Package 'midr'

January 1, 2025

Type Package

Title Create an Interpretable Surrogate of Black-Box ML Models

Version 0.4.1

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, gridExtra, ISLR2, kernlab, knitr, nnet, 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

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 `new_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(new_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, ...)

## 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 'glmnet'
get.yhat(X.model, newdata, ...)

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

Arguments

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

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, "response")[1:5]
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,
  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"),
  cells.count = c(100L, 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.
term	a character string specifying the component function to be plotted.
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.
plot.main	logical. If FALSE, the main layer is not drawn.
add.intercept	logical. If TRUE, the intercept is added to the MID values.
include.main.effects	
	logical. If TRUE, the main effects are included in the interaction plot.
interaction.type	
	character string. The method for plotting the interaction effects.
scale.type	a character string or function specifying the color type of interaction plot. One of "default", "viridis", "gradient" or a function that returns a continuous color scale for fill aesthetics.
scale.palette	a character vector of color names. The colors are used for the interaction plot when scale.type is "default".
cells.count	an integer or integer-valued vector of length two, specifying the number of cells for the raster type interaction plot.

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) 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")
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 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,
  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 be visualized.
limits	NULL or a numeric vector of length two specifying the limits of the scale. NAs are replaced by the minimum and/or maximum MID values.
plot.main	logical. If FALSE, the main layer is not drawn.
centered	logical. If TRUE, the ICE values of each observation are set to zero at the leftmost point of the variable.

draw.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.

term   an optional character string specifying an interaction term. If passed, the ICE
                     curve for the specified term is plotted.

variable.alpha   a name of the variable to be used to set alpha.

variable.colour

                     a name of the variable to be used to set colour.

variable.linetype

                     a name of the variable to be used to set linetype.

variable.linewidth

                     a name of the variable to be used to set linewidth.

...   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")
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 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", "heatmap"),
  plot.main = TRUE,
  max.bars = 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 string specifying the type of the plot. One of "barplot" or "heatmap".
plot.main	logical. If TRUE, the main layer is not drawn.
max.bars	an integer specifying the maximum number of bars in the barplot.
scale.palette	a character vector of length two to be used as the color palette for the heatmap.
...	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() or geom_tile().

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   *Fit MID Models*

Description

interpret() is used to fit a MID model specifically as an interpretable surrogate for black-box ML 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"),
    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.
...	for interpret.default(), optional arguments including special aliases such as ok for singular.ok and ie for interaction. For interpret.formula(), optional parameters to be passed 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", and "inverse".
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 strength of the smoothing penalty. The default is 0.
kappa	the strength of the penalty for the centralization. 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".
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.
formula	a symbolic description of the MID 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 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

The prediction function of a fitted MID model $\hat{f}(X)$ has the following structure:

$$\hat{f}(X) = f_\phi + \Sigma_{j \in D} \, f_j(X_j) + \Sigma_{j,k \in D} \, f_{j,k}(X_j, X_k)$$

where, f_ϕ is the intercept, $f_j(X_j)$ is the main effect of the variable j, and $f_{j,k}(X_j, X_k)$ is the second-order interaction between the two variables j and k. The effects of quantitative variables are modeled as piecewise functions of degree 1 (piecewise linear function) or 0 (step function).

The MID values for each sample point are determined using the constrained least squares method. The loss function is $E[(\hat{Y} - \hat{f}(X))^2]$, where $\hat{Y}$ is the model prediction or the response variable, and the constraint functions are $E[f_j(X_j)] = 0$ for each variable j and $E[f_{j,k}(X_j, X_k)] = E[f_{j,k}(X_j, X_k)|X_j] = E[f_{j,k}(X_j, X_k)|X_k] = 0$ for each pair of variables (j, k).

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" 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.
uninterpreted.rate	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

```
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 the number of knots by setting the '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)
# perform a pseudo smoothing by setting the '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 ICE of MID Models*

Description

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

Usage

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

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

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.
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.

...   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, ...)
```

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 "uninterpreted.rate".
...	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.

Examples

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

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. If the `data` has only one observation, the output has the special class "mid.breakdown".
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.

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 a data frame of the class "mid.importance".

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),
  add.intercept = FALSE,
  include.main.effects = FALSE,
  max.plots = NULL,
  engine = c("ggplot2", "base", "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.
`add.intercept`	logical. If TRUE, the intercept is added to the MID values and the plotting scale is shifted.
`include.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"
)
```

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.

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 new_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(new_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(new_x = c(4:8, NA))

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

plot.mid   *Plot MID with Basic Functions*

Description

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

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"),
  cells.count = c(100L, 100L),
  ...
)
```

Arguments

x	a "mid" object to be visualized.
term	a character string specifying the component function to be plotted.
add.intercept	logical. If TRUE, the intercept is added to the MID values and the plotting scale is shifted.
include.main.effects	
	logical. If TRUE, the main effects are included in the interaction plot.
scale.type	character string. The color type of the interaction plot. One of "default", "viridis", "gradient".
scale.palette	a character vector of color names specifying the colors to be used in the interaction plot.
cells.count	an integer or integer-valued vector of length two specifying the number of cells for the raster type interaction plot.
...	optional parameters to be passed to the graphing function.

Details

The S3 method of `plot()` for "mid" objects creates a visualization of a MID component function using `base::plot()` for a main effect of a quantitative variable, `graphics::barplot()` for a main effect of a qualitative variable, and `graphics::filled.contour()` for an interaction.

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(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 ICE of MID Model with Basic Functions*

Description

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

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 be visualized.
centered	logical. If TRUE, the ICE values of each observation are set to zero at the leftmost point of the variable.
draw.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.
term	an optional character string specifying the interaction term. If passed, the ICE for the specified term is plotted.
variable.alpha	a name of the variable to be used to set `alpha`.
variable.colour	a name of the variable to be used to set `colour`.
variable.linetype	a name of the variable to be used to set `linetype`.

variable.linewidth
                 a name of the variable to be used to set `linewidth`.

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

...   optional parameters to be passed to `base::plot()`.

Details

The S3 method of `plot()` for "mid.conditional" objects creates an visualization of ICE curves of a fitted MID model using `base:plot()`.

Value

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

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 MID Importance with Basic Functions*

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", "heatmap"),
  max.bars = NA,
  scale.palette = c("#FFFFFF", "#464646"),
  ...
)
```

Arguments

x   a "mid.importance" object to be visualized.

type   a character string specifying the type of the plot. One of "barplot" or "heatmap".

max.bars   an integer specifying the maximum number of bars in the barplot.

scale.palette   a character vector of length two to be used as the color palette for the heatmap.

...   optional parameters to be passed to the graphing function.

Details

The S3 method of `plot()` for "mid.importance" objects creates a visualization of the MID importance using `graphics::barplot()` or `graphics::image()`.

Value

    `plot.mid.importance()` produces a barplot or heatmap 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)
plot(imp, type = "heatmap")
```

---

    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 = 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 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 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 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 func-
                          tion.

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), omit.values = FALSE, ...)
```

Arguments

x   a "mid" object to be printed.

digits   an integer specifying the number of significant digits.

omit.values   logical. If TRUE, MID values of the main effects are not 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))
```

---

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 rate, 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.

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 "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.

Value

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

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")
```

---

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.
...	not used.
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".

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 mod-
ifications. 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

```
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")
Y2 <- shuffled(X)
ggplot2::ggplot(Y2, ggplot2::aes(x1, x2, alpha = weights(Y2))) +
  ggplot2::geom_point() +
  ggplot2::labs(title = "shuffled")
Y3 <- augmented(X)
ggplot2::ggplot(Y3, ggplot2::aes(x1, x2, alpha = weights(Y3))) +
  ggplot2::geom_point() +
  ggplot2::labs(title = "augmented")
Y4 <- latticized(X)
ggplot2::ggplot(Y4, ggplot2::aes(x1, x2, size = weights(Y4))) +
  ggplot2::geom_point() +
  ggplot2::labs(title = "latticized")
```

---

weighted.quantile   *Weighted Sample Quantile*

Description

weighted.quantile() produces weighted sample quantiles corresponding to the given probabili-
ties.

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.rmse	*Weighted Loss Functions*

Description

`weighted.rmse()`, `weighted.mae()` and `weighted.medae()` compute the loss from a weighted vector of prediction errors.

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 errors.
y	an optional numeric vector. If passed, the loss is calculated for the differences between x and y.
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.rmse() returns the root mean square error, weighted.mae() returns the mean absolute error, and weighted.medae() returns the median absolute error for a weighted vector.

Value

weighted.rmse() (root mean squared 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))
```

---

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