Package 'farver'

November 20, 2018

Type Package

Title Vectorised Colour Conversion and Comparison

Version 1.1.0

Date 2018-11-20

Maintainer Thomas Lin Pedersen <thomasp85@gmail.com>

Description The encoding of colour can be handled in many different ways, using different colour spaces. As different colour spaces have different uses, efficient conversion between these representations are important. The 'farver' package provides a set of functions that gives access to very fast colour space conversion and comparisons implemented in C++, and offers 100-fold speed improvements over the 'convertColor' function in the 'grDevices' package.

License MIT + file LICENSE

Encoding UTF-8

SystemRequirements C++11

Imports Rcpp (>= 0.12.15)

LinkingTo Rcpp

RoxygenNote 6.1.1

URL https://github.com/thomasp85/farver

BugReports https://github.com/thomasp85/farver/issues

Suggests testthat, covr

NeedsCompilation yes

Author Thomas Lin Pedersen [cre, aut],
Berendea Nicolae [aut] (Author of the ColorSpace C++ library),
Romain François [aut] (<https://orcid.org/0000-0002-2444-4226>)

Repository CRAN

Date/Publication 2018-11-20 20:50:08 UTC

R topics documented:

`farver-package` *farver: Vectorised Colour Conversion and Comparison*

Description

The encoding of colour can be handled in many different ways, using different colour spaces. As different colour spaces have different uses, efficient conversion between these representations are important. The 'farver' package provides a set of functions that gives access to very fast colour space conversion and comparisons implemented in C++, and offers 100-fold speed improvements over the 'convertColor' function in the 'grDevices' package.

Author(s)

Maintainer: Thomas Lin Pedersen <thomasp85@gmail.com>

Authors:

- Berendea Nicolae (Author of the ColorSpace C++ library)
- Romain François <romain@purrple.cat> (0000-0002-2444-4226)

See Also

Useful links:

- https://github.com/thomasp85/farver
- Report bugs at https://github.com/thomasp85/farver/issues

`as_white_ref` *Convert value to a tristimulus values normalised to Y=100*

Description

This function can take either the name of a standardised illuminants, x and y chromaticity coordinates or X, Y, and Z tristimulus values and converts it to tristimulus values normalised to Y=100. All Illuminant series A-F are supported and can be queried both on the CIE 1931 2° and CIE 1964 10° chromaticity coordinates.

Usage

```
as_white_ref(x, fow = 2)
```

Arguments

x	A string giving the name of the standardized illuminant or a 2 (chromaticity) or 3 (trsitimulus) length numeric vector.
fow	The field-of-view for the illuminant - either 2 or 10

Value

A 3-length vector with tristimulus values

Examples

```
# Using names
as_white_ref('D65')

# Using chromaticity values
as_white_ref(c(0.3, 0.4))
```

---

compare_colour *Calculate the distance between colours*

Description

There are many ways to measure the distance between colours. farver provides 5 different algorithms, ranging from simple euclidean distance in RGB space, to different perceptual measures such as CIE2000.

Usage

```
compare_colour(from, to = NULL, from_space, to_space = from_space,
  method = "euclidean", white_from = "D65", white_to = white_from)
```

Arguments

from, to	Numeric matrices with colours to compare - the format is the same as that for convert_colour(). If to is not set from will be compared with itself and only the upper triangle will get calculated
from_space, to_space	The colour space of from and to respectively. to_space defaults to be the same as from_space.
method	The method to use for comparison. Either 'euclidean', 'cie1976', 'cie94', 'cie2000', or 'cmc'
white_from, white_to	The white reference of the from and to colour space. Will only have an effect for relative colour spaces such as Lab and luv. Any value accepted by as_white_ref() allowed.

Value

A numeric matrix with the same number of rows as colours in `from` and the same number of columns as colours in `to`. If `to` is not given, only the upper triangle will be returned.

Examples

```
r <- t(col2rgb(rainbow(10)))
h <- t(col2rgb(heat.colors(15)))

# Compare two sets of colours
compare_colour(r, h, 'rgb', method = 'cie2000')

# Compare a set of colours with itself
compare_colour(r, from_space = 'rgb', method = 'cmc')
```

---

convert_colour *Convert between colour spaces*

Description

This function lets you convert between different representations of colours. The API is reminiscent of `grDevices::convertColor()`, but the performance is much better. It is not assured that `grDevices::convertColor()` and `convert_colour()` provide numerically equivalent conversion at 16bit level as the formula used are potentially slightly different. For all intend and purpose, the resulting colours will be equivalent though.

Usage

```
convert_colour(colour, from, to, white_from = "D65",
  white_to = white_from)
```

Arguments

`colour`	A numeric matrix (or an object coercible to one) with colours encoded in the rows and the different colour space values in the columns. For all colourspaces except `'cmyk'` this will mean a matrix with three columns - for `'cmyk'` it means four columns.
`from, to`	The input and output colour space. Allowed values are: "cmy", "cmyk", "hsl", "hsb", "hsv", "lab", "hunterlab", "lch", "luv", "rgb", "xyz", "yxy"
`white_from, white_to`	The white reference of the from and to colour space. Will only have an effect for relative colour spaces such as Lab and luv. Any value accepted by `as_white_ref()` allowed.

Value

A numeric matrix with the same number of rows as `colour` and either 3 or 4 columns depending on the value of `to`. If `colour` is given as a `data.frame` the output will be a data.frame as well

Note

This function and `convertColor()` are not numerically equivalent due to rounding errors, but for all intend and purpose they give the same results.

See Also

`grDevices::convertColor()`, `grDevices::col2rgb()`

Examples

```
spectrum <- t(col2rgb(rainbow(10)))
convert_colour(spectrum, 'rgb', 'lab')
```

Index