# Package 'condvis'

March 27, 2026

**Type** Package

**Title** Conditional Visualization for Statistical Models

**Version** 0.5-2

**Date** 2026-02-28

**Depends** R (>= 2.1.0)

**Imports** graphics, grDevices, stats, utils, MASS

**Suggests** RColorBrewer, shiny, scagnostics, cluster, hdrcde, gplots,
TSP, DendSer, testthat, e1071

**Description** Exploring fitted models by interactively taking 2-D and 3-D
sections in data space.

**License** GPL (>= 2)

**LazyData** false

**BugReports** https://github.com/markajoc/condvis/issues

**URL** https://markajoc.github.io/condvis/

**RoxygenNote** 5.0.1.9000

**NeedsCompilation** no

**Author** Mark O'Connell [aut, cre],
Catherine Hurley [aut],
Katarina Domijan [aut],
Achim Zeileis [ctb] (spineplot, see copied.R),
R Core Team [ctb] (barplot, see copied.R)

**Maintainer** Mark O'Connell <mark_ajoc@yahoo.ie>

**Repository** CRAN

**Date/Publication** 2026-03-27 08:20:02 UTC

# Contents

condvis-package            *Conditional Visualization for Statistical Models*

### Description

Exploring statistical models by interactively taking 2-D and 3-D sections in data space. The main
functions for end users are ceplot (see example below) and condtour. Requires XQuartz on
Mac OS, and X11 on Linux. A website for the package is available at markajoc.github.io/condvis.
Source code is available to browse at GitHub. Bug reports and feature requests are very welcome
at GitHub.

### Details

| | |
|---|---|
| Package: | condvis |
| Type: | Package |
| Version: | 0.5-1 |
| Date: | 2018-09-13 |
| License: | GPL (>= 2) |

### Author(s)

Mark O'Connell <mark_ajoc@yahoo.ie>, Catherine Hurley <catherine.hurley@mu.ie>, Katarina
Domijan <katarina.domijan@mu.ie>.

## References

O'Connell M, Hurley CB and Domijan K (2017). "Conditional Visualization for Statistical Models: An Introduction to the **condvis** Package in R."*Journal of Statistical Software*, **81**(5), pp. 1-20. <URL:http://dx.doi.org/10.18637/jss.v081.i05>.

## Examples

```
## Not run:
mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$am <- as.factor(mtcars$am)

library(mgcv)
model1 <- list(
  quadratic = lm(mpg ~ cyl + am + qsec + wt + I(wt^2), data = mtcars),
  additive = gam(mpg ~ cyl + am + qsec + s(wt), data = mtcars))

ceplot(data = mtcars, model = model1, sectionvars = "wt")

## End(Not run)
```

---

arrangeC                  *Make a list of variable pairings for condition selecting plots produced by plotxc*

---

## Description

This function arranges a number of variables in pairs, ordered by their bivariate relationships. The goal is to discover which variable pairings are most helpful in avoiding extrapolations when exploring the data space. Variable pairs with strong bivariate dependencies (not necessarily linear) are chosen first. The bivariate dependency is measured using [savingby2d](#). Each variable appears in the output only once.

## Usage

```
arrangeC(data, method = "default")
```

## Arguments

| | |
|---|---|
| data | A dataframe |
| method | The character name for the method to use for measuring bivariate dependency, passed to [savingby2d](#). |

## Details

If data is so big as to make arrangeC very slow, a random sample of rows is used instead. The bivariate dependency measures are rough, and the ordering algorithm is a simple greedy one, so it is not worth allowing it too much time. This function exists mainly to provide a helpful default ordering/pairing for [ceplot](#).

**Value**

A list containing character vectors giving variable pairings.

**References**

O'Connell M, Hurley CB and Domijan K (2017). "Conditional Visualization for Statistical Models: An Introduction to the **condvis** Package in R."*Journal of Statistical Software*, **81**(5), pp. 1-20. <URL:http://dx.doi.org/10.18637/jss.v081.i05>.

**See Also**

[savingby2d](savingby2d)

**Examples**

```
data(powerplant)

pairings <- arrangeC(powerplant)

dev.new(height = 2, width = 2 * length(pairings))
par(mfrow = c(1, length(pairings)))

for (i in seq_along(pairings)){
 plotxc(powerplant[, pairings[[i]]], powerplant[1, pairings[[i]]],
   select.col = NA)
}
```

---

ceplot                          *Interactive conditional expectation plot*

---

**Description**

Creates an interactive conditional expectation plot, which consists of two main parts. One part is a single plot depicting a section through a fitted model surface, or conditional expectation. The other part shows small data summaries which give the current condition, which can be altered by clicking with the mouse.

**Usage**

```
ceplot(data, model, response = NULL, sectionvars = NULL,
  conditionvars = NULL, threshold = NULL, lambda = NULL,
  distance = c("euclidean", "maxnorm"), type = c("default", "separate",
  "shiny"), view3d = FALSE, Corder = "default", selectortype = "minimal",
  conf = FALSE, probs = FALSE, col = "black", pch = NULL,
  residuals = FALSE, xsplotpar = NULL, modelpar = NULL,
  xcplotpar = NULL)
```

## Arguments

| | |
|---|---|
| `data` | A dataframe containing the data to plot |
| `model` | A model object, or list of model objects |
| `response` | Character name of response in `data` |
| `sectionvars` | Character name of variable(s) from `data` on which to take a section, can be of length 1 or 2. |
| `conditionvars` | Character names of conditioning variables from `data`. These are the predictors which we can set to single values in order to produce a section. Can be a list of vectors of length 1 or 2. Can be a character vector, which is then paired up using [arrangeC](). If NULL, an attempt will be made to extract all variable names which are not `response` or `sectionvars` from `model`, and these will be arranged using [arrangeC](). |
| `threshold` | This is a threshold distance. Points further than `threshold` away from the current section will not be visible. Passed to [similarityweight](). |
| `lambda` | A constant to multiply by number of factor mismatches in constructing a general dissimilarity measure. If left NULL, behaves as though `lambda` is set greater than `threshold`, and so only observations whose factor levels match the current section are visible. Passed to [similarityweight](). |
| `distance` | A character vector describing the type of distance measure to use, either `"euclidean"` (default) or `"maxnorm"`. |
| `type` | This specifies the type of interactive plot. `"default"` places everything on one device. `"separate"` places condition selectors on one device and the section on another. (These two options require XQuartz on OS X). `"shiny"` produces a Shiny application. |
| `view3d` | Logical; if TRUE plots a three-dimensional regression surface if possible. |
| `Corder` | Character name for method of ordering conditioning variables. See [arrangeC](). |
| `selectortype` | Type of condition selector plots to use. Must be `"minimal"` if `type` is `"default"`. If `type` is `"separate"`, can be `"pcp"` (see [plotxc.pcp]()) or `"full"` (see [plotxc.full]()). |
| `conf` | Logical; if TRUE plots confidence bounds (or equivalent) for models which provide this. |
| `probs` | Logical; if TRUE, shows predicted class probabilities instead of just predicted classes. Only available if S specifies two numeric predictors and the model's predict method provides this. |
| `col` | Colour for observed data. |
| `pch` | Plot symbols for observed data. |
| `residuals` | Logical; if TRUE, plots a residual versus predictor plot instead of the usual scale of raw response. |
| `xsplotpar` | Plotting parameters for section visualisation as a list, passed to [plotxs](). Can specify `xlim`, `ylim`. |
| `modelpar` | Plotting parameters for models as a list, passed to [plotxs](). Not used. |
| `xcplotpar` | Plotting parameters for condition selector plots as a list, passed to [plotxc](). Can specify `col` for highlighting current section, `cex`, and `trim` (see [plotxc]()). |

**References**

O'Connell M, Hurley CB and Domijan K (2017). "Conditional Visualization for Statistical Models: An Introduction to the **condvis** Package in R."*Journal of Statistical Software*, **81**(5), pp. 1-20. <URL:http://dx.doi.org/10.18637/jss.v081.i05>.

**See Also**

condtour, similarityweight

**Examples**

```
## Not run:
## Example 1: Multivariate regression, xs one continuous predictor

mtcars$cyl <- as.factor(mtcars$cyl)

library(mgcv)
model1 <- list(
    quadratic = lm(mpg ~ cyl + hp + wt + I(wt^2), data = mtcars),
    additive = mgcv::gam(mpg ~ cyl + hp + s(wt), data = mtcars))

conditionvars1 <- list(c("cyl", "hp"))

ceplot(data = mtcars, model = model1, response = "mpg", sectionvars = "wt",
  conditionvars = conditionvars1, threshold = 0.3, conf = T)

## Example 2: Binary classification, xs one categorical predictor

mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$am <- as.factor(mtcars$am)

library(e1071)
model2 <- list(
  svm = svm(am ~ mpg + wt + cyl, data = mtcars, family = "binomial"),
  glm = glm(am ~ mpg + wt + cyl, data = mtcars, family = "binomial"))

ceplot(data = mtcars, model = model2, sectionvars = "wt", threshold = 1,
  type = "shiny")

## Example 3: Multivariate regression, xs both continuous

mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$gear <- as.factor(mtcars$gear)

library(e1071)
model3 <- list(svm(mpg ~ wt + qsec + cyl + hp + gear,
  data = mtcars, family = "binomial"))

conditionvars3 <- list(c("cyl","gear"), "hp")

ceplot(data = mtcars, model = model3, sectionvars = c("wt", "qsec"),
  threshold = 1, conditionvars = conditionvars3)
```

```
ceplot(data = mtcars, model = model3, sectionvars = c("wt", "qsec"),
     threshold = 1, type = "separate", view3d = T)

## Example 4: Multi-class classification, xs both categorical

mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$vs <- as.factor(mtcars$vs)
mtcars$am <- as.factor(mtcars$am)
mtcars$gear <- as.factor(mtcars$gear)
mtcars$carb <- as.factor(mtcars$carb)

library(e1071)
model4 <- list(svm(carb ~ ., data = mtcars, family = "binomial"))

ceplot(data = mtcars, model = model4, sectionvars = c("cyl", "gear"),
  threshold = 3)

## Example 5: Multi-class classification, xs both continuous

data(wine)
wine$Class <- as.factor(wine$Class)
library(e1071)

model5 <- list(svm(Class ~ ., data = wine, probability = TRUE))

ceplot(data = wine, model = model5, sectionvars = c("Hue", "Flavanoids"),
  threshold = 3, probs = TRUE)

ceplot(data = wine, model = model5, sectionvars = c("Hue", "Flavanoids"),
  threshold = 3, type = "separate")

ceplot(data = wine, model = model5, sectionvars = c("Hue", "Flavanoids"),
  threshold = 3, type = "separate", selectortype = "pcp")

## Example 6: Multi-class classification, xs with one categorical predictor,
##            and one continuous predictor.

mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$carb <- as.factor(mtcars$carb)

library(e1071)
model6 <- list(svm(cyl ~ carb + wt + hp, data = mtcars, family = "binomial"))

ceplot(data = mtcars, model = model6, threshold = 1, sectionvars = c("carb",
  "wt"), conditionvars = "hp")

## End(Not run)
```

| condtour | *Conditional tour; a tour through sections in data space* |
|---|---|

**Description**

Whereas [ceplot](#) allows the user to interactively choose sections to visualise, condtour allows the user to pre-select all sections to visualise, order them, and cycle through them one by one. ']' key advances the tour, and '[' key goes back. Can adjust threshold for the current section visualisation with ',' and '.' keys.

**Usage**

```
condtour(data, model, path, response = NULL, sectionvars = NULL,
  conditionvars = NULL, threshold = NULL, lambda = NULL,
  distance = c("euclidean", "maxnorm"), view3d = FALSE,
  Corder = "default", conf = FALSE, col = "black", pch = NULL,
  xsplotpar = NULL, modelpar = NULL, xcplotpar = NULL)
```

**Arguments**

| | |
|---|---|
| data | A dataframe. |
| model | A fitted model object, or a list of such objects. |
| path | A dataframe, describing the sections to take. Basically a dataframe with its colnames being conditionvars. |
| response | Character name of response variable in data. |
| sectionvars | Character name(s) of variables in data on which to take sections. |
| conditionvars | Character name(s) of variables in data on which to condition. |
| threshold | Threshold distance. Observed data which are a distance greater than threshold from the current section are not visible. Passed to [similarityweight](#). |
| lambda | A constant to multiply by number of factor mismatches in constructing a general dissimilarity measure. If left NULL, behaves as though lambda is set greater than threshold, and so only observations whose factor levels match the current section are visible. Passed to [similarityweight](#). |
| distance | The type of distance measure to use, either "euclidean" (default) or "maxnorm". |
| view3d | Logical; if TRUE, plots a three-dimensional regression surface when possible. |
| Corder | Character name for method of ordering conditioning variables. See [arrangeC](#). |
| conf | Logical; if TRUE, plots confidence bounds or equivalent when possible. |
| col | Colour for observed data points. |
| pch | Plot symbols for observed data points. |
| xsplotpar | Plotting parameters for section visualisation as a list, passed to [plotxs](#). Not used. |
| modelpar | Plotting parameters for models as a list, passed to [plotxs](#). Not used. |
| xcplotpar | Plotting parameters for condition selector plots as a list, passed to [plotxc](#). Can specify cex.axis, cex.lab , tck, col for highlighting current section, cex. |

## Value

Produces a set of interactive plots. One device displays the current section. A second device shows the the current section in the space of the conditioning predictors given by conditionvars. A third device shows some simple diagnostic plots; one to show approximately how much data are visible on each section, and another to show what proportion of data are *visited* by the tour.

## See Also

ceplot, similarityweight

## Examples

```
## Not run:

data(powerplant)
library(e1071)
model <- svm(PE ~ ., data = powerplant)
path <- makepath(powerplant[-5], 25)
condtour(data = powerplant, model = model, path = path$path,
  sectionvars = "AT")

data(wine)
wine$Class <- as.factor(wine$Class)
library(e1071)
model5 <- list(svm(Class ~ ., data = wine))
conditionvars1 <- setdiff(colnames(wine), c("Class", "Hue", "Flavanoids"))
path <- makepath(wine[, conditionvars1], 50)
condtour(data = wine, model = model5, path = path$path, sectionvars = c("Hue"
  , "Flavanoids"), threshold = 3)


## End(Not run)
```

---

cont2color                    *Assign colours to numeric vector*

---

## Description

This function assigns colours on a linear scale to a numeric vector. Default is to try to use RColorBrewer for colours, and cm.colors otherwise. Can provide custom range, breaks and colours.

## Usage

```
cont2color(x, xrange = NULL, breaks = NULL, colors = NULL)
```

## Arguments

| | |
|---|---|
| `x` | A numeric vector. |
| `xrange` | The range to use for the colour scale. |
| `breaks` | The number of breaks at which to change colour. |
| `colors` | The colours to use. Defaults to a diverging colour scheme; either `"PiYG"` from RColorBrewer if available, or [cm.colors](#) otherwise. |

## Details

Uses the RColorBrewer package if installed. Coerces `x` to numeric with a warning.

## Value

A character vector of colours.

## See Also

[factor2color](#)

## Examples

```
x <- runif(200)
plot(x, col = cont2color(x, c(0,1)))

plot(x, col = cont2color(x, c(0,0.5)))

plot(sort(x), col = cont2color(sort(x), c(0.25,0.75)), pch = 16)
abline(h = c(0.25, 0.75), lty = 3)
```

---

crab                                          *Brockmann's crab data*

---

## Description

Abstract from original paper: Horseshoe crabs arrive on the beach in pairs and spawn in the high intertidal during the springtime, new and full moon high tides. Unattached males also come to the beach, crowd around the nesting couples and compete with attached males for fertilizations. Satellite males form large groups around some couples while ignoring others, resulting in a non-random distribution that cannot be explained by local environmental conditions or habitat selection. In experimental manipulations, pairs that had satellites regained them after they had been removed whereas pairs with no satellites continued nesting alone, which means that satellites were not simply accumulating around the pairs that had been on the beach the longest. Manipulations also revealed that satellites were not just copying the behaviour of other males. Based on the evidence from observations and experiments, the most likely explanation for the nonrandom distribution of satellite males among nesting pairs is that unattached males are preferentially attracted to some females over others. Females with many satellites were larger and in better condition, but did not lay more eggs,

than females with few or no satellites.

satellites response variable; number of satellites around female crab
color color of crab
spine condition of spine
weight weight of crab
width width of carapace

### Format

173 observations on 5 variables.

### Source

https://onlinecourses.science.psu.edu/stat504/node/169

### References

Brockmann, H. (1996), "Satellite male groups in horseshoe crabs," *Ethology*, **102**-1, pp. 1-21.

### Examples

    data(crab)

---

dist1                           *Minkowski distance*

---

### Description

Calculate Minkowski distance between one point and a set of other points.

### Usage

    dist1(x, X, p = 2, inf = FALSE)

### Arguments

| | |
|---|---|
| x | A numeric vector describing point coordinates. |
| X | A numeric matrix describing coordinates for several points. |
| p | The power in Minkowski distance, defaults to 2 for Euclidean distance. |
| inf | Logical; switch for calculating maximum norm distance (sometimes known as Chebychev distance) which is the limit of Minkowski distance as $p$ tends to infinity. |

### Value

A numeric vector. These are distance^p, for speed of computation.

## See Also

[similarityweight](#)

## Examples

```
x <- runif(5000)
y <- runif(5000)

x1 <- 0.5
y1 <- 0.5

dev.new(width = 4, height = 5.3)
par(mfrow = c(2, 2))

for(p in c(0.5, 1, 2, 10)){
  d <- dist1(x = c(x1, y1), X = cbind(x, y), p = p) ^ (1/p)
  col <- rep("black", length(x))
  col[d < 0.3] <- "red"
  plot(x, y, pch = 16, col = col, asp = 1, main = paste("p = ", p, sep = ""))
}
```

---

factor2color                         *Assign colours to factor vector*

---

## Description

This function takes a factor vector and returns suitable colours representing the factor levels. Default is to try to use RColorBrewer for colours, and [rainbow](#) otherwise. Can provide custom colours.

## Usage

```
factor2color(x, colors = NULL)
```

## Arguments

x                  A factor vector.

colors             The colours to use. Defaults to a qualitative colour scheme; either "Set3" from
                   RColorBrewer if available, or [rainbow](#) otherwise.

## Details

Uses the RColorBrewer package if installed. Coerces x to factor with a warning.

## Value

A character vector of colours.

## See Also

[cont2color](#)

## Examples

```
plot(iris[, c("Petal.Length", "Petal.Width")], pch = 21,
  bg = factor2color(iris$Species))
legend("topleft", legend = levels(iris$Species),
  fill = factor2color(as.factor(levels(iris$Species))))
```

---

interpolate                    *Interpolate*

---

## Description

Interpolate a numeric or factor vector.

## Usage

```
interpolate(x, ...)

## S3 method for class 'numeric'
interpolate(x, ninterp = 4L, ...)

## S3 method for class 'integer'
interpolate(x, ninterp = 4L, ...)

## S3 method for class 'factor'
interpolate(x, ninterp = 4L, ...)

## S3 method for class 'character'
interpolate(x, ninterp = 4L, ...)
```

## Arguments

| | |
|---|---|
| x | A numeric or factor vector. |
| ... | Not used. |
| ninterp | The number of points to interpolate between observations. It should be an even number for sensible results on a factor/character vector. |

---

makepath                          *Make a default path for conditional tour*

---

### Description

Provides a default path (a set of sections), useful as input to a conditional tour ([condtour](#)). Clusters the data using k-means or partitioning around medoids (from the cluster package). The cluster centres/prototypes are then ordered to create a sensible way to visit each section as smoothly as possible. Ordering uses either the DendSer or TSP package. Linear interpolation is then used to create intermediate points between the path nodes.

### Usage

```
makepath(x, ncentroids, ninterp = 4)
```

### Arguments

| | |
|---|---|
| x | A dataframe |
| ncentroids | The number of centroids to use as path nodes. |
| ninterp | The number of points to linearly interpolate between path nodes. |

### Value

A list with two dataframes: centers giving the path nodes, and path giving the full interpolated path.

### See Also

[condtour](#)

### Examples

```
d <- data.frame(x = runif(500), y = runif(500))
plot(d)
mp1 <- makepath(d, 5)
points(mp1$centers, type = "b", col = "blue", pch = 16)
mp2 <- makepath(d, 40)
points(mp2$centers, type = "b", col = "red", pch = 16)
```

---

**plotxc** *Condition selector plot*

---

### Description

Data visualisations used to select sections for [ceplot](ceplot).

### Usage

```
plotxc(xc, xc.cond, name = NULL, trim = NULL, select.colour = NULL,
  select.lwd = NULL, cex.axis = NULL, cex.lab = NULL, tck = NULL,
  select.cex = 1, hist2d = NULL, fullbin = NULL, ...)
```

### Arguments

| | |
|---|---|
| xc | A numeric or factor vector, or a dataframe with two columns |
| xc.cond | Same type as xc, representing a single point in data space to highlight. |
| name | The variable name for xc |
| trim | Logical; if TRUE, long tails of continuous data are chopped off at the 5th and 95th percentiles. |
| select.colour | Colour to highlight xc.cond |
| select.lwd | Line weight to highlight xc.cond |
| cex.axis | Axis text scaling |
| cex.lab | Label text scaling |
| tck | Plot axis tick size |
| select.cex | Plot symbol size |
| hist2d | If TRUE, a scatterplot is visualised as a 2-D histogram. Default behaviour is to use a 2-D histogram if there are over 2,000 observations. |
| fullbin | A cap on the counts in a bin for the 2-D histogram, helpful with skewed data. Larger values give more detail about data density. Defaults to 25. |
| ... | Passed to condvis:::spineplot2. |

### Value

Produces a plot, and returns a list containing the relevant information to update the plot at a later stage.

### References

O'Connell M, Hurley CB and Domijan K (2017). "Conditional Visualization for Statistical Models: An Introduction to the **condvis** Package in R."*Journal of Statistical Software*, **81**(5), pp. 1-20. <URL:http://dx.doi.org/10.18637/jss.v081.i05>.

**See Also**

ceplot, plotxs.

plotxs, ceplot, condtour

**Examples**

```
## Histogram, highlighting the first case.

data(mtcars)
obj <- plotxc(mtcars[, ”mpg”], mtcars[1, ”mpg”])
obj$usr

## Barplot, highlighting 'cyl' = 6.

plotxc(as.factor(mtcars[, ”cyl”]), 6, select.colour = ”blue”)

## Scatterplot, highlighting case 25.

plotxc(mtcars[, c(”qsec”, ”wt”)], mtcars[25, c(”qsec”, ”wt”)],
  select.colour = ”blue”, select.lwd = 1, lty = 3)

## Boxplot, where 'xc' contains one factor, and one numeric.

mtcars$carb <- as.factor(mtcars$carb)
plotxc(mtcars[, c(”carb”, ”wt”)], mtcars[25, c(”carb”, ”wt”)],
  select.colour = ”red”, select.lwd = 3)

## Spineplot, where 'xc' contains two factors.

mtcars$gear <- as.factor(mtcars$gear)
mtcars$cyl <- as.factor(mtcars$cyl)
plotxc(mtcars[, c(”cyl”, ”gear”)], mtcars[25, c(”cyl”, ”gear”)],
  select.colour = ”red”)

## Effect of 'trim'.

x <- c(-200, runif(400), 200)
plotxc(x, 0.5, trim = FALSE, select.colour = ”red”)
plotxc(x, 0.5, trim = TRUE, select.colour = ”red”)
```

---

plotxc.pcp                    *Condition selector plot*

---

**Description**

Multivariate data visualisations used to select sections for ceplot. Basically visualises a dataset
and highlights a single point.

## Usage

```
plotxc.pcp(Xc, Xc.cond, select.colour = NULL, select.lwd = 3,
  cex.axis = NULL, cex.lab = NULL, tck = NULL, select.cex = 1, ...)

plotxc.full(Xc, Xc.cond, select.colour = NULL, select.lwd = 3,
  cex.axis = NULL, cex.lab = NULL, tck = NULL, select.cex = 0.6, ...)
```

## Arguments

| | |
|---|---|
| Xc | A dataframe. |
| Xc.cond | A dataframe with one row and same names as Xc. |
| select.colour | Colour to highlight Xc.cond |
| select.lwd | Line weight to highlight Xc.cond |
| cex.axis | Axis text scaling |
| cex.lab | Label text scaling |
| tck | Plot axis tick size |
| select.cex | Plot symbol size |
| ... | not used. |

## Value

Produces a plot, and returns a list containing the relevant information to update the plot at a later stage.

## See Also

[ceplot](), [plotxs](), [plotxc]()

---

| plotxs | *Visualise a section in data space* |
|---|---|

---

## Description

Visualise a section in data space, showing fitted models where they intersect the section, and nearby observations. The weights for observations can be calculated with [similarityweight](). This function is mainly for use in [ceplot]() and [condtour]().

## Usage

```
plotxs(xs, y, xc.cond, model, model.colour = NULL, model.lwd = NULL,
  model.lty = NULL, model.name = NULL, yhat = NULL, mar = NULL,
  col = "black", weights = NULL, view3d = FALSE, theta3d = 45,
  phi3d = 20, xs.grid = NULL, prednew = NULL, conf = FALSE,
  probs = FALSE, pch = 1, residuals = FALSE, main = NULL, xlim = NULL,
  ylim = NULL)
```

## Arguments

| | |
|---|---|
| xs | A dataframe with one or two columns. |
| y | A dataframe with one column. |
| xc.cond | A dataframe with a single row, with all columns required for passing to [predict](predict) methods of models in model. |
| model | A fitted model object, or a list of such objects. |
| model.colour | Colours for fitted models. If model is a list, this should be of same length as model. |
| model.lwd | Line weight for fitted models. If model is a list, this should be of same length as model. |
| model.lty | Line style for fitted models. If model is a list, this should be of same length as model. |
| model.name | Character labels for models, for legend. |
| yhat | Fitted values for the observations in y. Calculated if needed and not provided. Only used if showing residuals, or xs has two columns. |
| mar | Margins for plot. |
| col | Colours for observed data. Should be of length nrow(xs). |
| weights | Similarity weights for observed data. Should be of length nrow(xs). Usually calculated with [similarityweight](similarityweight). |
| view3d | Logical; if TRUE plots a three-dimensional regression surface if possible. |
| theta3d, phi3d | Angles defining the viewing direction. theta3d gives the azimuthal direction and phi3d the colatitude. See [persp](persp). |
| xs.grid | The grid of values defining the part of the section to visualise. Calculated if not provided. |
| prednew | The y values where the models in model intersect the section. Useful when providing theta3d, phi3d, or weights, where the predict methods have been called elsewhere. |
| conf | Logical; if TRUE plots confidence bounds (or equivalent) for models which provide this. |
| probs | Logical; if TRUE, shows predicted class probabilities instead of just predicted classes. Only available if xs contains two numeric predictors and the model's predict method provides this. |
| pch | Plot symbols for observed data |
| residuals | Logical; if TRUE, plots a residual versus predictor plot instead of the usual scale of raw response. |
| main | Character title for plot, default is "Conditional expectation". |
| xlim | Graphical parameter passed to plotting functions. |
| ylim | Graphical parameter passed to plotting functions. |

## Value

A list containing relevant information for updating the plot.

## References

O'Connell M, Hurley CB and Domijan K (2017). "Conditional Visualization for Statistical Models: An Introduction to the **condvis** Package in R." *Journal of Statistical Software*, **81**(5), pp. 1-20. <URL:http://dx.doi.org/10.18637/jss.v081.i05>.

## See Also

plotxc, ceplot, condtour

## Examples

```
data(mtcars)
model <- lm(mpg ~ ., data = mtcars)
plotxs(xs = mtcars[, "wt", drop = FALSE], y = mtcars[, "mpg", drop = FALSE],
  xc.cond = mtcars[1, ], model = list(model))
```

---

| powerplant | *Tuefekci's powerplant data* |
|---|---|

---

## Description

The dataset contains 9568 data points collected from a Combined Cycle Power Plant over 6 years (2006-2011), when the power plant was set to work with full load. Features consist of hourly average ambient variables Temperature (T), Ambient Pressure (AP), Relative Humidity (RH) and Exhaust Vacuum (V) to predict the net hourly electrical energy output (EP) of the plant.

A combined cycle power plant (CCPP) is composed of gas turbines (GT), steam turbines (ST) and heat recovery steam generators. In a CCPP, the electricity is generated by gas and steam turbines, which are combined in one cycle, and is transferred from one turbine to another. While the Vacuum is collected from and has effect on the Steam Turbine, the other three of the ambient variables affect the GT performance.

## Format

9568 observations on 5 continuous variables.

## Source

UCI repository. https://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant

## References

Tuefekci, P. (2014), Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods, *International Journal of Electrical Power & Energy Systems*, **60**, pp. 126-140, ISSN 0142-0615.

## Examples

```
data(powerplant)
head(powerplant)
```

---

savingby2d                *Assess advantage of 2-D view over 1-D view for identifying extrapola-*
                          *tion*

---

## Description

A simple algorithm to evaluate the advantage of by taking a bivariate marginal view of two variables, when trying to avoid extrapolations, rather than two univariate marginal views.

## Usage

```
savingby2d(x, y = NULL, method = "default")
```

## Arguments

| | |
|---|---|
| x | A numeric or factor vector. Can also be a dataframe containing x and y, if y is NULL. |
| y | A numeric or factor vector. |
| method | Character; criterion used to quantify bivariate relationships. Can be "default", a scagnostic measure, or "DECR" to use a density estimate confidence region. |

## Details

If given two continuous variables, the variables are both scaled to mean 0 and variance 1. Then the returned value is the ratio of the area of the convex hull of the data to the area obtained from the product of the ranges of the two areas, i.e. the area of the bounding rectangle.

If given two categorical variables, all combinations are tabulated. The returned value is the number of non-zero table entries divided by the total number of table entries.

If given one categorical and one continuous variable, the returned value is the weighted mean of the range of the continuous variable within each category divided by the overall range of the continuous variable, where the weights are given by the number of observations in each level of the categorical variable.

Requires package scagnostics if a scagnostics measure is specified in method. Requires package hdrcde if "DECR" (density estimate confidence region) is specified in method. These only apply to cases where x and y are both numeric.

## Value

A number between 0 and 1. Values near 1 imply no benefit to using a 2-D view, whereas values near 0 imply that a 2-D view reveals structure hidden in the 1-D views.

### References

O'Connell M, Hurley CB and Domijan K (2017). "Conditional Visualization for Statistical Models: An Introduction to the **condvis** Package in R."*Journal of Statistical Software*, **81**(5), pp. 1-20. <URL:http://dx.doi.org/10.18637/jss.v081.i05>.

### See Also

[similarityweight](#)

### Examples

```
x <- runif(1000)
y <- runif(1000)
plot(x, y)
savingby2d(x, y)
## value near 1, no real benefit from bivariate view

x1 <- runif(1000)
y1 <- x1 + rnorm(sd = 0.3, n = 1000)
plot(x1, y1)
savingby2d(x1, y1)
## smaller value indicates that the bivariate view reveals some structure
```

---

| similarityweight | *Calculate the similarity weight for a set of observations* |
| --- | --- |

---

### Description

Calculate the similarity weight for a set of observations, based on their distance from some arbitary points in data space. Observations which are very similar to the point under consideration are given weight 1, while observations which are dissimilar to the point are given weight zero.

### Usage

```
similarityweight(x, data, threshold = NULL, distance = NULL,
  lambda = NULL)
```

### Arguments

| | |
| --- | --- |
| x | A dataframe describing arbitrary points in the space of the data (i.e., with same `colnames` as `data`). |
| data | A dataframe representing observed data. |
| threshold | Threshold distance outside which observations will be assigned similarity weight zero. This is numeric and should be > 0. Defaults to 1. |
| distance | The type of distance measure to be used, currently just two types of Minkowski distance: `"euclidean"` (default), and `"maxnorm"`. |

lambda                A constant to multiply by the number of categorical mismatches, before adding to the Minkowski distance, to give a general dissimilarity measure. If left NULL, behaves as though lambda is set larger than threshold, meaning that one factor mismatch guarantees zero weight.

## Details

Similarity weight is assigned to observations based on their distance from a given point. The distance is calculated as Minkowski distance between the numeric elements for the observations whose categorical elements match, with the option to use a more general dissimilarity measure comprising Minkowski distance and a mismatch count.

## Value

A numeric vector or matrix, with values from 0 to 1. The similarity weights for the observations in data arranged in rows for each row in x.

## References

O'Connell M, Hurley CB and Domijan K (2017). "Conditional Visualization for Statistical Models: An Introduction to the **condvis** Package in R."*Journal of Statistical Software*, **81**(5), pp. 1-20. <URL:http://dx.doi.org/10.18637/jss.v081.i05>.

## See Also

[dist1](#)

## Examples

```
## Say we want to find observations similar to the first observation.
## The first observation is identical to itself, so it gets weight 1. The
## second observation is similar, so it gets some weight. The rest are more
## different, and so get zero weight.

data(mtcars)
similarityweight(x = mtcars[1, ], data = mtcars)

## By increasing the threshold, we can find observations which are more
## approximately similar to the first row. Note that the second observation
## now has weight 1, so we lose some ability to discern how similar
## observations are by increasing the threshold.

similarityweight(x = mtcars[1, ], data = mtcars, threshold = 5)

## Can provide a number of points to 'x'. Here we see that the Mazda RX4 Wag
## is more similar to the Merc 280 than the Mazda RX4 is.

similarityweight(mtcars[1:2, ], mtcars, threshold = 3)
```

---

wine *Italian wine data*

---

## Description

`Class` 3 different cultivars
`Alcohol` Alcohol
`Malic` Malic acid
`Ash` Ash
`Alcalinity` Alcalinity of ash
`Magnesium` Magnesium
`Phenols` Total phenols
`Flavanoids` Flavanoids
`Nonflavanoid` Nonflavanoid phenols
`Proanthocyanins` Proanthocyanins
`Intensity` Color intensity
`Hue` Hue
`OD280` OD280/OD315 of diluted wines
`Proline` Proline

## Format

178 observations on 14 variables.

## Source

UCI repository. https://archive.ics.uci.edu/ml/datasets/Wine

## References

S. Aeberhard, D. Coomans and O. de Vel (1992), Comparison of Classifiers in High Dimensional Settings, *Technical Report* **92**-02, Dept. of Computer Science and Dept. of Mathematics and Statistics, James Cook University of North Queensland.

## Examples

```
data(wine)
pairs(wine[, -1], col = factor2color(wine$Class), cex = 0.2)
```

# Index