The Journal of Open Source Software

Qualpal: Qualitative Color Palettes for Everyone

Johan Larsson [1]

1 Department of Mathematical Sciences, University of Copenhagen, Denmark

DOI: 10.21105/joss.08936

Software

- Review
- Repository
- Archive

Editor: Daniel S. Katz

Reviewers:

- @dmreagan
- @clauswilke

Submitted: 19 August 2025
Published: 16 October 2025

Summary

Qualpal is a C++ library, command-line tool, R package, and web app for creating qualitative color palettes with maximally distinct colors. It helps scientists and anyone working with data visualization choose colors that remain clear and accessible. Qualpal features flexible palette generation from multiple input formats, such as the HSL and LCH_{ab} color spaces or fixed sets of RGB colors, and can adapt palettes to color vision deficiencies (CVD) across the full dichromacy spectrum at any severity. At its core, Qualpal is a lightweight C++ library with no external dependencies, making it easy to integrate into other software and programming languages.

Statement of need

Effective visualization of categorical data requires color palettes with easily distinguishable colors—for both people with normal color vision and those with CVD. Designing a palette is therefore an optimization problem, where the goal is to maximize the minimum difference among the colors in the palette in order to make the palette as distinct as possible. This is a non-trivial problem, since the number of possible palettes grows exponentially with the number of colors in the palette. And as the number of colors in a palette *increases*, the minimum distance between colors necessarily *decreases*, since the colors must be spread out more densely in the color space. As a result, any given palette can, at best, be optimal only for a particular size. And since users may also have specific requirements in terms of, for instance, hue, lightness, saturation, adaptation to CVD, or background color, it is impossible to provide a set of fixed palettes to cover these needs. Therefore, there is a need for flexible palette generation tools that can accommodate a wide range of user requirements and preferences.

This problem has been tackled by, for instance, Glasbey et al. (2007), who developed an algorithm based on simulated annealing that is available in the Python package Glasbey (McInnes, 2025). Other tools include iWantHue (Jacomy, 2013/2025), Colorgorical (Gramazio et al., 2016), distinctipy (Roberts et al., 2019/2024), and Palettailor (Lu et al., 2021). All of these packages rely on some metric to measure the distance between colors and use some form of optimization algorithm, such as simulated annealing, to find a set of colors that maximizes the minimum distance between them in the palette. We summarize these existing packages and their features in Table 1 and Table 2, respectively.

Table 1: Summary of related work and packages, in terms of their algorithms, color difference metrics, input types, and implementation languages.

Package	Algorithm	Metrics	Input	Language
`Glasbey`	Simulated annealing	CIE76	LCH_{ab}, Fixed	`Python`
`iWantHue`	k-means, force vector	CIE76	LCH_{ab}	`JavaScript`
`Colorgorical`	Random sampling	CIEDE2000	LCH_{ab}	`Python, C`

Package	Algorithm	Metrics	Input	Language
`distinctipy`	Random sampling	L$_{uv}$ approx	Pastel filter	`Python`
`Palettailor`	Simulated annealing	CIEDE2000	Hue, lightness	`JavaScript`
`Qualpal`	Farthest points	CIEDE2000, DIN99d, CIE76	HSL, LCH$_{ab}$, Fixed	`C++`

All of these existing packages have different strengths and weaknesses. `qualpal` is, however, the first `C++` library, CLI tool, and R package for generating qualitative color palettes. It is also the first package to implement a farthest point sampling algorithm for generating qualitative color palettes, and the only one to support multiple types of CVD. In addition, it is the only package to support input from the HSL color space, which represents an intuitive way to specify colors in terms of hue, saturation, and lightness. It also supports multiple metrics for measuring color distance, including CIEDE2000 (Sharma et al., 2005) and DIN99d (Cui et al., 2002), where the former is the current standard for color difference advocated by the International Commission on Illumination (CIE) and the latter is based on Euclidean distances in the DIN99d color space, which improves upon the CIE76 metric that uses the CIE$_{L}$a$_{b}$ color space.

Table 2: Summary of features of existing packages, in terms of color vision adaptation (CVD), availability of a web app (Web), command-line interface (CLI), ability to extend existing palettes, option to adapt to a background color, and possibility to create palettes with related blocks (such as pairs).

Package	CVD	Web	CLI	Extend	Back-ground	Blocks
`Glasbey`	✓			✓		✓
`iWantHue`	✓	✓[1]				✓
`Colorgorical`		✓[2]				
`distinctipy`	✓					
`Palettailor`		✓[3]			✓	
`Qualpal`	✓	✓[4]	✓	✓	✓	

Examples

In this section we show some examples of palettes generated with Qualpal. We begin with a palette generated from candidate colors from part of the HSL color space, defined by hue in $[0^\circ, 60^\circ]$ and $[170, 360)$[5], saturation in $[0, 0.7]$, and lightness in $[0.2, 0.8]$. This produces the palette shown in Figure 1. The command to generate this palette is:

```
qualpal -n 5 -i colorspace "-190:60" "0:0.7" "0.2:0.8"
```

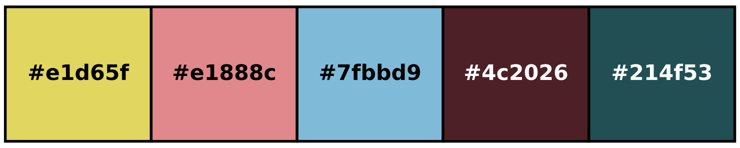

Figure 1: A palette of five colors generated from input colors as a subspace of the HSL colorspace.

[1] https://medialab.github.io/iwanthue/
[2] http://vrl.cs.brown.edu/color (but down at the time of writing)
[3] https://iamkecheng.github.io/palettailor/
[4] https://qualpal.cc
[5] Note that we specify `-190` as the starting point in Qualpal to wrap around the hue wheel.

Next, we show another palette generated from similar input from the HSL color space. But this time we adapt the palette to the color vision deficiency types protanomaly (at 80% severity) and tritanomaly (at full severity). The resulting palette is shown in Figure 2. Here, we show how to generate this palette using the C++ library interface:

```
#include <qualpal.h>

auto pal = qualpal::Qualpal{}
             .setInputColorspace({ -190, 60 }, { 0, 0.7 }, { 0.2, 0.8 })
             .setCvd({ { "protan", 0.8 }, { "tritan", 1.0 } })
             .generate(4);
```

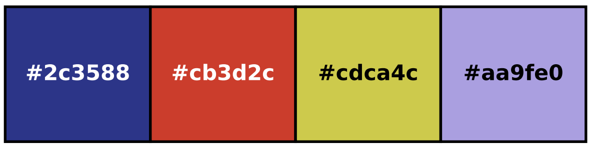

Figure 2: A palette of four colors, generated from colors sampled from a portion of the HSL color space, and adapted to protanomaly and tritanomaly.

Qualpal also features a set of built-in palettes. In this example, we begin with a palette derived from Johannes Vermeer's *Girl with a Pearl Earring*, and pick four colors from it. We also optimize the palette to distinguish it from a background color of white. The resulting palette is displayed in Figure 3. This time, we have used the R package qualpalr:

```
library(qualpalr)
pal <- qualpal(4, "Vermeer:PearlEarring", bg = "white")
```

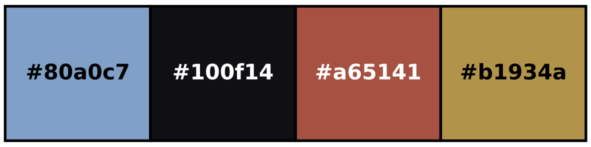

Figure 3: A palette derived from the colors in Vermeer's *Girl with a Pearl Earring*, optimized to be distinguished from a white canvas.

Finally, we show how to extend an existing palette, in this case one consisting of three colors inspired by the Bauhaus art movement. We consider a fixed set of four input colors from the Okabe-Ito palette (Okabe & Ito, 2008) as input. The result is shown in Figure 4 and the CLI command we used to generate this palette is:

```
qualpal -n 4 \
  -i hex "#000000" "#e69f00" "#56b4e9" "#009e73" \
  --extend "#B33A3A" "#2F5DA5" "#E1B84A"
```

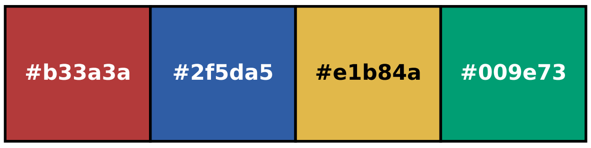

Figure 4: An example of extending an existing palette of three colors (blue, red, and yellow), with candidates from the Okabe-Ito palette.

Summary of the algorithm

Qualpal begins with a set of input colors. These can be a fixed set of colors provided by the user, one of the built-in palettes, or a subspace in the LCH_{ab} or HSL color spaces. In the latter case, we use a quasi-random Halton sequence (Halton, 1964) to distribute colors throughout this subspace. The input colors are then (optionally) projected into a color space corresponding to one or several CVD types, such as protanopia or deuteranopia, using simulation methods described by Machado et al. (2009).

Next, we compute a full color distance matrix for the colors in the input set, using the CIEDE2000 (Sharma et al., 2005) color difference metric by default. Finally, we run a farthest point sampling algorithm loosely based on the work by Schlömer et al. (2011), which iteratively swaps colors between a candidate palette and its complement set until no swap can improve the minimum distance between colors in the candidate palette. Optionally, a background color can be included in this step, in which case the palette is optimized to be distinct from it. The algorithm is deterministic (unlike the other algorithms from Table 1) and takes roughly 0.1 seconds to generate a 10-color palette from a set of 1000 input colors on a modern laptop.

Acknowledgements

Bruce Lindbloom's webpage has been instrumental in the development of Qualpal, serving as a vital reference for color space conversions and color difference calculations.

References

Cui, G., Luo, M. R., Rigg, B., Roesler, G., & Witt, K. (2002). Uniform colour spaces based on the DIN99 colour-difference formula. *Color Research & Application*, *27*(4), 282–290. https://doi.org/10.1002/col.10066

Glasbey, C., van der Heijden, G., Toh, V. F. K., & Gray, A. (2007). Colour displays for categorical images. *Color Research & Application*, *32*(4), 304–309. https://doi.org/10.1002/col.20327

Gramazio, C. C., Laidlaw, D. H., & Schloss, K. B. (2016). Colorgorical: Creating discriminable and preferable color palettes for information visualization. *IEEE Transactions on Visualization and Computer Graphics*, *23*(1), 521–530. https://doi.org/10.1109/TVCG.2016.2598918

Halton, J. H. (1964). Algorithm 247: Radical-inverse quasi-random point sequence. *Commun. ACM*, *7*(12), 701–702. https://doi.org/10.1145/355588.365104

Jacomy, M. (2025). *iWantHue*. Médialab Sciences Po. https://github.com/medialab/iwanthue (Original work published 2013)

Lu, K., Feng, M., Chen, X., Sedlmair, M., Deussen, O., Lischinski, D., Cheng, Z., & Wang, Y. (2021). Palettailor: Discriminable colorization for categorical data. *IEEE Transactions on Visualization and Computer Graphics*, *27*(02), 475–484. https://doi.org/10.1109/TVCG.2020.3030406

Machado, Gustavo. M., Oliveira, Manuel. M., & Fernandes, Leandro. A. (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

McInnes, L. (2025). *Glasbey* (Version 0.3.0). https://github.com/lmcinnes/glasbey

Okabe, M., & Ito, K. (2008, September 24). *Color universal design (CUD): How to make figures and presentations that are friendly to colorblind people* [JFLY]. https://jfly.uni-koeln.de/color/

Roberts, J., Crall, J., Ang, K.-M., & Brandt, Y. (2024). *Distinctipy* (Version v1.3.4). Zenodo. https://doi.org/10.5281/zenodo.10480933 (Original work published 2019)

Schlömer, T., Heck, D., & Deussen, O. (2011). Farthest-point optimized point sets with maximized minimum distance. *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*, 135–142. https://doi.org/10.1145/2018323.2018345

Sharma, G., Wu, W., & Dalal, E. N. (2005). The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. *Color Research & Application*, *30*(1), 21–30. https://doi.org/10.1002/col.20070