Package 'autoimage'

October 8, 2020

Type Package

Title Multiple Heat Maps for Projected Coordinates

Version 2.2.1

Date 2020-10-07

Author Joshua French

Maintainer Joshua French <joshua.french@ucdenver.edu>

BugReports https://github.com/jfrench/autoimage/issues

Description Functions for displaying multiple images or scatterplots with a color scale, i.e., heat maps, possibly with projected coordinates. The package relies on the base graphics system, so graphics are rendered rapidly.

Depends R (>= 2.10)

Imports colorspace, fields, mapproj, ggplot2, maps, MBA

Suggests abind, testthat, spatstat, gear, broom, gridExtra, knitr, rmarkdown

License GPL (>= 2)

LazyData TRUE

RoxygenNote 7.1.0

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2020-10-07 22:00:03 UTC

R topics documented:

`autoimage` *Automatic facetting of multiple projected images*

Description

`autoimage` plots a sequence of images (with possibly projected coordinates) while also automatically plotting a color scale matching the image colors to the values of `z`. Many options are available for legend customization. The coordinates can be irregularly spaced, on a regular grid, or on an irregular grid. `z` can be a numeric vector, matrix, or array, depending on the context.

Usage

```
autoimage(
  x,
  y,
  z,
  legend = "horizontal",
  proj = "none",
  parameters,
  orientation,
  common.legend = TRUE,
  map = "none",
  size,
  lratio,
```

```
  outer.title,
  ...
)
```

Arguments

x	Locations of grid points at which the values in z are measured. The values must be finite and non-missing. These arguments can be either vectors or matrices depending on the type of data to be displayed. See Details.
y	Locations of grid points at which the values in z are measured. The values must be finite and non-missing. These arguments can be either vectors or matrices depending on the type of data to be displayed. See Details.
z	A numeric or logical vector or matrix containing the values to be plotted (NAs are allowed).
legend	A character string indicating where the color scale should be placed. The default is "horizontal". The other valid options are "none" and "vertical".
proj	A character string indicating what projection should be used for the included x and y coordinates. The default is "none". The other valid choices correspond to the "projection" argument in the mapproject function, which is used for the projection.
parameters	A numeric vector specifying the values of the parameters argument in the mapproject. This may be necessary when proj != "none".
orientation	A vector c(latitude,longitude,rotation) which describes where the "North Pole" should be when computing the projection. See mapproject for more details.
common.legend	A logical value indicating whether a common legend scale should be used for all images provided in the z array. Default is TRUE. If FALSE, a separate legend is used for each image.
map	The name of the map to draw on the image. Default is "none". Other options include "world", "usa", "state", "county", "france", "nz" (New Zealand), "italy", "lakes", and "world2", all from the maps package.
size	A vector of length two indicating the number of rows and columns that should be used for the series of image data in z. Note that prod(size) must match the length of the third dimension of z (if it is an array), or c(1,1) if z is a matrix.
lratio	A numeric value indicating the ratio of the smaller dimension of the legend scale to the width of the image. Default is 0.1 + 0.1 * k, where k is the number of image rows if legend == "horizontal" or the number of image columns if legend == "vertical".
outer.title	A title related to all of the images that is plotted in the outer margin of the figure.
...	Additional arguments passed to the image or poly.image functions. e.g., xlab, ylab, xlim, ylim, zlim, etc.

Details

The mapproject function is used to project the x and y coordinates when proj != "none".

If multiple images are to be plotted (i.e., if z is an array), then the main argument can be a vector with length matching dim(z)[3], and each successive element of the vector will be used to add a title to each successive image plotted. See the Examples.

Additionally, if common.legend = FALSE, then separate limits for the z-axis of each image can be provided as a list. Specifically, if dim(z)[3] == k, then zlim should be a list of length k, and each element of the list should be a 2-dimensional vector providing the lower and upper limit, respectively, of the legend for each image. Alternatively, if zlim is a list of length k, then common.legend is set to FALSE.

The range of zlim is cut into n partitions, where n is the length of col.

It is generally desirable to increase lratio when more images are plotted simultaneously.

The multiple plots are constructed using the autolayout function, which is incompatible with the mfrow and mfcol arguments in the par function and is also incompatible with the split.screen function.

The mtext.args argument can be passed through ... in order to customize the outer title. This should be a named list with components matching the arguments of mtext.

Lines can be added to each image by passing the lines argument through .... In that case, lines should be a list with components x and y specifying the locations to draw the lines. The appearance of the plotted lines can be customized by passing a named list called lines.args through .... The components of lines.args should match the arguments of lines. See Examples.

Points can be added to each image by passing the points argument through .... In that case, points should be a list with components x and y specifying the locations to draw the points. The appearance of the plotted points can be customized by passing a named list called points.args through .... The components of points.args should match the components of points. See Examples.

Text can be added to each image by passing the text argument through .... In that case, text should be a list with components x and y specifying the locations to draw the text, and labels, a component specifying the actual text to write. The appearance of the plotted text can be customized by passing a named list called text.args through .... The components of text.args should match the components of text. See Examples.

The legend scale can be modified by passing legend.axis.args through .... The argument should be a named list corresponding to the arguments of the axis function. See Examples.

The image axes can be modified by passing axis.args through .... The argument should be a named list corresponding to the arguments of the axis function. The exception to this is that arguments xat and yat can be specified (instead of at) to specify the location of the x and y ticks. If xat or yat are specified, then this overrides the xaxt and yaxt arguments, respectively. See the paxes function to see how axis.args can be used.

The legend margin can be customized by passing legend.mar to pimage through .... This should be a numeric vector indicating the margins of the legend, identical to how par("mar") is specified.

The various options of the labeling, axes, and legend are largely independent. e.g., passing col.axis through ... will not affect the axis unless it is passed as part of the named list axis.args. However, one can set the various par options prior to plotting to simultaneously affect the appearance of multiple aspects of the plot. See Examples for pimage. After plotting, reset.par() can be used to reset the graphics device options to their default values.

See Also

pimage

Examples

```
data(narccap)
# restructure data for 2 images
tasmax2 <- tasmax[,,1:2]

# plot irregularly gridded images with separate legends
# and usa border
autoimage(lon, lat, tasmax2, common.legend = FALSE, map = "usa")

# plot irregularly gridded images with common legend and world lines
# customize world lines
# add and customize title
autoimage(lon, lat, tasmax2, map = "world",
          lines.args = list(col = "white", lwd = 2),
          outer.title = "Maximum Daily Surface Air Temperature (K)",
          mtext.args = list(col = "blue", cex = 2))

# plot irregularly-spaced responsed as images with separate legends
# and county borders.  Add observed data locations with custom point
# options.  Add text at locations of Denver and Colorado Springs.
data(co, package = "gear")
autoimage(co$lon, co$lat, co[,c("Al", "Ca")], common.legend = FALSE,
          map = "county", main = c("Aluminum", "Cadmium"),
          points = list(x = co$lon, y = co$lat),
          points.args = list(pch = 20, col = "white"),
          text = list(x = c(-104.98, -104.80), y = c(39.74, 38.85),
                      labels = c("Denver", "Colorado Springs")),
          text.args = list(col = "red"))

# customize margins and lratio for large plot
# also use projection
# specify manual lines (though in this case it is the same as using
# map = "world")
data(worldMapEnv, package = "maps")
worldpoly <- maps::map("world", plot = FALSE)
par(mar = c(1.1, 4.1, 2.1, 1.1))
autoimage(lon, lat, tasmax, lines = worldpoly,
          proj = "bonne", parameters = 40,
          main = c("day 1", "day 2", "day 3", "day 4", "day 5"),
          ylab = "",
          axes = FALSE,
          lratio = 0.5)
```

---

autolayout *Divide device into rows and columns*

Description

autolayout divides the current device into equal-sized rows and equal-sized columns based on the specified arguments.

Usage

```
autolayout(
  size,
  legend = "none",
  common.legend = TRUE,
  lratio = 0.2,
  outer = FALSE,
  show = TRUE,
  reverse = FALSE,
  legend.mar
)
```

Arguments

size	A vector of length two indicating the number of rows and columns that should be used for the series of image data in z. Note that prod(size) must match the length of the third dimension of z (if it is an array), or c(1,1) if z is a matrix.
legend	A character string indicating where the color scale should be placed. The default is "horizontal". The other valid options are "none" and "vertical".
common.legend	A logical value indicating whether a common legend scale should be used for all images provided in the z array. Default is TRUE. If FALSE, a separate legend is used for each image.
lratio	A numeric value indicating the ratio of the width of the legend scale to the width of the each image. Default is lratio = 0.2.
outer	A logical value indicating whether the room should be left for an outer title that is common for all plots. Depends on setting the oma argument of the par function.
show	A logical value indicating whether the layout.show function should be called after the layout is constructed.
reverse	A logical value indicating whether the legend scale should be plotted before the image. Default is FALSE.
legend.mar	The margins for the legend. (See the mar argument of par). If not specified, then sensible values are chosen based on the current vector par("mar").

Details

The rows and columns are constructed using the layout function, which is incompatible with the mfrow and mfcol arguments in the par function and is also incompatible with the split.screen function.

Note par parameters are NOT RESET after executing the layout function so the the user can use existing layout for plots.

If legend = "horizontal" or legend = "vertical", then a portion of the device is dedicated to a legend.

If common.legend = TRUE, then one legend region is created for the entire set of plots. If common.legend = FALSE, then a separate legend region is created for each individual plot.

With respective to ordering of the plotting regions: A common legend is plotted after all other plots, while individual legends are plotted after each respective plot.

See Also

image, image.plot, axis

Examples

```
# basic 2x2 layout
autolayout(c(2, 2))
# 3x2 layout with space for legends
autolayout(c(3, 2), legend = "h")
autolayout(c(3, 2), legend = "v")
# 3x2 layout with individuals legends
autolayout(c(3, 2), legend = "h", common.legend = FALSE)
autolayout(c(3, 2), legend = "v", common.legend = FALSE)
# if outer title is desired
autolayout(c(2, 2), outer = TRUE)
# reset oma parameters
par(oma = c(0, 0, 0, 0))
# impact of lratio when legend used
autolayout(c(2, 2), legend = "h", lratio = 0.5)
autolayout(c(2, 2), legend = "h", lratio = 0.2)
```

---

autolegend *Add legend to projected image.*

Description

autolegend adds a color scale to the current device based on the information from the last calls to the autolayout and pimage functions.

Usage

```
autolegend()
```

Details

Internally, autolegend calls the .legend.scale.args, .legend.horizontal, and .legend.mar functions to obtain the relevant information.

See Also

autolayout, pimage, legend.scale

Examples

```
data(narccap)
autolayout(c(1, 1), legend = "h")
pimage(lon, lat, tasmax[,,1], legend = "none")
autolegend()

# common legend with distinct lines
autolayout(c(1, 2), legend = "h")
pimage(lon, lat, tasmax[,,1], legend = "none", map = "world")
pimage(lon, lat, tasmax[,,2], legend = "none", map = "usa",
       proj = "bonne", parameters = 40)
autolegend()

# separate legends with distinct lines
autolayout(c(1, 2), legend = "v", common.legend = FALSE)
pimage(lon, lat, tasmax[,,1], legend = "none", map = "state",
       proj = "bonne", parameters = 40, axes = FALSE)
autolegend()
pimage(lon, lat, tasmax[,,2], legend = "none", map = "usa",
       proj = "albers", parameters = c(32, 45), axes = FALSE)
autolegend()

data(worldMapEnv, package = "maps")
# extract hawaii and alaskan borders
hiak <- maps::map("world", c("USA:Hawaii", "USA:Alaska"),
                  plot = FALSE)
# extract colorado cities from us.cities
data(us.cities, package = "maps")
codf <- us.cities[us.cities$country.etc == "CO", ]
# select smaller subset of colorado cities
codf <- codf[c(3, 5, 7:11, 15, 18), ]
# extract capitals from us.cities
capdf <- us.cities[us.cities$capital == 2,]

# setup plotting area
autolayout(c(1, 2), legend = "h", common.legend = FALSE, outer = TRUE)
# create image of NARCCAP data.
# xlim is chosen so to include alaska and hawaii
# add grey state borders
pimage(lon, lat, tasmax[,,1], legend = "none", proj = "mercator",
       map = "state", xlim = c(-180, 20),
       lines.args = list(col = "grey"))
# add hawaii and alaskan borders
plines(hiak, proj = "mercator", col = "grey")
# add state captials to image
ppoints(capdf$lon, capdf$lat, proj = "mercator", pch = 16)
# title image
title("tasmax for North America")
# add legend for plot
autolegend()
# load colorado geochemical data
data(co, package = "gear")
```

```
# create image for colorado aluminum measurements
# use bonne projection
# customize legend colors
# add grey county borders
pimage(co$lon, co$lat, co$Al, map = "county", legend = "none",
       proj = "bonne", parameters = 39,
       paxes.args = list(grid = FALSE),
       col = fields::tim.colors(64),
       lines.args = list(col = "grey"))
# add colorado city points to image
ppoints(codf$lon, codf$lat, pch = 16, proj = "bonne")
# add names of colorado cities to image
ptext(codf$lon, codf$lat, labels = codf$name, proj = "bonne", pos = 4)
# title plot
title("Colorado Aluminum levels (%)")
# add legend to current image
autolegend()
# add common title for plots
mtext("Two complicated maps", col = "purple", outer = TRUE, cex = 2)

reset.par() # reset device default
```

---

automar *Sensible legend margins*

Description

automar determines sensible margins for legend.scale based on the value currently set for par("mar").

Usage

```
automar(legend = "none")
```

Arguments

legend	A character string indicating the orientation of the legend.scale. The default is "none". The other valid options are "horizontal" and "vertical".

Details

The margins produced by automar are based on the current choice of par("mar"). If the user has specified a poor choice for par("mar"), then automar might also produce a poor choice for the legend margin.

Examples

```
automar()
automar("h")
automar("v")
```

`autopoints` *Automatic facetting of multiple projected scatterplots*

Description

`autopoints` plots a sequence of scatterplots (with possibly projected coordinates) while also automatically plotting a color scale matching the image colors to the values of `z`. Many options are available for customization. See the Examples below or execute `vignette("autopoints")` to better understand the possibilities.

Usage

```
autopoints(
  x,
  y,
  z,
  legend = "horizontal",
  proj = "none",
  parameters,
  orientation,
  common.legend = TRUE,
  map = "none",
  size,
  lratio,
  outer.title,
  n = 5,
  ...
)
```

Arguments

`x`	Numeric vectors of coordinates at which the values in `z` are measured.
`y`	Numeric vectors of coordinates at which the values in `z` are measured.
`z`	A numeric vector containing the values to be plotted.
`legend`	A character string indicating where the color scale should be placed. The default is `"horizontal"`. The other valid options are `"none"` and `"vertical"`.
`proj`	A character string indicating what projection should be used for the included `x` and `y` coordinates. The default is `"none"`. The other valid choices correspond to the `"projection"` argument in the `mapproject` function, which is used for the projection.
`parameters`	A numeric vector specifying the values of the `parameters` argument in the `mapproject`. This may be necessary when `proj != "none"`.
`orientation`	A vector `c(latitude,longitude,rotation)` which describes where the "North Pole" should be when computing the projection. See `mapproject` for more details.

common.legend	A logical value indicating whether a common legend scale should be used for all images provided in the z array. Default is TRUE. If FALSE, a separate legend is used for each image.
map	The name of the map to draw on the image. Default is "none". Other options include "world", "usa", "state", "county", "france", "nz" (New Zealand), "italy", "lakes", and "world2", all from the maps package.
size	A vector of length two indicating the number of rows and columns that should be used for the series of image data in z. Note that prod(size) must match the length of the third dimension of z (if it is an array), or c(1,1) if z is a matrix.
lratio	A numeric value indicating the ratio of the smaller dimension of the legend scale to the width of the image. Default is lratio = 0.2.
outer.title	A title related to all of the images that is plotted in the outer margin of the figure.
n	integer giving the *desired* number of intervals. Non-integer values are rounded down.
...	Additional arguments passed to the plot. e.g., xlab, ylab, xlim, ylim, zlim, etc.

Details

The n argument specifies the desired number of color partitions, but may not be exact. This is used to create "pretty" color scale tick labels using the pretty function. If zlim or breaks are provided, then the pretty function is not used to determine the color scale tick lables, and the user may need to manually specify breaks to get the color scale to have "pretty" tick labels. If col is specified, then n is set to length(col), but the functions otherwise works the same (i.e., pretty tick labels are not automatic if zlim or breaks are specified.

The mapproject function is used to project the x and y coordinates when proj != "none".

If multiple scatterplots are to be plotted (i.e., if z is a matrix with more than 1 column), then the main argument can be a vector with length matching ncol(z), and each successive element of the vector will be used to add a title to each successive scatterplot. See the Examples.

Additionally, if common.legend = FALSE, then separate z limits and breaks for the z-axis of each image can be provided as a list. Specifically, if ncol(z) == k, then zlim should be a list of length k, and each element of the list should be a 2-dimensional vector providing the lower and upper limit, respectively, of the legend for each image. Similarly, the breaks argument should be a list of length k, and each element of the list should be a vector specifying the breaks for the color scale for each plot. Note that the length of each element of breaks should be 1 greater then the number of colors in the color scale.

The range of zlim is cut into n partitions, where n is the length of col.

It is generally desirable to increase lratio when more images are plotted simultaneously.

The multiple plots are constructed using the autolayout function, which is incompatible with the mfrow and mfcol arguments in the par function and is also incompatible with the split.screen function.

The mtext.args argument can be passed through ... in order to customize the outer title. This should be a named list with components matching the arguments of mtext.

Lines can be added to each image by passing the lines argument through .... In that case, lines should be a list with components x and y specifying the locations to draw the lines. The appearance

of the plotted lines can be customized by passing a named list called `lines.args` through `....` The components of `lines.args` should match the arguments of `lines`. See Examples.

Points can be added to each image by passing the `points` argument through `....` In that case, `points` should be a list with components `x` and `y` specifying the locations to draw the points. The appearance of the plotted points can be customized by passing a named list called `points.args` through `....` The components of `points.args` should match the components of `points`. See Examples.

Text can be added to each image by passing the `text` argument through `....` In that case, `text` should be a list with components `x` and `y` specifying the locations to draw the text, and `labels`, a component specifying the actual text to write. The appearance of the plotted text can be customized by passing a named list called `text.args` through `....` The components of `text.args` should match the components of `text`. See Examples.

The legend scale can be modified by passing `legend.axis.args` through `....` The argument should be a named list corresponding to the arguments of the `axis` function. See Examples.

The axes can be modified by passing `axis.args` through `....` The argument should be a named list corresponding to the arguments of the `axis` function. The exception to this is that arguments `xat` and `yat` can be specified (instead of `at`) to specify the location of the x and y ticks. If `xat` or `yat` are specified, then this overrides the `xaxt` and `yaxt` arguments, respectively. See the `paxes` function to see how `axis.args` can be used.

The legend margin can be customized by passing `legend.mar` to `autpoints` through `....` This should be a numeric vector indicating the margins of the legend, identical to how `par("mar")` is specified.

The various options of the labeling, axes, and legend are largely independent. e.g., passing `col.axis` through `...` will not affect the axis unless it is passed as part of the named list `axis.args`. However, one can set the various `par` options prior to plotting to simultaneously affect the appearance of multiple aspects of the plot. See Examples for `pimage`. After plotting, `reset.par()` can be used to reset the graphics device options to their default values.

See Also

`autoimage`, `heat_ppoints`

Examples

```
data(co, package = "gear")
easting = co$easting
northing = co$northing
# heated scatterplot for Aluminum and Cadmium
autopoints(easting, northing, co[,c("Al", "Ca")],
           common.legend = FALSE, map = "state",
           main = c("Al", "Ca"), lratio = 0.2,
           legend.mar = c(0.3, 0.1, 0.1, 0.1))

# more complicated heat scatterplot for Aluminum and
# Cadmium used more advanced options
autopoints(co$lon, co$lat, co[,c("Al", "Ca")],
          common.legend = FALSE,
          map = "county", main = c("Aluminum", "Cadmium"),
```

```
            proj = "bonne", parameters = 40,
            text = list(x = c(-104.98, -104.80), y = c(39.74, 38.85),
                        labels = c("Denver", "Colorado Springs")),
            text.args = list(col = "blue"))
```

`autosize` *Automatically select plot matrix dimensions*

Description

`autosize` automatically makes a sensible choice for the dimensions of a plot matrix based on `n`, the number of plots. Only works for `n <= 36`. The dimensions are chosen to be as close to a square as possible.

Usage

```
autosize(n)
```

Arguments

`n`	The number of plots. Should be a positive integer.

Value

A vector of length 2 with the number of rows and number of columns for the plot matrix.

Examples

```
autosize(3)
autosize(9)
autosize(11)
autosize(24)
```

`blank.plot` *Draw blank plot*

Description

`blank.plot` draws a blank plot (no data, axis, or labels) on the current device.

Usage

```
blank.plot()
```

Details

Used by the `autoimage` function to fill remaining regions with white space when there are more plotting regions than images to plot.

See Also

`autoimage`

Examples

```
blank.plot()
```

---

canada *Provincial and territorial boundaries of Canada, 2001*

Description

An list-like object with components `x` and `y` specifying the provincial and territorial boundaries of Canada during the 2001 census. The coordinates are in longitude/latitude coordinates. The data was derived from the shapefiles provided by Statistics Canada. The object also has a component `range` specifying the range of the data in the order `c(min(x),max(x),min(y),max(y))`. Lastly, the object has a final component, `names`, which provides the name of the region each polygon is associated with. The object has class `map` for compatibility with the `maps` package.

Usage

```
data(canada)
```

Format

Contains:

x Longitude coordinates for Canadian boundaries

y Latitude coordinates for Canadian boundaries

range Range of x- and y-values

names Region name for each polygon

Source

Shapefile made available by Statistics Canada. https://www.statcan.gc.ca/eng/start.

copoly *Colorado state border*

Description

A list-like object with components `x` and `y` specifying the borders of the state of Colorado in longitude/latitude coordinates. This was derived from the `stateMapEnv` data set in the `maps` package. The object also has a component `range` specifying the range of the data in the order `c(min(x),max(x),min(y),max(y))`. Lastly, the object has a final component, `names`, which provides names for each polygon. In this case, the only name is `"colorado"`. The object has class `map` for compatibility with the `maps` package.

Usage

```
data(copoly)
```

Format

Contains:

x longitude coordinates for Colorado border

y latitude coordinates for Colorado border

range Range of x- and y-values

names Name of polygon

Source

The `stateMapEnv` data set in the `maps` package.

ggautoimage *Display images using ggplot2*

Description

`ggautoimage` produces a sequence of images in a manner similar to `autoimage` using the `ggplot2` package.

Usage

```
ggautoimage(
  x,
  y,
  z,
  f,
  proj = "none",
  parameters,
  orientation,
  lines,
  points,
  interp.args
)
```

Arguments

x	A numeric vector specifying the x coordinate locations.
y	A numeric vector specifying the y coordinate locations.
z	A numeric vector specifying the response for each (x,y) location.
f	A factor variable distinguishing between different facets, i.e., the different images to be constructed.
proj	A character string indicating what projection should be used for the included x and y coordinates. The default is "none". The other valid choices correspond to the "projection" argument in the mapproject function, which is used for the projection.
parameters	A numeric vector specifying the values of the parameters argument in the mapproject. This may be necessary when proj != "none".
orientation	A vector c(latitude,longitude,rotation) which describes where the "North Pole" should be when computing the projection. See mapproject for more details.
lines	A named list with components x and y specifiying the locations to be connected by lines. Distinct lines should be separated by NA values. See Details.
points	A named list with components x and y specifiying the locations to be plot points.
interp.args	A named list with component matching the non xyz arguments of the mba.surf function. Used to customize interpolation, when required.

Details

If x and y do not form a regular grid, then the mba.surf function is used to interpolate the locations onto a regular grid before constructing the image. This interpolation can be customized by passing interp.args through .... interp.args should be a named list with components matching the non xyz arguments of the mba.surf function.

When proj != "none", the mapproject function is used to project the x and y coordinates. In that case, proj must correspond to one of the choices for the projection argument in the mapproject function. Necessary arguments for mapproject should be provided through the parameters and orientation arguments. See Examples or mapproject for more details.

Lines can be added to each image by providing the `lines` argument. In that case, `lines` should be a list with components `x` and `y` specifying the locations to draw the lines. If more than one unconnected line should be drawn, then the coordinates should be separated by NA. e.g., to draw a line from (1, 1) to (2, 2) and (3, 3) to (4, 4) (with a gap between the two lines), you would specify lines as lines(x = c(1:2, NA, 3:4), y =c(1:2, NA, 3:4)). Also, see Examples.

Points can be added to each image by providing the `points` argument. In that case, `points` should be a list with components `x` and `y` specifying the locations to draw the points.

See Also

autoimage, image.plot, axis

Examples

```
data(narccap)
# setup image for two days of narccap data
x <- rep(c(lon), 2)
y <- rep(c(lat), 2)
z <- c(tasmax[, , 1:2])
f <- factor(rep(c("day 1", "day 2"), each = length(lon)))
# load national borders
data("worldMapEnv", package = "maps")
lines <- maps::map("world", plot = FALSE)
# obtain us captial cities
data(us.cities, package = "maps")
cap <- us.cities[us.cities$capital == 2, ]
# convert to list format
points <- list(x = cap$lon, y = cap$lat)

## Not run:
# basic images
ggautoimage(x, y, z, f)
# basic images with national borders and U.S. captials
ggautoimage(x, y, z, f, lines = lines, points = points)
# project coordinates with national borders and U.S. capitals
ggautoimage(x, y, z, f, lines = lines, points = points,
            proj = "bonne", parameters = 40)
# finer interpolation grid
ggautoimage(x, y, z, f, lines = lines, points = points,
            interp.args = list(no.X = 100, no.Y = 100))

## End(Not run)
```

`heat_ppoints` *Display colored scatterplot for projected coordinates*

Description

heat_ppoints plots a "heated" scatterplot for (potentially) projected locations. A color scale is automatically provided with the scatterplot. The function is similar in purpose to pimage, but the z-values are not interpolated. The color scale can be changed by passing a vector of colors to the col argument.

Usage

```
heat_ppoints(
  x,
  y,
  z,
  legend = "horizontal",
  proj = "none",
  parameters,
  orientation,
  lratio = 0.2,
  map = "none",
  n = 5,
  ...
)
```

Arguments

x, y	Numeric vectors of coordinates at which the values in z are measured.
z	A numeric vector containing the values to be plotted.
legend	A character string indicating where the color scale should be placed. The default is "horizontal". The other valid options are "none" and "vertical".
proj	A character string indicating what projection should be used for the included x and y coordinates. The default is "none". The other valid choices correspond to the "projection" argument in the mapproject function, which is used for the projection.
parameters	A numeric vector specifying the values of the parameters argument in the mapproject. This may be necessary when proj != "none".
orientation	A vector c(latitude,longitude,rotation) which describes where the "North Pole" should be when computing the projection. See mapproject for more details.
lratio	A numeric value indicating the ratio of the smaller dimension of the legend scale to the width of the image. Default is lratio = 0.2.
map	The name of the map to draw on the image. Default is "none". Other options include "world", "usa", "state", "county", "france", "nz" (New Zealand), "italy", "lakes", and "world2", all from the maps package.
n	integer giving the *desired* number of intervals. Non-integer values are rounded down.
...	Additional arguments passed to the plot function. e.g., xlab, ylab, xlim, ylim, zlim, etc. Additionally, arguments that can be used to further customize the plot (like adding lines or points), as described in Details and Examples.

Details

When proj != "none", the mapproject function is used to project the x and y coordinates. In that case, proj must correspond to one of the choices for the projection argument in the mapproject function. Necessary arguments for mapproject should be provided via the parameters and orientation arguments. See Examples and the mapproject function.

Valid options for legend are "none", "horizontal", and "vertical". If legend = "none", then no color scale is provided. If legend = "horizontal", then a color scale is included under the plot. If legend = "vertical", then a color scale is added to the right of the plot.

Lines can be added to each plot by passing the lines argument through .... In that case, lines should be a list with components x and y specifying the locations to draw the lines. The appearance of the plotted lines can be customized by passing a named list called lines.args through .... The components of lines.args should match the arguments of lines. See Examples.

Points can be added to each image by passing the points argument through .... In that case, points should be a list with components x and y specifying the locations to draw the points. The appearance of the plotted points can be customized by passing a named list called points.args through .... The components of points.args should match the components of points. See Examples.

Text can be added to each image by passing the text argument through .... In that case, text should be a list with components x and y specifying the locations to draw the text, and labels, a component specifying the actual text to write. The appearance of the plotted text can be customized by passing a named list called text.args through .... The components of text.args should match the components of text. See Examples.

The legend scale can be modified by passing legend.axis.args through .... The argument should be a named list corresponding to the arguments of the axis function. See Examples.

The plot axes can be modified by passing axis.args through .... The argument should be a named list corresponding to the arguments of the axis function. The exception to this is that arguments xat and yat can be specified (instead of at) to specify the location of the x and y ticks. If xat or yat are specified, then this overrides the xaxt and yaxt arguments, respectively. See the paxes function to see how axis.args can be used.

The legend margin can be customized by passing legend.mar to heat_ppoints through .... This should be a numeric vector indicating the margins of the legend, identical to how par("mar") is specified.

The various options of the labeling, axes, and legend are largely independent. e.g., passing col.axis through ... will not affect the axis unless it is passed as part of the named list axis.args. However, one can set the various par options prior to plotting to simultaneously affect the appearance of multiple aspects of the plot. See Examples. After plotting, reset.par() can be used to reset the graphics device options to their default values.

See Also

plot, axis, pimage

Examples

```
data(co, package = "gear")
# heated scatterplot for data on an irregular grid
```

```
heat_ppoints(co$lon, co$lat, co$Al, legend = "v", map = "state")
reset.par()

# change color scale
heat_ppoints(co$lon, co$lat, co$Al, col = cm.colors(5))
reset.par()

# Use custom border, x and y limits, breaks for legend axis
data(copoly)
heat_ppoints(co$lon, co$lat, co$Al, legend = "h",
       xlab = "longitude", ylab = "latitude",
       proj = "bonne", parameters = 40,
       lines = copoly,
       lines.args = list(lwd = 2, col = "grey"),
       xlim = c(-109.1, -102),
       ylim = c(36.8, 41.1),
       breaks = seq(0, 10, len = 6))
reset.par()
```

`inarccap` *Interpolated maximum daily surface air temperatures on a regular grid.*

Description

These data are the `narccap` data interpolated onto a regular 140×115 grid using the `akima::interp` function.

Usage

```
data(inarccap)
```

Format

Contains:

ilon A vector of of longitude coordinates.

ilat A vector of latitude coordinates.

itasmax A $140 \times 115 \times 5$ array of tasmax values.

See Also

`narccap`, `akima::interp`

`legend.scale` *Color scale legend*

Description

`legend.scale` plots a color gradient with an associated quantitative scale.

Usage

```
legend.scale(
  zlim,
  col = colorspace::sequential_hcl(n = 12, palette = "Viridis"),
  horizontal = TRUE,
  breaks,
  axis.args
)
```

Arguments

`zlim`	A two-dimensional vector containing the minimum and maximum quantitative limits, respectively, for the color scale.
`col`	A vector of colors used for the color scale. Typically, this is a gradient of colors. The default is the 12 colors generated by `colorspace::sequential_hcl(n = 12,palette = "Viridis")`.
`horizontal`	A logical value indicating whether the legend should extend horizontally (`TRUE`) or vertically (`FALSE`). The default is `TRUE`.
`breaks`	The sequence of values defining the partition of `zlim`. The length should be one more than then number of colors. If not specified, then equidistant breaks are used.
`axis.args`	A list of named elements corresponding to the arguments of the `axis` function. This is used to modify the appearance of the scale of the legend. See Examples.

Details

The length of the `col` vector indicates the number of partitions for the quantitative range.

References

The code for this function is derived from the internals of the `image.plot` function written by Doug Nychka and from the `image.scale` function written by Marc Taylor and discussed at https://menugget.blogspot.com/2013/12/new-version-of-imagescale-function.html.

See Also

`image`, `image.plot`, `axis`

Examples

```
# default horizontal scale
legend.scale(c(0, 1))

# default vertical scale
legend.scale(c(0, 1), horizontal = FALSE)

# different color scheme with 24 colors
legend.scale(c(0, 1), col = cm.colors(24))

# irregular color breaks
legend.scale(c(0, 1), col = heat.colors(4),
             breaks = c(0, 0.5, 0.75, 0.875, 1))

# irregular color breaks with modified ticks and vertical
# orientation of labels
legend.scale(c(0, 1), col = heat.colors(4),
             breaks = c(0, 0.5, 0.75, 0.875, 1),
             axis.args = list(at = c(0, 0.5, 0.75, 0.875, 1), las = 2))

# change size of axis labels
legend.scale(c(0, 1), axis.args = list(cex.axis = 2))

# change color of axis labels and ticks
blue.axes <- list(col.axis = "blue", col.ticks = "blue")
legend.scale(c(0, 1), axis.args = blue.axes)

# log base 10 values with colors labeled on original scale
options(scipen = 2)
log.axis <- list(at = 0:6, labels = 10^(0:6), las = 2)
legend.scale(c(0, 6), col = heat.colors(6), axis.args = log.axis)
```

---

narccap *Maximum daily surface air temperatures on a grid.*

Description

These data are taken from the North American Regional Climate Change Assessment Program (NARCCAP). Specifically, the data provide maximum daily surface air temperature (K) (abbreviated tasmax) for locations in the United States, Mexico, and Canada for the five consecutive days of May 15, 2041 to May 19, 2041 simulated using the Canadian Regional Climate Model (Caya and Laprise, 1999) forced by the Community Climate System Model atmosphere-ocean general circular model (Collins et al., 2006)

Usage

```
data(narccap)
```

Format

Contains:

lon A 140×115 matrix of longitude coordinates.

lat A 140×115 matrix of latitude coordinates.

tasmax A 140×115×5 array of tasmax values.

Source

The National Center for Atmospheric Research (NCAR) through the North American Regional Climate Change Assessment Program (NARCCAP) <doi:10.5065/D6RN35ST>.

References

Mearns, L.O., et al., 2007, updated 2012. The North American Regional Climate Change Assessment Program dataset, National Center for Atmospheric Research Earth System Grid data portal, Boulder, CO. Data downloaded 2016-08-12. <doi:10.5065/D6RN35ST>.

Mearns, L. O., W. J. Gutowski, R. Jones, L.-Y. Leung, S. McGinnis, A. M. B. Nunes, and Y. Qian: A regional climate change assessment program for North America. EOS, Vol. 90, No. 36, 8 September 2009, pp. 311-312.

D. Caya and R. Laprise. A semi-implicit semi-Lagrangian regional climate model: The Canadian RCM. Monthly Weather Review, 127(3):341-362, 1999.

M. Collins, B. B. Booth, G. R. Harris, J.M. Murphy, D. M. Sexton, and M. J. Webb. Towards quantifying uncertainty in transient climate change. Climate Dynamics, 27(2-3):127-147, 2006.

`parrows` *Projected arrows function*

Description

`parrows` takes pairs of coordinates and draws arrows between them, possibly after projection.

Usage

```
parrows(x0, y0, x1 = x0, y1 = y0, proj, ...)
```

Arguments

`x0, y0`	coordinates of points from which to draw.
`x1, y1`	coordinates of points to which to draw
`proj`	A character string indicating what projection should be used for the included `x` and `y` coordinates. The default is `"none"`. The other valid choices correspond to the `"projection"` argument in the `mapproject` function, which is used for the projection.
`...`	Additional arguments passed to the `arrows`

Details

The mapproject function is used for projection.

See Also

arrows, mapproject, pimage

Examples

```
data(narccap)
# plot image using bonne projection (w/o grid lines)
pimage(lon, lat, tasmax[,,1], proj = "bonne",
       parameters = 40, paxes.args = list(grid = FALSE))
# load some data for larger U.S. cities
data(us.cities, package = "maps")
cityxy <- list(x = us.cities$long[1:5], y = us.cities$lat[1:5])
parrows(cityxy$x[1:4], cityxy$y[1:4], cityxy$x[2:5], cityxy$y[2:5],
        proj = "bonne", col = "orange")
```

---

paxes *Display axes for projected coordinates*

Description

paxes addes x and y axes to an existing plot for projected coordinates.

Usage

```
paxes(proj, xlim, ylim, xaxp, yaxp, grid = TRUE, axis.args, ...)
```

Arguments

proj	A character string indicating what projection should be used for the included x and y coordinates. The default is "none". The other valid choices correspond to the "projection" argument in the mapproject function, which is used for the projection.
xlim	A vector with the minimum and maximum value of the x coordinates. Taken from par("usr") if not provided.
ylim	A vector with the minimum and maximum value of the y coordinates. Taken from par("usr") if not provided.
xaxp	A vector of the form c(x1,x2,n) giving the coordinates of the extreme tick marks and the number of intervals between tick marks. Overrides xlim.
yaxp	A vector of the form c(x1,x2,n) giving the coordinates of the extreme tick marks and the number of intervals between tick marks. Overrides ylim.
grid	A logical value indicating whether grid lines should be displayed with the axes. Default is TRUE.

axis.args	A named list with components matching the arguments of axis. See Details and Examples.
...	Other arguments passed to the [graphics]{lines} function used to plot the grid lines.

Details

The mapproject function is used for projection.

axis.args should be a named list matching the arguments of axis. The exception is that xat and yat can be specified to induce different spacing of the ticks on the x and y axes. Thus, the at argument is ignored and replaced by xat and yat, as appropriate.

See Also

image, mapproject

Examples

```
data(narccap)
# plot image using mercator projection (w/o axes)
pimage(lon, lat, tasmax[,,1], proj = "mercator", axes = FALSE)
# add axes with grey grid lines, blue text, and custom spacing
paxes("mercator", xlim = range(lon), ylim = range(lat),
      col = "grey",
      axis.args = list(col.axis = "blue",
                       xat = c(-160, -100, -90, -80, -20)))
```

---

pimage *Display image for projected coordinates*

Description

pimage plots an image for (potentially) projected locations. A color scale is automatically provided with the image. The function is essentially an extension of the image function and the x and y locations can be irregularly-spaced locations, sequences of increasing values for locations on a regular grid, or matrices (with dimensions matching those of z) for locations on an irregular grid. Functionality for automatic projection is provided.

Usage

```
pimage(
  x,
  y,
  z,
  legend = "horizontal",
  proj = "none",
  parameters,
  orientation,
```

```
  lratio = 0.2,
  map = "none",
  ...
)
```

Arguments

x, y	Locations of grid points at which the values in z are measured. The values must be finite and non-missing. These arguments can be either vectors or matrices depending on the type of data to be displayed. See Details.
z	A numeric or logical vector or matrix containing the values to be plotted (NAs are allowed).
legend	A character string indicating where the color scale should be placed. The default is "horizontal". The other valid options are "none" and "vertical".
proj	A character string indicating what projection should be used for the included x and y coordinates. The default is "none". The other valid choices correspond to the "projection" argument in the mapproject function, which is used for the projection.
parameters	A numeric vector specifying the values of the parameters argument in the mapproject. This may be necessary when proj != "none".
orientation	A vector c(latitude,longitude,rotation) which describes where the "North Pole" should be when computing the projection. See mapproject for more details.
lratio	A numeric value indicating the ratio of the smaller dimension of the legend scale to the width of the image. Default is lratio = 0.2.
map	The name of the map to draw on the image. Default is "none". Other options include "world", "usa", "state", "county", "france", "nz" (New Zealand), "italy", "lakes", and "world2", all from the maps package.
...	Additional arguments passed to the image or poly.image functions. e.g., xlab, ylab, xlim, ylim, zlim, etc. Additionally, arguments that can be used to further customize the plot (like adding lines or points), as described in Details and Examples.

Details

If x, y, and z are numeric vectors of the same length, then the mba.surf function is used to predict the response on a regular grid using multilevel B-splines before constructing the image. This interpolation can be customized by passing interp.args through .... interp.args should be a named list with component matching the non xyz arguments of the mba.surf function.

If x are y are vectors of increasing values and nrow(z) == length(x) and ncol(z) == length(y), then an image on a regular grid is constructed.

If x, y and z are matrices with the same dimensions, then an image for irregularly gridded data is constructed.

When proj != "none", the mapproject function is used to project the x and y coordinates. In that case, proj must correspond to one of the choices for the projection argument in the mapproject

function. Necessary arguments for mapproject should be provided via the parameters and orientation arguments. See Examples and the mapproject function.

Valid options for legend are "none", "horizontal", and "vertical". If legend = "none", then no color scale is provided. If legend = "horizontal", then a color scale is included under the image. If legend = "vertical", then a color scale is added to the right of the image.

Lines can be added to each image by passing the lines argument through .... In that case, lines should be a list with components x and y specifying the locations to draw the lines. The appearance of the plotted lines can be customized by passing a named list called lines.args through .... The components of lines.args should match the arguments of lines. See Examples.

Points can be added to each image by passing the points argument through .... In that case, points should be a list with components x and y specifying the locations to draw the points. The appearance of the plotted points can be customized by passing a named list called points.args through .... The components of points.args should match the components of points. See Examples.

Text can be added to each image by passing the text argument through .... In that case, text should be a list with components x and y specifying the locations to draw the text, and labels, a component specifying the actual text to write. The appearance of the plotted text can be customized by passing a named list called text.args through .... The components of text.args should match the components of text. See Examples.

The legend scale can be modified by passing legend.axis.args through .... The argument should be a named list corresponding to the arguments of the axis function. See Examples.

The image axes can be modified by passing axis.args through .... The argument should be a named list corresponding to the arguments of the axis function. The exception to this is that arguments xat and yat can be specified (instead of at) to specify the location of the x and y ticks. If xat or yat are specified, then this overrides the xaxt and yaxt arguments, respectively. See the paxes function to see how axis.args can be used.

The legend margin can be customized by passing legend.mar to pimage through .... This should be a numeric vector indicating the margins of the legend, identical to how par("mar") is specified.

The various options of the labeling, axes, and legend are largely independent. e.g., passing col.axis through ... will not affect the axis unless it is passed as part of the named list axis.args. However, one can set the various par options prior to plotting to simultaneously affect the appearance of multiple aspects of the plot. See Examples. After plotting, reset.par() can be used to reset the graphics device options to their default values.

See Also

image, image.plot, axis

Examples

```
# image plot for data on an irregular grid
pimage(lon, lat, tasmax[,,1], legend = "h", map = "world")
# same plot but with projection and vertical legend
pimage(lon, lat, tasmax[,,1], legend = "v", map = "world",
       proj = "bonne", parameters = 45)
# different projection
pimage(lon, lat, tasmax[,,1], proj = "albers",
```

```
       parameters = c(33, 45), map = "world")

reset.par() # reset graphics device
# image plot for non-gridded data
data(co, package = "gear")
pimage(co$longitude, co$latitude, co$Al)

# show observed locations on image,
# along with Colorado border, locations of Denver and Colorado
# Springs
data(copoly)
copoints <- list(x = co$lon, y = co$lat)
pimage(co$longitude, co$latitude, co$Al,
       lines = copoly,
       lines.args = list(lwd = 2, col = "grey"),
       points = copoints,
       points.args = list(pch = 21, bg = "white"),
       text = list(x = c(-104.98, -104.80), y = c(39.74, 38.85),
                   labels = c("Denver", "Colorado Springs")),
       text.args = list(col = "purple"),
       xlim = c(-109.1, -102),
       ylim = c(36.8, 41.1))

# image plot for data on irregular grid
# notice the poor axis labeling
data(narccap)
pimage(lon, lat, tasmax[,,1], proj = "bonne",
       parameters = 45, map = "world")
# same plot but customize axis labeling
# need to extend horizontally-running axis lines
# farther to the west and east
# also need the vertically-running lines
# to run further north/sount
# will need manual adjusting depending on size
# of current device
pimage(lon, lat, tasmax[,,1], proj = "bonne",
       parameters = 45, map = "world",
       xaxp = c(-200, 0, 10), yaxp = c(-10, 80, 9))

# the same effect can be acheived by specifying axis.args
# we also modify the color and size of the axis labels
pimage(lon, lat, tasmax[,,1], proj = "bonne",
       parameters = 45, map = "world",
       axis.args = list(xat = seq(-200, 0, by = 20),
                        yat = seq(0, 70, by = 10),
                        col.axis = "blue",
                        cex.axis = 0.5))

# modify colors of legend, map, line type for grid lines
# and customize axis
pimage(lon, lat, tasmax[,,1],
       legend = "v", proj = "bonne", parameters = 45,
       map = "state",
```

```
        paxes.args = list(lty = 3),
        legend.axis.args = list(col = "blue", col.axis = "blue"),
        col = fields::tim.colors(64),
        xlab = "longitude",
        ylab = "latitude",
        main = "temperature (K)")
reset.par() # reset graphics device

# change many aspects of plot appearance using par
par(cex.axis = 0.5, cex.lab = 0.5, mgp = c(1.5, 0.5, 0),
    mar = c(2.1, 2.1, 4.1, 0.2), col.axis = "orange",
    col.main = "blue", family = "mono")
pimage(lon, lat, tasmax[,,1])
title("very customized plot")
reset.par()
```

`plines` *Projected lines function*

Description

`plines` takes coordinates and joins the corresponding points with line segments, possibly after projection.

Usage

```
plines(x, y = NULL, type = "l", proj, ...)
```

Arguments

`x`	coordinate vectors of points to join.
`y`	coordinate vectors of points to join.
`type`	character indicating the type of plotting; actually any of the types as in `plot.default`.
`proj`	A character string indicating what projection should be used for the included `x` and `y` coordinates. The default is `"none"`. The other valid choices correspond to the `"projection"` argument in the `mapproject` function, which is used for the projection.
`...`	Further graphical parameters (see `par`) may also be supplied as arguments, particularly, line type, `lty`, line width, `lwd`, color, `col` and for `type = "b"`, `pch`. Also the line characteristics `lend`, `ljoin` and `lmitre`.

Details

The `mapproject` function is used for projection.

See Also

pimage, mapproject, lines

Examples

```
data(narccap)
# plot image using bonne projection (w/o grid lines)
pimage(lon, lat, tasmax[,,1], proj = "bonne",
       parameters = 40, paxes.args = list(grid = FALSE))
# get national boundaries
data(worldMapEnv, package = "maps")
worldpoly <- maps::map("world", plot = FALSE)
# add boundaries to existing plot
plines(worldpoly, proj = "bonne")
```

---

ppoints *Projected points function*

Description

ppoints draws a sequence of points for projected coordinates.

Usage

```
ppoints(x, y = NULL, type = "p", proj, ...)
```

Arguments

x	coordinate vectors of points to plot.
y	coordinate vectors of points to plot.
type	character indicating the type of plotting; actually any of the types as in plot.default.
proj	A character string indicating what projection should be used for the included x and y coordinates. The default is "none". The other valid choices correspond to the "projection" argument in the mapproject function, which is used for the projection.
...	Further graphical parameters passed to the points function.

Details

The mapproject function is used for projection.

See Also

points, mapproject, pimage

Examples

```
data(narccap)
# plot image using bonne projection (w/o grid lines)
pimage(lon, lat, tasmax[,,1], proj = "bonne",
       parameters = 40, paxes.args = list(grid = FALSE))
# get U.S. cities with population of about 40k or more
data(us.cities, package = "maps")
# add cities to existing plot
ppoints(us.cities$long, us.cities$lat, proj = "bonne")
```

---

ppolygon *Projected polygon function*

Description

`ptext` draws polygons using the vertices given in `x` and `y`, possibly with projected coordinates.

Usage

```
ppolygon(x, y = NULL, proj, ...)
```

Arguments

`x`	vectors containing the coordinates of the vertices of the polygon.
`y`	vectors containing the coordinates of the vertices of the polygon.
`proj`	A character string indicating what projection should be used for the included `x` and `y` coordinates. The default is `"none"`. The other valid choices correspond to the `"projection"` argument in the `mapproject` function, which is used for the projection.
`...`	graphical parameters such as `xpd`, `lend`, `ljoin` and `lmitre` can be given as arguments.

Details

The `mapproject` function is used for projection.

See Also

`polygon`, `mapproject`, `pimage`

Examples

```
data(narccap)
# plot image using bonne projection (w/o grid lines)
pimage(lon, lat, tasmax[,,1], proj = "bonne",
       parameters = 40, paxes.args = list(col = "grey"))
# filled polygon for Colorado border
data(copoly)
ppolygon(copoly, proj = "bonne", col = "orange")
```

psegments *Projected segments function*

Description

psegments takes pairs of coordinates and draws line segments between them, possibly after projection.

Usage

```
psegments(x0, y0, x1 = x0, y1 = y0, proj, ...)
```

Arguments

x0, y0	coordinates of points from which to draw.
x1, y1	coordinates of points to which to draw
proj	A character string indicating what projection should be used for the included x and y coordinates. The default is "none". The other valid choices correspond to the "projection" argument in the mapproject function, which is used for the projection.
...	Additional arguments to pass to the segments function.

Details

The mapproject function is used for projection.

See Also

segments, mapproject, pimage

Examples

```
data(narccap)
# plot image using bonne projection (w/o grid lines)
pimage(lon, lat, tasmax[,,1], proj = "bonne",
       parameters = 40, paxes.args = list(grid = FALSE))
# some locations for u.s. cities
# taken from data(us.cities, package = "maps")
boston <- c(-71.02, 42.34)
la <- c(-118.41, 34.11)
ny <- c(-73.94, 40.67)
sf <- c(-122.45, 37.77)
# plot segments between sf, la and ny boston
x0 <- c(sf[1], ny[1])
y0 <- c(sf[2], ny[2])
x1 <- c(la[1], boston[1])
y1 <- c(la[2], boston[2])
psegments(x0, y0, x1, y1, proj = "bonne", lwd = 3)
```

```
citycoords <- rbind(sf, la, ny, boston)
cityxy <- list(x = citycoords[,1], y = citycoords[,2])
citynames <- c("san francisco", "los angeles", "new york", "boston")
ptext(cityxy, labels = citynames, proj = 'bonne')
```

---

ptext *Projected text function*

Description

ptext draws the character strings given in labels at the provided coordinates, possibly after projection.

Usage

```
ptext(x, y = NULL, labels, proj, ...)
```

Arguments

x	numeric vectors of coordinates where the text labels should be written. If the length of x and y differs, the shorter one is recycled.
y	numeric vectors of coordinates where the text labels should be written. If the length of x and y differs, the shorter one is recycled.
labels	a character vector or expression specifying the *text* to be written. An attempt is made to coerce other language objects (names and calls) to expressions, and vectors and other classed objects to character vectors by as.character. If labels is longer than x and y, the coordinates are recycled to the length of labels.
proj	A character string indicating what projection should be used for the included x and y coordinates. The default is "none". The other valid choices correspond to the "projection" argument in the mapproject function, which is used for the projection.
...	further graphical parameters (from par), such as srt, family and xpd.

Details

The mapproject function is used for projection.

A non-character labels argument is automatically converted to a character vector using link[base]{as.character}.

See Also

text, mapproject, pimage

Examples

```
data(narccap)
# plot image using bonne projection (w/o grid lines)
pimage(lon, lat, tasmax[,,1], proj = "bonne",
       parameters = 40, paxes.args = list(grid = FALSE))
# get national boundaries
data(worldMapEnv, package = "maps")
worldpoly <- maps::map("world", plot = FALSE)
plines(worldpoly, proj = "bonne")
# add U.S. city names to existing plot
data(us.cities, package = "maps")
citysmall <- head(us.cities)
ptext(x = citysmall$lon, y = citysmall$lat,
      labels = citysmall$name, proj = "bonne")
```

`reset.par` *Reset par*

Description

`reset.par` resets the arguments of `par` to the default values when first opening R (as of version 3.2.2).

Usage

```
reset.par()
```

See Also

`par`

Examples

```
par("mar") #current values of mar
par(mar = c(0, 0, 0, 0)) # change values of mar
par("mar") # changed values of mar
reset.par() # reset to defaults (not necessarilly current values)
par("mar") # should be c(5.1, 4.1, 4.1, 2.1)
```

`rotate` *Rotate coordinates*

Description

`rotate` rotates the coordinates by angle theta around a pivot point.

Usage

```
rotate(coords, theta, pivot = c(0, 0))
```

Arguments

`coords`	A 2-column matrix with the coordinates to be rotated.
`theta`	The angle (in radians) to rotate the coordinates.
`pivot`	The pivot point around which the coordinates are rotated. Default is c(0, 0), i.e., the origin.

Examples

```
# coordinates to rotate
coords <- matrix(rnorm(20), ncol = 2)
# rotate coordinates pi/6 radians around the original
rcoords <- rotate(coords, pi/6)
#compare original coordinates to rotated coordinates
par(mfrow = c(1, 2))
plot(coords)
plot(rcoords)
```

Index