Package ‘tactile’

April 23, 2018

Title New and Extended Plots, Methods, and Panel Functions for 'lattice'

Version 0.2.0

Description Extensions to 'lattice', providing new high-level functions, methods for existing functions, panel functions, and a theme.

Depends R (>= 3.4.0), lattice

Imports grDevices, grid, gridExtra, latticeExtra, MASS, RColorBrewer, stats, utils

Suggests covr, forecast, knitr, rmarkdown, testthat, zoo

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

URL https://github.com/jolars/tactile

BugReports https://github.com/jolars/tactile/issues

VignetteBuilder knitr

NeedsCompilation no

Author Johan Larsson [aut, cre] (<https://orcid.org/0000-0002-4029-5945>),
  Deepayan Sarkar [ctb, cph] (lattice),
  Brian Ripley [ctb] (stats::plot.acf)

Maintainer Johan Larsson <johanlarsson@outlook.com>

Repository CRAN

Date/Publication 2018-04-23 16:45:09 UTC

R topics documented:

  bubbleplot .................................................. 2
  bwplot2 .................................................. 3
  diag.panel.splom.density ................................. 4
bubbleplot

Description

Draws bubbleplots – trivariate plots where the third dimension is mapped to the size of the points drawn on the screen.

Usage

bubbleplot(x, data = NULL, ...)

## S3 method for class 'formula'
bubbleplot(x, data = NULL, maxsize = 3,
  bubblekey = TRUE, panel = panel.bubbleplot, groups = NULL,
  subset = TRUE,
  drop.unused.levels = lattice:::getOption("drop.unused.levels"), ..., outer,
  allow.multiple)

Arguments

x A formula of the form \( z \sim x \times y \), where \( x \) and \( y \) have the usual interpretation in trellis graphics (see \texttt{lattice:::xyplot()}) and \( z \) is mapped to the size of bubbles.

data A data.frame, list or environment wherein the formula and groups arguments can be evaluated.
Further arguments to pass to `lattice::xyplot()`.

`maxsize`  Maximum size (in cex) for the bubbles.

`bubblekey`  Set to TRUE to draw an informative legend about the bubbles. Uses `lattice::draw.key()`. See the `key` section of the documentation in `lattice::xyplot()`. If both `auto.key` and `bubblekey` are given and their space arguments (positions) conflict, `bubblekey` will silently override the position of `auto.key`.

`panel`  See `lattice::xyplot()`. Here, we are passing an additional variable, z, which is then used in `panel.bubbleplot()`.

`groups`  See `lattice::xyplot()`

`subset`  See `lattice::xyplot()`

`drop.unused.levels`  See `lattice::xyplot()`

`outer`  Ignored.

`allow.multiple`  Ignored.

Value

An object of class "trellis". The `update` method can be used to update components of the object and the `print` method (usually called by default) will plot it on an appropriate plotting device.

Author(s)

Johan Larsson

Examples

```r
bubbleplot(disp ~ hp * wt, groups = cyl, data = mtcars, auto.key = TRUE)
bubbleplot(disp ~ hp * mpg | factor(cyl), groups = gear, data = mtcars, auto.key = TRUE)
```

Description

An extended version of `lattice::bwplot()`. The only modification is to group and stack box plots if groups is provided.

Usage

```r
bwplot2(x, data = NULL, ...)
```

```r
## S3 method for class 'formula'
bwplot2(x, data = NULL, allow.multiple = is.null(groups))
```
```r
drop.unused.levels = lattice.getOption("drop.unused.levels"), ..., 
subset = TRUE)

### S3 method for class 'numeric'
bwplot2(x, data = NULL, xlab = deparse(substitute(x)), 
...)

Arguments

x see `lattice::bwplot()
data see `lattice::bwplot()
... arguments passed down to `lattice::bwplot()
allow.multiple see `lattice::bwplot()
outer see `lattice::bwplot()
auto.key see `lattice::bwplot()
groups see `lattice::bwplot()
drop.unused.levels see `lattice::bwplot()
subset see `lattice::bwplot()
xlab see `lattice::bwplot()

Value

An object of class "trellis". The `update` method can be used to update components of the object
and the `print` method (usually called by default) will plot it on an appropriate plotting device.

Examples

bwplot2(variety ~ yield, 
groups = site, 
data = barley, 
par.settings = tactile.theme())
```

---

diag.panel.splom.density

*Diagonal Density Panels*

---

**Description**

Plots univariate density estimates to be used in a `lattice::splom()` call with the `diag.panel` argument.

**Usage**

```r
diag.panel.splom.density(x, bw = "nrd0", adjust = 1, kernel = "gaussian", 
weights = NULL, n = 512, ...)
```
Arguments

x data vector corresponding to that row / column (which will be the same for diagonal 'panels').

bw the smoothing bandwidth to be used. The kernels are scaled such that this is the standard deviation of the smoothing kernel. (Note this differs from the reference books cited below, and from S-PLUS.)

bw can also be a character string giving a rule to choose the bandwidth. See \texttt{bw.nrd}.

The default, "nrd0", has remained the default for historical and compatibility reasons, rather than as a general recommendation, where e.g., "SJ" would rather fit, see also Venables and Ripley (2002).

The specified (or computed) value of bw is multiplied by adjust.

adjust the bandwidth used is actually adjust*bw. This makes it easy to specify values like 'half the default' bandwidth.

kernel a character string giving the smoothing kernel to be used. This must partially match one of "gaussian", "rectangular", "triangular", "epanechnikov", "biweight", "cosine" or "optcosine", with default "gaussian", and may be abbreviated to a unique prefix (single letter).

"cosine" is smoother than "optcosine", which is the usual 'cosine' kernel in the literature and almost MSE-efficient. However, "cosine" is the version used by S.

weights numeric vector of non-negative observation weights, hence of same length as x.

The default \texttt{NULL} is equivalent to weights = rep(1/nx, nx) where nx is the length of (the finite entries of) x[].

n the number of equally spaced points at which the density is to be estimated.

When n > 512, it is rounded up to a power of 2 during the calculations (as \texttt{fft} is used) and the final result is interpolated by \texttt{approx}. So it almost always makes sense to specify n as a power of two.

... Further arguments passed on to \texttt{lattice::diag.panel.splom()} and \texttt{lattice::panel.lines()}.

See Also

\texttt{lattice::diag.panel.splom()}, \texttt{lattice::splom().stats::density()}.

Examples

\begin{verbatim}
  splom(~ iris[1:4],
         data = iris,
         diag.panel = diag.panel.splom.density,
         pscales = 0)
\end{verbatim}
## Description

A data set that has been manually transcribed from Table 5 of Elkins and Grove’s *Ternary feldspar experiments and thermodynamic models*.

## Usage

feldspar

## Format

A data frame of 40 rows and 7 columns:

- **Experiment** The ID of the experiment
- **Feldspar** Coexisting feldspars, *Alkali* or *Plagioclase*
- **Or** Proportion of orthoclase
- **An** Proportion of anorthite
- **Ab** Proportion of albite
- **Temperature** Temperature of the reaction (degrees centigrade)
- **Pressure** Pressure of the reaction (bars)

## Abstract

This paper reports the results of 20 experiments in which mixes of two or three feldspars were reacted to produce coexisting plagioclase feldspar (PF) and alkali feldspar (AF). Starting materials with similar bulk compositions were prepared using different combinations of two and three minerals, and experiments were designed to produce similar AF and PF minerals in the experimental products from different starting binary and ternary compositions. The coexisting AF and PF compositions produced as products define compositional fields that are elongate parallel to the ternary solvus. In 11 experiments reaction was sufficient to produce fields of coexisting AF and PF, or AF, PF, and melt with a bulk composition close to that of the starting mixture. In six experiments significant reaction occurred in the form of reaction rim overgrowths on seeds of the starting materials. Three experiments produced AF, PF, and melt from a natural granite starting material. A two-feldspar thermometer is presented in which temperature is constrained by equilibria among all three components - Albite, Orthoclase, and Anorthite - in coexisting ternary feldspars.

## Source

panel.bubbleplot  
Panel Function for Bubble Plots

Description
Panel Function for Bubble Plots

Usage
panel.bubbleplot(x, y, z, groups = NULL, subscripts, cex = NULL, ...)

Arguments
- x: variables to be plotted in the scatterplot
- y: variables to be plotted in the scatterplot
- z: A numeric vector that areas of circles will be mapped to.
- groups: Grouping variable (see lattice::xyplot()).
- subscripts: A vector of indexes to specify which observation to plot. Normally does not need to be provided by the user.
- cex: Is used internally and user settings will be ignored.
- ...: Further arguments to pass to lattice::panel.xyplot().

Value
Plots a layer inside a panel of a lattice plot.

panel.ci  
Panel function for confidence interval

Description
Panel function for confidence interval

Usage
panel.ci(x, y, lower, upper, groups = NULL, subscripts, col, fill = if (is.null(groups)) plot.line$col else superpose.line$col, alpha = 0.15, lty = 0, lwd = if (is.null(groups)) plot.line$lwd else superpose.line$lwd, grid = FALSE, ..., col.line = if (is.null(groups)) plot.line$col else superpose.line$col)
Arguments

**x**  
variables to be plotted in the scatterplot

**y**  
variables to be plotted in the scatterplot

**lower**  
lower confidence limits

**upper**  
upper confidence limits

**groups**  
an optional grouping variable. If present, `panel.superpose` will be used instead to display each subgroup

**subscripts**  
see `lattice::xyplot()`

**col**  
default colours are obtained from `plot.symbol` and `plot.line` using `trellis.par.get`.

**fill**  
other graphical parameters. `fill` serves the purpose of `bg` in `points` for certain values of `pch`

**alpha**  
opacity for the fill

**lty**  
other graphical parameters. `fill` serves the purpose of `bg` in `points` for certain values of `pch`

**lwd**  
other graphical parameters. `fill` serves the purpose of `bg` in `points` for certain values of `pch`

**grid**  
A logical flag, character string, or list specifying whether and how a background grid should be drawn. This provides the same functionality as `type="g"`, but is the preferred alternative as the effect `type="g"` is conceptually different from that of other `type` values (which are all data-dependent). Using the `grid` argument also allows more flexibility.

Most generally, `grid` can be a list of arguments to be supplied to `panel.grid`, which is called with those arguments. Three shortcuts are available:

- **TRUE**: roughly equivalent to `list(h = -1, v = -1)`
- **"h"**: roughly equivalent to `list(h = -1, v = 0)`
- **"v"**: roughly equivalent to `list(h = 0, v = -1)`

No grid is drawn if `grid = FALSE`.

...  
Extra arguments, if any, for `panel.xyplot`. In most cases `panel.xyplot` ignores these. For types "r" and "smooth", these are passed on to `panel.lmline` and `panel.loess` respectively.

**col.line**  
default colours are obtained from `plot.symbol` and `plot.line` using `trellis.par.get`.

Examples

```r
mod <- lm(Petal.Width ~ Petal.Length*Species, data = iris)
newdat <- expand.grid(Petal.Length = seq(1, 7, by = 0.1),
                     Species = c("setosa", "versicolor", "virginica"))
pred <- predict(mod, newdat, interval = "confidence")

dd <- cbind(newdat, pred)

xyplot(fit ~ Petal.Length, groups = Species, data = dd,
       prepanel = prepanel.ci, auto.key = list(lines = TRUE, points = FALSE),
       ylab = "Petal Width",
       xlab = "Petal Length",
```
Panel function to go along with `lattice::qqmath()` and `lattice::panel.qqmathline()`. Adds filled confidence bands to the Q-Q-plot.

**Usage**

```r
classification.qmathci(x, y = x, distribution = qnorm, probs = c(0.25, 0.75),
qtype = 7, groups = NULL, ci = 0.95, alpha = 0.25,
col = trellis.par.get("plot.line")$col, ..., col.line)
```

**Arguments**

- `x` The original sample, possibly reduced to a fewer number of quantiles, as determined by the `f.value` argument to `qqmath`
- `y` an alias for `x` for backwards compatibility
- `distribution` quantile function for reference theoretical distribution.
- `probs` numeric vector of length two, representing probabilities. Corresponding quantile pairs define the line drawn.
- `qtype` the type of quantile computation used in `quantile`
- `groups` optional grouping variable. If non-null, a line will be drawn for each group.
- `ci` Confidence level
- `alpha` Alpha level for the color fill
- `col` Color fill for the confidence bands.
- `...` Arguments passed to `lattice::panel.superpose()` and `lattice::panel.polygon()` for plotting.
- `col.line` Color fill for the confidence bands. Is used internally by `lattice::panel.superpose()` and should generally not be changed.

**Details**

The function tries to figure out the density function counterpart to that provided in the argument `distribution` by regular expressions.

**Value**

Augments a trellis plot panel, such as that created by `lattice::qqmath()`, with confidence levels.
Author(s)

Johan Larsson.

See Also

lattice::panel.qqmathline(), lattice::qqmath(), and lattice::panel.qqmath().

Examples

qqmath(~ height | voice.part, aspect = "xy", data = singer,
     prepanel = prepanel.qqmathline,
     panel = function(x, ...) {
       panel.qqmathci(x, ...)
       panel.qqmathline(x, ...)
       panel.qqmath(x, ...)
     })
panel.ternaryplot.clip

  density_breaks  Breaks for the density plot if used (see panel.ternaryplot.density()).
xlab             X axis label (the left dimension)
ylab             Y axis label (the right dimension)
zlab             Z axis label (the top dimension)
xlab.default     Internal argument
ylab.default     Internal argument
zlab.default     Internal argument
...              Arguments passed down to subsequent panel functions.

Value

Plots a layer inside a panel of a lattice plot.

See Also

The building blocks of this function are available as the separate panel functions panel.ternaryplot.xyplot(), panel.ternaryplot.grid(), panel.ternaryplot.scales(), panel.ternaryplot.clip(), panel.ternaryplot.response(), and panel.ternaryplot.density() in case the user would like to get complete control of the customization.

panel.ternaryplot.clip

Plot Region Clipping for Ternary Plots

Description

Plot Region Clipping for Ternary Plots

Usage

panel.ternaryplot.clip(xl = current.panel.limits()$x,
yl = current.panel.limits()$y, border = "transparent", col = if
  (background$col == "transparent") "#FFFFFF" else background$col)

Arguments

  xl               X axis limits
  yl               Y axis limits
  border           Border color
  col              Polygon fill

Value

Plots a layer inside a panel of a lattice plot.
**panel.ternaryplot.density**

*Two-Dimensional Density Estimation for Ternary Plots*

**Description**

Two-Dimensional Density Estimation for Ternary Plots

**Usage**

```r
panel.ternaryplot.density(x, y, z, subscripts, n = 100, region = TRUE,
                          contour = FALSE, labels = isTRUE(contour), density_breaks = NULL, ...)
```

**Arguments**

- `x`  Numeric vector
- `y`  Numeric vector
- `z`  Numeric vector
- `subscripts` See `lattice::panel.xyplot()`.
- `n`  Number of grid points in each direction. Can be scalar or a length-2 integer vector.
- `region` Fill density or response estimates with a color gradient.
- `contour` Draw contour lines for density and response estimates.
- `labels` Label contour lines.
- `density_breaks` Breaks for the density plot if used (see `panel.ternaryplot.density()`).
- `...` Arguments that will be passed on to `lattice::panel.lines()`, `lattice::panel.polygon()`, and `lattice::panel.text()`.

**Value**

Plots a layer inside a panel of a lattice plot.

---

**panel.ternaryplot.grid**

*Reference Grid for Ternary Plot*

**Description**

Reference Grid for Ternary Plot
**Usage**

```r
panel.ternaryplot.grid(at = seq.int(0, 1, by = 0.2),
  alpha = reference.line$alpha, col = reference.line$col,
  lty = reference.line$lty, lwd = reference.line$lwd)
```

**Arguments**

- `at` Where to draw the reference lines
- `alpha` Alpha
- `col` Color
- `lty` Line type
- `lwd` Line weight

**Value**

Plots a layer inside a panel of a lattice plot.

---

**panel.ternaryplot.response**

*Response Panels for Ternary Plots*

**Description**

Response Panels for Ternary Plots

**Usage**

```r
panel.ternaryplot.response(x, y, z, subscripts, response, region = TRUE,
  contour = TRUE, labels = isTRUE(contour), fun = c("glm", "lm"),
  formula = response ~ poly(x, y), ...)
```

**Arguments**

- `x` Numeric vector
- `y` Numeric vector
- `z` Numeric vector
- `subscripts` See `lattice::panel.xyplot()`.
- `response` An optional response variable
- `region` Fill density or response estimates with a color gradient.
- `contour` Draw contour lines for density and response estimates.
- `labels` Label contour lines.
- `fun` Function to apply to the response variable.
- `formula` Formula for the function.
- `...` Arguments passed on to `lattice::panel.lines()`, `lattice::panel.polygon()`, `lattice::panel.text()`.
panel.ternaryplot.scales

 Axes and Labels for Ternary Plots

Description

Axes and Labels for Ternary Plots

Usage

panel.ternaryplot.scales(xlab, ylab, zlab, xlab.default, ylab.default, zlab.default, at = seq.int(0, 1, by = 0.2), ...)

Arguments

xlab Labels, have to be lists. Typically the user will not manipulate these, but instead control this via arguments to cloud directly.
ylab Labels, have to be lists. Typically the user will not manipulate these, but instead control this via arguments to cloud directly.
zlab Labels, have to be lists. Typically the user will not manipulate these, but instead control this via arguments to cloud directly.
xlab.default for internal use
ylab.default for internal use
zlab.default for internal use
at Where to draw tick marks.
... Currently ignored.

Value

Plots a layer inside a panel of a lattice plot.
panel.ternaryplot.xyplot

Ternary Plot Wrapper for lattice::xyplot

Description

This mainly exists to enable users to string together their own ternary plot functions.

Usage

panel.ternaryplot.xyplot(x, y, z, subscripts, ...)

Arguments

- `x`: Numeric vector of values in the original space
- `y`: Numeric vector of values in the original space
- `z`: Numeric vector of values in the original space
- `subscripts`: see `lattice::xyplot()`.
- `...`: Arguments that are passed on to `lattice::panel.xyplot()`.

Value

Plots a layer inside a panel of a lattice plot.

prepanel.ci

Prepanel for ciplot

Description

Prepanel for ciplot

Usage

prepanel.ci(x, y, lower, upper, subscripts, groups = NULL, ...)

Arguments

- `x`: `x` and `y` values, numeric or factor
- `y`: `x` and `y` values, numeric or factor
- `lower`: lower confidence limits
- `upper`: upper confidence limits
- `subscripts`: See `xyplot`. Whenever appropriate, calculations are done separately for each group and then combined.
- `groups`: See `xyplot`. Whenever appropriate, calculations are done separately for each group and then combined.
- `...`: other arguments, usually ignored
**Examples**

```r
mod <- lm(Petal.Width ~ Petal.Length*Species, data = iris)
newdat <- expand.grid(Petal.Length = seq(1, 7, by = 0.1),
                       Species = c("setosa", "versicolor", "virginica"))
pred <- predict(mod, newdat, interval = "confidence")
dd <- cbind(newdat, pred)

xyplot(fit ~ Petal.Length, groups = Species, data = dd,
       prepanel = prepanel.ci,
       ylab = "Petal Width",
       xlab = "Petal Length",
       lower = dd$lw, upper = dd$upr, alpha = 0.3,
       panel = function(...) {
         panel.ci(..., grid = TRUE)
         panel.xyplot(type = "l", ...)
       })
```

**qqmath.zoo**

**Q-Q Plots for Zoo Objects**

**Description**

Draw quantile-Quantile plots of a sample against a theoretical distribution, possibly conditioned on other variables.

**Usage**

```r
## S3 method for class 'zoo'
qqmath(x, data = NULL, xlab = "Theoretical quantiles",
       ylab = "Sample quantiles", ref = TRUE, ci = TRUE, ...)
```

**Arguments**

- `x`: A zoo object
- `data`: Ignored
- `xlab`: X axis label
- `ylab`: Y axis label
- `ref`: Plot a reference line via `lattice::panel.qqmathline()`.
- `ci`: Plot confidence levels via `panel.qqmathci()`.
- `...`: Parameters to pass on to `lattice::qqmath()`.

**Value**

Plots and returns a trellis object.
Author(s)

Original by Deepayan Sarkar.

See Also

lattice::qqmath(), zoo::zoo(), lattice::panel.qqmathline().

Examples

```r
if (require(zoo))
  qqmath(zoo(lh))
```

Description

A custom theme for lattice that tries to make away with some of the (in this author's opinion) excessive margins that result from the default settings. It also provides a different color theme based partly on latticeExtra::custom.theme().

Usage

tactile.theme(fontsize = c(12, 8), color = TRUE, ...)

Arguments

- `fontsize` A vector of two numeric scalars for text and symbols respectively.
- `color` Colorized theme.
- `...` Additional named options.

Details

The theme currently modifies the default lattice theme so that

- paddings (margins) are minimized,
- axis tick lengths are halved, and
- title size is decreased slightly.

Value

A list of graphical parameters that for instance could be supplied inside a call to lattice::xyplot() or set via lattice::lattice.options().
Description

A ternary plot is a triangular diagram that displays proportions of three variables. It can be used to map three-dimensional data to a two-dimensional surface with the caveat that the data's original scales are lost (unless it was proportional data to begin with).

Usage

ternaryplot(x, data, ...)  

## S3 method for class 'formula'
ternaryplot(x, data = NULL, response = NULL,  
groups = NULL, density = FALSE, region = density || !is.null(response),  
contour = density || !is.null(response), labels = !is.null(response),  
colorkey = region, xlab, ylab, zlab, xlim = c(-0.15, 1.15),  
ylim = c(-0.3, 1), panel = panel.ternaryplot,  
default.prepanel = lattice.getOption("prepanel.default.xyplot"),  
drop.unused.levels = lattice.getOption("drop.unused.levels"),  
subset = TRUE, ...)

## S3 method for class 'data.frame'
ternaryplot(x, data = NULL, ...)

## S3 method for class 'matrix'
ternaryplot(x, data = NULL, ...)

Arguments

x  
See Methods (by class).

data  
A data frame in which the formula, groups, and conditioning variables are evaluated.

...  
Arguments that are passed on to other methods, particularly panel.ternaryplot().

response  
An optional response variable

groups  
A variable or expression to be evaluated in data and used to distinguish groups by varying graphical parameters.

density  
Compute two-dimensional density estimates via MASS::kde2d().
ternaryplot

region Fill density or response estimates with a color gradient.
contour Draw contour lines for density and response estimates.
labels Label contour lines.
colorkey if TRUE automatically computes a colorkey for density or response estimates. Can also be a list (see lattice::levelplot() for details on this).
xlab X axis label (the left dimension)
ylab Y axis label (the right dimension)
zlab Z axis label (the top dimension)
xlim X limits for the plot region.
ylim Y limits for the plot region.
panel The panel function.
default.prepanel The default prepanel function.
drop.unused.levels Drop unused conditioning or groups levels.
subset An expression that evaluates to a logical or integer indexing vector. Like groups, it is evaluated in data. Only the resulting rows of data are used for the plot.

Value

An object of class "trellis". The update method can be used to update components of the object and the print method (usually called by default) will plot it on an appropriate plotting device.

Methods (by class)

- formula: A formula of the form top ~ left * right. Variables will be evaluated inside data if provided.
- data.frame: A data frame for which the first three columns will be mapped to the left, right, and top dimensions of the ternary plot respectively.
- matrix: A matrix for which the first three columns will be mapped to the left, right, and top dimensions of the ternary plot respectively.

Examples

ternaryplot(Fertility ~ Agriculture * Catholic, data = swiss)
ternaryplot(Catholic ~ Examination * Education, response = Infant.Mortality, data = swiss, contour = FALSE)
ternaryplot(Or ~ An * Ab | Feldspar, data = feldspar)
ternaryplot(Or ~ An * Ab, groups = Feldspar, data = feldspar, density = TRUE)
xyplot.acf  

Plot Autocovariance and Autocorrelation Functions

Description

This is a version of stats::plot.acf().

Usage

```r
## S3 method for class 'acf'
xyplot(x, data = NULL, ci = 0.95, ci_type = c("white", "ma"),
       ci_col = trellis.par.get("add.line")[col, ci_lty = 2, ...)
```

Arguments

- `x`  
  An 'acf' object.
- `data`  
  Ignored
- `ci`  
  Confidence level.
- `ci_type`  
  Type of confidence level.
- `ci_col`  
  Line color for the confidence levels.
- `ci_lty`  
  Line type for the confidence levels.
- `...`  
  Arguments passed on to lattice::xyplot().

Value

Returns and plots a trellis object.

Author(s)

Original by Brian Ripley.

See Also

lattice::xyplot(), stats::plot.acf(), stats::acf().

Examples

```r
z <- ts(matrix(rnorm(400), 100, 4), start = c(1961, 1), frequency = 12)
xyplot(acf(z))
```
xyplot.Arima

Diagnostic Plots for ARIMA Models

Description

Diagnostic plots modelled after stats::tsdiag() with some modifications and corrections of p-values in the Box–Ljung test.

Usage

```r
## S3 method for class 'Arima'
xyplot(x, data = NULL, which = 1:4, lag.max = NULL,
gof.lag = NULL, qq.aspect = "iso", na.action = na.pass, main = NULL,
layout = NULL, ...) 
```

Arguments

- `x` A fitted time-series model of class `Arima`.
- `data` Ignored
- `which` A sequence of integers between 1 and 4, identifying the plots to be shown.
- `lag.max` Number of lags to compute ACF for.
- `gof.lag` The maximum number of lags for the Ljung–Box test.
- `qq.aspect` Aspect of Q-Q plot (see lattice::qqmath()).
- `na.action` Treatment of NAs.
- `main` Optional titles for the plots. Can also be TRUE, in which case a default list of titles will be added.
- `layout` Either a numeric vector with (columns, rows) to use in the call to gridExtra::grid.arrange(), or a layout matrix which will then be passed as the layout_matrix in grid.arrange().
- `...` Parameters to pass to xyplot().

Value

Plots a lattice plot and returns a trellis object.

See Also

stats::tsdiag(), stats::arima(), lattice::xyplot(), gridExtra::grid.arrange(), stats::Box.test().

Examples

```r
fit <- arima(lh, order = c(1, 1, 0))
xyplot(fit, layout = c(2, 2))
xyplot(fit, which = c(1:2, 4), layout = rbind(c(1, 1), c(2, 3)))
```
xyplot.forecast

Plot Forecasts with Trellis Graphics

Description

Plot forecasts from \texttt{forecast::forecast()}. It is built mostly to resemble the \texttt{forecast::autoplot.forecast()} and \texttt{forecast::plot.forecast()} functions, but in addition tries to plot the predictions on the original scale.

Usage

```r
## S3 method for class 'forecast'
xyplot(x, data = NULL, ci = TRUE, ci_levels = x$level,
    ci_key = ci, ci_pal = hcl(0, 0, 45:100),
    ci_alpha = trellis.par.get("regions")$alpha, ...)
```

Arguments

- `x` An object of class \texttt{forecast}.
- `data` Data of observations left out of the model fit, usually "future" observations.
- `ci` Plot confidence intervals for the predictions.
- `ci_levels` The prediction levels to plot as a subset of those forecasted in `x`.
- `ci_key` Set to TRUE to draw a key automatically or provide a list (if length(ci_levels) > 5 should work with \texttt{lattice::draw.colorkey()} and otherwise with \texttt{lattice::draw.key()})
- `ci_pal` Color palette for the confidence bands.
- `ci_alpha` Fill alpha for the confidence interval.
- `...` Arguments passed on to \texttt{lattice::panel.xyplot()}. 

Details

This function requires the \texttt{zoo} package.

Value

An object of class "trellis". The \texttt{update} method can be used to update components of the object and the \texttt{print} method (usually called by default) will plot it on an appropriate plotting device.

See Also

\texttt{lattice::panel.xyplot()}, \texttt{forecast::forecast()}, \texttt{lattice::xyplot.ts()}. 

Examples

```r
if (require(forecast)) {
  train <- window(USAccDeaths, c(1973, 1), c(1977, 12))
  test <- window(USAccDeaths, c(1978, 1), c(1978, 12))
  fit <- arima(train, order = c(0, 1, 1),
               seasonal = list(order = c(0, 1, 1)))
  fcast1 <- forecast(fit, 12)
  xyplot(fcast1, test, grid = TRUE, auto.key = list(corner = c(0, 0.99)),
         ci_key = list(title = "PI Level"))

  # A fan plot
  fcast2 <- forecast(fit, 12, level = seq(0, 95, 10))
  xyplot(fcast2, test, ci_pal = heat.colors(100))
}
```

---

**Description**

Lattice plot diagnostics for `lm` objects, mostly mimicking the behavior of `stats::plot.lm()` but based on `lattice::xyplot()` instead.

**Usage**

```r
## S3 method for class 'lm'
xyplot(x, data = NULL, which = c(1:3, 5), main = FALSE,
       id.n = 3, labels.id = names(residuals(x)), cex.id = 0.75,
       cook.levels = c(0.5, 1), label.pos = c(4, 2), layout = NULL, ...)
```

**Arguments**

- `x`  
  `lm` object, typically result of `lm` or `glm`.
- `data`  
  Only provided for method consistency and is ignored.
- `which`  
  if a subset of the plots is required, specify a subset of the numbers 1:6
- `main`  
  if TRUE plots default titles. Can also be a list or character vector of length 6.
- `id.n`  
  number of points to be labelled in each plot, starting with the most extreme.
- `labels.id`  
  vector of labels, from which the labels for extreme points will be chosen. NULL uses observation numbers.
- `cex.id`  
  magnification of point labels.
- `cook.levels`  
  levels of Cook’s distance at which to draw contours.
- `label.pos`  
  positioning of labels, for the left half and right half of the graph respectively, for plots 1-3.
- `layout`  
  a numeric vector with [columns, rows] to use in the call to `gridExtra::grid.arrange()`, or a layout matrix which will then be passed as the layout_matrix in `grid.arrange()`.
- `...`  
  arguments to be passed to `lattice::xyplot()`.
Value

A list of trellis objects or a single trellis object.

Author(s)

Original by John Maindonald and Martin Maechler. Adaptation to lattice by Johan Larsson.

See Also

stats::lm(), stats::plot.lm(), lattice::xyplot()

Examples

fit <- lm(Sepal.Length ~ Sepal.Width, data = iris)
xyplot(fit)
xyplot(fit, which = 5)
Index

*Topic datasets
  feldspar, 6

approx, 5
bubbleplot, 2
bw.nrd, 5
bwplot2, 3
diag.panel.splom.density, 4
feldspar, 6
fft, 5
forecast::autoplot.forecast(), 22
forecast::forecast(), 22
forecast::plot.forecast(), 22

glm, 23
ggridExtra::grid.arrange(), 21, 23
lattice::bwplot(), 3, 4
lattice::diagnostics.splom(), 5
lattice::draw.colorkey(), 22
lattice::draw.key(), 3, 22
lattice::lattice.options(), 17
lattice::levelplot(), 19
lattice::panel.lines(), 5, 12, 13
lattice::panel.polygon(), 9, 12, 13
lattice::panel.qqmath(), 10
lattice::panel.qqmathline(), 9, 10, 16, 17
lattice::panel.superpose(), 9
lattice::panel.text(), 12, 13
lattice::panel.xyplot(), 7, 10, 12, 13, 15, 22
lattice::qqmath(), 9, 10, 16, 17, 21
lattice::splom(), 4, 5
lattice::xyplot(), 2, 3, 7, 8, 15, 17, 20, 21, 23, 24
lattice::xyplot.ts(), 22
latticeExtra::custom.theme(), 17
lm, 23
MASS::kde2d(), 10, 18
panel.bubbleplot(), 7
panel.bubbleplot(), 3
panel.ci, 7
panel.grid, 8
panel.qqmathci, 9
panel.qqmathci(), 16
panel.superpose, 8
panel.ternaryplot(), 10
panel.ternaryplot(), 18
panel.ternaryplot.clip, 11
panel.ternaryplot.clip(), 11
panel.ternaryplot.density, 12
panel.ternaryplot.density(), 11, 12
panel.ternaryplot.grid, 12
panel.ternaryplot.grid(), 11
panel.ternaryplot.response, 13
panel.ternaryplot.response(), 11
panel.ternaryplot.scales, 14
panel.ternaryplot.scales(), 11
panel.ternaryplot.xyplot, 15
panel.ternaryplot.xyplot(), 10, 11
points, 8
prepanel.ci, 15
print, 3, 4, 19, 22
qqmath.zoo, 16
quantile, 9
stats::acf(), 20
stats::arima(), 21
stats::Box.test(), 21
stats::density(), 5
stats::lm(), 24
stats::plot.acf(), 20
stats::plot.lm(), 23, 24
stats::tsdiag(), 21

25
tactile.theme, 17
ternaryplot, 18
trellis.par.get, 8

update, 3, 4, 19, 22

xyplot, 15
xyplot(), 21
xyplot.acf, 20
xyplot.Arima, 21
xyplot.forecast, 22
xyplot.lm, 23

zoo::zoo(), 17