Package 'pdp'

October 14, 2022

Type Package

Title Partial Dependence Plots

Version 0.8.1

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

License GPL (>= 2)

URL https://github.com/bgreenwell/pdp, http://bgreenwell.github.io/pdp/

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

Depends R (>= 3.6.0)

Suggests adabag, AmesHousing, C50, caret, covr, Cubist, doParallel, dplyr, e1071, earth, gbm, gridExtra, ICEbox, ipred, keras, kernlab, magrittr, MASS, Matrix, mda, mlbench, nnet, party, partykit, randomForest, ranger, reticulate, rpart, tinytest, xgboost (>= 0.6-0), knitr, rmarkdown, vip

Imports foreach, ggplot2 (>= 3.0.0), grDevices, lattice, methods, rlang (>= 0.3.0), stats, utils

LazyData TRUE

RoxygenNote 7.1.2

Encoding UTF-8

VignetteBuilder knitr

NeedsCompilation yes

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

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

Repository CRAN

Date/Publication 2022-06-07 20:40:05 UTC

R topics documented:

`autoplot.partial` *Plotting Partial Dependence Functions*

Description

Plots partial dependence functions (i.e., marginal effects) using `ggplot2` graphics.

Usage

```
## S3 method for class 'partial'
autoplot(
  object,
  center = FALSE,
  plot.pdp = TRUE,
  pdp.color = "red",
  pdp.size = 1,
  pdp.linetype = 1,
  rug = FALSE,
  smooth = FALSE,
  smooth.method = "auto",
  smooth.formula = y ~ x,
  smooth.span = 0.75,
  smooth.method.args = list(),
  contour = FALSE,
  contour.color = "white",
  train = NULL,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  legend.title = "yhat",
  ...
)

## S3 method for class 'ice'
autoplot(
```

```
  object,
  center = FALSE,
  plot.pdp = TRUE,
  pdp.color = "red",
  pdp.size = 1,
  pdp.linetype = 1,
  rug = FALSE,
  train = NULL,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  ...
)

## S3 method for class 'cice'
autoplot(
  object,
  plot.pdp = TRUE,
  pdp.color = "red",
  pdp.size = 1,
  pdp.linetype = 1,
  rug = FALSE,
  train = NULL,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  ...
)
```

Arguments

object	An object that inherits from the "partial" class.
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 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.color	Character string specifying the color to use for the partial dependence function when plot.pdp = TRUE. Default is "red".
pdp.size	Positive number specifying the line width to use for the partial dependence function when plot.pdp = TRUE. Default is 1.
pdp.linetype	Positive number specifying the line type to use for the partial dependence function when plot.pdp = TRUE. Default is 1.
rug	Logical indicating whether or not to include rug marks on the predictor axes. Default is FALSE.
smooth	Logical indicating whether or not to overlay a LOESS smooth. Default is FALSE.

smooth.method	Character string specifying the smoothing method (function) to use (e.g., "auto", "lm", "glm", "gam", "loess", or "rlm"). Default is "auto". See geom_smooth for details.
smooth.formula	Formula to use in smoothing function (e.g., y ~ x, y ~ poly(x, 2), or y ~ log(x)).
smooth.span	Controls the amount of smoothing for the default loess smoother. Smaller numbers produce wigglier lines, larger numbers produce smoother lines. Default is 0.75.
smooth.method.args	List containing additional arguments to be passed on to the modeling function defined by smooth.method.
contour	Logical indicating whether or not to add contour lines to the level plot.
contour.color	Character string specifying the color to use for the contour lines when contour = TRUE. Default is "white".
train	Data frame containing the original training data. Only required if rug = TRUE or chull = TRUE.
xlab	Character string specifying the text for the x-axis label.
ylab	Character string specifying the text for the y-axis label.
main	Character string specifying the text for the main title of the plot.
legend.title	Character string specifying the text for the legend title. Default is "yhat".
...	Additional (optional) arguments to be passed onto geom_line, geom_point, or scale_fill_viridis_c.

Value

A "ggplot" object.

Examples

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

# Load required packages
library(ggplot2)  # for autoplot() generic
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") %>%
```

```
  autoplot(rug = TRUE, train = boston) + theme_bw()

# Partial dependence of cmedv on lstat and rm
boston.rf %>%
  partial(pred.var = c("lstat", "rm"), chull = TRUE, progress = TRUE) %>%
  autoplot(contour = TRUE, legend.title = "cmedv",
           option = "B", direction = -1) + theme_bw()

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

## End(Not run)
```

`boston` *Boston Housing Data*

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 `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 [http://www.ics.uci.edu/~mlearn/MLRepository.html] 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)

## S3 method for class 'dgCMatrix'
exemplar(object)
```

Arguments

object	A data frame, matrix, or dgCMatrix (the latter two are supported by xgboost).

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("lattice", "ggplot2"),
  smooth = FALSE,
  rug = FALSE,
  chull = FALSE,
  levelplot = TRUE,
  contour = FALSE,
  contour.color = "white",
  alpha = 1,
  train,
  cats = NULL,
  check.class = TRUE,
  progress = FALSE,
  parallel = FALSE,
  paropts = NULL,
  ...
)

## 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 predict.
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.
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 `partial()` will compute predictions across the predictors specified in `pred.var` 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 `ice = TRUE` 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 train argument and specifying `pred.grid` manually.

`quantiles` Logical indicating whether or not to use the sample quantiles of the continuous predictors listed in `pred.var`. If `quantiles = TRUE` and `grid.resolution = NULL` 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 `pred.var` when `quantiles = TRUE`.

`trim.outliers` Logical indicating whether or not to trim off outliers from the continuous predictors listed in `pred.var` (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 `type = "auto"` then `partial` will try to extract the necessary information from `object`.

`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 `inv.link = exp` 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 `type = "classification"`).

`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 `plot.gbm`.)

`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 `plotPartial` for plotting details.

plot.engine	Character string specifying which plotting engine to use whenever plot = TRUE. Options include "lattice" (default) or "ggplot2".
smooth	Logical indicating whether or not to overlay a LOESS smooth. Default is FALSE.
rug	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 plot = TRUE. Default is FALSE.
chull	Logical indicating whether or not to restrict the values of the first two variables in pred.var to lie within the convex hull of their training values; this affects pred.grid. This helps reduce the risk of interpreting the partial dependence plot outside the region of the data (i.e., extrapolating).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".
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). In fact, this option only affects ICE/c-ICE curves and level plots.
train	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.
cats	Character string indicating which columns of train should be treated as categorical variables. Only used when train inherits from class "matrix" or "dgCMatrix".
check.class	Logical indicating whether or not to make sure each column in pred.grid has the correct class, levels, etc. Default is TRUE.
progress	Logical indicating whether or not to display a text-based progress bar. Default is FALSE.
parallel	Logical indicating whether or not to run partial in parallel using a backend provided by the foreach package. Default is FALSE.
paropts	List containing additional options to be passed onto foreach when parallel = TRUE.

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 plotPartial function which is automatically called whenever plot = TRUE. Specifically, when plot = TRUE, 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 `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 plotPartial function offers more flexible plotting
pd <- partial(boston.rf, pred.var = c("lstat", "rm"), grid.resolution = 40)
plotPartial(pd, levelplot = FALSE, zlab = "cmedv", drape = TRUE,
            colorkey = FALSE, screen = list(z = -20, x = -60))

# The autplot function can be used to produce graphics based on ggplot2
library(ggplot2)
autoplot(pd, contour = TRUE, legend.title = "Partial\ndependence")

#
# Individual conditional expectation (ICE) curves
#

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

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

# Fit a random forest to the Pima Indians diabetes data
data (pima)  # load the boston housing 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)
```

---

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

Description

Partial dependence plots (PDPs) help visualize the relationship between a subset of the features (typically 1-3) and the response while accounting for the average effect of the other predictors in the model. They are particularly effective with black box models like random forests and support vector machines.

Details

The development version can be found on GitHub: https://github.com/bgreenwell/pdp. As of right now, pdp exports four functions:

- `partial` - construct partial dependence functions (i.e., objects of class `"partial"`) from various fitted model objects;
- `plotPartial` - plot partial dependence functions (i.e., objects of class `"partial"`) using `lattice` graphics;
- `autoplot` - plot partial dependence functions (i.e., objects of class `"partial"`) using `ggplot2` graphics;
- `topPredictors` - extract most "important" predictors from various types of fitted models.

`pima` *Pima Indians Diabetes Data*

Description

Diabetes test results collected by the the US National Institute of Diabetes and Digestive and Kidney Diseases from a population of women who were at least 21 years old, of Pima Indian heritage, and living near Phoenix, Arizona. The data were taken directly from `PimaIndiansDiabetes2`.

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`).

References

Newman, D.J. & Hettich, S. & Blake, C.L. & Merz, C.J. (1998). UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

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 (2010). mlbench: Machine Learning Benchmark Problems. R package version 2.1-1.

Examples

```
head(pima)
```

`plotPartial` *Plotting Partial Dependence Functions*

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 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 "red".
pdp.lwd	Integer specifying the line width to use for the partial dependence function when plot.pdp = TRUE. Default is 1. See 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 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 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 co.intervals and equal.count for further details.
overlap	The fraction of overlap of the conditioning variables. See co.intervals and equal.count for further 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)
```

---

topPredictors *Extract Most "Important" Predictors (Experimental)*

Description

Extract the most "important" predictors for regression and classification models.

Usage

```
topPredictors(object, n = 1L, ...)

## Default S3 method:
topPredictors(object, n = 1L, ...)

## S3 method for class 'train'
topPredictors(object, n = 1L, ...)
```

Arguments

object	A fitted model object of appropriate class (e.g., "gbm", "lm", "randomForest", etc.).
n	Integer specifying the number of predictors to return. Default is 1 meaning return the single most important predictor.
...	Additional optional arguments to be passed onto varImp.

Details

This function uses the generic function varImp to calculate variable importance scores for each predictor. After that, they are sorted at the names of the n highest scoring predictors are returned.

Examples

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

Load required packages
library(ggplot2)
library(randomForest)

# Fit a random forest to the mtcars dataset
```

```
data(mtcars, package = "datasets")
set.seed(101)
mtcars.rf <- randomForest(mpg ~ ., data = mtcars, mtry = 5, importance = TRUE)

# Topfour predictors
top4 <- topPredictors(mtcars.rf, n = 4)

# Construct partial dependence functions for top four predictors
pd <- NULL
for (i in top4) {
  tmp <- partial(mtcars.rf, pred.var = i)
  names(tmp) <- c("x", "y")
  pd <- rbind(pd,  cbind(tmp, predictor = i))
}

# Display partial dependence functions
ggplot(pd, aes(x, y)) +
  geom_line() +
  facet_wrap(~ predictor, scales = "free") +
  theme_bw() +
  ylab("mpg")


## End(Not run)
```

Index