Package 'midr'

June 23, 2025

Type Package

Title Learning from Black-Box Models by Maximum Interpretation Decomposition

Version 0.5.0

Description The goal of 'midr' is to provide a model-agnostic method for interpreting and explaining black-box predictive models by creating a globally interpretable surrogate model. The package implements 'Maximum Interpretation Decomposition' (MID), a functional decomposition technique that finds an optimal additive approximation of the original model. This approximation is achieved by minimizing the squared error between the predictions of the black-box model and the surrogate model. The theoretical foundations of MID are described in Iwasawa & Matsumori (2025) [Forthcoming], and the package itself is detailed in Asashiba et al. (2025) <doi:10.48550/arXiv.2506.08338>.

License MIT + file LICENSE

Encoding UTF-8

Imports graphics, grDevices, RcppEigen, rlang, stats, utils

Suggests datasets, ggplot2, khroma, knitr, RColorBrewer, rmarkdown, scales, shapviz, testthat, viridisLite

Config/testthat/edition 3

RoxygenNote 7.3.2

URL https://github.com/ryo-asashi/midr, https://ryo-asashi.github.io/midr/

BugReports https://github.com/ryo-asashi/midr/issues

NeedsCompilation no

Author Ryoichi Asasihba [aut, cre], Hirokazu Iwasawa [aut], Reiji Kozuma [ctb]

Maintainer Ryoichi Asasihba <ryoichi.asashiba@gmail.com>

Repository CRAN

Date/Publication 2025-06-23 10:10:02 UTC

Contents

`color.theme` *Color Themes for Graphics*

Description

`color.theme()` returns an object of class "color.theme" that provides two types of color functions.

Usage

```
color.theme(
  colors,
  type = c("sequential", "qualitative", "diverging"),
  name = NULL,
  pkg = NULL,
  ...
```

```
)

## S3 method for class 'color.theme'
plot(x, n = NULL, text = x$name, ...)

## S3 method for class 'color.theme'
print(x, display = TRUE, ...)
```

Arguments

colors	one of the following: a color theme name such as "Viridis" with the optional suffix "_r" for color themes in reverse order ("Viridis_r"), a character vector of color names, a palette function, or a ramp function to be used to create a color theme.
type	a character string specifying the type of the color theme: One of "sequential", "qualitative" or "diverging".
name	an optional character string, specifying the name of the color theme.
pkg	an optional character string, specifying the package in which the palette is to be searched for. Available options include "viridisLite", "RColorBrewer", "khroma", "grDevices" and "midr".
...	optional arguments to be passed to palette or ramp functions.
x	a "color.theme" object to be displayed.
n	integer. the number of colors.
text	a character string to be displayed.
display	logical. If TRUE, colors are displayed in the plot area.

Details

"color.theme" objects is a container of the two types of color functions: `palette(n)` returns a color name vector of length `n`, and `ramp(x)` returns color names for each values of `x` within [0, 1]. Some color themes are "qualitative" and do not contain `ramp()` function. The color palettes implemented in the following packages are available: `grDevices`, `viridisLite`, `RColorBrewer` and `khroma`.

Value

`color.theme()` returns a "color.theme" object containing following components:

ramp	the function that takes a numeric vector `x` of the values within [0, 1] and returns a color name vector.
palette	the function that takes an integer `n` and returns a color name vector of length `n`.
type	the type of the color theme; "sequential", "diverging" or "qualitative".
name	the name of the color theme.

Examples

```
ct <- color.theme("Mako")
ct$palette(5L)
ct$ramp(seq.int(0, 1, 1/4))
ct <- color.theme("RdBu")
ct$palette(5L)
ct$ramp(seq.int(0, 1, 1/4))
ct <- color.theme("Tableau 10")
ct$palette(10L)
pals <- c("midr", "grayscale", "bluescale", "shap", "DALEX")
pals <- unique(c(pals, hcl.pals(), palette.pals()))
pals <- lapply(pals, color.theme)
old.par <- par(no.readonly = TRUE)
par(mfrow = c(5L, 2L))
for (pal in pals) plot(pal, text = paste(pal$name, "-", pal$type))
par(old.par)
```

---

factor.encoder *Encoder for Qualitative Variables*

Description

`factor.encoder()` returns an encoder for a qualitative variable.

Usage

```
factor.encoder(
  x,
  k,
  use.catchall = TRUE,
  catchall = "(others)",
  tag = "x",
  frame = NULL,
  weights = NULL
)

factor.frame(levels, catchall = "(others)", tag = "x")
```

Arguments

`x`	a vector to be encoded as a qualitative variable.
`k`	an integer specifying the maximum number of distinct levels. If not positive, all unique values of `x` are used as levels.
`use.catchall`	logical. If `TRUE`, less frequent levels are dropped and replaced by the catchall level.
`catchall`	a character string to be used as the catchall level.
`tag`	character string. The name of the variable.

frame	a "factor.frame" object or a character vector that defines the levels of the variable.
weights	optional. A numeric vector of sample weights for each value of x.
levels	a vector to be used as the levels of the variable.

Details

factor.encoder() extracts the unique values (levels) from the vector x and returns a list containing the encode() function to convert a vector into a dummy matrix using one-hot encoding. If use.catchall is TRUE and the number of levels exceeds k, only the most frequent k - 1 levels are used and the other values are replaced by the catchall.

Value

factor.encoder() returns a list containing the following components:

frame	an object of class "factor.frame".
encode	a function to encode x into a dummy matrix.
n	the number of encoding levels.
type	the type of encoding.

factor.frame() returns a "factor.frame" object containing the encoding information.

Examples

```
data(iris, package = "datasets")
enc <- factor.encoder(x = iris$Species, use.catchall = FALSE, tag = "Species")
enc$frame
enc$encode(x = c("setosa", "virginica", "ensata", NA, "versicolor"))

frm <- factor.frame(c("setosa", "virginica"), "other iris")
enc <- factor.encoder(x = iris$Species, frame = frm)
enc$encode(c("setosa", "virginica", "ensata", NA, "versicolor"))

enc <- factor.encoder(x = iris$Species, frame = c("setosa", "versicolor"))
enc$encode(c("setosa", "virginica", "ensata", NA, "versicolor"))
```

---

get.yhat *Wrapper Prediction Function*

Description

get.yhat() works as a proxy prediction function for many classes of fitted models.

Usage

```
get.yhat(X.model, newdata, ...)

## Default S3 method:
get.yhat(X.model, newdata, target = -1L, ...)

## S3 method for class 'mid'
get.yhat(X.model, newdata, ...)

## S3 method for class 'lm'
get.yhat(X.model, newdata, ...)

## S3 method for class 'glm'
get.yhat(X.model, newdata, ...)

## S3 method for class 'rpart'
get.yhat(X.model, newdata, target = -1L, ...)

## S3 method for class 'randomForest'
get.yhat(X.model, newdata, target = -1L, ...)

## S3 method for class 'ranger'
get.yhat(X.model, newdata, target = -1L, ...)

## S3 method for class 'svm'
get.yhat(X.model, newdata, target = -1L, ...)

## S3 method for class 'ksvm'
get.yhat(X.model, newdata, target = -1L, ...)

## S3 method for class 'AccurateGLM'
get.yhat(X.model, newdata, ...)

## S3 method for class 'glmnet'
get.yhat(X.model, newdata, ...)

## S3 method for class 'model_fit'
get.yhat(X.model, newdata, target = -1L, ...)

## S3 method for class 'rpf'
get.yhat(X.model, newdata, target = -1L, ...)
```

Arguments

X.model	a fitted model object.
newdata	a data.frame or matrix.
...	optional parameters that are passed to the prediction method for the model.

target — an integer or character vector specifying the target levels for the prediction, used for the models that returns a matrix or data.frame of class probabilities. Default is -1, representing the probability of not being the base level.

Details

get.yhat() is a wrapper prediction function for many classes of models. Although many predictive models have their own method of stats::predict(), the structure and the type of the output of these methods are not uniform. get.yhat() is designed to always return a simple numeric vector of model predictions. The design of get.yhat() is strongly influenced by DALEX::yhat().

Value

get.yhat() returns a numeric vector of model predictions for the newdata.

Examples

```
data(trees, package = "datasets")
model <- glm(Volume ~ ., trees, family = Gamma(log))
predict(model, trees[1:5, ], type = "response")
get.yhat(model, trees[1:5, ])
```

---

ggmid *Plot MID with ggplot2 Package*

Description

For "mid" objects, ggmid() visualizes a MID component function using the ggplot2 package.

Usage

```
ggmid(object, ...)

## S3 method for class 'mid'
ggmid(
  object,
  term,
  type = c("effect", "data", "compound"),
  theme = NULL,
  intercept = FALSE,
  main.effects = FALSE,
  data = NULL,
  jitter = 0.3,
  cells.count = c(100L, 100L),
  limits = c(NA, NA),
  ...
)
```

```
## S3 method for class 'mid'
autoplot(object, ...)
```

Arguments

object	a "mid" object to be visualized.
...	optional parameters to be passed to the main layer.
term	a character string specifying the component function to be plotted.
type	character string. The method for plotting the interaction effects.
theme	a character string specifying the color theme or any item that can be used to define "color.theme" object.
intercept	logical. If TRUE, the intercept is added to the MID values.
main.effects	logical. If TRUE, the main effects are included in the interaction plot.
data	a data.frame to be plotted with the corresponding MID values. If not passed, data is extracted from parent.env() based on the function call of the "mid" object.
jitter	a numeric value specifying the amount of jitter for points.
cells.count	an integer or integer-valued vector of length two, specifying the number of cells for the raster type interaction plot.
limits	NULL or a numeric vector of length two specifying the limits of the plotting scale. NAs are replaced by the minimum and/or maximum MID values.

Details

The S3 method of ggmid() for "mid" objects creates a "ggplot" object that visualizes a MID component function. The main layer is drawn using geom_line() or geom_path() for a main effect of a quantitative variable, geom_col() for a main effect of a qualitative variable, and geom_raster() or geom_rect() for an interaction effect. For other methods of ggmid(), see help(ggmid.mid.importance), help(ggmid.mid.breakdown) or help(ggmid.mid.conditional).

Value

ggmid.mid() returns a "ggplot" object.

Examples

```
data(diamonds, package = "ggplot2")
set.seed(42)
idx <- sample(nrow(diamonds), 1e4)
mid <- interpret(price ~ (carat + cut + color + clarity)^2, diamonds[idx, ])
ggmid(mid, "carat")
ggmid(mid, "clarity")
ggmid(mid, "carat:clarity", main.effects = TRUE)
ggmid(mid, "clarity:color", type = "data", theme = "Mako", data = diamonds[idx, ])
ggmid(mid, "carat:color", type = "compound", data = diamonds[idx, ])
```

ggmid.mid.breakdown *Plot MID Breakdown with ggplot2 Package*

Description

For "mid.breakdown" objects, `ggmid()` visualizes the breakdown of a prediction by component functions.

Usage

```
## S3 method for class 'mid.breakdown'
ggmid(
  object,
  type = c("waterfall", "barplot", "dotchart"),
  theme = NULL,
  terms = NULL,
  max.bars = 15L,
  width = NULL,
  vline = TRUE,
  catchall = "others",
  format = c("%t=%v", "%t"),
  ...
)

## S3 method for class 'mid.breakdown'
autoplot(object, ...)
```

Arguments

`object`	a "mid.breakdown" object to be visualized.
`type`	a character string specifying the type of the plot. One of "waterfall", "barplot" or "dotchart".
`theme`	a character string specifying the color theme or any item that can be used to define "color.theme" object.
`terms`	an optional character vector specifying the terms to be displayed.
`max.bars`	an integer specifying the maximum number of bars in the plot.
`width`	a numeric value specifying the width of the bars.
`vline`	logical. If `TRUE`, the vertical line is drawn at zero or the intercept.
`catchall`	a character string to be used as the catchall label.
`format`	a character string or character vector of length two to be used as the format of the axis labels. "t" and "v" immediately after the percent sign are replaced with the corresponding term and value.
`...`	optional parameters to be passed to the main layer.

Details

The S3 method of `ggmid()` for "mid.breakdown" objects creates a "ggplot" object that visualizes the breakdown of a single model prediction. The main layer is drawn using `geom_col()`.

Value

`ggmid.mid.breakdown()` returns a "ggplot" object.

Examples

```
data(diamonds, package = "ggplot2")
set.seed(42)
idx <- sample(nrow(diamonds), 1e4)
mid <- interpret(price ~ (carat + cut + color + clarity)^2, diamonds[idx, ])
mbd <- mid.breakdown(mid, diamonds[1L, ])
ggmid(mbd, type = "waterfall")
ggmid(mbd, type = "waterfall", theme = "midr")
ggmid(mbd, type = "barplot", theme = "Set 1")
ggmid(mbd, type = "dotchart", size = 3, theme = "Cividis")
```

`ggmid.mid.conditional` *Plot ICE of MID Model with ggplot2 Package*

Description

For "mid.conditional" objects, `ggmid()` visualizes ICE curves of a MID model.

Usage

```
## S3 method for class 'mid.conditional'
ggmid(
  object,
  type = c("iceplot", "centered"),
  theme = NULL,
  term = NULL,
  var.alpha = NULL,
  var.color = NULL,
  var.linetype = NULL,
  var.linewidth = NULL,
  reference = 1L,
  dots = TRUE,
  sample = NULL,
  ...
)

## S3 method for class 'mid.conditional'
autoplot(object, ...)
```

Arguments

object	a "mid.conditional" object to be visualized.
type	a character string specifying the type of the plot. One of "iceplot" or "centered". If "centered", the ICE values of each observation are set to zero at the leftmost point of the varriable.
theme	a character string specifying the color theme or any item that can be used to define "color.theme" object.
term	an optional character string specifying an interaction term. If passed, the ICE curve for the specified term is plotted.
var.alpha	a name of the variable or an expression to be used to set alpha.
var.color	a name of the variable or an expression to be used to set colour.
var.linetype	a name of the variable or an expression to be used to set linetype.
var.linewidth	a name of the variable or an expression to be used to set linewidth.
reference	an integer specifying the index of the sample points to be used as reference point for the centered ICE plot. Default is 1. If negative, the maximum value of the variable is used.
dots	logical. If TRUE, the points representing the predictions for each observation are plotted.
sample	an optional vector specifying the names of observations to be plotted.
...	optional parameters to be passed to the main layer.

Details

The S3 method of ggmid() for "mid.conditional" objects creates a "ggplot" object that visualizes ICE curves of a fitted MID model using geom_line().

Value

ggmid.mid.conditional() returns a "ggplot" object.

Examples

```
data(airquality, package = "datasets")
library(midr)
mid <- interpret(Ozone ~ .^2, airquality, lambda = 0.1)
ice <- mid.conditional(mid, "Temp", data = airquality)
ggmid(ice, var.color = "Wind")
ggmid(ice, type = "centered", theme = "Purple-Yellow",
      var.color = factor(Month), var.linetype = Wind > 10)
```

ggmid.mid.importance *Plot MID Importance with ggplot2 Package*

Description

For "mid.importance" objects, ggmid() visualizes the importance of MID component functions.

Usage

```
## S3 method for class 'mid.importance'
ggmid(
  object,
  type = c("barplot", "dotchart", "heatmap", "boxplot"),
  theme = NULL,
  max.bars = 30L,
  ...
)

## S3 method for class 'mid.importance'
autoplot(object, ...)
```

Arguments

object	a "mid.importance" object to be visualized.
type	a character string specifying the type of the plot. One of "barplot", "heatmap", "dotchart" or "boxplot".
theme	a character string specifying the color theme or any item that can be used to define "color.theme" object.
max.bars	an integer specifying the maximum number of bars in the barplot, boxplot and dotchart.
...	optional parameters to be passed to the main layer.

Details

The S3 method of ggmid() for "mid.importance" objects creates a "ggplot" object that visualizes the term importance of a fitted MID model. The main layer is drawn using geom_col(), geom_tile(), geom_point() or geom_boxplot().

Value

ggmid.mid.importance() returns a "ggplot" object.

Examples

```
data(diamonds, package = "ggplot2")
set.seed(42)
idx <- sample(nrow(diamonds), 1e4)
mid <- interpret(price ~ (carat + cut + color + clarity)^2, diamonds[idx, ])
imp <- mid.importance(mid)
ggmid(imp, theme = "Tableau 10")
ggmid(imp, type = "dotchart", theme = "Okabe-Ito", size = 3)
ggmid(imp, type = "heatmap", theme = "Blues")
ggmid(imp, type = "boxplot", theme = "Accent")
```

`interpret` *Fit MID Models*

Description

`interpret()` is used to fit a MID model specifically as an interpretable surrogate for black-box predictive models. A fitted MID model consists of a set of component functions, each with up to two variables.

Usage

```
interpret(object, ...)

## Default S3 method:
interpret(
  object,
  x,
  y = NULL,
  weights = NULL,
  pred.fun = get.yhat,
  link = NULL,
  k = c(NA, NA),
  type = c(1L, 1L),
  frames = list(),
  interaction = FALSE,
  terms = NULL,
  singular.ok = FALSE,
  mode = 1L,
  method = NULL,
  lambda = 0,
  kappa = 1e+06,
  na.action = getOption("na.action"),
  verbosity = 1L,
  encoding.digits = 3L,
  use.catchall = FALSE,
  catchall = "(others)",
```

```
  max.ncol = 10000L,
  nil = 1e-07,
  tol = 1e-07,
  pred.args = list(),
  ...
)

## S3 method for class 'formula'
interpret(
  formula,
  data = NULL,
  model = NULL,
  pred.fun = get.yhat,
  weights = NULL,
  subset = NULL,
  na.action = getOption("na.action"),
  verbosity = 1L,
  mode = 1L,
  drop.unused.levels = FALSE,
  pred.args = list(),
  ...
)
```

Arguments

object	a fitted model object to be interpreted.
...	for interpret.default(), optional arguments can be provided, including fit.intercept, interpolate.beta, weighted.norm, and weighted.encoding. Special character aliases are also supported, such as ok for singular.ok and ie for interaction. For interpret.formula(), any arguments to be passed on to interpret.default().
x	a matrix or data.frame of predictor variables to be used in the fitting process. The response variable should not be included.
y	an optional numeric vector of the model predictions or the response variable.
weights	a numeric vector of sample weights for each observation in x.
pred.fun	a function to obtain predictions from a fitted model, where the first argument is for the fitted model and the second argument is for new data. The default is get.yhat().
link	a character string specifying the link function: one of "logit", "probit", "cauchit", "cloglog", "identity", "log", "sqrt", "1/mu^2", "inverse", "translogit", "transprobit", "identity-logistic" and "identity-gaussian", or an object containing two functions linkfun() and linkinv(). See help(make.link).
k	an integer or integer-valued vector of length two. The maximum number of sample points for each variable. If a vector is passed, k[1L] is used for main effects and k[2L] is used for interactions. If an integer is passed, k is used for main effects and sqrt(k) is used for interactions. If not positive, all unique values are used as sample points.

type	an integer or integer-valued vector of length two. The type of encoding. The effects of quantitative variables are modeled as piecewise linear functions if type is 1, and as step functions if type is 0. If a vector is passed, type[1L] is used for main effects and type[2L] is used for interactions.
frames	a named list of encoding frames ("numeric.frame" or "factor.frame" objects). The encoding frames are used to encode the variable of the corresponding name. If the name begins with "|" or ":", the encoding frame is used only for main effects or interactions, respectively.
interaction	logical. If TRUE and if terms and formula are not supplied, all interactions for each pair of variables are modeled and calculated.
terms	a character vector of term labels specifying the set of component functions to be modeled. If not passed, terms includes all main effects, and all interactions if interaction is TRUE.
singular.ok	logical. If FALSE, a singular fit is an error.
mode	an integer specifying the method of calculation. If mode is 1, the centralization constraints are treated as penalties for the least squares problem. If mode is 2, the constraints are used to reduce the number of free parameters.
method	an integer specifying the method to be used to solve the least squares problem. A non-negative value will be passed to RcppEigen::fastLmPure(). If negative, stats::lm.fit() is used.
lambda	the penalty factor for pseudo smoothing. The default is 0.
kappa	the penalty factor for centering constraints. Used only when mode is 1. The default is 1e+6.
na.action	a function or character string specifying the method of NA handling. The default is "na.omit".
verbosity	the level of verbosity. 0: fatal, 1: warning (default), 2: info or 3: debug.
encoding.digits	an integer. The rounding digits for encoding numeric variables. Used only when type is 1.
use.catchall	logical. If TRUE, less frequent levels of qualitative variables are dropped and replaced by the catchall level.
catchall	a character string specifying the catchall level.
max.ncol	integer. The maximum number of columns of the design matrix.
nil	a threshold for the intercept and coefficients to be treated as zero. The default is 1e-7.
tol	a tolerance for the singular value decomposition. The default is 1e-7.
pred.args	optional parameters other than the fitted model and new data to be passed to pred.fun().
formula	a symbolic description of the MID model to be fit.
data	a data.frame, list or environment containing the variables in formula. If not found in data, the variables are taken from environment(formula).
model	a fitted model object to be interpreted.

`subset` an optional vector specifying a subset of observations to be used in the fitting process.

`drop.unused.levels` logical. If `TRUE`, unused levels of factors will be dropped.

Details

`interpret()` returns a global surrogate model of the target predictive model. The prediction function of this surrogate model is derived from Maximum Interpretation Decomposition (MID) applied to the prediction function of the target model (denoted $f(\mathbf{x})$).

The prediction function of the global surrogate model, denoted $\mathcal{F}(\mathbf{x})$, has the following structure:

$$\mathcal{F}(\mathbf{x}) = f_\phi + \sum_j f_j(x_j) + \sum_{j<k} f_{jk}(x_j, x_k)$$

where f_ϕ is the intercept, $f_j(x_j)$ is the main effect of feature j, and $f_{jk}(x_j, x_k)$ is the second-order interaction effect between features j and k.

To ensure the identifiability (uniqueness) of these decomposed components, they are subject to centering constraints during the fitting process. Specifically, each main effect function $f_j(x_j)$ is constrained such that its average over the data distribution of feature X_j is zero. Similarly, each second-order interaction effect function $f_{jk}(x_j, x_k)$ is constrained such that its conditional average over X_j (for any fixed value x_k) is zero, and its conditional average over X_k (for any fixed value x_j) is also zero.

The surrogate model is fitted using the least squares method, which minimizes the squared error between the predictions of the target model $f(\mathbf{x})$ and the surrogate model $\mathcal{F}(\mathbf{x})$ (typically evaluated on a representative dataset).

Value

`interpret()` returns a "mid" object with the following components:

`weights` a numeric vector of the sample weights.

`call` the matched call.

`terms` the term labels.

`link` a "link-glm" or "link-midr" object containing the link function.

`intercept` the intercept.

`encoders` a list of variable encoders.

`main.effects` a list of data frames representing the main effects.

`interacions` a list of data frames representing the interactions.

`ratio` the ratio of the sum of squared error between the target model predictions and the fitted MID values, to the sum of squared deviations of the target model predictions.

`fitted.matrix` a matrix showing the breakdown of the predictions into the effects of the component functions.

`linear.predictors` a numeric vector of the linear predictors.

fitted.values	a numeric vector of the fitted values.
residuals	a numeric vector of the working residuals.
na.action	information about the special handlings of NAs.

Examples

```
# fit a MID model as a surrogate model
data(cars, package = "datasets")
model <- lm(dist ~ I(speed^2) + speed, cars)
mid <- interpret(dist ~ speed, cars, model)
plot(mid, "speed", intercept = TRUE)
points(cars)

# customize the flexibility of a MID model
data(Nile, package = "datasets")
mid <- interpret(x = 1L:100L, y = Nile, k = 100L)
plot(mid, "x", intercept = TRUE, limits = c(600L, 1300L))
points(x = 1L:100L, y = Nile)
# reduce the number of knots by setting the 'k' parameter
mid <- interpret(x = 1L:100L, y = Nile, k = 10L)
plot(mid, "x", intercept = TRUE, limits = c(600L, 1300L))
points(x = 1L:100L, y = Nile)
# perform a pseudo smoothing by setting the 'lambda' parameter
mid <- interpret(x = 1L:100L, y = Nile, k = 100L, lambda = 100L)
plot(mid, "x", intercept = TRUE, limits = c(600L, 1300L))
points(x = 1L:100L, y = Nile)

# fit a MID model as a predictive model
data(airquality, package = "datasets")
mid <- interpret(Ozone ~ .^2, na.omit(airquality), lambda = .4)
plot(mid, "Wind")
plot(mid, "Temp")
plot(mid, "Wind:Temp", theme = "RdBu")
plot(mid, "Wind:Temp", main.effects = TRUE)
```

---

mid.breakdown *Calculate MID Breakdown*

Description

mid.breakdown() calculates the MID breakdown of a prediction of the MID model.

Usage

```
mid.breakdown(
  object,
  data = NULL,
  sort = TRUE,
  digits = 6L,
```

```
  format = c("%s", "%s, %s")
)

## S3 method for class 'mid.breakdown'
print(x, digits = max(3L, getOption("digits") - 2L), ...)
```

Arguments

object	a "mid" object.
data	a data.frame containing a single observation to be used to calculate the MID breakdown. If NULL, data is extracted from parent.env() based on the function call of the "mid" object.
sort	logical. If TRUE, the output data frame is sorted by MID .
digits	an integer specifying the minimum number of significant digits.
format	a character vector of length two to be used as the formats of the sprintf() function for each value or pair of values of predictor variables.
x	a "mid.importance" object to be printed.
...	additional parameters to be passed to print.data.frame() to print the importance of component functions.

Details

mid.breakdown() returns an object of class "mid.breakdown".

Value

mid.breakdown() returns an object of the class "mid.breakdown" containing the following components.

breakdown	the data frame containing the breakdown of the prediction.
data	the data frame containing the values of predictor variables used for the prediction.
intercept	the intercept of the MID model.
prediction	the predicted value.

Examples

```
data(airquality, package = "datasets")
mid <- interpret(Ozone ~ .^2, airquality, lambda = 1)
mbd <- mid.breakdown(mid, airquality[1L, ])
mbd
```

`mid.conditional` *Calculate ICE of MID Models*

Description

`mid.conditional()` creates an object to draw ICE curves of a MID model.

Usage

```
mid.conditional(
  object,
  variable,
  data = NULL,
  keep.effects = TRUE,
  n.samples = 100L,
  max.nrow = 100000L,
  type = c("response", "link")
)

## S3 method for class 'mid.conditional'
print(x, digits = max(3L, getOption("digits") - 2L), ...)
```

Arguments

`object`	a "mid" object.
`variable`	a character string or expression specifying the variable for the ICE calculation.
`data`	a data frame containing observations for which ICE values are calculated. If not passed, data is extracted from `parent.env()` based on the function call of the "mid" object.
`keep.effects`	logical. If `TRUE`, the effects of component functions are stored in the output object.
`n.samples`	integer. The number of sample points for the calculation.
`max.nrow`	an integer specifying the maximum number of rows of the output data frames.
`type`	the type of prediction required. The default is "response". "link" is possible if the MID model uses a link function.
`x`	a "mid.conditional" object to be printed.
`digits`	an integer specifying the minimum number of significant digits to be printed.
`...`	additional parameters to be passed to `print.default()` to print the sample point vector.

Details

`mid.conditional()` obtains predictions for hypothetical observations from a MID model and returns a "mid.conditional" object. The graphing functions `ggmid()` and `plot()` can be used to generate the ICE curve plots.

Value

`mid.conditional()` returns an object of class "mid.conditional" with the following components:

`terms`	the character vector of relevant terms.
`observed`	the data frame of the actual observations and the corresponding predictions.
`conditional`	the data frame of the hypothetical observations and the corresponding predictions.
`values`	the sample points of the variable.

Examples

```
data(airquality, package = "datasets")
mid <- interpret(Ozone ~ .^2, airquality, lambda = 1)
mc <- mid.conditional(mid, "Wind", airquality)
mc
```

`mid.extract` *Extract Components from MID Models*

Description

`mid.extract()` returns a component of a MID model.

Usage

```
mid.extract(object, component, ...)

mid.encoding.scheme(object, ...)

mid.frames(object, ...)

mid.terms(
  object,
  main.effect = TRUE,
  interaction = TRUE,
  require = NULL,
  remove = NULL,
  ...
)

## S3 method for class 'mid'
terms(x, ...)

## S3 method for class 'mid.importance'
terms(x, ...)
```

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

## S3 method for class 'mid'
model.frame(object, ...)
```

Arguments

object	a "mid" object.
component	a literal character string or name. The name of the component to extract, such as "frames", "encoding.scheme" and "terms".
...	optional parameters to be passed to the function used to extract the component.
main.effect	logical. If FALSE, the main effect terms are excluded.
interaction	logical. If FALSE, the interaction terms are excluded.
require	a character vector of variable names. The terms that are not related to any of the specified names are excluded.
remove	a character vector of variable names. The terms that are related to at least one of the specified names are excluded.
x	a "mid" or "mid.importance" object.

Value

`mid.extract()` returns the `component` extracted from the `object`, `mid.encoding.scheme()` returns a data frame containing the information about encoding schemes, `mid.frames()` returns a list of the encoding frames, `mid.terms()` returns a character vector of the term labels, and

Examples

```
data(trees, package = "datasets")
mid <- interpret(Volume ~ .^2, trees, k = 10)
mid.extract(mid, encoding.scheme)
mid.extract(mid, frames)
mid.extract(mid, Girth)
mid.extract(mid, intercept)
```

---

mid.importance *Calculate MID Importance*

Description

`mid.importance()` calculates the MID importance of a fitted MID model.

Usage

```
mid.importance(object, data = NULL, weights = NULL, sort = TRUE, measure = 1L)

## S3 method for class 'mid.importance'
print(x, digits = max(3L, getOption("digits") - 2L), ...)
```

Arguments

object	a "mid" object.
data	a data frame containing the observations to be used to calculate the MID importance. If NULL, the fitted.matrix of the MID model is used.
weights	an optional numeric vector of sample weights.
sort	logical. If TRUE, the output data frame is sorted by MID importance.
measure	an integer specifying the measure of the MID importance. Possible alternatives are 1 for the mean absolute effect, 2 for the root mean square effect, and 3 for the median absolute effect.
x	a "mid.importance" object to be printed.
digits	an integer specifying the minimum number of significant digits to be printed.
...	additional parameters to be passed to print.data.frame() to print the importance of component functions.

Details

mid.importance() returns an object of class "mid.importance". The MID importance is defined for each component function of a MID model as the mean absolute effect in the given data.

Value

mid.importance() returns an object of the class "mid.importance" containing the following components.

importance	the data frame of calculated importances.
predictions	the matrix of the fitted or predicted MID values.
measure	the type of the importance measure.

Examples

```
data(airquality, package = "datasets")
mid <- interpret(Ozone ~ .^2, airquality, lambda = 1)
imp <- mid.importance(mid)
imp
```

---

mid.plots *Plot Multiple MID Component Functions*

Description

mid.plots() applies ggmid() or plot() to the component functions of a "mid" object.

Usage

```
mid.plots(
  object,
  terms = mid.terms(object, interaction = FALSE),
  limits = c(NA, NA),
  intercept = FALSE,
  main.effects = FALSE,
  max.plots = NULL,
  engine = c("ggplot2", "graphics"),
  ...
)
```

Arguments

object	a "mid" object.
terms	a character vector. The names of the terms to be visualized.
limits	NULL or a numeric vector of length two specifying the limits of the plotting scale. NAs are replaced by the minimum and/or maximum MID values.
intercept	logical. If TRUE, the intercept is added to the MID values and the plotting scale is shifted.
main.effects	logical. If TRUE, the main effects are included in the interaction plot.
max.plots	an integer specifying the number of maximum number of plots.
engine	character string. One of "ggplot2" or "graphics".
...	optional parameters to be passed to ggmid() or plot().

Value

If engine is "ggplot2", mid.plots() returns a list of "ggplot" objects. Otherwise mid.plots() produces plots and returns NULL.

Examples

```
data(diamonds, package = "ggplot2")
set.seed(42)
idx <- sample(nrow(diamonds), 1e4L)
mid <- interpret(price ~ (carat + cut + color + clarity) ^ 2, diamonds[idx, ])
mid.plots(mid, c("carat", "color", "carat:color", "clarity:color"), limits = NULL)
```

---

numeric.encoder *Encoder for Quantitative Variables*

Description

numeric.encoder() returns an encoder for a quantitative variable.

Usage

```
numeric.encoder(
  x,
  k,
  type = 1L,
  encoding.digits = NULL,
  tag = "x",
  frame = NULL,
  weights = NULL
)

numeric.frame(
  reps = NULL,
  breaks = NULL,
  type = NULL,
  encoding.digits = NULL,
  tag = "x"
)

## S3 method for class 'encoder'
print(x, digits = NULL, ...)
```

Arguments

x	a numeric vector to be encoded.
k	an integer specifying the coarseness of the encoding. If not positive, all unique values of x are used as sample points.
type	an integer specifying the encoding method. If 1, values are encoded to a [0, 1] scale based on linear interpolation of the knots. If 0, values are encoded to 0 or 1 using ont-hot encoding on the intervals.
encoding.digits	an integer specifying the rounding digits for the encoding in case type is 1.
tag	character string. The name of the variable.
frame	a "numeric.frame" object or a numeric vector that defines the sample points of the binning.
weights	optional. A numeric vector of sample weights for each value of x.
reps	a numeric vector to be used as the representative values (knots).
breaks	a numeric vector to be used as the binning breaks.
digits	the minimum number of significant digits to be used.
...	not used.

Details

numeric.encoder() selects sample points from the variable x and returns a list containing the encode() function to convert a vector into a dummy matrix. If type is 1, k is considered the maximum number of knots, and the values between two knots are encoded as two decimals, reflecting

the relative position to the knots. If `type` is `0`, `k` is considered the maximum number of intervals, and the values are converted using one-hot encoding on the intervals.

Value

`numeric.encoder()` returns a list containing the following components:

`frame`	an object of class "numeric.frame".
`encode`	a function to encode `x` into a dummy matrix.
`n`	the number of encoding levels.
`type`	the type of encoding, "linear" or "constant".

`numeric.frame()` returns a "numeric.frame" object containing the encoding information.

Examples

```
data(iris, package = "datasets")
enc <- numeric.encoder(x = iris$Sepal.Length, k = 5L, tag = "Sepal.Length")
enc$frame
enc$encode(x = c(4:8, NA))

frm <- numeric.frame(breaks = seq(3, 9, 2), type = 0L)
enc <- numeric.encoder(x = iris$Sepal.Length, frame = frm)
enc$encode(x = c(4:8, NA))

enc <- numeric.encoder(x = iris$Sepal.Length, frame = seq(3, 9, 2))
enc$encode(x = c(4:8, NA))
```

---

plot.mid *Plot MID with graphics Package*

Description

For "mid" objects, `plot()` visualizes a MID component function.

Usage

```
## S3 method for class 'mid'
plot(
  x,
  term,
  type = c("effect", "data", "compound"),
  theme = NULL,
  intercept = FALSE,
  main.effects = FALSE,
  data = NULL,
  jitter = 0.3,
  cells.count = c(100L, 100L),
```

```
  limits = NULL,
  ...
)
```

Arguments

x	a "mid" object to be visualized.
term	a character string specifying the component function to be plotted.
type	character string.
theme	a character vector of color names or a character string specifying the color theme.
intercept	logical. If TRUE, the intercept is added to the MID values and the plotting scale is shifted.
main.effects	logical. If TRUE, the main effects are included in the interaction plot.
data	a data.frame to be plotted with the corresponding MID values. If not passed, data is extracted from parent.env() based on the function call of the "mid" object.
jitter	a numeric value specifying the amount of jitter for points.
cells.count	an integer or integer-valued vector of length two specifying the number of cells for the raster type interaction plot.
limits	NULL or a numeric vector of length two specifying the limits of the plotting scale. NAs are replaced by the minimum and/or maximum MID values.
...	optional parameters to be passed to the graphing function. Possible arguments are "col", "fill", "pch", "cex", "lty", "lwd" and aliases of them.

Details

The S3 method of plot() for "mid" objects creates a visualization of a MID component function using the functions of the graphics package.

Value

plot.mid() produces a line plot or bar plot for a main effect and a filled contour plot for an interaction and returns NULL.

Examples

```
data(diamonds, package = "ggplot2")
set.seed(42)
idx <- sample(nrow(diamonds), 1e4)
mid <- interpret(price ~ (carat + cut + color + clarity)^2, diamonds[idx, ])
plot(mid, "carat")
plot(mid, "clarity")
plot(mid, "carat:clarity", main.effects = TRUE)
plot(mid, "clarity:color", type = "data", theme = "Mako", data = diamonds[idx, ])
plot(mid, "carat:color", type = "compound", data = diamonds[idx, ])
```

`plot.mid.breakdown` *Plot MID Breakdown with graphics Package*

Description

For "mid.breakdown" objects, `plot()` visualizes the breakdown of a prediction by component functions.

Usage

```
## S3 method for class 'mid.breakdown'
plot(
  x,
  type = c("waterfall", "barplot", "dotchart"),
  theme = NULL,
  terms = NULL,
  max.bars = 15L,
  width = NULL,
  vline = TRUE,
  catchall = "others",
  format = c("%t=%v", "%t"),
  ...
)
```

Arguments

`x`	a "mid.breakdown" object to be visualized.
`type`	a character string specifying the type of the plot. One of "barplot" or "dotchart".
`theme`	a character string specifying the color theme or any item that can be used to define "color.theme" object.
`terms`	an optional character vector specifying the terms to be displayed.
`max.bars`	an integer specifying the maximum number of bars in the barplot, boxplot and dotchart.
`width`	a numeric value specifying the width of the bars.
`vline`	logical. If `TRUE`, the vertical line is drawn at zero or the intercept.
`catchall`	a character string to be used as the catchall label.
`format`	a character string or character vector of length two to be used as the format of the axis labels. "t" and "v" immediately after the percent sign are replaced with the corresponding term and value.
`...`	optional parameters to be passed to the graphing function. Possible arguments are "col", "fill", "pch", "cex", "lty", "lwd" and aliases of them.

Details

The S3 method of `plot()` for "mid.breakdown" objects creates a visualization of the MID breakdown using the functions of the graphics package.

Value

`plot.mid.breakdown()` produces a plot and returns `NULL`.

Examples

```
data(diamonds, package = "ggplot2")
set.seed(42)
idx <- sample(nrow(diamonds), 1e4)
mid <- interpret(price ~ (carat + cut + color + clarity)^2, diamonds[idx, ])
mbd <- mid.breakdown(mid, diamonds[1L, ])
plot(mbd, type = "waterfall")
plot(mbd, type = "waterfall", theme = "midr")
plot(mbd, type = "barplot", theme = "Set 1")
plot(mbd, type = "dotchart", theme = "Cividis")
```

---

plot.mid.conditional *Plot ICE of MID Model with graphics Package*

Description

For "mid.conditional" objects, `plot()` visualizes ICE curves of a MID model.

Usage

```
## S3 method for class 'mid.conditional'
plot(
  x,
  type = c("iceplot", "centered"),
  theme = NULL,
  term = NULL,
  var.alpha = NULL,
  var.color = NULL,
  var.linetype = NULL,
  var.linewidth = NULL,
  reference = 1L,
  dots = TRUE,
  sample = NULL,
  ...
)
```

Arguments

x	a "mid.conditional" object to be visualized.
type	a character string specifying the type of the plot. One of "iceplot" or "centered". If "centered", the ICE values of each observation are set to zero at the leftmost point of the varriable.

theme	a character string specifying the color theme or any item that can be used to define "color.theme" object.
term	an optional character string specifying the interaction term. If passed, the ICE for the specified term is plotted.
var.alpha	a name of the variable or an expression to be used to set alpha.
var.color	a name of the variable or an expression to be used to set colour.
var.linetype	a name of the variable or an expression to be used to set linetype.
var.linewidth	a name of the variable or an expression to be used to set linewidth.
reference	an integer specifying the index of the sample points to be used as reference point for the centered ICE plot. Default is 1. If negative, the maximum value of the variable is used.
dots	logical. If TRUE, the points representing the predictions for each observation are plotted.
sample	an optional vector specifying the names of observations to be plotted.
...	optional parameters to be passed to the graphing function. Possible arguments are "col", "fill", "pch", "cex", "lty", "lwd" and aliases of them.

Details

The S3 method of plot() for "mid.conditional" objects creates an visualization of ICE curves of a fitted MID model using the functions of the graphics package.

Value

plot.mid.conditional() produces an ICE plot and invisibly returns the ICE matrix used for the plot.

Examples

```
data(airquality, package = "datasets")
library(midr)
mid <- interpret(Ozone ~ .^2, airquality, lambda = 0.1)
ice <- mid.conditional(mid, "Temp", data = airquality)
plot(ice, var.color = "Wind")
plot(ice, type = "centered", theme = "Purple-Yellow",
     var.color = factor(Month), var.linetype = Wind > 10)
```

---

plot.mid.importance *Plot MID Importance with graphics Package*

Description

For "mid.importance" objects, plot() visualizes the importance of MID component functions.

Usage

```
## S3 method for class 'mid.importance'
plot(
  x,
  type = c("barplot", "dotchart", "heatmap", "boxplot"),
  theme = NULL,
  max.bars = 30L,
  ...
)
```

Arguments

x	a "mid.importance" object to be visualized.
type	a character string specifying the type of the plot. One of "barplot", "heatmap", "dotchart" or "boxplot".
theme	a character string specifying the color theme or any item that can be used to define "color.theme" object.
max.bars	an integer specifying the maximum number of bars in the barplot, boxplot and dotchart.
...	optional parameters to be passed to the graphing function. Possible arguments are "col", "fill", "pch", "cex", "lty", "lwd" and aliases of them.

Details

The S3 method of `plot()` for "mid.importance" objects creates a visualization of the MID importance using the functions of the graphics package.

Value

`plot.mid.importance()` produces a plot and returns `NULL`.

Examples

```
data(diamonds, package = "ggplot2")
set.seed(42)
idx <- sample(nrow(diamonds), 1e4)
mid <- interpret(price ~ (carat + cut + color + clarity)^2, diamonds[idx, ])
imp <- mid.importance(mid)
plot(imp, theme = "Tableau 10")
plot(imp, type = "dotchart", theme = "Okabe-Ito")
plot(imp, type = "heatmap", theme = "Blues")
plot(imp, type = "boxplot", theme = "Accent")
```

predict.mid *Predict Method for fitted MID Models*

Description

The method of `predict()` for "mid" objects obtains predictions from a fitted MID model.

Usage

```
## S3 method for class 'mid'
predict(
  object,
  newdata = NULL,
  na.action = "na.pass",
  type = c("response", "link", "terms"),
  terms = object$terms,
  ...
)

mid.f(object, term, x, y = NULL)
```

Arguments

`object`	a "mid" object to be used to make predictions.
`newdata`	a data frame of the new observations.
`na.action`	a function or character string specifying what should happen when the data contain `NA`s.
`type`	the type of prediction required. The default is on the scale of the response varialbe. The alternative "link" is on the scale of the linear predictors. The "terms" option returns a matrix giving the fitted values of each term in the model formula on the linear predictor scale.
`terms`	a character vector of term labels, specifying a subset of component functions to be used to make predictions.
`...`	not used.
`term`	a character string specifying the component function of a fitted MID model.
`x`	a matrix, data frame or vector to be used as the input to the first argument of the component function. If a matrix or data frame is passed, inputs for both `x` and `y` are extracted from it.
`y`	a vector to be used as the input to the second argument of the component function.

Details

The S3 method of `predict()` for MID models returns the model predictions. `mid.f()` works as a component function of a MID model.

Value

`predict.mid()` returns a numeric vector of MID model predictions.

Examples

```
data(trees, package = "datasets")
idx <- c(5L, 10L, 15L, 20L, 25L, 30L)
mid <- interpret(Volume ~ .^2, trees[-idx,], lambda = 1)
trees[idx, "Volume"]
predict(mid, trees[idx,])
predict(mid, trees[idx,], type = "terms")
mid.f(mid, "Girth", trees[idx,])
mid.f(mid, "Girth:Height", trees[idx,])
predict(mid, trees[idx,], terms = c("Girth", "Height"))
```

---

print.mid *Print MID Models*

Description

For "mid" objects, `print()` prints the MID values and the uninterpreted rate.

Usage

```
## S3 method for class 'mid'
print(x, digits = max(3L, getOption("digits") - 2L), main.effects = FALSE, ...)
```

Arguments

`x`	a "mid" object to be printed.
`digits`	an integer specifying the number of significant digits.
`main.effects`	logical. If `TRUE`, MID values of the main effects are printed.
`...`	not used.

Details

The S3 method of `print()` for "mid" objects prints the MID values of a fitted MID model and its uninterpreted rate.

Value

`print.mid()` returns the "mid" object passed to the function without any modification.

Examples

```
data(cars, package = "datasets")
print(interpret(dist ~ speed, cars))
```

scale_color_theme *Color Scales for ggplot2 Graphics based on Color Themes*

Description

scale_color_theme() and family functions returns color scales for the "colour" and "fill" aesthetics of ggplot objects.

Usage

```
scale_color_theme(
  theme,
  ...,
  discrete = NULL,
  middle = 0,
  aesthetics = "colour"
)

scale_colour_theme(
  theme,
  ...,
  discrete = NULL,
  middle = 0,
  aesthetics = "colour"
)

scale_fill_theme(theme, ..., discrete = NULL, middle = 0, aesthetics = "fill")
```

Arguments

theme	one of the following: a color theme name such as "Viridis", a character vector of color names, a palette function, or a ramp function to be used to create a color theme.
...	optional arguments to be passed to ggplot2::continuous_scale() or ggplot2::discrete_scale().
discrete	logical. If TRUE, a discrete scale is returned.
middle	a numeric value specifying the middle point for the diverging color themes.
aesthetics	character string: "fill" or "color".

Value

scale_color_theme() returns a "ScaleContinuous" or "ScaleDiscrete" object that can be added to a "ggplot" object.

Examples

```
data(txhousing, package = "ggplot2")
cities <- c("Houston", "Fort Worth", "San Antonio", "Dallas", "Austin")
df <- subset(txhousing, city %in% cities)
d <- ggplot2::ggplot(data = df, ggplot2::aes(x = sales, y = median)) +
  ggplot2::geom_point(ggplot2::aes(colour = city))
d + scale_color_theme("Set 1")
d + scale_color_theme("R3")
d + scale_color_theme("Blues", discrete = TRUE)
d + scale_color_theme("SunsetDark", discrete = TRUE)
data(faithfuld, package = "ggplot2")
v <- ggplot2::ggplot(faithfuld) +
  ggplot2::geom_tile(ggplot2::aes(waiting, eruptions, fill = density))
v + scale_fill_theme("Plasma")
v + scale_fill_theme("Spectral")
v + scale_fill_theme("Spectral_r")
v + scale_fill_theme("midr", middle = 0.017)
```

---

shapviz.mid *Calculate SHAP of MID Predictions*

Description

`shapviz.mid()` is a S3 method of `shapviz::shapviz()` for the fitted MID models.

Usage

```
## S3 method for class 'mid'
shapviz(object, data = NULL)
```

Arguments

`object`	a "mid" object.
`data`	a data frame containing observations for which SHAP values are calculated. If not passed, data is extracted from `parent.env()` based on the function call of the "mid" object.

Details

The S3 method of `shapviz()` for the "mid" objects returns an object of class "shapviz" to be used to create SHAP plots with the functions of the shapviz package such as `sv_waterfall()` and `sv_importance()`.

Value

`shapviz.mid()` returns an object of class "shapviz".

summary.mid *Summarize MID Models*

Description

For "mid" objects, summary() prints information about the fitted MID model.

Usage

```
## S3 method for class 'mid'
summary(object, digits = max(3L, getOption("digits") - 2L), top.n = 10L, ...)
```

Arguments

object	a "mid" object to be summarized.
digits	an integer specifying the number of significant digits.
top.n	an integer specifying the maximum number of terms to be printed with the MID importance values.
...	not used.

Details

The S3 method of summary() for "mid" objects prints basic information about the MID model including the uninterpreted variation ratio, residuals, encoding schemes, and MID importance.

Value

summary.mid() returns the "mid" object passed to the function without any modification.

Examples

```
data(cars, package = "datasets")
summary(interpret(dist ~ speed, cars))
```

theme_midr *Theme for ggplot Objects*

Description

theme_midr() returns a complete theme for "ggplot" objects. par.midr() can be used to set graphical parameters at the package default.

Usage

```
theme_midr(
  grid_type = c("none", "x", "y", "xy"),
  base_size = 11,
  base_family = "serif",
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)

par.midr(...)
```

Arguments

grid_type	one of "none", "x", "y" or "xy".
base_size	base font size, given in pts.
base_family	base font family.
base_line_size	base size for line elements.
base_rect_size	base size for rect elements.
...	optional arguments in tag = value form to be passed to graphics::par().

Value

theme_midr() provides a ggplot2 theme customized for the midr package. par.midr() returns the previous values of the changed parameters in an invisible named list.

Examples

```
X <- data.frame(x = 1:10, y = 1:10)
ggplot2::ggplot(X) +
  ggplot2::geom_point(ggplot2::aes(x, y)) +
  theme_midr()
ggplot2::ggplot(X) +
  ggplot2::geom_col(ggplot2::aes(x, y)) +
  theme_midr(grid_type = "y")
ggplot2::ggplot(X) +
  ggplot2::geom_line(ggplot2::aes(x, y)) +
  theme_midr(grid_type = "xy")
old.par <- par.midr()
plot(y ~ x, data = X)
plot(y ~ x, data = X, type = "l")
plot(y ~ x, data = X, type = "h")
par(old.par)
```

weighted *Weighted Data Frames*

Description

`weighted()` returns a data frame with sample weights.

Usage

```
weighted(data, weights = NULL)

augmented(data, weights = NULL, size = nrow(data), r = 0.01)

shuffled(data, weights = NULL, size = nrow(data))

latticized(
  data,
  weights = NULL,
  k = 10L,
  type = 0L,
  use.catchall = TRUE,
  catchall = "(others)",
  frames = list(),
  keep.mean = TRUE
)

## S3 method for class 'weighted'
weights(object, ...)
```

Arguments

`data`	a data frame.
`weights`	a numeric vector of sample weights for each observation in `data`.
`size`	integer. The number of random observations whose values are sampled from the marginal distribution of each variable.
`r`	a numeric value specifying the ratio of the total weights for the random observations to the sum of sample weights. The weight for the random observations is calculated as `sum(attr(data, "weights")) * r / size`.
`k`	integer. The maximum number of sample points for each variable. If not positive, all unique values are used as sample points.
`type`	integer. The type of encoding of quantitative variables to be passed to `numeric.encoder()`.
`use.catchall`	logical. If `TRUE`, less frequent levels of factor variables are dropped and replaced by the catchall level.
`catchall`	a character string to be used as the catchall level.
`frames`	a named list of encoding frames ("numeric.frame" or "factor.frame" objects).

keep.mean	logical. If TRUE, the representative values of each group is the average of the corresponding group.
object	a data frame with the attribute "weights".
...	not used.

Details

weighted() returns a data frame with the "weights" attribute that can be extracted using stats::weights(). augmented(), shuffled() and latticized() return a weighted data frame with some data modifications. These functions are designed for use with interpret(). As the modified data frames do not preserve the original correlation structure of the variables, the response variable (y) should always be replaced by the model predictions (yhat).

Value

weighted() returns a data frame with the attribute "weights". augmented() returns a weighted data frame of the original data and the shuffled data with relatively small weights. shuffled() returns a weighted data frame of the shuffled data. latticized() returns a weighted data frame of latticized data, whose values are grouped and replaced by the representative value of the corresponding group.

Examples

```
set.seed(42)
x1 <- runif(1000L, -1, 1)
x2 <- x1 + runif(1000L, -1, 1)
weights <- (abs(x1) + abs(x2)) / 2
x <- data.frame(x1, x2)
xw <- weighted(x, weights)
ggplot2::ggplot(xw, ggplot2::aes(x1, x2, alpha = weights(xw))) +
  ggplot2::geom_point() +
  ggplot2::ggtitle("weighted")
xs <- shuffled(xw)
ggplot2::ggplot(xs, ggplot2::aes(x1, x2, alpha = weights(xs))) +
  ggplot2::geom_point() +
  ggplot2::ggtitle("shuffled")
xa <- augmented(xw)
ggplot2::ggplot(xa, ggplot2::aes(x1, x2, alpha = weights(xa))) +
  ggplot2::geom_point() +
  ggplot2::ggtitle("augmented")
xl <- latticized(xw)
ggplot2::ggplot(xl, ggplot2::aes(x1, x2, size = weights(xl))) +
  ggplot2::geom_point() +
  ggplot2::ggtitle("latticized")
```

weighted.mse *Weighted Loss Functions*

Description

weighted.mse(), weighted.rmse(), weighted.mae() and weighted.medae() compute the loss based on the differences of two numeric vectors or deviations from the mean of a numeric vector.

Usage

```
weighted.mse(x, y = NULL, w = NULL, ..., na.rm = FALSE)

weighted.rmse(x, y = NULL, w = NULL, ..., na.rm = FALSE)

weighted.mae(x, y = NULL, w = NULL, ..., na.rm = FALSE)

weighted.medae(x, y = NULL, w = NULL, ..., na.rm = FALSE)
```

Arguments

x	a numeric vector.
y	an optional numeric vector. If passed, the loss is calculated for the differences between x and y. If not, the loss is calculated for the deviations of x from the weighted mean of itself.
w	a numeric vector of sample weights for each value in x.
...	optional parameters.
na.rm	logical. If TRUE, any NA and NaNs are removed from x before the calculation.

Details

weighted.mse() returns the mean square error, weighted.rmse() returns the root mean square error, weighted.mae() returns the mean absolute error, and weighted.medae() returns the median absolute error between two weighted vectors x and y. If y is not passed, these functions return the corresponding statistic based on the deviations from the mean of x.

Value

weighted.mse() (mean square error), weighted.rmse() (root mean square error), weighted.mae() (mean absolute error) and weighted.medae (median absolute error) returns a single numeric value.

Examples

```
weighted.rmse(x = c(0, 10), y = c(0, 0), w = c(99, 1))
weighted.mae(x = c(0, 10), y = c(0, 0), w = c(99, 1))
weighted.medae(x = c(0, 10), y = c(0, 0), w = c(99, 1))
# compute uninterpreted rate
mid <- interpret(dist ~ speed, cars)
```

```
weighted.mse(cars$dist, predict(mid, cars)) / weighted.mse(cars$dist)
mid$ratio
```

`weighted.quantile` *Weighted Sample Quantile*

Description

`weighted.quantile()` produces weighted sample quantiles corresponding to the given probabilities.

Usage

```
weighted.quantile(
  x,
  w = NULL,
  probs = seq(0, 1, 0.25),
  na.rm = FALSE,
  names = TRUE,
  digits = 7L,
  type = 1L,
  ...
)
```

Arguments

`x`	a numeric vector whose weighted sample quantiles are wanted.
`w`	a numeric vector of the sample weights for each value in `x`.
`probs`	a numeric vector of probabilities with values in `[0, 1]`.
`na.rm`	logical. If `TRUE`, any `NA` and `NaN`s are removed from `x` before the quantiles are computed.
`names`	logical. If `TRUE`, the result has a "names" attribute.
`digits`	used only when `names` is `TRUE`. The precision to use when formatting the percentages.
`type`	an integer between `1` and `9` selecting the quantile algorithms. Only `1` is available for the weighted quantile.
`...`	further arguments passed to `stats::quantile()` when the weights is not passed.

Details

`weighted.quantile()` is a wrapper function of `stats::quantile()` for weighted quantiles. For the weighted quantile, only the "type 1" quantile, the inverse of the empirical distribution function, is available.

Value

`weighted.quantile()` returns weighted sample quantiles corresponding to the given probabilities.

Examples

```
stats::quantile(x = 1:10, type = 1L, probs = c(0, .25, .50, .75, 1))
weighted.quantile(x = 1:10, w = 1:10, probs = c(0, .25, .50, .75, 1))
```

---

weighted.tabulate *Weighted Tabulation for Vectors*

Description

`weighted.tabulate()` returns the sum of weights for each integer in the vector `bin`.

Usage

```
weighted.tabulate(bin, w = NULL, nbins = max(1L, bin, na.rm = TRUE))
```

Arguments

`bin`	a numeric vector of positive integers, or a factor.
`w`	a numeric vector of the sample weights for each value in `bin`.
`nbins`	the number of bins to be used.

Details

`weighted.tabulate()` is a wrapper function of `tabulate()` to reflect sample weights.

Value

`weighted.tabulate()` returns an numeric vector.

Examples

```
tabulate(bin = c(2, 2, 3, 5))
weighted.tabulate(bin = c(2, 2, 3, 5), w = 1:4)
```

Index