Package 'midr'

December 23, 2024

Type Package

Title Create an Interpretable Surrogate of Black-Box ML Models

Version 0.3.3

Description Construct a globally interpretable surrogate of a black-box ML model by decomposing the prediction function of the target model into a set of component functions including the intercept, the main effects and the second-order interactions.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports ggplot2,
graphics,
grDevices,
RcppEigen,
rlang,
stats,
utils

Suggests datasets,
DALEX,
e1071,
gridExtra,
ISLR2,
kernlab,
knitr,
ranger,
rmarkdown,
testthat

Config/testthat/edition 3

RoxygenNote 7.3.1

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

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

R topics documented:

`factor.encoder` *Encoder for Qualitative Variables*

Description

Returns a list consisting of the encoding information of a predictor variable as 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 max number of the levels after the encoding. If not positive, all unique values of x are chosen as sample points.
`use.catchall`	logical. If TRUE, less frequent levels are dropped and replaced with the catchall level.

catchall	a character used as the name of "catchall" level for unused levels.
tag	name of the variable.
frame	a factor.frame object or a vector, containing the information about the levels of the variable.
weights	optional. a numeric vector indicating the weight of each value of 'x'.
levels	a vector to be used as the levels of the variable.

Value

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

frame	a data frame containing the encoding information.
encode	a function to encode new data into a dummy matrix.
n	the number of encoding levels.
type	type of encoding.

Examples

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

---

get.yhat *Get Predicted Values from Fitted Models*

Description

Returns the predicted values or the target class probabilities of a predictive model for the newdata.

Usage

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

## Default S3 method:
get.yhat(X.model, newdata, ...)

## 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, ...)

## S3 method for class 'randomForest'
```

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

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

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

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

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

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

Arguments

X.model	a model object to be interpreted.
newdata	a data.frame or a matrix.
...	other parameters that will be passed to the predict function.

Value

`get.yhat()` returns a numeric vector of model predictions.

Examples

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

---

ggmid *Plot MID Values with ggplot2 Package*

Description

Creates a ggplot object representing mid values of the functional decomposition term.

Usage

```
ggmid(object, ...)

## S3 method for class 'mid'
ggmid(
  object,
  term,
  limits = c(NA, NA),
  plot.main = TRUE,
```

```
  add.intercept = FALSE,
  include.main.effects = FALSE,
  interaction.type = c("default", "raster", "rectangle"),
  scale.type = "default",
  scale.palette = c("#2f7a9a", "#FFFFFF", "#7e1952"),
  partition = 100L,
  ...
)

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

Arguments

object	a mid object to be visualized.
...	optional parameters to be passed to the main layer (geom_line, geom_path, geom_bar, or geom_rect) of each plot.
term	a name of the functional decomposition term to be plotted.
limits	NULL or a numeric vector of length two providing limits of the scale. NA is replaced by the minimum or maximum mid value.
plot.main	logical. If TRUE, lines, bars or rectangles representing mid values are drawn.
add.intercept	logical. If TRUE, the intercept is added to the mid values and the scale for the plot is shifted.
include.main.effects	logical. If TRUE, the main effects are added to the interaction mid values.
interaction.type	a character, specifying the plotting method of interaction effects.
scale.type	color type of interaction plots. One of "default", "viridis", "gradient" or a function that returns a continuous colour scale for fill aestetics like ggplot2::scale_fill_viridis_c.
scale.palette	a character vector of color names, specifying the colors to be used in the interaction plot.
partition	an integer specifying the coarseness of the grid for a "raster" type interaction plot.

Value

ggmid() returns a ggplot object.

Examples

```
data(diamonds, package = "ggplot2")
model <- lm(price ~ carat + cut + color + clarity + carat:clarity, diamonds)
mid <- interpret(price ~ carat + cut + color + clarity + carat:clarity,
                 data = diamonds, model = model)
ggmid(mid, "carat")
ggplot2::autoplot(mid, "clarity")
ggmid(mid, "carat:clarity")
ggmid(mid, "carat:clarity", add.intercept = TRUE,
      include.main.effects = TRUE, scale.type = "viridis")
```

`ggmid.mid.conditional` *Plot MID Individual Conditional Expectations with ggplot2 Package*

Description

Creates a ggplot object representing the MID-based individual conditional expectations

Usage

```
## S3 method for class 'mid.conditional'
ggmid(
  object,
  limits = c(NA, NA),
  plot.main = TRUE,
  centered = FALSE,
  draw.dots = TRUE,
  sample = NULL,
  term = NULL,
  variable.alpha = NULL,
  variable.colour = NULL,
  variable.linetype = NULL,
  variable.linewidth = NULL,
  ...
)

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

Arguments

`object`	a mid.conditional object to visualize.
`limits`	NULL or a numeric vector of length two providing limits of the scale. NA is replaced by the minimum or maximum mid value.
`plot.main`	logical. If TRUE, lines representing the individual conditional expectations are drawn.
`centered`	logical.
`draw.dots`	logical. If TRUE, points representing the predictions at the observed values are
`sample`	a vector specifying the set of names of the observations to be plotted.
`term`	an optional character specifying one of the relevant terms. If passed, the individual conditional expectations for the specified term are plotted.
`variable.alpha`	a name of the predictor variable to use to set `alpha` for each plot.
`variable.colour`	a name of the predictor variable to use to set `color` for each plot.
`variable.linetype`	a name of the predictor variable to use to set `linetype` for each plot.
`variable.linewidth`	a name of the predictor variable to use to set `linewidth` for each plot.
`...`	optional parameters to be directly passed to `ggplot2::geom_line()`.

Value

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

Examples

```
data(airquality, package = "datasets")
mid <- interpret(Ozone ~ .^2, airquality, lambda = 1)
mc <- mid.conditional(mid, "Wind", airquality)
ggmid(mc, variable.colour = "Solar.R", centered = TRUE)
```

---

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

Description

Creates a ggplot object representing the MID-based term importance

Usage

```
## S3 method for class 'mid.importance'
ggmid(
  object,
  type = c("barplot", "heatmap"),
  plot.main = TRUE,
  max.terms = NA,
  scale.palette = c("#FFFFFF", "#464646"),
  ...
)

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

Arguments

object	a mid.importance object to be visualized.
type	a character or an integer, specifying the type of the plot. Possible alternatives are "barplot" and "heatmap".
plot.main	logical. If TRUE, lines, bars or rectangles representing mid values are drawn.
max.terms	an integer, specifying the maximum number of component terms to be plotted in the barplot.
scale.palette	color palette used to draw the interaction heatmap.
...	optional arguments to be passed to graphic functions.

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)
ggmid(imp, type = "heatmap")
```

`interpret` *Create an Interpretable Surrogate of Black-Box ML Models*

Description

Construct a predictive model consisting of a set of 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"),
  encoding.digits = 3L,
  use.catchall = FALSE,
  catchall = "(others)",
  max.ncol = 3000L,
  nil = 1e-07,
  tol = 1e-07,
  ...
)

## S3 method for class 'formula'
interpret(
  formula,
```

```
  data = NULL,
  model = NULL,
  pred.fun = get.yhat,
  weights = NULL,
  subset = NULL,
  na.action = getOption("na.action"),
  mode = 1L,
  drop.unused.levels = FALSE,
  ...
)
```

Arguments

object	a fitted model object to be interpreted.
...	special aliases for some arguments, including 'ie' for 'interaction' and 'ok' for 'singular.ok'.
x	a matrix or a data frame of predictor variables to be used to interpret the model prediction. The response variable should not be included.
y	an optional numeric vector representing of the model prediction or the response variable.
weights	optional. a numeric vector indicating weights for each row in the data.
pred.fun	a function that takes two arguments, 'X.model' and 'newdata' to be used to make a prediction. Default is `get.yhat()`, which uses `DALEX::yhat()` if the DALEX package is installed.
link	name of the link function. One of "logit", "probit", "cauchit", "cloglog", "identity", "log", "sqrt", "1/mu^2", "inverse".
k	an integer or a numeric vector of length two for main effects and interactions, specifying the maximum number of sample points for each numeric predictor variable. If an integer is passed, k is used for main effect terms and the square root of k is used for interaction terms. If not positive, all unique values are used as sample points.
type	an integer or a vector of length two, specifying the type of piecewise functions to be fit on numeric variables. '0' is for step functions on discretized intervals, and '1' is for piecewise linear functions connecting at representative values.
frames	a named list of encoding frames, which specifies bins for quantitative features or levels for qualitative features.
interaction	logical. If TRUE, and if `terms` and `formula` are not supplied, all second order interaction effects for each pair of features in `x` are calculated.
terms	a character vector of term labels, specifying the set of decomposition terms. If not passed, all main effects (and all second order interactions if `interaction` is TRUE) of `x` are used.
singular.ok	logical. If FALSE, a singular fit is an error.
mode	an integer specifying the general method of calculation. When `mode` is set to 1, centralization constraints are treated as penalties for the least squares problem. If `mode` is 2, the centralization constraints are used to reduce the number of unknown parameters, making the calculation safer and more robust.
method	an integer or a vector of length two, specifying the methods to be used to solve the least squares problem. A non-negative value will be passed to RcppEigen::fastLmPure and if a negative value is passed, stats::lm.fit will be used.

lambda	a numeric parameter for the penalty of weighted ridge regularization.
kappa	a numeric parameter for the penalty of the centralization constraints. Only used if mode is 1.
na.action	a function or a character which indicates what should happen when the data contain missing values (NAs). The default is na.omit.
encoding.digits	an integer specifying the rounding digits for encoding numeric variables when type is 1 (piecewise linear functions).
use.catchall	logical. If TRUE, less frequent levels are dropped and replaced with the catchall level.
catchall	a character used as the name of "catchall" level for unused levels of each factor variable.
max.ncol	an integer which indicates the maximum number of columns of the design matrix.
nil	a threshold for the intercept and coefficients to be treated as zero. Default is 1e-7.
tol	a tolerance for the singular value decomposition. Default is 1e-7.
formula	a symbolic description of the decomposition model to be fit.
data	a data frame containing the variables in the formula. If not found in data, the variables are taken from environment(formula).
model	a model object to be interpreted.
subset	an index vector specifying the rows to be used in the training sample.
drop.unused.levels	logical. If TRUE, unused levels of factors will be dropped.

Value

interpret() returns an object of class "mid", which is a list containing the following components:

weights	a numeric vector of the weights.
call	the matched call.
terms	a character vector of decomposition term names.
link	a list of class "mid-link", specifying the link function used.
intercept	the fitted zeroth-order effect.
main.effects	a list of data frames representing the fitted first-order (main) effects of each variable.
me.encoders	a list of encoders for the first-order decomposition.
interacions	a list of data frames representing the fitted second-order interactions.
ie.encoders	a list of encoders for the second-order decomposition.
uninterpreted.rate	the ratio of the interpretation loss to the original variance of model predictions (yhat).
fitted.matrix	a matrix containing the breakdown of the fitted values into the functional decomposition terms.
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 on the special handlings of NAs.

Examples

```
data(cars, package = "datasets")
model <- lm(dist ~ I(speed^2) + speed, cars)
mid <- interpret(dist ~ speed, cars, model)
plot(mid, "speed", add.intercept = TRUE) +
  points(cars)
summary(mid)

data(Nile, package = "datasets")
mid <- interpret(x = 1L:100L, y = Nile, k = 100L)
plot(mid, "x", add.intercept = TRUE, ylim = c(600L, 1300L)) +
  points(x = 1L:100L, y = Nile)
# reduce number of knots by k parameter
mid <- interpret(x = 1L:100L, y = Nile, k = 10L)
plot(mid, "x", add.intercept = TRUE, ylim = c(600L, 1300L)) +
  points(x = 1L:100L, y = Nile)
# pseudo-smoothing by lambda parameter
mid <- interpret(x = 1L:100L, y = Nile, k = 100L, lambda = 100L)
plot(mid, "x", add.intercept = TRUE, ylim = c(600L, 1300L)) +
  points(x = 1L:100L, y = Nile)

data(airquality, package = "datasets")
airquality$Month <- factor(airquality$Month)
model <- glm(Ozone ~ .^2, Gamma(log), airquality)
mid <- interpret(Ozone ~ .^2, na.omit(airquality), model, lambda = .1)
summary(mid)
plot(mid, "Wind")
plot(mid, "Temp")
plot(mid, "Wind:Month", include.main.effects = TRUE)
```

---

mid.conditional *Calculate and Visualize MID-based Individual Conditional Expectation*

Description

Creates a data frame to be used to visualize the individual conditional expectation.

Usage

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

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

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

Arguments

object	a mid object to compute the individual conditional expectations.
variable	a character or an expression specifying the predictor variable to calculate the individual conditional expectations for.
data	a data frame representing the observation results.
keep.effects	logical. If TRUE, the effects of component terms are stored in the output object.
partition	an integer specifying the number of the sample values.
max.nrow	the maximum number of rows of the output data frame.
type	the type of the prediction to use when the model has a link function. The default is response.
x	a mid.conditional object to print.
...	additional arguments to be passed to the methods for data.frame.

Value

mid.conditional() returns a 'mid.conditional' object that contains the following components:

terms	a character vector of relevant terms.
observed	a data frame of the observations and the corresponding predictions.
conditional	a data frame of the hypothetical observations.
values	a numeric vector of the representative values of the target variable.

Examples

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

---

mid.extract *Extract Information from a mid Object*

Description

Returns information on the mid object including a summary for encoders, uninterpreted rates, or other special components.

Usage

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

mid.encoding.info(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, ...)
```

Arguments

object	a mid object.
component	a literal character string that specifies the name of the component to extract. Any component of the mid object can be extracted. In addition, some special values such as "frames", "encoding.info", and names of component functions like "x1:x2" can be used.
...	optional arguments to be passed to the specific functions used to extract information.
main.effect	logical. If FALSE, all main effects are excluded.
interaction	logical. If FALSE, all interactions are excluded.
require	a character vector of feature names. Only terms related to at least one of the specified features will be returned.
remove	a character vector of feature names. All terms related to at least one of the specified features will not be returned.
x	a mid object or a mid.importance object.

Value

`mid.extract()` returns the 'component' extracted from the mid object. `mid.encoding.info()` returns basic information of the encoders. `mid.frames()` returns encoding frames of each feature. `mid.terms()` returns a character vector of term names.

Examples

```
data(trees, package = "datasets")
mid <- interpret(Volume ~ ., trees, k = 10)
mid.extract(mid, encoding.info)
mid.extract(mid, uninterpreted.rate)
```

```
mid.extract(mid, frames)
mid.extract(mid, Girth)
mid.extract(mid, intercept)
```

`mid.importance` *Calculate and Visualize MID-based Importance and Breakdown*

Description

Creates a data frame showing the importance of each functional decomposition term. `mid.importance()` returns a object of class "mid.importance", for which methods for `ggplot2::autoplot()` and `graphics::barplot()` are defined.

Usage

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

Arguments

`object`	a mid object.
`data`	data to be used to calculate the importance. If NULL, the fitted matrix is extracted from the mid object. If the number of row equals to one, the mid breakdown will be calculated.
`weights`	a numeric vector of weights.
`sort`	logical. If TRUE, the data.frame will be sorted by magnitude of importance
`measure`	an integer specifying the type of function to evaluate the importance of each effect. Possible values are "1" for the mean absolute effect, "2" for the root mean square effect, and "3" for the median absolute effect.

Value

`mid.importance` returns a data frame for the term importance, measured as the average magnitude of the effect.

Examples

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

mid.plots *Create Plots for Multiple Functional Decomposition Terms*

Description

Returns a list of ggplot objects, each representing the mid values for the corresponding functional decomposition term.

Usage

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

Arguments

object	a mid object.
terms	a character vector that specifies the names of the terms to be visualized.
limits	NULL or a numeric vector of length two providing limits of the scale. NA will be replaced by the minimum or maximum mid value in all terms.
add.intercept	logical. If TRUE, the intercept is added to the mid values and the scale for the plot is shifted.
include.main.effects	logical. If TRUE, the main effects are added to the interaction mid values.
max.plots	the number of maximum number of plots.
engine	a name of the package used to create plots. Possible values are "ggplot2" and "base" ("graphics").
...	optional parameters to be passed to ggmid() function.

Value

If engine is 'ggplot2', `mid.plots()` returns a list of ggplot objects, otherwise `mid.plots()` creates plots and returns nothing.

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

Returns a list consisting of the encoding information of a predictor variable as 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"
)
```

Arguments

x	a numeric vector which is to be encoded.
k	an integer specifying the coarseness of the encoding. If not positive, all unique values of x are chosen as sample points.
type	an integer specifying the shape of the function to be fit. 1 is for a piecewise linear function and 0 is for a piecewise constant (step) function.
encoding.digits	an integer specifying the rounding digits for encoding numeric variables when type is 1 (piecewise linear functions).
tag	name of the corresponding predictor variable.
frame	a numeric.frame object containing the information about the binning of the variable.
weights	optional. a numeric vector indicating the weight of each value of 'x'.
reps	a numeric vector specifying the representative values of each bin.
breaks	a numeric vector to be used as the breaks of the binning.

Value

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

`frame`	a data frame containing the encoding information.
`encode`	a function to encode new data into a dummy matrix.
`n`	the number of encoding levels.
`type`	type of encoding: 'linear' (first degree) or 'constant' (zeroth degree).

Examples

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

---

plot.mid *Plot MID Values with graphics Package*

Description

Creates a plot showing mid values of the functional decomposition term.

Usage

```
## S3 method for class 'mid'
plot(
  x,
  term,
  add.intercept = FALSE,
  include.main.effects = FALSE,
  scale.type = "default",
  scale.palette = c("#2f7a9a", "#FFFFFF", "#7e1952"),
  m = 100L,
  ...
)
```

Arguments

`x`	mid object to be visualized.
`term`	name of term to be plotted.
`add.intercept`	logical. If TRUE, the intercept is added to the mid values and the scale for the plot is shifted.
`include.main.effects`	logical. If TRUE, the main effects are added to the interaction mid values.
`scale.type`	color type of interaction plots. One of "default", "viridis", "gradient" or a function that returns a continuous colour scale for `fill` aestetics like `ggplot2::scale_fill_viridis_c`.
`scale.palette`	a character vector of color names, specifying the colors to be used in the interaction plot.
`m`	an integer specifying the coarseness of the grid for a interaction plot.
`...`	optional parameters to be passed to `plot()`, `barplot()` or `filled.contour()`.

Value

`plot.mid()` produces a line plot or a bar plot for the main effect and a filled contour plot for the interaction.

Examples

```
data(airquality, package = "datasets")
airquality$Month <- factor(airquality$Month)
mid <- interpret(Ozone ~ .^2, airquality, lambda = 1)
plot(mid, "Temp")
plot(mid, "Month")
plot(mid, "Wind:Temp")
plot(mid, "Solar.R:Month", scale.type = "viridis",
     add.intercept = TRUE, include.main.effects = TRUE)
```

---

plot.mid.conditional *Plot MID Individual Conditional Expectations with graphics Package*

Description

Creates a plot showing the MID-based individual conditional expectations

Usage

```
## S3 method for class 'mid.conditional'
plot(
  x,
  centered = FALSE,
  draw.dots = TRUE,
  sample = NULL,
  term = NULL,
  variable.alpha = NULL,
  variable.colour = NULL,
  variable.linetype = NULL,
  variable.linewidth = NULL,
  scale.palette = NULL,
  ...
)
```

Arguments

`x`	a mid.conditional object to visualize.
`centered`	logical.
`draw.dots`	logical. If TRUE, points representing the predictions at the observed values are
`sample`	a vector specifying the set of names of the observations to be plotted.
`term`	an optional character specifying one of the relevant terms. If passed, the individual conditional expectations for the specified term are plotted.
`variable.alpha`	a name of the predictor variable to use to set `alpha` for each plot.
`variable.colour`	a name of the predictor variable to use to set `color` for each plot.

variable.linetype
: a name of the predictor variable to use to set linetype for each plot.

variable.linewidth
: a name of the predictor variable to use to set linewidth for each plot.

scale.palette
: a character vector of color names, specifying the colors to be used.

...
: optional parameters to be directly passed to ggplot2::geom_line().

Value

plot.mid.conditional() produces an ICE plot for the conditional effect of the target variable.

Examples

```
data(airquality, package = "datasets")
mid <- interpret(Ozone ~ .^2, airquality, lambda = 1)
mc <- mid.conditional(mid, "Wind", na.omit(airquality))
plot(mc, variable.colour = "Solar.R", centered = TRUE)
```

---

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

Description

Creates a plot showing the MID-based term importance

Usage

```
## S3 method for class 'mid.importance'
plot(
  x,
  type = c("barplot", "heatmap"),
  max.terms = NA,
  scale.palette = c("#FFFFFF", "#464646"),
  ...
)
```

Arguments

x
: a mid.importance object to plot.

type
: a character or an integer, specifying the type of the plot. Possible alternatives are "barplot" and "heatmap".

max.terms
: an integer, specifying the maximum number of component terms to be plotted in the barplot.

scale.palette
: color palette used to draw the interaction heatmap.

...
: optional arguments to be passed to graphic functions.

Value

plot.mid.importance() produces a bar plot or a heat map for the term importance.

Examples

```
data(diamonds, package = "ggplot2")
set.seed(42)
mid <- interpret(price ~ (carat + cut + color + clarity)^2,
                 diamonds[sample(nrow(diamonds), 1e4), ])
imp <- mid.importance(mid)
ggmid(imp)
ggmid(imp, type = "heatmap")
```

`predict.mid` *Predict Method for MID-based Surrogate Models*

Description

Returns predictions of the fitted mid object and optionally breakdown of those predictions into the functional decomposition terms.

Usage

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

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

Arguments

`object`	a mid object to be used as a surrogate model.
`newdata`	data frame for which the predictions are to be made.
`na.action`	a function which indicates what should happen when the data contain missing values (NAs).
`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, specifying names of the terms to be used to make predictions.
`...`	not used.
`term`	a character, specifying the name of the term of the decomposed function.
`x`	a vector to be used as inputs to the first argument of the decomposed function. If a matrix or data.frame is passed, inputs are extracted from the matrix or data.frame.
`y`	a vector to be used as inputs to the second argument of the decomposed interaction function.

Value

predict.mid() returns a numeric vector of MID model predictions. mid.f() works as a component function of the MID model and returns a term effects.

Examples

```
data(cars, package = "datasets")
mid <- interpret(dist ~ speed, cars, lambda = 1)
predict(mid, newdata = data.frame(speed = 5:25))
mid.f(mid, "speed", 5:25) + mid$intercept
```

---

print.mid *Print Method for MID Objects*

Description

Prints the summary of the fitted mid object.

Usage

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

Arguments

x	a mid object to be printed.
digits	number of significant digits.
omit.values	logical. If TRUE, mid values of main effect terms are not printed.
...	not used.

Value

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

Examples

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

summary.mid *Summary Method for MID Objects*

Description

Prints the summary of the fitted mid object.

Usage

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

Arguments

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

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 *Themes for ggplot Objects*

Description

Returns a complete theme for ggplot objects.

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
)
```

Arguments

grid_type	one of "none", "x", "y" or "both".
base_size	a positive value for the base font size, given in pts.
base_family	a character specifying the base font family.
base_line_size	a positive value for the base size for line elements.
base_rect_size	a positive value for the base size for rect elements.

Value

`theme_midr()` provides a ggplot2 theme customized for the `midr` package.

Examples

```
ggplot2::theme_set(theme_midr())
```

---

weighted *Weighted Data Frames*

Description

Returns a data frame of class `weighted` and `data.frame`, which contains the additional attribute "weights". Weights can be extracted using functions like `attr()` or `stats::weights()`.

Usage

```
weighted(data, weights = NULL, ...)

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

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

latticized(
  data,
  weights = NULL,
  k = 10L,
  type = 1L,
  use.catchall = TRUE,
  catchall = "(others)",
  frames = list(),
  keep.mean = TRUE
)

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

Arguments

data	a data frame for which weights are to be added.
weights	a numeric vector, specifying the weights of each record.
...	not used.
size	an integer, specifying the number of new rows to be added to the original data frame.
ratio	a numeric value. Weights for the new rows are calculated as `nrow(data) * ratio / size`.
k	an integer or a numeric vector of length two for main effects and interactions, specifying the maximum number of sample points for each numeric predictor variable. If an integer is passed, k is used for main effect terms and the square root of k is used for interaction terms. If not positive, all unique values are used as sample points.
type	an integer or a vector of length two, specifying the type of piecewise functions to be fit on numeric variables. '0' is for step functions on discretised intervals, and '1' is for piecewise linear functions connecting at representative values.
use.catchall	logical. If TRUE, less frequent levels of factor variables are dropped and replaced with the catchall level.
catchall	a character used as the name of "catchall" level for unused levels of each factor variable. Used only when `factor.option` is not 0.
frames	a named list of encoding frames, which specifies bins for quantitative features or levels for qualitative features.
keep.mean	logical. If TRUE, the mean is used
object	a weighted data frame.

Value

`weighted()` returns a data frame with the 'weights' attribute. `augmented()`, `shuffled()`, and `latticized()` returns a weighted data frame with the additional, randomized or simplified values.

Examples

```
x1 <- runif(1000L, -1, 1)
x2 <- x1 + runif(1000L, -1, 1)
X <- weighted(cbind(x1, x2), (abs(x1) + abs(x2)) / 2)
Y1 <- weighted(X)
ggplot2::ggplot(Y1, ggplot2::aes(x1, x2, alpha = weights(Y1))) +
  ggplot2::geom_point() +
  ggplot2::labs(title = "weighted") +
  ggplot2::theme_bw()
Y2 <- shuffled(X)
ggplot2::ggplot(Y2, ggplot2::aes(x1, x2, alpha = weights(Y2))) +
  ggplot2::geom_point() +
  ggplot2::labs(title = "shuffled") +
  ggplot2::theme_bw()
Y3 <- augmented(X)
ggplot2::ggplot(Y3, ggplot2::aes(x1, x2, alpha = weights(Y3))) +
  ggplot2::geom_point() +
  ggplot2::labs(title = "augmented") +
  ggplot2::theme_bw()
```

```
Y4 <- latticized(X)
ggplot2::ggplot(Y4, ggplot2::aes(x1, x2, size = weights(Y4))) +
  ggplot2::geom_point() +
  ggplot2::labs(title = "latticized") +
  ggplot2::theme_bw()
```

`weighted.quantile` *Weighted Sample Quantile*

Description

Returns weighted sample quantiles corresponding to the given probabilities. For the weighted quantiles, only "type 1" quantiles of `stats::quantile()` (the inverse of empirical distribution function) is available.

Usage

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

Arguments

`x`	an object containing the values whose weighted quantiles is to be computed.
`w`	a numeric vector of the same length as 'x' giving the weights to use for elements of it.
`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 selecting the quantile algorithms. Only 1 is available for the weighted quantile.
`...`	further arguments passed to `stats::quantile()` when the weights is not passed.

Value

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

Examples

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

weighted.rmse *Weighted Loss Functions*

Description

Returns a weighted loss calculated for the given vector(s). `weighted.rmse` is for the root mean square error, `weighted.mae` is for the mean absolute error, and `weighted.medae` is for the median absolute error

Usage

```
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 of deviations based on which the loss is calculated.
`y`	an optional numeric vector. If 'y' is passed, the loss is calculated based on the difference of x minus y.
`w`	a numeric vector of the same length as 'x' giving the weights to use for elements of it.
`...`	optional augments passed to other functions and methods.
`na.rm`	logical. If TRUE, any `NA` and `NaN`s are removed from 'x' before the quantiles are computed.

Value

`weighted.rmse()` (root mean squared error), `weighted.mae()` (mean absolute error) and `weighted.medae` (median absolute error) returns a weighted loss between two numeric vectors.

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))
```

weighted.tabulate *Weighted Tabulation for Vectors*

Description

Returns the sum of weights for each interger occurs in the integer valued argument `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 same length as 'bin' giving the weights to use for elements of it.
nbins	the number of bins to be used.

Value

`weighted.tabulate()` returns an integer valued vector.

Examples

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

Index