Package: RcppColors (via r-universe)

October 28, 2026

Type Package

Title Color Mappings and 'C++' Header Files for Color Conversion

Version 0.6.0

Maintainer Stéphane Laurent <laurent_step@outlook.fr>

Description Provides 'C++' header files to deal with color conversion from some color spaces to hexadecimal with 'Rcpp', and exports some color mapping functions for usage in R. Also exports functions to convert colors from the 'HSLuv' color space for usage in R. 'HSLuv' is a human-friendly alternative to HSL.

License GPL-3

URL https://github.com/stla/RcppColors

BugReports https://github.com/stla/RcppColors/issues

Imports Rcpp (>= 1.0.8)

LinkingTo Rcpp

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation yes

Author Stéphane Laurent [cre, aut], Scott Spencer [aut]

Repository https://cran.r-universe.dev

Date/Publication 2023-10-21 05:30:21 UTC

RemoteUrl https://github.com/cran/RcppColors

RemoteRef HEAD

RemoteSha 39e654ba04ed923c9c89c289c7ed579c1b5fcac0

Contents

RcppColors-package 2
colorMap1 2
hsi 6

hsl 7
hsluv 8
permuteRGB 9

Index **11**

`RcppColors-package` *'C++' header files for conversion from some color spaces to hexadecimal.*

Description

This package is mainly intended to be used with 'Rcpp', but it also provides some R functions for color conversion and color mappings.

Details

See README for a description of the available 'C++' functions and how to use the package.

Author(s)

Stéphane Laurent.

Maintainer: Stéphane Laurent <laurent_step@outlook.fr>

`colorMap1` *Color mappings functions*

Description

Functions mapping each complex number to a color.

Usage

```
colorMap1(
  Z,
  bkgcolor = "#15191e",
  nancolor = "#000000",
  reverse = c(FALSE, FALSE, FALSE),
  nthreads = 1L
)

colorMap2(
  Z,
  bkgcolor = "#15191e",
  nancolor = "#000000",
  reverse = c(FALSE, FALSE, FALSE),
```

```
  nthreads = 1L
)

colorMap3(
  Z,
  bkgcolor = "#15191e",
  nancolor = "#000000",
  s = 80,
  n = 5,
  nthreads = 1L
)

colorMap4(
  Z,
  bkgcolor = "#15191e",
  nancolor = "#000000",
  reverse = c(FALSE, FALSE, FALSE),
  nthreads = 1L
)

colorMap5(
  Z,
  bkgcolor = "#15191e",
  nancolor = "#000000",
  reverse = c(FALSE, FALSE, FALSE),
  nthreads = 1L
)

colorMap6(
  Z,
  bkgcolor = "#15191e",
  nancolor = "#000000",
  reverse = c(FALSE, FALSE, FALSE),
  nthreads = 1L
)

colorMap7(
  Z,
  bkgcolor = "#15191e",
  nancolor = "#000000",
  reverse = c(FALSE, FALSE, FALSE),
  nthreads = 1L
)

colorMap8(
  Z,
  bkgcolor = "#15191e",
  nancolor = "#000000",
```

```
  reverse = c(FALSE, FALSE, FALSE),
  nthreads = 1L
)

colorMap9(
  Z,
  bkgcolor = "#15191e",
  nancolor = "#000000",
  reverse = c(FALSE, FALSE, FALSE),
  nthreads = 1L
)

colorMap10(
  Z,
  bkgcolor = "#15191e",
  nancolor = "#000000",
  reverse = c(FALSE, FALSE, FALSE),
  nthreads = 1L
)

colorMap11(
  Z,
  bkgcolor = "#15191e",
  nancolor = "#000000",
  reverse = c(FALSE, FALSE, FALSE),
  nthreads = 1L
)

colorMap12(
  Z,
  bkgcolor = "#15191e",
  nancolor = "#000000",
  reverse = c(FALSE, FALSE, FALSE),
  nthreads = 1L
)

colorMap13(
  Z,
  bkgcolor = "#15191e",
  nancolor = "#000000",
  reverse = c(FALSE, FALSE, FALSE),
  nthreads = 1L
)

colorMap14(
  Z,
  bkgcolor = "#15191e",
  nancolor = "#000000",
```

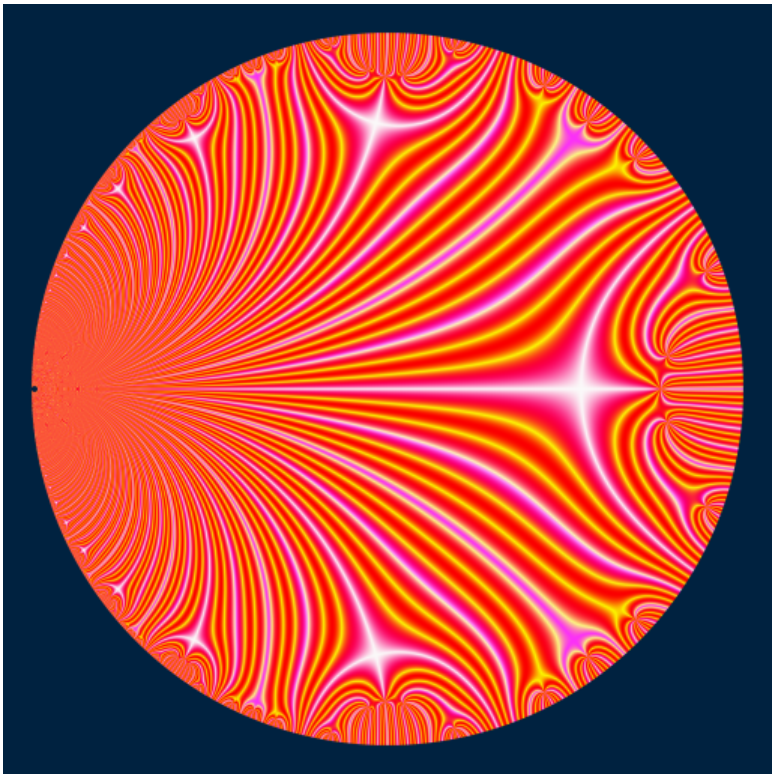
```
  reverse = c(FALSE, FALSE, FALSE),
  nthreads = 1L
)
```

Arguments

Z	complex number, vector or matrix
bkgcolor	background color; it is applied for the NA values of Z
nancolor	color for infinite and NaN values
reverse	logical vector of length three; for each component of the color space (e.g. R, G, B or H, S, L), whether to reverse it (e.g. R -> 255-R)
nthreads	number of threads used for parallel computation
s	saturation, a number between 0 and 100
n	number of rays drawn in a cycle; it should be a positive integer but any non-zero numeric value is accepted

Value

A string or a character vector or a character matrix, having the same size as Z. Each entry is a color given by a hexadecimal string.



Examples

```
library(RcppColors)

iota <- function(z){
  (z + 1i) / (1i*z + 1)
}
f <- function(z){
  q <- exp(2i * pi * z)
  r <- q - 4*q^2 + 2*q^3 + 8*q^4 - 5*q^5 - 8*q^6 + 6*q^7 - 23*q^9
  r / Mod(r)
}
g <- function(z){
  ifelse(
    Mod(z) >= 1,
    NA_complex_,
    f(iota(Conj(z)))
  )
}

x <- y <- seq(-1, 1, len = 1500)
W <- outer(y, x, function(x, y) complex(real = x, imaginary = y))
Z <- g(W)
image <- colorMap1(Z)

opar <- par(mar = c(0,0,0,0), bg = "#15191E")
plot(
  c(-100, 100), c(-100, 100), type = "n",
  xlab = "", ylab = "", axes = FALSE, asp = 1
)
rasterImage(image, -100, -100, 100, 100)
par(opar)
```

---

hsi *HSI color specification*

Description

Converts a color given in HSI coordinates to a hexadecimal string.

Usage

```
hsi(h = 360, s = 100, i = 100, alpha = NULL)
```

Arguments

h	the hue, a number between 0 and 360
s	the saturation, a number between 0 and 100
i	the intensity, a number between 0 and 100
alpha	opacity, a number between 0 and 1, or NULL

Value

The `hsi` function returns a hexadecimal string representing the corresponding color.

Examples

```
saturation <- 100
f <- Vectorize(
  function(x, y){
    z <- complex(real = x, imaginary = y)
    modulus <- Mod(z)
    if(modulus > 1){
      return("#ffffff")
    }
    radians <- Arg(z)
    if(radians < 0){
      radians <- radians + 2*pi
    }
    degrees <- 360 * radians / 2 / pi
    hsi(h = degrees, s = saturation, i = 100*modulus)
  }
)

x <- y <- seq(-1, 1, length.out = 200L)
image <- outer(x, y, f)

opar <- par(mar = c(0, 0, 0, 0))
plot(NULL, xlim = c(-1, 1), ylim = c(-1, 1), asp = 1)
rasterImage(image, -1, -1, 1, 1)
par(opar)
```

`hsl` *HSL color specification*

Description

Converts a color given in HSL coordinates to a hexadecimal string.

Usage

```
hsl(h = 360, s = 100, l = 100, alpha = NULL)
```

Arguments

`h`	the hue, a number between `0` and `360`
`s`	the saturation, a number between `0` and `100`
`l`	the lightness, a number between `0` and `100`
`alpha`	opacity, a number between `0` and `1`, or `NULL`

Value

The `hsl` function returns a hexadecimal string representing the corresponding color.

Examples

```
saturation <- 100
f <- Vectorize(
  function(x, y){
    z <- complex(real = x, imaginary = y)
    modulus <- Mod(z)
    if(modulus > 1){
      return("#ffffff")
    }
    radians <- Arg(z)
    if(radians < 0){
      radians <- radians + 2*pi
    }
    degrees <- 360 * radians / 2 / pi
    hsl(h = degrees, s = saturation, l = 100*modulus)
  }
)

x <- y <- seq(-1, 1, length.out = 200L)
image <- outer(x, y, f)

opar <- par(mar = c(0, 0, 0, 0))
plot(NULL, xlim = c(-1, 1), ylim = c(-1, 1), asp = 1)
rasterImage(image, -1, -1, 1, 1)
par(opar)
```

---

hsluv *HSLuv color specification*

Description

Converts a color given in HSLuv coordinates to a hexadecimal string or a RGB color specification.

Usage

```
hsluv(h = 360, s = 100, l = 100, alpha = NULL)

hsluv2rgb(h = 360, s = 100, l = 100)
```

Arguments

`h`	the hue, a number between `0` and `360`
`s`	the saturation, a number between `0` and `100`
`l`	the lightness, a number between `0` and `100`
`alpha`	opacity, a number between `0` and `1`, or `NULL`

Value

The `hsluv` function returns a hexadecimal string representing a color, and the `hsluv2rgb` returns the RGB coordinates of this color, a named vector of three integers between `0` and `255`.

Examples

```
saturation <- 100
f <- Vectorize(
  function(x, y){
    z <- complex(real = x, imaginary = y)
    modulus <- Mod(z)
    if(modulus > 1){
      return("#ffffff")
    }
    radians <- Arg(z)
    if(radians < 0){
      radians <- radians + 2*pi
    }
    degrees <- 360 * radians / 2 / pi
    hsluv(h = degrees, s = saturation, l = 100*modulus)
  }
)

x <- y <- seq(-1, 1, length.out = 200L)
image <- outer(x, y, f)

opar <- par(mar = c(0, 0, 0, 0))
plot(NULL, xlim = c(-1, 1), ylim = c(-1, 1), asp = 1)
rasterImage(image, -1, -1, 1, 1)
par(opar)
```

`permuteRGB` *RGB permutation*

Description

Permutes the R-G-B components of a color.

Usage

```
permuteRGB(hexcolor, permutation = "gbr")
```

Arguments

`hexcolor`	vector or matrix or array of hexadecimal colors
`permutation`	a character string with three letters `"r"`, `"g"` and `"b"`

Value

The colors after applying the permutation.

Examples

```
library(RcppColors)
x <- y <- seq(-1.7, 1.7, length.out = 512L)
zarray <- outer(y, x, function(x, y) {
  z <- x + 1i*y
  (1 + 1i) * log(sin((z^3 - 1)))
})
# image
img1 <- colorMap1(zarray)
# r -> b, g -> r, b -> g
img2 <- permuteRGB(img1, "brg")
# plot
opar <- par(mar = c(0,0,0,0), mfrow = c(1, 2), bg = "#002240")
plot(
  c(0, 1), c(0, 1), type = "n", asp = 1,
  xlab = NA, ylab = NA, axes = FALSE
)
rasterImage(img1, 0, 0, 1, 1, interpolate = TRUE)
plot(
  c(0, 1), c(0, 1), type = "n", asp = 1,
  xlab = NA, ylab = NA, axes = FALSE
)
rasterImage(img2, 0, 0, 1, 1, interpolate = TRUE)
par(opar)
```

Index

∗ **package**
RcppColors-package, 2

colorMap1, 2
colorMap10 *(*colorMap1*)*, 2
colorMap11 *(*colorMap1*)*, 2
colorMap12 *(*colorMap1*)*, 2
colorMap13 *(*colorMap1*)*, 2
colorMap14 *(*colorMap1*)*, 2
colorMap2 *(*colorMap1*)*, 2
colorMap3 *(*colorMap1*)*, 2
colorMap4 *(*colorMap1*)*, 2
colorMap5 *(*colorMap1*)*, 2
colorMap6 *(*colorMap1*)*, 2
colorMap7 *(*colorMap1*)*, 2
colorMap8 *(*colorMap1*)*, 2
colorMap9 *(*colorMap1*)*, 2

hsi, 6
hsl, 7
hsluv, 8
hsluv2rgb *(*hsluv*)*, 8

permuteRGB, 9

RcppColors *(*RcppColors-package*)*, 2
RcppColors-package, 2