Package 'graphx'

February 3, 2022

Type Package

Title Graphics Routines for Scientific Research

Version 1.0

Author Danail Obreschkow

Maintainer Danail Obreschkow <danail.obreschkow@gmail.com>

Description Routines to produce and export publication-ready scientific plots and movies (e.g., used in Obreschkow et al. (2020) <doi:10.1093/mnras/staa445>). These include special color scales, projection routines, and bitmap handling routines.

Depends R (>= 3.1.1)

Imports magicaxis, pracma, plotrix, cubature, png, jpeg, raster, MASS, sp, docore

Suggests EBImage

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2022-02-03 14:30:04 UTC

R topics documented:

`.graphx.env` *Package environment*

Description

Environment used to store global variables in the package, e.g. used for subplot routine.

Usage

```
.graphx.env
```

Format

An object of class `environment` of length 0.

Value

None

Author(s)

Danail Obreschkow

`cmplx2col` *Convert complex numbers to color*

Description

Converts a complex number (or a vector/array thereof) into a color-string, such that brightness represents the complex amplitude and hue represents the complex phase.

Usage

```
cmplx2col(z, max = NULL, hue = 0, saturation = 1, gamma = 1)
```

Arguments

`z`	complex number or vector/array of complex numbers
`max`	value of the complex module that corresponds to full brightness; if not given, this is set equal to the maximum complex amplitude in z.
`hue`	hue value of positive reals
`saturation`	saturation value of colors
`gamma`	positive number to adjust the non-linearity of the color scale (1=linear)

Value

Returns a single color or vector/array of colors

Author(s)

Danail Obreschkow

`colorbar` *Vertical color bar*

Description

Adds a vertical color bar to a plot with a custom axis.

Usage

```
colorbar(
  xleft,
  ybottom,
  xright,
  ytop,
  col = gray.colors(256, 0, 1),
  n = length(col),
  clim = c(0, 1),
  show.border = TRUE,
  text = "",
  line = 2,
  show.axis = TRUE,
  side = "right",
  lwd = 1,
  nticks = 5,
  at = NULL,
  ...
)
```

Arguments

`xleft`	left x-coordinate of the bar
`ybottom`	bottom y-coordinate of the bar
`xright`	right x-coordinate of the bar
`ytop`	top y-coordinate of the bar
`col`	vector of colors
`n`	number of polygons used to draw the bar
`clim`	2-vector specifying the range of values, linearly represented by the full color scale
`show.border`	logical flag specifying whether to draw a rectangular border around the bar
`text`	axis label
`line`	distance between label and color bar
`show.axis`	logical flag specifying whether to draw an axis
`side`	character string, which specifies the location of the axis on the bar. This has to be 'left' or 'right' (default).
`lwd`	linewidth of border and ticks
`nticks`	number of ticks on the axis
`at`	vector of values specifying the tick positions on the axis, this overrides nticks.
`...`	optional arguments to be passed to the function `axis`

Value

None

Author(s)

Danail Obreschkow

Examples

```
## Plot a spherical function with color bar
nplot(xlim=c(0,1.2), asp=1)
f = function(theta,phi) cos(10*theta+phi)
sp = sphereplot(f, 200, col=planckcolors(200), phi0=0.1, theta0=pi/3,
add=TRUE, center=c(0.5,0.5), radius=0.4, clim=c(-1,1))
colorbar(1,0.1,1.1,0.9,col=sp$col,clim=sp$clim)
```

`cubehelix` *Cube Helix colour palette*

Description

Generate the Cube Helix colour palette, designed to appropriately display of intensity images, as the brightness increases monotonically when displayed in greyscale.

Usage

```
cubehelix(n, r = 1.5, hue = 1, gamma = 1, rev = FALSE)
```

Arguments

`n`	integer number of colours in the scale
`r`	real number specifying the number of rotations of the helix over the scale
`hue`	non-negative number specifying the colour intensity from grey (0) to normal (1) and over-saturated (>1)
`gamma`	positive number specifying the relative importance of low vs high values
`rev`	logical flag indicating whether the ordering of the colors should be reversed

Details

This scheme was published by Green, D. A., 2011, "A colour scheme for the display of astronomical intensity images." Bulletin of the Astronomical Society of India, 39, 289.

Value

Returns an n-vector of RGB colour strings.

Author(s)

Danail Obreschkow

`errlines` *Draw a line with uncertainty regions*

Description

Draw a line with uncertainty regions

Usage

```
errlines(
  x,
  y,
  errp,
  errn = errp,
  col = "black",
  alpha = 0.5,
  smooth = FALSE,
  df = NULL,
  ...
)
```

Arguments

`x`	vector of x-coordinates
`y`	vector of y-coordinates
`errp`	vector of y-errors in the positive direction (upwards)
`errn`	vector of y-errors in the negative direction (downwards)
`col`	color of the line
`alpha`	transparency of the uncertainty region
`smooth`	logical flag indicating whether the line should be smoothed
`df`	df (=degrees of freedom) parameter of `smooth.spline` function
`...`	additional parameters used by `lines`

Value

None

Author(s)

Danail Obreschkow

graphx *Graphics Routines for Scientific Research*

Description

Routines to produce and export publication-ready scientific plots and movies (e.g., used in Obreschkow et al. (2020) <doi:10.1093/mnras/staa445>). These include special color scales, projection routines, and bitmap handling routines.

Author(s)

Danail Obreschkow

invert *Invert and shift colors of an image*

Description

Invert the brightness of each color channel in an image and/or circularly shifts the hue value. Optionally, a Gaussian blur can be applied.

Usage

```
invert(
  img = NULL,
  invert = TRUE,
  colshift = 0,
  blur = 0,
  file.in = "",
  file.out = "",
  format = "png",
  show.image = TRUE
)
```

Arguments

`img`	n-by-m-by-3 array or n-by-m-by-4 array representing an rgb(+alpha) image
`invert`	logical flag indicating whether the channel-brightness should be inverted
`colshift`	numerical value between 0 and 1 setting the circular shift of the hue value. If `invert=TRUE`, choosing `colshift=0.5` preserves the colors, while inverting black and white.
`blur`	numerical value >=0 defining the standard deviation of an optional Gaussian blur.

`file.in`	optional input filename, which can be used to load an image instead of providing it via `img`. This filename is ignored if `img` is specified.
`file.out`	optional output filename.
`format`	one of "png" or "jpg" specifying the file format of the input and output image.
`show.image`	logical flag specifying whether the image is displayed in the R console.

Value

Returns an n-by-m-by-3 array or n-by-m-by-4 array of the processed image.

Author(s)

Danail Obreschkow

Examples

```
img = yinyangyong # this is an example image included in the package

# invert brightness of all channels
invert(img)

# invert brightness, but preserve hue
invert(img, colshift=0.5)
```

`lightness` *Change lightness of a color*

Description

Change lightness of a color

Usage

```
lightness(col, light = 0.5)
```

Arguments

`col`	is a color or vector/array of colors, specified as text (e.g. ’purple’) or 7/9-character (e.g. ’#A020F0’)
`light`	lightness value, according to a HSL scheme, between 0 and 1 or a vector/array thereof

Value

Returns a 9-character color or vector/array of 9-character colors.

Author(s)

Danail Obreschkow

See Also

`transparent`

Examples

```
# Generate different lightnesses of the same color
plot(runif(50),runif(50),pch=20,cex=10,col=lightness('purple',runif(50)))
```

`makeframe` *Display a single movie frame*

Description

Displays a single movie-frame in the R-console, exactly as used in a movie generated with `makemovie`.

Usage

```
makeframe(
  frame.draw,
  frame.index,
  width = 1080,
  height = 720,
  keep.frame = FALSE
)
```

Arguments

`frame.draw`	function that plots an individual frame. This function must have exactly one argument ’x’, which can be integer (e.g. a simple frame index) or real (e.g. a time).
`frame.index`	list of frame indices ’x’ to be included in the movie
`width`	number of pixels along the horizontal axis
`height`	number of pixels along the vertical axis
`keep.frame`	logical flag specifying whether to keep the temporary png-file of the frame

Value

None

Author(s)

Danail Obreschkow

See Also

`makemovie`

Examples

```
## Example: Movie of a manual clock

# Function to draw a single clock face with two hands
frame = function(time) {
  par(mar=c(0,0,0,0))
  nplot(xlim=c(-1.1,1.1),ylim=c(-1.1,1.1),pty='s')
  plotrix::draw.circle(0,0,1,col='#aaaaff')
  radius = c(0.5,0.9)
  speed = 2*pi/c(720,60)
  lwd = c(4,2)
  graphics::arrows(0,0,radius*sin(speed*time),radius*cos(speed*time),lwd=lwd)
}

# Produce movie
## Not run:
makeframe(frame,15,200,200)

## End(Not run)
```

`makemovie` *Produce a movie from frame-drawing function*

Description

Generates an MP4-movie provided a custom function that plots individual frames. The routine has been developed and tested for MacOS and it requires on a working installation of ffmpeg.

Usage

```
makemovie(
  frame.draw,
  frame.index,
  output.path,
  output.filename,
  width = 1080,
  height = 720,
  fps = 60,
  smear = NULL,
  keep.frames = FALSE,
  quiet = TRUE,
  separator = "/",
  ffmpeg.cmd = "ffmpeg",
  ffmpeg.opt = "-vcodec libx264 -crf 18 -pix_fmt yuv420p",
  manual = FALSE
)
```

Arguments

`frame.draw`	function that plots an individual frame. This function must have exactly one argument 'x', which can be integer (e.g. a simple frame index) or real (e.g. a time).
`frame.index`	list of frame indices 'x' to be included in the movie
`output.path`	character specifying the directory, where the movie and temporary frames are saved
`output.filename`	movie filename without path. This filename should end on the extension '.mp4'.
`width`	number of pixels along the horizontal axis
`height`	number of pixels along the vertical axis
`fps`	number of frames per second
`smear`	optional number of sub-frames used to smear each frame. If given, the function frame.draw must accept continuous arguments in between the values of frame.index.
`keep.frames`	logical flag specifying whether the temporary directory with the individual frame files should be kept
`quiet`	logical flag; if true, all console outputs produced by 'ffmpeg' are suppressed
`separator`	filename separate of the system ('/' for Mac, Linux, Unix; '\' for Windows)
`ffmpeg.cmd`	command used to call ffmpeg form a terminal. Normally, this is just 'ffmpeg'.
`ffmpeg.opt`	compression and formatting options used with ffmpeg
`manual`	logical flag; if true, ffmpeg is not called from within the code and the frames are never deleted. The suggested linux command line is returned as output.

Value

Linux command line to convert frames into movie using ffmpeg.

Author(s)

Danail Obreschkow

See Also

`makeframe`

Examples

```
## Example: Movie of a manual clock

# Function to draw a single clock face with two hands
frame = function(time) {
  par(mar=c(0,0,0,0))
  nplot(xlim=c(-1.1,1.1),ylim=c(-1.1,1.1),pty='s')
  plotrix::draw.circle(0,0,1,col='#aaaaff')
  radius = c(0.5,0.9)
  speed = 2*pi/c(720,60)
  lwd = c(4,2)
  graphics::arrows(0,0,radius*sin(speed*time),radius*cos(speed*time),lwd=lwd)
}

# Produce movie
## Not run:
makemovie(frame,seq(0,60,0.5),'~/testmovie','movie.mp4',200,200)

## End(Not run)
```

`mollweide` *Mollweide projection*

Description

Performs a Mollweide projection (also known as Babinet projection, homalographic projection, homolographic projection, and elliptical projection) of longitude and latitude coordinates. The most important feature of the Mollweide projection is that it preserves surface areas, which makes it a commonly used projection in geography, astronomy and cosmology. The total surface area of the standard projection is equal to the surface area of the unit sphere (4pi); and the shape of the fully projected sphere is an ellipse (with axes lengths 2*sqrt(2) and sqrt(2)).

Usage

```
mollweide(lon, lat, lon0 = 0, radius = 1, deg = FALSE)
```

Arguments

`lon`	longitude or vector of longitudes in radians (unless deg=TRUE)
`lat`	latitude or vector of latitudes in radians (unless deg=TRUE), must lie between -pi/2 and +pi/2
`lon0`	latitude of null meridian, which will be projected on to x=0
`radius`	radius of spherical projection, such that the surface area of the projection equals 4piR^2
`deg`	logical flag; if set to TRUE, the input arguments `lon`, `lat`, `lon0` are assumed to be in degrees (otherwise in radians)

Value

Returns a data frame of 2D Cartesian coordinates `x` and `y`.

Author(s)

Danail Obreschkow

Examples

```
lon = runif(1e4,0,2*pi)
lat = asin(runif(1e4,-1,1)) # = uniform sampling of the sphere
plot(mollweide(lon,lat),xlim=c(-3,3),ylim=c(-1.5,1.5),pch=16,cex=0.5)
plotrix::draw.ellipse(0,0,2*sqrt(2),sqrt(2),border='orange',lwd=2)
```

`nplot` *Make empty plot area*

Description

Open an empty plot

Usage

```
nplot(
  xlim = c(0, 1),
  ylim = c(0, 1),
  xlab = "",
  ylab = "",
  xaxs = "i",
  yaxs = "i",
  xaxt = "n",
  yaxt = "n",
  bty = "n",
  ...
)
```

Arguments

`xlim, ylim`	vectors with plotting limits.
`xlab, ylab`	horizontal and vertical labels.
`xaxs, yaxs`	style of the axis interval (see `par`).
`xaxt, yaxt`	character which specifies the x axis type (see `par`).
`bty`	character specifying the border type (see `par`).
`...`	additional arguments used by `plot`

Value

None

Author(s)

Danail Obreschkow

`pdf2jpg` *Convert pdf to jpg*

Description

Calls the console "convert" function to convert a pdf-file into a jpeg-image. Requires "convert" to be installed already.

Usage

```
pdf2jpg(
  pdf.filename,
  jpg.filename,
  quality = 100,
  background = "white",
  dim = c(1600, 1200),
  remove.pdf = FALSE,
  verbose = TRUE
)
```

Arguments

`pdf.filename`	filename of pdf input file
`jpg.filename`	filename of jpeg output file
`quality`	quality of jpeg compression from 1 (worst) to 100 (best)
`background`	color of background
`dim`	size in pixels of the jpeg-image
`remove.pdf`	logical flag. If TRUE, the pdf file is deleted after the conversion.
`verbose`	logical flag to turn on/off console statements.

Value

None

Author(s)

Danail Obreschkow

`planckcolors` *Planck CMB colour palette*

Description

Generates color scale matching the one normally used to display the Planck CMB temperature map from -300uK to +300uK.

Usage

```
planckcolors(n)
```

Arguments

`n`	integer number of colors in the scale

Value

Returns an n-vector of RGB colour strings.

Author(s)

Danail Obreschkow

#' @examples nplot() rasterImage(rbind(planck.colors(1e3)),0,0,1,1)

`rasterflip` *Flip array to be displayed with rasterImage()*

Description

Flips the array A to be displayed with rasterImage, such that the first index runs from left to right and second index runs from bottom to top, like in standard Cartesian coordinates. In this way `rasterImage(rasterflip(A))` has the same orientation as `image(A)`.

Usage

```
rasterflip(A)
```

Arguments

`A`	n-by-m array of a monochromatic image or n-by-m-by-k array of a color image (where k is 3 or 4)

Author(s)

Danail Obreschkow

smoothcontour *Draw smoothed contours*

Description

Draw smoothed iso-countours for a density field. The contours are computed using the `contourLines` routine and smoothed using the `smooth.spline` function. Both open and closed contour lines are handled correctly.

Usage

```
smoothcontour(
  x = seq(0, 1, length.out = nrow(z)),
  y = seq(0, 1, length.out = ncol(z)),
  z,
  levels,
  smoothing = 0.5,
  min.radius = 1,
  lwd = 1,
  lty = 1,
  col = "black",
  ...
)
```

Arguments

`x, y`	vectors containing the locations of grid lines at which the values of z are measured. These must be in ascending order. By default, equally spaced values from 0 to 1 are used.
`z`	matrix representing the density field on which the contours are plotted.
`levels`	vector of the iso-contour levels.
`smoothing`	value between 0 and 1 specifying the degree of smoothing.
`min.radius`	numerical value. If larger than 0, all contours with a mean radius (in pixels) below `min.radius` are removed.
`lwd`	vector of line widths (see `par`)
`lty`	vector of line types (see `par`)
`col`	vector of colors (see `par`)
`...`	additional parameters to be passed to the function `lines`.

Value

None

Author(s)

Danail Obreschkow

See Also

contourLines, smooth.spline

Examples

```
set.seed(1)
f = function(x) cos(2*x[1]-x[2]-1)^2*exp(-x[1]^2-x[2]^2-x[1]*x[2])
x = seq(-3,3,length=100)
m = pracma::meshgrid(x)
z = array(Vectorize(function(x,y) f(c(x,y)))(m$Y,m$X)+rnorm(4e4,sd=0.1),dim(m$X))
image(x,x,z,col=terrain.colors(100))
contour(x,x,z,levels=c(0.2,0.5),add=TRUE)
smoothcontour(x,x,z,levels=c(0.2,0.5),lwd=3,smoothing=0.8,min.radius=2)
```

spectrumcolors *Spectrum colour palette*

Description

Generates smooth rainbow color scale from red to purple, similar to rainbow, but with improved smoothness

Usage

```
spectrumcolors(n, alpha = 1, rev = FALSE)
```

Arguments

n	integer number of colors in the scale
alpha	alpha transparency value (0=fully transparent, 1=fully opaque)
rev	logical flag indicating whether the ordering of the colors should be reversed

Value

Returns an n-vector of RGB colour strings.

Author(s)

Danail Obreschkow

#' @examples nplot() rasterImage(rbind(spectrumcolors(1e3)),0,0,1,0.5) rasterImage(rbind(rainbow(1e3,end=5/6)),0,0.5,1,1
text(0.5,0.25,'spectrum') text(0.5,0.75,'rainbow') abline(h=0.5)

sphereplot *Plot a spherical function or point set*

Description

Plots a spherical function or a point set in a 2D projection using only standard R graphics. This avoids compatibility issues of rgl, e.g. knitting markdown documents.

Usage

```
sphereplot(
  f,
  n = 100,
  theta0 = pi/2,
  phi0 = 0,
  angle = 0,
  projection = "globe",
  col = gray.colors(256, 0, 1),
  clim = NULL,
  add = FALSE,
  center = c(0, 0),
  radius = 1,
  nv = 500,
  show.border = TRUE,
  show.grid = TRUE,
  grid.phi = seq(0, 330, 30)/180 * pi,
  grid.theta = seq(30, 150, 30)/180 * pi,
  pch = 16,
  pt.col = "black",
  pt.cex = 0.5,
  lwd = 0.5,
  lty = 1,
  line.col = "black",
  background = "white",
  ...
)
```

Arguments

`f`	must be either of: (1) NULL to plot just grid without spherical function (2) a vectorized real function f(theta,phi) of the polar angle theta [0,pi] and azimuth angle [0,2pi] (3) an n-by-2 array of values theta and phi
`n`	number of grid cells in each dimension used in the plot
`theta0`	polar angle in radians at the center of the projection

`phi0`	azimuth angle in radians at the center of the projection
`angle`	angle in radians between vertical axis and central longitudinal great circle
`projection`	type of projection: "globe" (default), "cylindrical", "mollweide"
`col`	color map
`clim`	2-element vector specifying the values of f corresponding to the first and last color in col
`add`	logical flag specifying whether the sphere is to be added to an existing plot
`center`	center of the sphere on the plot
`radius`	radius of the sphere on the plot
`nv`	number or vertices used for grid lines and border
`show.border`	logical flag specifying whether to show a circular border around the sphere
`show.grid`	logical flag specifying whether to show grid lines
`grid.phi`	vector of phi-values of the longitudinal grid lines
`grid.theta`	vector of theta-values of the latitudinal grid lines
`pch`	point type
`pt.col`	point color
`pt.cex`	point size
`lwd`	line width of grid lines and border
`lty`	line type of grid lines and border
`line.col`	color of grid lines and border
`background`	background color
`...`	additional arguments to be passed to the function f

Value

Returns a list containing the vector `col` of colors and 2-vector `clim` of values corresponding to the first and last color.

Author(s)

Danail Obreschkow

Examples

```
## Plot random points on the unit sphere in Mollweide projection
set.seed(1)
f = cbind(acos(runif(5000,-1,1)),runif(5000,0,2*pi))
sphereplot(f,theta0=pi/3,projection='mollweide',pt.col='red')

## Plot a spherical function with color bar
nplot(xlim=c(0,1.2), asp=1)
f = function(theta,phi) cos(10*theta+phi)
sp = sphereplot(f, 200, col=planckcolors(200), phi0=0.1, theta0=pi/3,
add=TRUE, center=c(0.5,0.5), radius=0.4, clim=c(-1,1))
colorbar(1,0.1,1.1,0.9,col=sp$col,clim=sp$clim)
```

subplot *Insert a sub-panel into plot*

Description

Insert a sub-panel into an existing plotting area. To open a subplot, call `par(subplot(...))`, to close it, you must call `par(subplot('off'))`.

Usage

```
subplot(xleft = NA, ybottom, xright, ytop)
```

Arguments

`xleft`	lower x-coordinate of the sub-panel relative to the current plot.
`ybottom`	lower y-coordinate of the sub-panel relative to the current plot.
`xright`	upper x-coordinate of the sub-panel relative to the current plot.
`ytop`	upper y-coordinate of the sub-panel relative to the current plot.

Value

graphical parameters to user with `par`.

Author(s)

Danail Obreschkow

Examples

```
# main plot
f = function(x) x*sin(1/(x+.Machine$double.eps))
curve(f,0,1,n=1000,ylim=c(-1,1),xlab='',ylab='',xaxs='i',yaxs='i')
rect(0.02,-0.1,0.1,0.1,border='blue',col=transparent('blue',0.2))

# subplot
par(subplot(0.55,-0.8,0.93,0))
curve(f,0.02,0.1,n=500,ylim=c(-0.1,0.1),xlab='',ylab='',xaxs='i',yaxs='i')
rect(0.02,-0.1,0.1,0.1,border=NA,col= transparent('blue',0.2))
par(subplot('off'))
```

`transparent` *Add transparency to a color*

Description

Add transparency to a color

Usage

```
transparent(col, alpha = 0.5)
```

Arguments

`col`	is a color or vector/array of colors, specified as text (e.g. 'purple') or 7/9-character (e.g. '#A020F0')
`alpha`	transparency value between 0 and 1 or a vector/array thereof

Value

Returns a 9-character color or vector/array of 9-character colors.

Author(s)

Danail Obreschkow

See Also

`lightness`

Examples

```
# Add different transparencies of the same color
plot(runif(50),runif(50),pch=20,cex=10,col=transparent('purple',runif(50)))

# Add the same transparency to different colors
plot(runif(50),runif(50),pch=20,cex=10,col=transparent(rainbow(50)))
```

`transzoom` *Zoom, translate and rotate array image*

Description

Zoom/rotate/translate an image relative to its center for images represented as simple arrays.

Usage

```
transzoom(
  img = NULL,
  zoom = 1,
  shift = c(0, 0),
  angle = 0,
  size = NULL,
  col = "black",
  filter = "bilinear",
  file.in = "",
  file.out = "",
  format = "png",
  show.image = TRUE
)
```

Arguments

`img`	n-by-m-by-3 array or n-by-m-by-4 array representing an rgb(+alpha) image.
`zoom`	zoom factor (>0).
`shift`	2-vector specifying the vertical+horizontal translation in units of output pixels (i.e. after zooming).
`angle`	rotation angle in degrees.
`size`	2-vector specifying the vertical+horizontal dimensions of the output image. If not given, this is taken to be identical to the input image.
`col`	background color
`filter`	affine transformation filter; either 'none' or 'bilinear'
`file.in`	optional input filename, which can be used to load an image instead of providing it via `img`. This filename is ignored if `img` is specified.
`file.out`	optional output filename.
`format`	one of "png" or "jpg" specifying the file format of the input and output image.
`show.image`	logical flag specifying whether the image is displayed in the R console.

Value

Returns an n-by-m-by-3 array or n-by-m-by-4 array of the processed image.

Author(s)

Danail Obreschkow

Examples

```
img = yinyangyong # this is an example image included in the package
transzoom(img, zoom=2) # zoom by a factor 2
```

---

wavelength2col *Convert wavelength to RGB*

Description

Converts a given wavelength of light to an approximate RGB color value, using black in the invisible range.

Usage

```
wavelength2col(wavelength)
```

Arguments

`wavelength`	wavelength value (or vector), in nanometers.

Value

Returns a color string or vector of color strings with the same number of elements as `wavelength`.

Author(s)

Danail Obreschkow

Source

Smoothed implementation of the original Fortran version by Dan Bruton (http://www.physics.sfasu.edu/astro/color/spectra.ht
and the R-function by Michael Friendly (https://gist.github.com/friendly).

Examples

```
lambda = seq(300,800)
col = matrix(wavelength2col(lambda),nrow=1)
plot(NA,xlim=range(lambda),ylim=c(0,1),xaxs='i',xlab='wavelength [nm]',yaxs='i',yaxt='n',ylab='')
rasterImage(col,min(lambda),0,max(lambda),1)
```

`yinyangyong` *Yin-Yang-Yong image*

Description

A data set containing a 500-by-500-by-4 array, representing an image of a Yin-Yang-Yong symbol with transparency. This symbol was 'reinvented' many times. The particular version in this package is the one drawn by the author D. Obreschkow in 1999.

Usage

```
yinyangyong
```

Format

An object of class `array` of dimension 500 x 500 x 4.

Value

None

Author(s)

Danail Obreschkow

Examples

```
nplot(pty='s')
rasterImage(yinyangyong,0,0,1,1)
```

Index