The Journal of Open Source Software

colorspace: A Python Toolbox for Manipulating and Assessing Colors and Palettes

Reto Stauffer [1,2] **and Achim Zeileis** [1]

1 Department of Statistics, Universität Innsbruck, Austria **2** Digital Science Center, Universität Innsbruck, Austria

DOI: 10.21105/joss.07120

Software

- Review
- Repository
- Archive

Editor: Julia Romanowska

Reviewers:

- @hollowscene
- @dmreagan

Submitted: 29 July 2024
Published: 29 October 2024

Summary

The Python *colorspace* package provides a toolbox for mapping between different color spaces, which can then be used to generate a wide range of perceptually-based color palettes for qualitative or quantitative (sequential or diverging) information. These palettes (as well as any other sets of colors) can be visualized, assessed, and manipulated in various ways, e.g., by color swatches, emulating the effects of color vision deficiencies, or depicting the perceptual properties. Finally, *colorspace* integrates seamlessly with standard Python graphics packages like *matplotlib*, *seaborn*, and *plotly*, making it a valuable resource for both developers and practitioners to customize, assess, and implement color palettes in their data visualization workflows.

Statement of need

Color is an integral element of visualizations and graphics and is essential for communicating (scientific) information. However, colors need to be chosen carefully so that they support the information displayed for all viewers (see e.g., Tufte, 1990; Ware, 2004; Wilke, 2019). Therefore, suitable color palettes have been proposed in the literature (e.g., Brewer, 1999; Crameri et al., 2020; Ihaka, 2003) and many software packages transitioned to better color defaults over the last decade. A prominent example from the Python community is *matplotlib* 2.0 (Hunter et al., 2017), which replaced the classic "jet" palette (a variation of the infamous "rainbow") by the perceptually-based "viridis" palette. Hence a wide range of useful palettes for different purposes is provided in a number of Python packages today, including *cmcramery* (Rollo, 2024), *colormap* (Cokelaer, 2024), *colormaps* (Patel, 2024), *matplotlib* (Hunter, 2007), *palettable* (Davis, 2023), and *seaborn* (Waskom, 2021).

However, colors are provided as a fixed set in most graphics packages. While this makes it easy to use them in different applications, it is usually not easy to modify the perceptual properties or to set up new palettes following the same principles. The *colorspace* package addresses this by supporting color descriptions using different color spaces (hence the package name), including some that are based on human color perception. One notable example is the Hue-Chroma-Luminance (HCL) model, which represents colors by coordinates on three perceptually-based axes: hue (type of color), chroma (colorfulness), and luminance (brightness). Selecting colors along paths along these axes allows for intuitive construction of palettes that closely match many of the palettes provided in the packages listed above.

In addition to functions and interactive apps for HCL-based colors, the *colorspace* package also offers functions and classes for handling, transforming, and visualizing color palettes (from any source). In particular, this includes the simulation of color vision deficiencies (Machado et al., 2009) but also contrast ratios, desaturation, lightening/darkening, etc.

The *colorspace* Python package was inspired by the eponymous R package (Zeileis et al., 2020). It comes with extensive documentation at https://retostauffer.github.io/python-colorspace/, including many practical examples. The package complements existing graphics packages in Python both for casual users and data visualization experts. Selected highlights are presented in the following, motivating its usefulness for various kinds of graphics in different fields of application and research.

Key functionality

HCL-based color palettes

The key functions and classes for constructing color palettes using hue-chroma-luminance paths (and then mapping these to hex codes) are:

- `qualitative_hcl`: For qualitative or unordered categorical information, where every color should receive a similar perceptual weight.
- `sequential_hcl`: For ordered/numeric information from high to low (or vice versa).
- `diverging_hcl`: For ordered/numeric information around a central neutral value, where colors diverge from neutral to two extremes.

These functions provide a range of named palettes inspired by well-established packages but actually implemented using HCL paths. Additionally, the HCL parameters can be modified or new palettes can be created from scratch.

As an example, Figure 1 depicts color swatches for four viridis variations. The first, `pal1`, sets up the palette from its name. It is identical to the second, `pal2`, which employes the HCL specification directly: the hue ranges from purple (300) to yellow (75), colorfulness (chroma) increases from 40 to 95, and luminance (brightness) from dark (15) to light (90). The `power` parameter chooses a linear change in chroma and a slightly nonlinear path for luminance.

In `pal3` and `pal4`, the most HCL properties are kept the same but some are modified: `pal3` uses a triangular chroma path from 40 via 90 to 20, yielding muted colors at the end of the palette. `pal4` just changes the starting hue for the palette to green (200) instead of purple. All four palettes are visualized by the `swatchplot` function from the package.

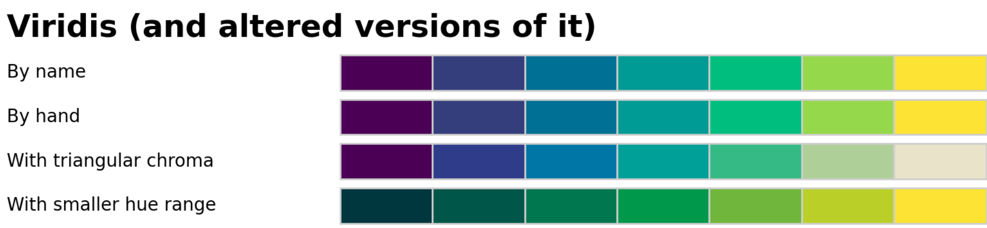

Figure 1: Swatches of four HCL-based sequential palettes: `pal1` is the predefined HCL-based viridis palette, `pal2` is identical to `pal2` but created "by hand" and `pal3` and `pal4` are modified versions with a triangular chroma paths and reduced hue range, respectively.

The objects returned by the palette functions provide a series of methods, e.g., `pal1.settings` for displaying the HCL parameters, `pal1(3)` for obtaining a number of hex colors, or `pal1.cmap()` for setting up a *matplotlib* color map, among others.

```
from colorspace import palette, sequential_hcl, swatchplot
pal1 = sequential_hcl(palette = "viridis")
pal2 = sequential_hcl(h = [300, 75], c = [40, 95], l = [15, 90],
                      power = [1., 1.1])
pal3 = sequential_hcl(palette = "viridis", cmax = 90,  c2 = 20)
```

The Journal of Open Source Software

```
pal4 = sequential_hcl(palette = "viridis", h1 = 200)
swatchplot({"Viridis (and altered versions of it)": [
                palette(pal1(7), "By name"),
                palette(pal2(7), "By hand"),
                palette(pal3(7), "With triangular chroma"),
                palette(pal4(7), "With smaller hue range")
            ]}, figsize = (8, 1.75));
```

An overview of the named HCL-based palettes in *colorspace* is depicted in Figure 2.

```
from colorspace import hcl_palettes
hcl_palettes(plot = True, figsize = (20, 15))
```

Figure 2: Overview of the predefined (fully customizable) HCL color palettes.

Palette visualization and assessment

To better understand the properties of palette `pal4`, defined above, Figure 3 shows its HCL spectrum (left) with separate lines for the hue, chroma, and luminance coordinates and the corresponding path through the three-dimensional HCL space (right) where hue co-varies along with chroma and luminance.

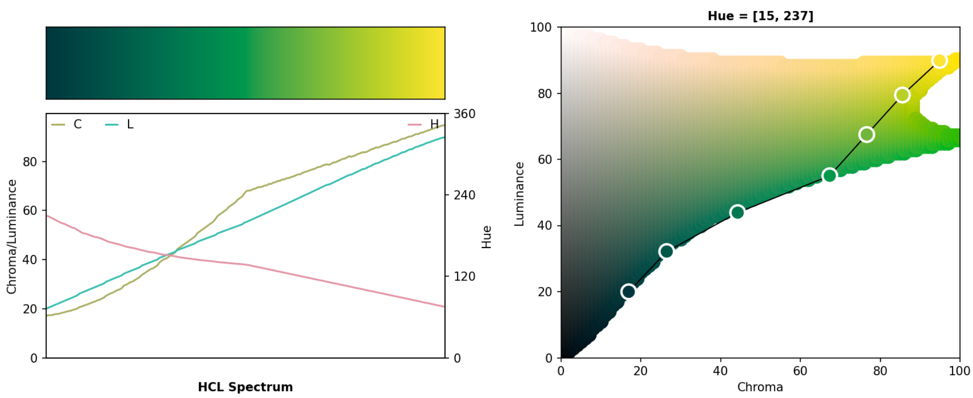

Figure 3: Hue-chroma-luminance spectrum plot (left) and corresponding path in the chroma-luminance coordinate system (where hue changes with luminance) for the custom sequential palette `pal4`.

The spectrum in the first panel shows how the hue (right axis) changes from about 200 (green) to 75 (yellow), while chroma and luminance (left axis) increase from about 20 to 95. Note that the kink in the chroma curve for the greenish colors occurs because such dark greens cannot have higher chromas when represented through RGB-based hex codes. The same is visible in the second panel where the path moves along the outer edge of the HCL space.

```
pal4.specplot(figsize = (5, 5));
pal4.hclplot(n = 7, figsize = (5, 5));
```

Color vision deficiency

Another important assessment of a color palette is how well it works for viewers with color vision deficiencies. This is exemplified in Figure 4, which depicts a demo plot (heatmap) under "normal" vision (left), deuteranomaly (colloquially known as "red-green color blindness", center), and desaturated (gray scale, right). The palette in the top row is the traditional fully-saturated RGB rainbow, deliberately selected here as a palette with poor perceptual properties. It is contrasted with a perceptually-based sequential blue-yellow HCL palette in the bottom row.

The sequential HCL palette is monotonic in luminance so that it is easy to distinguish high-density and low-density regions under deuteranomaly and desaturation. However, the rainbow is non-monotonic in luminance and parts of the red-green contrasts collapse under deuteranomaly, making it much harder to interpret correctly.

```
from colorspace import rainbow, sequential_hcl
col1 = rainbow(end = 2/3, rev = True)(7)
col2 = sequential_hcl("Blue-Yellow", rev = True)(7)

from colorspace import demoplot, deutan, desaturate
import matplotlib.pyplot as plt
fig, ax = plt.subplots(2, 3, figsize = (9, 4))
demoplot(col1, "Heatmap", ax = ax[0,0], ylabel = "Rainbow", title = "Original")
demoplot(col2, "Heatmap", ax = ax[1,0], ylabel = "HCL (Blue-Yellow)")
demoplot(deutan(col1), "Heatmap", ax = ax[0,1], title = "Deuteranope")
demoplot(deutan(col2), "Heatmap", ax = ax[1,1])
demoplot(desaturate(col1), "Heatmap", ax = ax[0,2], title = "Desaturated")
demoplot(desaturate(col2), "Heatmap", ax = ax[1,2])
plt.show()
```

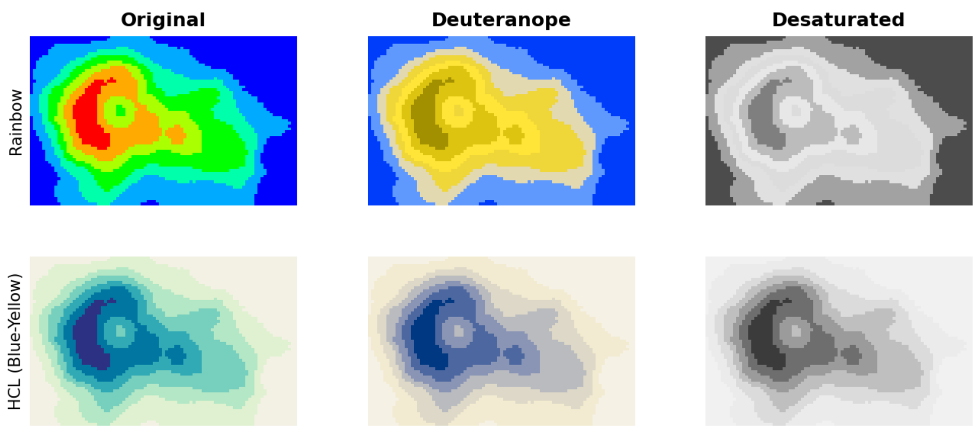

Figure 4: Example of color vision deficiency emulation and color manipulation using a heatmap. Top/bottom: RGB rainbow based palette and HCL based sequential palette. Left to right: Original colors, deuteranope color vision, and desaturated representation.

Integration with Python graphics packages

To illustrate that *colorspace* can be easily combined with different graphics workflows in Python, Figure 5 shows a heatmap (two-dimensional histogram) from *matplotlib* and multi-group density from *seaborn*. The code below employs an example data set from the package (using *pandas*) with daily maximum and minimum temperature. For *matplotlib* the colormap (`.cmap()`; `LinearSegmentedColormap`) is extracted from the adapted viridis palette `pal3` defined above. For *seaborn* the hex codes from a custom qualitative palette are extracted via `.colors(4)`.

```
from colorspace import dataset, qualitative_hcl
import matplotlib.pyplot as plt
import seaborn as sns

df = dataset("HarzTraffic")

fig = plt.hist2d(df.tempmin, df.tempmax, bins = 20,
                 cmap = pal3.cmap().reversed())
plt.title("Joint density daily min/max temperature")
plt.xlabel("minimum temperature [deg C]")
plt.ylabel("maximum temperature [deg C]")
plt.show()

pal = qualitative_hcl("Dark 3", h1 = -180, h2 = 100)
g = sns.displot(data = df, x = "tempmax", hue = "season", fill = "season",
                kind = "kde", rug = True, height = 4, aspect = 1,
                palette = pal.colors(4))
g.set_axis_labels("temperature [deg C]")
g.set(title = "Distribution of daily maximum temperature given season")
plt.show()
```

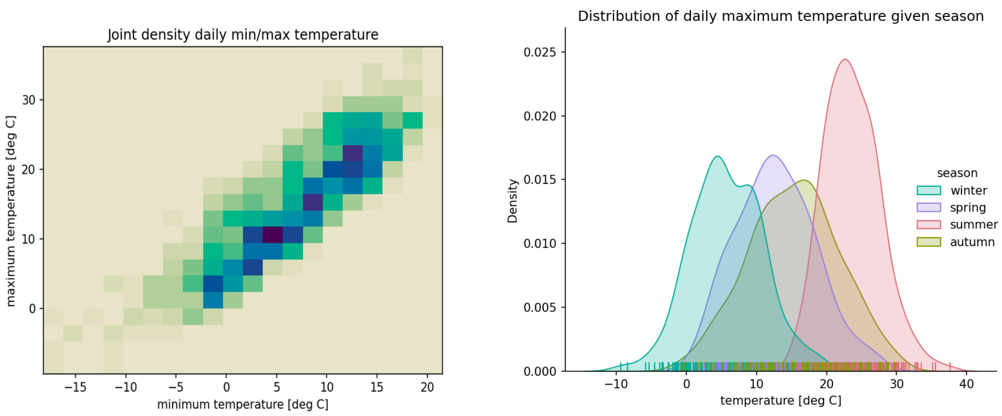

Figure 5: Example of a `matplotlib` heatmap and a `seaborn` density using custom HCL-based colors.

Dependencies and availability

The *colorspace* package is available from PyPI at https://pypi.org/project/colorspace. It is designed to be lightweight, requiring only *numpy* (Harris et al., 2020) for the core functionality. Only a few features rely on *matplotlib*, *imageio* (Klein et al., 2024), and *pandas* (The Pandas Development Team, 2024). More information and an interactive interface can be found on https://hclwizard.org/. Package development is hosted on GitHub at https://github.com/retostauffer/python-colorspace. Bug reports, code contributions, and feature requests are warmly welcome.

References

Brewer, C. A. (1999). Color use guidelines for data representation. *Proceedings of the Section on Statistical Graphics, American Statistical Association*, 55–60.

Cokelaer, T. (2024). *Colormap* (Version v1.1.0). Python Package Index (PyPI). https://pypi.org/project/colormap/

Crameri, F., Shephard, G. E., & Heron, P. J. (2020). The misuse of colour in science communication. *Nature Communications*, *11*(5444), 1–10. https://doi.org/10.1038/s41467-020-19160-7

Davis, M. (2023). *palettable: Color palettes for Python* (Version v3.3.3). Python Package Index (PyPI). https://pypi.org/project/palettable/

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/mcse.2007.55

Hunter, J. D., Dale, D., Firing, E., Droettboom, M., & the Matplotlib Development Team. (2017). *What's new in matplotlib 2.0, changes to the default style*. https://matplotlib.org/stable/users/prev_whats_new/dflt_style_changes.html

Ihaka, R. (2003). Colour for presentation graphics. In K. Hornik, F. Leisch, & A. Zeileis (Eds.), *Proceedings of the 3rd international workshop on distributed statistical computing, vienna, austria*. https://www.R-project.org/conferences/DSC-2003/Proceedings/Ihaka.pdf

Klein, A., Wallkötter, S., Silvester, S., Rynes, A., actions-user, Müller, P., Nunez-Iglesias, J., Harfouche, M., Schrangl, L., Dennis, Lee, A., Pandede, McCormick, M., OrganicIrradiation, Rai, A., Ladegaard, A., van Kemenade, H., Smith, T. D., Vaillant, G., … Singleton, J. (2024). *Imageio/imageio* (Version v2.34.2). Zenodo. https://doi.org/10.5281/zenodo.12514964

Machado, G. M., Oliviera, M. M., & Fernandes, L. A. F. (2009). A physiologically-based model for simulation of color vision deficiency. *IEEE Transactions on Visualization and Computer Graphics*, *15*(6), 1291–1298. https://doi.org/10.1109/tvcg.2009.113

Patel, P. (2024). *Colormaps* (Version v0.4.2). Python Package Index (PyPI). https://pypi.org/project/colormaps/

Rollo, C. (2024). *cmcrameri: Python wrapper around Fabio Crameri's perceptually uniform colormaps* (Version v1.9). Python Package Index (PyPI). https://pypi.org/project/cmcrameri/

The Pandas Development Team. (2024). *pandas-Dev/Pandas: pandas* (Version v2.2.2). Zenodo. https://doi.org/10.5281/zenodo.10957263

Tufte, E. (1990). *Envisioning information*. Graphics Press.

Ware, C. (2004). Color. In *Information visualization: Perception for design* (pp. 103–149). Morgan Kaufmann Publishers Inc.

Waskom, M. L. (2021). seaborn: Statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. https://doi.org/10.21105/joss.03021

Wilke, C. O. (2019). *Fundamentals of data visualization*. O'Reilly Media. ISBN: 1492031089

Zeileis, A., Fisher, J. C., Hornik, K., Ihaka, R., McWhite, C., Murrell, P., Stauffer, R., & Wilke, C. O. (2020). colorspace: A toolbox for manipulating and assessing colors and palettes. *Journal of Statistical Software*, *96*(1), 1–49. https://doi.org/10.18637/jss.v096.i01