Package ‘pdp’  

Type Package  
Title Partial Dependence Plots  
Version 0.7.0  
Description A general framework for constructing partial dependence (i.e., marginal effect) plots from various types machine learning models in R.  
License GPL (>= 2)  
BugReports https://github.com/bgreenwell/pdp/issues  
Depends R (>= 3.2.5)  
Suggests adabag, AmesHousing, C50, caret, Cubist, doParallel, dplyr, e1071, earth, gbm, ipred, keras, kernlab, MASS, mda, mnet, party, partykit, progress, randomForest, ranger, rpart, testthat, xgboost (>= 0.6-0), knitr, rmarkdown, vip  
Imports ggplot2 (>= 0.9.0), grDevices, gridExtra, lattice, magrittr, methods, mgcv, plyr, stats, viridis, utils  
LazyData TRUE  
RoxygenNote 6.1.0  
Encoding UTF-8  
NeedsCompilation yes  
Author Brandon Greenwell [aut, cre] (<https://orcid.org/0000-0002-8120-0084>)  
Maintainer Brandon Greenwell <greenwell.brandon@gmail.com>  
Repository CRAN  
Date/Publication 2018-08-27 17:50:03 UTC  

R topics documented:  

- autoplot.partial ................................................................. 2  
- boston .............................................................................. 4  

1
Plots partial dependence functions (i.e., marginal effects) using ggplot2 graphics.

Usage

```r
## 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", palette = c("viridis", "magma", "inferno",
"plasma", "cividis"), 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".
autoplot.partial

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".
palette Character string indicating the colormap option to use. Five options are available: "viridis" (the default), "magma", "inferno", "plasma", and "cividis".
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.

Value

A "ggplot" object.

Examples

```r
# Not run:
#
# Regression example (requires randomForest package to run)
#
# Load required packages
library(ggplot2)  # required to use autoplot
library(randomForest)
```
```r
# 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)

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

# ICE curves and c-ICE curves
age.ice <- partial(boston.rf, pred.var = "lstat", ice = TRUE)
grid.arrange(
  autoplot(age.ice, alpha = 0.5),  # ICE curves
  autoplot(age.ice, center = TRUE, alpha = 0.5),  # c-ICE curves
  ncol = 2
)

```  

---

**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


Examples

head(boston)
Usage

partial(object, ...)

## Default S3 method:
partial(object, pred.var, pred.grid, pred.fun = NULL,
grid.resolution = NULL, ice = FALSE, center = 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", palette = c("viridis", "magma", "inferno",
"plasma", "cividis"), alpha = 1, train, cats = NULL,
check.class = TRUE, progress = "none", parallel = FALSE,
paropts = NULL, ...)
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".

palette Character string indicating the colormap option to use. Five options are available: "viridis" (the default), "magma", "inferno", "plasma", and "cividis".

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 Character string giving the name of the progress bar to use while constructing the partial dependence function. See create_progress_bar for details. Default is "none".

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 not to 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


Examples

```r
# 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 = "text")

# 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 autoplot 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 = "text")

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

## End(Not run)

---

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 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


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, alpha = 1, ...)

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

## 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, palette = c("viridis",
  "magma", "inferno", "plasma", "cividis"), alpha = 1, 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.
**plotPartial**

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.

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).

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 Color vector to be used for trivariate displays. If levelplot is TRUE, defaults to the wonderful Matplotlib 'viridis' color map provided by the viridis package. See viridis for details.

palette Character string indicating the colormap option to use. Five options are available: "viridis" (the default), "magma", "inferno", "plasma", and "cividis".

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**

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

# Load required packages
library(ggplot2) # required to use autoplot
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 = "text") %>%
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 = 0.5)
p2 <- plotPartial(age.ice, center = TRUE, alpha = 0.5)
grid.arrange(p1, p2, ncol = 2)

## End(Not run)

---

### progress_progress

**Progress bar**

More informative progress bar powered by `progress_bar`.

**Usage**

```
progress_progress(...)
```

**Arguments**

... Additional optional arguments to be passed onto `progress_bar`.

---

### 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

```r
## 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)

# Top four 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

*Topic datasets
  boston, 4
  pima, 11

 autoplot.cice (autoplot.partial), 2
 autoplot.ice (autoplot.partial), 2
 autoplot.partial, 2

 boston, 4
 BostonHousing2, 4

 co.intervals, 13
 create_progress_bar, 8

equal.count, 13

foreach, 8

geom_smooth, 3

ggplot2, 2, 10

ice, 9

lattice, 8, 10, 12

par, 12
partial, 2, 5, 12
pdp, 10
pdp-package (pdp), 10
pima, 11
PimaIndiansDiabetes2, 11
plot.gbm, 7
plotPartial, 7, 12
predict, 6
progress_bar, 14
progress_progress, 14

topPredictors, 14

varImp, 15

viridis, 13