

Package ‘pdp’

October 8, 2026

Type Package

Title Partial Dependence Plots

Version 0.10.0

Description A general framework for constructing partial dependence (i.e., marginal effect) plots from various types of machine learning models in R.

License GPL (>= 2)

URL <https://github.com/bgreenwell/pdp>,
<https://bgreenwell.github.io/pdp/>,
<https://bgreenwell.r-universe.dev/pdp>

BugReports <https://github.com/bgreenwell/pdp/issues>

Depends R (>= 3.6.0)

Suggests C50, caret, covr, doParallel, e1071, earth, foreach, gbm, gridExtra, ICEbox, kernlab, knitr, magrittr, MASS, Matrix, mlbench, nnet, party, randomForest, ranger, rmarkdown, rpart, tinytest, xgboost (>= 3.1.3.1)

Imports graphics, grDevices, lattice, methods, stats, tinyplot, utils

LazyData TRUE

RoxygenNote 7.3.3

Encoding UTF-8

VignetteBuilder knitr

NeedsCompilation yes

Author Brandon M. Greenwell [aut, cre] (ORCID:
<<https://orcid.org/0000-0002-8120-0084>>)

Maintainer Brandon M. Greenwell <greenwell.brandon@gmail.com>

Repository CRAN

Date/Publication 2026-10-08 17:10:02 UTC

Contents

boston	2
exemplar	3
partial	4
pima	10
plot.partial	11
plotPartial	14
Index	18

boston	<i>Boston Housing Data</i>
--------	----------------------------

Description

Data on median housing values from 506 census tracts in the suburbs of Boston from the 1970 census. This data frame is a corrected version of the original data by Harrison and Rubinfeld (1978) with additional spatial information. The data were taken directly from [mlbench::BostonHousing2\(\)](#) and unneeded columns (i.e., name of town, census tract, and the uncorrected median home value) were removed.

Usage

data(boston)

Format

A data frame with 506 rows and 16 variables.

- lon Longitude of census tract.
- lat Latitude of census tract.
- cmedv Corrected median value of owner-occupied homes in USD 1000's
- crim Per capita crime rate by town.
- zn Proportion of residential land zoned for lots over 25,000 sq.ft.
- indus Proportion of non-retail business acres per town.
- chas Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
- nox Nitric oxides concentration (parts per 10 million).
- rm Average number of rooms per dwelling.
- age Proportion of owner-occupied units built prior to 1940.
- dis Weighted distances to five Boston employment centers.
- rad Index of accessibility to radial highways.
- tax Full-value property-tax rate per USD 10,000.
- ptratio Pupil-teacher ratio by town.
- b $\$1000(B - 0.63)^2$ where B is the proportion of blacks by town.
- lstat Percentage of lower status of the population.

References

- Harrison, D. and Rubinfeld, D.L. (1978). Hedonic prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5, 81-102.
- Gilley, O.W., and R. Kelley Pace (1996). On the Harrison and Rubinfeld Data. *Journal of Environmental Economics and Management*, 31, 403-405.
- Newman, D.J. & Hettich, S. & Blake, C.L. & Merz, C.J. (1998). UCI Repository of machine learning databases <https://archive.ics.uci.edu/>. Irvine, CA: University of California, Department of Information and Computer Science.
- Pace, R. Kelley, and O.W. Gilley (1997). Using the Spatial Configuration of the Data to Improve Estimation. *Journal of the Real Estate Finance and Economics*, 14, 333-340.
- Friedrich Leisch & Evgenia Dimitriadou (2010). mlbench: Machine Learning Benchmark Problems. R package version 2.1-1.

Examples

```
head(boston)
```

exemplar

Exemplar observation

Description

Construct a single "exemplar" record from a data frame. For now, all numeric columns (including [Date](#) objects) are replaced with their corresponding median value and non-numeric columns are replaced with their most frequent value.

Usage

```
exemplar(object, ...)

## S3 method for class 'data.frame'
exemplar(object, ...)

## S3 method for class 'matrix'
exemplar(object, cats = NULL, ...)

## S3 method for class 'dgCMatrix'
exemplar(object, cats = NULL, ...)
```

Arguments

object	A data frame, matrix, or dgCMatrix (the latter two are supported by xgboost::xgboost()).
...	Additional optional arguments (currently ignored).

cats Character string indicating which columns of object should be treated as categorical variables and summarized by their most frequent value (rather than a rounded median). Only used when object inherits from class "matrix" or "dgCMatrix"; data frames handle this automatically for factor and character columns. Default is NULL.

Value

A data frame with the same number of columns as object and a single row.

Examples

```
set.seed(1554) # for reproducibility
train <- data.frame(
  x = rnorm(100),
  y = sample(letters[1L:3L], size = 100, replace = TRUE,
            prob = c(0.1, 0.1, 0.8))
)
exemplar(train)
```

partial

Partial Dependence Functions

Description

Compute partial dependence functions (i.e., marginal effects) for various model fitting objects.

Usage

```
partial(object, ...)

## Default S3 method:
partial(
  object,
  pred.var,
  pred.grid,
  pred.fun = NULL,
  grid.resolution = NULL,
  ice = FALSE,
  center = FALSE,
  approx = FALSE,
  quantiles = FALSE,
  probs = 1:9/10,
  trim.outliers = FALSE,
  type = c("auto", "regression", "classification"),
  inv.link = NULL,
  which.class = 1L,
  prob = FALSE,
```

```

    recursive = TRUE,
    plot = FALSE,
    plot.engine = c("tinypplot", "lattice"),
    smooth = FALSE,
    rug = FALSE,
    chull = FALSE,
    levelplot = TRUE,
    contour = FALSE,
    contour.color = "white",
    alpha = 1,
    train,
    cats = NULL,
    check.class = TRUE,
    batch.size = NULL,
    progress = FALSE,
    parallel = FALSE,
    paropts = NULL,
    frac = 1,
    ...
)

## S3 method for class 'model_fit'
partial(object, ...)

```

Arguments

object	A fitted model object of appropriate class (e.g., "gbm", "lm", "randomForest", "train", etc.).
...	Additional optional arguments to be passed onto <code>stats::predict()</code> .
pred.var	Character string giving the names of the predictor variables of interest. For reasons of computation/interpretation, this should include no more than three variables. Can be omitted whenever pred.grid is supplied, in which case it defaults to <code>colnames(pred.grid)</code> .
pred.grid	Data frame containing the joint values of interest for the variables listed in pred.var.
pred.fun	Optional prediction function that requires two arguments: object and newdata. If specified, then the function must return a single prediction or a vector of predictions (i.e., not a matrix or data frame). Default is NULL.
grid.resolution	Integer giving the number of equally spaced points to use for the continuous variables listed in pred.var when pred.grid is not supplied. If left NULL, it will default to the minimum between 51 and the number of unique data points for each of the continuous independent variables listed in pred.var.
ice	Logical indicating whether or not to compute individual conditional expectation (ICE) curves. Default is FALSE. See Goldstein et al. (2014) for details.
center	Logical indicating whether or not to produce centered ICE curves (c-ICE curves). Only used when ice = TRUE. Default is FALSE. See Goldstein et al. (2014) for details.

approx	Logical indicating whether or not to compute a faster, but approximate, marginal effect plot (similar in spirit to the plotmo package). If TRUE, then <code>partial()</code> will compute predictions across the predictors specified in <code>pred.var</code> while holding the other predictors constant (a "poor man's partial dependence" function as Stephen Milborrow, the author of plotmo , puts it). Default is FALSE. Note this works with <code>ice = TRUE</code> as well. WARNING: This option is currently experimental. Use at your own risk. It is possible (and arguably safer) to do this manually by passing a specific "exemplar" observation to the <code>train</code> argument and specifying <code>pred.grid</code> manually.
quantiles	Logical indicating whether or not to use the sample quantiles of the continuous predictors listed in <code>pred.var</code> . If <code>quantiles = TRUE</code> and <code>grid.resolution = NULL</code> the sample quantiles will be used to generate the grid of joint values for which the partial dependence is computed.
probs	Numeric vector of probabilities with values in $[0, 1]$. (Values up to $2e-14$ outside that range are accepted and moved to the nearby endpoint.) Default is $1:9/10$ which corresponds to the deciles of the predictor variables. These specify which quantiles to use for the continuous predictors listed in <code>pred.var</code> when <code>quantiles = TRUE</code> .
trim.outliers	Logical indicating whether or not to trim off outliers from the continuous predictors listed in <code>pred.var</code> (using the simple boxplot method) before generating the grid of joint values for which the partial dependence is computed. Default is FALSE.
type	Character string specifying the type of supervised learning. Current options are "auto", "regression" or "classification". If <code>type = "auto"</code> then <code>partial</code> will try to extract the necessary information from <code>object</code> .
inv.link	Function specifying the transformation to be applied to the predictions before the partial dependence function is computed (experimental). Default is NULL (i.e., no transformation). This option is intended to be used for models that allow for non-Gaussian response variables (e.g., counts). For these models, predictions are not typically returned on the original response scale by default. For example, Poisson GBMs typically return predictions on the log scale. In this case setting <code>inv.link = exp</code> will return the partial dependence function on the response (i.e., raw count) scale.
which.class	Integer specifying which column of the matrix of predicted probabilities to use as the "focus" class. Default is to use the first class. Only used for classification problems (i.e., when <code>type = "classification"</code>).
prob	Logical indicating whether or not partial dependence for classification problems should be returned on the probability scale, rather than the centered logit. If FALSE, the partial dependence function is on a scale similar to the logit. Default is FALSE.
recursive	Logical indicating whether or not to use the weighted tree traversal method described in Friedman (2001). This only applies to objects that inherit from class "gbm". Default is TRUE which is much faster than the exact brute force approach used for all other models. (Based on the C++ code behind <code>gbm::plot.gbm()</code> .)
plot	Logical indicating whether to return a data frame containing the partial dependence values (FALSE) or plot the partial dependence function directly (TRUE). Default is FALSE. See <code>plotPartial()</code> for plotting details.

<code>plot.engine</code>	Character string specifying which plotting engine to use whenever <code>plot = TRUE</code> . Options include "tinypplot" (default; lightweight base R graphics via the tinyplot package) or "lattice".
<code>smooth</code>	Logical indicating whether or not to overlay a LOESS smooth. Default is FALSE.
<code>rug</code>	Logical indicating whether or not to include a rug display on the predictor axes. The tick marks indicate the min/max and deciles of the predictor distributions. This helps reduce the risk of interpreting the partial dependence plot outside the region of the data (i.e., extrapolating). Only used when <code>plot = TRUE</code> . Default is FALSE.
<code>chull</code>	Logical indicating whether or not to restrict the values of the first two variables in <code>pred.var</code> to lie within the convex hull of their training values; this affects <code>pred.grid</code> . This helps reduce the risk of interpreting the partial dependence plot outside the region of the data (i.e., extrapolating). Default is FALSE.
<code>levelplot</code>	Logical indicating whether or not to use a false color level plot (TRUE) or a 3-D surface (FALSE). Default is TRUE.
<code>contour</code>	Logical indicating whether or not to add contour lines to the level plot. Only used when <code>levelplot = TRUE</code> . Default is FALSE.
<code>contour.color</code>	Character string specifying the color to use for the contour lines when <code>contour = TRUE</code> . Default is "white".
<code>alpha</code>	Numeric value in $[0, 1]$ specifying the opacity alpha (most useful when plotting ICE/c-ICE curves). Default is 1 (i.e., no transparency). In fact, this option only affects ICE/c-ICE curves and level plots.
<code>train</code>	An optional data frame, matrix, or sparse matrix containing the original training data. This may be required depending on the class of object. For objects that do not store a copy of the original training data, this argument is required. For reasons discussed below, it is good practice to always specify this argument.
<code>cats</code>	Character string indicating which columns of <code>train</code> should be treated as categorical variables. Only used when <code>train</code> inherits from class "matrix" or "dgCMatrix".
<code>check.class</code>	Logical indicating whether or not to make sure each column in <code>pred.grid</code> has the correct class, levels, etc. Default is TRUE.
<code>batch.size</code>	Optional positive integer specifying the (approximate) maximum number of rows to score per call to <code>stats::predict()</code> . By default (<code>batch.size = NULL</code>), <code>partial()</code> calls <code>stats::predict()</code> once per grid point (i.e., <code>nrow(train)</code> rows at a time). Specifying a larger batch size (e.g., <code>batch.size = 1e6</code>) stacks multiple grid points into a single call to <code>stats::predict()</code> , which is often substantially faster since it avoids the per-call overhead of most prediction methods, at the cost of additional memory. Requires the prediction function to return one prediction per row of newdata, so it cannot be used with a <code>pred.fun</code> that aggregates its own predictions. Prediction names are also ignored when batching (i.e., <code>yhat.id</code> will always contain integer IDs). Ignored whenever the recursive method is used (i.e., for "gbm" objects with <code>recursive = TRUE</code>).
<code>progress</code>	Logical indicating whether or not to display a text-based progress bar. Default is FALSE.

<code>parallel</code>	Logical indicating whether or not to run <code>partial</code> in parallel using a backend provided by the <code>foreach</code> package. Default is <code>FALSE</code> .
<code>paropts</code>	List containing additional options to be passed onto <code>foreach::foreach()</code> when <code>parallel = TRUE</code> .
<code>frac</code>	Numeric value in $(0, 1]$ specifying the fraction of the training data to randomly sample (without replacement) before computing the partial dependence function. Default is 1 (i.e., use all of the training data). Mostly useful for reducing the number of ICE curves and/or computation time; use <code>base::set.seed()</code> for reproducibility. Ignored whenever the recursive method is used (i.e., for "gbm" objects with <code>recursive = TRUE</code>).

Value

By default, `partial` returns an object of class `c("data.frame", "partial")`. If `ice = TRUE` and `center = FALSE` then an object of class `c("data.frame", "ice")` is returned. If `ice = TRUE` and `center = TRUE` then an object of class `c("data.frame", "cice")` is returned. These three classes determine the behavior of the plotting functions that are automatically called whenever `plot = TRUE`. Specifically, when `plot = TRUE` and `plot.engine = "tinyplot"` (the default), the plot is drawn directly (as a side effect) and the data frame of partial dependence values is returned invisibly. When `plot = TRUE` and `plot.engine = "lattice"`, a "trellis" object is returned (see **lattice** for details); the "trellis" object will also include an additional attribute, `"partial.data"`, containing the data displayed in the plot.

Note

In some cases it is difficult for `partial` to extract the original training data from object. In these cases an error message is displayed requesting the user to supply the training data via the `train` argument in the call to `partial`. In most cases where `partial` can extract the required training data from object, it is taken from the same environment in which `partial` is called. Therefore, it is important to not change the training data used to construct object before calling `partial`. This problem is completely avoided when the training data are passed to the `train` argument in the call to `partial`.

It is recommended to call `partial` with `plot = FALSE` and store the results. This allows for more flexible plotting, and the user will not have to waste time calling `partial` again if the default plot is not sufficient.

It is possible to retrieve the last printed "trellis" object, such as those produced by `plotPartial`, using `trellis.last.object()`.

If `ice = TRUE` or the prediction function given to `pred.fun` returns a prediction for each observation in `newdata`, then the result will be a curve for each observation. These are called individual conditional expectation (ICE) curves; see Goldstein et al. (2015) and `ICEbox::ice()` for details.

References

- J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, **29**: 1189-1232, 2001.
- Goldstein, A., Kapelner, A., Bleich, J., and Pitkin, E., Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation. (2014) *Journal of Computational and Graphical Statistics*, **24**(1): 44-65, 2015.

Examples

```
## Not run:
#
# Regression example (requires randomForest package to run)
#

# Fit a random forest to the boston housing data
library(randomForest)
data(boston) # load the boston housing data
set.seed(101) # for reproducibility
boston.rf <- randomForest(cmedv ~ ., data = boston)

# Using randomForest's partialPlot function
partialPlot(boston.rf, pred.data = boston, x.var = "lstat")

# Using pdp's partial function
head(partial(boston.rf, pred.var = "lstat")) # returns a data frame
partial(boston.rf, pred.var = "lstat", plot = TRUE, rug = TRUE)

# The partial function allows for multiple predictors
partial(boston.rf, pred.var = c("lstat", "rm"), grid.resolution = 40,
        plot = TRUE, chull = TRUE, progress = TRUE)

# The plot method produces lightweight base R graphics via the tinyplot
# package by default; set `lattice = TRUE` for lattice graphics (e.g., for
# 3-D surfaces or paneled three-predictor displays)
pd <- partial(boston.rf, pred.var = c("lstat", "rm"), grid.resolution = 40)
plot(pd, contour = TRUE)
plot(pd, lattice = TRUE, levelplot = FALSE, zlab = "cmedv", drape = TRUE,
     colorkey = FALSE, screen = list(z = -20, x = -60))

#
# Individual conditional expectation (ICE) curves
#

# Use partial to obtain ICE/c-ICE curves
rm.ice <- partial(boston.rf, pred.var = "rm", ice = TRUE)
plot(rm.ice, rug = TRUE, train = boston, alpha = 0.2)
plot(rm.ice, center = TRUE, alpha = 0.2, rug = TRUE, train = boston)

#
# Classification example (requires randomForest package to run)
#

# Fit a random forest to the (synthetic) diabetes data
data(pima) # load the synthetic diabetes data
set.seed(102) # for reproducibility
pima.rf <- randomForest(diabetes ~ ., data = pima, na.action = na.omit)

# Partial dependence of positive test result on glucose (default logit scale)
partial(pima.rf, pred.var = "glucose", plot = TRUE, chull = TRUE,
        progress = TRUE)
```

```
# Partial dependence of positive test result on glucose (probability scale)
partial(pima.rf, pred.var = "glucose", prob = TRUE, plot = TRUE,
        chull = TRUE, progress = TRUE)

## End(Not run)
```

pima

Synthetic Diabetes Data

Description

A fully synthetic diabetes data set created by Matthias Templ to mimic the Pima Indians diabetes data analyzed by Smith et al. (1988). Every value is synthetic and no row corresponds to a real person. The data were taken directly from [mlbench::SynthDiabetes2](#), which mimics the missing-value pattern of the original data (physically impossible zeros are coded as NA).

Usage

```
data(pima)
```

Format

A data frame with 768 observations on 9 variables.

- pregnant Number of times pregnant.
- glucose Plasma glucose concentration (glucose tolerance test).
- pressure Diastolic blood pressure (mm Hg).
- triceps Triceps skin fold thickness (mm).
- insulin 2-Hour serum insulin (mu U/ml).
- mass Body mass index (weight in kg/(height in m)^2).
- pedigree Diabetes pedigree function.
- age Age (years).
- diabetes Factor indicating the diabetes test result (neg/pos).

Details

Earlier versions of pdp shipped a copy of the original data (taken from `mlbench::PimaIndiansDiabetes2`) under this name. The original data had most likely been shared without the consent of the participants, and both the UCI repository and `mlbench` (as of version 2.1-11) have stopped distributing it. The `pima` name is kept so existing code continues to run, but results will differ from those based on the original data.

References

- Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., and Johannes, R.S. (1988). Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus. In Proceedings of the Symposium on Computer Applications and Medical Care, 261-265.
- Brian D. Ripley (1996), Pattern Recognition and Neural Networks, Cambridge University Press, Cambridge.
- Grace Whaba, Chong Gu, Yuedong Wang, and Richard Chappell (1995), Soft Classification a.k.a. Risk Estimation via Penalized Log Likelihood and Smoothing Spline Analysis of Variance, in D. H. Wolpert (1995), The Mathematics of Generalization, 331-359, Addison-Wesley, Reading, MA.
- Friedrich Leisch & Evgenia Dimitriadou (2026). mlbench: Machine Learning Benchmark Problems. R package version 2.1-11.

Examples

```
head(pima)
```

plot.partial

Plotting Partial Dependence Functions

Description

Plot partial dependence functions (i.e., marginal effects) and individual conditional expectation (ICE) curves using lightweight base R graphics via the **tinypplot** package, or **lattice** graphics whenever `lattice = TRUE`.

Usage

```
## S3 method for class 'partial'
plot(
  x,
  center = FALSE,
  plot.pdp = TRUE,
  pdp.col = "red2",
  pdp.lwd = 2,
  pdp.lty = 1,
  smooth = FALSE,
  rug = FALSE,
  contour = FALSE,
  contour.color = "white",
  train = NULL,
  alpha = 1,
  color.by = NULL,
  bars = FALSE,
  legend.title = "yhat",
  lattice = FALSE,
```

```

    ...
)

## S3 method for class 'ice'
plot(
  x,
  center = FALSE,
  plot.pdp = TRUE,
  pdp.col = "red2",
  pdp.lwd = 2,
  pdp.lty = 1,
  rug = FALSE,
  train = NULL,
  alpha = 1,
  color.by = NULL,
  lattice = FALSE,
  ...
)

## S3 method for class 'cice'
plot(
  x,
  plot.pdp = TRUE,
  pdp.col = "red2",
  pdp.lwd = 2,
  pdp.lty = 1,
  rug = FALSE,
  train = NULL,
  alpha = 1,
  color.by = NULL,
  lattice = FALSE,
  ...
)

```

Arguments

x	An object that inherits from class "partial", "ice", or "cice"; typically the result of a call to partial() .
center	Logical indicating whether or not to produce centered ICE curves (c-ICE curves). Only useful when x represents a set of ICE curves; see partial() for details. Default is FALSE.
plot.pdp	Logical indicating whether or not to plot the partial dependence function on top of the ICE curves. Default is TRUE.
pdp.col	Character string specifying the color to use for the partial dependence function when plot.pdp = TRUE. Default is "red2".
pdp.lwd	Integer specifying the line width to use for the partial dependence function when plot.pdp = TRUE. Default is 2. See graphics::par() for more details.

pdp.lty	Integer or character string specifying the line type to use for the partial dependence function when <code>plot.pdp = TRUE</code> . Default is 1. See graphics::par() for more details.
smooth	Logical indicating whether or not to overlay a LOESS smooth. Default is FALSE.
rug	Logical indicating whether or not to include rug marks (i.e., the min/max and deciles of the predictor distribution) on the predictor axes. Not currently supported for faceted displays (i.e., partial dependence of two predictors where at least one is a factor). Default is FALSE.
contour	Logical indicating whether or not to add contour lines to the false color level plot used for two continuous predictors. Default is FALSE.
contour.color	Character string specifying the color to use for the contour lines when <code>contour = TRUE</code> . Default is "white".
train	Data frame containing the original training data. Only required if <code>rug = TRUE</code> .
alpha	Numeric value in $[0, 1]$ specifying the opacity alpha; most useful when plotting ICE/c-ICE curves. Default is 1 (i.e., no transparency).
color.by	Optional character string specifying the name of a column in <code>train</code> used to color the individual ICE/c-ICE curves; continuous variables are binned into (at most) five groups. Requires <code>train</code> and assumes the curve IDs (i.e., the <code>yhat.id</code> column) correspond to the rows of <code>train</code> (which is the case whenever <code>ice = TRUE</code>). Default is NULL.
bars	Logical indicating whether or not to use a bar plot (rather than points) whenever the predictor of interest is a factor. Default is FALSE.
legend.title	Character string specifying the text for the legend title of the false color level plot used for two continuous predictors. Default is "yhat".
lattice	Logical indicating whether or not to draw the display using lattice graphics instead of <code>tinypLOT</code> /base graphics. The lattice engine additionally supports three-predictor (paneled) displays and 3-D surfaces; see Details. Default is FALSE.
...	Additional optional arguments to be passed on to tinypLOT::tinypLOT() (e.g., <code>palette</code> , <code>main</code> , or <code>theme</code>) or, whenever <code>lattice = TRUE</code> , to the underlying lattice display (see Details).

Details

When `lattice = TRUE`, the display is constructed with **lattice** graphics (this subsumes the now-deprecated `plotPartial()` interface). In that case, additional lattice-specific options can be supplied via ...:

- `levelplot` - use a false color level plot (TRUE; default) or a 3-D [lattice::wireframe\(\)](#) surface (FALSE) for two continuous predictors;
- `chull` - overlay the convex hull of the first two predictors (requires `train`);
- `col.regions` - color palette for level/wireframe plots;
- `number/overlap` - number of conditioning intervals (and their fraction of overlap) used to panel a third (continuous) predictor;
- any other argument accepted by [lattice::xyplot\(\)](#), [lattice::levelplot\(\)](#), [lattice::wireframe\(\)](#), or [lattice::dotplot\(\)](#) (e.g., `screen` or `drape`). The `tinypLOT`-specific arguments `color.by`, `bars`, and `legend.title` are ignored when `lattice = TRUE`.

Value

Draws a plot as a side effect. The tinyplot engine (invisibly) returns `x`; the lattice engine (`lattice = TRUE`) (invisibly) returns the "trellis" object, which can be captured for further manipulation (e.g., arranging multiple displays with `gridExtra::grid.arrange()`).

Examples

```
## Not run:
#
# Regression example (requires randomForest package to run)
#

# Fit a random forest to the Boston housing data
library(randomForest)
data(boston) # load the boston housing data
set.seed(101) # for reproducibility
boston.rf <- randomForest(cmedv ~ ., data = boston)

# Partial dependence of cmedv on lstat
pd <- partial(boston.rf, pred.var = "lstat")
plot(pd, rug = TRUE, train = boston)

# Partial dependence of cmedv on lstat and rm
pd2 <- partial(boston.rf, pred.var = c("lstat", "rm"), chull = TRUE)
plot(pd2, contour = TRUE)

# ICE and c-ICE curves
rm.ice <- partial(boston.rf, pred.var = "rm", ice = TRUE)
plot(rm.ice, rug = TRUE, train = boston, alpha = 0.2)
plot(rm.ice, center = TRUE, alpha = 0.2)

## End(Not run)
```

plotPartial

Plotting Partial Dependence Functions (deprecated)

Description

Plots partial dependence functions (i.e., marginal effects) using **lattice** graphics.

Usage

```
plotPartial(object, ...)

## S3 method for class 'ice'
plotPartial(
  object,
  center = FALSE,
  plot.pdp = TRUE,
```

```

    pdp.col = "red2",
    pdp.lwd = 2,
    pdp.lty = 1,
    rug = FALSE,
    train = NULL,
    ...
)

## S3 method for class 'cice'
plotPartial(
  object,
  plot.pdp = TRUE,
  pdp.col = "red2",
  pdp.lwd = 2,
  pdp.lty = 1,
  rug = FALSE,
  train = NULL,
  ...
)

## S3 method for class 'partial'
plotPartial(
  object,
  center = FALSE,
  plot.pdp = TRUE,
  pdp.col = "red2",
  pdp.lwd = 2,
  pdp.lty = 1,
  smooth = FALSE,
  rug = FALSE,
  chull = FALSE,
  levelplot = TRUE,
  contour = FALSE,
  contour.color = "white",
  col.regions = NULL,
  number = 4,
  overlap = 0.1,
  train = NULL,
  ...
)

```

Arguments

object	An object that inherits from the "partial" class.
...	Additional optional arguments to be passed onto dotplot, levelplot, xyplot, or wireframe.
center	Logical indicating whether or not to produce centered ICE curves (c-ICE curves). Only useful when object represents a set of ICE curves; see partial() for de-

	tails. Default is FALSE.
plot.pdp	Logical indicating whether or not to plot the partial dependence function on top of the ICE curves. Default is TRUE.
pdp.col	Character string specifying the color to use for the partial dependence function when plot.pdp = TRUE. Default is "red".
pdp.lwd	Integer specifying the line width to use for the partial dependence function when plot.pdp = TRUE. Default is 1. See graphics::par() for more details.
pdp.lty	Integer or character string specifying the line type to use for the partial dependence function when plot.pdp = TRUE. Default is 1. See graphics::par() for more details.
rug	Logical indicating whether or not to include rug marks on the predictor axes. Default is FALSE.
train	Data frame containing the original training data. Only required if rug = TRUE or chull = TRUE.
smooth	Logical indicating whether or not to overlay a LOESS smooth. Default is FALSE.
chull	Logical indicating whether or not to restrict the first two variables in pred.var to lie within the convex hull of their training values; this affects pred.grid. Default is FALSE.
levelplot	Logical indicating whether or not to use a false color level plot (TRUE) or a 3-D surface (FALSE). Default is TRUE.
contour	Logical indicating whether or not to add contour lines to the level plot. Only used when levelplot = TRUE. Default is FALSE.
contour.color	Character string specifying the color to use for the contour lines when contour = TRUE. Default is "white".
col.regions	Vector of colors to be passed on to lattice::levelplot() 's col.region argument. Defaults to grDevices::hcl.colors(100) (which is the same viridis color palette used in the past).
number	Integer specifying the number of conditional intervals to use for the continuous panel variables. See graphics::co.intervals() and lattice::equal.count() for further details.
overlap	The fraction of overlap of the conditioning variables. See graphics::co.intervals() and lattice::equal.count() for further details.

Details

Deprecated: `plotPartial()` is deprecated and will be removed in a future release; please use `plot(..., lattice = TRUE)` instead, which produces the same displays through a single interface (see [plot.partial\(\)](#) for details).

Examples

```
## Not run:
#
# Regression example (requires randomForest package to run)
#
```

```
# Load required packages
library(gridExtra) # for `grid.arrange()`
library(magrittr)  # for forward pipe operator `%>%`
library(randomForest)

# Fit a random forest to the Boston housing data
data(boston) # load the boston housing data
set.seed(101) # for reproducibility
boston.rf <- randomForest(cmedv ~ ., data = boston)

# Partial dependence of cmedv on lstat
boston.rf %>%
  partial(pred.var = "lstat") %>%
  plotPartial(rug = TRUE, train = boston)

# Partial dependence of cmedv on lstat and rm
boston.rf %>%
  partial(pred.var = c("lstat", "rm"), chull = TRUE, progress = TRUE) %>%
  plotPartial(contour = TRUE, legend.title = "rm")

# ICE curves and c-ICE curves
age.ice <- partial(boston.rf, pred.var = "lstat", ice = TRUE)
p1 <- plotPartial(age.ice, alpha = 0.1)
p2 <- plotPartial(age.ice, center = TRUE, alpha = 0.1)
grid.arrange(p1, p2, ncol = 2)

## End(Not run)
```

Index

- * **datasets**
 - boston, [2](#)
 - pima, [10](#)
- base::set.seed(), [8](#)
- boston, [2](#)
- Date, [3](#)
- dgCMatrix, [3](#)
- exemplar, [3](#)
- foreach::foreach(), [8](#)
- gbm::plot.gbm(), [6](#)
- graphics::co.intervals(), [16](#)
- graphics::par(), [12](#), [13](#), [16](#)
- ICEbox::ice(), [8](#)
- lattice::dotplot(), [13](#)
- lattice::equal.count(), [16](#)
- lattice::levelplot(), [13](#), [16](#)
- lattice::wireframe(), [13](#)
- lattice::xyplot(), [13](#)
- mlbench::BostonHousing2(), [2](#)
- mlbench::SynthDiabetes2, [10](#)
- partial, [4](#)
- partial(), [12](#), [15](#)
- pima, [10](#)
- plot.cice(plot.partial), [11](#)
- plot.ice(plot.partial), [11](#)
- plot.partial, [11](#)
- plot.partial(), [16](#)
- plotPartial, [14](#)
- plotPartial(), [6](#)
- stats::predict(), [5](#), [7](#)
- tinypLOT::tinypLOT(), [13](#)
- xgboost::xgboost(), [3](#)