# Package 'stripless'

July 23, 2025

**Type** Package

**Title** Structured Trellis Displays Without Strips for Lattice Graphics

**Version** 1.0-3

**Date** 2016-09-09

**Description** For making Trellis-type conditioning plots without strip labels. This is useful for displaying the structure of results from factorial designs and other studies when many conditioning variables would clutter the display with layers of redundant strip labels. Settings of the variables are encoded by layout and spacing in the trellis array and decoded by a separate legend. The functionality is implemented by a single S3 generic strucplot() function that is a wrapper for the Lattice package's xyplot() function. This allows access to all Lattice graphics capabilities in the usual way.

**License** GPL-2 | GPL-3

**Depends** R (>= 3.2.0), utils,lattice

**Suggests** datasets,grDevices,knitr,faraway,rmarkdown

**Imports** stats,graphics,grid

**LazyData** true

**VignetteBuilder** knitr

**RoxygenNote** 5.0.1

**NeedsCompilation** no

**Author** Bert Gunter [aut, cre]

**Maintainer** Bert Gunter <bgunter@comcast.net>

**Repository** CRAN

**Date/Publication** 2016-09-12 08:11:45

## Contents

---

`stripless-package` *Trellis Displays Without Strips For Lattice Graphics*

---

## Description

Stripless provides a simple interface to make trellis plots without strip labels using lattice graphics. Instead of having strip labels identify each panel's settings of the conditioning variables, their position in a structured row/column layout encodes this information, and a legend decodes them. This avoids cluttering the plot area with redundant layers of strip labels when there are many conditioning variables, as is often the case for factorial designs, especially the (fractions of) two level factorials widely used in industry.

## Details

The package consists of a generic `strucplot` function and methods, an associated `print` method for objects inheriting from class `"structured"`, and some (currently 1) additional trellis panel functions. This functionality is implemented as a wrapper to lattice's `xyplot` function, so all other plot specifications (panel functions, colors, titles, etc.) are given through the usual `xyplot` arguments.

See `strucplot` for a full explanation of the structured plotting paradigm, how `strucplot` implements it, and examples illustrating how it works.

## Note

The author has made a considerable effort to provide clear, complete documentation of the package's functionality. He would therefore appreciate receiving any reports of errors, inconsistencies, or infelicities in the Help docs, as well as suggestions for improvement in the docs or underlying package functionality.

---

`defaultStrucLegend` *Default legend function for* `strucplot` *displays.*

---

## Description

A structure legend is always printed on the console. It can also be optionally added to the trellis plot. This function constructs a default legend for this option.

## Usage

```
defaultStrucLegend(struc, legendLoc = c("top", "right", "bottom", "left",
  "newpage"), legendLab = NA, heading = c("Horizontal", "Vertical"),
  miss = "No Conditioning", abbrevLength = c(0, 0), cex.font = 1,
  cex.lab = 1.25, col = "black", ...)
```

## Arguments

| | |
|---|---|
| struc | The "structure" attribute of a strucplot object |
| legendLoc | One of c("top","right","bottom","left","newpage") The first 4 specify on which side of the trellis display to place the legend. The last indicates that the legend will be plotted at the center of a separate page. |
| legendLab | A character string title for the legend. A value of NA omits the title. Default = NA |
| heading | A character vector of length 2 giving the headings for the horizontal and vertical conditioning variables portions of the legend. Default = c("Horizontal", "Vertical"). |
| miss | A character string to use when there is no conditioning either horizontally or vertically. Default = "No Conditioning" |
| abbrevLength | Default = c(0,0). Either a length 2 vector or a named list to control lengths of factor names and levels in the legend. If a length 2 vector, it gives the minlength argument for the abbreviate function for abbreviating all the factor names and their levels, in that order. A value of 0 for either means "don't abbreviate." For back compatibility, a single numeric y will also be accepted and changed to c(y,0). If a named list, the names must be those of of the conditioning factors to abbreviate, and values length 2 vectors as above to control abbreviation lengths for the corresponding factor names and levels. |
| cex.font | Multiplier for text font size in a trellis legend. Default = 1. |
| cex.lab | Multiplier for legend title font size in a trellis legend. Default = 1.25. |
| col | Text color for text in a trellis legend. Default = "black". |
| ... | Additional arguments to be used in a gp list for controlling text appearance in a trellis legend. See gpar for possibilities. |

## Value

A text grob that can be included as part of the strucplot trellis plot.

## See Also

print.structured

## Examples

```
# Controlling the console and plot legends
 library(datasets) ## for the barley data
#
 out <- strucplot(variety~yield|year*site,data=barley, horizontal=TRUE,
```

```
   panel=panel.dotplot, col = "darkblue",
   scales = list(alternat = 1, y = list(cex=.5)),
   spacings = list(x=0, y=.5))
#
# Default with legend on top; note that no title is the plot legend default
  print(out, legendLoc = "t", abbrev= list(site = c(0,4)))
#
# Include title on plot and reduce default font sizes in red text
  print(out, legendLoc = "T", abbrev = list(site = c(0,6)),
   legendLab = "Structure", cex.lab = 1, cex.font = .75, col = "darkred" )
```

---

`displayStruc` *Construct text For* `strucplot` *legends.*

---

## Description

Construct character strings for print and legend functions to display on the console or lattice plot.

## Usage

```
displayStruc(struc, heading = c("Horizontal", "Vertical"),
  miss = "No Conditioning", abbrevLength = c(0, 0), ...)
```

## Arguments

| | |
|---|---|
| `struc` | The "structure" attribute of a `strucplot` object |
| `heading` | A character vector of length 2 giving the headings for the horizontal and vertical conditioning variables portions of the legend. Default = `c("Horizontal", "Vertical")`. |
| `miss` | A character string to use when there is no conditioning either horizontally or vertically. Default = `"No Conditioning"` |
| `abbrevLength` | Default = `c(0,0)`. Either a length 2 vector or a named list to control lengths of factor names and levels in the legend. If a length 2 vector, it gives the `minlength` argument for the abbreviate function for abbreviating all the factor names and their levels, in that order. A value of 0 for either means "don't abbreviate." For back compatibility, a single numeric y will also be accepted and changed to `c(y,0)`. If a named list, the names must be those of of the conditioning factors to abbreviate, and values length 2 vectors as above to control abbreviation lengths for the corresponding factor names and levels. |
| `...` | Additional arguments, presently ignored |

## Details

This is an exported auxiliary function that constructs the character strings for displaying plot structure on the console and possibly also in a legend on the lattice display. While not intended to be directly called by the user, it is exported for use by those who wish to supply custom legends for the lattice display.

## Value

A character vector of length 2 giving character strings, including \n line breaks.

## Examples

```
require(datasets)
# quakes data
#
# Create and save plot
 out <- strucplot(lat ~ long|cut(mag,5)*cut(depth,4), data = quakes,
  col="blue", main = "Earthquake locations, by magnitude and depth")

 displayStruc(attr(out,"structure"))
```

---

# panel.bars *strucplot Panel Functions*

---

## Description

Panel functions for strucplot.

## Usage

```
panel.bars(..., summaryFUN = function(x) mean(x, na.rm = TRUE),
  col = "darkblue", grid = TRUE, col.grid = "lightgray")
```

## Arguments

| | |
|---|---|
| ... | Arguments to barchart. See panel.barchart. |
| summaryFUN | The function that summarizes replicated response values. Default = fuction(x) mean(x, na.rm = TRUE). |
| col | Color of the bars. Default = "darkblue". |
| grid | Should an appropriate background grid be plotted? Default = TRUE. |
| col.grid | The background grid color. Default = "lightgray". |

## Details

panel.bars A wrapper for panel.barchart that plots bars that summarize the responses at each setting of the conditioning variables. By default, bars are vertical, but setting the optional parameter, horizontal to TRUE plots horizontal bars.

## Examples

```
# A half fraction of a 2^5 full factorial with pseudo-replicate responses
# at each design point,

# Build the design matrix
x <- c(-1,1)
ff <- expand.grid(x,x,x,x)
ff[[5]] <- do.call(mapply,c(FUN=prod,ff))
ff <- ff[rep(1:16,e=2),] ## replicates each row twice
names(ff) <- LETTERS[1:5]

# Add a column for the response
ff$y <-c(155.5, 154.8, 158.4, 156.2, 154.8, 152.4, 159.7, 155.5, 161.8,
159.7, 159, 158.4, 159.7, 157.7, 161.8, 158, 155.9, 151.7, 159,
158, 154.1, 156.9, 158.4, 158.4, 159, 154.8, 158.4, 156.2, 161.1,
156.9, 162.6, 159)

# Plot using panel.bars
strucplot(~ y|., data = ff, panel = panel.bars)

# It is often useful to plot the bars the other way, too
strucplot(~ y|., data = ff, panel = panel.bars, horizontal = TRUE)
```

---

# print.structured *Methods for structured objects.*

---

## Description

Print/plot and summary methods for class "structured" objects.

plot and print methods are the same for "structured" objects

Summary method for "structured" objects.

Print method for "summary.structured" objects.

## Usage

```
## S3 method for class 'structured'
print(x, legendLoc = c("left", "right", "top", "bottom",
  "newpage"), legend = defaultStrucLegend, lbl = "PLOT STRUCTURE",
  abbrevLength = c(0, 0), ...)

## S3 method for class 'structured'
plot(x, legendLoc = c("left", "right", "top", "bottom",
  "newpage"), legend = defaultStrucLegend, lbl = "PLOT STRUCTURE",
  abbrevLength = c(0, 0), ...)

## S3 method for class 'structured'
```

```
summary(object, ...)

## S3 method for class 'summary.structured'
print(x, lbl = "PLOT STRUCTURE", ...)
```

## Arguments

| | |
|---|---|
| x, object | The object to be displayed or summarized. |
| legendLoc | An optional location of a legend describing the plot layout to be used as a legend on the trellis plot. Note that a text legend is *always* printed on the console. If used, this argument specified the position of a legend in the trellis display. It must be one of "left", "right", "top", "bottom", or "newpage" (case doesn't matter and matching is done using match.arg, so legendLoc can be abbreviated). Any of the first four of these will become the name of a component of the legend argument of the xyplot call, and so must not conflict with any in a legend argument that may already be part of the strucplot call.<br>If legendLoc = "newpage", the legend will be plotted centered on a new trellis page. |
| legend | A *function* that constructs a grob to use as a plot legend. It must accept at least 3 arguments named "struc", "legendLoc", and "abbrevLength", and also have a ... argument. The first three will be passed the structure attribute of the object to be plotted and the legendLoc and abbrevLength arguments of the print call. The ... argument will be passed the ... argument of the print call, so additional function arguments should be included there (as name = value pairs, as usual).<br>The default is the defaultStrucLegend function. Its Help file should be consulted for its full argument list. |
| lbl | Label for console legend. Default = "PLOT STRUCTURE" |
| abbrevLength | Default = c(0,0). Either a length 2 vector or a named list to control lengths of factor names and levels in the legend. If a length 2 vector, it gives the minlength argument for the abbreviate function for abbreviating all the factor names and their levels, in that order. A value of 0 for either means "don't abbreviate." For back compatibility, a single numeric y will also be accepted and changed to c(y,0). If a named list, the names must be those of of the conditioning factors to abbreviate, and values length 2 vectors as above to control abbreviation lengths for the corresponding factor names and levels. |
| ... | Further arguments to pass down to either the print methods, the legend function, or print.trellis. Care should be taken to ensure that the names of arguments do not conflict. Note that heading and miss arguments of defaultStrucLegend are also used by the console display. |

## Details

The print and plot methods produce a plot and informative legend for "structured" objects. The plot method is an alias for the print method. The summary method gives a simple summary of the object with a print

## Note

Do not use the `packet.panel` argument of `print.trellis`, as this will totally mess up the display.

## See Also

`print.trellis`, `defaultStrucLegend`

## Examples

```
require(datasets)
# quakes data
#
# Create and save plot
out <- strucplot(lat ~ long|cut(mag,5)*cut(depth,4), data = quakes,
  col="blue", main = "Earthquake locations, by magnitude and depth")

# Summary:
summary(out)

# Default output -- structure legend on console only
   print(out)

# Add legends to the plot on either right or bottom (note partial matching)
   print(out, legendLoc = "right")
   print(out, legendLoc = "b")
#
# Plot the legend by itself on a separate page
   print(out, legendLoc = "newp")
#
# Extra grid "gp" arguments to alter text appearance
   print(out, legendLoc = "b",col="blue",fontface = "italic", abbrev = 5)
#
# ******* Using the "abbrev" argument with the 'barley' data set ****
#
  out <- strucplot(variety~yield|year*site,data=barley, horizontal=TRUE,
  panel=panel.dotplot, col = "darkblue", scales = list(alternat = 1,
  y = list(cex=.5)), spacings = list(x=0, y=.5))

# Default
  print(out)
#
# Abbreviate factor names and levels
  print(out, abbrev = c(3,4))
#
# abbreviate just the levels of 'site' and change the console legend title
  print(out, abbrev = list(site = c(0,4)), lbl = "Structure Key")
#
# Note that the 'abbreviate' argument is shared by console and plot
# legends; as are the optional 'heading' and 'miss' arguments
# by the 'defaultStrucLegend' function.
  print(out,abbrev = list(site = c(0,4)),legendLoc="t",
   heading = c("Left-Right", "Up-Down"))
```

stripless-internal *Unexported Utility and xyLayout Functions*

## Description

Unexported functions not intended to be directly called by users.

For xyLayout: Unexported generic and methods used by strucplot to create, check and/or fix a xyLayout list.

## Usage

```
check2lvl(d, sep = ".")

make_2level_with_center(lvls, sep = ".")

makeSpacings(spacings, levelLength)

chkSpacings(spacings)

xyLayout(xylay, ...)

## S3 method for class 'list'
xyLayout(xylay = list(),
  n = stop("Number of conditioning factors is missing"))

## S3 method for class 'matrix'
xyLayout(xylay, n)

## S3 method for class 'data.frame'
xyLayout(xylay, n)

## Default S3 method:
xyLayout(..., n)
```

## Arguments

| | |
|---|---|
| d | Condition list from strucParseFormula to be checked to see if it represents a 2 level design with center point. |
| sep | sep argument for paste(). |
| lvls | A list of length 3 character vectors that are the level names of factors in a design to be checked. |
| spacings | A non-decreasing vector of positive values |
| levelLength | An integer vector giving the number of levels per factor at each level of the plotting hierarchy |

| | |
|---|---|
| xylay | An appropriate list, matrix, or vector for determining the plot structure. Can also be missing, length 0, etc. |
| n | The number of conditioning factors. The integers in the combined list that is created will be a permutation of 1, 2, . . . ,n . |
| ... | x and/or y vector for xylay list. The remaining component will be constructed as needed. |

## Details

**Brief utility function descriptions:**

**check2lvl** Checks for a 2 level design with center point

**make_2level_with_center** Makes a 2 level design with center point to compare to a fraction of a $3^n$ design

**chkSpacings** Checks and constructs proper spacings list if possible

**makeSpacings** Constructs a vector of spacings values for the "between" argument of xyplot

**xyLayout:** The various methods provide convenience and flexibility for specifying the xyLayout list argument that controls the format of strucplot displays. Essentially any sensible way of specifying xyLayout should work. See the Help page for strucplot for details.

## Value

For xyLayout: a list of class "xyLayout" suitable for the xyLayout argument of strucplot.

## See Also

strucplot

---

# strucParseFormula *Parse Trellis Formula for* strucplot

---

## Description

This is a wrapper for latticeParseFormula that allows the option of using a "." for the conditioning variables (i.e. after "|") instead of explicitly writing them out. When so used, it means "all variables in the data argument *except* those already used to the left of the |". See the Help for strucplot for examples.

## Usage

```
strucParseFormula(form, data = list())
```

## Arguments

| | |
|---|---|
| form | The strucplot formula to be parsed. |
| data | An optional data frame containing values for any variables in the formula. Default = list(), which means that all variables will be looked up in the formula's environment. |

### Details

Note that this is a convenience option only; the conditioning can always be explicitly given. Also note that the two options cannot be combined: either a "." and *only* a "." must be used or *all* the conditioning variables must be written out.

### Value

Same as `latticeParseFormula` with an additional "form" attribute that is the formula used with all conditioning explicitly given.

### See Also

`latticeParseFormula` `xyplot`

### Examples

```
exdat <- data.frame(x = 1:5, alongname = sample( letters[1:3],5, rep=TRUE),
  butalongername = sample(LETTERS[1:2],5, rep = TRUE))
  y <- runif(5)

strucParseFormula (y ~ x| alongname*butalongername, data = exdat)

# The same
strucParseFormula (y ~ x|., data = exdat)

# The 'data' argument is required with '.'
## Not run:
strucParseFormula (y ~ x|.)

## End(Not run)
```

---

## strucplot *Structured Trellis Plots*

---

### Description

Structured Trellis Plots Without Strip Labels

### Usage

```
strucplot(obj, ...)

## S3 method for class 'formula'
strucplot(obj, data = list(), groups = NULL,
  xyLayout = list(), spacings = list(x = 0:9, y = 0:9), center = FALSE,
  ...)
```

```
## Default S3 method:
strucplot(...)

## S3 method for class 'data.frame'
strucplot(obj, col = "darkblue", groups = NULL, ...)

## S3 method for class 'list'
strucplot(obj, ...)

## S3 method for class 'matrix'
strucplot(obj, ...)

## S3 method for class 'lm'
strucplot(obj, newdata = model.frame(obj)[-1],
  ylab = "Predicted Response", predictArgs = NULL, ...)
```

## Arguments

| | |
|---|---|
| obj | Argument determining method dispatch. For the formula method, a xyplot type formula of the form ~y\|f1*f2*...*fn or y~x\|f1*f2*...*fn (where "..." means the actual variable names). Instead of explicitly specifying the conditioning variables, i.e. the variables after the \|, you can instead use a "." after \|. This is interpreted to mean "all variables in the data argument *except* those to the left of the \|". For example, the second formula above could be written as y~x\|. where f1, f2, ..., fn and possibly x and y are the variables in data (typically columns of a data frame).<br>Note 1: Extended formulas and 3-d formulas (see xyplot) are not implemented.<br>Note 2: For the lm method, the model object should contain a model component. See predict.lm and Help pages for predict methods for classes inheriting from "lm" for other arguments and for exactly what is plotted.<br>It is preferable that this be the the first argument of the call. |
| data | For the formula method, a data frame (or more precisely, anything that is a valid envir argument in eval, e.g., a list or an environment) containing values for any variables in the formula, as well as groups and subset if applicable. If not found in data, or if data is unspecified, the variables are looked for in the environment of the formula. |
| groups | The groups parameter of xyplot |
| xyLayout | In its most general form, a list with named "x" and "y" components, either or both of which can be missing (or NULL). If there are *n* conditioning factors and both x and y are given, then the combined components (e.g. via unlist) must be a permutation of the sequence 1, 2,..., n (with no duplicates).<br>The integers in x specify the indices of the conditioning factors and their levels in the x direction hierarchy. Correspondingly for y. For a fuller explanation of how this controls the plot layout, see the **Overview** section and the examples.<br>For convenience, the xyLayout argument can be specified in several other ways. The basic idea is that the full list will be *filled in* "appropriately" if possible. Specifically, this means: |

- If both components are missing, NULL, or of zero length the list is constructed by splitting the conditioning factors equally in the horizontal and vertical directions, with horizontal gettting one more if the number of conditioning variables is odd;
- If only one component of the list is given, the missing component gets the remaining factors, if any, in order(note that this means if both directions are not in order low to high, then both must be explicitly specified in the xyLayout list);
- If component names are missing or empty(i.e. ""), the first one is "x" and the second is "y". Nonmissing component names must be "x" and/or "y" and must be unique;
- A 1 or 2 column matrix can be used instead of a list with an even number of conditioning variables. If the column names are "x" or "y", they will be used. Otherwise, the x component is first and y second;
- An unnamed vector can be given in place of a list with a single unnamed component (which would be assumed to be the "x" component)

As a result, all the following would produce exactly the same layout for n = 5 conditioning factors when used as the xyLayout argument:

- `list(x = 1:3, y = 4:5)`
- `list (x = 1:3)`
- `list (y = 4:5)`
- argument missing or NULL
- `1:3`
- `cbind(y = 4:5)`
- `matrix(1:3,ncol=1)`

Note that arbitrary integer vectors can be used, not just (ordered) sequences; e.g. the following are correct and equivalent when there are 6 conditioning factors:

- `list(y = c(1,5,3), x = c(2,4,6))`
- `cbind(y=c(1,5,3), x=c(2,4,6))`
- `cbind(y = c(1,5,3))`

*But be careful*: `xyLayout = c(2,4,6)` is equivalent to `xyLayout = list(x = c(2,4,6), y = c(1,3,5))`, which is different than the previous, since order matters!

`spacings` A list with x and y components that are nondecreasing sequences of positive vaues. The ith value in the sequence gives the spacing in character heights between the sets of panels at each level for the ith conditioning variable in the component's direction. If either a single vector or a list with single component is given, it is replicated. See the **Overview** for further explanation. Default is `list(x=0:9,y=0:9)`

`center` Logical, default FALSE. If the conditioning factors constitute a 2-level design with a center point and center is TRUE, a compact trellis plot that omits panels for the missing settings of the factors is drawn. If FALSE or the design is not of this form, all panels are shown, which may be informative in any case to better visualize the design sparsity.

| | |
|---|---|
| col | Fill color for the data frame method. |
| newdata | For the lm method and objects inheriting from the lm class. The 'newdata' argument for the associated predict method. Consult predict.lm for details. |
| ylab | For the lm method. Optional y axis label for predicted values. |
| predictArgs | For the lm method. A **list** of optional named arguments for the relevant predict method. These are will be used as part of the args list when invoking the relevant predict method using do.call. |
| ... | Further arguments to panel functions, methods, or xyplot. |

## Details

A structured display is a trellis display without strip labels. The factors and levels are encoded by plot layout and spacing and decoded by a separate legend. See the **Overview** below for a detailed explanation.

## Value

The formula method returns an object of type `c("structured", "trellis")`; or `c("doe","structured","trellis")` for 2-level designs with a center point. This is a trellis object with additional `structure` and `formula` attributes. The lm method returns a class of `c("modelFit","structured","trellis")`. At present, this extension is ignored, but future print/plot methods may take advantage of it. The other methods return the same object as the formula method.

The `structure` attribute provides xyLayout information. It is a list with x and y components, each of which in turn is a list, one of which may be empty (i.e. of length 0). The components of the x list have names of the conditioning variables in the horizontal direction of the layout with values the variable values; analogously for the y list.

The `formula` attribute gives the actual formula used in the trellis call, i.e. with the actual conditioning variable names substituted for ".".

## Methods (by class)

- `formula`: Formula Method
- `default`: Default method prints an error message
- `data.frame`: Data frame method with no response to show design structure
- `list`: Converts to a data frame and calls the data.frame method
- `matrix`: Converts to a data frame and calls the data.frame method
- `lm`: Plots predicted values for fitted models inheriting from class "lm"

## Overview

The trellis display paradigm breaks down when there are more than 2 or 3 conditioning variables, because the plotting area becomes cluttered with multiple layers of strip labels identifying the panel settings. This is especially a problem if one wants to show the structure or results of studies that are (fractions of) factorial designs with the design factors as conditioning variables. For example, in industrial type experiments, it is not uncommon to have 5 or more experimental factors.

It is also often the case that there are multiple responses – e.g. several different characteristics of a product or a functional response like an IR spectrum, MRI scan, or surface plot. It can be desirable to display such results so that direct visual comparison of the complex responses can be made.

The `strucplot` function enables such functionality by omitting the strip labels and laying out the panels in a regular array, a 'xyLayout' in which the position of the panels rather than their strip labels identifies the variable settings.

Because the lattice package already has these capabilities, `strucplot` is implemented as a wrapper to lattice's `xyplot` function. This provides a convenient interface, simplifies the code and, most importantly, allows the user to access all of lattice's (and the underlying grid graphics') functionality in the usual way. Only two extra arguments – 'xyLayout' and 'spacings' – are used to do this, although the default 'spacings' argument normally need not be changed. (There is also a third 'center' argument for 2 level factorial designs, which are commonly used in industrial experiments, that is explained below.)

## How does strucplot() work?

Suppose that the data consist of a numeric vector response y for 4 conditioning factors, f1, f2, f3, and f4, where f1 and f2 each have 2 levels, f3 has 3 levels, and f4 has 4 levels. (Because the conditioning variables are coerced to factors by `factor`, if level orderings other than that given by these coercions are wanted, the user should do them explicitly before calling `strucplot`).

Then the call, `strucplot(~y|f1*f2*f3*f4)` would produce a trellis plot without strip labels with 12 (= 3 x 4) rows and 4 (= 2 x 2) columns of panels, some of which may be empty if the corresponding factor settings are missing. The default xyLayout argument that produces this is: `xyLayout = list(x = 1:2, y = 3:4)`. It splits the conditioning variables as evenly as possible into 2 groups with the x component getting the first 2 variables, f1 and f2, and the y component getting the second 2 variables, f3 and f4. (if there are an odd number of variables, the x component gets one more variable than the y). This means the levels of the x variables, f1 and f2, vary from left to right across each row. For the y variables, if `as.table = TRUE`, the default, the levels of the y variables, f3 and f4, vary from top to bottom down each column; otherwise if `as.table = FALSE`, from bottom to top up each column. Since there are 4 combinations of levels for the x variables and 12 for y, this gives a 12 row by 4 column display.

The panels are displayed in each direction in reverse lexicographic order, where the 'alphabets' are the factor levels. This means that the first variable changes the fastest; the second the next fastest, and so on. Using (i,j) to denote setting in which the first factor is at the ith level and the second is at the jth, this translates to (for `as.table = TRUE`):

**Row ordering** (f1,f2): (1,1), (2,1), (1,2), (2,2)

**Column ordering from top down** (f3, f4): ((1,1), (2,1), (3,1), (1,2), (2,2), ... , (1,4), (2,4), (3,4)

If one component is missing, if it is x, there will only be 1 column; if it is y, only 1 row. The nonmissing component must still be correctly specified to provide the panel ordering.

## Panel spacing

Variable spacing between the panels hierarchically groups them to identify their settings. The default spacing for both x and y directions is 0:9. This means that panels corresponding to the first, fastest changing, variable are separated by 0 units (= character heights); groups of panels at each

fixed level of the second next fastest changing variable are separated by 1 unit; groups of groups of panels at fixed levels of the third are separated by 2 units; and so forth.

For the example, this means that the row spacing would look like ('X' indicates a panel): XX XX . And for columns it would be going down: XXX XXX XXX XXX . The spacings can be different for x and y, but this is usually unnecessary.

### Effective xyLayout specification

The default layout is often enhanced by changing the order of the factors and the xyLayout; for example, ordering the factors from those with the least change among levels to the most, or vice-versa; or setting the "most important" factors along rows to facilitate visual comparison.

The order of the variables – and hence which vary in the x or y direction – is given both by the left to right order of the conditioning in the formula and the xyLayout argument. Thus, in the example, conditioning with `~|f3*f1*f2*f4` and setting the layout with `xyLayout = list(x=4, y=3:1)` is equivalent to `~|f4*f2*f1*f3` and `xyLayout = list(x=1,y=2:4)` and produces a display with 12 rows and 4 columns in which the row panels now correspond to the 4 f4 levels and the column panels to the levels of f2, f1, and f3. This redundancy is deliberate: it allows changing layouts via the xyLayout argument to avoid rewriting a formula with long names in a different order.

### 2 level designs with a center point – the 'center' argument

Finally, the 'center' argument, a logical with default = `TRUE`, controls the display when the conditioning factors are arranged as a 2 level factorial design with a single "pseudo"-center point. "pseudo" here means that the settings of the factors at the center need not be exactly in the middle for numeric factors. If the design is not of this form, the 'center' argument is ignored.

For such designs, when `center = TRUE`, a more compact display will be drawn in which a center panel corresponding to the center point is shown as the single panel in its row and column, but all other empty panels corresponding to settings where some of the conditioning variables are at their mid levels and some are not, are omitted. Examples are given below.

If it is desired to show all the empty panels, which can be useful to informatively represent the actual design sparsity, set `center = FALSE`.

## Note

Because 'xyLayout' and the number of levels in the conditioning variables determine the plot structure, a 'layout' argument in the call will be ignored. Other `xyplot` arguments that are ignored are 'skip','between', 'drop.unused.levels', 'strip','perm.cond',and 'index.cond'. All other `xyplot` should work as expected.

## See Also

lattice

## Examples

```
# Compare standard xyplot vs. strucplot of quakes data in datasets package.
#  Cut depth into 6 groups and magitude into 5.

require(datasets)
```

```
# Note that as.table = TRUE is used to make depths increase
# down the page. For strucplot(), this is the default.

xyplot(lat ~ long|cut(mag,5)*cut(depth,6),  data = quakes, col="blue",
 as.table = TRUE,type = c("g","p"))

# Compare to:
strucplot(lat ~ long|cut(mag,5)*cut(depth,6), data = quakes, col="blue",
   type = c("g", "p"))

# Visualizing designs:

# A half fraction of a 2^5 (a 2^(5-1)) design

# Build the design matrix
 ff <- do.call(expand.grid,rep(list(c(-1,1)),4))
 ff[[5]] <- do.call(mapply,c(FUN = prod,ff))
  names(ff) <- LETTERS[1:5]

# Show the design
strucplot(ff)

# Plotting a 2 level design with a center point

# Add a center point to ff and plot
 ffCenter <- rbind(ff,rep(0,5))
 strucplot(ffCenter)

# Use center = TRUE for a more compact display and show legend below.
 print(strucplot(ffCenter, center = TRUE),legendLoc = "bottom")

# The "npk" data. See help("npk") for details.
# Visualize design with blocks the vertical factor and the rest horizontal
 strucplot(npk[,-5], xyLay = list(x = 2:4, y =1))

 # Plot the yield
 strucplot(~yield |., xyLay = list(x=2:4, y=1),data = npk, col = "darkblue",
   panel = function(...){
     panel.grid(h = -1, v = 0)
     panel.xyplot(...)}
   )

# It may be more informative to plot bars instead of points.
# See help(panel.bars) for details.
#
# Note also "shortcut" ways to specify the xyLayout
 strucplot(~yield |., xyLay = list(x=2:4),data = npk,
           panel = panel.bars)

# Include a conditioning variable in the formula to reduce the
# dimensionality of conditioning. Show legend on right of plot.
 print(strucplot(yield ~ N|., xyLay = 2:3, data = npk,
           panel = panel.bars), legendLoc = "right")
```

```
# Use the horizontal = TRUE argument of panel.bars to plot the bars
# horizontally. The left and right hand sides of the formula must also
# be switched for 2-sided formulas (not for 1-sided).
 strucplot( N ~ yield |., xyLay = list(y=1), data = npk,
           panel = panel.bars, horizontal = TRUE)

# Fit a linear model with all main effects and 2 factor interactions in N,P,K
# and plot the fits, using the "newdata" argument to plot predictions at
# non-design points).
 require("stats")
   npk.aov <- aov(yield ~ block + (N+P+K)^2, data = npk)
   full <- do.call(expand.grid,lapply(npk[,-5],levels))
   plot(strucplot(npk.aov,  xyLay = list(x = 2:4),panel = panel.bars,
           newdata = full),legendLoc = "bottom")

  # Compare to a grouped plot:
  ypred <- predict(npk.aov, new = full)
   plot(
     strucplot(ypred ~ N|K*block, groups = full$P, data = full,
       panel= function(x,y, groups, subscripts,cex=1.25,...){
         panel.grid(h=-1, v=0)
         panel.superpose(x,y,cex= cex, type = c("p","l"),...,
                   panel.groups = panel.xyplot,
                   groups=groups, subscripts =subscripts)},
       auto.key = list(points=FALSE,lines=TRUE, columns = 2,
              title = "P",cex.title=1), ylab = "Predicted Response" ),
     legendLoc = "right")

## Cleanup
rm(full, npk.aov, ypred,ff,ffCenter)
```

# Index