

# Package ‘RcppColors’

October 12, 2022

**Type** Package

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

**Version** 0.1.1

**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** 7.2.1

**NeedsCompilation** yes

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

**Repository** CRAN

**Date/Publication** 2022-08-20 07:50:02 UTC

## R topics documented:

RcppColors-package . . . . .	2
colorMap1 . . . . .	2
hsluv . . . . .	4

<b>Index</b>	<b>6</b>
--------------	----------

---

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)  
)
```

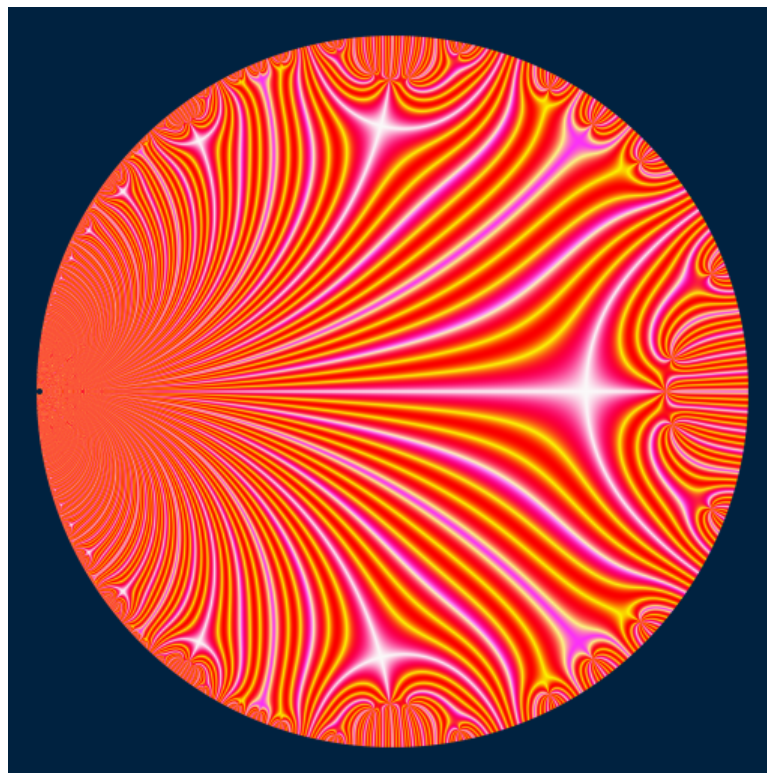
```
colorMap2(  
  Z,  
  bkgcolor = "#15191e",  
  nancolor = "#000000",  
  reverse = c(FALSE, FALSE, FALSE)  
)
```

**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 color component (e.g. R, G, B), whether to reverse it (e.g. R -> 255-R)

**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)

```

---

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)
```

# Index

\* **package**

RcppColors-package, [2](#)

colorMap1, [2](#)

colorMap2 (colorMap1), [2](#)

hsluv, [4](#)

hsluv2rgb (hsluv), [4](#)

RcppColors (RcppColors-package), [2](#)

RcppColors-package, [2](#)