. A data frame in which to look for variables with which to predict. The training data are used if omitted and if `object$light` is `FALSE`.
- `...`: Additional arguments. Ignored.

Value

A data.frame with a column named `.resid`

See Also

Other brokenstick: `fitted.brokenstick()`

---

set_control

Set controls to steer calculations

Description

Set controls to steer calculations

Usage

```r
set_control(
  method = c("kr", "lmer"),
  kr = control_kr(...),
  lmer = lmerControl(check.nobs.vs.nRE = "warning"),
  ...
)
```

Arguments

- `method`: String indicating estimation method: "kr" or "lmer"
- `kr`: A list generated by `control_kr`.
- `lmer`: A list generated by `lme4::lmerControl`. The default is set to `lmerControl(check.nobs.vs.nRE = "warning")`, which turns fatal errors with respect to the number of parameters into warnings. Use `lmerControl(check.nobs.vs.nRE = "ignore")` to silence `lmer()`.
- `...`: Forwards arguments to `control_kr()`
Value

For method "kr", a list returned by `control_kr()`. For method "lmer", an object of class `lmerControl`. For other methods, `set_control()` returns NULL.

Examples

```r
# defaults
control <- set_control()
control
```

---

**smocc_200**

*Infant growth of 0-2 years, SMOCC data extract*

Description

Longitudinal height and weight measurements during ages 0-2 years for a representative sample of 1933 Dutch children born in 1988-1989. The dataset `smocc_200` is a subset of the full data covering 200 children.

Format

A tibble with 1940 rows and 7 columns:

- `id` ID, unique id of each child (numeric)
- `age` Decimal age, 0-2.12 years (numeric)
- `sex` Sex, "male" or "female" (character)
- `ga` Gestational age, completed weeks (numeric)
- `bw` Birth weight in grammes (numeric)
- `hgt` Height measurement in cm (34-102) (numeric)
- `hgt_z` Height in SDS relative Fourth Dutch Growth Study 1997 (numeric)

Source

weightloss

Weight loss self-measurement data

Description

Longitudinal weight measurements from 12 individuals with 63 daily measurement under three conditions.

Format

A `data.frame` with 695 rows and 6 columns:

- `subject` ID, consecutive person number 1-12 (integer)
- `day` Measurement day, 0-62 (integer)
- `sex` Sex, 1 = male, 0 = female (integer)
- `week` Week number, 1-9 (integer)
- `condition` Condition (control, diet, activity) (factor)
- `body_weight` Body weight in kg (numeric)

Note

Constructed from file `pone.0232680.s001.csv`. We renumbered `subject` to consecutive integers 1-2 (as in the paper), corrected an error in the `condition` variable for subjects 4 and 12 to match the paper’s Figure 4, and filtered the records to the ones with an observed `body_weight` variable.

Source

Index

* brokenstick
  fitted.brokenstick, 9
  residuals.brokenstick, 22
* datasets
  fit_200, 10
  fit_200_light, 10
  smocc_200, 24
  weightloss, 25

base::set.seed(), 8
brokenstick, 2, 7, 10
brokenstick(), 5, 10
brokenstick-class, 5
brokenstick-pkg, 6

coda::acfplot(), 4
coda::autocorr(), 4
coda::crosscorr(), 4
coda::cumuplot(), 4
coda::densplot(), 4
coda::effectiveSize(), 4
coda::geweke.plot(), 4
coda::mcmc, 6
coda::raftery.diag(), 4
coda::traceplot(), 4
coef(), 5
control.kr, 7, 23
control.kr(), 3, 4, 13, 23, 24

EB, 7, 8
fit_200, 10
fit_200_light, 10
fitted(), 5
fitted.brokenstick, 9, 23

get_knots, 11
get_knots(), 5, 17, 19
get_omega, 11
get_omega(), 5
get_r2, 12

ggplot2::ggplot, 16

kr, 12
lme4::lmer(), 3, 4
lme4::lmerControl, 23
lme4::lmerControl(), 3, 4
lme4::merMod, 4, 6

make_basis, 14
mice::mice.impute.2l.norm(), 13
model.frame(), 5
model.matrix(), 5

parse_formula, 15
plot(), 5
plot.brokenstick, 15, 18
plot_trajectory, 16, 16
plot_trajectory(), 15
predict(), 5
predict.brokenstick, 7, 16, 18
predict.brokenstick(), 9, 15, 17, 18
print(), 5

residuals(), 5
residuals.brokenstick, 9, 22

set_control, 23
set_control(), 3, 6
smocc_200, 24
splines::bs(), 3, 6
summary(), 5

weightloss, 25