Package 'rasterVis'

August 29, 2016

Type Package

Title Visualization Methods for Raster Data

Version 0.40

Date 2016-04-16

Encoding UTF-8

Description Methods for enhanced visualization and interaction with raster data. It implements visualization methods for quantitative data and categorical data, both for univariate and multivariate rasters. It also provides methods to display spatiotemporal rasters, and vector fields. See the website for examples.

URL http://oscarperpinan.github.io/rastervis

BugReports https://github.com/oscarperpinan/rastervis/issues

License GPL-3

LazyLoad yes

Depends R (>= 2.14.0), methods, raster (>= 2.0-12), lattice, latticeExtra

Imports stats, utils, parallel, grid, grDevices, RColorBrewer, hexbin, sp (>= 1.0-6), zoo, viridisLite

Suggests rgl, ggplot2, colorspace, dichromat

NeedsCompilation no

Author Oscar Perpinan Lamigueiro [cre, aut], Robert Hijmans [aut]

Maintainer Oscar Perpinan Lamigueiro <oscar.perpinan@gmail.com>

Repository CRAN

Date/Publication 2016-04-16 19:48:46

R topics documented:

`rasterVis-package` *Visualization methods for* `raster`

Description

The raster package defines classes and methods for spatial raster data access and manipulation. The rasterVis package complements raster providing a set of methods for enhanced visualization and interaction.

Details

Package:	rasterVis
Type:	Package
Version:	0.10
Date:	2011-06-17
License:	GPL-3
LazyLoad:	yes
Depends:	methods, raster, sp, grid, lattice, latticeExtra, zoo, hexbin, mgcv

Author(s)

Oscar Perpiñán Lamigueiro and Robert Hijmans

Maintainer: Oscar Perpiñán Lamigueiro <oscar.perpinan@upm.es>

bwplot-methods *Box and whisker plots of Raster objects.*

Description

Methods for bwplot and RasterStackBrick objects using a combination of panel.violin and panel.bwplot to compose the graphic.

Usage

```
## S4 method for signature 'RasterStackBrick,missing'
bwplot(x, data=NULL, layers, FUN,
            maxpixels = 1e+05,
            xlab='', ylab='', main='',
            violin=TRUE,
            par.settings=rasterTheme(),
            scales=list(x=list(rot=45, cex=0.8)),
            ...)
## S4 method for signature 'formula,Raster'
bwplot(x, data, dirXY,
            maxpixels = 1e+05,
            xscale.components=xscale.raster,
            yscale.components=yscale.raster,
            horizontal=FALSE,
            violin=TRUE,
            par.settings=rasterTheme(),
            ...)
```

Arguments

x	A RasterStackBrick object or a formula.
data	NULL or a Raster object.
layers	A numeric or character which should indicate the layers to be displayed.
dirXY	A direction as a function of the coordinates (see xyLayer).
FUN	A function to applied to the z slot of a RasterStackBrick object. The result of this function is used as the grouping variable of the plot.
maxpixels	A numeric, for sampleRandom.
xscale.components, yscale.components	Graphical parameters of lattice. See xyplot for details.
horizontal	Defaults to FALSE, meaning that the right hand of the formula is a factor or shingle.
xlab, ylab, main	Labels for axis and title.
violin	Logical, if TRUE the panel is built with panel.violin and panel.bwplot.

par.settings, scales
: See xyplot for details.

...
: Additional arguments for bwplot

Author(s)

Oscar Perpiñán Lamigueiro

See Also

bwplot, panel.violin, subset

Examples

```
f <- system.file("external/test.grd", package="raster")
r <- raster(f)
s <- stack(r, r-500, r+500)
bwplot(s)

## Not run:

stackSIS <- stack(listFich)
stackSIS <- stackSIS*24 ##from irradiance (W/m2) to irradiation Wh/m2
setwd(old)

idx <- seq(as.Date('2008-01-15'), as.Date('2008-12-15'), 'month')

SISmm <- setZ(stackSIS, idx)
names(SISmm) <- month.abb

bwplot(SISmm)
##FUN applies to z if not NULL
bwplot(SISmm, FUN=as.yearqtr)

## End(Not run)
## Not run:
##http://neo.sci.gsfc.nasa.gov/Search.html?group=64
pop <- raster('875430rgb-167772161.0.FLOAT.TIFF')
pop[pop==99999] <- NA
levelplot(pop, zscaleLog=10, par.settings=BTCTheme,
          panel=panel.levelplot.raster, interpolate=TRUE)

##http://neo.sci.gsfc.nasa.gov/Search.html?group=20
landClass <- raster('241243rgb-167772161.0.TIFF')
landClass[landClass==254] <- NA


s <- stack(pop, landClass)
names(s) <- c('pop', 'landClass')

bwplot(asinh(pop) ~ landClass|cut(y, 3), data=s,
       layout=c(3, 1), violin=FALSE)
```

```
bwplot(asinh(pop) ~ cut(y, 5)|landClass, data=s,
       scales=list(x=list(rot=45)), layout=c(4, 5),
       strip=strip.custom(strip.levels=TRUE))

## End(Not run)
```

---

densityplot-methods *Density plots for Raster objects.*

Description

Draw kernel density plots (with lattice) of Raster objects.

Usage

```
## S4 method for signature 'RasterLayer,missing'
densityplot(x, data=NULL, maxpixels = 1e+05,
            xlab='', ylab='', main='', col='black', ...)


## S4 method for signature 'RasterStackBrick,missing'
densityplot(x, data=NULL, layers, FUN,
            maxpixels = 1e+05,
            xlab='', ylab='', main='',
            par.settings=rasterTheme(),
            ...)

## S4 method for signature 'formula,Raster'
densityplot(x, data, dirXY,
            maxpixels = 1e+05,
            xscale.components=xscale.raster,
            yscale.components=yscale.raster,
            auto.key = list(space = 'right'),
            par.settings=rasterTheme(),...)
```

Arguments

x	A Raster* object or a formula.
data	NULL or a Raster object.
layers	A numeric or character which should indicate the layers to be displayed.
dirXY	A direction as a function of the coordinates (see xyLayer).
FUN	A function to applied to the z slot of a RasterStackBrick object. The result of this function is used as the grouping variable of the plot.
maxpixels	A numeric, for sampleRandom.

xlab, ylab, main, col, xscale.components, yscale.components, par.settings, auto.key
: Arguments for densityplot.

...
: Additional arguments for densityplot

Author(s)

Oscar Perpiñán Lamigueiro

See Also

densityplot,xscale.raster, yscale.raster, rasterTheme

Examples

```
f <- system.file("external/test.grd", package="raster")
r <- raster(f)
densityplot(r)
s <- stack(r, r+500, r-500)
densityplot(s)

## Not run:

old <- getwd()
##change to your folder...
setwd('CMSAF')
listFich <- dir(pattern='2008')
stackSIS <- stack(listFich)
stackSIS <- stackSIS*24 ##from irradiance (W/m2) to irradiation Wh/m2
setwd(old)

idx <- seq(as.Date('2008-01-15'), as.Date('2008-12-15'), 'month')

SISmm <- setZ(stackSIS, idx)
names(SISmm) <- month.abb

densityplot(SISmm)
##FUN applies to z if not NULL
densityplot(SISmm, FUN=as.yearqtr)

## End(Not run)
## Not run:
##http://neo.sci.gsfc.nasa.gov/Search.html?group=64
pop <- raster('875430rgb-167772161.0.FLOAT.TIFF')
pop[pop==99999] <- NA
levelplot(pop, zscaleLog=10, par.settings=BTCTheme,
          panel=panel.levelplot.raster, interpolate=TRUE)

##http://neo.sci.gsfc.nasa.gov/Search.html?group=20
landClass <- raster('241243rgb-167772161.0.TIFF')
landClass[landClass==254] <- NA
```

```
s <- stack(pop, landClass)
names(s) <- c('pop', 'landClass')

densityplot(~asinh(pop)|landClass, data=s,
            scales=list(relation='free'),
            strip=strip.custom(strip.levels=TRUE))

## End(Not run)
```

Formula methods *Formula methods*

Description

Formula methods

Usage

```
## S4 method for signature 'formula,Raster'
xyplot(x, data, dirXY, maxpixels=1e5,
            alpha=0.05,
            xscale.components=xscale.raster, yscale.components=yscale.raster,
            par.settings=rasterTheme(),...)
## S4 method for signature 'formula,Raster'
hexbinplot(x, data, dirXY,
            xscale.components=xscale.raster, yscale.components=yscale.raster,
            par.settings=rasterTheme(),...)
```

Arguments

x	A formula describing the variables to be related. It may include the layer names (which are internally converted to valid ones with make.names) and the x, y variables representing the coordinates of the Raster object. Besides, if dirXY is not missing, the variable dirXY can also be included in the formula.
data	A Raster object.
dirXY	A direction as a function of the coordinates (see xyLayer).
maxpixels	A numeric, for sampleRegular.
alpha	A numeric, transparency of the points.
xscale.components, yscale.components,par.settings	Customization of lattice. See xyplot for details.
...	Additional arguments for the xyplot and hexbinplot functions.

Author(s)

Oscar Perpiñán Lamigueiro

Examples

```
f <- system.file("external/test.grd", package="raster")
r <- raster(f)
names(r)

xyplot(test~y, data=r, alpha=0.5)

## Not run:
##Solar irradiation data from CMSAF
##Data available from http://www.box.net/shared/rl51y1t9sldxk54ogd44

old <- getwd()
##change to your folder...
setwd('CMSAF')
listFich <- dir(pattern='2008')
stackSIS <- stack(listFich)SISmm <- SISmm*24 ##from irradiance (W/m2) to irradiation Wh/m2
setwd(old)

names(SISmm) <- month.abb

##Relation between the January & February versus July radiation for four
##differents longitude regions.
xyplot(Jan+Feb~Jul|cut(x, 4), data=SISmm, auto.key=list(space='right'))
##Faster with hexbinplot
hexbinplot(Jan~Jul|cut(x, 6), data=SISmm)

## End(Not run)
```

---

gplot-methods *Use ggplot to plot a Raster* object*

Description

A wrapper function around ggplot (ggplot2 package). Note that the function in the raster package is called gplot with a single 'g'.

Usage

```
## S4 method for signature 'Raster'
gplot(x, maxpixels=50000,...)
```

Arguments

x	A Raster* object
maxpixels	Maximum number of pixels to use
...	Additional arguments for ggplot

Author(s)

Robert J. Hijmans and Oscar Perpiñán; based on an example by Paul Hiemstra

See Also

plot, spplot

Examples

```
## Not run:
r <- raster(system.file("external/test.grd", package="raster"))
s <- stack(r, r*2)
names(s) <- c('meuse', 'meuse x 2')

library(ggplot2)

theme_set(theme_bw())
gplot(s) + geom_tile(aes(fill = value)) +
          facet_wrap(~ variable) +
          scale_fill_gradient(low = 'white', high = 'blue') +
          coord_equal()

## End(Not run)
```

---

histogram-methods *Histogram of Raster objects.*

Description

Draw histograms (with lattice) of Raster objects.

Usage

```
## S4 method for signature 'RasterLayer,missing'
histogram(x, data=NULL, maxpixels = 1e+05, nint=100,
            xlab='', ylab='', main='', col='gray', ...)

## S4 method for signature 'RasterStackBrick,missing'
histogram(x, data=NULL, layers, FUN,
            maxpixels = 1e+05, nint=100,
            xlab='', ylab='', main='', col='gray',
            between=list(x=0.5, y=0.2),
            as.table=TRUE,
            scales=list(x=list(relation='free'),
              y=list(relation='free',
                draw=FALSE)),
            names.attr,
```

```
            par.settings=rasterTheme(),
            ...)

## S4 method for signature 'formula,Raster'
histogram(x, data, dirXY,
            maxpixels = 1e+05,
            strip=TRUE,
            par.settings=rasterTheme(),...)
```

Arguments

x	A Raster* object or a formula.
data	NULL or a Raster object.
layers	A numeric or character which should indicate the layers to be displayed.
dirXY	A direction as a function of the coordinates (see xyLayer).
FUN	A function to applied to the z slot of a RasterStackBrick object. The result of this function is used as the grouping variable of the plot.
nint	Number of breaks for the histogram. See the documentation of lattice::histogram at lattice for details.
maxpixels	A numeric, for sampleRandom.
xlab, ylab, main, col	Arguments for histogram.
names.attr	Character or expression, names to use in each panel. If missing its default value is the result of names(x) (after subsetting the layers to be displayed).
between, as.table, scales, strip, par.settings	Graphical parameters of lattice. See lattice::xyplot for details.
...	Additional arguments for lattice::histogram

Note

If you need different breakpoints in each panel, set breaks explicitely with NULL, a numeric or a character (for example, 'Sturges'; see hist for details)

Author(s)

Oscar Perpiñán Lamigueiro

See Also

histogram,xscale.raster, yscale.raster, rasterTheme

Examples

```
f <- system.file("external/test.grd", package="raster")
r <- raster(f)
histogram(r)
s <- stack(r, r+500, r-500)
## Same breakpoints across panels
histogram(s)
## Each panel with different breakpoints
histogram(s, breaks=NULL)
histogram(s, breaks='Sturges')
histogram(s, breaks=30)


## Not run:
##Solar irradiation data from CMSAF http://dx.doi.org/10.5676/EUM_SAF_CM/RAD_MVIRI/V001
old <- setwd(tempdir())
download.file('https://raw.github.com/oscarperpinan/spacetime-vis/master/data/SISmm2008_CMSAF.zip',
   'SISmm2008_CMSAF.zip', method='wget')
unzip('SISmm2008_CMSAF.zip')

listFich <- dir(pattern='\.nc')
stackSIS <- stack(listFich)
stackSIS <- stackSIS*24 ##from irradiance (W/m2) to irradiation Wh/m2

idx <- seq(as.Date('2008-01-15'), as.Date('2008-12-15'), 'month')

SISmm <- setZ(stackSIS, idx)
names(SISmm) <- month.abb

histogram(SISmm)
histogram(SISmm, FUN=as.yearqtr)

## With the formula interface you can create histograms for a set of variables
histogram(~ Jan + Dec, data=SISmm)
## Or use the coordinates for generating zonal histograms.
## For example, five histograms for each latitude zone
histogram(~Jan|cut(y, 5), data=SISmm)
## More sophisticated bands can be defined using the dirXY argument
histogram(~Jan|cut(dirXY, 5), dirXY = x^2 + y^2, data=SISmm)

setwd(old)

## End(Not run)

## Not run:
##http://neo.sci.gsfc.nasa.gov/Search.html?group=64
pop <- raster('875430rgb-167772161.0.FLOAT.TIFF')
pop[pop==99999] <- NA
levelplot(pop, zscaleLog=10, par.settings=BTCTheme,
          panel=panel.levelplot.raster, interpolate=TRUE)

##http://neo.sci.gsfc.nasa.gov/Search.html?group=20
```

```
landClass <- raster('241243rgb-167772161.0.TIFF')
landClass[landClass==254] <- NA


s <- stack(pop, landClass)
names(s) <- c('pop', 'landClass')

histogram(~asinh(pop)|landClass, data=s,
            scales=list(relation='free'),
            strip=strip.custom(strip.levels=TRUE))

## End(Not run)
```

---

horizonplot-methods *Horizon plots of Raster objects.*

Description

This method draws horizon graphs for each zone as calculated with zonal from the directions defined by xyLayer

Usage

```
## S4 method for signature 'RasterStackBrick,missing'
horizonplot(x, data = NULL,
    dirXY = y, stat = 'mean', digits = 0,
    origin = mean,
    xlab = 'Time', ylab = 'direction',
    colorkey = TRUE, colorkey.digits = 1,
    scales=list(y = list(relation = "same")),
    ...)
```

Arguments

x	A RasterStackBrick object.
data	Not used.
dirXY	A direction as a function of the coordinates (see xyLayer).
stat	a function to be applied to summarize the values by zone. See zonal for details.
digits	An integer, number of digits for zonal.
origin	From the latticeExtra help page: "the baseline y value for the first (positive) segment (i.e. the value at which red changes to blue)." It can be: a number, used acrros all panels, or a function (or a character defining a function), evaluated with the values in each panel. The default is the mean function.
xlab, ylab	Labels of the axis.

colorkey — If `colorkey = TRUE` a suitable color scale bar is constructed using the values of `origin` and `horizonscale` (see below). For additional information see `levelplot`.

colorkey.digits — Digits for rounding values in `colorkey` labels

scales — From the `lattice::xyplot` help page: "A list determining how the x- and y-axes (tick marks and labels) are drawn. The list contains parameters in `name=value` form, and may also contain two other lists called `x` and `y` of the same form. Components of `x` and `y` affect the respective axes only, while those in `scales` affect both. When parameters are specified in both lists, the values in `x` or `y` are used." In `horizonplot` the most interesting component is `relation`, a character string that determines how axis limits are calculated for each panel. Possible values are `"same"` (default), `"free"` and `"sliced"`. "For 'relation="same"', the same limits, usually large enough to encompass all the data, are used for all the panels. For 'relation="free"', limits for each panel is determined by just the points in that panel. Behavior for 'relation="sliced"' is similar, except that the length (max - min) of the scales are constrained to remain the same across panels."

... — Additional arguments for the `horizonplot` function. `horizonscale` is the most interesting, being (from the `latticeExtra` help page) "the scale of each color segment. There are 3 positive segments and 3 negative segments. If this is a given as a number then all panels will have comparable distances, though not necessarily the same actual values (similar in concept to 'scales$relation = "sliced"')". On the other hand, `col.regions` is used to choose the color scale.

Details

(Extracted from the reference): "The horizon graph allows to examine how a large number of items changed through time, to spot extraordinary behaviors and predominant patterns, view each of the items independently from the others when they wish, make comparisons between the items, and view changes that occurred with enough precision to determine if further examination is required."

References

http://www.perceptualedge.com/articles/visual_business_intelligence/time_on_the_horizon.pdf

See Also

`horizonplot`, `xyplot`, `levelplot`.

Examples

```
## Not run:
library(zoo)

url <- "ftp://ftp.wiley.com/public/sci_tech_med/spatio_temporal_data/"
sst.dat = read.table(paste(url, "SST011970_032003.dat", sep=''), header = FALSE)
sst.ll = read.table(paste(url, "SSTlonlat.dat", sep=''), header = FALSE)
```

```
spSST <- SpatialPointsDataFrame(sst.ll, sst.dat)
gridded(spSST) <- TRUE
proj4string(spSST) = "+proj=longlat +datum=WGS84"
SST <- brick(spSST)

idx <- seq(as.Date('1970-01-01'), as.Date('2003-03-01'), by='month')
idx <- as.yearmon(idx)
SST <- setZ(SST, idx)
names(SST) <- as.character(idx)

horizonplot(SST)

horizonplot(SST, stat='sd')

## Different scales for each panel, with colors representing deviations
## from the origin in *that* panel
horizonplot(SST, scales=list('free'))

## origin may be a function...
horizonplot(SST, origin=mean)
## ...or a number
horizonplot(SST, origin=0)

## A different color palette
horizonplot(SST, origin=0, col.regions=brewer.pal(n=6, 'PuOr'))

## The width of each color segment can be defined with horizonscale
horizonplot(SST, horizonscale=1, origin=0)

## End(Not run)

## Not run:


old <- getwd()
##change to your folder...
setwd('CMSAF')
listFich <- dir(pattern='2008')
stackSIS <- stack(listFich)
stackSIS <- stackSIS*24 ##from irradiance (W/m2) to irradiation Wh/m2
setwd(old)

idx <- seq(as.Date('2008-01-15'), as.Date('2008-12-15'), 'month')

SISmm <- setZ(stackSIS, idx)
names(SISmm) <- month.abb

horizonplot(SISmm)

## End(Not run)
```

hovmoller-methods *Hovmoller plots*

Description

Hovmoller plots of Raster objects.

Usage

```
## S4 method for signature 'RasterStackBrick'
hovmoller(object, dirXY=y, FUN=mean,
digits=2, xlab='Direction', ylab='Time',
par.settings=rasterTheme(), xscale.components=xscale.raster,
add.contour=FALSE, labels=FALSE, region=TRUE, ...)
```

Arguments

object	A RasterStackBrick with a non-empty z slot.
dirXY	A direction as a function of the coordinates (see xyLayer).
FUN	A function to be applied to the zones calculated with dirXY and zonal.
digits	An integer, number of digits for zonal.
xlab, ylab	Labels of the axis.
par.settings	Customization of lattice. See levelplot and rasterTheme for details.
xscale.components	See xscale.raster.
labels, region	Customization of contourplot when add.contour is TRUE.
add.contour	Logical, if TRUE a contourplot with filled regions is drawn.
...	Additional arguments for the contourplot and levelplot functions.

Details

Extracted from wikipedia: "A Hovmöller diagram is a commonly used way of plotting meteorological data to highlight the role of waves. The axes of a Hovmöller diagram are typically longitude or latitude (abscissa or x-axis) and time (ordinate or y-axis) with the value of some field represented through color or shading." The direction defined by dirXY and the function FUN allows for a variety of diagrams with this method.

Author(s)

Oscar Perpiñán Lamigueiro

References

- Hovmoller, E. 1949. The trough and ridge diagram. Tellus 1, 62–66.
- http://www2.mmm.ucar.edu/episodes/Hovmoller/noJS/hovm200707.htm
- http://www.esrl.noaa.gov/psd/map/clim/sst.shtml

See Also

levelplot, zonal, panel.2dsmoother

Examples

```
## Not run:
##Solar irradiation data from CMSAF http://dx.doi.org/10.5676/EUM_SAF_CM/RAD_MVIRI/V001
old <- setwd(tempdir())
download.file('https://raw.github.com/oscarperpinan/spacetime-vis/master/data/SISmm2008_CMSAF.zip',
   'SISmm2008_CMSAF.zip', method='wget')
unzip('SISmm2008_CMSAF.zip')

listFich <- dir(pattern='\\.nc')
stackSIS <- stack(listFich)
stackSIS <- stackSIS*24 ##from irradiance (W/m2) to irradiation Wh/m2

idx <- seq(as.Date('2008-01-15'), as.Date('2008-12-15'), 'month')

SISmm <- setZ(stackSIS, idx)
names(SISmm) <- month.abb

## Latitude as default
hovmoller(SISmm, xlab='Latitude')

## With contour lines and labels
hovmoller(SISmm, labels=TRUE, add.contour=TRUE,
          xlab='Latitude')

## Smooth color regions with latticeExtra::panel.2dsmoother
hovmoller(SISmm, panel=panel.2dsmoother, n=1000,
          labels=FALSE, add.contour=TRUE,
          xlab='Latitude')

## Using a function of coordinates
hovmoller(SISmm, dirXY=sqrt(x^2+y^2))

## End(Not run)

## Not run:
library(zoo)
library(rasterVis)

## DESCRIPTION: http://iridl.ldeo.columbia.edu/SOURCES/.CAC/.sst/
setwd(tempdir())
download.file('http://iridl.ldeo.columbia.edu/SOURCES/.CAC/.sst/data.nc', destfile = 'SST.nc')
SST <- stack('SST.nc')
idx <- seq(as.Date('1970-01-01'), as.Date('2003-03-01'), by='month')
tt <- as.yearmon(idx)
SST <- setZ(SST, tt)
names(SST) <- as.character(tt)

## Extract month value from a Date or yearmon object
```

```
month <- function(x)format(x, '%m')
## Compute anomaly using monthly grouping with ave
anomaly <- function(x){
    ## Monthly means
    mm <- ave(x, month(tt), FUN = mean)
    ## Monthly standard deviation
    msd <- ave(x, month(tt), FUN = sd)
    ## anomaly
    (x - mm)/msd
}
## Use anomaly with calc
SSTanom <- calc(SST, anomaly)
SSTanom <- setZ(SSTanom, tt)

## Ok, let's see the result
hovmoller(SSTanom,
          at = seq(-3, 3, .25),
          panel = panel.levelplot.raster,
          interpolate = TRUE,
          yscale.components = yscale.raster.subticks,
          par.settings = BuRdTheme)

## End(Not run)
```

`Interaction` *Interaction with trellis objects.*

Description

`chooseRegion` provides a set of points (in the form of a `SpatialPoints`) inside a region defined by several mouse clicks. `identifyRaster` labels and returns points of a `trellis` graphic according to mouse clicks.

Usage

```
chooseRegion(sp = TRUE, proj = as.character(NA))
## S4 method for signature 'Raster'
identifyRaster(object, layer=1, values=FALSE, pch=13, cex=0.6, col='black',...)
```

Arguments

`sp`	logical, if TRUE the result is a `SpatialPoints` object, otherwise it is a logical vector as returned by `in.out`
`proj`	A character string for the `proj4string` of `SpatialPoints`.
`object`	A Raster object.
`layer`	A numeric or character which should indicate the layer to be chosen.
`values`	logical, if TRUE the values are returned.

pch, cex, col	Graphical parameters for panel.identify and panel.link.splom.
...	Additional arguments for trellis.focus,panel.link.splom and panel.identify.

Details

When called, these functions wait for the user to identify points (in the panel being drawn) via mouse clicks. Clicks other than left-clicks close the region (for chooseRegion) and the procedure (for identifyRaster).

Note

chooseRegion needs the package mgcv to be installed.

Author(s)

Oscar Perpiñán Lamigueiro

See Also

panel.identify,panel.link.splom, trellis.focus, grid.locator

Examples

```
f <- system.file("external/test.grd", package="raster")
r <- raster(f)
levelplot(r)
##Do not close the last graphical window
##Use the left button of the mouse to identify points and the right button to finish
chosen_r <- identifyRaster(r, values=TRUE)
chosen_r
s <- stack(r, r-500, r+500)
levelplot(s)
chosen_s <- identifyRaster(s, values=TRUE)
chosen_s

## Not run:
##The package mgcv is needed for the next example
##Use the left button of the mouse to build a border with points, and the right button to finish.
##The points enclosed by the border will be highlighted and returned as a SpatialPoints object.
levelplot(s)
reg <- chooseRegion()
summary(reg)

## End(Not run)

## Not run:
##Solar irradiation data from CMSAF
##Data available from http://www.box.net/shared/rl51y1t9sldxk54ogd44

old <- getwd()
##change to your folder...
setwd('CMSAF')
```

```
listFich <- dir(pattern='2008')
stackSIS <- stack(listFich)
stackSIS <- stackSIS*24 ##from irradiance (W/m2) to irradiation Wh/m2
setwd(old)

idx <- seq(as.Date('2008-01-15'), as.Date('2008-12-15'), 'month')

SISmm <- setZ(stackSIS, idx)
names(SISmm) <- month.abb

levelplot(SISmm)

##Do not close the last graphical window
##Interaction
##Use the left button of the mouse to identify points and the right button to finish
chosen <- identifyRaster(SISmm, layer=3, values=TRUE)
chosen
##Use the left button of the mouse to build a border with points, and the right button to finish.
##The points enclosed by the border will be highlighted and returned as a SpatialPoints object.
reg <- chooseRegion()
summary(reg)

## End(Not run)
```

`levelplot-methods` *Level and contour plots of Raster objects.*

Description

Level and contour plots of Raster objects with `lattice` methods and marginal plots with `grid` objects.

Usage

```
## S4 method for signature 'Raster,missing'
levelplot(x, data = NULL, layers,
            margin = list(),
            maxpixels = 1e5,
            par.settings = rasterTheme(),
            between = list(x=0.5, y=0.2),
            as.table = TRUE,
            xlab=if(isLonLat(x)) 'Longitude' else NULL,
            ylab=if(isLonLat(x)) 'Latitude' else NULL,
            main=NULL,
            names.attr,
            scales  =list(),
            xscale.components = xscale.raster,
            yscale.components = yscale.raster,
            zscaleLog = NULL,
```

```
            colorkey = list(space='right'),
            panel = panel.levelplot,
            pretty = FALSE,
            contour = FALSE, region = TRUE, labels = FALSE,
            FUN.margin = NULL,
            scales.margin = NULL, axis.margin = NULL,
            ..., att=1L)

## S4 method for signature 'Raster,missing'
contourplot(x, data=NULL, layers, ...)
```

Arguments

x	A Raster object.
data	Not used.
layers	A numeric or character which should indicate the layers to be displayed.
maxpixels	A positive integer giving the number of cells to display, for sampleRegular.
margin	A list or a logical. If it is TRUE, two marginal graphics show the column (x) and row (y) summaries of the Raster* object. The summary is computed with the function mean. If it is a list, it contains parameters in 'name=value' form that define both margins, and may contain two other lists called 'x' and 'y' whose components affect the respective margins only: • draw: A logical. If TRUE (default) the marginal graphics are drawn. • FUN: A function to summarise the Raster* by rows and columns (default: mean). • scales: A list with components x (columns) and y (rows). Each of these components must be a numeric vector of length 2 defining the range for each marginal plot. If scales = NULL (default) the range is internally computed. If any of the elements of the vectors is NA, the corresponding limit of the range will be calculated internally. If any of the vectors is of length 1, it is assumed that it defines the lower limit of the range, and the upper limit is calculated internally. • axis: Logical or a list. Its default value is FALSE. If it is TRUE or a list, a simple axis is drawn with the marginal graphic. If it is a list, its components define the graphical parameters of the axis using grid::gpar. The default value is gpar(col = 'darkgrey', fontsize = 7)
FUN.margin, scales.margin, axis.margin	Deprecated arguments. Use margin as a list instead
att	Integer or character to choose which variable (column) in the RAT table should be used.
xlab, ylab, main	A character string or expression describing the axis and title labels. These arguments are used by the underlying lattice::xyplot function, which provides this information in its help page: "main, xlab and ylab are usually a character string or an expression that gets used as the label, but can also be a list that controls further details. Expressions

are treated as specification of LaTeX-like markup as described in plotmath. The label can be a vector, in which case the components will be spaced out horizontally (or vertically for ylab). This feature can be used to provide column or row labels rather than a single axis label.

When main (etc.) is a list, the actual label should be specified as the xlab component (which may be unnamed if it is the first component). The label can be missing, in which case the default will be used. Further named arguments are passed on to textGrob; this can include arguments controlling positioning like just and rot as well as graphical parameters such as col and font (see gpar for a full list).

main, xlab and ylab can also be arbitrary '"grob"'s (grid graphical objects)."

names.attr Character or expression, names to use in each panel. If missing its default value is the result of names(x) (after subsetting the layers to be displayed).

xscale.components, yscale.components

See xscale.raster and yscale.raster.

between, as.table, par.settings, scales, panel

Graphical parameters used by lattice::xyplot. Adapted from the help page of this function:

- between: A list with components code and code (both usually 0 by default), numeric vectors specifying the space between the panels (units are character heights). x and y are repeated to account for all panels in a page and any extra components are ignored.
- as.table: A logical flag that controls the order in which panels should be displayed: if TRUE (the default), left to right, top to bottom (as in a table). If FALSE panels are drawn left to right, bottom to top.
- par.settings: A list to choose some display settings temporarily. This list is supplied to trellis.par.set. When the resulting object is plotted, these options are applied temporarily for the duration of the plotting, after which the settings revert back to what they were before. This enables the user to attach some display settings to the trellis object itself rather than change the settings globally.
  rasterVis includes some functions with predefined themes that can be directly supplied to par.settings: rasterTheme (default), RdBuTheme and BuRdTheme, GrTheme, BTCTheme, PuOrTheme and streamTheme (for streamplot). These themes are defined using custom.theme. You can use rasterTheme or custom.theme to define your own theme (see examples for details).
- scales: A list determining how the x- and y-axes (tick marks and labels) are drawn. The list contains parameters in name=value form, and may also contain two other lists called x and y of the same form. Components of x and y affect the respective axes only, while those in scales affect both. When parameters are specified in both lists, the values in x or y are used. For example, use scales=list(draw=FALSE) to suppress ticks and labels in both axis. Read the help page of lattice::xyplot to know about the possible components of scales.
- panel: A function object or a character string giving the name of a predefined function. The default is panel.levelplot. Another useful option is

panel.levelplot.raster.

colorkey, pretty, contour, region, labels

Graphical parameters supplied to lattice::levelplot. Adapted from its help page:

- colorkey: logical specifying whether a color key is to be drawn alongside the plot (default is TRUE), or a list describing the color key.
- pretty: A logical flag, indicating whether to use pretty cut locations and labels. It is FALSE for levelplot and TRUE for contourplot.
- contour: A logical flag, indicating whether to draw contour lines. It is TRUE for contourplot and FALSE for levelplot.
- region: A logical flag, indicating whether regions between contour lines should be filled as in a level plot. It is FALSE for contourplot and TRUE for levelplot.
- labels: Tipically a logical indicating whether the labels are to be drawn (default is TRUE for contourplot), a character or expression vector giving the labels associated with the at values, or a list whose components define the labels and their graphical parameters. Read the help page of panel.levelplot for details.

zscaleLog

Controls whether the Raster* will be log transformed before being passed to the panel function. Defaults to NULL, in which case the Raster* is not transformed. Other possible values are any number that works as a base for taking logarithm, TRUE (which is equivalent to 10), "e" (for the natural logarithm), and FALSE (that is equivalent to NULL). As a side effect, the colorkey is labeled differently.

...

Additional arguments for lattice::levelplot, lattice::xyplot, panel.levelplot and panel.levelplot.raster. The most important ones are:

- layout: From the help page of lattice::xyplot: layout is a numeric vector of length 2 or 3 giving the number of columns, rows, and pages (optional) in a multipanel display. The number of pages is by default set to as many as is required to plot all the panels, and so rarely needs to be specified.
  For example, with layout=c(1, 1) each panel (corresponding to a layer of a RasterStackBrick) object will be printed in a separate page (which could be useful to generate a series of output files to build an animation.)
- xlim, ylim: From the help page of lattice::xyplot: A numeric vector of length 2 giving left and right limits for x-axis, and lower and upper limits for the y-axis.
- shrink: From the help page of panel.levelplot: Either a numeric vector of length 2 (meant to work as both x and y components), or a list with components x and y which are numeric vectors of length 2. This allows the rectangles to be scaled proportional to the z-value. The specification can be made separately for widths (x) and heights (y). The elements of the length 2 numeric vector gives the minimum and maximum proportion of shrinkage (corresponding to min and max of z).
- border, border.lty, border.lwd: Graphical parameters (color, type of line, width of line, respectively) of each rectangle borders. See the help page of panel.levelplot for details.

Details

The result of the levelplot method is similar to the spplot method for Raster objects defined in the raster package. However, this method does not use the spplot of the sp package and, therefore, no conversion between classes is needed.

The contourplot method is a wrapper for levelplot with the next additional default settings: cuts=7, labels=TRUE, contour=TRUE, pretty=TRUE, region=TRUE and colorkey=TRUE (see the help of contourplot for details.)

levelplot displays categorical data with a convenient legend. You should use ratify to define a layer as a categorical variable. It is able to display multilayer categorical rasters **only if** they share the same RAT (Raster Attribute Table). levelplot is not able to display multilayer rasters with factor **and** numeric layers. See ratify and the examples below for details.

Author(s)

Oscar Perpiñán Lamigueiro

See Also

spplot, lattice::levelplot

Examples

```
f <- system.file("external/test.grd", package="raster")
r <- raster(f)
levelplot(r)

## Change the color theme
levelplot(r, par.settings=GrTheme())
levelplot(r, par.settings=PuOrTheme())

myTheme=rasterTheme(region=brewer.pal('Blues', n=9))
levelplot(r, par.settings=myTheme)

## Define the legend breaks
my.at <- seq(100, 1850, 500)
levelplot(r, at=my.at)

myColorkey <- list(at=my.at, ## where the colors change
                   labels=list(
                     at=my.at ## where to print labels
                     ))
levelplot(r, at=my.at, colorkey=myColorkey)

myColorkey <- list(at=my.at, ## where the colors change
                   labels=list(
                     labels=letters[seq_along(my.at)], ## labels
                     at=my.at ## where to print labels
                     ))
levelplot(r, at=my.at, colorkey=myColorkey)
```

```
## shrink and border color
rCenter <- (maxValue(r) + minValue(r)) / 2
levelplot(r - rCenter, par.settings=RdBuTheme(), shrink=c(.8, 15), border='black')

## With subticks
levelplot(r, xscale.components=xscale.raster.subticks,
             yscale.components=yscale.raster.subticks)

levelplot(r, xscale.components=xscale.raster.subticks,
             yscale.components=yscale.raster.subticks,
             scales=list(x=list(rot=30, cex=0.8)))

## log-scale
levelplot(r^2, zscaleLog=TRUE, contour=TRUE)

## Customizing axis and title labels
levelplot(r, margin=FALSE,
          main=list('My plot', col='red'),
          xlab=c('This is the', 'X-Axis'),
          ylab=list('Y-Axis', rot=30, fontface='bold')
          )

## xlim and ylim to display a smaller region
levelplot(r, xlim=c(179000, 181000), ylim=c(329500, 334000))

## RasterStacks
s <- stack(r, r+500, r-500)
levelplot(s, contour=TRUE)
contourplot(s, labels=list(cex=0.4), cuts=12)

## Use of layout
levelplot(s, layout=c(1, 3))
levelplot(s, layout=c(1, 1))

## names.attr to change the labels of each panel
levelplot(s, names.attr=c('R', 'R + 500', 'R - 500'))

## Defining the scales for the marginal plot
levelplot(r, margin = list(axis = TRUE,
                           scales = list(x=c(100, 600),
                                         y=c(100, 1000))))
## if a component of the list is null, it is internally calculated
levelplot(r, margin=list(axis = TRUE, scales = list(x=c(100, 1000))))

## Add a layer of sampling points
## and change the theme
pts <- sampleRandom(r, size=20, sp=TRUE)

## Using +.trellis and layer from latticeExtra
levelplot(r, par.settings = BTCTheme) + layer(sp.points(pts, col = 'red'))
contourplot(r, labels = FALSE) + layer(sp.points(pts, col = 'red'))

## or with a custom panel function
```

```
levelplot(r, par.settings=BTCTheme,
          panel=function(...){
            panel.levelplot(...)
            sp.points(pts, col='red')
            })


## Categorical data
r <- raster(nrow=10, ncol=10)
r[] = 1
r[51:100] = 3
r[3:6, 1:5] = 5
r <- ratify(r)

rat <- levels(r)[[1]]
rat$landcover <- c('Pine', 'Oak', 'Meadow')
rat$class <- c('A1', 'B2', 'C3')
levels(r) <- rat
r

levelplot(r, col.regions=c('palegreen', 'midnightblue', 'indianred1'))

## with 'att' you can choose another variable from the RAT
levelplot(r, att=2, col.regions=c('palegreen', 'midnightblue', 'indianred1'))
levelplot(r, att='class', col.regions=c('palegreen', 'midnightblue', 'indianred1'))

r2 <- raster(r)
r2[] = 3
r2[51:100] = 1
r2[3:6, 1:5] = 5

r3 <- init(r, function(n)sample(c(1, 3, 5), n, replace=TRUE))

## Multilayer categorical Raster* are displayed only if their RATs are the same
levels(r2) <- levels(r3) <- levels(r)

s <- stack(r, r2, r3)
names(s) <- c('A', 'B', 'C')

levelplot(s)
levelplot(s, att=2)

## Not run:
##Solar irradiation data from CMSAF http://dx.doi.org/10.5676/EUM_SAF_CM/RAD_MVIRI/V001
old <- setwd(tempdir())
download.file('https://raw.github.com/oscarperpinan/spacetime-vis/master/data/SISmm2008_CMSAF.zip',
   'SISmm2008_CMSAF.zip', method='wget')
unzip('SISmm2008_CMSAF.zip')

listFich <- dir(pattern='\.nc')
stackSIS <- stack(listFich)
stackSIS <- stackSIS*24 ##from irradiance (W/m2) to irradiation Wh/m2
```

```
idx <- seq(as.Date('2008-01-15'), as.Date('2008-12-15'), 'month')

SISmm <- setZ(stackSIS, idx)
names(SISmm) <- month.abb

levelplot(SISmm)

levelplot(SISmm, layers=1, margin = list(FUN = 'median'), contour=TRUE)

setwd(old)

## End(Not run)
```

---

plot3D *Interactive 3D plot of a RasterLayer*

Description

Make an interactive 3D plot (map) of a RasterLayer. This is a wrapper around `surface3d` in the rgl package. You can use `decorate3d` to add axes.

Usage

```
## S4 method for signature 'RasterLayer'
plot3D(x, maxpixels=1e5,
   zfac=1, drape=NULL, col=terrain.colors,
   at=100, rev=FALSE,
   useLegend=TRUE, adjust=TRUE, ...)
```

Arguments

`x`	a RasterLayer object
`maxpixels`	Maximum number of pixels to use
`zfac`	Numeric, to set the elevation scale relative to x and y
`drape`	RasterLayer, to 'drape' colors representing the values of this layer on the 3D representation of layer `x`. In this case`x` typically has elevation data
`col`	A color palette generating function such as `rainbow`, `heat.colors`, and `topo.colors`, or one or your own making
`at`	A numeric variable of breakpoints defining intervals along the range of `x` or a number definining the number of intervals the range of `x` will be divided into.
`rev`	Logical. If `TRUE`, the color palette values are reversed in order
`useLegend`	Logical. If `TRUE` (default) the content of the slot `x@legend@colortable` is used instead of `col` and `at`.
`adjust`	Logical. If `TRUE`, the x and y axes are scaled relative to the cell (z) values
`...`	Any argument that can be passed to `surface3d`

Note

Previous versions opened a new device with each call of `plot3D`. This behaviour has been modified, and now a new device is opened only if none is active. Thus, you should use `rgl::open3d()` if you need to display a new scene without modifying the current one.

Author(s)

Robert J. Hijmans and Oscar Perpiñán

Examples

```
 ## Not run:
## rgl is needed to use plot3D
library(rgl)

data(volcano)
r <- raster(volcano)
extent(r) <- c(0, 610, 0, 870)

## level plot as reference
levelplot(r, col.regions=terrain.colors)

plot3D(r)
## Use different colors with a predefined function
plot3D(r, col=rainbow)
## or with a custom function using colorRampPalette
myPal <- colorRampPalette(brewer.pal(11, 'PuOr'))
plot3D(r, col=myPal)

## With at you can define an homogeneus color table for different Rasters

r2 <- r + 100
r3 <- r + 200
s <- stack(r, r2, r3)

maxVal <- max(maxValue(s))
minVal <- min(minValue(s))
N <- 40
breaks <- seq(minVal, maxVal, length=N)

plot3D(r, at=breaks)
plot3D(r2, at=breaks)
plot3D(r3, at=breaks)

## Default: x-axis and y-axis are adjusted with z-values. Therefore,
## labels with decorate3d() are useless
plot3D(r, adjust=TRUE)
decorate3d()
## Compare the graphic limits
par3d('bbox')
## with the extent of the Raster
extent(r)
```

```
## Set adjust=FALSE to fix it
plot3D(r, adjust=FALSE)
decorate3d()
## Once again, compare the graphic limits
par3d('bbox')
## with the extent of the Raster
extent(r)

## zfac controls the z values so z-axis will be distorted
plot3D(r, adjust=FALSE, zfac=2)
decorate3d()
par3d('bbox')


## With drape you can disconnect the z-axis from the colors
drape <- cut(r^4, 4)
plot3D(r, drape=drape)
## Compare with:
plot3D(r, at=4)

## End(Not run)
```

---

rasterTheme *Themes for* raster *with* lattice.

Description

Auxiliary functions for the customization of trellis graphics with lattice.

xscale.raster and yscale.raster suppress the right and top axis, respectively. xscale.raster.subticks and yscale.raster.subticks also suppress those axis and draw subticks.

rasterTheme is a customization of the custom.theme.2 function of latticeExtra using the magma palette of the viridis package. The other palettes provided by this package are available through the viridisTheme, infernoTheme, and plasmaTheme.

YlOrRdTheme, BuRdTheme, RdBuTheme, GrTheme and BTCTheme are variations of rasterTheme using palettes of the RColorBrewer and hexbin packages.

streamTheme is a variation of rasterTheme using black for the region, gray for the panel background and a sequential palette for points.

Usage

```
yscale.raster(lim, ...)
xscale.raster(lim, ...)
yscale.raster.subticks(lim, ...)
xscale.raster.subticks(lim, ...)

rasterTheme(region = magma(10),
```

```
            pch=19, cex=0.7,
            strip.background = list(col = 'transparent'),
            strip.shingle = list(col = 'transparent'),
            strip.border = list(col = 'transparent'),
            add.lines = list(lwd = .4),
            ...)

magmaTheme(region = magma(10), ...)
infernoTheme(region = inferno(10), ...)
plasmaTheme(region = plasma(10), ...)
viridisTheme(region = viridis(10), ...)

YlOrRdTheme(region = brewer.pal(9, 'YlOrRd'), ...)
RdBuTheme(region = brewer.pal(9, 'RdBu'), ...)
BuRdTheme(region = rev(brewer.pal(9, 'RdBu')), ...)
PuOrTheme(region = brewer.pal(9, 'PuOr'), ...)
GrTheme(region = rev(brewer.pal(9, 'Greys')), ...)
BTCTheme(region = BTC(n=9), ...)

streamTheme(region = 'black',
            symbol = brewer.pal(n=5, name='Blues'),
            alpha = 0.6,
            panel.background = list(col='gray20'),
            ...)
```

Arguments

`lim`	Range of data.
`pch`	Symbol used for points.
`cex`	A numeric multiplier to control the size of the points.
`region`	A vector of colors that is used to define a continuous color gradient using `colorRampPalette` to fill in regions. Note that the length of this gradient is set by `custom.theme` to exactly 100 colors.
`symbol`	A palette to display symbols.
`panel.background`	Parameters of the panel background.
`alpha, ...`	Additional arguments for `custom.theme.2`, `yscale.components.default` and `xscale.components.default`
`strip.background, strip.shingle, strip.border`	List whose components define the configuration of the strip areas. Read the help page of `trellis.par.get` for details.
`add.lines`	List whose components define the lines superposed on the graphic. Read the help page of `trellis.par.get` for details.

Author(s)

Oscar Perpiñán Lamigueiro

See Also

custom.theme, custom.theme.2, BTC, xscale.components.default, xscale.components.subticks

---

splom-methods *Scatter plot matrices of Raster objects.*

Description

Draw conditional scatter plot matrices with hexagonally binning.

Usage

```
## S4 method for signature 'RasterStackBrick,missing'
splom(x, data=NULL,maxpixels=1e5, plot.loess=FALSE, colramp=BTC, varname.cex=0.6,...)
```

Arguments

x	A RasterStackBrick object.
data	Not used.
maxpixels	A numeric, for sampleRandom.
plot.loess	Logical, should a loess fit be drawn?.
colramp	A function accepting an integer n as argument and returning n colors (for hexbinplot).
varname.cex	A numerical multiplier to control the size of the variables names.
...	Additional arguments for splom.

Note

While the hexagonal binning is quite fast for large datasets, the use of the loess fit will slow this function.

Author(s)

Oscar Perpiñán Lamigueiro

See Also

hexbinplot, splom

Examples

```
## Not run:
##Solar irradiation data from CMSAF
##Data available from http://www.box.net/shared/rl51y1t9sldxk54ogd44

old <- getwd()
##change to your folder...
setwd('CMSAF')
listFich <- dir(pattern='2008')
stackSIS <- stack(listFich)
stackSIS <- stackSIS*24 ##from irradiance (W/m2) to irradiation Wh/m2
setwd(old)

idx <- seq(as.Date('2008-01-15'), as.Date('2008-12-15'), 'month')

SISmm <- setZ(stackSIS, idx)
names(SISmm) <- month.abb

splom(SISmm)

## End(Not run)
```

---

vectorplot-methods *Vector plots of Raster objects.*

Description

`vectorplot` displays vector fields from Raster objects using arrows.

`streamplot` displays streamlines with a procedure inspired by the FROLIC algorithm (see references): for each point (*droplet*) of a jittered regular grid, a short streamline portion (*streamlet*) is calculated by integrating the underlying vector field at that point. The main color of each streamlet indicates local vector magnitude (*slope*). Streamlets are composed of points whose sizes, positions and color degradation encode the local vector direction (*aspect*).

Usage

```
## S4 method for signature 'Raster'
vectorplot(object, layers,
            narrows=2e3, lwd.arrows=0.6, col.arrows='black',
            length=unit(5e-2, 'npc'),
            maxpixels=1e5, region=TRUE, margin=FALSE,
            isField=FALSE, reverse=FALSE,
            unit='radians', scaleSlope=TRUE,
            aspX=0.08, aspY=aspX,
            key.arrow = NULL,
            ...)

## S4 method for signature 'RasterStack'
```

```
vectorplot(object, layers,
            narrows=2e3, lwd.arrows=0.6, col.arrows='black',
            length=unit(5e-2, 'npc'),
            maxpixels=1e5, region=TRUE, margin=FALSE,
            isField=FALSE, reverse=FALSE,
            unit='radians', scaleSlope=TRUE,
            aspX=0.08, aspY=aspX,
            key.arrow = NULL,
            uLayers, vLayers, ...)

## S4 method for signature 'Raster'
streamplot(object, layers,
            droplet = list(), streamlet = list(),
            par.settings=streamTheme(),
            isField = FALSE, reverse=FALSE,
            parallel=TRUE, mc.cores=detectCores(), cl=NULL,
            ...)

## S4 method for signature 'RasterStack'
streamplot(object, layers,
            droplet = list(), streamlet = list(),
            par.settings=streamTheme(),
            isField = FALSE, reverse=FALSE,
            parallel=TRUE, mc.cores=detectCores(), cl=NULL,
            ...)
```

Arguments

object	A Raster object. If isField=FALSE the vector field is calculated internally from the result of terrain.
layers	A numeric or character which should indicate the layers to be displayed.
maxpixels	A numeric, number of cells to be shown if region=TRUE or if region is a Raster* object.
narrows	A numeric, number of arrows.
lwd.arrows	Numeric, width of the lines of the arrows
col.arrows	character, color of the arrows
length	Unit, extent of the arrow head.
margin	Logical, if TRUE two marginal graphics show the summaries of the object.
scaleSlope	Logical or numeric. If TRUE the slope (vector magnitude) is scaled (but not centered) with its standard deviation. If it is a numeric, the slope is scaled with this value. It is not used if isField='dXY'.
aspX, aspY	Numeric. Multipliers to convert the slope values into horizontal (aspX) and vertical (aspY) displacements.
key.arrow	A reference (or legend) vector. If is not NULL, it is a list with two named components, size, a numeric to define the length of the arrow (default 1), and label, a character to define the label (default '').

`uLayers, vLayers`
: Numeric, indexes of layers with horizontal and vertical components, respectively, when `isField='dXY'` and the `RasterStack` has more than 2 layers. If missing, the horizontal components are the layers of the first half of the object, and the vertical components are the layers of the second half.

`droplet`
: A list whose elements define the droplet configuration:
  - cropExtent: Percentage of the object extent to be cropped (default .97) to avoid droplets at boundaries
  - pc: A numeric. It is the percentage of cells used to compute droplets. Its default value is 0.5. Therefore, only the 0.5% of the cells are used. For example, if you use a Raster with 180 rows and 360 columns (64800 cells), with this default value `streamplot` will produce 324 droplets.

`streamlet`
: A list whose elements define the streamlet configuration:
  - L: length of the streamlet (number of points, default 10)
  - h: streamlet calculation step (default `mean(res(object))`).

`par.settings`
: A list to define the graphical parameters. For `streamplot` there is an specific theme, `streamTheme`.

`parallel`
: Logical, TRUE (default) to use `parallel` package.

`cl`
: a cluster object. Read the help page of `parLapply` for details.

`mc.cores`
: The number of cores to use if `parallel=TRUE` and no `cl` is provided. Read the help page of `mclapply` for details.

`region`
: Logical, if `TRUE` the region is displayed as the background using `levelplot`. It can be a `Raster*` with the same extent and resolution as `object`.

`reverse`
: Logical, TRUE if arrows or streamlets go against the direction of the gradient.

`isField`
: If `TRUE` the object is a vector field. Thus, `object` must be a Raster* with two layers, slope (local vector magnitude) and aspect (local vector direction), in this order, following the philosophy of `terrain`. The slope layer will be used as the background if `region` is `TRUE`.

  If `isField='dXY'` `object` must be a Raster* with two layers representing the horizontal and the vertical components, respectively. The slope is computed and used as the background if `region` is TRUE.

`unit`
: Character, angle units of the `aspect` layer if `isField=TRUE`: 'radians' or 'degrees'.

`...`
: Additional arguments for `levelplot`

Author(s)

Oscar Perpiñán Lamigueiro

References

R. Wegenkittl and E. Gröller, Fast Oriented Line Integral Convolution for Vector Field Visualization via the Internet, Proceedings IEEE Visualization '97, 1997, http://www.cg.tuwien.ac.at/research/vis-dyn-syst/frolic/frolic_crc.pdf

See Also

panel.arrows, levelplot, terrain, mclapply, parLapply

Examples

```
## Not run:
proj <- CRS('+proj=longlat +datum=WGS84')

df <- expand.grid(x=seq(-2, 2, .01), y=seq(-2, 2, .01))
df$z <- with(df, (3*x^2 + y)*exp(-x^2-y^2))
r1 <- rasterFromXYZ(df, crs=proj)
df$z <- with(df, x*exp(-x^2-y^2))
r2 <- rasterFromXYZ(df, crs=proj)
df$z <- with(df, y*exp(-x^2-y^2))
r3 <- rasterFromXYZ(df, crs=proj)
s <- stack(r1, r2, r3)
names(s) <- c('R1', 'R2', 'R3')

vectorplot(r1)
vectorplot(r2, par.settings=RdBuTheme())
vectorplot(r3, par.settings=PuOrTheme())

## scaleSlope, aspX and aspY
vectorplot(r1, scaleSlope=FALSE)
vectorplot(r1, scaleSlope=1e-5)
vectorplot(r1, scaleSlope=5e-6, alpha=0.6)
vectorplot(r1, scaleSlope=TRUE, aspX=0.1, alpha=0.6)
vectorplot(r1, scaleSlope=TRUE, aspX=0.3, alpha=0.3)


## Reference vector
# Default size (1)
vectorplot(r1, region = FALSE,
           key.arrow = list(label = 'm/s'))
vectorplot(r1, region = FALSE,
           key.arrow = list(size = 2, label = 'm/s'))

## A vector field defined with horizontal and vertical components
u <- v <- raster(xmn=0, xmx=2, ymn=0, ymx=2, ncol=1e3, nrow=1e3)
x <- init(u, v='x')
y <- init(u, v='y')
u <- y * cos(x)
v <- y * sin(x)
field <- stack(u, v)
names(field) <- c('u', 'v')

vectorplot(field, isField='dXY', narrows=5e2)

## We can display both components as the background
vectorplot(field, isField='dXY', narrows=5e2, region=field)

## It is also possible to use a RasterStack
```

```
## with more than 2 layers when isField='dXY'
u1 <- cos(y) * cos(x)
v1 <- cos(y) * sin(x)
u2 <- sin(y) * sin(x)
v2 <- sin(y) * cos(x)
field <- stack(u, u1, u2, v, v1, v2)
names(field) <- c('u', 'u1', 'u2', 'v', 'v1', 'v2')

vectorplot(field, isField='dXY',
           narrows=300, lwd.arrows=.4,
           par.settings=BTCTheme(),
           layout=c(3, 1))

## uLayer and vLayer define which layers contain
## horizontal and vertical components, respectively
vectorplot(field, isField='dXY',
           narrows=300,
           uLayer=1:3,
           vLayer=6:4)

######################################################################
## Streamplot
######################################################################
## If no cluster is provided, streamplot uses parallel::mclapply except
## with Windows. Therefore, next code could spend a long time under
## Windows.
streamplot(r1)

## With a cluster
hosts <- rep('localhost', 4)
cl <- parallel::makeCluster(hosts)
streamplot(r2, cl=cl,
           par.settings=streamTheme(symbol=brewer.pal(n=5,
                                                      name='Reds')))
parallel::stopCluster(cl)

## Without parallel
streamplot(r3, parallel=FALSE,
           par.settings=streamTheme(symbol=brewer.pal(n=5,
                                                      name='Greens')))

## Configuration of droplets and streamlets
streamplot(s, layout=c(1, 3), droplet=list(pc=.2), streamlet=list(L=20),
           par.settings=streamTheme(cex=.6))

## End(Not run)
```

---

xyLayer *xyLayer*

Description

Create a RasterLayer from a function of the coordinates.

Usage

```
xyLayer(object, dirXY = y)
```

Arguments

object	A Raster object.
dirXY	A expression indicating the function of x and y (coordinates of the Raster object) to be evaluated.

Value

A RasterLayer object.

Author(s)

Oscar Perpiñán Lamigueiro.

See Also

init, substitute, eval

Examples

```
f <- system.file("external/test.grd", package="raster")
r <- raster(f)
dirX <- xyLayer(r, x)
dirXY <-xyLayer(r, sqrt(x^2 + y^2))
levelplot(dirXY, margin=FALSE)
```

---

xyplot-methods *xyplot for Raster objects*

Description

Scatter plots of space-time Raster objects for directions defined by xyLayer

Usage

```
## S4 method for signature 'RasterStackBrick,missing'
xyplot(x, data=NULL, dirXY=y,
       stat='mean', xlab='Time', ylab='',
       digits=0, par.settings=rasterTheme(),...)
```

Arguments

x	A RasterStackBrick object whose z slot is not NULL.
data	Not used.
dirXY	A direction as a function of the coordinates (see xyLayer).
stat	a function to be applied to summarize the values by zone. See zonal for details.
xlab, ylab	Labels of the axis.
par.settings	Customization of lattice. See xyplot for details.
digits	An integer, number of digits for zonal.
...	Additional arguments for the xyplot function.

Author(s)

Oscar Perpiñán Lamigueiro

See Also

zonal

Examples

```
## Not run:
##Solar irradiation data from CMSAF http://dx.doi.org/10.5676/EUM_SAF_CM/RAD_MVIRI/V001
old <- setwd(tempdir())
download.file('https://raw.github.com/oscarperpinan/spacetime-vis/master/data/SISmm2008_CMSAF.zip',
   'SISmm2008_CMSAF.zip', method='wget')
unzip('SISmm2008_CMSAF.zip')

listFich <- dir(pattern='\.nc')
stackSIS <- stack(listFich)
stackSIS <- stackSIS*24 ##from irradiance (W/m2) to irradiation Wh/m2

idx <- seq(as.Date('2008-01-15'), as.Date('2008-12-15'), 'month')

SISmm <- setZ(stackSIS, idx)
names(SISmm) <- month.abb

xyplot(SISmm)
## Formula interface
xyplot(Jan ~ Jul, data=SISmm)
## Different scatterplots for each latitude zone
xyplot(Jan ~ Dec|cut(y, 4), data=SISmm)
## More sophisticated bands can be defined using the dirXY argument
xyplot(Jan ~ Dec|cut(dirXY, 4), dirXY=x^2 + y^2, data=SISmm)

## End(Not run)

## Not run:
library(zoo)
```

```
url <- "ftp://ftp.wiley.com/public/sci_tech_med/spatio_temporal_data/"
sst.dat = read.table(paste(url, "SST011970_032003.dat", sep=''), header = FALSE)
sst.ll = read.table(paste(url, "SSTlonlat.dat", sep=''), header = FALSE)

spSST <- SpatialPointsDataFrame(sst.ll, sst.dat)
gridded(spSST) <- TRUE
proj4string(spSST) = "+proj=longlat +datum=WGS84"
SST <- brick(spSST)

idx <- seq(as.Date('1970-01-01'), as.Date('2003-03-01'), by='month')
idx <- as.yearmon(idx)
SST <- setZ(SST, idx)
names(SST) <- as.character(idx)

xyplot(SST)

## End(Not run)
```

Index