Package ‘vivid’

August 21, 2021

Title Variable Importance and Variable Interaction Displays

Version 0.2.1

Language en-US

Description A suite of plots for displaying variable importance and two-way variable interaction jointly. Can also display partial dependence plots laid out in a pairs plot or ‘zenplots’ style.

License GPL (>= 2)

Encoding UTF-8

Imports condvis2, ggplot2, GGally, RColorBrewer, colorspace, stats, DendSer, ggalt, dplyr, igraph, flashlight, ggnewscale, tidyr

Suggests intergraph (>= 2.0-2), network (>= 1.12.0), sna (>= 2.3-2), mlr, MASS, tidymodels, e1071, gridExtra, lemon, mlr3, mlr3learners, sp, scales, ranger, vip, knitr, rmarkdown, randomForest, testthat (>= 3.0.0), labeling, zenplots, covr

VignetteBuilder knitr

RoxygenNote 7.1.1

Config/testthat/edition 3

NeedsCompilation no

Author Alan Inglis [aut, cre], Andrew Parnell [aut], Catherine Hurley [aut]

Maintainer Alan Inglis <alan.inglis@mu.ie>

Repository CRAN

Date/Publication 2021-08-21 01:30:02 UTC

R topics documented:

as.data.frame.vivid ........................................... 2
pdpPairs .................................................... 3
pdpVars ..................................................... 5
pdpZen ....................................................... 6
vip2vivid ..................................................... 8
Description

Takes a matrix of class `vivid` and turn it into a data frame containing variable names, Vimp and Vint values, and the row and column index from the original matrix.

Usage

```r
## S3 method for class 'vivid'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

- `x`: A matrix of class `vivid` to be converted to a data frame.
- `row.names`: NULL or a character vector giving the row names for the data frame. Missing values are not allowed.
- `optional`: Logical. If TRUE, setting row names and converting column names (to syntactic names: see `make.names`) is optional. Note that all of R’s base package `as.data.frame()` methods use optional only for column names treatment, basically with the meaning of `data.frame(*, check.names = !optional)`. See also the `make.names` argument of the matrix method.
- `...`: Additional arguments to be passed to or from methods.

Value

A data frame of Vimp and Vint values and their index from the vivid matrix.

Examples

```r
library(ranger)
aq <- na.omit(airquality)
aq <- aq[1:20,] # for speed
rF <- ranger(Ozone ~ ., data = aq, importance = "permutation")
myMat <- vivi(fit = rF, data = aq, response = "Ozone")
myDf <- as.data.frame(myMat)
myDf
```
pdpPairs

Description

Creates a pairs plot showing bivariate pdp on upper diagonal, ice/univariate pdp on the diagonal and data on the lower diagonal

Usage

pdpPairs(
  data,
  fit,
  response,
  vars = NULL,
  pal = rev(RColorBrewer::brewer.pal(11, "RdYlBu")),
  fitlims = "pdp",
  gridSize = 10,
  nmax = 500,
  class = 1,
  nIce = 30,
  comboImage = FALSE,
  predictFun = NULL,
  convexHull = FALSE,
  probability = FALSE
)

Arguments

data  Data frame used for fit.

fit    A supervised machine learning model, which understands condvis2::CVpredict

response  The name of the response for the fit.

vars  The variables to plot (and their order), defaults to all variables other than response.

pal  A vector of colors to show predictions, for use with scale_fill_gradientn

fitlims  Specifies the fit range for the color map. Options are a numeric vector of length 2, "pdp" (default), in which cases limits are calculated from the pdp, or "all", when limits are calculated from the observations and pdp. Predictions outside fitlims are squished on the color scale.

gridSize  The size of the grid for evaluating the predictions.

nmax  Uses sample of nmax data rows for the pdp. Default is 500. Use all rows if NULL.

class  Category for classification, a factor level, or a number indicating which factor level.
nIce Number of ice curves to be plotted, defaults to 30.
comboImage If TRUE draws pdp for mixed variable plots as an image, otherwise an interaction plot.
predictFun Function of (fit, data) to extract numeric predictions from fit. Uses condvis2::CVpredict by default, which works for many fit classes.
convexHull If TRUE, then the convex hull is computed and any points outside the convex hull are removed.
probability if TRUE, then returns the partial dependence for classification on the probability scale. If FALSE (default), then the partial dependence is returned on a near logit scale.

Value
A pairs plot

Examples

# Load in the data:
aq <- na.omit(airquality)
f <- lm(Ozone ~ ., data = aq)
pdpPairs(aq, f, "Ozone")

# Run an mlr ranger model:
library(mlr3)
library(mlr3learners)
library(ranger)
library(MASS)
Boston1 <- Boston[, c(4:6, 8, 13:14)]
Boston1$chas <- factor(Boston1$chas)
task <- TaskRegr$new(id = "Boston1", backend = Boston1, target = "medv")
learner <- lrn("regr.ranger", importance = "permutation")
fit <- learner$train(task)
pdpPairs(Boston1[1:30, ], fit, "medv")
pdpPairs(Boston1[1:30, ], fit, "medv", comboImage = TRUE)
viv <- vivi(Boston1[1:30, ], fit, "medv")
# show top variables only
pdpPairs(Boston1[1:30, ], fit, "medv", comboImage = TRUE, vars = rownames(viv)[1:4])

library(ranger)
rf <- ranger(Species ~ ., data = iris, probability = TRUE)
pdpPairs(iris, rf, "Species") # prediction probs for first class, setosa
pdpPairs(iris, rf, "Species", class = "versicolor") # prediction probs versicolor
# and with mlr3
task <- TaskClassif$new(id = "iris", backend = iris, target = "Species")
fitm <- lrn("classif.ranger", importance = "permutation", predict_type = "prob")$train(task)
pdpPairs(iris, fitm, "Species", class = "versicolor")
### pdpVars

**Description**

Displays the individual conditional expectation (ICE) curves and aggregated partial dependence for each variable in a grid.

**Usage**

```r
pdpVars(
  data,    # Data frame used for fit.
  fit,     # A supervised machine learning model, which understands condvis2::CVpredict
  response,# The name of the response for the fit.
  vars = NULL,  # The variables to plot (and their order), defaults to all variables other than response.
  pal = rev(RColorBrewer::brewer.pal(11, "RdYlBu")),  # A vector of colors to show predictions, for use with scale_fill_gradientn
  gridSize = 10,  # The size of the grid for evaluating the predictions.
  nmax = 500,  # Uses sample of nmax data rows for the pdp. Default is 500. Use all rows if NULL.
  class = 1,  # Category for classification, a factor level, or a number indicating which factor level.
  nIce = 30,  # Number of ice curves to be plotted, defaults to 30.
  predictFun = NULL,  # Function of (fit, data) to extract numeric predictions from fit. Uses condvis2::CVpredict by default, which works for many fit classes.
  limits = NULL,  # A vector determining the limits of the predicted values.
  draw = TRUE,  # A vector determining the limits of the predicted values.
  probability = FALSE
)
```

**Arguments**

- **data**: Data frame used for fit.
- **fit**: A supervised machine learning model, which understands condvis2::CVpredict
- **response**: The name of the response for the fit.
- **vars**: The variables to plot (and their order), defaults to all variables other than response.
- **pal**: A vector of colors to show predictions, for use with scale_fill_gradientn
- **gridSize**: The size of the grid for evaluating the predictions.
- **nmax**: Uses sample of nmax data rows for the pdp. Default is 500. Use all rows if NULL.
- **class**: Category for classification, a factor level, or a number indicating which factor level.
- **nIce**: Number of ice curves to be plotted, defaults to 30.
- **predictFun**: Function of (fit, data) to extract numeric predictions from fit. Uses condvis2::CVpredict by default, which works for many fit classes.
- **limits**: A vector determining the limits of the predicted values.
- **draw**: A vector determining the limits of the predicted values.
- **probability**: A vector determining the limits of the predicted values.
pdpZen

Create a zenplot displaying partial dependence values.

Description

Constructs a zigzag expanded navigation plot (zenplot) displaying partial dependence values.

Usage

pdpZen(
  data,
  fit,
  response,
  zpath = NULL,
  pal = rev(RColorBrewer::brewer.pal(11, "RdYlBu")),
  fitlims = "pdp",
  gridSize = 10,
  nmax = 500,
  class = 1,
)
pdpZen

```r
comboImage = FALSE,
  rug = TRUE,
predictFun = NULL,
  convexHull = FALSE,
  probability = FALSE,
...
)
```

**Arguments**

- **data**: Data frame used for fit
- **fit**: A supervised machine learning model, which understands condvis2::CVpredict
- **response**: The name of the response for the fit
- **zpath**: Plot shows consecutive pairs of these variables. Defaults to all variables other than response. Recommend constructing zpath with calcZpath.
- **pal**: A vector of colors to show predictions, for use with scale_fill_gradientn
- **fitlims**: Specifies the fit range for the color map. Options are a numeric vector of length 2, "pdp" (default), in which cases limits are calculated from the pdp, or "all", when limits are calculated from the observations and pdp predictions outside fitlims are squished on the color scale.
- **gridSize**: The size of the grid for evaluating the predictions.
- **nmax**: Uses sample of nmax data rows for the pdp. Default is 500. Use all rows if NULL.
- **class**: Category for classification, a factor level, or a number indicating which factor level.
- **comboImage**: If TRUE draws pdp for mixed variable plots as an image, otherwise an interaction plot.
- **rug**: If TRUE adds rugs for the data to the pdp plots
- **predictFun**: Function of (fit, data) to extract numeric predictions from fit. Uses condvis2::CVpredict by default, which works for many fit classes.
- **convexHull**: If TRUE, then the convex hull is computed and any points outside the convex hull are removed.
- **probability**: if TRUE, then returns the partial dependence for classification on the probability scale. If FALSE (default), then the partial dependence is returned on a near logit scale.
- **...**: passed on to zenplot

**Value**

A zenplot of partial dependence values.
Examples

```r
## Not run:
# To use this function, install zenplots and graph from Bioconductor.
if (!requireNamespace("graph", quietly = TRUE)) {
  install.packages("BiocManager")
  BiocManager::install("graph")
}
install.packages("zenplots")

library(MASS)
library(ranger)
Boston1 <- Boston
Boston1$chas <- factor(Boston1$chas)
rf <- ranger(medv ~ ., data = Boston1)
pdpZen(Boston1[1:30, ], rf, response = "medv", zpath = names(Boston1)[1:4], comboImage = T)
# Find the top variables in rf
set.seed(123)
viv <- vivi(Boston1, rf, "medv", nmax = 30) # use 30 rows, for speed
pdpZen(Boston1, rf, response = "medv", zpath = rownames(viv)[1:4], comboImage = T)
zpath <- zPath(viv, cutoff = .2) # find plots whose interaction score exceeds .2
pdpZen(Boston1, rf, response = "medv", zpath = zpath, comboImage = T)
## End(Not run)
```

 vip2vivid

---

### Description

@description Takes measured importance and interactions from the vip package and turns them into a matrix which can be used for plotting. Accepts any of the variable importance methods supplied by vip.

### Usage

```r
vip2vivid(importance, interaction, reorder = TRUE)
```

### Arguments

- **importance**: Measured importance from the vip package using `vi` function.
- **interaction**: Measured interaction from the vip package using `vint` function.
- **reorder**: If TRUE (default) uses DendSer to reorder the matrix of interactions and variable importances.

### Value

A matrix of interaction values, with importance on the diagonal.
## Not run:
library(ranger)
library(vip)
aq <- na.omit(airquality) # get data
nameAq <- names(aq[-1]) # get feature names

rF <- ranger(Ozone ~ ., data = aq, importance = "permutation") # create ranger random forest fit
vImp <- vi(rF) # vip importance
vInt <- vint(rF, feature_names = nameAq) # vip interaction

vip2vivid(vImp, vInt)

## End(Not run)

---

### Description

Creates a matrix displaying variable importance on the diagonal and variable interaction on the off-diagonal.

### Usage

```r
vivi(
  data, 
  fit, 
  response, 
  gridSize = 50, 
  importanceType = NULL, 
  nmax = 500, 
  reorder = TRUE, 
  class = 1, 
  predictFun = NULL, 
  normalized = FALSE
)
```

### Arguments

- **data**: Data frame used for fit.
- **fit**: A supervised machine learning model, which understands `condvis2::CVpredict`.
- **response**: The name of the response for the fit.
- **gridSize**: The size of the grid for evaluating the predictions.
- **importanceType**: One of either "%IncMSE" or "IncNodePurity" for use with randomForest. Or set to equal "agnostic" to override embedded importance measures and return agnostic importance values.
nmax

Maximum number of data rows to consider. Default is 500. Use all rows if NULL.

reorder

If TRUE (default) uses DendSer to reorder the matrix of interactions and variable importances.

class

Category for classification, a factor level, or a number indicating which factor level.

predictFun

Function of (fit, data) to extract numeric predictions from fit. Uses condvis2::CVpredict by default, which works for many fit classes.

normalized

Should Friedman’s H-statistic be normalized or not. Default is FALSE.

Value

A matrix of interaction values, with importance on the diagonal.

Examples

```r
aq <- na.omit(airquality)
f <- lm(Ozone ~ ., data = aq)
m <- vivi(fit = f, data = aq, response = "Ozone")  # as expected all interactions are zero
viviHeatmap(m)

# Run an mlr ranger model:
library(mlr3)
library(mlr3learners)
library(ranger)
aq <- na.omit(airquality)
aq_Task <- TaskRegr$new(id = "airQ", backend = aq, target = "Ozone")
aq_lrn <- lrn("regr.ranger", importance = "permutation")
aq_fit <- aq_lrn$train(aq_Task)
m <- vivi(fit = aq_fit, data = aq, response = "Ozone")
viviHeatmap(m)

library(ranger)
rf <- ranger(Species ~ ., data = iris, importance = "impurity", probability = TRUE)
vivi(fit = rf, data = iris, response = "Species")
```

Description

Reorders a square matrix so that values of high importance and interaction strength are pushed to the top left of the matrix.
Usage

vividReorder(d)

Arguments

d A matrix such as that returned by vivi

Value

A reordered version of d.

Examples

f <- lm(Sepal.Length ~ ., data = iris[, -5])
m <- vivi(fit = f, data = iris[, -5], response = "Sepal.Length")
corimp <- abs(cor(iris[, -5])[, -1])
viviUpdate(m, corimp) # use correlation as importance and reorder

viviHeatmap

Description

Plots a Heatmap showing variable importance on the diagonal and variable interaction on the off-diagonal.

Usage

viviHeatmap(
  mat,
  intPal = rev(colorspace::sequential_hcl(palette = "Blues 3", n = 100)),
  impPal = rev(colorspace::sequential_hcl(palette = "Reds 3", n = 100)),
  intLims = NULL,
  impLims = NULL,
  angle = NULL
)

Arguments

mat A matrix, such as that returned by vivi, of values to be plotted.
intPal A vector of colours to show interactions, for use with scale_fill_gradientn.
impPal A vector of colours to show importance, for use with scale_fill_gradientn.
intLims Specifies the fit range for the color map for interaction strength.
impLims Specifies the fit range for the color map for importance.
angle The angle to display the x-axis labels.
Value

A heatmap plot showing variable importance on the diagonal and variable interaction on the off-diagonal.

Examples

```r
library(ranger)
aq <- na.omit(airquality)
rF <- ranger(Ozone ~ ., data = aq, importance = "permutation")
myMat <- vivi(fit = rF, data = aq, response = "Ozone")
viviHeatmap(myMat)
```

Description

Create a Network plot displaying variable importance and variable interaction.

Usage

```r
viviNetwork(mat, intThreshold = NULL, intLims = NULL, impLims = NULL, intPal = rev(colorspace::sequential_hcl(palette = "Blues 3", n = 100)), impPal = rev(colorspace::sequential_hcl(palette = "Reds 3", n = 100)), removeNode = FALSE, layout = igraph::layout_in_circle, cluster = NULL, nudge_x = 0.05, nudge_y = 0.03)
```

Arguments

- **mat**: A matrix, such as that returned by vivi, of values to be plotted.
- **intThreshold**: Remove edges with weight below this value if provided.
- **intLims**: Specifies the fit range for the color map for interaction strength.
- **impLims**: Specifies the fit range for the color map for importance.
- **intPal**: A vector of colours to show interactions, for use with `scale_fill_gradientn`.
- **impPal**: A vector of colours to show importance, for use with `scale_fill_gradientn`. 

viviUpdate

removeNode If TRUE, then removes nodes with no connecting edges when thresholding interaction values.

layout igraph layout function or a numeric matrix with two columns, one row per node. Defaults to igraph::layout_as_circle

cluster Either a vector of cluster memberships for nodes or an igraph clustering function.

nudge_x Nudge (centered) labels by this amount, outward horizontally

nudge_y Nudge (centered) labels by this amount, outward vertically

Value

A plot displaying interaction strength between variables on the edges and variable importance on the nodes.

Examples

library(ranger)
aq <- na.omit(airquality)
rF <- ranger(Ozone ~ ., data = aq, importance = "permutation")
myMat <- vivi(fit = rF, data = aq, response = "Ozone")
viviNetwork(myMat)

viviUpdate(viviUpdate)

Description

Creates a matrix displaying updated variable importance on the diagonal and variable interaction on the off-diagonal.

Usage

viviUpdate(mat, newImp, reorder = TRUE)

Arguments

mat A matrix, such as that returned by vivi.

newImp A named vector of variable importances.

reorder If TRUE (default) uses DendSer to reorder the matrix of interactions and variable importances.

Value

A matrix of values, of class vivid, with updated variable importances.
Examples

```r
f <- lm(Sepal.Length ~ ., data = iris[, -5])
m <- vivi(iris[, -5], f, "Sepal.Length")
corimp <- abs(cor(iris[, -5])[1, -1])

viviUpdate(m, corimp) # use correlation as updated importance
```

---

**zPath**

Constructs a zenpath for connecting and displaying pairs.

**Usage**

```r
zPath(
  viv,
  cutoff = NULL,
  method = c("greedy.weighted", "strictly.weighted"),
  connect = TRUE
)
```

**Arguments**

- **viv**: A matrix, created by vivi to be used to calculate the path.
- **cutoff**: Do not include any variables that are below the cutoff interaction value.
- **method**: String indicating the method to use. The available methods are: "greedy.weighted": Sort all pairs according to a greedy (heuristic) Euler path with x as weights visiting each edge precisely once. "strictly.weighted": Strictly respect the order of the weights - so the first, second, third, and so on, adjacent pair of numbers of the output of zenpath() corresponds to the pair with largest, second-largest, third-largest, and so on, weight. see zenpath
- **connect**: If connect is TRUE, connect the edges from separate eulerians (strictly.weighted only).

**Details**

Construct a path of indices to visit to order variables

**Value**

Returns a zpath from viv showing pairs with viv entry over the cutoff
Examples

```
## Not run:
# To use this function, install zenplots and graph from Bioconductor.
if (!requireNamespace("graph", quietly = TRUE)) {
  install.packages("BiocManager")
  BiocManager::install("graph")
}
install.packages("zenplots")

aq <- na.omit(airquality) * 1.0

# Run an mlr3 ranger model:
library(mlr3)
library(mlr3learners)
library(ranger)
ozonet <- TaskRegr$new(id = "airQ", backend = aq, target = "Ozone")
ozonel <- lrn("regr.ranger", importance = "permutation")
ozonef <- ozonel$train(ozonet)

viv <- vivi(aq, ozonef, "Ozone")

# Calculate Zpath:
zpath <- zPath(viv, 0.8)
zpath

## End(Not run)
```
Index

as.data.frame.vivid, 2
pdpPairs, 3
pdpVars, 5
pdpZen, 6

vip2vivid, 8
vivi, 9
vividReorder, 10
viviHeatmap, 11
viviNetwork, 12
viviUpdate, 13

zPath, 14