Package 'classmap'

May 8, 2026

Type Package

Title Visualizing Classification Results

Date 2026-04-28

Version 1.2.7

Author Jakob Raymaekers [aut, cre],
Peter Rousseeuw [aut]

Depends R (>= 4.1.0)

Suggests knitr, reshape2, svd, rpart.plot, nnet, robCompositions, rmarkdown, fairml, robustHD, vioplot

Imports stats, graphics, ggplot2, robustbase, e1071, cellWise, cluster, kernlab, gridExtra, rpart, randomForest, scales, lpSolve, diptest

Maintainer Jakob Raymaekers <jakob.raymaekers@kuleuven.be>

Description Tools to visualize the results of a classification or a regression.
The graphical displays include stacked plots, silhouette plots, quasi residual plots, class maps, predictions plots, and predictions correlation plots.
Implements the techniques described and illustrated in Raymaekers J., Rousseeuw P.J., Hubert M. (2022). Class maps for visualizing classification results. \emph{Technometrics}, 64(2), 151–165. <doi:10.1080/00401706.2021.1927849> (open access), Raymaekers J., Rousseeuw P.J.(2022). Silhouettes and quasi residual plots for neural nets and tree-based classifiers. \emph{Journal of Computational and Graphical Statistics}, 31(4), 1332–1343. <doi:10.1080/10618600.2022.2050249>, and Rousseeuw, P.J. (2026). Explainable Linear and Generalized Linear Models by the Predictions Plot. The American Statistician, 80, 157-163, <doi:10.1080/00031305.2025.2539235> (open access), and Montalcini, C., Rousseeuw, P.J. (2025). The bixplot: A variation on the boxplot suited for bimodal data, <doi:10.48550/arXiv.2510.09276> (open access).
Examples can be found in the vignettes:
``Discriminant_analysis_examples",``K_nearest_neighbors_examples",
``Support_vector_machine_examples", ``Rpart_examples", ``Random_forest_examples",
``Neural_net_examples", ``predsplot_examples", and ``bixplot_examples".

URL <doi:10.1080/00401706.2021.1927849>,
<doi:10.1080/10618600.2022.2050249>,
<doi:10.1080/00031305.2025.2539235>,
<doi:10.48550/arXiv.2510.09276>

License GPL (>= 2)

Encoding UTF-8

LazyData true

VignetteBuilder knitr, rmarkdown

RoxygenNote 7.1.2

NeedsCompilation no

Repository CRAN

Date/Publication 2026-04-29 11:40:08 UTC

Contents

`bixplot` *Boxplot version suited for bimodal and multimodal data, combining density, box, and rug elements with automatic cluster detection*

Description

Draws a bixplot for one or more numeric variables. A bixplot extends the violin plot and boxplot by automatically testing each variable for unimodality (via Hartigan's dip test) and, when multimodality is detected, fitting a constrained k-medoids clustering to identify and separately display the modes. Each variable is rendered as a filled density body, a box-and-whisker summary, and a rug of individual data values. The rug can optionally be colored by an external numeric or factor variable. Input can be provided as vectors, a data frame, a matrix, a list of vectors, or a formula.

Usage

```
bixplot(...,
        names = NA,
        add = FALSE,
        at = NULL,
        horizontal = FALSE,
        col = "gray60", bodyCol = NULL,
        bodyOpaque = 0.5,
        bodyW = 0.80,
        bodysize = "area_from_count",
        modeCol = c("cadetblue3", "hotpink2",
                    "lawngreen", "orange", "cyan3",
                    "coral2", "gray60", "darkgoldenrod2",
                    "slateblue2", "brown2", "khaki3"),
        curveCol = "black",
        curveLwd = 1,
        border = "black", boxCol = NULL,
        boxOpaque = 0.5,
        boxW = 0.32,
        boxLwd = 2,
        makewhiskers = TRUE,
        innerwhiskers = TRUE,
        rugCol = "black",
        rugoutCol = par("fg"),
        rugNumeric = NULL,
        rugNumericColors = c("blue", "green"),
        colorbarW = 0.12,
        rugFactor = NULL,
        rugFactorColors = c("red", "blue", "forestgreen"),
        rugLwd = par("lwd"),
        rugoutLwd = 2 * rugLwd,
        rugOpaque = 0.4,
        jittering = TRUE,
```

```
          jitteramount = NULL,
          rugW = 0.12,
          stickCol = "black",
          stickLwd = 2,
          boxwex = 1,
          width = NULL,
          side = "no",
          tick = TRUE,
          diplevel = 0.01,
          minN = 15,
          kmax = NULL,
          bigN = 500,
          clusMinN = 3,
          bw = "SJ-dpi",
          kernel = "gaussian",
          cut = 3,
          cutmin = -Inf,
          cutmax = Inf,
          xlim = NULL,
          ylim = NULL,
          cex.axis = 1,
          main = "bixplot",
          cex.main = 1, line.main = 1,
          cex.colorbar = 1,
          xlab = "",
          ylab = "",
          cex.lab = 1, line.lab = 2.2,
          xaxs = "r", yaxs = "r", las = 1,
          ann = !add,
          plot = TRUE,
          log = NULL)
```

Arguments

`...`	one or more numeric vectors to plot, or a single data frame or matrix whose columns are plotted, or a single list of numeric vectors (which may have different lengths), or a formula such as `y ~ grp` where `y` is a numeric vector to be split into groups by the grouping variable `grp` (usually a factor). Note that `~ g1 + g2` is equivalent to `~ g1:g2`. When a formula is used it may be necessary to also supply a `data =` argument. Missing values are silently removed.
`names`	character vector of group labels printed next to each variable. If `NA` (the default), labels are extracted from the input data; if the data carry no names the integers `1, 2, 3, ...` are used. [Argument from boxplot.]
`add`	logical. If `TRUE`, the bixplot is added to the current plot without drawing new axes. Defaults to `FALSE`. [Argument from `boxplot`.]
`at`	numeric vector specifying the axis positions at which the bixplots are drawn, particularly useful when `add = TRUE`. Defaults to `1:p` where `p` is the number of variables. [Argument from `boxplot`.]

`horizontal` logical. If `TRUE`, bixplots are drawn horizontally. Defaults to `FALSE` (vertical). [Argument from `boxplot`.]

`col` color(s) for the density body of variables deemed unimodal. Can be a single color or a vector recycled to length `p`. When `NA`, the body is not filled. [Argument from `boxplot`.]

`bodyCol` if not `NULL`, overrides `col` for the body fill color.

`bodyOpaque` numeric between 0 (transparent) and 1 (opaque) controlling the opacity of the filled density body. Defaults to `0.5`.

`bodyW` numeric vector (recycled to length `p`) giving the width of the widest density body for each variable. When two or more modes are shown for a variable, this is the width of the widest mode body. Set to zero to suppress the body entirely. Defaults to `0.80`.

`bodysize` character string determining how the density bodies of individual modes within the same variable are sized relative to each other. One of `"area_from_count"` (the default; body area proportional to cluster membership count), `"area_is_constant"` (all modes have equal area), or `"width_is_constant"` (all modes have equal width). In all cases the widest body has width `bodyW[j]`, possibly scaled by `width`.

`modeCol` color(s) of the density bodies for variables with more than one detected mode (cluster). Cycled across all modes of all multimodal variables. If `NULL`, each mode inherits the `col` color of its variable.

`curveCol` color(s) of the density curve boundary drawn around each body. Can be a single color or a vector recycled to length `p`. Set to `NA` to suppress the curve. Defaults to `"black"`.

`curveLwd` single number giving the line width of the density curve.

`border` color(s) of the box and whiskers. Can be a single color or a vector recycled to length `p`. Set to `NA` to suppress the box. [Argument from `boxplot`.]

`boxCol` if not `NULL`, overrides `border` for the box color.

`boxOpaque` numeric between 0 and 1 controlling the opacity of the lines making up the boxplot. Defaults to `0.5`.

`boxW` numeric vector (recycled to length `p`) giving the width of the interquartile box for each variable. Set to zero to suppress the box. Defaults to `0.32`.

`boxLwd` single number giving the line width of the box and whiskers.

`makewhiskers` logical. If `TRUE` (the default), whiskers are drawn extending from the box to the most extreme non-outlying values.

`innerwhiskers` logical. Relevant only when a variable has more than one mode. If `TRUE`, whiskers are also drawn between adjacent modes. If `FALSE`, inner whiskers are omitted because their interpretation is ambiguous when mode densities overlap.

`rugCol` color(s) of the rug tick marks for each variable. Can be a single color or a vector recycled to length `p`. Set to `NA` to suppress the rug. Defaults to `"black"`.

`rugoutCol` color(s) of the portion of rug lines that extends outside the density body. If `NULL` or `NA`, the same color as `rugCol` is used throughout. Defaults to `par("fg")`.

`rugNumeric` optional numeric vector of the same length as each `y` variable (requires all variables to have the same length) used to color the rug lines via a continuous color palette. Cannot be combined with `rugFactor`.

`rugNumericColors` vector of two or three colors passed to `colorRampPalette` to construct the palette for `rugNumeric`. Defaults to `c("blue", "green")`.

`colorbarW` width of the color bar legend for `rugNumeric` relative to the main plot width. Only has an effect when `rugNumeric` is specified. Defaults to `0.12`.

`rugFactor` optional factor variable of the same length as each `y` variable used to color the rug lines by factor level. Cannot be combined with `rugNumeric`.

`rugFactorColors` character vector of colors for the levels of `rugFactor`, recycled as needed. Defaults to `c("red", "blue", "forestgreen")`.

`rugLwd` single number giving the line width of the rug marks. Defaults to `par("lwd")`.

`rugoutLwd` single number giving the line width of the part of rug lines outside the density body. Defaults to `2 * rugLwd` to make isolated points visually prominent.

`rugOpaque` numeric between 0 and 1 controlling the opacity of the rug marks. Defaults to `0.4`.

`jittering` logical. If `TRUE` (the default), rug tick positions are jittered via `jitter` to make tied values distinguishable.

`jitteramount` amount of jittering passed to `jitter`. If `NULL`, the default amount is used.

`rugW` numeric vector (recycled to length `p`) giving the width of the rug (i.e. the length of each tick mark). Set to zero to suppress the rug. Defaults to `0.12`.

`stickCol` color(s) of the vertical or horizontal "stick" drawn when `side = "both"`, which separates the two half-bixplots sharing an axis. Set to `NA` to suppress the stick. Can be a single color or a vector recycled to length `p`. Defaults to `"black"`.

`stickLwd` single number giving the line width of the stick. Set to zero to suppress the stick. Defaults to `2`.

`boxwex` scale factor applied uniformly to the widths of the body, box and rug across all bixplots. [Argument from `boxplot`.] Defaults to `1`.

`width` optional numeric vector of length `p` giving the relative widths of the bixplots. Entries must be strictly positive; they are divided by their maximum so that the widest bixplot has relative width 1. The resulting ratios multiply the body, box and rug widths. If `NULL` (the default), all bixplots have equal width. [Argument from `boxplot`.]

`side` character string specifying which side of the variable axis the body, box and rug are drawn on. One of `"no"` (the default; each bixplot is symmetric about its axis), `"first"` or `"second"` (all half-bixplots on that side), or `"both"` (adjacent variables are plotted on alternate sides of shared axes, halving the number of axes needed). [Argument from `beanplot`.]

`tick` logical indicating whether tick marks are drawn on the group-label axis. Only has an effect when `add = FALSE`. Defaults to `TRUE`.

`diplevel` significance level for Hartigan's dip test for unimodality. A cluster search is only performed for variables whose dip test p-value is at most `diplevel`. Defaults to `0.01`. Increasing this value may yield more clusters; decreasing it fewer.

minN	minimum number of observations required per potential cluster. Defaults to 15. The maximum number of clusters searched is bounded by floor(n / minN). If n < 2 * minN, clustering is not attempted.
kmax	maximum number of clusters to consider. Internally capped at 5. If NULL (the default), it is set to min(floor(n / minN), 5). Setting kmax = 1 treats all variables as unimodal and the display resembles a violin plot.
bigN	when a variable has more than bigN non-missing values, a sample of bigN observations (always including the minimum and maximum) is drawn without replacement before computing densities and clusters, to reduce computation time. Defaults to 500; values below 300 are silently raised to 300.
clusMinN	minimum number of unique values that each cluster must contain. The constrained clustering enforces this bound. Defaults to 3.
bw	bandwidth for density, used to construct the density body. Can be a numeric value or a character string naming a bandwidth selector (see ?density). Defaults to "SJ-dpi".
kernel	kernel for density. Defaults to "gaussian".
cut	the density is computed from min(y) - cut * bw to max(y) + cut * bw where bw is the numeric bandwidth. Defaults to 3. [Argument from beanplot.]
cutmin	if finite, the density of every variable and mode is truncated to begin no lower than cutmin. Defaults to -Inf (no effect). [Argument from beanplot.]
cutmax	if finite, the density of every variable and mode is truncated to end no higher than cutmax. Defaults to Inf (no effect). [Argument from beanplot.]
xlim	numeric vector of length 2 giving the limits of the group axis (whether horizontal is TRUE or FALSE). If NULL, limits are set automatically. [Convention from boxplot.]
ylim	numeric vector of length 2 giving the limits of the numeric value axis (whether horizontal is TRUE or FALSE). If NULL, limits are set automatically. [Convention from boxplot.]
cex.axis	character expansion factor for axis tick labels.
main	title of the plot. Defaults to "bixplot".
cex.main	character expansion factor for the title.
line.main	margin line for the title (passed to title).
cex.colorbar	character expansion factor for the color bar axis labels when rugNumeric is used.
xlab	label for the horizontal axis. Defaults to "".
ylab	label for the vertical axis. Defaults to "".
cex.lab	character expansion factor for axis labels.
line.lab	margin line for axis labels. Defaults to 2.2.
xaxs	axis interval calculation style for the x-axis (see par). Defaults to "r".
yaxs	axis interval calculation style for the y-axis. Defaults to "r".
las	orientation of axis tick labels (see par). Defaults to 1 (always horizontal).

ann	logical indicating whether the plot should be annotated with xlab, ylab and main. Defaults to !add. [Argument from boxplot.]
plot	if TRUE (the default), the summary list is returned invisibly. If FALSE, it is also printed to the console. [Argument from boxplot.]
log	this argument from boxplot must be NULL. If a non-NULL value is supplied, a warning is issued and the argument is ignored, because a log transformation can change the number and position of modes. Apply the log transform explicitly to the data before calling bixplot if desired.

Details

For each variable, bixplot proceeds as follows. Hartigan's dip test is applied to test for unimodality; if the p-value exceeds diplevel, or if the sample is too small (n < 2 * minN), the variable is treated as unimodal (k = 1). Otherwise, constrained k-medoids clustering (via pamc1d) is fitted for k = 2, ..., kmax clusters, and the best k is selected by the highest mean silhouette width (computed via silhouette). If no k > 1 yields a positive mean silhouette width, the variable is treated as unimodal.

A single global bandwidth is computed once per variable and reused for the density of every mode, ensuring that density bodies are comparable across modes. Mode sizes are scaled according to bodysize.

When side = "both", adjacent variables are paired and plotted as half-bixplots on opposite sides of shared axes, and axis labels are combined automatically from the names of each pair.

The arguments rugNumeric and rugFactor require all variables to have the same number of observations. They cannot be specified simultaneously.

Value

A list returned invisibly when plot = TRUE (or visibly when plot = FALSE), containing:

call	the matched call.
p	the number of variables plotted.
<name_1>, <name_2>, ...	one list entry per variable, named after the variable. For a unimodal variable the list contains values (the sorted non-missing observations) and fivenumbersummary (the five-number summary from fivenum used to draw the box). For a multimodal variable the list additionally contains clustering (an integer vector of cluster assignments) and one sub-list per cluster named cluster_1, cluster_2, ..., each containing members and fivenumbersummary for that cluster.

Author(s)

P.J. Rousseeuw

References

Montalcini, C., Rousseeuw, P.J. (2025). The bixplot: A variation on the boxplot suited for bimodal data, doi:10.48550/arXiv.2510.09276 (open access).

See Also

pamc1d, pam, silhouette, dip.test, density, boxplot

Examples

```
set.seed(1)
# A unimodal and a clearly bimodal variable
x1 <- rnorm(100)
x2 <- c(rnorm(60, mean = -3), rnorm(60, mean = 3))
bixplot(x1, x2, names = c("unimodal", "bimodal"),
        main = "Basic bixplot example")

# Formula interface, coloring rug by a factor
n <- 150
grp <- factor(rep(c("A", "B", "C"), each = 50))
y   <- c(rnorm(50, 0), rnorm(50, 4), rnorm(50, 8))
bixplot(y ~ grp, main = "Formula interface")

# Horizontal layout with an external numeric rug variable
set.seed(42)
vals <- c(rnorm(80, mean = -2), rnorm(80, mean = 2))
covariate <- runif(160)
bixplot(vals, horizontal = TRUE,
        rugNumeric = covariate,
        rugNumericColors = c("purple", "yellow"),
        main = "Horizontal bixplot with numeric rug")

# Side-by-side ("both") half-bixplots
bixplot(x1, x2, side = "both",
        names = c("unimodal vs bimodal"),
        main = "side = both")

# For more examples, we refer to the vignette:
## Not run:
vignette("bixplot_examples")

## End(Not run)
```

`classmap` *Draw the class map to visualize classification results.*

Description

Draw the class map to visualize classification results, based on the output of one of the `vcr.*.*` functions in this package. The vertical axis of the class map shows each case's `PAC`, the conditional probability that it belongs to an alternative class. The `farness` on the horizontal axis is the probability of a member of the given class being at most as far from the class as the case itself.

Usage

```
classmap(vcrout, whichclass, classLabels = NULL, classCols = NULL,
         main = NULL, cutoff = 0.99, plotcutoff = TRUE,
         identify = FALSE, cex = 1, cex.main = 1.2, cex.lab = NULL,
         cex.axis = NULL, opacity = 1,
         squareplot = TRUE,  maxprob = NULL, maxfactor = NULL)
```

Arguments

vcrout	output of vcr.*.train or vcr.*.newdata. Required.
whichclass	the number or level of the class to be displayed. Required.
classLabels	the labels (levels) of the classes. If NULL, they are taken from vcrout.
classCols	a list of colors for the class labels. There should be at least as many as there are levels. If NULL the classCols are taken as 2, 3, 4, ...
main	title for the plot.
cutoff	cases with overall farness vcrout$ofarness > cutoff are flagged as outliers.
plotcutoff	If true, plots the cutoff on the farness values as a vertical line.
identify	if TRUE, left-click on a point to get its number, then ESC to exit.
cex	passed on to graphics::plot.
cex.main	same, for title.
cex.lab	same, for labels on horizontal and vertical axes.
cex.axis	same, for axes.
opacity	determines opacity of plotted dots. Value between 0 and 1, where 0 is transparent and 1 is opaque.
squareplot	If TRUE, makes the axes of the plot equally long.
maxprob	draws the farness axis at least upto probability maxprob. If NULL, the limits are obtained automatically.
maxfactor	if not NULL, a number slightly higher than 1 to increase the space at the right hand side of the plot, to make room for marking points.

Value

Executing the function plots the class map and returns

coordinates	a matrix with 2 columns containing the coordinates of the plotted points. The first coordinate is the quantile of the farness probability. This makes it easier to add text next to interesting points. If identify = T, the attribute ids of coordinates contains the row numbers of the identified points in the matrix coordinates.

Author(s)

Raymaekers J., Rousseeuw P.J.

References

Raymaekers J., Rousseeuw P.J., Hubert M. (2021). Class maps for visualizing classification results. *Technometrics*, 64(2), 151–165. doi:10.1080/00401706.2021.1927849

Raymaekers J., Rousseeuw P.J.(2021). Silhouettes and quasi residual plots for neural nets and tree-based classifiers. *Journal of Computational and Graphical Statistics*, 31(4), 1332–1343. doi:10.1080/10618600.2022.2050249

See Also

vcr.da.train, vcr.da.newdata,
vcr.knn.train, vcr.knn.newdata,
vcr.svm.train, vcr.svm.newdata,
vcr.rpart.train, vcr.rpart.newdata,
vcr.forest.train, vcr.forest.newdata,
vcr.neural.train, vcr.neural.newdata

Examples

```
vcrout <- vcr.da.train(iris[, 1:4], iris[, 5])
classmap(vcrout, "setosa", classCols = 2:4) # tight class
classmap(vcrout, "versicolor", classCols = 2:4) # less tight
# The cases misclassified as virginica are shown in blue.
classmap(vcrout, "virginica", classCols = 2:4)
# The case misclassified as versicolor is shown in green.

# For more examples, we refer to the vignettes:
## Not run:
vignette("Discriminant_analysis_examples")
vignette("K_nearest_neighbors_examples")
vignette("Support_vector_machine_examples")
vignette("Rpart_examples")
vignette("Random_forest_examples")
vignette("Neural_net_examples")

## End(Not run)
```

---

confmat.vcr *Build a confusion matrix from the output of a function* vcr.*.*.

Description

Build a confusion matrix from the output of a function vcr.*.*. Optionally, a separate column for outliers can be added to the confusion matrix.

Usage

```
confmat.vcr(vcrout, cutoff = 0.99, showClassNumbers = FALSE,
            showOutliers = TRUE, silent = FALSE)
```

Arguments

vcrout	output of vcr.*.train or vcr.*.newdata.
cutoff	cases with overall farness vcrout$ofarness > cutoff are flagged as outliers.
showClassNumbers	if TRUE, the row and column names are the number of each level instead of the level itself. Useful for long level names.
showOutliers	if TRUE and some points were flagged as outliers, it adds an extra column on the right of the confusion matrix for these outliers, with label "outl".
silent	if FALSE, the confusion matrix and accuracy are shown on the screen.

Value

A confusion matrix

Author(s)

Raymaekers J., Rousseeuw P.J.

References

Raymaekers J., Rousseeuw P.J., Hubert M. (2021). Class maps for visualizing classification results. *Technometrics*, 64(2), 151–165. doi:10.1080/00401706.2021.1927849

See Also

vcr.da.train, vcr.da.newdata,
vcr.knn.train, vcr.knn.newdata,
vcr.svm.train, vcr.svm.newdata,
vcr.rpart.train, vcr.rpart.newdata,
vcr.forest.train, vcr.forest.newdata,
vcr.neural.train, vcr.neural.newdata

Examples

```
vcrout <- vcr.knn.train(scale(iris[, 1:4]), iris[, 5], k = 5)
# The usual confusion matrix:
confmat.vcr(vcrout, showOutliers = FALSE)

# Cases with ofarness > cutoff are flagged as outliers:
confmat.vcr(vcrout, cutoff = 0.98)

# With the default cutoff = 0.99 only one case is flagged here:
confmat.vcr(vcrout)
# Note that the accuracy is computed before any cases
# are flagged, so it does not depend on the cutoff.

confmat.vcr(vcrout, showClassNumbers = TRUE)
# Shows class numbers instead of labels. This option can
# be useful for long level names.
```

```
# For more examples, we refer to the vignettes:
## Not run:
vignette("Discriminant_analysis_examples")
vignette("K_nearest_neighbors_examples")
vignette("Support_vector_machine_examples")
vignette("Rpart_examples")
vignette("Random_forest_examples")
vignette("Neural_net_examples")

## End(Not run)
```

`data_bookReviews` *Amazon book reviews data*

Description

This is a subset of the data used in the paper, which was assembled by Prettenhofer and Stein (2010). It contains 1000 reviews of books on Amazon, of which 500 were selected from the original training data and 500 from the test data.

The full dataset has been used for a variety of things, including classification using svm. The subset was chosen small enough to keep the computation time low, while still containing the examples in the paper.

Usage

```
data("data_bookReviews")
```

Format

A data frame with 1000 observations on the following 2 variables.

`review` the review in text format (character)

`sentiment` factor indicating the sentiment of the review: negative (1) or positive (2)

Source

Prettenhofer, P., Stein, B. (2010). Cross-language text classification using structural correspondence learning. *Proceedings of the 48th annual meeting of the association for computational linguistics*, 1118-1127.

Examples

```
data(data_bookReviews)
# Example review:
data_bookReviews[5, 1]

# The data are used in:
## Not run:
```

```
vignette("Support_vector_machine_examples")

## End(Not run)
```

`data_floralbuds` *Floral buds data*

Description

This data on floral pear bud detection was first described by Wouters et al. The goal is to classify the instances into buds, branches, scales and support. The numeric vectors resulted from a multispectral vision sensor and describe the scanned images.

Usage

```
data("data_floralbuds")
```

Format

A data frame with 550 observations on the following 7 variables.

`X1` numeric vector

`X2` numeric vector

`X3` numeric vector

`X4` numeric vector

`X5` numeric vector

`X6` numeric vector

`y` a factor with levels `branch` bud `scales` `support`

Source

Wouters, N., De Ketelaere, B., Deckers, T. De Baerdemaeker, J., Saeys, W. (2015). Multispectral detection of floral buds for automated thinning of pear. *Comput. Electron. Agric.* 113, C, 93–103. <doi:10.1016/j.compag.2015.01.015>

Examples

```
data("data_floralbuds")
str(data_floralbuds)
summary(data_floralbuds)

# The data are used in:
## Not run:
vignette("Discriminant_analysis_examples")
vignette("Neural_net_examples")

## End(Not run)
```

`data_instagram` *Instagram data*

Description

This dataset contains information on fake (spam) accounts on Instagram. The original source is https://www.kaggle.com/free4ever1/instagram-fake-spammer-genuine-accounts by Bardiya Bakhshandeh.

The data contains information on 696 Instagram accounts. For each account, 11 variables were recorded describing its characteristics. The goal is to detect fake instagram accounts, which are used for spamming.

Usage

```
data("data_instagram")
```

Format

A data frame with 696 observations on the following variables.

profile.pic binary, indicates whether profile has picture.

nums.length.username ratio of number of numerical chars in username to its length.

fullname.words number of words in full name.

nums.length.fullname ratio of number of numerical characters in full name to its length.

name..username binary, indicates whether the name and username of the profile are the same.

description.length length of the description/biography of the profile (in number of characters).

external.URL binary, indicates whether profile has external url.

private binary, indicates whether profile is private or not.

X.posts number of posts made by profile.

X.followers number of followers.

X.follows numbers of follows.

y whether profile is fake or not.

dataType vector taking the values "train" or "test" indicating whether the observation belongs to the training or the test data.

Source

https://www.kaggle.com/free4ever1/instagram-fake-spammer-genuine-accounts

Examples

```
data(data_instagram)
str(data_instagram)

# The data are used in:
## Not run:
vignette("Random_forest_examples")

## End(Not run)
```

`data_latenc` *Latenc data*

Description

Latency times in seconds of exploratory behavior in fish during three experimental rounds.

Usage

```
data("data_latenc")
```

Format

A data frame with 40 observations on the following variables.

round 1 Latency times for round 1 of the experiment.

round 2 Latency times for round 1 of the experiment.

round 3 Latency times for round 1 of the experiment.

Source

Kerr, Nicky R., and Travis Ingram. "Personality does not predict individual niche variation in a freshwater fish." Behavioral Ecology 32.1 (2021): 159-167.

Examples

```
data(data_latenc)
str(data_latenc)

# The data are used in:
## Not run:
vignette("bixplot_examples")

## End(Not run)
```

`data_titanic` *Titanic data*

Description

This dataset contains information on 1309 passengers of the RMS Titanic. The goal is to predict survival based on 11 characteristics such as the travel class, age and sex of the passengers.

The original data source is https://www.kaggle.com/c/titanic/data

The data is split up in a training data consisting of 891 observations and a test data of 418 observations. The response in the test set was obtained by combining information from other data files, and has been verified by submitting it as a 'prediction' to kaggle and getting perfect marks.

Usage

```
data("data_titanic")
```

Format

A data frame with 1309 observations on the following variables.

PassengerId a unique identified for each passenger.

Pclass travel class of the passenger.

Name name of the passenger.

Sex sex of the passenger.

Age age of the passenger.

SibSp number of siblings and spouses traveling with the passenger.

Parch number of parents and children traveling with the passenger.

Ticket Ticket number of the passenger.

Fare fare paid for the ticket.

Cabin cabin number of the passenger.

Embarked Port of embarkation. Takes the values C (Cherbourg), Q (Queenstown) and S (Southampton).

y factor indicating casualty or survivor.

dataType vector taking the values "train" or "test" indicating whether the observation belongs to the training or the test data.

Source

https://www.kaggle.com/c/titanic/data

Examples

```
data("data_titanic")
traindata <- data_titanic[which(data_titanic$dataType == "train"), -13]
testdata <- data_titanic[which(data_titanic$dataType == "test"), -13]
str(traindata)
table(traindata$y)

# The data are used in:
## Not run:
vignette("Rpart_examples")

## End(Not run)
```

---

makeFV *Constructs feature vectors from a kernel matrix.*

Description

Constructs feature vectors from a kernel matrix.

Usage

```
makeFV(kmat, transfmat = NULL, precS = 1e-12)
```

Arguments

kmat	a kernel matrix. If transfmat is NULL, we are dealing with training data and then kmat must be a square kernel matrix (of size n by n when there are n cases). Such a PSD matrix kmat can e.g. be produced by makeKernel or by kernlab::kernelMatrix. If on the other hand transfmat is not NULL, we are dealing with a test set. See details for the precise working.
transfmat	transformation matrix. If not NULL, it is the value transfmat of makeFV on training data. It has to be a square matrix, with as many rows as there were training data.
precS	if not NULL, eigenvalues of kmat below precS will be set equal to precS.

Details

If transfmat is non-NULL, we are dealing with a test set. Denote the number of cases in the test set by $m \geq 1$. Each row of kmat of the test set then must contain the kernel values of a new case with all cases in the training set. Therefore the kernel matrix kmat must have dimensions m by n. The matrix kmat can e.g. be produced by makeKernel. It can also be obtained by running kernlab::kernelMatrix on the union of the training set and the test set, yielding an $(n + m)$ by $(n + m)$ matrix, from which one then takes the $[(n + 1) : m, 1 : n]$ submatrix.

Value

A list with components:

`Xf`	When makeKV is applied to the training set, `Xf` has coordinates of n points (vectors), the plain inner products of which equal the kernel matrix of the training set. That is, `kmat` = `Xf` `Xf`'. The `Xf` are expressed in an orthogonal basis in which the variance of the coordinates is decreasing, which is useful when plotting the first few coordinates. When `makeFV` is applied to a test set, `Xf` are coordinates of the feature vectors of the test set in the same space as those of the training set, and then `kmat` = `Xf` %*% `Xf` of `training` `data`.
`transfmat`	square matrix for transforming kmat to `Xf`.

Author(s)

Raymaekers J., Rousseeuw P.J., Hubert, M.

References

Raymaekers J., Rousseeuw P.J., Hubert M. (2021). Class maps for visualizing classification results. *Technometrics*, 64(2), 151–165. doi:10.1080/00401706.2021.1927849

See Also

`makeKernel`

Examples

```
library(e1071)
set.seed(1); X <- matrix(rnorm(200 * 2), ncol = 2)
X[1:100, ] <- X[1:100, ] + 2
X[101:150, ] <- X[101:150, ] - 2
y <- as.factor(c(rep("blue", 150), rep("red", 50)))
cols <- c("deepskyblue3", "red")
plot(X, col = cols[as.numeric(y)], pch = 19)
# We now fit an SVM with radial basis kernel to the data:
svmfit <- svm(y~.,  data = data.frame(X = X, y = y),  scale = FALSE,
             kernel = "radial", cost = 10, gamma = 1, probability = TRUE)
Kxx <- makeKernel(X, svfit = svmfit)
outFV <- makeFV(Kxx)
Xf <- outFV$Xf # The data matrix in this feature space.
dim(Xf) # The feature vectors are high dimensional.
# The inner products of Xf match the kernel matrix:
max(abs(as.vector(Kxx - crossprod(t(Xf), t(Xf))))) # 3.005374e-13 # tiny, OK
range(rowSums(Xf^2)) # all points in Xf lie on the unit sphere.
pairs(Xf[, 1:5], col = cols[as.numeric(y)])
# In some of these we see spherical effects, e.g.
plot(Xf[, 1], Xf[, 5], col = cols[as.numeric(y)], pch = 19)
# The data look more separable here than in the original
# two-dimensional space.
```

```
# For more examples, we refer to the vignette:
## Not run:
vignette("Support_vector_machine_examples")

## End(Not run)
```

`makeKernel` *Compute kernel matrix*

Description

Computes kernel value or kernel matrix, where the kernel type is extracted from an svm trained by `e1071::svm`.

Usage

```
makeKernel(X1, X2 = NULL, svfit)
```

Arguments

`X1`	first matrix (or vector) of coordinates.
`X2`	if not `NULL`, second data matrix or vector. If NULL, `X2` is assumed equal to `X1`.
`svfit`	output from `e1071::svm`
.	

Value

the kernel matrix, of dimensions `nrow(X1)` by `nrow(X2)`. When both `X1` and `X2` are vectors, the result is a single number.

Author(s)

Raymaekers J., Rousseeuw P.J.

References

Raymaekers J., Rousseeuw P.J., Hubert M. (2021). Class maps for visualizing classification results. *Technometrics*, 64(2), 151–165. doi:10.1080/00401706.2021.1927849

See Also

`makeFV`

Examples

```
library(e1071)
set.seed(1); X <- matrix(rnorm(200 * 2), ncol = 2)
X[1:100, ] <- X[1:100, ] + 2
X[101:150, ] <- X[101:150, ] - 2
y <- as.factor(c(rep("blue", 150), rep("red", 50))) # two classes
# We now fit an SVM with radial basis kernel to the data:
set.seed(1) # to make the result of svm() reproducible.
svmfit <- svm(y~.,  data = data.frame(X = X, y = y),  scale = FALSE,
             kernel = "radial", cost = 10, gamma = 1, probability = TRUE)
Kxx <- makeKernel(X, svfit = svmfit)
# The result is a square kernel matrix:
dim(Kxx) # 200 200
Kxx[1:5, 1:5]

# For more examples, we refer to the vignette:
## Not run:
vignette("Support_vector_machine_examples")

## End(Not run)
```

---

pamc1d *Constrained k-medoids clustering for univariate data*

Description

Performs constrained partitioning around medoids (PAM) clustering on univariate data. This is a modification of the constrained k-means algorithm by Bradley, Bennett and Demiriz (2000), adapted to use medians instead of means as cluster centers, and to enforce a minimum cluster size constraint on unique values rather than all cases. This prevents tied observations from being split across separate clusters.

Usage

```
pamc1d(y, k, minsize = 4, countwhat = "unique",
       stand = TRUE, maxit = 100, verbose = TRUE)
```

Arguments

y	a numeric vector of univariate data to be clustered. Missing values (NA) are removed before clustering.
k	integer specifying the desired number of clusters.
minsize	integer giving the minimum number of members required in each cluster. When countwhat = "unique" (the default), this refers to unique values; when countwhat = "any" it refers to all cases including duplicates. Defaults to 4.
countwhat	character string specifying whether the minimum cluster size constraint applies to "unique" values (the default) or to "any" cases including duplicates.

stand	logical. If TRUE (the default), the data are standardized by subtracting the mean and dividing by the standard deviation before clustering. The returned cluster centers are on the standardized scale.
maxit	integer giving the maximum number of iterations in the reassignment loop. Defaults to 100.
verbose	logical. If TRUE (the default), intermediate results including the clustering vector, cluster table, and objective function value are printed at each iteration.

Details

The algorithm starts from an unconstrained PAM solution obtained via pam. If any cluster fails to meet the minsize constraint, a constrained reassignment loop is entered. At each iteration, a transportation linear program is solved (via lp.transport) to reassign cases to clusters in a way that minimizes the total L1 distance to the cluster medians while satisfying the size constraints. Cluster medians are then recomputed, and the procedure repeats until no assignment changes or maxit iterations are reached.

When countwhat = "unique", only the unique values are passed to the linear program, and duplicate cases follow the cluster assignment of the unique value they match. This ensures that tied observations are never separated into different clusters.

The objective function is the mean absolute deviation of all cases from their assigned cluster median.

Value

A list with the following components:

iter	integer, the number of iterations performed.
converged	logical, TRUE if the algorithm converged (no assignment changes in the final iteration), FALSE if maxit was reached before convergence.
clustering	integer vector of length n (after removing NAs) giving the cluster assignment of each observation. Observations are sorted by value.
obj	numeric, the value of the objective function (mean absolute deviation from cluster medians) at the final iteration.
centers	numeric vector of length k with the median of each cluster on the (possibly standardized) scale.
clustable	a table giving the number of members in each cluster.

Author(s)

Rousseeuw P.J.

References

Bradley P.S., Bennett K.P., Demiriz A. (2000). Constrained k-means clustering. *Microsoft Research Technical Report*, MSR-TR-2000-65.

See Also

pam, lp.transport

Examples

```
set.seed(42)
y <- c(rnorm(30, mean = 0), rnorm(30, mean = 5), rnorm(30, mean = 10))

# Basic usage with k = 3 clusters and minimum cluster size of 4
result <- pamc1d(y, k = 3, minsize = 4, verbose = FALSE)

# Inspect the results
result$converged
result$clustable
result$centers

# Apply the constraint to all cases (not just unique values)
result2 <- pamc1d(y, k = 3, minsize = 4, countwhat = "any",
                  verbose = FALSE)

# Example with tied values
y_tied <- c(rep(1, 10), rep(3, 10), rep(5, 10),
            rep(7, 10), rep(9, 10))
result3 <- pamc1d(y_tied, k = 2, minsize = 2,
                  countwhat = "unique", verbose = TRUE)
result3$clustable
```

`predscor` *Draws a predictions correlation plot, which visualizes the correlations between the prediction terms in a regression fit.*

Description

Computes the correlations between the prediction terms in a regression fit, and displays them graphically in a way that takes the standard deviations of the prediction terms into account. The input variables of the regression can be numerical, categorical, logical and character, and the regression model can be linear or generalized linear. The regression formula in `lm` or `glm` may contain transformations and interactions.

Usage

```
predscor(fit, maxnpreds = 8, sort.by.stdev = TRUE, adj.order = FALSE,
cell.length = "stdev", plot.abs.cor = FALSE, palette = NULL,
diagonalcolor = "black")
```

Arguments

`fit`	an output object of `lm` or `glm`.
`maxnpreds`	the maximal number of prediction terms to plot. When there are more prediction terms than this, those with smallest standard deviations are combined.
`sort.by.stdev`	if `TRUE`, sorts the prediction terms by decreasing standard deviation.

adj.order	if TRUE, this modifies the order of the prediction terms in an attempt to bring highly correlated prediction terms close to each other in the predscor display. If sort.by.stdev is TRUE, this happens after that sorting.
cell.length	if "stdev", the sides of the square cells on the diagonal of the correlation matrix are proportional to the standard deviation of their prediction term. If "sqrt" they are proportional to the square root of the standard deviation. If "equal" all sides are the same.
plot.abs.cor	if FALSE, the default, positive and negative correlations are shown in different colors, typically red and blue. If TRUE the absolute values of the correlations are shown.
palette	a vector with colors to display correlations ranging from -1 to 1. If NULL, the default palette shows positive correlations in red, negative correlations in blue, and uses white for correlation zero.
diagonalcolor	color of the cells on the diagonal of the correlation matrix. The default is "black".

Value

A list containing

cormat	the correlation matrix of the prediction terms.
predterms	matrix of cases by prediction terms.
predsummary	data frame with the standard deviation of each prediction term and the total linear prediction.

Author(s)

Rousseeuw, P.J.

References

Rousseeuw, P.J. (2025). Explainable Linear and Generalized Linear Models by the Predictions Plot https://arxiv.org/abs/2412.16980v2 (open access).

See Also

predsplot

Examples

```
data(data_titanic)
attach(data_titanic)
Pclass = factor(Pclass, unique(Pclass))
Sex = factor(Sex, labels = c("F","M"))
fit <- glm(y ~ Sex + Age + SibSp + Parch + Pclass, family=binomial)
predscor(fit)

# For more examples, we refer to the vignette:
## Not run:
```

```
vignette("predsplot_examples")

## End(Not run)
```

---

predsplot *Make a predictions plot*

Description

Plots the prediction terms of a regression, together with the total prediction. The input variables of the regression can be numerical, categorical, logical and character, and are visualized by histograms, densities, or bar plots. The regression model can be linear or generalized linear. The regression formula in lm() or glm() may contain transformations and interactions. The plot shows all cases in the data, but can also be made for a single case, either in-sample or out-of-sample, to explain its prediction.

Usage

```
predsplot(fit, maxnpreds = 8, sort.by.stdev = TRUE, displaytype
                    = "histogram", totalpred.type = "response",
                    trunc.totalpred = TRUE, casetoshow = NULL, staircase =
                    FALSE, verbose = TRUE, nfact = 8, main = NULL,
                    cex.main = 1, xlab = NULL, ylab = NULL, cex.labs = 1,
                    cex.varnames = 1, cex.values = 1, cex.levelnames = 1,
                    maxchar.level = 5, plotcentercept = FALSE, vlinewidth
                    = 0.25, hlinewidth = 0.25, drawborder = TRUE,
                    borderwidth = 1, densitycolors = NULL, predupcolor =
                    "red", preddowncolor = "blue", predpointsize = 3,
                    draw.segments = TRUE, predsegmentwidth = 0.8, profile
                    = FALSE, bw = "nrd0", adjust = 1)
```

Arguments

`fit`	an output object of `lm` or `glm`.
`maxnpreds`	the maximal number of prediction terms to plot. When there are more prediction terms than this, those with smallest standard deviations are combined.
`sort.by.stdev`	if `TRUE`, sorts the prediction terms by decreasing standard deviation.
`displaytype`	when `"histogram"`, the distributions of the numerical prediction terms and the total prediction are displayed as histograms. When `"density"`, the default density estimate of R is plotted instead.
`totalpred.type`	when fit is a `glm` object, option `"response"` plots labels of the total prediction that are obtained by the inverse link function. Option `"linear"` plots labels of the total linear prediction. When fit is an `lm` object, argument `totalpred.type` is ignored.

`trunc.totalpred`
: if `TRUE`, the default, the range of the total prediction is truncated so ymin and ymax are determined by the individual prediction terms. `FALSE` may make the prediction terms look small in the plot.

`casetoshow`
: if not `NULL`, the particular case to be displayed. This can be a case number or row name of a case in the dataset, or a list or vector with input values of a new case.

`staircase`
: if `TRUE` and casetoshow is not `NULL`, the prediction for the case is shown in staircase style.

`verbose`
: if `TRUE`, some intermediate results are shown on the console.

`nfact`
: if a numeric input variable has at most `nfact` unique values, it will be displayed as if it were a factor. This may be useful since R considers a binary variable to be numeric.

`main`
: main title of the plot.

`cex.main`
: its size.

`xlab`
: horizontal axis label.

`ylab`
: vertical axis label.

`cex.labs`
: size of the axis labels.

`cex.varnames`
: size of the variable names. It may help to shorten the names in the data frame before running the regression.

`cex.values`
: size of the values of the numeric variables.

`cex.levelnames`
: size of the level names of non-numeric variables.

`maxchar.level`
: only the first maxchar.level characters of the non-numeric levels are displayed.

`plotcentercept`
: logical, whether to list the centercept in the plot.

`vlinewidth`
: width of the vertical lines.

`hlinewidth`
: width of the horizontal line.

`drawborder`
: if `TRUE`, draws a box outside the entire plot.

`borderwidth`
: width of the border. Defaults to 1.

`densitycolors`
: a vector with 4 colors. The first is for numeric input variables pointing up, the second for numeric input variables pointing down, the third for prediction terms without orientation and the total prediction, and the fourth for factors. If `NULL`, the default colors are used.

`predupcolor`
: the color of a positive prediction for casetoshow.

`preddowncolor`
: the color of a negative prediction for casetoshow.

`predpointsize`
: the size of the points displaying the predictions for casetoshow.

`draw.segments`
: if `TRUE`, also plots a line segment from the center of each prediction term to the prediction term for casetoshow, in the same color as the prediction.

`predsegmentwidth`
: the width of that line segment.

`profile`
: when casetoshow is not `NULL` and staircase is `FALSE`, this plots the profile of the case with a feint grey line.

bw	the bandwidth of the density estimation, only used when displaytype = "density". This is the argument 'bw' of the function density.
adjust	multiplier of the bandwidth of the density estimation, only used when displaytype = "density". This is the argument 'adjust' of the function density.

Value

A list with items

p	the predictions plot, which is a ggplot2 object.
totpred	vector with the total linear prediction of all cases for which it can be computed.
centercept	the centercept, which is the total linear prediction when all prediction terms have their average value.
predterms	matrix of cases by prediction terms.
predsummary	data frame with the standard deviation of each prediction term and the total linear prediction.
casetotpred	a number, the total linear prediction for casetoshow if one is given.
casepredterms	a vector with the values of the prediction terms for casetoshow.
casesummary	data frame which shows all prediction terms for casetoshow together with the centercept, total linear prediction, and for a glm fit also the total prediction in response units.

Author(s)

Rousseeuw, P.J.

References

Rousseeuw, P.J. (2026). Explainable Linear and Generalized Linear Models by the Predictions Plot. The American Statistician, 80, 157-163, doi:10.1080/00031305.2025.2539235 (open access)

See Also

predscor

Examples

```
data(data_titanic)
attach(data_titanic)
survival = y
Pclass = factor(Pclass, unique(Pclass))
Sex = factor(Sex, labels = c("F","M"))
fit <- glm(survival ~ Sex + Age + SibSp + Parch + Pclass, family=binomial)
predsplot(fit, main = "Titanic data", displaytype = "density")

# For more examples, we refer to the vignette:
## Not run:
vignette("predsplot_examples")

## End(Not run)
```

qresplot *Draw a quasi residual plot of PAC versus a data feature*

Description

Draw a quasi residual plot to visualize classification results. The vertical axis of the quasi residual plot shows each case's probability of alternative class (PAC). The horizontal axis shows the feature given as the second argument in the function call.

Usage

```
qresplot(PAC, feat, xlab = NULL, xlim = NULL,
         main = NULL, identify = FALSE, gray = TRUE,
         opacity = 1, squareplot = FALSE, plotLoess = FALSE,
         plotErrorBars = FALSE, plotQuantiles = FALSE,
         grid = NULL, probs = c(0.5, 0.75),
         cols = NULL, fac = 1, cex = 1,
         cex.main = 1.2, cex.lab = 1,
         cex.axis = 1, pch = 19)
```

Arguments

PAC	vector with the PAC values of a classification, typically the $PAC in the return of a call to a function vcr.*.*
feat	the PAC will be plotted versus this data feature. Note that feat does not have to be one of the explanatory variables of the model. It can be another variable, a combination of variables (like a sum or a principal component score), the row number of the cases if they were recorded succesively, etc.
xlab	label for the horizontal axis, i.e. the name of variable feat.
xlim	limits for the horizontal axis. If NULL, the range of feat is used.
main	title for the plot.
identify	if TRUE, left-click on a point to get its number, then ESC to exit.
gray	logical, if TRUE (the default) the plot region where $PAC < 0.5$ gets a light gray background. Points in this region were classified into their given class, and the points above this region were misclassified.
opacity	determines opacity of plotted dots. Value between 0 and 1, where 0 is transparent and 1 is opaque.
squareplot	if TRUE, the horizontal and vertical axis will get the same length.
plotLoess	if TRUE, a standard loess curve is fitted and superimposed on the plot. May not work well if feat is discrete with few values. At most one of the options plotLoess, plotErrorbars, or plotQuantiles can be selected.
plotErrorBars	if TRUE, the average PAC and its standard error are computed on the intervals of a grid (see option grid). Then a red curve connecting the averages is plotted, as well as two blue curves corresponding to the average plus or minus

one standard error. At most one of the options plotLoess, plotErrorbars, or plotQuantiles can be selected.

plotQuantiles — if TRUE, one or more quantiles of the PAC are computed on the intervals of a grid (see option grid). The quantiles correspond the probabilities in option probs. Then the curves connecting the quantiles are plotted. At most one of the options plotLoess, plotErrorbars, or plotQuantiles can be selected.

grid — only used when plotErrorBars or plotQuantiles are selected. This is a vector with increasing feat values, forming the grid. If NULL, the grid consists of the minimum and the maximum of feat, with 9 equispaced points between them.

probs — only used when plotQuantiles is selected. This is a vector with probabilities determining the quantiles. If NULL, defaults to c(0.5, 0.75).

cols — only used when plotquantiles is selected. A vector with the colors of the quantile curves. If NULL the cols are taken as 2, 3, ...

fac — only used when plotLoess, plotErrorBars or plotQuantiles are selected. A real number to multiply the resulting curves. A value fac > 1 can be useful to better visualize the curves when they would be too close to zero. By default (fac = 1) this is not done.

cex — passed on to plot.

cex.main — same, for title.

cex.lab — same, for labels on horizontal and vertical axes.

cex.axis — same, for axes.

pch — plot character for the points, defaults to 19.

Value

coordinates — a matrix with 2 columns containing the coordinates of the plotted points. This makes it easier to add text next to interesting points. If identify = TRUE, the attribute ids of coordinates contains the row numbers of the identified points in the matrix coordinates.

Author(s)

Raymaekers J., Rousseeuw P.J.

References

Raymaekers J., Rousseeuw P.J.(2021). Silhouettes and quasi residual plots for neural nets and tree-based classifiers. *Journal of Computational and Graphical Statistics*, 31(4), 1332–1343. doi:10.1080/10618600.2022.2050249

Examples

```
library(rpart)
data("data_titanic")
traindata <- data_titanic[which(data_titanic$dataType == "train"), -13]
set.seed(123) # rpart is not deterministic
rpart.out <- rpart(y ~ Pclass + Sex + SibSp +
```

```
                  Parch + Fare + Embarked,
                data = traindata, method = 'class', model = TRUE)
mytype <- list(nominal = c("Name", "Sex", "Ticket", "Cabin", "Embarked"), ordratio = c("Pclass"))
x_train <- traindata[, -12]
y_train <- traindata[,  12]
vcrtrain <- vcr.rpart.train(x_train, y_train, rpart.out, mytype)
# Quasi residual plot versus age, for males only:
PAC <- vcrtrain$PAC[which(x_train$Sex == "male")]
feat <- x_train$Age[which(x_train$Sex == "male")]
qresplot(PAC, feat, xlab = "Age (years)", opacity = 0.5,
         main = "quasi residual plot for male passengers",
         plotLoess = TRUE)
text(x = 14, y = 0.60, "loess curve", col = "red", cex = 1)
```

`silplot` *Draw the silhouette plot of a classification*

Description

Draw the silhouette plot to visualize classification results, based on the output of one of the `vcr.*.*` functions in this package. The horizontal axis of the silhouette plot shows each case's `s(i)`.

Usage

```
silplot(vcrout, classLabels = NULL, classCols = NULL,
        showLegend = TRUE, showClassNumbers = FALSE,
        showCases = FALSE, drawLineAtAverage = FALSE,
        topdown = TRUE, main = NULL, summary = TRUE)
```

Arguments

`vcrout`	output of `vcr.*.train` or `vcr.*.newdata`. Required.
`classLabels`	the labels (levels) of the classes. If `NULL`, they are taken from `vcrout`.
`classCols`	a list of colors for the classes. There should be at least as many as there are levels. If `NULL` a default palette is used.
`showLegend`	if `TRUE`, a legend is shown to the right of the plot.
`showClassNumbers`	if `TRUE`, the legend will show the class numbers instead of the class labels.
`showCases`	if `TRUE`, the plot shows the numbers of the cases. They are only readable when the number of cases is relatively small.
`topdown`	if `TRUE` (the default), the silhouettes are plotted from top to bottom. Otherwise they are plotted from left to right.
`drawLineAtAverage`	if `TRUE`, drwas a line at the average value of the `s(i)`.
`main`	title for the plot. If `NULL`, a default title is used.
`summary`	if `TRUE`, puts a summary table on the screen with for each class its number, label, number of class members, and the average of its `s(i)`.

Value

A ggplot object containing the silhouette plot.

Author(s)

Raymaekers J., Rousseeuw P.J.

References

Raymaekers J., Rousseeuw P.J.(2021). Silhouettes and quasi residual plots for neural nets and tree-based classifiers. *Journal of Computational and Graphical Statistics*, 31(4), 1332–1343. doi:10.1080/10618600.2022.2050249

See Also

`vcr.da.train`, `vcr.da.newdata`,
`vcr.knn.train`, `vcr.knn.newdata`,
`vcr.svm.train`, `vcr.svm.newdata`,
`vcr.rpart.train`, `vcr.rpart.newdata`,
`vcr.forest.train`, `vcr.forest.newdata`,
`vcr.neural.train`, `vcr.neural.newdata`

Examples

```
vcrout <- vcr.da.train(iris[, 1:4], iris[, 5])
silplot(vcrout)
# For more examples, we refer to the vignettes:
## Not run:
vignette("Discriminant_analysis_examples")
vignette("K_nearest_neighbors_examples")
vignette("Support_vector_machine_examples")
vignette("Rpart_examples")
vignette("Forest_examples")
vignette("Neural_net_examples")

## End(Not run)
```

---

stackedplot *Make a vertically stacked mosaic plot of class predictions.*

Description

Make a vertically stacked mosaic plot of class predictions from the output of `vcr.*.train` or `vcr.*.newdata`. Optionally, the outliers for each class can be shown as a gray rectangle at the top.

Usage

```
stackedplot(vcrout, cutoff = 0.99, classCols = NULL,
classLabels = NULL, separSize=1, minSize=1.5,
showOutliers = TRUE, showLegend = FALSE, main = NULL,
htitle = NULL, vtitle = NULL)
```

Arguments

vcrout	output of vcr.*.train or vcr.*.newdata.
cutoff	cases with overall farness vcrout$ofarness > cutoff are flagged as outliers.
classCols	user-specified colors for the classes. If NULL a default palette is used.
classLabels	names of given labels. If NULL they are taken from vcrout.
separSize	how much white between rectangles.
minSize	rectangles describing less than minSize percent of the data, are shown as minSize percent.
showOutliers	if TRUE, shows a separate class in gray with the outliers, always at the top.
showLegend	if TRUE, a legend is shown to the right of the plot. Default FALSE, since the legend is not necessary as the colors are already visible in the bottom part of each stack.
main	title for the plot.
htitle	title for horizontal axis (given labels). If NULL, a default title is shown.
vtitle	title for vertical axis (predicted labels). If NULL, a default title is shown.

Value

A ggplot object.

Author(s)

Raymaekers J., Rousseeuw P.J.

References

Raymaekers J., Rousseeuw P.J., Hubert M. (2021). Class maps for visualizing classification results. *Technometrics*, 64(2), 151–165. doi:10.1080/00401706.2021.1927849

See Also

vcr.da.train, vcr.da.newdata,
vcr.knn.train, vcr.knn.newdata,
vcr.svm.train, vcr.svm.newdata,
vcr.rpart.train, vcr.rpart.newdata,
vcr.forest.train, vcr.forest.newdata,
vcr.neural.train, vcr.neural.newdata

Examples

```
data("data_floralbuds")
X <- data_floralbuds[, 1:6]; y <- data_floralbuds[, 7]
vcrout <- vcr.da.train(X, y)
cols <- c("saddlebrown", "orange", "olivedrab4", "royalblue3")
stackedplot(vcrout, classCols = cols, showLegend = TRUE)

# The legend is not really needed, since we can read the
# color of a class from the bottom of its vertical bar:
stackedplot(vcrout, classCols = cols, main = "Stacked plot of QDA on foral buds data")

# If we do not wish to show outliers:
stackedplot(vcrout, classCols = cols, showOutliers = FALSE)

# For more examples, we refer to the vignettes:
## Not run:
vignette("Discriminant_analysis_examples")
vignette("K_nearest_neighbors_examples")
vignette("Support_vector_machine_examples")
vignette("Rpart_examples")
vignette("Random_forest_examples")
vignette("Neural_net_examples")

## End(Not run)
```

`vcr.da.newdata` *Carry out discriminant analysis on new data, and prepare to visualize its results.*

Description

Predicts class labels for new data by discriminant analysis, using the output of `vcr.da.train` on the training data. For new data cases whose label in `yintnew` is non-missing, additional output is produced for constructing graphical displays such as the `classmap`.

Usage

```
vcr.da.newdata(Xnew, ynew=NULL, vcr.da.train.out)
```

Arguments

`Xnew`	data matrix of the new data, with the same number of columns as in the training data. Missing values are not allowed.
`ynew`	factor with class membership of each new case. Can be `NA` for some or all cases. If `NULL`, is assumed to be `NA` everywhere.
`vcr.da.train.out`	output of `vcr.da.train` on the training data.

Value

A list with components:

yintnew	number of the given class of each case. Can contain NA's.
ynew	given class label of each case. Can contain NA's.
levels	levels of the response, from vcr.da.train.out.
predint	predicted class number of each case. Always exists.
pred	predicted label of each case.
altint	number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is NA for cases whose ynew is missing.
altlab	label of the alternative class. Is NA for cases whose ynew is missing.
PAC	probability of the alternative class. Is NA for cases whose ynew is missing.
fig	distance of each case i to each class g. Always exists.
farness	farness of each case i from its given class. Is NA for cases whose ynew is missing.
ofarness	For each case i, its lowest fig[i,g] to any class g. Always exists.
classMS	list with center and covariance matrix of each class, from vcr.da.train.out.
lCurrent	log of mixture density of each case in its given class. Is NA for cases with missing ynew.
lPred	log of mixture density of each case in its predicted class. Always exists.
lAlt	log of mixture density of each case in its alternative class. Is NA for cases with missing ynew.

Author(s)

Raymaekers J., Rousseeuw P.J.

References

Raymaekers J., Rousseeuw P.J., Hubert M. (2021). Class maps for visualizing classification results. *Technometrics*, 64(2), 151–165. doi:10.1080/00401706.2021.1927849

See Also

vcr.da.train, classmap, silplot, stackedplot

Examples

```
vcr.train <- vcr.da.train(iris[, 1:4], iris[, 5])
inds <- c(51:150) # a subset, containing only 2 classes
iris2 <- iris[inds, ] # fake "new" data
iris2[c(1:10, 51:60), 5] <- NA
vcr.test <- vcr.da.newdata(iris2[, 1:4], iris2[, 5], vcr.train)
vcr.test$PAC[1:25] # between 0 and 1. Is NA where the response is.
```

```
plot(vcr.test$PAC, vcr.train$PAC[inds]); abline(0, 1) # match
plot(vcr.test$farness, vcr.train$farness[inds]); abline(0, 1) # match
confmat.vcr(vcr.train) # for comparison
confmat.vcr(vcr.test)
stackedplot(vcr.train) # for comparison
stackedplot(vcr.test)
classmap(vcr.train, "versicolor", classCols = 2:4) # for comparison
classmap(vcr.test, "versicolor", classCols = 2:4) # has fewer points

# For more examples, we refer to the vignette:
## Not run:
vignette("Discriminant_analysis_examples")

## End(Not run)
```

`vcr.da.train` *Carry out discriminant analysis on training data, and prepare to visualize its results.*

Description

Custom DA function which prepares for graphical displays such as the `classmap`. The disciminant analysis itself is carried out by the maximum a posteriori rule, which maximizes the density of the mixture.

Usage

```
vcr.da.train(X, y, rule = "QDA", estmethod = "meancov")
```

Arguments

`X`	a numerical matrix containing the predictors in its columns. Missing values are not allowed.
`y`	a factor with the given class labels.
`rule`	either "`QDA`" for quadratic discriminant analysis or "`LDA`" for linear discriminant analysis.
`estmethod`	function for location and covariance estimation. Should return a list with the center `$m` and the covariance matrix `$S`. The default is "`meancov`" (classical mean and covariance matrix), and the option "`DetMCD`" (based on `robustbase::covMcd`) is also provided.

Value

A list with components:

`yint`	number of the given class of each case. Can contain `NA`'s.
`y`	given class label of each case. Can contain `NA`'s.

levels	levels of y
predint	predicted class number of each case. For each case this is the class with the highest posterior probability. Always exists.
pred	predicted label of each case.
altint	number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is NA for cases whose y is missing.
altlab	label of the alternative class. Is NA for cases whose y is missing.
PAC	probability of the alternative class. Is NA for cases whose y is missing.
figparams	parameters for computing fig, can be used for new data.
fig	distance of each case i from each class g. Always exists.
farness	farness of each case from its given class. Is NA for cases whose y is missing.
ofarness	for each case i, its lowest fig[i,g] to any class g. Always exists.
classMS	list with center and covariance matrix of each class
lCurrent	log of mixture density of each case in its given class. Is NA for cases with missing y.
lPred	log of mixture density of each case in its predicted class. Always exists.
lAlt	log of mixture density of each case in its alternative class. Is NA for cases with missing y.

Author(s)

Raymaekers J., Rousseeuw P.J.

References

Raymaekers J., Rousseeuw P.J., Hubert M. (2021). Class maps for visualizing classification results. *Technometrics*, 64(2), 151–165. doi:10.1080/00401706.2021.1927849

See Also

vcr.da.newdata, classmap, silplot, stackedplot

Examples

```
data("data_floralbuds")
X <- data_floralbuds[, 1:6]; y <- data_floralbuds[, 7]
vcrout <- vcr.da.train(X, y, rule = "QDA")
# For linear discriminant analysis, put rule = "LDA".
confmat.vcr(vcrout) # There are a few outliers
cols <- c("saddlebrown", "orange", "olivedrab4", "royalblue3")
stackedplot(vcrout, classCols = cols)
classmap(vcrout, "bud", classCols = cols)

# For more examples, we refer to the vignette:
## Not run:
```

```
vignette("Discriminant_analysis_examples")

## End(Not run)
```

---

vcr.forest.newdata *Prepare for visualization of a random forest classification on new data.*

Description

Produces output for the purpose of constructing graphical displays such as the classmap on new data. Requires the output of vcr.forest.train as an argument.

Usage

```
vcr.forest.newdata(Xnew, ynew = NULL, vcr.forest.train.out,
                   LOO = FALSE)
```

Arguments

Xnew	data matrix of the new data, with the same number of columns d as in the training data. Missing values are not allowed.
ynew	factor with class membership of each new case. Can be NA for some or all cases. If NULL, is assumed to be NA everywhere.
vcr.forest.train.out	output of vcr.forest.train on the training data.
LOO	leave one out. Only used when testing this function on a subset of the training data. Default is LOO=FALSE.

Value

A list with components:

yintnew	number of the given class of each case. Can contain NA's.
ynew	given class label of each case. Can contain NA's.
levels	levels of the response, from vcr.forest.train.out.
predint	predicted class number of each case. Always exists.
pred	predicted label of each case.
altint	number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is NA for cases whose ynew is missing.
altlab	alternative label if yintnew was given, else NA.
PAC	probability of the alternative class. Is NA for cases whose ynew is missing.
fig	distance of each case i from each class g. Always exists.
farness	farness of each case from its given class. Is NA for cases whose ynew is missing.
ofarness	for each case i, its lowest fig[i,g] to any class g. Always exists.

Author(s)

Raymaekers J., Rousseeuw P.J.

References

Raymaekers J., Rousseeuw P.J.(2021). Silhouettes and quasi residual plots for neural nets and tree-based classifiers. *Journal of Computational and Graphical Statistics*, 31(4), 1332–1343. doi:10.1080/10618600.2022.2050249

See Also

vcr.forest.train, classmap, silplot, stackedplot

Examples

```
library(randomForest)
data("data_instagram")
traindata <- data_instagram[which(data_instagram$dataType == "train"), -13]
set.seed(71) # randomForest is not deterministic
rfout <- randomForest(y ~ ., data = traindata, keep.forest = TRUE)
mytype <- list(symm = c(1, 5, 7, 8)) # These 4 columns are
# (symmetric) binary variables. The variables that are not
# listed are interval-scaled by default.
x_train <- traindata[, -12]
y_train <- traindata[, 12]
vcrtrain <- vcr.forest.train(X = x_train, y = y_train,
                             trainfit = rfout, type = mytype)
testdata <- data_instagram[which(data_instagram$dataType == "test"), -13]
Xnew <- testdata[, -12]
ynew <- testdata[, 12]
vcrtest <- vcr.forest.newdata(Xnew, ynew, vcrtrain)
confmat.vcr(vcrtest)
stackedplot(vcrtest, classCol = c(4, 2))
silplot(vcrtest, classCols = c(4, 2))
classmap(vcrtest, "genuine", classCols = c(4, 2))
classmap(vcrtest, "fake", classCols = c(4, 2))

# For more examples, we refer to the vignette:
## Not run:
vignette("Random_forest_examples")

## End(Not run)
```

`vcr.forest.train` *Prepare for visualization of a random forest classification on training data*

Description

Produces output for the purpose of constructing graphical displays such as the `classmap` and `silplot`. The user first needs to train a random forest on the data by `randomForest::randomForest`. This then serves as an argument to `vcr.forest.train`.

Usage

```
vcr.forest.train(X, y, trainfit, type = list(),
                 k = 5, stand = TRUE)
```

Arguments

`X`	A rectangular matrix or data frame, where the columns (variables) may be of mixed type.
`y`	factor with the given class labels. It is crucial that `X` and `y` are exactly the same as in the call to `randomForest::randomForest`. `y` is allowed to contain `NA`'s.
`trainfit`	the output of a `randomForest::randomForest` training run.
`k`	the number of nearest neighbors used in the farness computation.
`type`	list for specifying some (or all) of the types of the variables (columns) in `X`, used for computing the dissimilarity matrix, as in `cluster::daisy`. The list may contain the following components: `"ordratio"` (ratio scaled variables to be treated as ordinal variables), `"logratio"` (ratio scaled variables that must be logarithmically transformed), `"asymm"` (asymmetric binary) and `"symm"` (symmetric binary variables). Each component's value is a vector, containing the names or the numbers of the corresponding columns of `X`. Variables not mentioned in the `type` list are interpreted as usual (see argument `X`).
`stand`	whether or not to standardize numerical (interval scaled) variables by their range as in the original `cluster::daisy` code for the farness computation. Defaults to `TRUE`.

Value

A list with components:

`X`	The data used to train the forest.
`yint`	number of the given class of each case. Can contain `NA`'s.
`y`	given class label of each case. Can contain `NA`'s.
`levels`	levels of `y`
`predint`	predicted class number of each case. For each case this is the class with the highest posterior probability. Always exists.
`pred`	predicted label of each case.
`altint`	number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is `NA` for cases whose `y` is missing.

altlab	label of the alternative class. Is NA for cases whose y is missing.
PAC	probability of the alternative class. Is NA for cases whose y is missing.
figparams	parameters for computing fig, can be used for new data.
fig	distance of each case i from each class g. Always exists.
farness	farness of each case from its given class. Is NA for cases whose y is missing.
ofarness	for each case i, its lowest fig[i,g] to any class g. Always exists.
trainfit	The trained random forest which was given as an input to this function.

Author(s)

Raymaekers J., Rousseeuw P.J.

References

Raymaekers J., Rousseeuw P.J.(2021). Silhouettes and quasi residual plots for neural nets and tree-based classifiers. *Journal of Computational and Graphical Statistics*, 31(4), 1332–1343. doi:10.1080/10618600.2022.2050249

See Also

vcr.forest.newdata, classmap, silplot, stackedplot

Examples

```
library(randomForest)
data("data_instagram")
traindata <- data_instagram[which(data_instagram$dataType == "train"), -13]
set.seed(71) # randomForest is not deterministic
rfout <- randomForest(y~., data = traindata, keep.forest = TRUE)
mytype <- list(symm = c(1, 5, 7, 8)) # These 4 columns are
# (symmetric) binary variables. The variables that are not
# listed are interval-scaled by default.
x_train <- traindata[, -12]
y_train <- traindata[, 12]
# Prepare for visualization:
vcrtrain <- vcr.forest.train(X = x_train, y = y_train,
                             trainfit = rfout, type = mytype)
confmat.vcr(vcrtrain)
stackedplot(vcrtrain, classCols = c(4, 2))
silplot(vcrtrain, classCols = c(4, 2))
classmap(vcrtrain, "genuine", classCols = c(4, 2))
classmap(vcrtrain, "fake", classCols = c(4, 2))

# For more examples, we refer to the vignette:
## Not run:
vignette("Random_forest_examples")

## End(Not run)
```

`vcr.knn.newdata` *Carry out a k-nearest neighbor classification on new data, and prepare to visualize its results.*

Description

Predicts class labels for new data by k nearest neighbors, using the output of `vcr.knn.train` on the training data. For cases in the new data whose given label `ynew` is not `NA`, additional output is produced for constructing graphical displays such as the `classmap`.

Usage

```
vcr.knn.newdata(Xnew, ynew = NULL, vcr.knn.train.out, LOO = FALSE)
```

Arguments

`Xnew`	If the training data was a matrix of coordinates, `Xnew` must be such a matrix with the same number of columns. If the training data was a set of dissimilarities, `Xnew` must be a rectangular matrix of dissimilarities, with each row containing the dissmilarities of a new case to all training cases. Missing values are not allowed.
`ynew`	factor with class membership of each new case. Can be `NA` for some or all cases. If `NULL`, is assumed to be `NA` everywhere.
`vcr.knn.train.out`	output of `vcr.knn.train` on the training data.
`LOO`	leave one out. Only used when testing this function on a subset of the training data. Default is `LOO=FALSE`.

Value

A list with components:

`yintnew`	number of the given class of each case. Can contain `NA`'s.
`ynew`	given class label of each case. Can contain `NA`'s.
`levels`	levels of the response, from `vcr.knn.train.out`.
`predint`	predicted class number of each case. Always exists.
`pred`	predicted label of each case.
`altint`	number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is `NA` for cases whose `ynew` is missing.
`altlab`	label of the alternative class. Is `NA` for cases whose `ynew` is missing.
`PAC`	probability of the alternative class. Is `NA` for cases whose `ynew` is missing.
`fig`	distance of each case i from each class g. Always exists.

farness	farness of each case from its given class. Is NA for cases whose ynew is missing.
ofarness	for each case i, its lowest fig[i,g] to any class g. Always exists.
k	the requested number of nearest neighbors, from vcr.knn.train.out.
ktrues	for each case this contains the actual number of elements in its neighborhood. This can be higher than k due to ties.
counts	a matrix with 3 columns, each row representing a case. For the neighborhood of each case it says how many members it has from the given class, the predicted class, and the alternative class. The first and third entry is NA for cases whose ynew is missing.

Author(s)

Raymaekers J., Rousseeuw P.J.

References

Raymaekers J., Rousseeuw P.J., Hubert M. (2021). Class maps for visualizing classification results. *Technometrics*, 64(2), 151–165. doi:10.1080/00401706.2021.1927849

See Also

vcr.knn.train, classmap, silplot, stackedplot

Examples

```
data("data_floralbuds")
X <- data_floralbuds[, 1:6]; y <- data_floralbuds[, 7]
set.seed(12345); trainset <- sample(1:550, 275)
vcr.train <- vcr.knn.train(X[trainset, ], y[trainset], k = 5)
vcr.test <- vcr.knn.newdata(X[-trainset, ], y[-trainset], vcr.train)
confmat.vcr(vcr.train) # for comparison
confmat.vcr(vcr.test)
cols <- c("saddlebrown", "orange", "olivedrab4", "royalblue3")
stackedplot(vcr.train, classCols = cols) # for comparison
stackedplot(vcr.test, classCols = cols)
classmap(vcr.train, "bud", classCols = cols) # for comparison
classmap(vcr.test, "bud", classCols = cols)

# For more examples, we refer to the vignette:
## Not run:
vignette("K_nearest_neighbors_examples")

## End(Not run)
```

vcr.knn.train *Carry out a k-nearest neighbor classification on training data, and prepare to visualize its results.*

Description

Carries out a k-nearest neighbor classification on the training data. Various additional output is produced for the purpose of constructing graphical displays such as the classmap.

Usage

```
vcr.knn.train(X, y, k)
```

Arguments

X	This can be a rectangular matrix or data frame of (already standardized) measurements, or a dist object obtained from stats::dist or cluster::daisy. Missing values are not allowed.
y	factor with the given (observed) class labels. There need to be non-missing y in order to be able to train the classifier.
k	the number of nearest neighbors used. It can be selected by running cross-validation using a different package.

Value

A list with components:

yint	number of the given class of each case. Can contain NA's.
y	given class label of each case. Can contain NA's.
levels	levels of y
predint	predicted class number of each case. Always exists.
pred	predicted label of each case.
altint	number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is NA for cases whose y is missing.
altlab	label of the alternative class. Is NA for cases whose y is missing.
PAC	probability of the alternative class. Is NA for cases whose y is missing.
figparams	parameters used to compute fig.
fig	distance of each case i from each class g. Always exists.
farness	farness of each case from its given class. Is NA for cases whose y is missing.
ofarness	for each case i, its lowest fig[i,g] to any class g. Always exists.

k	the requested number of nearest neighbors, from the arguments. Will also be used for classifying new data.
ktrues	for each case this contains the actual number of elements in its neighborhood. This can be higher than k due to ties.
counts	a matrix with 3 columns, each row representing a case. For the neighborhood of each case it says how many members it has from the given class, the predicted class, and the alternative class. The first and third entry is NA for cases whose y is missing.
X	If the argument X was a data frame or matrix of coordinates, as.matrix(X) is returned here. This is useful for classifying new data.

Author(s)

Raymaekers J., Rousseeuw P.J.

References

Raymaekers J., Rousseeuw P.J., Hubert M. (2021). Class maps for visualizing classification results. *Technometrics*, 64(2), 151–165. doi:10.1080/00401706.2021.1927849

See Also

vcr.knn.newdata, classmap, silplot, stackedplot

Examples

```
vcrout <- vcr.knn.train(iris[, 1:4], iris[, 5], k = 5)
confmat.vcr(vcrout)
stackedplot(vcrout)
classmap(vcrout, "versicolor", classCols = 2:4)
# The cases misclassified as virginica are shown in blue.

# For more examples, we refer to the vignette:
## Not run:
vignette("K_nearest_neighbors_examples")

## End(Not run)
```

---

vcr.neural.newdata *Prepare for visualization of a neural network classification on new data.*

Description

Prepares graphical display of new data fitted by a neural net that was modeled on the training data, using the output of vcr.neural.train on the training data.

Usage

```
vcr.neural.newdata(Xnew, ynew = NULL, probs,
                   vcr.neural.train.out)
```

Arguments

Xnew	data matrix of the new data, with the same number of columns as in the training data. Missing values in Xnew are not allowed.
ynew	factor with class membership of each new case. Can be NA for some or all cases. If NULL, is assumed to be NA everywhere.
probs	posterior probabilities obtained by running the neural net on the new data.
vcr.neural.train.out	output of vcr.neural.train on the training data.

Value

A list with components:

yintnew	number of the given class of each case. Can contain NA's.
ynew	given class label of each case. Can contain NA's.
levels	levels of the response, from vcr.svm.train.out.
predint	predicted class number of each case. Always exists.
pred	predicted label of each case.
altint	number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is NA for cases whose ynew is missing.
altlab	alternative label if yintnew was given, else NA.
PAC	probability of the alternative class. Is NA for cases whose ynew is missing.
fig	distance of each case i from each class g. Always exists.
farness	farness of each case from its given class. Is NA for cases whose ynew is missing.
ofarness	for each case i, its lowest fig[i,g] to any class g. Always exists.

Author(s)

Raymaekers J., Rousseeuw P.J.

References

Raymaekers J., Rousseeuw P.J.(2021). Silhouettes and quasi residual plots for neural nets and tree-based classifiers. *Journal of Computational and Graphical Statistics*, 31(4), 1332–1343. doi:10.1080/10618600.2022.2050249

See Also

vcr.neural.train, classmap, silplot, stackedplot

Examples

```
# For examples, we refer to the vignette:
## Not run:
vignette("Neural_net_examples")

## End(Not run)
```

`vcr.neural.train` *Prepare for visualization of a neural network classification on training data.*

Description

Produces output for the purpose of constructing graphical displays such as the `classmap`. The user first needs train a neural network. The representation of the data in a given layer (e.g. the final layer before applying the softmax function) then serves as the argument `X` to `vcr.neural.train`.

Usage

```
vcr.neural.train(X, y, probs, estmethod = meancov)
```

Arguments

`X`	the coordinates of the `n` objects of the training data, in the layer chosen by the user. Missing values are not allowed.
`y`	factor with the given class labels of the objects. Make sure that the levels are in the same order as used in the neural net, i.e. the columns of its binary "once-hot-encoded" response vectors.
`probs`	posterior probabilities obtained by the neural net, e.g. in keras. For each case (row of `X`), the classes have probabilities that add up to 1. Each row of the matrix probs contains these probabilities. The columns of probs must be in the same order as the levels of `y`.
`estmethod`	function for location and covariance estimation. Should return a list with `$m` and `$S`. Can be `meancov` (classical mean and covariance matrix) or `DetMCD`. If one or more classes have a singular covariance matrix, the function automatically switches to the PCA-based farness used in `vcr.svm.train`.

Value

A list with components:

`X`	the coordinates of the `n` objects of the training data, in the layer chosen by the user.
`yint`	number of the given class of each case. Can contain `NA`'s.
`y`	given class label of each case. Can contain `NA`'s.

levels	levels of y
predint	predicted class number of each case. For each case this is the class with the highest posterior probability. Always exists.
pred	predicted label of each case.
altint	number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is NA for cases whose y is missing.
altlab	label of the alternative class. Is NA for cases whose y is missing.
ncolX	number of columns in X. Keep??
PAC	probability of the alternative class. Is NA for cases whose y is missing.
computeMD	Whether or not the farness is computed using the Mahalanobis distance.
classMS	list with center and covariance matrix of each class
PCAfits	if not NULL, PCA fits to each class, estimated from the training data but also useful for new data.
figparams	parameters for computing fig, can be used for new data.
fig	distance of each case i from each class g. Always exists.
farness	farness of each case from its given class. Is NA for cases whose y is missing.
ofarness	for each case i, its lowest fig[i,g] to any class g. Always exists.

Author(s)

Raymaekers J., Rousseeuw P.J.

References

Raymaekers J., Rousseeuw P.J.(2021). Silhouettes and quasi residual plots for neural nets and tree-based classifiers. *Journal of Computational and Graphical Statistics*, 31(4), 1332–1343. doi:10.1080/10618600.2022.2050249

See Also

vcr.neural.newdata, classmap, silplot, stackedplot

Examples

```
# For examples, we refer to the vignette:
## Not run:
vignette("Neural_net_examples")

## End(Not run)
```

`vcr.rpart.newdata` *Prepare for visualization of an rpart classification on new data.*

Description

Produces output for the purpose of constructing graphical displays such as the `classmap` on new data. Requires the output of `vcr.rpart.train` as an argument.

Usage

```
vcr.rpart.newdata(Xnew, ynew = NULL, vcr.rpart.train.out,
                  LOO = FALSE)
```

Arguments

`Xnew`	data matrix of the new data, with the same number of columns d as in the training data. Missing values are not allowed.
`ynew`	factor with class membership of each new case. Can be `NA` for some or all cases. If `NULL`, is assumed to be `NA` everywhere.
`vcr.rpart.train.out`	output of `vcr.rpart.train` on the training data.
`LOO`	leave one out. Only used when testing this function on a subset of the training data. Default is `LOO=FALSE`.

Value

A list with components:

`yintnew`	number of the given class of each case. Can contain `NA`'s.
`ynew`	given class label of each case. Can contain `NA`'s.
`levels`	levels of the response, from `vcr.rpart.train.out`.
`predint`	predicted class number of each case. Always exists.
`pred`	predicted label of each case.
`altint`	number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is `NA` for cases whose `ynew` is missing.
`altlab`	alternative label if yintnew was given, else `NA`.
`PAC`	probability of the alternative class. Is `NA` for cases whose `ynew` is missing.
`fig`	distance of each case i from each class g. Always exists.
`farness`	farness of each case from its given class. Is `NA` for cases whose `ynew` is missing.
`ofarness`	for each case i, its lowest `fig[i,g]` to any class g. Always exists.

Author(s)

Raymaekers J., Rousseeuw P.J.

References

Raymaekers J., Rousseeuw P.J.(2021). Silhouettes and quasi residual plots for neural nets and tree-based classifiers. *Journal of Computational and Graphical Statistics*, 31(4), 1332–1343. doi:10.1080/10618600.2022.2050249

See Also

vcr.rpart.train, classmap, silplot, stackedplot

Examples

```
library(rpart)
data("data_titanic")
traindata <- data_titanic[which(data_titanic$dataType == "train"), -13]
str(traindata); table(traindata$y)
set.seed(123) # rpart is not deterministic
rpart.out <- rpart(y ~ Pclass + Sex + SibSp +
                   Parch + Fare + Embarked,
                 data = traindata, method = 'class', model = TRUE)
y_train <- traindata[, 12]
x_train <- traindata[, -12]
mytype <- list(nominal = c("Name", "Sex", "Ticket", "Cabin", "Embarked"), ordratio = c("Pclass"))
# These are 5 nominal columns, and one ordinal.
# The variables not listed are by default interval-scaled.
vcrtrain <- vcr.rpart.train(x_train, y_train, rpart.out, mytype)
testdata <- data_titanic[which(data_titanic$dataType == "test"), -13]
dim(testdata)
x_test <- testdata[, -12]
y_test <- testdata[, 12]
vcrtest <- vcr.rpart.newdata(x_test, y_test, vcrtrain)
confmat.vcr(vcrtest)
silplot(vcrtest, classCols = c(2, 4))
classmap(vcrtest, "casualty", classCols = c(2, 4))
classmap(vcrtest, "survived", classCols = c(2, 4))

# For more examples, we refer to the vignette:
## Not run:
vignette("Rpart_examples")

## End(Not run)
```

---

vcr.rpart.train *Prepare for visualization of an rpart classification on training data.*

Description

Produces output for the purpose of constructing graphical displays such as the classmap. The user first needs to train a classification tree on the data by rpart::rpart. This then serves as an argument to vcr.rpart.train.

Usage

```
vcr.rpart.train(X, y, trainfit, type = list(),
                k = 5, stand = TRUE)
```

Arguments

X	A rectangular matrix or data frame, where the columns (variables) may be of mixed type and may contain NA's.
y	factor with the given class labels. It is crucial that X and y are exactly the same as in the call to rpart::rpart. y is allowed to contain NA's.
k	the number of nearest neighbors used in the farness computation.
trainfit	the output of an rpart::rpart training cycle.
type	list for specifying some (or all) of the types of the variables (columns) in X, used for computing the dissimilarity matrix, as in cluster::daisy. The list may contain the following components: "ordratio" (ratio scaled variables to be treated as ordinal variables), "logratio" (ratio scaled variables that must be logarithmically transformed), "asymm" (asymmetric binary) and "symm" (symmetric binary variables). Each component's value is a vector, containing the names or the numbers of the corresponding columns of X. Variables not mentioned in the type list are interpreted as usual (see argument X).
stand	whether or not to standardize numerical (interval scaled) variables by their range as in the original cluster::daisy code for the farness computation. Defaults to TRUE.

Value

A list with components:

X	The input data X. Keep??
yint	number of the given class of each case. Can contain NA's.
y	given class label of each case. Can contain NA's.
levels	levels of y
predint	predicted class number of each case. For each case this is the class with the highest posterior probability. Always exists.
pred	predicted label of each case.
altint	number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is NA for cases whose y is missing.

altlab	label of the alternative class. Is NA for cases whose y is missing.
PAC	probability of the alternative class. Is NA for cases whose y is missing.
figparams	parameters for computing fig, can be used for new data.
fig	distance of each case i from each class g. Always exists.
farness	farness of each case from its given class. Is NA for cases whose y is missing.
ofarness	for each case i, its lowest fig[i,g] to any class g. Always exists.
trainfit	the trainfit used to build the VCR object.

Author(s)

Raymaekers J., Rousseeuw P.J.

References

Raymaekers J., Rousseeuw P.J.(2021). Silhouettes and quasi residual plots for neural nets and tree-based classifiers. *Journal of Computational and Graphical Statistics*, 31(4), 1332–1343. doi:10.1080/10618600.2022.2050249

See Also

vcr.rpart.newdata, classmap, silplot, stackedplot

Examples

```
library(rpart)
data("data_titanic")
traindata <- data_titanic[which(data_titanic$dataType == "train"), -13]
str(traindata); table(traindata$y)
set.seed(123) # rpart is not deterministic
rpart.out <- rpart(y ~ Pclass + Sex + SibSp +
                     Parch + Fare + Embarked,
                   data = traindata, method = 'class', model = TRUE)
y_train <- traindata[, 12]
x_train <- traindata[, -12]
mytype <- list(nominal = c("Name", "Sex", "Ticket", "Cabin", "Embarked"), ordratio = c("Pclass"))
# These are 5 nominal columns, and one ordinal.
# The variables not listed are by default interval-scaled.
vcrtrain <- vcr.rpart.train(x_train, y_train, rpart.out, mytype)
confmat.vcr(vcrtrain)
silplot(vcrtrain, classCols = c(2, 4))
classmap(vcrtrain, "casualty", classCols = c(2, 4))
classmap(vcrtrain, "survived", classCols = c(2, 4))

# For more examples, we refer to the vignette:
## Not run:
vignette("Rpart_examples")

## End(Not run)
```

`vcr.svm.newdata` *Prepare for visualization of a support vector machine classification on new data.*

Description

Carries out a support vector machine classification of new data using the output of `vcr.svm.train` on the training data, and computes the quantities needed for its visualization.

Usage

```
vcr.svm.newdata(Xnew, ynew = NULL, vcr.svm.train.out)
```

Arguments

`Xnew`	data matrix of the new data, with the same number of columns as in the training data. Missing values in `Xnew` are not allowed.
`ynew`	factor with class membership of each new case. Can be `NA` for some or all cases. If `NULL`, is assumed to be `NA` everywhere.
`vcr.svm.train.out`	output of `vcr.svm.train` on the training data.

Value

A list with components:

`yintnew`	number of the given class of each case. Can contain `NA`'s.
`ynew`	given class label of each case. Can contain `NA`'s.
`levels`	levels of the response, from `vcr.svm.train.out`.
`predint`	predicted class number of each case. Always exists.
`pred`	predicted label of each case.
`altint`	number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is `NA` for cases whose `ynew` is missing.
`altlab`	alternative label if yintnew was given, else `NA`.
`PAC`	probability of the alternative class. Is `NA` for cases whose `ynew` is missing.
`fig`	distance of each case i from each class g. Always exists.
`farness`	farness of each case from its given class. Is `NA` for cases whose `ynew` is missing.
`ofarness`	for each case i, its lowest `fig[i,g]` to any class g. Always exists.

Author(s)

Raymaekers J., Rousseeuw P.J.

References

Raymaekers J., Rousseeuw P.J., Hubert M. (2021). Class maps for visualizing classification results. *Technometrics*, 64(2), 151–165. doi:10.1080/00401706.2021.1927849

See Also

vcr.svm.train, classmap, silplot, stackedplot, e1071::svm

Examples

```
library(e1071)
set.seed(1); X <- matrix(rnorm(200 * 2), ncol = 2)
X[1:100, ] <- X[1:100, ] + 2
X[101:150, ] <- X[101:150, ] - 2
y <- as.factor(c(rep("blue", 150), rep("red", 50)))
# We now fit an SVM with radial basis kernel to the data:
set.seed(1) # to make the result of svm() reproducible.
svmfit <- svm(y~., data = data.frame(X = X, y = y),
scale = FALSE, kernel = "radial", cost = 10,
gamma = 1, probability = TRUE)
vcr.train <- vcr.svm.train(X, y, svfit = svmfit)
# As "new" data we take a subset of the training data:
inds <- c(1:25, 101:125, 151:175)
vcr.test <- vcr.svm.newdata(X[inds, ], y[inds], vcr.train)
plot(vcr.test$PAC, vcr.train$PAC[inds]); abline(0, 1) # match
plot(vcr.test$farness, vcr.train$farness[inds]); abline(0, 1)
confmat.vcr(vcr.test)
cols <- c("deepskyblue3", "red")
stackedplot(vcr.test, classCols = cols)
classmap(vcr.train, "blue", classCols = cols) # for comparison
classmap(vcr.test, "blue", classCols = cols)
classmap(vcr.train, "red", classCols = cols) # for comparison
classmap(vcr.test, "red", classCols = cols)


# For more examples, we refer to the vignette:
## Not run:
vignette("Support_vector_machine_examples")

## End(Not run)
```

`vcr.svm.train` *Prepare for visualization of a support vector machine classification on training data.*

Description

Produces output for the purpose of constructing graphical displays such as the classmap. The user first needs to run a support vector machine classification on the data by e1071::svm, with

the option probability = TRUE. This classification can be with two or more classes. The output of e1071::svm is then an argument to vcr.svm.train. As e1071::svm does not output the data itself, it needs to be given as well, in the arguments X and y.

Usage

```
vcr.svm.train(X, y, svfit, ortho = FALSE)
```

Arguments

X	matrix of data coordinates, as used in e1071::svm. Missing values are not allowed.
y	factor with the given (observed) class labels. It is crucial that X and y are exactly the same as in the call to e1071::svm.
svfit	an object returned by e1071::svm, called with exactly the same X and y as above.
ortho	If TRUE, will compute farness in the orthogonal complement of the vector beta given by e1071::svm. Is only possible for 2 classes, else there would be several beta vectors.

Value

A list with components:

yint	number of the given class of each case. Can contain NA's.
y	given class label of each case. Can contain NA's.
levels	levels of the response y.
predint	predicted class number of each case. Always exists.
pred	predicted label of each case.
altint	number of the alternative class. Among the classes different from the given class, it is the one with the highest posterior probability. Is NA for cases whose y is missing.
altlab	label of the alternative class. Is NA for cases whose y is missing.
PAC	probability of the alternative class. Is NA for cases whose y is missing.
figparams	parameters used in fig, can be used for new data.
fig	distance of each case i from each class g. Always exists.
farness	farness of each case from its given class. Is NA for cases whose y is missing.
ofarness	for each case i, its lowest fig[i,g] to any class g. Always exists.
svfit	as it was input, will be useful for new data.
X	the matrix of data coordinates from the arguments. This is useful for classifying new data.

Author(s)

Raymaekers J., Rousseeuw P.J.

References

Raymaekers J., Rousseeuw P.J., Hubert M. (2021). Class maps for visualizing classification results. *Technometrics*, 64(2), 151–165. doi:10.1080/00401706.2021.1927849

See Also

vcr.knn.newdata, classmap, silplot, stackedplot, e1071::svm

Examples

```
library(e1071)
set.seed(1); X <- matrix(rnorm(200 * 2), ncol = 2)
X[1:100, ] <- X[1:100, ] + 2
X[101:150, ] <- X[101:150, ] - 2
y <- as.factor(c(rep("blue", 150), rep("red", 50)))
cols <- c("deepskyblue3", "red")
plot(X, col = cols[as.numeric(y)], pch = 19)
# We now fit an SVM with radial basis kernel to the data:
set.seed(1) # to make the result of svm() reproducible.
svmfit <- svm(y~., data = data.frame(X = X, y = y),
scale = FALSE, kernel = "radial", cost = 10,
gamma = 1, probability = TRUE)
plot(svmfit$decision.values, col = cols[as.numeric(y)]); abline(h = 0)
# so the decision values separate the classes reasonably well.
plot(svmfit, data = data.frame(X = X, y = y), X.2~X.1, col = cols)
# The boundary is far from linear (but in feature space it is).
vcr.train <- vcr.svm.train(X, y, svfit = svmfit)
confmat.vcr(vcr.train)
stackedplot(vcr.train, classCols = cols)
classmap(vcr.train, "blue", classCols = cols)
classmap(vcr.train, "red", classCols = cols)

# For more examples, we refer to the vignette:
## Not run:
vignette("Support_vector_machine_examples")

## End(Not run)
```

Index