Palettailor: Discriminable Colorization for Categorical Data

Kecheng Lu, Mi Feng, Xin Chen, Michael Sedlmair, Oliver Deussen, Dani Lischinski, Zhanglin Cheng, Yunhai Wang

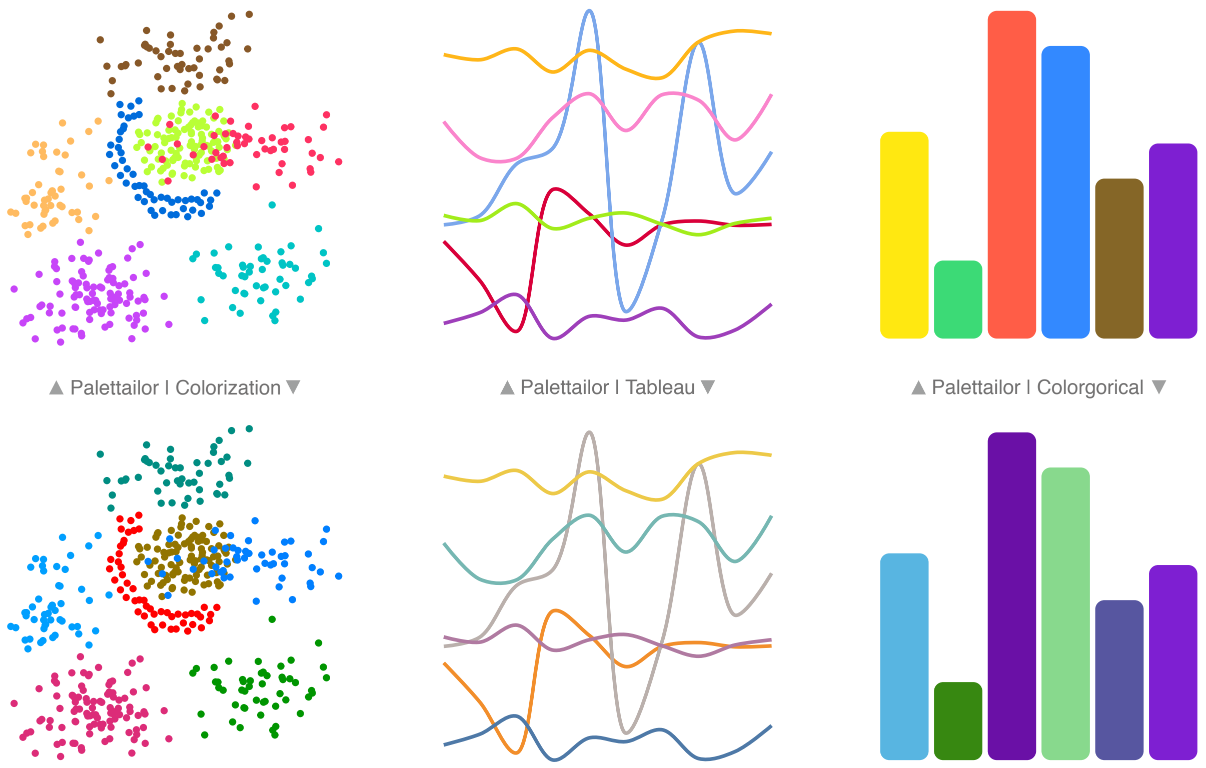


Fig. 1. Results for different types of categorical data visualizations: (left) Palettailor versus Colorization [6]; (center) Palettailor versus Tableau [30]; (right) Palettailor versus Colorgorical [10]. Our system integrates the creation and the assignment of colours to a visualization in a data-aware manner.

Abstract—We present an integrated approach for creating and assigning color palettes to different visualizations such as multi-class scatterplots, line, and bar charts. While other methods separate the creation of colors from their assignment, our approach takes data characteristics into account to produce color palettes, which are then assigned in a way that fosters better visual discrimination of classes. To do so, we use a customized optimization based on simulated annealing to maximize the combination of three carefully designed color scoring functions: point distinctness, name difference, and color discrimination. We compare our approach to state-of-the-art palettes with a controlled user study for scatterplots and line charts, furthermore we performed a case study. Our results show that Palettailor, as a fully-automated approach, generates color palettes with a higher discrimination quality than existing approaches. The efficiency of our optimization allows us also to incorporate user modifications into the color selection process.

Index Terms—Color Palette, Discriminability, Multi-Class Scatterplot, Line Chart, Bar Chart

- *K. Lu, X. Chen, Y. Wang are with Shandong University. E-mail: {lukecheng0407, chenxin199634, cloudseawang}@gmail.com.*
- *M. Feng is with Twitter Inc. E-mail: miamfeng@gmail.com.*
- *K. Lu and Z. Cheng are with Shenzhen VisuCA Key Lab, SIAT, China. E-mail: zl.cheng@siat.ac.cn.*
- *M. Sedlmair is with VISUS, University of Stuttgart, Germany. E-mail: michael.sedlmair@visus.uni-stuttgart.de.*
- *O. Deussen is with Konstanz University, Germany and Shenzhen VisuCA Key Lab, SIAT, China. E-mail: oliver.deussen@uni-konstanz.de.*
- *D. Lischinski is with Hebrew University, Jerusalem, Israel. E-mail: danix@mail.huji.ac.il*
- *Y. Wang and Z. Cheng are joint corresponding authors.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

1 Introduction

Visualizing categorical data in statistical graphics such as bar charts, line charts, or scatterplots is most commonly realized by encoding each category (or class) with a unique color. One major task during visual analysis is then to discriminate between the different classes. While it is well-known that class discriminability is strongly influenced by the assigned colors [9, 17], finding an appropriate set of colors for the different classes in a specific visualization is still a complex and time-consuming endeavor, even for experts.

The most common way to obtain an appropriate color mapping is to find a good color palette first and then assign the colors to classes in the best possible way. To ease this procedure, a few color palette tools have been provided, such as ColorBrewer [11] or Colorgorical [10], which allow users to select highly discriminable and preferable palettes. Since the creation of such palettes ignores the specific data of a visualization, a good palette might still not be optimal to visually discriminate classes in different forms of visualization. Hence, users often need to try different palettes and color assignment schemes until the desired result is achieved. Recently, Wang et al. [36] proposed a method that automatically assigns colors of a given palette to classes of multi-class scatterplots by maximizing their discriminability. This technique enables users to bypass the second stage of the standard color assignment process, but it is limited to scatterplots and still requires the author to select a good palette. In contrast, Chen et al. [6] proposed an automatic color selection approach for multi-class scatterplots by searching discriminative colors in the a* and b* channel of the CIELAB space. However, leaving out L* channel often does not allow to find colors with high enough discriminability, especially when the number of classes is large. Since such an approach directly colorizes multi-class scatterpots without any input palette, we refer to it as *Colorization* in this paper.

To fill this gap, we propose Palettailor, a data-aware color palette generation framework, that automatically generates categorical palettes with maximized discriminability for different visualization types. To achieve this goal, we adapted three color-scoring functions: Point Distinctness [36], Name Difference [13], and the Color Discrimination constraint. The first function is measured from the visualizations, while the last two terms from the palettes. By doing so, we combine the two stages of palette selection and color assignment to a single framework. Users only need to specify weights of the different color-scoring functions and background color to create suitable palettes. In terms of these functions, our notion of "data-aware" mainly refers to the spatial distribution and cardinality of categories in a given scatterplot, while data semantics can be reflected by, e.g., constraining color names (see Section 5.1). As far as we know, few existing color palette generation tools take all these factors into account.

To realize this idea, we customize state-of-the-art class separation measures [2] for designing color-based class discriminability. Wang et al. [36] measure Point Distinctness by using a K-Nearest-Neighbor-Graph with taking the two nearest neighbors per point, but we hypothesize that this might not work for all cases since the KNN method would only connect with k neighbors and might not be able to accurately characterize local class separability. Instead, we suggest to use an α-shape [34] graph that connects a set of points within an α-ball so as to better characterize the local discriminability.

With such a pre-computed geometry-based class separation, an optimal color palette can be found by evaluating all possible solutions and then ranking them accordingly. The color space, however, is too large for an exhaustive investigation and so we propose a customized simulated annealing algorithm [26] to rapidly and efficiently obtain a near-optimal solution. In doing so, we can generate the palettes for scatterplots and other types of visualizations with 40 classes in less than 15 seconds.

We evaluated our approach through carefully designed scatterplots and line charts by comparing our colorized results with the ones produced by the state-of-the-art palettes (e.g.,Tableau [30] and Colorgorical). We first conducted a pilot study for each visualization type to justify whether our design is valid. We then carried out an online study with three tasks for a scatterplot: the first two tasks investigated how well our generated palettes help users to discriminate classes, while the third task aimed at finding out if our results align with user preferences. Similarly, we carried out another online study for a line chart with a discrimination task and again measuring the alignment with user preferences. The results show that our method is able to produce palettes optimized for class discrimination and aligned with user preferences for different chart types that are more effective than state-of-the-art palettes in most cases.

On the basis of our method, we implemented a web-based palette generation tool that allows users to generate suitable palettes for multi-class scatterplots, but also for other categorical visualizations such as bar and line charts. In summary, the main contributions of this paper are:

- We propose a data-aware approach based on simulated annealing for automatically generating color palettes for different visualization types. It is based on user-defined weights for three different scoring functions and extends state-of-the-art separation measures (Section 3).
- We quantitatively evaluate the resulting color palettes and visualizations, and compare the results with state-of-the-art color palettes; online user studies show the usefulness of our approach (Section 4).
- We present an interactive tool[1] that demonstrates the practical applicability of our method and helps exploring categorical information visualizations (Section 5).

2 Related Work

Since we focus on color palette generation, related work is divided into color palette creation and color palette optimization. For the generation of color ramps, a slightly different color assignment problem, we refer readers to recent surveys [22, 24, 31, 37] .

2.1 Color Palette Creation

Creating a categorical color palette that enables viewers to distinguish between data elements is a crucial task for many visualizations. Previous attempts have derived practical guidelines based on perceptual constraints, such as "colors should be well separated" [12], "should not compete with each other" [32], and "should not be unappealing" [38]. Following these guidelines, a few interactive color palette creation systems have been developed. Healey [12] proposed to choose representative colors from ten hue regions in Munsell's color space, while maximizing the perceptual distances between them. Maxwell [20] selects colors that maximize the discriminability between classes, but also takes into account their spatial distribution. Their approach has two major limitations. One is that the generated colors are not aesthetically pleasing or easy to see; and the other is that it defines perceptual distances based on the so-called maximum scaled difference, which might not align well with human perception. We want to find a much more general solution. Colorgorical [10] also overcomes this drawback, but does not take into account the data distribution. The resulting palettes might thus not allow to discriminate classes in given data or in a particular visualization. The Colorization method proposed by Chen et al. [6] automatically selects colors for multi-class scatterplots in the CIELAB color space to yield a maximal color distinguishability. This technique targets the same problem as ours, but has three main drawbacks: (i) using density to encode class separability does not lead to an accurate measurement; (ii) leaving L* as a user-adjustable parameter and only optimizing a* and b* values limits the approach to a 2D plane in CIELAB and does not provide enough colors with large color differences; (iii) using Euclidean Distance for computing color distances leads to generated colors that are often at the boundary of the 2d plane due to the optimization. Our method avoids these drawbacks.

Another approach for designing discriminable palettes is to use prefabricated ones. A typical example is ColorBrewer [11], an online tool for generating color palettes optimized for choropleth maps. Although ColorBrewer offers different high-quality palettes for different cardinalities, it does not allow users to customize them. Colorgorical [10] enables users to customize palettes by specifying desired hues, but it does not take the underlying data into account and thus cannot customize colors for specific classes or visualizations. Our data-aware approach inherently offers such flexibility.

2.2 Color Palette Optimization

Once a color palette is selected, users often want to refine the colors until a desired set is achieved and then find a proper assignment to the categories within the data.

Color Optimization. Optimizing a selected color palette can be performed by applying different criteria, such as optimizing data comprehension, aesthetics, energy saving, or helping with color vision deficiency (CVD) [39]. Wang et al. [35] introduced a knowledge-based system that employs established color design rules to optimize chosen

[1] `http://www.palettailor.net`

colors for discriminating spatial structures in 2D and 3D visualizations. Chuang et al. [7] proposed to optimize color palettes with the goal of lowering the energy consumption of display devices, and Machado et al. [19] presented a physiologically-based model for simulating color perception so that the refined palettes can efficiently improve the visualization experience for individuals with CVD.

Lee et al. [17] proposed to measure the visual saliency of each visible point of a class and used this metric to optimize palettes for better class discrimination. This method takes the color contrast against the background into account, however, it aims to optimize the given color palette in terms of spatial adjacency of the given data. In contrast to their work, our approach directly produces discriminable colors for different categorical visualization types such as bar or line charts; furthermore, we also incorporate color contrast with the background into the discriminability measure.

Rather than optimizing palettes for a specific visualization, Fang et al. [8] provided an approach for maximizing perceptual distances among a set of given colors. Although this approach can incorporate various user-defined constraints, it is independent of the given data and thus the produced visualization might not clearly show the different class structures in the data.

Color Assignment. Assigning colors properly to different classes is a crucial step in categorical data visualization. Assuming that categorical values have semantics, Lin et al. [18] propose a method that automatically matches each value to a unique color from a given palette. Setlur and Stone [28] use linguistic information to generate semantically meaningful colors. Since most classes in a visualization might not have clear semantics, Wang et al. [36] allow user modification and customization within their optimization framework. To that end, they take spatial adjacency into account, but require a discriminable palette first. Our approach unifies palette creation and color assignment into a single procedure and is also able to adapt to user wishes. In addition, it supports palette completion and introduces semantics by handling color names. By constraining color selection with user-specified color names, our approach could also be extended to achieve data semantics-based colorization similar to existing approaches [18, 28].

3 Methods

Given categorical data with m classes $C = \{C_1, \cdots, C_m\}$, we assume to have a point set $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ where each $\mathbf{x}_i$ is associated with a class label $l(\mathbf{x}_i)$ and the j-th class (with n_j data points) consist of $\{\mathbf{x}_1^j, \cdots, \mathbf{x}_{n_j}^j\}, j \in M = \{1, \cdots, m\}$, the background color is represented by c_0. We formulate the search for a color palette $P = \{c_1, \cdots, c_m\}$ as an optimization problem with the **objective function** $E(P)$:

$$\arg\max_P E(P) = \omega_0 E_{PD} + \omega_1 E_{ND} + \omega_2 E_{CD}, \tag{1}$$

where each class C_i is assigned a unique color c_i and each weight ω_i is a value range from 0 to 1.

In the following, we will introduce the constraints we use for optimizing the colors in different visualization types: Point Distinctness(PD), Name Difference(ND) and Color Discrimination(CD), and then describe how we solve the overall optimization problem.

3.1 Scoring Functions

Since scatterplots are the most commonly used chart type with points as visual marks, we decided to use them to illustrate our method. The arguments, however, also hold for other types of visualizations, as shown later.

Point Distinctness. Similar to [36], we define the distinctness of a data point x_i from its neighbours using a color mapping P as follows:

$$\alpha(\mathbf{x}_i) = \frac{1}{|\Omega_i|} \sum_{\mathbf{x}_j \in \Omega_i} \Delta\varepsilon(c_i, c_j) g(d(\mathbf{x}_i, \mathbf{x}_j)), \tag{2}$$

where the Ω_i are the nearest neighbors of $\mathbf{x}_i$ with $c_i = P(l(\mathbf{x}_i)), c_j = P(l(\mathbf{x}_j))$ color mapping functions, $\Delta\varepsilon$ is the perceptual color distance

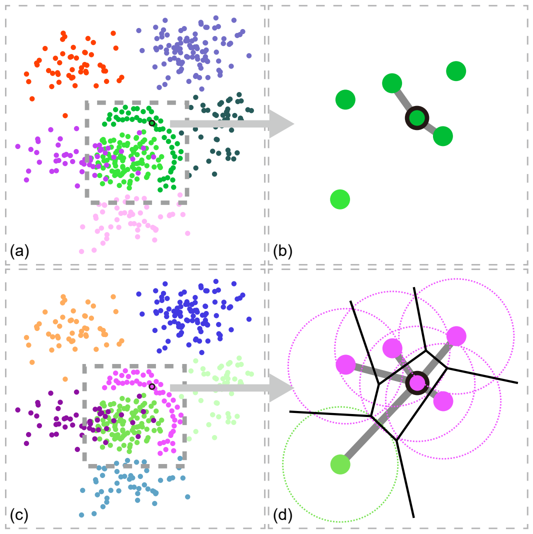


Fig. 2. Comparison of nearest neighbors definitions in KNN graphs and α-Shape graphs. (a) Results generated by a KNN graph using only point distinctness: the generated colors are hard to discriminate; (b) Nearest neighbors of the selected point in the KNN graph; (c) Results generated by α-Shape graph only using point distinctness: the generated colors are easily to discriminate; (d) Nearest neighbors of the selected point in the α-Shape graph.

metric called CIEDE2000 [29] and $g(d(\mathbf{x}_i, \mathbf{x}_j))$ is a distance-based function that assigns large weights to nearby points and small weights to distant points; we used $g(d) = \frac{1}{d}$ in our implementation.

In contrast to Wang et al. [36], we define Ω_i using an α-Shape graph [2] instead of a KNN graph. The reason for this is that nearest neighbors within of an KNN graph may have an arbitrarily large or small distance to the point of interest. Although the distance-based function $g(d)$ helps for reducing the influence of neighbors with large distances, having neighbors with only small distances might result in a colorization that is hard to discriminate. Fig. 2(a) shows an example where the two well-separated clusters enclosed by the rectangle receive similar colors for the KNN-based approach and are thus hard to distinguish. In contrast to this, the α-Shape graph connects only points that are neighbors in the Delaunay graph and in addition their α-balls intersect (see Fig. 2(d)). This property allows us to select the closest neighbors of a given point within a certain radius. Fig. 2(c) shows the colorization result, now the two nearby clusters enclosed by the rectangle can be discriminated easily.

The colorization results for the same scatterplot using these two different graphs are shown in Fig. 2(a,c).Since the two nearby clusters highlighted by the rectangle are well separated and dense, the k nearest neighbors of each point belong to the same class in the KNN graph and the point distinctness is zero; this leads to similar assigned colors for the two clusters. In contrast, point distinctness computed from the α-Shape graph takes into account only neighbors within a given radius and therefore results a discriminable colorization.

Point Distinctness is defined as the sum of the Point Distinctness of all points in each class:

$$E_{PD} = \sum_{j=1}^{m} \sum_{i=1}^{n_j} \alpha(\mathbf{x}_i^j). \tag{3}$$

Name Difference. Two colors might be perceptually different but are referenced by the same name, confusing color selection. For example, the color "Violet" is perceptually different from "Dark Purple", but they are often referred by the same name "Purple", making it difficult to reference them by name in a visualization. To model such name-color associations, Heer and Stone [13] introduce the concept of Name Difference (ND), that measures the probability of two colors having the

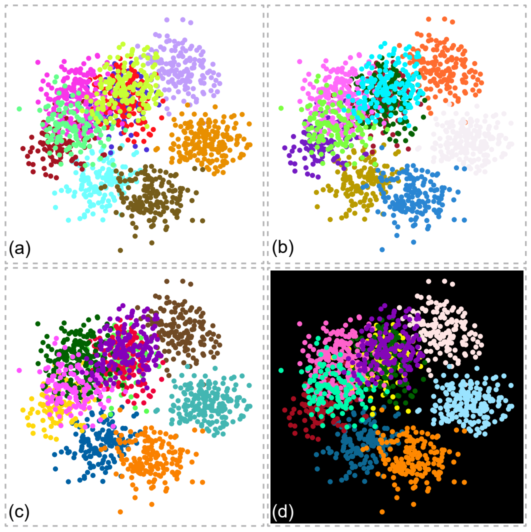


Fig. 3. Colorization results based on different optimization criteria: (a) without Class Discrimination constraint; (b) without considering background color making it hard to see points of one class; (c) integrating the white background into the optimization lets the class appear in a dark color; (d) integrating the black background color into the optimization changes the colorization of many classes in order to create the necessary contrast.

same name. It was used by Colorgorical [10] for generating palettes with perceptually separated colors. For a palette, its ND is defined as the minimal Hellinger distance [13] between the probability distributions of all color name pairs. Another option is to compute the cosine distance, the angle between the two distributions:

$$ND(c_i,c_j) = 1 - cos(T_{c_i},T_{c_j}) = 1 - \frac{T_{c_i}T_{c_j}}{||T_{c_i}||||T_{c_j}||}, \tag{4}$$

Where T is the name count matrix for the color terms, with T_c the row vector for color c. Due to its simplicity and familiarity, we use the cosine distance to measure the name difference of two colors. For a palette, our goal is to compose it as much as possible from colors with different color names, thus its overall Name Difference is defined as the average value of all color pairs:

$$E_{ND} = \frac{2}{m(m-1)} \sum_{i \neq j \in M} ND(c_i,c_j). \tag{5}$$

Color Discrimination. Since the spatial distance between marks might influence the perception of color differences [5], we employ a hard constraint into our optimization: each two colors should have a minimal CIEDE2000 distance of 10, so that each pair of colors can be distinguished even for large spatial distances. Meanwhile, the minimum color distance should be as large as possible to protect the optimization from reaching unsatisfying results. Fig. 3(a) shows that without using the color discrimination constraint classes with small overlap might be assigned improper colors: the green and limegreen classes are hard to distinguish, also the green and cyan classes. In addition, we also take the background color c_0 into consideration to prevent the optimization from generating colors indistinguishable from the background, see Fig. 3(b),(c),(d) for illustration. The formal definition of the Color Discrimination is:

$$E_{DC} = \min_{0 \leq i \neq j \leq m} \Delta\varepsilon(c_i,c_j). \tag{6}$$

To achieve good optimization results we need to balance the three terms. For the Point Distinctness, we use a normalized weight relative to an initial score at the beginning of the optimization taken from a random palette that satisfies basic requirements. Based on out experiments Name Difference is multiplied by a factor of 2.0, while the Color Discrimination constraint is multiplied with a factor of 0.1.

3.2 Extension to Line and Bar Charts

We extended our method to generate color palettes for other categorical visualizations such as line and bar charts. We achieved this by interpreting these charts as special cases of scatterplots, which allowed us to fit them into our optimization framework. We are well aware of the fact that this is a very simple form of implementation, but the results are promising.

Fig. 4 illustrates the process. To colorize a *line chart*, we discretize the lines into equidistant points and process them the same way as the points of a scatterplot. Note that we discretize line segments with large slopes result into more point samples, indicating that such lines need to be discriminated stronger from each other.

A *bar chart* is different from a line chart in that nearest neighbors are pre-defined by the adjacent bars. Thus we treat every bar as a single point class with a center at the middle of the bar; its nearest neighbors are the two adjacent bars. An example is shown in Fig. 4(c), where we took the length of the connecting lines between each two center points as the distance of the corresponding bars. In addition, we used the reverse function $g(d) = \frac{1}{d}$ to scale up distances of bars with similar heights for better distinguishing them. Our technique achieves high-quality results compared to state-of-the-art palettes and is more flexible and steerable, some results can be seen in Fig. 4(b,d). Due to the space limits, we show more results in the supplemental material.

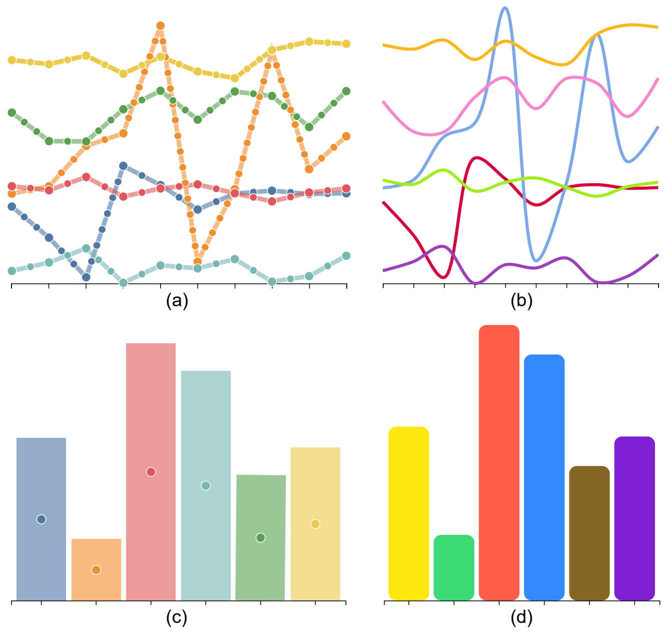


Fig. 4. Converting line and bar charts to point-based representation for colorizing them: (a) each curve in a line chart is discretized into equidistant points; (b) colorized line chart; (c) for a bar chart, the center of each bar is represented as a point and connected to its two adjacent bars for forming the graph; (d) colorized bar chart.

3.3 Simulated Annealing

Our method takes categorical data with m classes as input, then iteratively tries to find an optimized color palette, with each color assigned to a class. Similar to Gramazio et al. [10], we filter out strongly disliked colors whose luminance value is in the range of $L \in [35,75]$ and hue values $H \in [85°,114°]$. To find m colors with maximal energy with respect to Eq. 1, we use Simulated Annealing [1], a stochastic optimization method for finding optima. In comparison to the optimization adopted by Pham and Karaboga [26], Simulated Annealing yields better results at comparable run times, this way also facilitating an interactive generation of palettes.

Algorithm 1 Simulated Annealing for Colorization

```
1: randomly initialization $P_0$
2: best palette $P_{best} = P_0$
3: $t = 0$
4: while $T_t > T_{end}$ do
5:     $Q = P_t$
6:     if $random(0,1) < 0.5$ then
7:         randomly disturb one color of $Q$
8:     else
9:         randomly exchange two colors of $Q$
10:    end if
11:    while $\exists \Delta\varepsilon(c_i, c_j) < \tau; c_i \in Q$ do
12:        randomly disturb $c_i$ or $c_j$ to get a new $Q$
13:    end while
14:    $\Delta E = E(Q) - E(P_t)$
15:    if $\Delta E > 0$ then
16:        $P_{t+1} \leftarrow Q$
17:    else
18:        with probability $exp(\Delta E/T_t)$, $P_{t+1} \leftarrow Q$
19:    end if
20:    if $E(P_{best}) < E(Q)$ then
21:        $P_{best} = Q$
22:    end if
23:    $T_t = \alpha T_t, \quad t \leftarrow t+1$
24: end while
```

Starting with a high "temperature" and an initial guess, the method iteratively updates the palette, while gradually lowering the temperature until convergence is achieved. During this procedure, intermediate results worse than the current iteration can be accepted with a probability related to the "temperature". If the cooling of this temperature is slow, it is more likely to reach the global optimum. Algorithm 1 shows the details of our implementation. It has three customized components: generating a new solution, palette refinement by imposing hard constraints, and finding the optimal colorization. In the following we explain each of these components in detail. In line with suggestion of previous works [15], we use the following values: cooling coefficient $\alpha = 0.99$, initial temperature $T = 100,000$ and end temperature $T_{end} = 0.001$. The effect of different parameter values and number of iterations on the final result is shown in the supplemental material.

Initialization of P (line 1). We found that using random initialization can generate reasonable results in most cases. Although P can also be initialized by any existing good palette, the results are almost the same as the ones produced from random initializations. Since existing palettes also might not have enough colors, P_0 is set to a random initialization by default.

Generating a new solution (line 6-10). With the initial palette $P_0 = \{c_1, c_2, ..., c_m\}$, we create the new solution by two different ways: adding a small random offset to each component of the randomly selected color c_i in P, or by randomly exchanging two colors. These two ways of updating unify the generation of the palette and the color assignment in a single process that assigns each class its most appropriate color.

Palette refinement by imposing hard constraints (line 11-13). To guarantee that all colors in Q are sufficiently discriminable, we check if any pair of colors would be closer to each other than a noticeable difference threshold τ, which is suggested to be 10, cf. [5]. If this is the case, we randomly perturb these colors until the differences between all color pairs are larger than τ.

Finding the optimal colorization (line 14-22). After palette refinement, we score the created palette Q as well as the previous one P_t (line 14) to decide whether to accept this solution or not (line 15-19). Our goal is to generate a discriminable visualization and hence we preserve the best result for each iteration (line 20-22).

3.4 Time Complexity

We implemented our algorithm using JavaScript and tested it on a computer with an Intel Core i7-7700HQ processor with 32GB memory. The point distinctness can be decomposed into two steps [36]; thus, the input data can be pre-computed and we separate this pre-processing step from the simulated annealing algorithm. In this way, the performance of our method is not related to the number of data points but instead depends only on the number of classes.

In each iteration of the optimization we apply the scoring functions two times, with a time complexity of $O(m^2)$. For a total number of iterations t, the time complexity for the whole algorithm is $O(tm^2)$. Fig. 5 left shows that the optimization can find reasonable palettes for scatterplots with 20 classes in less than 3s and for 40 classes in less than 15s. Fig. 5 right shows the convergence curves, our method has a number of strong oscillations at the beginning and then oscillates more smoothly until it converges. We checked all generated palettes with 40 classes and found that the distances of all color pairs were larger than the noticeable difference threshold [5]. Thus, we assume that Palettailor can generate 40 or more discriminable colors, of course only under the assumption of side-by-side comparison [23].

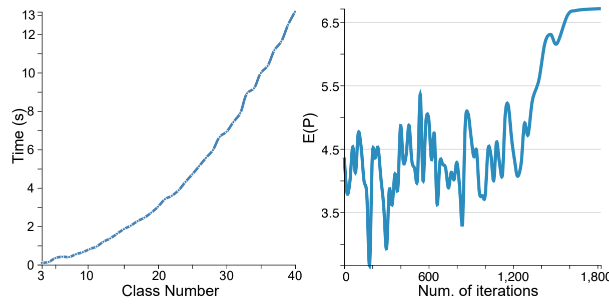


Fig. 5. Time complexity of our method: (left) optimization time versus number of Classes. (right) energy level $E(P)$ versus number of iterations.

4 Evaluation

We evaluate the quality of our palette generation method by comparing the resulting visualizations from our method with those from the existing methods: *Tableau* [30], *Colorgorical* [10] and *Colorization* [6]. As shown in Fig. 6, the existing palette generation methods have different levels of automation. For fully manual (*Tableau*) or partly manual methods (*Colorgorical*), we compared our method with both a best and a worst/random palette result (more details below).

To test these methods, we conducted two experiments through Amazon Mechanical Turk (AMT) with 155 participants in total. Experiment 1 focuses on multi-class scatterplots, while Experiment 2 tests our approach on line charts. We test the different methods with two types of tasks: discrimination tasks and preference tasks. In discrimination tasks, we ask participants questions that necessitate them to visually separate, or "discriminate", data from differently colored classes/lines. In preference tasks, we ask them about their subjective preferences between visual encodings with colors from the different palette generation methods.

Discrimination hypotheses. For discrimination tasks, we had the following hypotheses:

H1 Palettailor's results are not worse than the best cases from the manual process of designer-crafted palettes (*Tableau Best*) and auto-generated palettes (*Colorgorical Best*).

H2 Palettailor's results are better than the worst or random cases from the manual process of designer-crafted palettes (*Tableau Worst*) and auto-generated palettes (*Colorgorical Random*).

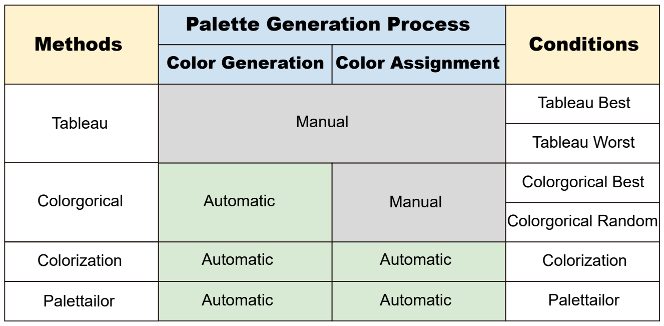
H3 Palettailor's results are better than the existing fully-automated method (*Colorization*).

Besides discriminability, we also want to find out that whether our results are aesthetically preferred by people or not, thus we have the other two hypotheses:

Preference hypotheses. In terms of preference tasks, we have the following hypotheses:

H4 The preferences of Palettailor's results are not worse than designer-crafted palettes (*Tableau*).

H5 Palettailor's results are preferred over the auto-generated palettes (*Colorgorical* and *Colorization*).



Methods	Palette Generation Process		Conditions
	Color Generation	Color Assignment	
Tableau	Manual		Tableau Best
			Tableau Worst
Colorgorical	Automatic	Manual	Colorgorical Best
			Colorgorical Random
Colorization	Automatic	Automatic	Colorization
Palettailor	Automatic	Automatic	Palettailor

Fig. 6. On the left column we show the existing methods that generate color palettes for visualizations including *Tableau* [30], *Colorgorical* [10] and *Colorization* [6], together with our proposed method *Palettailor*. These methods adopt different palette generation processes that have different levels of automation (shown in the middle columns). For example, *Colorgorical*'s process involves automatic color palette generation and manual color assignment, meaning that it needs manual effort to assign the colors from the palette to the visualization.

4.1 Experiment 1: Scatterplot Experiment

In this experiment, each participant completed one of three tasks from two different discrimination tasks and one preference task. As stimulus we used 30 multi-class scatterplot datasets for each task. For coloring the classes, we included six palette conditions – one from our method and the other five from existing methods (see right column of Fig. 6).

4.1.1 Experimental Design

Tasks & measures. We tested two discrimination tasks, and one preference task.

- *Counting (discrimination) task.* Following the methodology by Wang et al. [36], we asked participants to count the number of classes in a scatterplot and to choose an answer among several options. For each participant, we recorded the *time* taken for each trial, and counted the *errors* by calculating the differences between the actual number of classes and the participant's response. The counting task focuses on global discriminability. We reported the *error* measure as *time* is not of primary interest for our hypotheses.
- *Comparison (discrimination) task.* Following the methodology by Gleicher et al. [9], we asked participants to judge which of two specified class centroids was higher and to choose a value from a choice with two-alternatives. For each participant, we recorded the *time* taken and *error(0/1)* for each trial. The comparison task focuses on local discriminability.
- *Preference task.* Participants were shown a series of image pairs, each containing two scatterplots selected from two different conditions: one from *Palettailor* and the other from a benchmark condition. For each figure, the participant was asked to choose the plot they preferred. A neutral choice button was also provided.

To avoid potential carryover effects between tasks, we measured preference on the colored visualizations as a stand-alone task, instead of measuring it in the context of each discrimination task.

Conditions & sample palette generation. As shown in Fig. 6, we have six conditions in total, one from our method (*Palettailor*), and the other five conditions from the existing methods (*Tableau Best*, *Tableau Worst*, *Colorgorical Best*, *Colorgorical Random*, and *Colorization*).

Table 1. Organization of Datasets for the Discrimination Tasks: 30 datasets × 6 conditions. C: condition; G: participant group.

	C1	C2	C3	C4	C5	C6
Dataset 1	**G1**	G2	G3	G4	G5	G6
Dataset 2	G6	**G1**	G2	G3	G4	G5
Dataset 3	G5	G6	**G1**	G2	G3	G4
Dataset 4	G4	G5	G6	**G1**	G2	G3
Dataset 5	G3	G4	G5	G6	**G1**	G2
Dataset 6	G2	G3	G4	G5	G6	**G1**
...						
Dataset 29	G3	G4	G5	G6	**G1**	G2
Dataset 30	G2	G3	G4	G5	G6	**G1**

Since *Tableau* is a designer-crafted method, we included two conditions to mimic the best and worst discriminable results generated from the manual process. In a situation, for example, where the class number *m* is smaller than 10, a designer needs to manually choose *m* colors from the Tableau 10 palette. We run Wang et al.'s color assignment method [36] on the Tableau palette and select the first *m* colors with the highest and lowest scores for generating what we call *Tableau Best* and *Tableau Worst* conditions. We repeated this process multiple times and used the most suitable palettes as the final version in the experiment.

For Colorgorical, we specified the parameters in a way that we received a Low-Error setting: 20% Perceptual Distance, 40% Name Difference, 40% Pair Preference (see original paper for more details [10]). We repeatedly generated 5 palettes with these settings via their tool and randomly chose one of them. We then applied the color assignment process from Wang et al. [36] to simulate a designer's optimized assignment, and used a random assignment to represent a common default result (*Colorgorical Best* and *Colorgorical Random* conditions). We directly used Chens method [6] as the condition *Colorization*, since it is a fully-automated process. For Palettailor, the palettes were generated using the default weight setting of our method: [1,1,1]. Pilot tests showed that this setting performed best among different weight settings.

Scatterplot dataset generation. We used 30 multi-class scatterplot datasets with 6 to 10 classes in the experiment. Each dataset had two target classes that were meant for comparison. We generated the data points of the target classes using the method by Gleicher et al. [9]. The data points were placed according to a uniform random distribution in a 400×400 area, the height difference between these two classes was 30 pixels. Additionally, we generated the rest of the classes (distractors) using Gaussian random sampling and randomly placed them in an 500×500 area. This way the distractors have two kinds of relations to the target classes: intersecting them and including them. We used the Gaussian process to control the overall properties of each class such as size and density [27]. To simplify the variety of visual shapes we defined sizes to be large or small, and densities to be sparse or dense. Hence there were four types of classes: large & dense ($n = 100$, $\sigma = 50$), small & dense ($n = 50$, $\sigma = 20$), large & sparse ($n = 50$, $\sigma = 100$), and small & sparse ($n = 20$, $\sigma = 50$). See the supplemental material for more details.

Experiment organization. We used a *between-subjects* layout in the experiment to test the effects of 6 conditions using 30 datasets. We did not adopt a within-subjects design (i.e., each participant going through all the 180 condition-dataset combinations) because this would lead to unwanted learning effects, as one would see each dataset repeatedly six times. Thus instead, we had each participant to see different subsets of datasets for different conditions. As shown in Table 1, we used a Latin Square arrangement for the combination of datasets and conditions to avoid ordering effects. Thus there were six participant groups, each having 30 trials, every participant was assigned to one of the groups.

Pilot study & power analysis. We conducted a pilot study to check the experimental setup. It was also used to infer first insights about the effect sizes to be expected, the effect sizes were fed into a power

Table 2. Participants recruited for the formal study. G# specifies the participant group (more details see text)

Task	Participants	G1	G2	G3	G4	G5	G6
Counting	47	9	9	9	7	6	7
Comparison	33	6	5	6	5	5	6
Preference-1	25	5	5	5	5	5	NA
Slope	30	8	7	5	5	5	NA
Preference-2	20	5	5	5	5	NA	NA

analysis. For example, the power analysis (with an effect size Cohens d of 0.4, alpha of 0.05 and beta level of 0.8) suggested a minimum number of 24 participants for detecting a meaningful difference between *Palettailor* and *Colorgorical Best* in the counting task.

Participants. We recruited 105 participants for the main experiment using a crowd sourcing platform. According to the completion times in the pilot study, we paid each \$2.5 for the Counting Task, \$1.5 for the Comparison Task and \$0.5 compensation for Preference Task, in order to exceed the US minimum wage. All the participants gave us their informed consent, and no participant reported color vision deficiency.

Procedure. Each participant went through the following procedure: (1) viewing the instructions of the task and completing two training trials (except Preference Task); (2) performing the task as accurately as possible based on the instruction to do so; and (3) providing basic demographic information. See the supplemental material for more details.

4.1.2 Results

Following the methodology of previous studies, we analyzed the results using 95% confidence intervals, and also conducted Mann-Whitney tests to compare the differences between conditions. We selected this more conservative, non-parametric test as we observed in the pilot study that some measures were non-normally distributed. We also computed the effect size using *Cohen's d*, i.e., the difference in means of the conditions divided by the pooled standard deviation. In addition, we checked the data collected for both pilot and main studies, and found no significant interaction effects between datasets and conditions, so we focus on reporting the main effects.

Discriminability. Fig. 7(a)&(b) show the error results for the counting and comparison tasks, including effect sizes and p-values. Here, we focus on the analysis of *error*. We also measured *time*, but as it is not of primary interest for our hypotheses we provide the results in the supplemental materials.

The results are generally aligned with our discriminability hypotheses (**H1-3**).

First, we compared *Palettailor* to the best cases of the manual processes, i.e., *Tableau Best* and *Colorgorical Best*. We found that the *Palettailor* condition has significantly less errors ($p = 0.029$) than *Colorgorical Best* in the counting task. We did not observe other significant differences between these conditions. The result indicates that our approach appears not to be worse than the best cases of manually designer-crafted palettes and auto-generated palettes (**H1**), and might even lead to a better performance in the counting task.

Second, we compared *Palettailor* to the random or worst cases of the manual processes, i.e., *Tableau Worst* and *Colorgorical Random*. We found that in the counting task, *Palettailor* has a significantly lower error rate ($p < 0.001$) than *Colorgorical Random*. In the comparison task, *Palettailor* has a significant lower error rate ($p = 0.042$) than *Tableau Worst*. This partially confirms **H2** that our method is better than some of the manually created worst or random cases for certain discrimination tasks.

Finally, *Palettailor* is clearly better than the *Colorization* method, for both the counting task and the comparison task ($p < 0.001$). Thus, **H3** is fully confirmed and our method is indeed better than the existing fully-automated method in terms of discriminability.

Preference. For each comparison within the preference experiment, we assigned a preference score, i.e., 1 for preferring *Palettailor*, -1 for preferring another condition, and 0 for a neutral choice. We aggregated the scores for each pair containing *Palettailor* and a benchmark condition, e.g., *Tableau Best - Palettailor*. An average score value greater than 0 indicates that *Palettailor* is preferred over the other condition, and vice versa. The results are summarized in Fig. 7(d).

We found that *Palettailor* is significantly preferred over the designer-crafted palettes: *Tableau Best* ($p = 0.014$) and *Tableau Worst* ($p < 0.001$). This finding is not aligned with our hypothesis **H4**. We also found that *Palettailor* is significantly preferred over *Colorgorical Best* ($p < 0.001$), yet there is no clear preference between *Palettailor* and *Colorgorical Random* or *Colorization*. This observation partially aligns with **H5** and will be further discussed in Section 4.3.

4.2 Experiment 2: Line Chart Experiment

In addition to scatterplots, we also applied our method to line charts and evaluated its effectiveness in Experiment 2. We used 30 line chart datasets for each task as stimulus. The setup of Experiment 2 was similar to Experiment 1; the differences between the two will be described below.

4.2.1 Experimental Design

Tasks & measures. Each participant completed one of two tasks:

- *Slope (discrimination) task.* Following the methodology by Javed et al. [14], we asked participants to find out which line had the largest increase and to choose an answer among several options. For each participant, we recorded the *error (0/1)* and *time* taken for each trial. Similar to Experiment 1, we focused on reporting the *error* measure, see the supplemental materials for the details on *time*.
- *Preference task.* Similar to Experiment 1, each participant was shown a series of image pairs, each containing two line charts, one from the *Palettailor* condition and the other from one of the benchmark conditions. The participant was asked to choose the one s/he preferred.

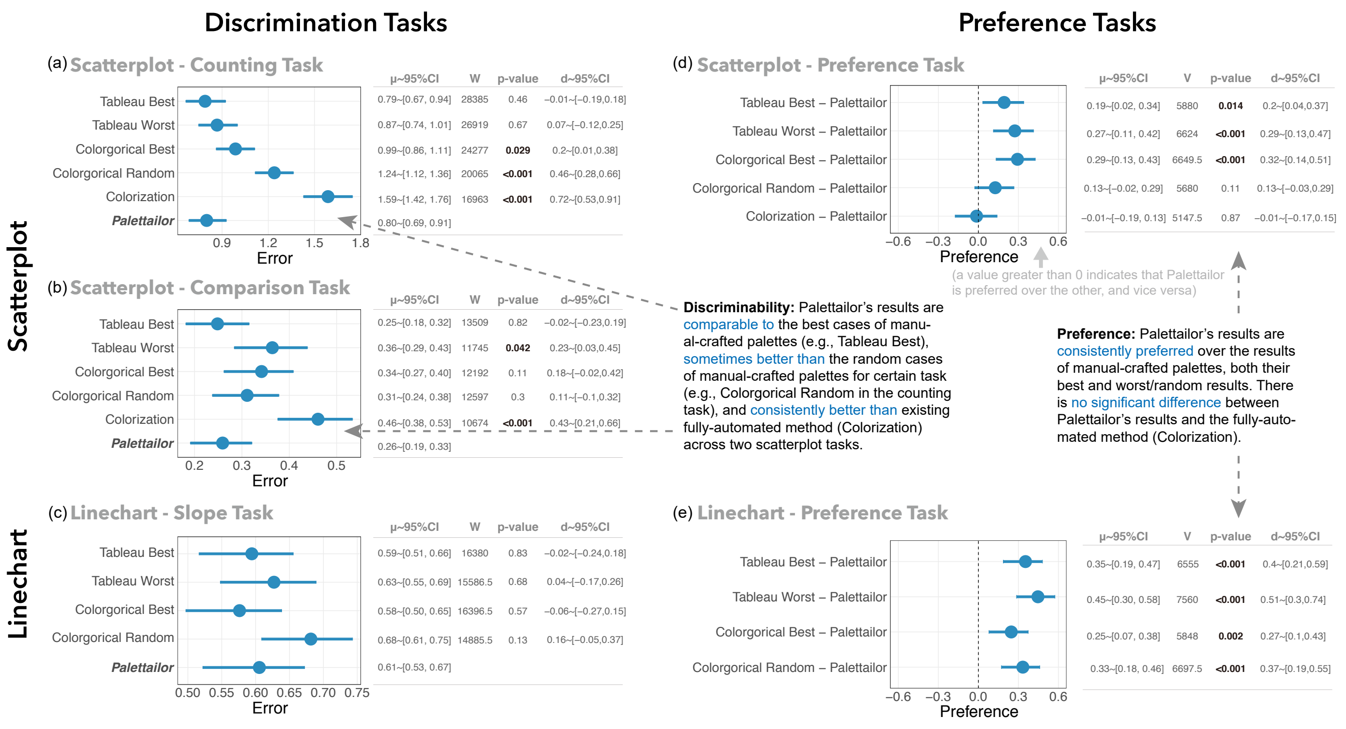
Conditions & sample palettes. In contrast to Experiment 1, we only tested five conditions in total, our method (*Palettailor*), as well as *Tableau Best*, *Tableau Worst*, *Colorgorical Best*, and *Colorgorical Random*. We did not include *Colorization* as it is not applicable to line charts. We used the same process as in Experiment 1 to generate the sample palettes for every condition.

Line chart dataset generation. We generated 30 different line charts with 6-10 lines following a carefully designed method. Every line chart had a *target* line with the largest increase; other lines with smaller increases were used to distract participants. We generated the target line with an increase of 0.35, with every point being within $[k-0.175,\ k+0.175]$ (k is a random number and $k \in [0,1]$). Then we repeated the process to generate the rest of the lines, each with an increase less than 0.3, i.e., every point within $[k-0.15, k+0.15]$. Each line chart was displayed in an 400×400 area, thus the minimum difference of increase between the largest and other lines on the screen is $0.05 \times 400 = 20$ pixels. See the supplemental material for more details.

The other aspects of the experiment were similar to Experiment 1. We used a *between-subjects* layout to avoid unwanted learning effects. We had 30 datasets and 5 conditions, the participants were divided into five groups, each going through a subset of 30 condition-dataset combinations. We conducted a pilot study including 11 participants from a crowd sourcing platform. As shown in Table 2, we recruited 50 participants in total for Experiment 2. Each participant went through the same procedure as that in Experiment 1, and was paid \$1.5 for the Slope Task and \$0.5 compensation for the Preference Task to exceed the US minimum wage.

4.2.2 Results

Discriminability. Fig. 7(c) shows the error results for the slope task. We observed no significant differences between *Palettailor* and *Tableau Best* or *Colorgorical Best*, which indicates that our approach is not worse than the best cases of manually designer-crafted palettes



(a) Scatterplot - Counting Task

	μ~95%CI	W	p-value	d~95%CI
Tableau Best	0.79~[0.67, 0.94]	28385	0.46	−0.01~[−0.19,0.18]
Tableau Worst	0.87~[0.74, 1.01]	26919	0.67	0.07~[−0.12,0.25]
Colorgorical Best	0.99~[0.86, 1.11]	24277	**0.029**	0.2~[0.01,0.38]
Colorgorical Random	1.24~[1.12, 1.36]	20065	**<0.001**	0.46~[0.28,0.66]
Colorization	1.59~[1.42, 1.76]	16963	**<0.001**	0.72~[0.53,0.91]
Palettailor	0.80~[0.69, 0.91]			

(b) Scatterplot - Comparison Task

	μ~95%CI	W	p-value	d~95%CI
Tableau Best	0.25~[0.18, 0.32]	13509	0.82	−0.02~[−0.23,0.19]
Tableau Worst	0.36~[0.29, 0.43]	11745	**0.042**	0.23~[0.03,0.45]
Colorgorical Best	0.34~[0.27, 0.40]	12192	0.11	0.18~[−0.02,0.42]
Colorgorical Random	0.31~[0.24, 0.38]	12597	0.3	0.11~[−0.1,0.32]
Colorization	0.46~[0.38, 0.53]	10674	**<0.001**	0.43~[0.21,0.66]
Palettailor	0.26~[0.19, 0.33]			

(c) Linechart - Slope Task

	μ~95%CI	W	p-value	d~95%CI
Tableau Best	0.59~[0.51, 0.66]	16380	0.83	−0.02~[−0.24,0.18]
Tableau Worst	0.63~[0.55, 0.69]	15586.5	0.68	0.04~[−0.17,0.26]
Colorgorical Best	0.58~[0.50, 0.65]	16396.5	0.57	−0.06~[−0.27,0.15]
Colorgorical Random	0.68~[0.61, 0.75]	14885.5	0.13	0.16~[−0.05,0.37]
Palettailor	0.61~[0.53, 0.67]			

(d) Scatterplot - Preference Task

	μ~95%CI	V	p-value	d~95%CI
Tableau Best – Palettailor	0.19~[0.02, 0.34]	5880	**0.014**	0.2~[0.04,0.37]
Tableau Worst – Palettailor	0.27~[0.11, 0.42]	6624	**<0.001**	0.29~[0.13,0.47]
Colorgorical Best – Palettailor	0.29~[0.13, 0.43]	6649.5	**<0.001**	0.32~[0.14,0.51]
Colorgorical Random – Palettailor	0.13~[−0.02, 0.29]	5680	0.11	0.13~[−0.03,0.29]
Colorization – Palettailor	−0.01~[−0.19, 0.13]	5147.5	0.87	−0.01~[−0.17,0.15]

(e) Linechart - Preference Task

	μ~95%CI	V	p-value	d~95%CI
Tableau Best – Palettailor	0.35~[0.19, 0.47]	6555	**<0.001**	0.4~[0.21,0.59]
Tableau Worst – Palettailor	0.45~[0.30, 0.58]	7560	**<0.001**	0.51~[0.3,0.74]
Colorgorical Best – Palettailor	0.25~[0.07, 0.38]	5848	**0.002**	0.27~[0.1,0.43]
Colorgorical Random – Palettailor	0.33~[0.18, 0.46]	6697.5	**<0.001**	0.37~[0.19,0.55]

Discriminability: Palettailor's results are comparable to the best cases of manual-crafted palettes (e.g., Tableau Best), sometimes better than the random cases of manual-crafted palettes for certain task (e.g., Colorgorical Random in the counting task), and consistently better than existing fully-automated method (Colorization) across two scatterplot tasks.

Preference: Palettailor's results are consistently preferred over the results of manual-crafted palettes, both their best and worst/random results. There is no significant difference between Palettailor's results and the fully-automated method (Colorization).

Fig. 7. Results for scatterplot and line chart experiments: For each task (e.g., the counting task in the scatterplot experiment, marked as *Scatterplot - Counting Task*), we give a confidence interval plot and a statistic table. In the table, for each condition we provide the statistics including the mean with 95% confidence interval (μ ~95%CI), the W-value and p-value from the Mann-Whitney test, and the effect size (d ~95%CI). For the discrimination tasks on the left, we conducted statistical tests to compare *Palettailor* to every other condition, and provide W or V values, p-value and effect size (d ~95%CI) accordingly in the tables.

Table 3. The summary of results. E1 and E2 indicate the scatterplot and line chart experiment. (**) means a hypothesis is confirmed; (*) means partly confirmed.

Hypothesis	*E1*	*E2*
H1 Discriminability is not worse than the best manual cases.	**	**
H2 Discriminability is better than the random manual cases.	*	
H3 Discriminability is better than the automated method.	**	NA
H4 Preference is not worse than the designer-crafted palettes.		
H5 Preference is better than the auto-generated palettes.	*	**

and auto-generated palettes (**H1**). Also, we did not observe clear differences between *Palettailor* and the worst or random manual cases, i.e., *Tableau Worst* or *Colorgorical Random*. This finding does not aligned with **H2**. We discuss the detailed implications of these results in the next section.

Preference. The results are summarized in Fig. 7(e), where an average preference score larger than 0 means that *Palettailor* was preferred over the other condition, and vice versa. We found that *Palettailor* is preferred over the designer-crafted palettes: *Tableau Best* ($p < 0.001$) and *Tableau Worst* ($p < 0.001$). This finding does not align with our hypothesis **H4**. We also found that *Palettailor* is preferred over *Colorgorical Best* ($p = 0.002$) and *Colorgorical Random* ($p < 0.001$), which aligns with **H5**.

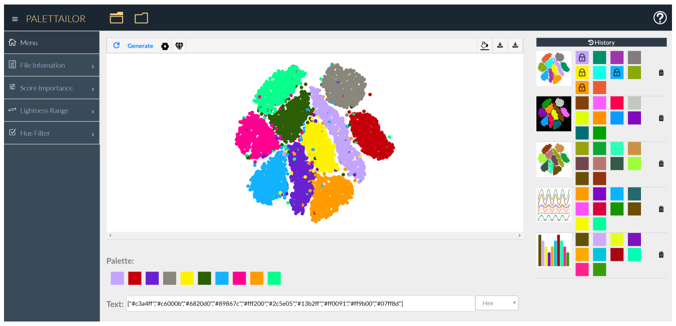
4.3 Discussion

The results of the two experiments are summarized in Table 3. In terms of *discriminability*, we found that Palettailor appears not to be worse than the best cases of color generation processes that involve manual steps (i.e., the *Tableau Best* and *Colorgorical Best* conditions). We also found that Palettailor is more effective than some less-optimized manual cases for certain tasks (e.g., the *Colorgorical Random* condition in the scatterplot-counting task). Furthermore, Palettailor is consistently better than the fully-automated approach (i.e., *Colorization*). The results indicate that Palettailor can automatically generate palettes that are ***not worse than*** those from the manual approaches, and that are ***more discriminable*** than those from the fully-automated palette generation approach.

However, although the results regarding discriminability are generally aligned with our hypotheses for the scatterplot experiment, we found no significant differences in the line chart experiment. One possible explanation is that the visual complexity of a line chart is generally lower than for a scatterplot. Thus, having discriminable clusters in a scatterplot may aid more to people's visual performance than having discriminable lines in a line chart. This leads to the necessity to further explore how Palettailor can benefit visual perception when applying it to different visualization types.

In terms of *preference*, we observed that Palettailor is preferred over designer-crafted palettes (i.e., the *Tableau* palettes), and that it is also preferred over some auto-generated palettes for certain visualizations (e.g., the *Colorgorical* palettes for scatterplot visualizations).

From our results it seems like that visualizations with brighter colors were generally preferred (i.e., the *Palettailor*, *Colorization* and sometimes *Colorgorical* palettes), and yet the designer-crafted palette *Tableau* was less preferred. There are several potential explanations. It is possible that the preference of bright colors is due to the cultural background of the participants [25], given that all the participants were recruited from the US. Another explanation is that for some people, when looking at visualizations with bright colors, they may gain *cognitive ease* [33]. In other words, they may be biased towards *believing* that it would be easier to tell apart clusters or lines (i.e., performing discrimination tasks), no matter whether the colors could truly aid discriminability. The result can also potentially be explained by *affective response* [3], i.e., more saturated colors are strongly correlated with intensity and positive impressions. Such affect might have caused higher rates of the brighter palettes. Future research may benefit the design of Palettailor in supporting people's aesthetic preferences.



Text: ["#c3a4ff","#c6000b","#6820d0","#89867c","#fff200","#2c5e05","#13b2ff","#ff0091","#ff9b00","#07ff8d"]

Fig. 8. A screenshot of our interactive system. Users can load different datasets and generate suitable color palettes. If they are satisfied with a result, they can download the image or export the palette definition to use in their own system. We also offer a history for users to find their previous results.

4.4 Limitations

The goal of our user study was an initial exploration of the effectiveness of our method. We thus focused on benchmarks from systematically generated datasets and sample palettes. At the same time, we fixed or limited some of the other factors, which may affect the generalizability of the study results. Although our proposed method is capable of supporting a variety of visualization types, in the study we only chose two types: scatterplots and line charts. While these are common visualization types, we did not cover visualizations with larger area marks, such as bar or area charts. In addition, our method is capable of tailoring palettes based on the background color, but for the study, we chose a white background as it is the most commonly used canvas color to display visualizations. Lastly, in our study we measured aesthetic preferences as a stand-alone task to avoid potential carryover effects. However, it might be possible that people's preference would change based on the tasks they are doing. Further studies need to be conducted to comprehensively examine these factors.

5 Interactive System for Colorization

Based on the automatic approach presented above, we developed an interactive color palette generation system (Fig. 8) that allows users to feed in personal preferences when looking for good colors. After the user uploads his/her data, different color palettes will be produced automatically. Users may also edit color palettes based on their preferences. A history and other interactive features support the user in finding a desired color palette.

5.1 User Interaction

Our system provides three main ways to interact with color palettes: adjustable palette generation, constrained palette generation, and palette completion.

Adjustable palette generation. Users can adjust the weights of the three scoring functions to generate palettes with different emphasis. For example, if a user wants to emphasize on local discrimination, then s/he can turn up Point Distinctness or turn down Color Discrimination or Name Difference. If a user wants to create a palette composed of as many color names as possible, s/he can turn up Name Difference.

Constrained palette generation. Users can also specify preferable colors for palette generation. We use 11 universal basic color terms (blue, brown, green, orange, pink, purple, red, yellow, black, grey, and white) [4] as our Hue Filter. For example, when a user selects some colors for generating a color palette, our system will automatically search for optimized palettes within the respective ranges. As our algorithm takes the background color into account, we also allow users to specify a background color and adjust the luminance range accordingly.

Palette completion. Our system enables users to fix colors of a palette. Once some colors are fixed, our system will evaluate the current palette as well as the data characteristics, and search for optimized other colors to complete it.

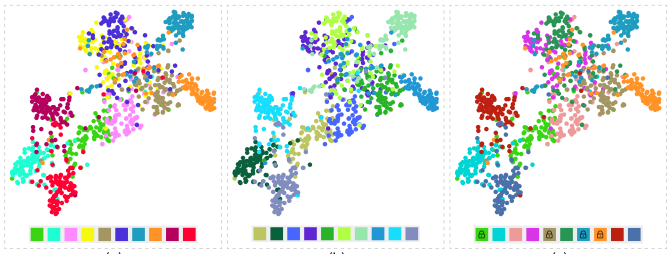


Fig. 9. CIFAR10 dataset: (a) Palette generated by default setting; (b) User-specified Hue Filter with "green" and "blue"; (c) After clicking on colors to lock them, our system completes the palette automatically.

5.2 Case Study

We conducted a case study with a data expert to evaluate the usability of our system. The dataset we used for this study was the CIFAR10 dataset [16] that includes classification results for images of ten object classes (e.g., cats, ships). The expert first used the default settings of our system to automatically produce a palette for assigning colors to different objects in the dataset. Seeing the result in Fig. 9(a), she was able to easily discriminate every class. After generating multiple palettes with our default settings, she used the Hue Filter and chose two preferable colors ("green" and "blue"). The result is shown in Fig. 9(b). Even though all colors are variants of blue and green, it is still possible to discriminate all classes. However, the expert preferred the previous results by our default setting, so she used the history to find the results. Rather than generating a totally new palette, she wanted to preserve some colors. So she clicked on these colors to lock them and produced another palette, shown in Fig. 9(c). Overall, she was very satisfied with the system and used all our modalities to create a good palette.

6 Conclusion

We presented Palettailor, a data-aware approach for producing color palettes for categorical visualizations that allows a better visual discrimination of classes while being visually pleasing. This goal is achieved by a simulated annealing-based optimization, which we feed with three scoring functions: point distinctness, name difference, and color discrimination constraint. We evaluated Palettailor through a crowd-sourcing study, which empirically demonstrated that our produced palettes allow for good class discrimination, are preferred by users, and can do that fully automatically.

Our current measure of point distinctness is based on position and color of the points in the underlying visualizations. However, class discriminability in human perception is also related to many other factors such as mark shape, mark size, and rendering order [21]. In the future, we would like to extend our model to incorporate such factors as well. Weighting the point distinctness by the number of points in each class is able to generate reasonable results in most cases. However, it might de-emphasize classes with fewer data points, such as outliers. To address this issue, we will integrate a user-steered parameter so as to assign priorities to small categories. In addition, we would also like to let our technique support more visualization techniques such as pie charts and parallel coordinates. We also want to include the generation of color palettes for people with color vision deficiencies.

Acknowledgments

This work is supported by the grants of the NSFC (61772315, 61861136012), the Open Project Program of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University (No.VRLAB2020C08), the CAS grant (GJHZ1862) and Deutsche Forschungsgemeinschaft (DFG) – Project-IDs DE 620/26-1, as well as 251654672 – TRR 161 Quantitative methods for visual computing.

References

[1] E. Aarts. A stochastic approach to combinatorial optimization and neural computing. *Simulated Annealing and Boltzmann Machines*, 1989.

[2] M. Aupetit and M. Sedlmair. SepMe: 2002 new visual separation measures. In *IEEE Pacific Visualization Symposium*, pp. 1–8, 2016. doi: 10.1109/pacificvis.2016.7465244

[3] L. Bartram, A. Patra, and M. Stone. Affective color in visualization. In *Proceedings of the 2017 CHI conference on human factors in computing systems*, pp. 1364–1374, 2017.

[4] B. Berlin and P. Kay. *Basic color terms: Their universality and evolution*. Univ of California Press, 1991.

[5] A. Brychtová and A. Çöltekin. The effect of spatial distance on the discriminability of colors in maps. *Cartography and Geographic Information Science*, 44(3):229–245, 2017.

[6] H. Chen, W. Chen, H. Mei, Z. Liu, K. Zhou, W. Chen, W. Gu, and K.-L. Ma. Visual abstraction and exploration of multi-class scatterplots. *IEEE Trans. Vis. & Comp. Graphics*, 20(12):1683–1692, 2014.

[7] J. Chuang, D. Weiskopf, and T. Möller. Energy aware color sets. *Computer Graphics Forum*, 28(2):203–211, 2009. doi: 10.1111/j.1467-8659.2009.01359.x

[8] H. Fang, S. Walton, E. Delahaye, J. Harris, D. Storchak, and M. Chen. Categorical colormap optimization with visualization case studies. *IEEE Trans. Vis. & Comp. Graphics*, 23(1):871–880, 2017. doi: 10.1109/tvcg.2016.2599214

[9] M. Gleicher, M. Correll, C. Nothelfer, and S. Franconeri. Perception of average value in multiclass scatterplots. *IEEE Trans. Vis. & Comp. Graphics*, 19(12):2316–2325, 2013. doi: 10.1109/tvcg.2013.183

[10] C. C. Gramazio, D. H. Laidlaw, and K. B. Schloss. Colorgorical: Creating discriminable and preferable color palettes for information visualization. *IEEE Trans. Vis. & Comp. Graphics*, 23(1):521–530, 2017. doi: 10.1109/tvcg.2016.2598918

[11] M. Harrower and C. A. Brewer. ColorBrewer.org: an online tool for selecting colour schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003. doi: 10.4324/9781351191234-18

[12] C. G. Healey. Choosing effective colours for data visualization. In *Proc. IEEE Conf. on Visualization*, pp. 263–270, 1996. doi: 10.1109/visual.1996.568118

[13] J. Heer and M. Stone. Color naming models for color selection, image editing and palette design. In *Proc. SIGCHI Conference on Human Factors in Computing Systems*, pp. 1007–1016, 2012. doi: 10.1145/2207676.2208547

[14] W. Javed, B. McDonnel, and N. Elmqvist. Graphical perception of multiple time series. *IEEE Trans. Vis. & Comp. Graphics*, 16(6):927–934, 2010.

[15] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, et al. Optimization by simmulated annealing. *Science*, 220(4598):671–680, 1983.

[16] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[17] S. Lee, M. Sips, and H.-P. Seidel. Perceptually driven visibility optimization for categorical data visualization. *IEEE Trans. Vis. & Comp. Graphics*, 19(10):1746–1757, 2013. doi: 10.1109/tvcg.2012.315

[18] S. Lin, J. Fortuna, C. Kulkarni, M. Stone, and J. Heer. Selecting semantically-resonant colors for data visualization. *Computer Graphics Forum*, 32(3pt4):401–410, 2013. doi: 10.1111/cgf.12127

[19] G. M. Machado, M. M. Oliveira, and L. A. Fernandes. A physiologically-based model for simulation of color vision deficiency. *IEEE Trans. Vis. & Comp. Graphics*, 15(6):1291–1298, 2009. doi: 10.1109/TVCG.2009.113

[20] B. A. Maxwell. Visualizing geographic classifications using color. *The Cartographic Journal*, 37(2):93–99, 2000. doi: 10.1179/caj.2000.37.2.93

[21] L. Micallef, G. Palmas, A. Oulasvirta, and T. Weinkauf. Towards perceptual optimization of the visual design of scatterplots. *IEEE Trans. Vis. & Comp. Graphics*, 23(6):1588–1599, 2017. doi: 10.1109/TVCG.2017.2674978

[22] S. Mittelstädt, D. Jäckle, F. Stoffel, and D. A. Keim. Colorcat : Guided design of colormaps for combined analysis tasks. In E. Bertini, ed., *Eurographics Conference on Visualization (EuroVis) : Short Papers*, pp. 115–119. The Eurographics Association, 2015. doi: 10.2312/eurovisshort.20151135

[23] T. Munzner. *Visualization Analysis and Design*. AK Peters visualization series. CRC Press, Boca Raton, FL, 2015. doi: 10.1109/pacificvis.2016.7465242

[24] P. Nardini, M. Chen, F. Samsel, R. Bujack, M. Bottinger, and G. Scheuermann. The making of continuous colormaps. *IEEE Trans. Vis. & Comp. Graphics*, December 2019. doi: 10.1109/tvcg.2019.2961674

[25] S. E. Palmer and K. B. Schloss. An ecological valence theory of human color preference. *Proceedings of the National Academy of Sciences*, 107(19):8877–8882, 2010.

[26] D. Pham and D. Karaboga. *Intelligent optimisation techniques: genetic algorithms, tabu search, simulated annealing and neural networks*. Springer Science & Business Media, 2012.

[27] M. Sedlmair, A. Tatu, T. Munzner, and M. Tory. A taxonomy of visual cluster separation factors. *Computer Graphics Forum*, 31(3pt4):1335–1344, 2012. doi: 10.1111/j.1467-8659.2012.03125.x

[28] V. Setlur and M. C. Stone. A linguistic approach to categorical color assignment for data visualization. *IEEE Trans. Vis. & Comp. Graphics*, 22(1):698–707, 2016. doi: 10.1109/tvcg.2015.2467471

[29] G. Sharma, W. Wu, and E. N. Dalal. The ciede2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. *Color Research & Application*, 30(1):21–30, 2005. doi: 10.1002/col.20070

[30] Tableau Software. The tableau visualization system. `http://www.tableausoftware.com/`.

[31] C. Tominski, G. Fuchs, and H. Schumann. Task-driven color coding. In *Proc. Int. Conf. on Information Visualisation*, pp. 373–380, 2008. doi: 10.1109/iv.2008.24

[32] E. R. Tufte, N. H. Goeler, and R. Benson. *Envisioning information*, vol. 126. Graphics press Cheshire, CT, 1990.

[33] A. Tversky and D. Kahneman. Judgment under uncertainty: Heuristics and biases. *science*, 185(4157):1124–1131, 1974.

[34] R. C. Veltkamp. The γ-neighborhood graph. *Computational Geometry*, 1(4):227–246, 1992.

[35] L. Wang, J. Giesen, K. T. McDonnell, P. Zolliker, and K. Mueller. Color design for illustrative visualization. *IEEE Trans. Vis. & Comp. Graphics*, 14(6):1739–1754, 2008. doi: 10.1109/tvcg.2008.118

[36] Y. Wang, X. Chen, T. Ge, C. Bao, M. Sedlmair, C.-W. Fu, O. Deussen, and B. Chen. Optimizing color assignment for perception of class separability in multiclass scatterplots. *IEEE Trans. Vis. & Comp. Graphics*, 25(1):820–829, 2019. doi: 10.1109/TVCG.2018.2864912

[37] M. Wijffelaars, R. Vliegen, J. J. Van Wijk, and E.-J. Van Der Linden. Generating color palettes using intuitive parameters. 27(3):743–750, 2008.

[38] A. Zeileis, K. Hornik, and P. Murrell. Escaping RGBland: selecting colors for statistical graphics. *Computational Statistics & Data Analysis*, 53(9):3259–3270, 2009. doi: 10.1016/j.csda.2008.11.033

[39] L. Zhou and C. D. Hansen. A survey of colormaps in visualization. *IEEE Trans. Vis. & Comp. Graphics*, 22(8):2051–2069, 2016. doi: 10.1109/tvcg.2015.2489649