Package ‘palr’

January 30, 2020

Type Package
Title Colour Palettes for Data
LazyData yes
Version 0.2.0
Description Colour palettes for data, based on some well known public data sets. Includes helper functions to map absolute values to known palettes, and capture the work of image colour mapping as raster data sets.

Depends R (>= 2.10)
Imports grDevices
Suggests knitr, rmarkdown, raster, testthat, covr, stars, viridis
VignetteBuilder knitr
License GPL-3
RoxygenNote 7.0.2
URL https://github.com/AustralianAntarcticDivision/palr

R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>bathy_deep_pal</td>
<td>2</td>
</tr>
<tr>
<td>chl_pal</td>
<td>3</td>
</tr>
</tbody>
</table>
bathy_deep_pal

col2hex ................................................. 4
ice_pal ................................................. 4
image_pal ............................................... 5
mk_timePal ............................................. 7
oisst ..................................................... 8
palr ....................................................... 8
sst_pal .................................................... 8

Index 10

bathy_deep_pal Bathymetry

Description

Deep bathymetry colours.

Usage

bathy_deep_pal(x, palette = FALSE, alpha = 1, ...)
bathyDeepPal(x, palette = FALSE, alpha = 1, ...)

Arguments

x a vector of data values or a single num (n)
palette logical, if TRUE return a list with matching colours and values
alpha value in 0,1 to specify opacity
... currently ignored

Details

Colour ramp suitable for deep waters (-5500) to sea level. The palette functions operate in 3 modes:
1) n colours - Pal(6) - returns 6 colours from the palette 2) data - Pal(c(10, 50, 100)) - return colours
for 3 ice concentrations 3) palette - Pal(palette = TRUE) - return the full palette and breaks Derived
from maps created in Matlab by Emmanuel Laurenceau.

Value

colours, palette, or function, see Details

Examples

plot(1:15, pch = 19, cex = 4, col = bathy_deep_pal(15))
Description

Ocean colour palette for chlorophyll-a.

Usage

```r
chl_pal(x, palette = FALSE, alpha = 1)
```

```r
chlPal(x, palette = FALSE, alpha = 1, ...)
```

Arguments

- `x`: a vector of data values or a single number
- `palette`: logical, if TRUE return a list with matching colours and values
- `alpha`: value in 0,1 to specify opacity
- `...`: currently unused

Details

Flexible control of the chlorophyll-a palette. If `x` is a single number, the function returns that many colours evenly spaced from the palette. If `x` is a vector of multiple values the palette is queried for colours matching those values, and these are returned. If `x` is missing and `palette` is FALSE then a function is returned that will generate n evenly spaced colours from the palette, as per `colorRampPalette`.

Value

colours, palette, or function, see Details

References

Derived from a file once found at 'http://oceancolor.gsfc.nasa.gov/DOCS/palette_chl_etc.txt'

Examples

```r
## Not run:
chl <- raadtools::readchla(xylim = c(100, 110, -50, -40))
## just get a small number of evenly space colours
plot(chl, col = chl_pal(10))
## store the full palette and work with values and colours
pal <- chl_pal()
## the standard full palette
plot(chl, breaks = pal$breaks, col = pal$cols)
## a custom set of values with matching colours
plot(chl, col = chl_pal(pal$breaks[seq(1, length(pal$breaks), length = 10)]))
```
## any number of colours stored as a function
myfun <- chl_pal()
plot(chl, col = myfun(18))
## just n colours
plot(chl, col = chl_pal(18))

## End(Not run)

---

### col2hex

*Colour to hex conversion.*

**Description**

Create colours from colour names in one easy step.

**Usage**

```r
col2hex(x, alpha = 1)
```

**Arguments**

- `x`: vector of colour names or hex strings
- `alpha`: optional transparency value in [0,1], can be per colour in `x`

**Value**

character string of hex colours

**Examples**

```r
col2hex(c("aliceblue", "firebrick"), alpha = c(1, .5))
col2hex(c("#FFFFFF", "#123456FF"), alpha = 0.1)
```

---

### ice_pal

*Sea ice colours*

**Description**

Colours for sea ice.

**Usage**

```r
ice_pal(x, palette = FALSE, alpha = 1, ...)
```

```r
icePal(x, palette = FALSE, alpha = 1, ...)
```
**image_pal**

**Map data values to colours**

**Description**

If no `col` is provided, the default image palette is used. The density can be controlled with `n` and the mapping with the optional `breaks`. If `breaks` is included as well as `n`, `n` is ignored.

**Usage**

```r
image_pal(x, col, ..., breaks = NULL, n = NULL, zlim = NULL)
image_raster(x, col, ..., breaks = NULL, n = NULL, zlim = NULL)
image_stars(x, col, ..., breaks = NULL, n = NULL, zlim = NULL)
```
Arguments

- `x`: numeric values, raster object (single layer only) or stars object (single variable, 2D array only).
- `col`: function to generate colours, or a vector of hex colours.
- `breaks`: optionally used to specify colour mapping.
- `n`: optionally used to specify density of colours from `col` (ignored if `breaks` is set).
- `zlim`: numeric range to clamp values to an absolute scale (ignored if `breaks` is set).

Details

The function `image_pal()` only returns hex character string colours. The function `image_raster()` will map a raster of numeric values to an RGB 3-layer (channel) raster brick, and `image_stars()` similarly for a 3-dimensional stars object.

Please note that the expansion to 3-channels is a fairly wasteful thing to do, the overall data is expanded from a single layer to three but this facilitates a specific task of creating textures for 3D mapping, and this is the only way to do it currently. It’s also useful in other situations, for controlling exactly the kind of plots we can achieve and for exporting to image formats such as ‘GeoTIFF’ or ‘PNG’.

Value

for `image_pal()` a vector of hex colours, for `image_raster` and `image_stars` a raster or stars object with 3 channel RGB (range 0,255)

Examples

```r
set.seed(28)
vals <- sort(rnorm(100))
cols <- image_pal(vals, zlim = c(-2.4, 2))
plot(vals, col = cols); abline(h = 2)
points(vals, pch = 19, cex = 0.1) ## zlim excluded some of the range
if (requireNamespace("raster", quietly = TRUE)) {
im <- image_raster(volcano)
library(raster)
plotRGB(im)

vv <- unique(quantile(volcano, seq(0, 1, length = 12)))
plotRGB(image_raster(volcano, breaks = vv))
plotRGB(image_raster(volcano, breaks = vv[-c(4, 6)], col = gray.colors(9)))
plotRGB(image_raster(volcano, n = 4))
plotRGB(image_raster(volcano, col = grey(seq(0.2, 0.8, by = 0.1))))

plotRGB(image_raster(volcano, col = viridis::magma(24)))
}
if (!requireNamespace("stars", quietly = TRUE)) {
```

library(stars)
x <- st_as_stars(volcano)
plot(image_stars(x), rgb = 1:3)
plot(image_stars(x, col = gray.colors), rgb = 1:3)
plot(image_stars(x))
plot(image_stars(x, col = rainbow, breaks = c(94, 100, 120, 150, 195)), rgb = 1:3)
}

mk_timePal

Time-indexed colour.

Description
Create a time-indexed colour map, useful for maintaining an absolute scale across time series as a function of date-time.

Usage

mk_timePal(x, col)

Arguments

x
date-times
col
colours, can be a function or an actual set of colours

Value

function of date-time

Examples

dts <- seq(as.Date("1749-01-01"), by = "1 month", length.out = length(sunspots))
d <- data.frame(date = dts, sunspots = as.vector(t(sunspots)))
tpal <- mk_timePal(d$date, col = sst_pal(50))
par(mfrow = c(2, 1))
plot(sunspots ~ date, col = tpal(date), data = d)
## colours maintained by absolute date
plot(sunspots ~ date, col = tpal(date), data = d[1500:1800, ], cex = 2)
## we can now insert new points and maintain this colour ramp
d2 <- data.frame(date = seq(min(d$date), max(d$date), by = "5 days"))
d2$sunspots <- approxfun(d$date, d$sunspots)(d2$date)
points(sunspots ~ date, col = tpal(date), data = d2, pch = 19, cex = 0.5)
**oisst**

*Sea surface temperature (SST)*.

**Description**

SST example raster data set, at 0.25 degree resolution for global coverage in "longitude180/latitude".

**Details**

Created using script in data-raw/ using ‘raadtools’ package.

**References**


**Examples**

```r
dim(oisst)
class(oisst)
image(oisst, useRaster = TRUE)
```

**palr**

*palr: colours for data*

**Description**

palr: colours for data

**sst_pal**

*SST colours*

**Description**

SST colours

**Usage**

```r
ssst_pal(x, palette = FALSE, alpha = 1, ...)
ssstPal(x, palette = FALSE, alpha = 1, ...)
```
**Arguments**

- **x**: a vector of data values or a single number
- **palette**: logical, if TRUE return a list with matching colours and values
- **alpha**: value in 0,1 to specify opacity
- **...**: currently ignored

**Value**

colours, palette, or function, see Details

**References**

Derived from a file once found at 'http://oceancolor.gsfc.nasa.gov/DOCS/palette_sst.txt'

**Examples**

data(oisst)
sstcols <- sst_pal(palette = TRUE)
image(oisst, col = sstcols$col, zlim = range(sstcols$breaks))
Index

bathy_deep_pal, 2
bathyDeepPal(bathy_deep_pal), 2

chl_pal, 3
chlPal(chl_pal), 3
col2hex, 4
colorRampPalette, 3

ice_pal, 4
icePal(ice_pal), 4
image_pal, 5
image_raster(image_pal), 5
image_stars(image_pal), 5

mk_timePal, 7

oisst, 8

palr, 8

sst_pal, 8
sstPal(sst_pal), 8