Package ‘prismatic’

October 17, 2021

Title  Color Manipulation Tools
Version  1.1.0
Description  Manipulate and visualize colors in an intuitive, low-dependency and functional way.
License  MIT + file LICENSE
URL  https://github.com/EmilHvitfeldt/prismatic
BugReports  https://github.com/EmilHvitfeldt/prismatic/issues
Depends  R (>= 3.2)
Imports  graphics, farver (>= 2.0.1), grDevices
Suggests  covr, crayon, testthat (>= 3.0.0)
Encoding  UTF-8
RoxygenNote  7.1.2
Config/testthat/edition  3
NeedsCompilation  no
Author  Emil Hvitfeldt [aut, cre] (<https://orcid.org/0000-0002-0679-1945>)
Maintainer  Emil Hvitfeldt <emilhhvitfeldt@gmail.com>
Repository  CRAN
Date/Publication  2021-10-17 04:30:02 UTC

R topics documented:

best_contrast .................................................. 2
check_color_blindness ........................................ 3
clr_alpha ...................................................... 3
clr_darken ..................................................... 4
clr_desaturate ................................................ 5
clr_extract .................................................... 6
clr_extract_chroma .......................................... 7
clr_extract_hue .............................................. 8
clr_extract_red ............................................. 9
best_contrast

Description

Finds the color in ‘y’ with the highest contrast to the color ‘x’.

Usage

best_contrast(x, y = c("#010101", "#FFFFFF"))

Arguments

x               Multiple colors
y               Multiple colors

Value

The elements of ‘y’ with highest contrast to ‘x’.

Examples

best_contrast("red")
best_contrast("grey20")
best_contrast("white")

best_contrast(rainbow(10), rainbow(3))
check_color_blindness  Visualize color vision deficiency

Description

Visualize color vision deficiency

Usage

check_color_blindness(col)

Arguments

- **col**: a color object or vector of any of the three kinds of R color specifications, i.e., either a color name (as listed by colors()), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa" (see rgb), or a positive integer i meaning palette()[i]. This function will showcase the effect of all 3 kinds of color vision deficiency at the same time side by side.

Value

Nothing

Examples

check_color_blindness(rainbow(10))

check_color_blindness(terrain.colors(10))

---

clr_alpha  Sets alpha in color

Description

Sets alpha in color

Usage

clr_alpha(col, alpha = 0.5)

Arguments

- **col**: a color object or vector of any of the three kinds of R color specifications, i.e., either a color name (as listed by colors()), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa" (see rgb), or a positive integer i meaning palette()[i].

- **alpha**: Numeric between 0 and 1. 0 will result in full transparency and 1 results in no transparency.
Value

a colors object

Examples

plot(clr_alpha(rainbow(10), 0.5))
plot(clr_alpha(rainbow(10), 0.2))
plot(clr_alpha(rainbow(10), seq(0, 1, length.out = 10)))

clr_darken

Make a color more dark

Description

Make a color more dark

Usage

clr_darken(col, shift = 0.5, space = c("HCL", "HSL", "combined"))

Arguments

col a color object or vector of any of the three kinds of R color specifications, i.e., either a color name (as listed by colors()), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa" (see rgb), or a positive integer i meaning palette()[i].
shift Numeric between 0 and 1, 0 will do zero darkening, 1 will do complete darkening turning the color to black. Defaults to 0.5.
space character string specifying the color space in which adjustment happens. Can be either "HCL", "HSL" or "combined". Defaults to "HCL".

Details

The colors will be transformed to HSL color space (hue, saturation, lightness) where the lightness of the color will be modified. The lightness of a color takes a value between 0 and 1, with 0 being black and 1 being white. The shift argument takes a value between 0 and 1, where 0 means that the lightness stays unchanged and 1 means completely black. As an example, if the lightness of the color is 0.6 and shift is 0.5, then the lightness be set to the halfway point between 0.6 and 0, which is 0.3.

If space = "HSL" then the colors are transformed to HSL space where the lightness value L is adjusted. If space = "HCL" then the colors are transformed to Cylindrical HCL space where the luminance value L is adjusted. If space = "combined" then the colors are transformed into HSL and Cylindrical HCL space. Where the color adjusting is happening HLS is copied to the values in the HCL transformation. Thus the "combined" transformation adjusts the luminance in HCL space and chroma in HSL space. For more information regarding use of color spaces, please refer to the colorspace paper https://arxiv.org/abs/1903.06490.
**clr_desaturate**

Value

a color object of same length as col.

Source

https://en.wikipedia.org/wiki/HSL_and_HSV
https://en.wikipedia.org/wiki/CIELUV
https://arxiv.org/abs/1903.06490

See Also

clr_lighten

Examples

# Using linear shift
plot(clr_darken(rep("red", 11), shift = seq(0, 1, 0.1)))
plot(clr_darken(rep("red", 11), shift = seq(0, 1, 0.1), space = "HSL"))
plot(clr_darken(rep("red", 11), shift = seq(0, 1, 0.1), space = "combined"))

plot(clr_darken(terrain.colors(10)))

# Using exponential shifts
plot(clr_darken(rep("red", 11), shift = log(seq(1, exp(1), length.out = 11))))

---

**clr_desaturate**

Make a color more desaturated

Description

Make a color more desaturated

Usage

clr_desaturate(col, shift = 0.5)

Arguments

col a color object or vector of any of the three kinds of R color specifications, i.e.,
either a color name (as listed by colors()), a hexadecimal string of the form
"#rrggbbaa" or "#rrggbbaa" (see rgb), or a positive integer i meaning palette()[i].

shift Numeric between 0 and 1, 0 will do zero desaturation, 1 will do complete desat-

uration. Defaults to 0.5.
Details

The colors will be transformed to HSL color space (hue, saturation, lightness) where the saturation of the color will be modified. The saturation of a color takes a value between 0 and 1, with 0 being black and 1 being white. The shift argument takes a value between 0 and 1, where 0 means that the saturation stays unchanged and 1 means completely desaturated. As an example, if the saturation of the color is 0.6 and shift is 0.5, then the saturation be set to the halfway point between 0.6 and 0 which is 0.3.

Value

a colors object of same length as col.

Source

https://en.wikipedia.org/wiki/HSL_and_HSV

See Also
clr_saturate

Examples

plot(clr_desaturate(terrain.colors(10), shift = 0.5))
plot(clr_desaturate(terrain.colors(10), shift = 0.9))
plot(clr_desaturate(rep("firebrick", 11), shift = seq(0, 1, 0.1)))

clr_extract

Extract Multiple Components

Description

Extract multiple color components at the same time.

Usage

clr_extract(
  col,
  components = c("red", "green", "blue", "hue_hsl", "saturation", "lightness",
                  "hue_hcl", "chroma", "luminance")
)
clr_extract_chroma

Arguments

- col: a color object or vector of any of the three kinds of R color specifications, i.e., either a color name (as listed by colors()), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa" (see rgb), or a positive integer i meaning palette()[i].
- components: character, components that should be extracted. See details for allowed components.

Details

The allowed values for 'components' are:
- red - green - blue - hue_hsl - saturation - lightness - hue_hcl - chroma - luminance

This function is to be preferred if you need to extract multiple components at the same time, since it doesn't need repeat transformations.

Value

data.frame of components

See Also

Other Extraction: clr_extract_chroma(), clr_extract_hue(), clr_extract_red()

Examples

clr_extract(rainbow(10))
clr_extract(rainbow(10), c("hue_hsl", "saturation"))
Details

The range of the value are
- hue ranges from 0 to 360 - luminance ranges from 0 to 100 - chroma while depended on hue and luminance will roughly be within 0 and 180

Use [clr_extract()] if you are planning to extraction multiple components.

Value

Numeric vector of values.

See Also

Other Extraction: clr_extract_hue(), clr_extract_red(), clr_extract()

Examples

clr_extract_hue(rainbow(100), "HCL")
clr_extract_chroma(rainbow(100))
clr_extract_luminance(rainbow(100))
Details

The range of the value are
- hue ranges from 0 to 360. in a circular fashion such that 0 and 360 are near identical. 0 is red
- saturation ranges from 0 to 100. 100 is full saturation, 0 is no saturation - lightness ranges from 0
to 100. 100 is full lightness, 0 is no lightness
Use [clr_extract()] if you are planning to extraction multiple components.

Value

Numeric vector of values.

See Also

Other Extraction: clr_extract_chroma(), clr_extract_red(), clr_extract()

Examples

clr_extract_hue(rainbow(100), "HSL")
clr_extract_saturation(rainbow(100))
clr_extract_lightness(rainbow(100))

clr_extract_red

Extract RGB components

Description

Extract the red, green, or blue color components from a vector of colors.

Usage

clr_extract_red(col)
clr_extract_green(col)
clr_extract_blue(col)
clr_extract_alpha(col)

Arguments

col a color object or vector of any of the three kinds of R color specifications, i.e.,
either a color name (as listed by colors()), a hexadecimal string of the form
"#rrggbh" or "#rrggbbaa" (see rgb), or a positive integer i meaning palette()[i].

Details

The values of the output will range between 0 and 255.
Use [clr_extract()] if you are planning to extraction multiple components.
Value

Numeric vector of values.

See Also

Other Extraction: `clr_extract_chroma()`, `clr_extract_hue()`, `clr_extract()`

Examples

```r
clr_extract_red(rainbow(100))
clr_extract_green(rainbow(100))
clr_extract_blue(rainbow(100))
clr_extract_alpha(rainbow(100))
```

---

**clr_grayscale**

Transform colors to greyscale

Description

This function has a selection of different methods to turn colors into grayscale.

Usage

```r
clr_grayscale(
  col,
  method = c("luma", "averaging", "min_decomp", "max_decomp", "red_channel",
             "green_channel", "blue_channel")
)
```

```r
clr_greyscale(
  col,
  method = c("luma", "averaging", "min_decomp", "max_decomp", "red_channel",
             "green_channel", "blue_channel")
)
```

Arguments

- `col` a color object or vector of any of the three kinds of R color specifications, i.e., either a color name (as listed by `colors()`), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa" (see `rgb`), or a positive integer i meaning `palette()[][i].`
- `method` character string specifying the grayscaling method. Can be one of "luma", "averaging", "min_decomp", "max_decomp", "red_channel", "green_channel" and "blue_channel". Defaults to "luma".
**clr_lighten**

**Details**

if method = "averaging" then the red, green and blue have been averaged together to create the grey value. This method does a poor job of representing the way the human eye sees color. If method = "luma" (the default) then then a weighted average is used to calculate the grayscale values. The BT. 709 method from the ITU Radiocommunication Sector have determined the weights. It method = "min_decomp" or method = "max_decomp", then a decomposition method is used where the minimum or maximum color value have been selected for the color value. So the color rgb(60, 120, 40) would have the min_decomp value of 40 and max_decomp value of 120. If method is "red_channel", "green_channel" or "blue_channel", then the corresponding color channel been selected for the values of grayscale.

**Value**

a colors object of same length as col.

**Source**

https://en.wikipedia.org/wiki/Luma

**Examples**

plot(clr_grayscale(rainbow(10)))
plot(clr_grayscale(terrain.colors(10)))

viridis_colors <- c(
    "#4B0055FF", "#422C70FF", "#185086FF", "#007094FF",
    "#008E98FF", "#00A890FF", "#00BE7DFF", "#6CD05EFF",
    "#BBDD38FF", "#FDE333FF"
)
plot(clr_grayscale(viridis_colors, method = "luma"))
plot(clr_grayscale(viridis_colors, method = "averaging"))
plot(clr_grayscale(viridis_colors, method = "min_decomp"))
plot(clr_grayscale(viridis_colors, method = "max_decomp"))
plot(clr_grayscale(viridis_colors, method = "red_channel"))
plot(clr_grayscale(viridis_colors, method = "green_channel"))
plot(clr_grayscale(viridis_colors, method = "blue_channel"))

---

<table>
<thead>
<tr>
<th>clr_lighten</th>
<th>Make a color more light</th>
</tr>
</thead>
</table>

**Description**

Make a color more light
Usage

clr_lighten(col, shift = 0.5, space = c("HCL", "HSL", "combined"))

Arguments

col a color object or vector of any of the three kinds of R color specifications, i.e.,
either a color name (as listed by colors()), a hexadecimal string of the form
"#rrggbbaa" (see rgb), or a positive integer i meaning palette()[i].
shift Numeric between 0 and 1, 0 will do zero lightening, 1 will do complete lighten-
ing turning the color to white. Defaults to 0.5.
space character string specifying the color space in which adjustment happens. Can
be either "HCL", "HSL" or "combined". Defaults to "HCL".

Details

The colors will be transformed to HSL color space (hue, saturation, lightness) where the lightness
of the color will be modified. The lightness of a color takes a value between 0 and 1, with 0 being
black and 1 being white. The shift argument takes a value between 0 and 1, where 0 means that
the lightness stays unchanged and 1 means completely white. As an example, if the lightness of the
color is 0.6 and shift is 0.5, then the lightness be set to the halfway point between 0.6 and 1 which
is 0.8.

If space = "HSL" then the colors are transformed to HSL space where the lightness value L is
adjusted. If space = "HCL" then the colors are transformed to Cylindrical HCL space where the
luminance value L is adjusted. If space = "combined" then the colors are transformed into HSL
and Cylindrical HCL space. Where the color adjusting is happening HLS is copied to the values in
the HCL transformation. Thus the "combined" transformation adjusts the luminance in HCL space
and chroma in HSL space. For more information regarding use of color spaces, please refer to the

Value

a colors object of same length as col.

Source

https://en.wikipedia.org/wiki/HSL_and_HSV
https://en.wikipedia.org/wiki/CIELUV
https://arxiv.org/abs/1903.06490

See Also

clr_darken
Examples

# Using linear shift
plot(clr_lighten(rep("red", 11), shift = seq(0, 1, 0.1)))
plot(clr_lighten(rep("red", 11), shift = seq(0, 1, 0.1), space = "HSL"))
plot(clr_lighten(rep("red", 11), shift = seq(0, 1, 0.1), space = "combined"))
plot(clr_lighten(terrain.colors(10)))

# Using exponential shifts
plot(clr_lighten(rep("red", 11), shift = log(seq(1, exp(1), length.out = 11)))))

clr_mix

Mixes a color into

Description

Mixes a color into

Usage

clr_mix(col, mix_in, ratio = 0.5)

Arguments

col a color object or vector of any of the three kinds of R color specifications, i.e.,
either a color name (as listed by colors()), a hexadecimal string of the form
"#rrggbbaa" (see rgb), or a positive integer i meaning palette()[i].
mix_in A single color any of the three kinds of R color specifications, i.e., either a
color name (as listed by colors()), a hexadecimal string of the form "#rrggbbaa" or
"#rrggbbaa" (see rgb), or a positive integer i meaning palette()[i].
ratio Numeric between 0 and 1. 0 will result on no mixing. 1 results in all the colors
turning to mix_in. Must be of length 1 or same length as col.

Value

a colors object

Examples

plot(clr_mix(rainbow(10), "blue"))
plot(clr_mix(rainbow(10), "red"))
plot(clr_mix(rainbow(10), "#5500EE"))
plot(clr_mix(rainbow(10), "black", seq(1, 0, length.out = 10)))
### clr_negate

**Negates colors in RGB space**

**Description**

Negates colors in RGB space.

**Usage**

```r
clr_negate(col)
```

**Arguments**

`col` a color object or vector of any of the three kinds of R color specifications, i.e., either a color name (as listed by `colors()`), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa" (see `rgb`), or a positive integer i meaning `palette()[i]`.

**Details**

The negation of color is happening in the red-green-blue colorspace RGB. Meaning that if we take the specification for Orange which is rgb(255, 165, 0), then we negate by taking the opposite number on the scale from 0 to 255, leaving us with rgb(0, 90, 255) which is a shade of blue.

**Value**

a colors object of same length as col.

**Examples**

```r
terr <- color(terrain.colors(10))

terr
clr_negate(terr)

plot(terr)
plot(clr_negate(terr))
```

### clr_protan

**Simulate color vision deficiency**

**Description**

Simulate color vision deficiency.
clr_protan

Usage

clr_protan(col, severity = 1)
clr_deutan(col, severity = 1)
clr_tritan(col, severity = 1)

Arguments

col a color object or vector of any of the three kinds of R color specifications, i.e., either a color name (as listed by colors()), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa" (see rgb), or a positive integer i meaning palette()[i].

severity A numeric. Severity of the color vision defect, a number between 0 and 1. 0 means no deficiency, 1 means complete deficiency. Defaults to 1.

Details

The matrices uses to perform transformations have been taken as the 1.0 value in table 1 in http://www.inf.ufrgs.br/~oliveira/pubs_files/CVD_Simulation/CVD_Simulation.html.

Value

a colors object of same length as col.

Source

http://www.inf.ufrgs.br/~oliveira/pubs_files/CVD_Simulation/CVD_Simulation.html

References


Examples

rainbow_colors <- color(rainbow(10))

plot(clr_protan(rainbow_colors))
plot(clr_deutan(rainbow_colors))
plot(clr_tritan(rainbow_colors))

viridis_colors <- c(
    "#4B0055FF", "#422C70FF", "#185086FF", "#007094FF", 
    "#008E98FF", "#00A890FF", "#00BE7DFF", "#6CD05EFF", 
    "#BBDD38FF", "#FDE333FF"
)

plot(clr_protan(viridis_colors))
plot(clr_deutan(viridis_colors))
plot(clr_tritan(viridis_colors))
**clr_rotate**

*Rotate the colors around the hue wheel*

**Description**

Rotate the colors around the hue wheel

**Usage**

```r
clr_rotate(col, degrees = 0)
```

**Arguments**

- `col`: a color object or vector of any of the three kinds of R color specifications, i.e., either a color name (as listed by `colors()`), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa" (see `rgb`), or a positive integer i meaning `palette()[i]`.
- `degrees`: Numeric between 0 and 360, denoting the amount of degrees the colors should be rotated. Defaults to 0.

**Details**

The colors will be transformed to HCL color space (Hue-Chroma-Luminance) where the hue of the color will be rotation.

**Value**

a `colors` object of same length as `col`.

**Source**


**Examples**

```r
plot(clr_rotate(terrain.colors(10)))
plot(clr_rotate(terrain.colors(10), degrees = 90))
plot(clr_rotate(terrain.colors(10), degrees = 180))
plot(clr_rotate(rep("magenta", 11), degrees = seq(0, 360, length.out = 11)))
```
clr_saturate

Make a color more saturated

Description
Make a color more saturated

Usage
clr_saturate(col, shift = 0.5)

Arguments
col: a color object or vector of any of the three kinds of R color specifications, i.e., either a color name (as listed by colors()), a hexadecimal string of the form "#rrggbbaa" or "#rrggbbaa" (see rgb), or a positive integer i meaning palette()[i].
shift: Numeric between 0 and 1, 0 will do zero saturation, 1 will do complete saturation. Defaults to 0.5.

Details
The colors will be transformed to HSL color space (hue, saturation, lightness) where the saturation of the color will be modified. The saturation of a color takes a value between 0 and 1, with 0 being black and 1 being white. The shift argument takes a value between 0 and 1, where 0 means that the saturation stays unchanged and 1 means completely saturated. As an example, if the saturation of the color is 0.6 and shift is 0.5, then the saturation be set to the halfway point between 0.6 and 1 which is 0.8.

Value
a color object of same length as col.

Source
https://en.wikipedia.org/wiki/HSL_and_HSV

See Also
clr_desaturate

Examples
plot(clr_saturate(terrain.colors(10), shift = 0.5))
plot(clr_saturate(terrain.colors(10), shift = 1))
plot(clr_saturate(rep("firebrick", 11), shift = seq(0, 1, 0.1)))
**color**  
*Turn vector to color vector*

**Description**

Turn vector to color vector

**Usage**

```r
color(col)
```

```r
colour(col)
```

**Arguments**

- `col` a color object or vector of any of the three kinds of R color specifications, i.e., either a color name (as listed by `colors()`), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa" (see `rgb`), or a positive integer `i` meaning `palette()[i]`.

**Details**

Alpha values will be automatically added to hexcodes. If none at present it will default to no alpha (FF).

**Value**

a `colors` object.

**Examples**

```r
terrain_10 <- color(terrain.colors(10))
terrain_10[1:4]
plot(terrain_10)
plot(terrain_10, labels = TRUE)
grey_10 <- color(gray.colors(10, start = 0, end = 1))
grey_10
plot(grey_10, labels = TRUE)
```
**contrast_ratio**

---

**Contrast Ratio Between Colors**

**Description**

Calculates the contrast ratio between 'x' and the colors 'y'. Contrast ratios can range from 1 to 21 with 1 being no contrast (same color) and 21 being highest contrast.

**Usage**

```
contrast_ratio(x, y)
```

**Arguments**

- **x**
  - A color object or vector of length 1 of any of the three kinds of R color specifications, i.e., either a color name (as listed by `colors()`), a hexadecimal string of the form "#rrggbbaa" or "#rrggbbaa" (see rgb), or a positive integer i meaning `palette()[i]`.

- **y**
  - A color object or vector of any of the three kinds of R color specifications, i.e., either a color name (as listed by `colors()`), a hexadecimal string of the form "#rrggbbaa" or "#rrggbbaa" (see rgb), or a positive integer i meaning `palette()[i]`.

**Details**

The formula for calculating contract ratio is

\[
\frac{(L_1 + 0.05)}{(L_2 + 0.05)}
\]

where

- \(L_1\) is the relative luminance of the lighter of the colors, and
- \(L_2\) is the relative luminance of the darker of the colors.

Relative luminance is calculated according to [https://www.w3.org/TR/2008/REC-WCAG20-20081211/#relativeluminancedef](https://www.w3.org/TR/2008/REC-WCAG20-20081211/#relativeluminancedef).

**Value**

The elements of 'y' with highest contrast to 'x'.

**Source**

[https://www.w3.org/TR/UNDERSTANDING-WCAG20/visual-audio-contrast-contrast.html](https://www.w3.org/TR/UNDERSTANDING-WCAG20/visual-audio-contrast-contrast.html)

**Examples**

```
contrast_ratio("red", "blue")
contrast_ratio("grey20", grey.colors(10))
contrast_ratio("white", c("white", "black"))
```
is_color

Test if the object is a color

Description
Test if the object is a color

Usage
is_color(x)

Arguments
x An object

Value
TRUE if the object inherits from the color class.

modify_hcl

Modify Individual HCL Axes

Description
This function lets you modify individual axes of a color in HCL color space.

Usage
modify_hcl(col, h, c, l)

Arguments
col a color object or vector of any of the three kinds of R color specifications, i.e., either a color name (as listed by colors()), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa" (see rgb), or a positive integer i meaning palette()[i].
h Expression to modify the hue of 'col'
c Expression to modify the chroma of 'col'
l Expression to modify the luminance of 'col'

Details
The expression used in 'h', 'c', and 'l' is evaluated in the 'hcl' space and and you have access to 'h', 'c', and 'l' as vectors along with vectors in the calling environment.
'h' ranges from 0 to 360, 'l' ranges from 0 to 100, and 'c' while depended on 'h' and 'l' will roughly be within 0 and 180, but often on a narrower range. Colors after modification will be adjusted to fit within the color space.
modify_hcl

Value

a colors object.

Source

https://en.wikipedia.org/wiki/HCL_color_space

Examples

plot(modify_hcl("red", h = 160))
plot(modify_hcl("red", h = h + 50))

plot(modify_hcl("red", h = h + 1:100))
plot(modify_hcl("red", c = c - 1:200))
plot(modify_hcl("red", l = l + 1:50))

plot(modify_hcl(rainbow(10), l = 25))

plot(modify_hcl(rainbow(10), h + h / 2, l = 70))
Index

* Extraction
  - clr_extract, 6
  - clr_extract_chroma, 7
  - clr_extract_hue, 8
  - clr_extract_red, 9

best_contrast, 2

check_color_blindness, 3
clr_alpha, 3
clr_darken, 4
clr_desaturate, 5
clr_deutan (clr_protan), 14
clr_extract, 6, 8–10
clr_extract_alpha (clr_extract_red), 9
clr_extract_blue (clr_extract_red), 9
clr_extract_chroma, 7, 7, 9, 10
clr_extract_green (clr_extract_red), 9
clr_extract_hue, 7, 8, 8, 10
clr_extract_lightness (clr_extract_hue), 8
clr_extract_luminance (clr_extract_hue), 8
clr_extract_red, 7–9, 9
clr_extract_saturation (clr_extract_hue), 8
clr_grayscale, 10
clr_greyscale (clr_grayscale), 10
clr_lighten, 11
clr_mix, 13
clr_negate, 14
clr_protan, 14
clr_rotate, 16
clr_saturate, 17
clr_tritan (clr_protan), 14

color, 18
colour (color), 18
contrast_ratio, 19

is_color, 20

modify_hcl, 20