Package 'paletteknife'

October 15, 2021

Title Create Colour Scales and Legend from Continuous or Categorical Vectors

Version 0.4.1

Description Streamlines the steps for adding colour scales and associated legends when working with base R graphics, especially for interactive use. Popular palettes are included and pretty legends produced when mapping a large variety of vector classes to a colour scale. An additional helper for adding axes and grid lines complements the base::plot() work flow.

License MIT + file LICENSE

URL https://github.com/johnxhobbs/paletteknife

BugReports https://github.com/johnxhobbs/paletteknife/issues

Encoding UTF-8

RoxygenNote 7.1.1

Depends graphics, grDevices

NeedsCompilation no

Author John Hobbs [aut, cre]

Maintainer John Hobbs <johnxhobbs@gmail.com>

Repository CRAN

Date/Publication 2021-10-15 07:40:05 UTC

R topics documented:

`autoaxis` *Auto Axis Tool*

Description

Overlay base plot axis with custom intervals

Usage

```
autoaxis(
  side,
  major = NA,
  major_grid = FALSE,
  minor = NA,
  minor_grid = FALSE,
  format = "%Y-%m-%d",
  spacing = TRUE,
  tck = -0.03,
  ...
)
```

Arguments

`side`	Side to add axis, 1 = bottom, 2 = left, 3 = top, 4 = right
`major`	Spacing of major axis ticks and labels (or approx. number of intervals if `spacing = FALSE`). If the axis is date or time, use a interval specified in `?seq.POSIXt`, such as 'sec' or 'week'
`major_grid`	Add grid lines corresponding to major axis ticks, `TRUE` to get default translucent black, otherwise colour (name or hex)
`minor`	Spacing (or number) of minor ticks (note, no label for minor). If given as a character string, it will pass to `seq.POSIXt`
`minor_grid`	Add gridlines for minor ticks, `TRUE` uses transparent black, otherwise colour string
`format`	Date or time format for major axis – `major` must be a character string in this case
`spacing`	Should `major` and `minor` be interpreted as tick spacing (default) or approximate number of ticks
`tck`	Size of axis tick: minor axis will always take half this value
`...`	Additional arguemnts passed to `axis()`, for example `las=2` for perpendicular labels

Details

Major and minor tick marks can be specified in a number of ways:

- As a character string if the axis is datetime, such as 'year' or 'hour' which are passed to `seq()`. These can be prefixed with an integer multiplier, for example '6 hour', '30-sec', or '10year'. Any non-alphanumeric separator can be used, or none.
- As a tick interval using the default `spacing = TRUE`
- As an approximate number of tick marks to include, using `pretty()` to find the best interval, using `spacing = FALSE`

Major adds labels and ticks, minor is just half-sized ticks marks. Both tick sizes can be changed (or direction changed) using `tck`.

Three different datetime axis are possible: year, day-offset, seconds-offset. Use `format` to specify how the label should appear, such as '%b %Y' (see `?strptime`)

- Year should be treated as a conventional numeric axis, use `major=1/12` not `major='month'`
- day-offset is an axis of `class(x)=='Date'` and is identified if the axis range exists within +/-9e4, meaning within dates 1723 - 2216
- second-offset is an axis of `class(x)=='POSIXct'` and is identified by a range outside of +/-9e4. This will give very strange results if your entire POSIXct axis is within 24 hours of 1970-01-01

A grid can be added at the same time by setting `major_grid` or `minor_grid` to `TRUE` or a colour string. If `TRUE`, a transparent black is used by default.

Any other options can be passed through to `axis()` directly (see `?axis`), most notably `las = 2` to rotate the labels, and `cex.axis` for label size.

This does NOT work well for `barplot()` categorical axis, for this continue to use the basic `axis()` function with custom labels, see examples.

Value

No return value (`NULL`)

Examples

```
plot(sunspots) # This time series is actually given in decimal years
  autoaxis(side=3, major=50, major_grid='coral', minor=10, minor_grid=TRUE, spacing=TRUE)
  autoaxis(side=4, major=11, minor=25, spacing=FALSE, las=2, cex.axis=0.5, tck=0.02)

plot(seq(as.POSIXct('2020-01-01'),as.POSIXct('2020-01-03'),length.out=1e3),
    rnorm(1e3), xlab='POSIXct', xaxt='n')
  autoaxis(side=1, major='day', minor='hour', format='%A')
  autoaxis(side=3, major='6 hour', format='%H:%M')

plot(seq(as.Date('2013-02-01'),as.Date('2020-01-03'),length.out=1e3),
    rnorm(1e3), xlab='Date', xaxt='n')
  autoaxis(side=1, major='year', minor='quarter', format='%Y')
  autoaxis(side=3, minor='3month', minor_grid=TRUE)
```

```
# For barplot() use base functions - remember to set width=1, space=0
# otherwise bars will not be plotted on integer x-coordinates
barplot(mtcars$mpg, width=1, space=0, ylab='mpg')
  # Adjust the x-axis down by 0.5 so that the tick is in centre of each bar
  axis(side=1, at=-0.5+1:length(mtcars$mpg), labels=rownames(mtcars), las=2 )
  # Often prettier, label each bar inside the bar itself using text()
  text(x=-1+1:length(mtcars$mpg), y=1, pos=4,
    labels=rownames(mtcars), srt=90, cex=0.7)
  # autoaxis can still be used for adjusting the numeric scale
  autoaxis(side=2, major=5, major_grid=TRUE, minor=1, minor_grid=TRUE)
```

`autocol` *Map Colours From Value*

Description

Create a vector of colours and associated legend for easier base plots

Usage

```
autocol(
  x,
  set = "",
  alpha = NA,
  limits = NA,
  na_colour = NA,
  bias = 1,
  legend_len = 6
)

pals.misc

pals.viridis

pals.rcolorbrewer
```

Arguments

`x`	Vector to be mapped to colours
`set`	Colour set to use – see Details for full list. A default `sasha` or `viridis` is chosen if empty.
`alpha`	Transparency as a single value or as another vector (recycled to fill). If it is a vector, all values are scaled from 0:max(alpha) meaning transparent:opaque. Single values must be in range 0-1. If `NA` no alpha channel is added.
`limits`	Colour scale limits as absolute range `c(0,10)` or `NA` = full range

na_colour	Colour to represent NA-values, default NA returns a colour of NA (thus not plotted)
bias	Skew to apply to colour-ramp (>1 increases resolution at low end, <1 at the high end)
legend_len	Continuous legend target size

Format

An object of class list of length 1.

An object of class list of length 8.

An object of class list of length 35.

Details

Helper function for using colours in R's default plot() and legend(). Colours from built-in palettes are automatically scaled to return a vector of colours and create options('autolegend') which contains the correct legend mapping for autolegend().

A discrete palette is used for factor and character inputs whilst a continuous palette is used for integer and numeric.

Colour sets built-in so far are held in lists starting pals. and can be visualized most easily with pals_display(). The set argument can be any of the colour set names listed here (such as 'magma'), or from palette.pals(), or finally as a custom-defined vector, such as set = rainbow(5).

The current lists of palettes included with paletteknife all being with pal.

- pals.viridis

  All of the continuous palette forked from the viridisLite package maintained by Simon Garnier.

  - Contains: cividis inferno magma mako plasma rocket turbo viridis

- pals.rcolorbrewer

  All of the palettes included in RColorBrewer

  - Categorical: Accent Set1 Set2 Set3 Paired Pastel1 Pastel2 Dark2
  - Continuous: Greys Blues BuGn BuPu Greens GnBu PuBu Purples PuBuGn YlGnBu YlGn YlOrBr YlOrRd Oranges OrRd Reds RdPu PuRd
  - Divergent: Spectral RdYlBu RdYlGn BrBG RdBu RdGy PiYG PRGn PuOr

- pals.misc

  - Sasha Trubetskoy (2017): *List of 20 Simple, Distinct Colors*: sasha

Custom limits can be specified using c(0,10). This is useful if multiple plots using the same range are required for cross-comparison. Default behaviour (limits = NA) sets the range to exactly fit.

The skew of the colourscale can be adjusted with bias, for example if x has an exponential distribution, a bias value > 1 will bring out contrast at the low end.

Value

A character vector of colours of equal length to input `x`, sampled from the chosen `set`. This allows it to be used for plotting directly. Information for a legend (containing every level for categorical data, or approximately length `legend_len` for continuous) is stored in `options('autolegend')` and not returned explicitly.

Examples

```
plot(iris$Sepal.Length, iris$Petal.Length, cex=3, pch=16,
    col=autocol(iris$Petal.Width, set='PuBuGn', alpha=0.8, legend_len=12) )
  autolegend('topleft', title='Petal.Width', ncol=3)
  # Also try simplest "autolegend()" for click-to-draw

# Try scales which include NA in both colour and alpha channel
with(airquality, plot(Temp, col=autocol(x=Solar.R, set='YlOrRd', alpha=Ozone,
    na_colour='cyan'), pch=16, cex=sqrt(Wind) ))
  # Note inset=1 draws on opposite side ie above not below plot area
  autolegend('bottom', inset=1, bty='n', horiz=TRUE)


# Here we want a summary plot ordered by level, so need to create a colour vector to match
# 'Alphabet' is a built-in colour set, see "palette.pals()"
mixedbag = as.factor(sample(letters,1000,replace=TRUE))
  plot(x=mixedbag, y=rnorm(1000), col=autocol(levels(mixedbag), set='Alphabet'))
  autolegend('bottom', ncol=9, cex=0.7)

# Maintain the order of strings
barplot(1:8, col=autocol(LETTERS[8:1]))
  autolegend('topleft')

# Any unusual formats are coerced to numeric and the legend converted back
mydates = as.Date('2000-01-01')+0:100
  plot(mydates, pch=16, col=autocol(mydates, set=rainbow(10), bias=2) )
  autolegend(x=0, y=mydates[100], title='My Dates')

# Timeseries objects plot as a line, but can overlay with points()
plot(airmiles)
  points(airmiles, pch=15, col=autocol(airmiles, set='Reds'))

# Use the limits to clip or augment the colour-scale
layout(matrix(1:2))
  plot(runif(10), col=autocol(1:10, limits=c(0,20)), pch=16,
    main='Data split over two plots with same scale')
  plot(runif(10), col=autocol(c(100,20:12), limits=c(0,20)), pch=16)
  text(1, 0.5, pos=4, xpd=NA,
'This point has a
value of 100 but
clipped to max
colour == 20')
  autolegend('bottom', inset=1, horiz=TRUE) # Draws above!
  layout(1)
```

`autolegend` *Add Auto-Generated Legend*

Description

Add a legend for the last `autocol()` set generated

Usage

```
autolegend(...)
```

Arguments

`...`	Arguments passed directly to `legend` – legend text and colours are taken automatically from `options('autolegend')`. See examples for useful parameters, including `pch` and `pt.cex`

Details

If no location (such as 'bottom', or an x,y coordinate) is given, then it calls the `locator()` crosshairs so the position of the legend can be picked interactively. All arguments are passed to `legend()`, see `?legend` for a full list.

Legend labels and fill are generated by either `autopal()` or `autocol()` and stored in the global `options('autolegend')` where they can be manipulated if needs be.

See more examples in ?autocol for a `plot()` and `autolegend()` work flow.

Value

No return value (`NULL`)

Examples

```
# Simplest version: click-to-draw with locator()
plot(1:10, pch=16, col=autocol(1:10, 'Blues', legend_len=5))
# autolegend() # Try me! And click on plot to add legend

# Other neat versions -- note ?legend
autolegend(x=6, y=4, ncol=2, title='Draw at (6,4)')
autolegend('topleft', title='"topleft"', ncol=2, bty='n')
autolegend('bottom', inset=1, horiz=TRUE, bty='n', title='Above plot')

# Use pch (and optionally pt.cex) in legend -- these get recycled
autolegend('bottom', horiz=TRUE, pch=16, pt.cex=3, title='pch=16, pt.cex=3')
autolegend('right', pch=1:10, pt.cex=2, title='pch=1:10')

# Manipulate the legend text, for example with format(), this is a bit long-winded!
heatmap(as.matrix(eurodist), col=autopal('turbo', limits=range(eurodist)) )
current_legend = options('autolegend')[[1]]
```

```
options(autolegend = list(format(current_legend[[1]], big.mark=','), current_legend[[2]]))
autolegend('bottom', inset=1, horiz=TRUE, title='Misleading miles between cities')

# No helper exists yet for creating size or shape legends -- follow this idea...
with(airquality, plot(Temp, pch=16, cex=Solar.R/100, col=autocol(Ozone, set='Reds')))
cex_legend = pretty(airquality$Solar.R)
legend('bottom', pt.cex=cex_legend/100, legend=cex_legend, pch=1,
  horiz=TRUE, title='Solar.R', bty='n' )
autolegend('bottom', inset=1, horiz=TRUE, title='Ozone', bty='n')
```

`autopal` *Auto-Palette*

Description

Return a palette vector from one of the built-in sets

Usage

```
autopal(set = "", n = 30, limits = NA, bias = 1, legend_len = 6)
```

Arguments

`set`	Colour set to use – see ?autocol for full list. A default `sasha` or `viridis` is chosen if empty.
`n`	Length of colour vector to return, must be at least 2
`limits`	Colour scale limits to pass to legend eg `c(0,10)` – if left as `NA` no autolegend will be generated
`bias`	Skew to apply to colour-ramp (>1 increases resolution at low end, <1 at the high end)
`legend_len`	Continuous legend target size

Details

This can be used where a palette is provided rather than a mapped colour vector, for example `image()`. The limits must be specified for `autolegend()` information to be updated. Custom colour limits can be set using `breaks` or `levels` (see examples) if the same colour range is needed across several plots.

See ?autocol for list of all available colour sets.

Value

A character vector of colours of length `n` giving a continuous colour palette sampled from `set`. If `limits` are specified, information for a colour legend is produced of approximate length `legend_len`. This is stored in `options('autolegend')` and not returned explicitly.

Examples

```
image(volcano, col=autopal('RdYlGn', n=100, limits=c(50,200), bias=1.5),
    breaks=seq(50,200,length.out=101) )
  autolegend('bottom', inset=1, ncol=5)

# Or using the slightly smarter filled.contour
filled.contour(volcano, col=autopal('RdYlGn', n=20, limits=c(100,150)),
  levels=seq(50,200,length.out=21) )
```

---

get_set *Helper to get a set by name - will search all included lists ("pals."), then built-in, or just returns itself if already a vector*

Description

Helper to get a set by name - will search all included lists ("pals."), then built-in, or just returns itself if already a vector

Usage

```
get_set(set = "", default = "turbo")
```

Arguments

set	Character string of single palette name or a vector of colours
default	What to use if an empty string is passed (the default case)

---

pals_display *Built-in Paletteknife Palettes*

Description

Plot a list of palettes for comparison

Usage

```
pals_display(pals = c(pals.misc, pals.rcolorbrewer, pals.viridis))
```

Arguments

pals	Named list of palettes (colour vectors)

Value

No return value (NULL)

Examples

```
pals_display(c(list(rainbow=rainbow(10), default=palette()),
               pals.misc, pals.rcolorbrewer[c('Paired','Set1','Set2')] ))

pals_display(list(rainbow=rainbow(45)[30:1], turbo=pals.viridis$turbo ))

pals_display(lapply(setNames(palette.pals(),palette.pals()), palette.colors, n=NULL))

# Bit of fun ordering a list of palettes (MUST be same palette size)
mat_cols = do.call(rbind, lapply(pals.rcolorbrewer[9:26],
                           function(hex) as.vector(rgb2hsv(col2rgb(hex)))))
pals_display(pals.rcolorbrewer[9:26][hclust(dist(mat_cols))$order])
```

Index