

# Package ‘colorspace’

April 24, 2009

**Version** 1.0-1

**Date** 2009-04-24

**Title** Color Space Manipulation

**Author** Ross Ihaka, Paul Murrell, Kurt Hornik, Achim Zeileis

**Maintainer** Ross Ihaka <ihaka@stat.auckland.ac.nz>

**Description** Carries out mapping between assorted color spaces including RGB, HSV, HLS, CIEXYZ, CIELUV, HCL (polar CIELUV), CIELAB and polar CIELAB. Qualitative, sequential, and diverging color palettes based on HCL colors are provided.

**Depends** R (>= 2.0.0), methods

**Suggests** KernSmooth, MASS, kernlab, mvtnorm, vcd

**License** BSD

**LazyLoad** yes

**Repository** CRAN

**Date/Publication** 2009-04-24 16:23:06

## R topics documented:

color-class . . . . .	2
coords . . . . .	3
hex . . . . .	4
hex2RGB . . . . .	5
HLS . . . . .	6
HSV . . . . .	7
LAB . . . . .	8
LUV . . . . .	9
mixcolor . . . . .	10
polarLAB . . . . .	11
polarLUV . . . . .	12

rainbow_hcl . . . . .	13
readhex . . . . .	15
readRGB . . . . .	16
RGB . . . . .	17
writehex . . . . .	18
XYZ . . . . .	19

<b>Index</b>	<b>20</b>
--------------	-----------

---

color-class	Class "color"
-------------	---------------

---

## Description

Objects from the class *color* represent colors in a number of color spaces. In particular, there are subclasses of color which correspond to RGB, HSV, HLS, CIEXYZ, CIELUV, CIELAB and polar versions of the last two spaces.

## Objects from the Class

Objects can be created by calls to the functions RGB, HSV, HLS, XYZ, LUV, LAB, polarLUV, and polarLAB. These are all subclasses of the virtual class *color*.

## Slots

**coords:** An object of class "matrix".

## Methods

[ signature(x = "color"): This method makes it possible to take subsets of a vector of colors.

**coerce** signature(from = "color", to = "RGB"): convert a color vector to RGB.

**coerce** signature(from = "color", to = "XYZ"): convert a color vector to XYZ.

**coerce** signature(from = "color", to = "LAB"): convert a color vector to LAB.

**coerce** signature(from = "color", to = "polarLAB"): convert a color vector to polarLAB.

**coerce** signature(from = "color", to = "HSV"): convert a color vector to HSV.

**coerce** signature(from = "color", to = "HLS"): convert a color vector to HLS.

**coerce** signature(from = "color", to = "LUV"): convert a color vector to LUV.

**coerce** signature(from = "color", to = "polarLUV"): convert a color vector to polarLUV.

**coords** signature(color = "color"): extract the color coordinates from a color vector.

**plot** signature(x = "color"): plot a color vector

**show** signature(object = "color"): show a color vector.

**Author(s)**

Ross Ihaka

**See Also**

[RGB](#), [XYZ](#), [HSV](#), [HLS](#), [LAB](#), [polarLAB](#), [LUV](#), [polarLUV](#), [mixcolor](#).

**Examples**

```
x = RGB(runif(1000), runif(1000), runif(1000))
plot(as(x, "LUV"))
```

---

coords

*Extract the numerical coordinates of a color*

---

**Description**

This function returns a matrix with three columns which give the coordinates of a color in its natural color space.

**Usage**

```
coords(color)
```

**Arguments**

color            A color.

**Value**

A numeric matrix giving the coordinates of the color.

**Author(s)**

Ross Ihaka.

**See Also**

[RGB](#), [XYZ](#), [LAB](#), [polarLAB](#), [LUV](#), [polarLUV](#), [mixcolor](#).

**Examples**

```
x = RGB(1, 0, 0)
coords(as(x, "HSV"))
```

---

`hex`*Convert Colors To Hexadecimal Strings*

---

**Description**

This functions converts “color” objects into hexadecimal strings.

**Usage**

```
hex(from, gamma = 2.2, fixup = FALSE)
```

**Arguments**

<code>from</code>	The color object to be converted.
<code>gamma</code>	The display <i>gamma</i> value.
<code>fixup</code>	Should the color be corrected to a valid RGB value before correction. The default is to convert out-of-gamut colors to the string "NA".

**Details**

The color objects are first converted to RGB color objects. They are then multiplied by 255 and rounded to obtain an integer value. These values are then converted to hexadecimal strings of the form "#RRGGBB" and suitable for use as color descriptions for R graphics. Out of gamut values are either corrected to valid RGB values by translating the the individual primary values so that they lie between 0 and 255.

**Value**

A vector of character strings.

**Author(s)**

Ross Ihaka

**See Also**

[hex2RGB](#), [RGB](#), [HSV](#), [XYZ](#), [LAB](#), [polarLAB](#), [LUV](#), [polarLUV](#).

**Examples**

```
hsv = HSV(seq(0, 360, length=7)[-7], 1, 1)
barplot(rep(1, 6), col=hex(hsv))
```

---

`hex2RGB`*Convert Hexadecimal Color Specifications To RGB Objects*

---

**Description**

This function takes a vector of strings of the form "#RRGGBB" (hexadecimal color descriptions) into RGB objects.

**Usage**

```
hex2RGB(x, gamma = 2.2)
```

**Arguments**

<code>x</code>	a vector of hexadecimal color descriptions.
<code>gamma</code>	the display gamma value. Passing a value of NA results in no gamma correction being applied.

**Details**

This function converts device dependent color descriptions of the form "#RRGGBB" into device independent sRGB colour descriptions.

**Value**

An RGB object describing the colours.

**Author(s)**

Ross Ihaka

**See Also**

[hex](#), [RGB](#), [HSV](#), [XYZ](#), [polarLAB](#), [LUV](#), [polarLUV](#).

**Examples**

```
rgb = hex2RGB(c("#FF0000", "#00FF00", "#0000FF"))
```

---

HLS

*Create HLS Colors*

---

## Description

This function creates colors of class `HLS`; a subclass of the virtual “color” class.

## Usage

```
HLS(H, L, S, names)
```

## Arguments

`H, L, S` These arguments give the hue, lightness, and saturation of the colors. The values can be provided in separate `H`, `L` and `S` vectors or in a three-column matrix passed as `H`.

`names` A vector of names for the colors (by default the row names of `H` are used).

## Details

This function creates colors in the `HLS` color space which corresponds to the standard `sRGB` color space (IEC standard 61966). The hues should lie between between 0 and 360, and the lightness and saturations should lie between 0 and 1.

## Value

An object of class “`HLS`” which inherits from class “color.”

## Author(s)

Ross Ihaka

## References

[www.srgb.com](http://www.srgb.com)

## See Also

[RGB](#), [XYZ](#), [LAB](#), [polarLAB](#), [LUV](#), [polarLUV](#).

## Examples

```
# A rainbow of full-intensity hues
hls = HLS(seq(0, 360, length=13)[-13], 0.5, 1)
```

**Description**

This function creates colors of class HSV; a subclass of the virtual “color” class.

**Usage**

```
HSV(H, S, V, names)
```

**Arguments**

*H, S, V* These arguments give the hue, saturation and value of the colors. The values can be provided in separate *H*, *S* and *V* vectors or in a three-column matrix passed as *H*.

*names* A vector of names for the colors (by default the row names of *H* are used).

**Details**

This function creates colors in the HSV color space which corresponds to the standard sRGB color space (IEC standard 61966). The hues should lie between 0 and 360, and the saturations and values should lie between 0 and 1.

**Value**

An object of class “HSV” which inherits from class “color.”

**Author(s)**

Ross Ihaka

**References**

[www.srgb.com](http://www.srgb.com)

**See Also**

[RGB](#), [XYZ](#), [LAB](#), [polarLAB](#), [LUV](#), [polarLUV](#).

**Examples**

```
# A rainbow of full-intensity hues
hsv = HSV(seq(0, 360, length=13)[-13], 1, 1)
```

## Description

This function creates colors of class “LAB”; a subclass of the virtual “color” class.

## Usage

```
LAB(L, A, B, names)
```

## Arguments

<code>L, A, B</code>	these arguments give the L, A and B coordinates of the colors. The values can be provided in separate L, A and B vectors or in a three-column matrix passed as L.
<code>names</code>	a vector of names for the colors (by default the row names of L are used).

## Details

The L, A and B values give the coordinates of the colors in the CIE  $L^*a^*b^*$  space. This is a transformation of the 1931 CIE XYZ space which attempts to produce perceptually based axes. Luminance takes values between 0 and 100, and the other coordinates take values between -100 and 100. The  $a$  and  $b$  coordinates measure positions on green/red and blue/yellow axes.

## Value

An object of class “LAB” which inherits from class “color.”

## Author(s)

Ross Ihaka

## See Also

[RGB](#), [HSV](#), [XYZ](#), [LAB](#), [polarLAB](#), [LUV](#), [polarLUV](#).

## Examples

```
## Show the LAB space
x = RGB(runif(1000), runif(1000), runif(1000))
plot(as(x, "LAB"))
```

**Description**

This function creates colors of class “LUV”; a subclass of the virtual “color” class.

**Usage**

```
LUV(L, U, V, names)
```

**Arguments**

<code>L, U, V</code>	these arguments give the L, U and V coordinates of the colors. The values can be provided in separate L, U and V vectors or in a three-column matrix passed as L.
<code>names</code>	a vector of names for the colors (by default the row names of L are used).

**Details**

The L, U and V values give the coordinates of the colors in the CIE (1976)  $L^*u^*v^*$  space. This is a transformation of the 1931 CIE XYZ space which attempts to produce perceptually based axes. Luminance takes values between 0 and 100, and the other coordinates take values between -100 and 100. The  $a$  and  $b$  coordinates measure positions on green/red and blue/yellow axes.

**Value**

An object of class “LUV” which inherits from class “color.”

**Author(s)**

Ross Ihaka

**See Also**

[RGB](#), [HSV](#), [XYZ](#), [LAB](#), [polarLAB](#), [polarLUV](#).

**Examples**

```
## Show the LUV space
x = RGB(runif(1000), runif(1000), runif(1000))
plot(as(x, "LUV"))
```

---

`mixcolor`*Compute the convex combination of two colors*

---

**Description**

This function can be used to compute the result of color mixing (it assumes additive mixing).

**Usage**

```
mixcolor(alpha, color1, color2, where = class(color1))
```

**Arguments**

<code>alpha</code>	The mixed color is obtained by combining an amount $1-\text{alpha}$ of <code>color1</code> with an amount <code>alpha</code> of <code>color2</code> .
<code>color1</code>	The first color.
<code>color2</code>	The second color.
<code>where</code>	The color space where the mixing is to take place.

**Value**

The mixed color. This is in the color space specified by `where`.

**Author(s)**

Ross Ihaka

**See Also**

[RGB](#), [HSV](#), [XYZ](#), [LAB](#), [polarLAB](#), [LUV](#), [polarLUV](#).

**Examples**

```
mixcolor(0.5, RGB(1, 0, 0), RGB(0, 1, 0))
```

**Description**

This function creates colors of class “polarLAB”; a subclass of the virtual “color” class.

**Usage**

```
polarLAB(L, C, H, names)
```

**Arguments**

<code>L, C, H</code>	these arguments give the L, C and H coordinates of the colors. The values can be provided in separate L, C and H vectors or in a three-column matrix passed as L.
<code>names</code>	A vector of names for the colors (by default the row names of L are used).

**Details**

The polarLAB space is a transformation of the CIE  $L^*a^*b^*$  space so that the  $a$  and  $b$  values are converted to polar coordinates. The radial component  $C$  measures chroma and the angular coordinate  $H$  is measures hue.

**Value**

An object of class “polarLAB” which inherits from class “color.”

**Author(s)**

Ross Ihaka

**See Also**

[RGB](#), [HSV](#), [XYZ](#), [LAB](#), [polarLAB](#), [LUV](#), [polarLUV](#).

**Examples**

```
## Show the polarLAB space
x = RGB(runif(1000), runif(1000), runif(1000))
plot(as(x, "polarLAB"))
```

---

`polarLUV`*Create polarLUV Colors*

---

**Description**

This function creates colors of class “polarLUV”; a subclass of the virtual “color” class.

**Usage**

```
polarLUV(L, C, H, names)
```

**Arguments**

<code>L, C, H</code>	these arguments give the L, C and H coordinates of the colors. The values can be provided in separate L, C and H vectors or in a three-column matrix passed as L.
<code>names</code>	A vector of names for the colors (by default the row names of L are used).

**Details**

The polarLUV space is a transformation of the CIE  $L^*u^*v^*$  space so that the  $u$  and  $v$  values are converted to polar coordinates. The radial component  $C$  measures chroma and the angular coordinate  $H$  is measures hue.

**Value**

An object of class “polarLUV” which inherits from class “color.”

**Author(s)**

Ross Ihaka

**See Also**

[RGB](#), [HSV](#), [XYZ](#), [LAB](#), [polarLAB](#), [LUV](#), [polarLUV](#).

**Examples**

```
## Show the polarLUV space
x = RGB(runif(1000), runif(1000), runif(1000))
plot(as(x, "polarLUV"))
```

rainbow\_hcl

*HCL and HSV Color Palettes***Description**

Color palettes based on the HCL and HSV color spaces.

**Usage**

```
rainbow_hcl(n, c = 50, l = 70, start = 0, end = 360*(n-1)/n,
  gamma = 2.4, fixup = TRUE, ...)

sequential_hcl(n, h = 260, c. = c(80, 0), l = c(30, 90), power = 1.5,
  gamma = 2.4, fixup = TRUE, ...)
heat_hcl(n, h = c(0, 90), c. = c(100, 30), l = c(50, 90), power = c(1/5, 1),
  gamma = 2.4, fixup = TRUE, ...)
terrain_hcl(n, h = c(130, 0), c. = c(80, 0), l = c(60, 95), power = c(1/10, 1),
  gamma = 2.4, fixup = TRUE, ...)

diverge_hcl(n, h = c(260, 0), c = 80, l = c(30, 90), power = 1.5,
  gamma = 2.4, fixup = TRUE, ...)
diverge_hsv(n, h = c(240, 0), s = 1, v = 1, power = 1,
  gamma = 2.4, fixup = TRUE, ...)
```

**Arguments**

n	the number of colors ( $\geq 1$ ) to be in the palette.
c, c.	chroma value in the HCL color description.
l	luminance value in the HCL color description.
start	the hue at which the rainbow begins.
end	the hue at which the rainbow ends.
h	hue value in the HCL or HSV color description, has to be in [0, 360] for HCL and in [0, 1] for HSV colors.
s	saturation value in the HSV color description.
v	value value in the HSV color description.
power	control parameter determining how chroma and luminance should be increased (1 = linear, 2 = quadratic, etc.).
gamma	gamma value of the display.
fixup	logical. Should the color be corrected to a valid RGB value before correction?
...	Other arguments passed to <a href="#">hex</a> .

## Details

All functions compute palettes based on either the HCL ([polarLUV](#)) or the HSV ([HSV](#)) color space.

`rainbow_hcl` computes a rainbow of colors (qualitative palette) defined by different hues given a single value of each chroma and luminance. It corresponds to `rainbow` which computes a rainbow in HSV space.

`sequential_hcl` gives a sequential palette starting at the full color  $HCL(h, c[1], l[1])$  through to a light color  $HCL(h, c[2], l[2])$  by interpolation.

`diverge_hcl` and `diverge_hsv`, compute a set of colors diverging from a neutral center (grey or white, without color) to two different extreme colors (blue and red by default). This is similar to `cm.colors`. For the diverging HSV colors, two hues  $h$  are needed, a maximal saturation  $s$  and a fixed value  $v$ . The saturation is then varied to obtain the diverging colors. For the diverging HCL colors, again two hues  $h$  are needed, a maximal chroma  $c$  and two luminances  $l$ . The colors are then created by an interpolation between the full color  $HCL(h[1], c, l[1])$ , a neutral color  $HCL(h, 0, l[2])$  and the other full color  $HCL(h[2], c, l[1])$ .

The palette `heat_hcl` gives an implementation of `heat.colors` in HCL space. By default, it goes from a red to a yellow hue, while simultaneously going to lighter colors (i.e., increasing luminance) and reducing the amount of color (i.e., decreasing chroma). The `terrain_hcl` palette simply calls `heat_hcl` with different parameters, providing colors similar in spirit to `terrain.colors`. The lighter colors are not strictly HCL colors, though.

## Value

A character vector with (s)RGB codings of the colors in the palette.

## Author(s)

Achim Zeileis <Achim.Zeileis@R-project.org>

## References

Zeileis A., Hornik K. and Murrell P. (2009), Escaping RGBland: Selecting Colors for Statistical Graphics. *Computational Statistics & Data Analysis*, **53**, 3259-3270. doi:10.1016/j.csda.2008.11.033  
Preprint available from <http://statmath.wu-wien.ac.at/~zeileis/papers/Zeileis+Hornik+Murrell-2009.pdf>.

## See Also

[polarLUV](#), [HSV](#), [hex](#)

## Examples

```
## convenience demo functions
wheel <- function(col, radius = 1, ...)
  pie(rep(1, length(col)), col = col, radius = radius, ...)

pal <- function(col, border = "light gray")
{
```

```

n <- length(col)
plot(0, 0, type="n", xlim = c(0, 1), ylim = c(0, 1), axes = FALSE, xlab = "", ylab = "")
rect(0:(n-1)/n, 0, 1:n/n, 1, col = col, border = border)
}

## qualitative palette
wheel(rainbow_hcl(12))

## a few useful diverging HCL palettes
par(mar = rep(0, 4), mfrow = c(4, 1))
pal(diverge_hcl(7))
pal(diverge_hcl(7, h = c(246, 40), c = 96, l = c(65, 90)))
pal(diverge_hcl(7, h = c(130, 43), c = 100, l = c(70, 90)))
pal(diverge_hcl(7, h = c(180, 70), c = 70, l = c(90, 95)))
pal(diverge_hcl(7, h = c(180, 330), c = 59, l = c(75, 95)))
pal(diverge_hcl(7, h = c(128, 330), c = 98, l = c(65, 90)))
pal(diverge_hcl(7, h = c(255, 330), l = c(40, 90)))
pal(diverge_hcl(7, c = 100, l = c(50, 90), power = 1))

## sequential palettes
pal(sequential_hcl(12))
pal(heat_hcl(12, h = c(0, -100), l = c(75, 40), c = c(40, 80), power = 1))
pal(terrain_hcl(12, c = c(65, 0), l = c(45, 95), power = c(1/3, 1.5)))
pal(heat_hcl(12, c = c(80, 30), l = c(30, 90), power = c(1/5, 1.5)))

## compare base and vcd palettes
par(mfrow = c(2, 1))
wheel(rainbow(12)); wheel(rainbow_hcl(12))
pal(diverge_hcl(7, c = 100, l = c(50, 90))); pal(diverge_hsv(7))
pal(diverge_hcl(7, h = c(180, 330), c = 59, l = c(75, 95))); pal(cm.colors(7))
pal(heat_hcl(12)); pal(heat.colors(12))
pal(terrain_hcl(12)); pal(terrain.colors(12))

```

---

readhex

*Read Hexadecimal Color Descriptions*


---

### Description

This function reads a set of hexadecimal color descriptions from a file and creates a color object containing the corresponding colors.

### Usage

```
readhex(file = "", class = "RGB")
```

### Arguments

file	The file containing the color descriptions.
class	The kind of color object to be returned.

**Details**

The file is assumed to contain hexadecimal color descriptions of the form #RRGGBB.

**Value**

An color object of the specified class containing the color descriptions.

**Author(s)**

Ross Ihaka.

**See Also**

[writehex](#), [readRGB](#), [hex2RGB](#), [RGB](#), [HSV](#), [XYZ](#), [LAB](#), [polarLAB](#), [LUV](#), [polarLUV](#),

**Examples**

```
## Not run:
rgb = readhex("pastel.txt")
hsv = readhex("pastel.txt", "HSV")
## End(Not run)
```

---

readRGB

*Read RGB Color Descriptions*

---

**Description**

This function reads a set of RGB color descriptions (of the form written by `gcolorsel`) from a file and creates a color object containing the corresponding colors.

**Usage**

```
readRGB(file = "", class = "RGB")
```

**Arguments**

<code>file</code>	The file containing the color descriptions.
<code>class</code>	The kind of color object to be returned.

**Details**

The file is assumed to contain RGB color descriptions consisting of three integer values in the range from 0 to 255 followed by a color name.

**Value**

An color object of the specified class containing the color descriptions.

**Author(s)**

Ross Ihaka.

**See Also**

[writehex](#), [readhex](#), [hex2RGB](#), [RGB](#), [HSV](#), [XYZ](#), [LAB](#), [polarLAB](#), [LUV](#), [polarLUV](#).

**Examples**

```
## Not run:
rgb = readRGB("pastel.rgb")
hsv = readRGB("pastel.rgb", "HSV")
## End(Not run)
```

---

RGB

*Create RGB Colors*

---

**Description**

This function creates colors of class `RGB`; a subclass of the virtual “color” class.

**Usage**

```
RGB(R, G, B, names)
```

**Arguments**

<code>R, G, B</code>	these arguments give the red, green and blue intensities of the colors (the values should lie between 0 and 1). The values can be provided in separate <code>R</code> , <code>G</code> and <code>B</code> vectors or in a three-column matrix passed as <code>R</code> .
<code>names</code>	A vector of names for the colors (by default the row names of <code>R</code> are used).

**Details**

This function creates colors in the RGB color space which is a linearly transformed version of CIEXYZ space.

**Value**

An object of class “RGB” which inherits from class “color.”

**Author(s)**

Ross Ihaka

**References**

[www.srgb.com](http://www.srgb.com)

**See Also**

[HSV](#), [XYZ](#), [LAB](#), [polarLAB](#), [LUV](#), [polarLUV](#).

**Examples**

```
# Create a random set of colors
rgb = RGB(R = runif(20), G = runif(20), B = runif(20))
```

---

writehex

*Write Hexadecimal Color Descriptions*

---

**Description**

Given a color object, this function writes a file containing the hexadecimal representation of the colors in the object.

**Usage**

```
writehex(x, file = "")
```

**Arguments**

x	a color object.
file	the name of the file to be written.

**Details**

This function converts the given color object to RGB and then writes hexadecimal strings (of the form #RRGGBB) representing the colors to the specified file.

**Value**

The name of the file is returned as the value of the function.

**Author(s)**

Ross Ihaka

**See Also**

[readhex](#), [readRGB](#), [hex2RGB](#), [RGB](#), [HSV](#), [XYZ](#), [LAB](#), [polarLAB](#), [LUV](#), [polarLUV](#).

**Examples**

```
x = RGB(runif(10), runif(10), runif(10))
writehex(x, "random.txt")
```

---

`XYZ`*Create XYZ Colors*

---

**Description**

This function creates colors of class XYZ; a subclass of the virtual “color” class.

**Usage**

```
XYZ(X, Y, Z, names)
```

**Arguments**

`X, Y, Z` these arguments give the X, Y and Z coordinates of the colors. The values can be provided in separate X, Y and Z vectors or in a three-column matrix passed as X.

`names` A vector of names for the colors (by default the row names of X are used).

**Details**

The X, Y and Z values are the levels of the CIE primaries. These are scaled so that the luminance of the display white-point is 100. The white-point is taken to be D65, which means that its coordinates are 95.047, 100.000, 108.883.

**Value**

An object of class “XYZ” which inherits from class “color.”

**Author(s)**

Ross Ihaka

**See Also**

[RGB](#), [HSV](#), [LAB](#), [polarLAB](#), [LUV](#), [polarLUV](#).

**Examples**

```
## Generate white in XYZ space  
white = XYZ(95.047, 100.000, 108.883)
```

# Index

## \*Topic **classes**

color-class, 2

## \*Topic **color**

coords, 3

hex, 4

hex2RGB, 5

HLS, 6

HSV, 7

LAB, 8

LUV, 9

mixcolor, 10

polarLAB, 11

polarLUV, 12

rainbow\_hcl, 13

readhex, 15

readRGB, 16

RGB, 17

writehex, 18

XYZ, 19

[, color-method (*color-class*), 2

cm.colors, 14

coerce, color, HLS-method  
(*color-class*), 2

coerce, color, HSV-method  
(*color-class*), 2

coerce, color, LAB-method  
(*color-class*), 2

coerce, color, LUV-method  
(*color-class*), 2

coerce, color, polarLAB-method  
(*color-class*), 2

coerce, color, polarLUV-method  
(*color-class*), 2

coerce, color, RGB-method  
(*color-class*), 2

coerce, color, XYZ-method  
(*color-class*), 2

color-class, 2

coords, 3

coords, color-method  
(*color-class*), 2

diverge\_hcl (*rainbow\_hcl*), 13

diverge\_hsv (*rainbow\_hcl*), 13

heat.colors, 14

heat\_hcl (*rainbow\_hcl*), 13

hex, 4, 5, 13, 14

hex2RGB, 4, 5, 16–18

HLS, 2, 6

HLS-class (*color-class*), 2

HSV, 2, 4, 5, 7, 8–12, 14, 16–19

HSV-class (*color-class*), 2

LAB, 2–4, 6, 7, 8, 8–12, 16–19

LAB-class (*color-class*), 2

LUV, 2–8, 9, 10–12, 16–19

LUV-class (*color-class*), 2

mixcolor, 2, 3, 10

plot, color-method (*color-class*), 2

polarLAB, 2–10, 11, 11, 12, 16–19

polarLAB-class (*color-class*), 2

polarLUV, 2–11, 12, 12, 14, 16–19

polarLUV-class (*color-class*), 2

rainbow, 14

rainbow\_hcl, 13

readhex, 15, 17, 18

readRGB, 16, 16, 18

RGB, 2–12, 16, 17, 17–19

RGB-class (*color-class*), 2

sequential\_hcl (*rainbow\_hcl*), 13

show, color-method (*color-class*), 2

terrain\_hcl (*rainbow\_hcl*), 13

writehex, 16, 17, 18

XYZ, 2–12, 16–18, 19

XYZ-class (*color-class*), 2