2014

Colormoo: An Algorithmic Approach to Generating Color Palettes

Joshua Rael
Claremont McKenna College

Pomona College
Department of Computer Science

Colormoo: An Algorithmic Approach to Generating Color Palettes

Joshua Rael

April 27, 2014

Submitted as part of the senior exercise for the degree of
Bachelor of Arts in Computer Science

Patrick McNally, Art Lee, advisors

Abstract

Selecting one color can be done with relative ease, but this task becomes more difficult with each subsequent color. Colormoo is an online tool aimed at solving this problem. We implement three algorithms for generating color palettes based off of a starting color. Data is collected for each palette that is generated. Our analysis reveals two of the algorithms are preferred, but under different circumstances. Furthermore, we find that users prefer palettes containing colors that are compatible, but not too similar. With refined heuristics, we believe these techniques can be extended and applied beyond the field of graphic design alone.

Acknowledgments

This work was done in collaboration with the Computer Science departments at both Pomona and Claremont McKenna College. We would like to thank all those involved throughout the development of this research. In particular, we would like to acknowledge Professor Rhett Bull for his early guidance, Professor Art Lee for his support and suggestions, and especially Professor Patrick McNally for his patience and encouragement at each step of this project.

Contents

List of Figures

Chapter 1

Introduction

While the way that designers have interacted with color has evolved throughout the history of graphic design, the problem of choosing colors continues to be a difficult task. Fortunately there are many resources available to aid in the process, though their number alone is enough to suggest there is no perfect solution. Color choice sits at a peculiar point between intuition and experience. A color may be chosen because it conforms to established set of rules or simply because the absence of it would leave more to be desired. One approach to finding compatible colors is by consensus. As such, modern tools have positioned themselves on the internet where they can be part of a much larger feedback loop. Two popular websites, Adobe Kuler and COLOURLovers, have developed forums through which users can create color palettes and submit them for public appraisal. On these two sites alone, more than two million palettes have been generated by tens of thousands of users [OAH11].

A designer may select a color for any number of reasons, but the value of that color often cannot be established until it sits alongside other colors. Herein lies the essence of this problem: picking one color can be done with relative ease, but the task becomes more difficult with each subsequent color. Despite all the excitement around seeking compatible colors, little has been done in exploration of this subproblem.

The goal of this research is to evaluate three different algorithms that produce color *themes*, i.e., a set of six compatible colors. We built an application that measures which algorithms are preferred and under what conditions. On Colormoo, designers can input a color of their choice, generate a series of themes, and select their favorite. We collect data for each palette that users select. Ultimately, we observe that two algorithms come out on

top and the properties of the user's starting color will make one approach more favorable than the other. These results, along with general observations of color preference, can provide helpful insight into future explorations of this problem, especially as it becomes relevant beyond the field of graphic design alone.

Chapter 2

Background

The problem of finding compatible colors is rooted in the different ways colors can be represented. Color experts manipulate these representations to create patterns which have been implemented in various online tools for generating color palettes.

2.1 Color Representation

Colors are represented numerically according to an underlying color model. This specification defines a point in the color space and can be used to determine how one color is related to another. Though there are several color models that have been developed in the study of color, we examine three for the way they are used in various graphics applications.

2.1.1 RGB

The red-green-blue (RGB) color model defines how screens emit colored light. This is the model computer monitors use to determine the color of each individual pixel. A color is represented by the amount of red, green, and blue required to produce it, each within the range of 0 to 255. These colors are typically represented by a three-tuple or by their hexadecimal (HEX) variant in base 16. For example, white light is represented as (255,255,255) where black is (0,0,0). Although, RGB is not the most intuitive way for humans to think about color, this is a popular format for referencing a particular color. HEX colors, for example, are used in stylesheets to display colors online.

Figure 2.1: The three different components of the HSL color space [Vir14].

2.1.2 HSL

The hue-saturation-lightness (HSL) model more closely represents the way we think about color (Figure 2.1). Hue is typically what people think when they hear the word color [Jac13]. It represents the rainbowed spectrum of light from red to violet. Hue is measured in degrees because it is commonly displayed as a color wheel. For example, red colors sit around 0 and 360 degrees, while blues are more in the 175-220 range. Saturation is the percentage of color intensity. Really vibrant colors will be highly saturated. Dull colors will have closer to 0% saturation. Finally, lightness is used to define a color on the darker or lighter end of the spectrum. Lightness is also measured on a percentage scale.

2.1.3 CIE LAB

Where HSL may be closer to the way that humans think about colors, CIE LAB is more akin to the way that humans *perceive* colors. In this model, the L stands for lightness, A for the red-green range, and B for the blue-yellow range. It is one of two systems accepted by the *Comission Internationale de I'Eclairage* (International Commission on Illumination) as a perceptually uniform color space [Inc00]. This is the idea that colors each have different weight associated with them. For example, regardless of how light or dark it is, a blue color is going to stick out more than a yellow color. While this is a valuable property of this model, working with CIE LAB is a little more difficult than working with RGB or HSL. CIE LAB is a much larger color space that is capable of representing colors not present in the other models. However, only a subset of the colors in CIE LAB are accessible to the human eye [Mac09]. These perceptual intricacies result in an irregular geometry that requires a more accounting on behalf of the user, especially as it relates to the problem of color selection.

2.2 Color Compatibility Algorithms

Finding compatible colors is not a problem unique to one audience. Architects, engineers, publishers, artists, and designers all use color every day. Each has their own toolset for color use in their respective fields. We explore a few of these techniques and observe how they are implemented in different consumer facing applications.

2.2.1 Color Theory

Isaac Newton was one of the first to organize colors geometrically. In 1704, Newton observed the relationships between different wavelengths of light and organized them as a circle of hue. He was then able to identify different patterns that arose out of this representation. This would become the foundation of the color wheel and modern color theory [Mac09].

Fixed rotations about the color wheel can be used to define a set of compatible colors or hue templates [OAH11]. These are the colors that are still taught in art and design classes today. They include, for example, complimentary colors that sit on opposite ends of the color wheel or analogous colors that reside next to one another in the color wheel. Figure 3.2 displays the templates that we use in Colormoo. Though designers typically only use hue templates as a starting point [OAH11], this approach has survived the test of time as the tried and true way to generate color palettes. Furthermore, it enables users to quickly find a set of colors that are compatible with some starting color.

2.2.2 Equidistant Colors

Graphics Editor Gregor Aisch offers a unique method of sampling color palettes that can be used with any color space [Ais11]. Specifically, Aisch recognizes the value in using equidistant colors. Equidistant colors are colors that are the same distance from each other when plotted out in a color space. To show this, Aisch performs a two-dimensional transformation of a three-dimensional color space. Two of the components define the x and y axis and the third component is held constant, respectively. One can sample equidistant colors along a straight line drawn in this space. For example, we might transform HSL to display saturation on the x-axis, lightness on the y-axis, and keep hue constant. Doing so would allow us to draw a line in the saturation-lightness space that would result in a monochromatic color scheme. This mapping could be very useful for representing data points

in statistics. Our implementation will keep lightness constant to give the user more control over the hue component. An example of this technique can be seen in our implementation in Figure 3.3. Regardless, this technique is a relatively simple way to produce a variety of different palettes and previous studies confirm that colors along a line drawn this kind of space are harmonious [OAH11].

2.2.3 K-Means Clustering

Mathieu Jacomy of the Sciences-Po Medialab took a more algorithmic approach to generating qualitative color palettes for data visualization. Jacomy explains that a data scientist wants to avoid distracting colors (avoiding neon colors, for example) and to keep homogeneity (making sure the colors are compatible with one another) [Jac13]. It is also very important that each color is distinct. With his web application, I Want Hue, he utilizes k-means clustering and force vectors to create clusters of colors that satisfy these requirements. Users can select the number of colors they need for their project. This defines the number of clusters that will be used to partition the color space. Users also have the freedom to adjust different parameters to determine which colors they want to compose their palette.

Jacomy used this approach with the HSL color space, but found better results with RGB. Ultimately he ended up using CIE LAB because "the distance in this space as perceptive distance... distant colors will be perceived [is] distinct." [Jac13] This property allows for a more even distribution of colors and therefore better clustering results.

2.2.4 Color Quantization

Another approach is to generate color palettes from images. This technique is called color quantization. There is a dissonance between the color capture capabilities of cameras and the color display capabilities of computers. Traditionally this approach is used to generate a palette of colors to more accurately display images on computer monitors [HLT09]. However, that palette can be used a little more generally. Oftentimes people admire photographs because they contain an appealing set of colors. A color quantization algorithm can identify the most prominent colors in a photo and to create a color theme [OAH11]. This method can be used on anything from photos of 20th century paintings to a quick snapshot of the sunset from a mobile phone.

Color quantization produces reliable results [OAH11]. It has the additional freedom of allowing users to sample colors from any image they find aesthetically pleasing. Tools that utilize this method are great for deriving new and fresh palettes. However, the palette will always be constrained to the colors present in that image. If the user already has a color that they would like to use in their design this may not be the best way to derive a color theme.

2.3 Application of Color Compatibility Algorithms

2.3.1 Representing Data

Much research in finding compatible colors can be found in the context of statistics and data visualization. In this field, color is used to create meaning and add dimensionality to a dataset. Colors help to better display trends and identify the intensity or scarcity in data. Since each color is used to represent a different data point, it is important that the color remains distinct from the other colors in the palette.

Significant interest in producing better color palettes for data visualization started with with ColorBrewer [HB03]. Their primary concern was that federal agencies and nonprofits very frequently produce thematic maps to communicate their research to the public. These maps are widely distributed and convey important insight into their work. Though their software generally includes default color schemes, there is little guidance on the effective use of that color and oftentimes they do not have time to develop a better color palette to represent their data. The team addresses this problem with a web application entitled ColorBrewer that provides more attractive color schemes based off of three primary use cases.

For data with a clear order to it, the ColorBrewer team implemented a set of sequential color schemes in which each color represents a step into a different range of values. For data that sits between two well defined extremes, they have a set of divergent color schemes. Thirdly, they created a set of qualitative colors that "rely primarily on differences in hue to create a colour scheme that does not imply order, merely a difference in kind" [HB03]. These are the schemes that most relate to the type of color themes we explore in this project.

It is important to note that none of these color palettes are generated dynamically on the site. Each was selected as a well saturated palette constructed by sampling colors along an arc drawn in a color space [HB03]. The user only has control insofar that they can choose the number of col-

ors present in the palette (between 3-12, though not all schemes can be expanded to 12 colors). Ultimately this means sacrificing a little artistic freedom when it comes to creating a color palette.

Jacomy's I Want Hue is a little more elegant in this regard. The same kind of palettes for data visualization can be created by restricting the color space using the input sliders on the site. These schemes are successful without relying on hard coded color values.

2.3.2 Color Themes

Color use in graphic design is a much more open problem. The goals of each project vary and, unlike more quantitative fields, taste is much more of a factor in deciding whether or not a color should be used. For this reason, the colors do not necessarily need to be distinct. In fact, themes with too many distinct colors tend to be rated lower than simpler themes that utilize fewer hues [OAH11]. That said, it is important to have some variation to retain visual interest.

The COLOURLovers site (www.colourlovers.com) is a popular destination for professionals and amateurs seeking this kind of color palette. Users can submit hand-picked color palettes for others to use. The most popular of the submissions rise to the top. Users are constantly taking the themes and modifying them for different artistic purposes, often merging with or incorporating other colors found on the site. Furthermore, users can search for palettes by defining a range of color that they would like to use in their designs. It is unsurprising that COLOURLovers produces such popular color themes because of the invaluable human input that goes into each palette.

To approach this concept from a different direction, we observe a trend on portfolio sites like Dribbble (www.dribbble.com), a social network for designers, where designers post the color palette they used alongside the work that incorporated it. This idea of visual DNA is appealing to many users of the site. An otherwise indifferent color palette becomes a lot more interesting when paired with an example of how it might be used. Additionally, since each image is tagged with these colors, search capabilities allow users to see how different artists use the same color in their work. As a result, sites like Dribbble have become a color palette tool in their own right.

Finally, there are several tools that leverage color theory and hue templates. Popular among them is Adobe Kuler (www.kuler.adobe.com). Like COLOURLovers, Kuler allows users to create and share color themes. Users can see the themes others have created and rate them on a five point scale. O'Donovan et al. performed an analysis of color compatibility by sampling

data from the two sites. An important difference between the two services is that the Kuler interface encourages the use of hue templates while the COLOURLovers interface does not. In comparing the two services, they found that users would not typically create themes based on hue templates unless encouraged to by the interface [OAH11]. Furthermore, they did not find a correlation between a high rating and a theme whose colors closely matched a hue template. However, when users did utilize hue templates, they found themes with complementary and analogous colors to be the most highly rated.

Chapter 3

Colormoo

Our research culminates in the development of Colormoo (Figure 3.1), an online tool for generating color palettes. The site is live and can be viewed at www.colormoo.com. The goal of this site was to evaluate different algorithms for generating color themes from a starting color.

3.1 The User Flow

Colormoo takes a starting color as input and outputs three color palettes based on that color. The user can input the starting color with a color specification or by manipulating the HSL sliders to dynamically select a color. The chosen color is visually displayed on the left-hand side of the input box. When the user clicks the generate button, the system presents the three different color palettes. When the user has selected the color palette they like best, Colormoo displays the numerical specifications for each color in three different formats: HEX, RGB, and HSL. At this point, the user can take these colors for use in their designs or start over to generate another set of palettes.

3.2 Three Color Palettes

Each palette uses a different algorithm to create five colors that are compatible with the user's starting color. To rid of any bias associated with the way each palette is generated, we randomize both the order of the palettes as they appear, as well as the individual colors within that palette.

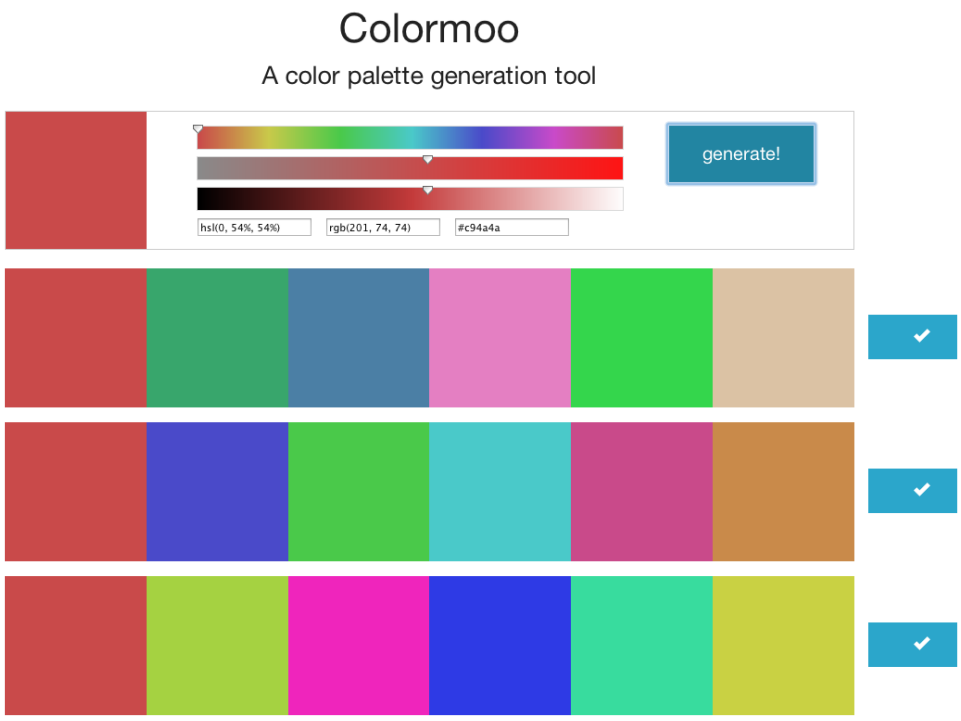

Figure 3.1: Colormoo enables users to input a color with a color specification or with the color sliders to generate three color palettes.

Figure 3.2: These three color wheels demonstrate the hue templates used in the theory palette. From left: complementary colors, analogous colors, and triadic colors. [Col12]

3.2.1 The Theory Palette

Like most tools that allow users to generate color themes based off of a starting color, the first palette is generated using color theory and hue templates. With the Grapefruit color conversion Python library, hue templates can be based on one of two underlying color wheels. The first is the conventional RGB color wheel in which colors are created by combining different levels of the three component colors. However, artists of more traditional media use red, yellow, and blue as primary colors. A more accurate color wheel for this case, often called the "artist's color wheel," is the Red-Yellow-Blue wheel. Since it is based on irregular physical properties pigment, it is generally believed that the RYB wheel is less scientifically accurate [Mac09]. For our hue templates, we use the RGB wheel to stay true to the properties of light, especially given that this tool will be used exclusively online.

We use a combination of complementary, analogous, and triadic hue templates to implement this palette (Figure 3.2). O'Donovan et al. found themes that incorporated complementary and analogous templates were among the most highly rated on Kuler [OAH11]. To add a little more variety, we also include the triadic template, where the colors are evenly spaced around the color wheel. This brings the total number of colors to five: one complimentary, two analogous, and two triadic colors.

3.2.2 The Random Palette

The second palette is comprised of five random colors. In this case, we utilize the RGB space by choosing a random number from 0 to 255 to represent each

of the color's associated RGB components. However, even to the untrained eye, it is unlikely that five random colors are going to look good together. Furthermore, there is no correlation between the derived colors and the starting color. Not only would the resulting palette be an unsatisfactory solution to the problem, but it would also appear too random. Users might be able to recognize it as random and, as such, be biased against it.

To account for this we first played with the idea of "mixing" the starting color with each random color. After creating a random color's RGB components, we add the corresponding component from the starting color and divide the sum by two. The result is a color whose RGB components are an average of the two colors. This is a better algorithm than purely random colors because it allows the user's choice to impact the resulting theme. However, while averaging proved to work well for some colors, if the starting color was too red, too blue, or too green (that is, having a high R, G, or B value) then the resulting palette was tinted as such. When placed next to the theory palette, for example, it was still fairly obvious that two different algorithms had been used. It wouldn't necessarily be interpreted as a random assortment of colors, but it was certainly distinct from the other palettes. Finally, it may be argued that the tinted theme contained colors that go well together, but the colors may be a little too similar to be considered a solution to this problem. Much like the theory palette, we want colors that go well together and offer variety.

Inspiration for a better solution came from Erica Schoonmaker on Dribbble [Sch11]. In one of her posts, Schoonmaker reveals that her secret for a cohesive color palette is to overlay a sixth color over a five color palette. She recommends the photoshop mask "soft light," but the idea is to add a very slight tint to the rest of the color palette. The result is a palette of different colors that subtly channel this outside color to become a little more united. We incorporate this idea into our previous solution. We were mixing the random color and the starting color evenly. Instead, we will decidedly use less of the starting color to let more of the random color shine through. Our final colors will be 10% the starting color and 90% the random color. These colors do not stick out from the other palettes as much as the previous iteration. Starting colors with high RGB components did not overwhelm the rest of the palette and, much like Schoonmaker's post, the colors in this palette appeared to be more consistent.

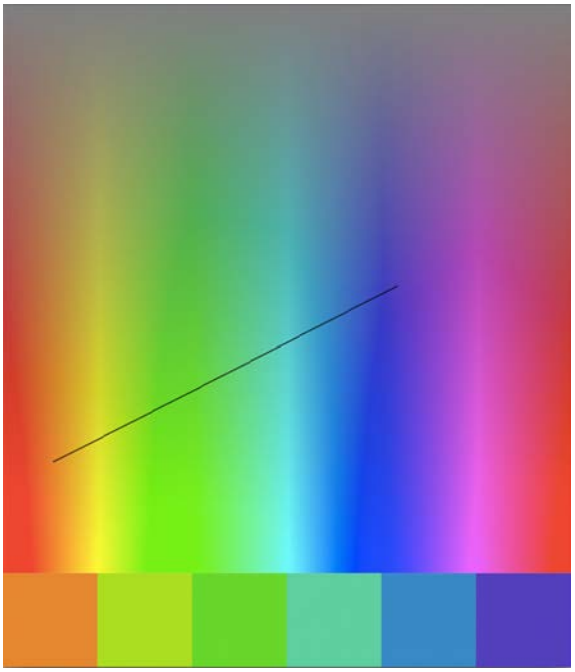

Figure 3.3: This is an example of our Pygame instance during the refinement of the equidistant palette. The x-axis represents hue and the y-axis is saturation. We sample colors evenly spaced along this line to create a color palette.

3.2.3 The Equidistant Palette

The third palette is generated using the technique demonstrated by Gregor Aisch (Figure 3.3). As mentioned earlier, the use of color in data visualization does not exactly match the goals of this project since we are not trying to find the best colors for representing data. However, by establishing a few heuristics and sampling more colors, this approach can also be used to create a palette for artistic purposes as well.

The value of equidistant colors is that they are evenly spaced in a way that makes sense to the human eye [Ais11]. For this reason, we use HSL rather than RGB because it is a more intuitive way for humans to think about color.

The key to Aisch's technique is determining which components to use on the x and y axis and which to hold constant. Since the hue component is

what we generally think of when we use the word "color," and therefore the component users will want control over, we can rule it out as the component that we should hold constant. However, deciding between keeping saturation or lightness constant is a little more difficult. If we keep saturation constant, the palette could very easily become divergent as the line plunges into the darker or lighter end of the space. If we keep lightness constant, the resulting color palette is a little more consistent. This is the option we ultimately settled on. Our transformation consists of hue along the x-axis and saturation on the y-axis. When the user selects a color, we use the lightness value of the starter color selected by the user.

Drawing a straight line in the space is done by using the starting color as the start point and randomly selecting a second color to serve as the end point. We define a couple heuristics to ensure that the line is ideal for deriving a good color palette. Specifically, we want to ensure the line spans across enough hues. If the line is too short, we might only sample red and yellow colors, for example. We enforce a restriction by calculating the distance between the two points and limiting it to a length of at least 180 units. In the same vein, we don't want a line that is too vertical because then we would sample colors with the same hue at various levels of saturation. To account for this we restrict the slope to being within the range of -1 and 1. For some colors this restriction may be a little conservative, however the results are still varied enough that this is not too much of a problem.

With two points on a line, we can use simple algebra to determine colors equidistant along the line. In theory we could find any number of equidistant points along this line, but after a certain point the colors become too similar. Their relationship to one another becomes more obvious as the sample begins to resemble a gradient between the two endpoints. Five points might be the maximum number of colors we can generate and still have variety, especially given the number of restrictions we have already placed on the kinds of palettes we can generate. Fortunately, for our case, that is all we need.

Chapter 4

Methodology

With Colormoo, our goal is to determine which, if any, of the algorithms produces the most favorable results. This chapter outlines the technology used in creating the web application and the data that was collected to justify our results.

4.1 Technology

The logic behind the palette generation is a single Python file that implements the algorithms described in Chapter 3. This file can be viewed in Appendix A. This file primarily utilizes Grapefruit, a Python library for performing basic color conversions between color spaces. Grapefruit was also used to calculate the hue templates for the theory palette and mix the colors used in the random palette.

Initial experimentation was done locally using the Pygame gaming framework. The first prototypes supported a command line interface for selecting colors whose corresponding palettes would be generated in the Pygame instance. It was in this stage of development that the heuristics for each palette were refined. An example of this prototyping environment can be seen in Figure 3.3.

With the logic written, we worked to bring the functionality online in order to test it with more users than we could locally. By publishing Colormoo online, we also hoped to create a tool that users could refer to later as part of their design workflow. This was done with a combination of the Django Python web framework and Heroku, a platform for web applications. We chose Django to avoid porting the logic to another language like Javascript. Django is also very well documented and tightly integrated with Heroku.

Heroku was chosen because it abstracts the complicated process of deploying an application on a remote server. This made it relatively easy to setup a development environment with the correct dependencies and enabled us to quickly push new changes to the site as necessary.

The frontend of Colormoo was implemented with Django's templating system, with standard HTML and CSS. The Zurb Foundation front-end framework is lightly utilized for creating the layout and improving the overall aesthetics of the interface. Most of the interactivity of the site is afforded by JQuery and Javascript. In particular, the color selecting interface is a JQuery plugin offered by Virtualsoft [Vir14]. This plugin was chosen because it offers both the color sliders and the color specification interfaces for selecting a color. It is also very customizable, which in our case allowed us to change the appearance to fit with the rest of the site.

We set up a Postgres database on Heroku to collect data on palette preference. From the site we use Django's database API to push data to the server. This is essentially a Python wrapper for the Postgres commands. We record each palette selection with a JQuery post request to an empty Django view. This view processes the request and pushes necessary figures to the database.

4.2 Data Collection

Upon visiting the site, users are instructed to select their favorite of three palettes generated from the starting color of their choice. Data is collected on both the palette they select (theory, random, or equidistant) and the color they used to generate that palette.

Our data is stored in two tables. The first is the choices table. It has a row for each of the three theme generating algorithms. It has three columns, the palette id, the palette's name, and a count for the number of times that the palette is selected. The second table is where we store the starting colors. We write to this table when the user has selected their favorite palette. Therefore, the number of rows is representative of the number of times the site was used effectively. It contains five columns: an id, the corresponding palette's id, and the associated hue, saturation, and lightness values of the color. We use this information to understand the relationship between the color and the selected palette. It is worth noting that we do not collect any data on unique visits to our site, therefore our data is only representative of the number of times that our site was used.

4.2.1 Users

Colormoo was shared on a number of websites where people might be interested in generating color palettes. We attribute most visits to the Hacker News (www.news.ycombinator.com) and Designer News (www.news.layervault.com) communities. These are forums where professional web developers and designers share techniques and offer insight into their workflow. For this reason, we believe our users are primarily concerned with the usefulness of colors as they relate to user interface and visual design. They may not be color experts, but we expect that they are approaching this problem with an experienced eye.

4.2.2 Postprocessing

The data required some postprocessing before we could begin analysis. Every time the app is loaded, the starting color is initially set to a pink color. To account for this oversight, we had to toss much of the data that included these initial component values. We also had to clean the data where it appeared that users were testing the capabilities of the application. This was made evident by duplicate rows of data in which the only difference was a single column value. This anomaly can be attributed to the lack of context that is provided for the application on the forums where it was published. Users may have repeatedly generated the same results to determine how to use the site. It could also be that users liked more than one of the palettes and wanted the color specifications for both. Unfortunately, with the current implementation, there is no other way to support this kind of behavior. To account for this potentially extraneous data, we only take the first instance of the repeated trials.

Chapter 5

Results

Colormoo was published online where it experienced traction as a result of being posted in two design-focused forums. In total, the app was used 1875 times. From the data collected in those sessions we are able to make conclusions about user color palette preference. Contrary to our initial hypothesis, there was no algorithm that was a clear favorite. Instead, users seemed to prefer both the theory and random palettes equally. The theory palette was chosen 663 times with the random palette close behind at 660. The equidistant palette was the least popular with a total of 552 votes. With such a close tie between the theory and random palettes, it makes sense to look at the different circumstances under which each palette was chosen.

5.1 Results by Component

5.1.1 Hue

There does not seem to be any correlation with the hue of a color and whether or not users preferred a particular palette. As we see in Figure 5.1, each palette is appropriately represented across all the hues that were generated on the site. On the one hand, this is a characteristic that we would like to have present in a good theme generating algorithm. If there is a situation in which hue is the only property we care about, any one of the three algorithms would be a good choice. On the other hand, this result prevents us from crafting an optimal algorithm that uses one technique for one range of hues and another technique for the remaining range.

While we can't speak to palette preference, this data does offer interesting insight into hue preference. In aggregate, the most popular hues sit in the 180-220 range, which represent blue colors. There is a large spike

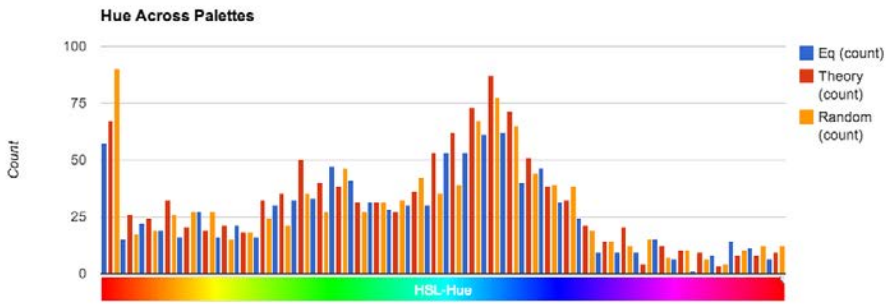

Figure 5.1: While there is no correlation between hue and a specific color palette, we can observe user preference for red and blue colors.

around pure red at 0 degrees, but this could also be attributed to red being located at the end of the slider. However, these results are consistent with previous studies of color preference that found a general preference for reds and blues [OAH11]. The least popular colors sit above 252 degrees. These represent violet and magenta respectively.

5.1.2 Saturation

When we look at the saturation of the starting colors selected by the user (Figure 5.2), we see more variation across the different palettes than we see with hue. In particular, as the starting color becomes more saturated, users tend to prefer the theory palette. This is most noticeable with levels of saturation above 70%. However as the starting color becomes less saturated the random palette becomes more popular. These two observations hold true especially for the most extreme values of saturation. When the starting color has 0% saturation, there is an overwhelming support for the random palette. Likewise, when the starting color has 100% saturation the theory palette is very clearly the favorite.

5.1.3 Lightness

Similar to saturation, there is more of a preference for one palette over the other for different values of lightness (Figure 5.3). With a lighter starting color, users preferred the theory palette. With the exception of the lightest colors in the 88-100% range, the theory palette was preferred for every color

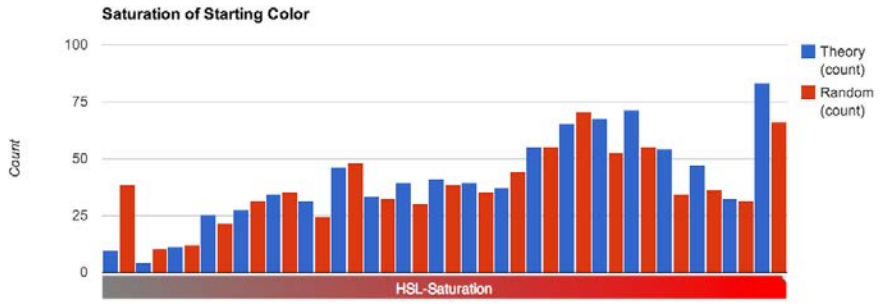

Figure 5.2: The theory palette is more popular with highly saturated colors.

above a 48% lightness value. Everything below that point was favored by the random palette.

5.2 Discussion

Intuitively it makes sense that for more saturated colors, users prefer the theory palette and for less saturated colors prefer the random palette. Saturation is essentially a measure of how dull or intense a color appears. In other words, saturation is the amount of color present in a particular color. When there is a lot of color present, it makes sense that users would recognize theoretical color relationships and prefer the palette that best utilizes that color intensity. When there is very little color present, users might appreciate the additional intensity afforded by a random set of colors.

The lightness results are a little more surprising. When users select a light starting color, the results imply that they are trying to produce a lighter color palette, such as the one generated using color theory. In this case the result is usually a pastel palette. However, when users select a dark color, the data suggests they are not trying to create a dark palette. Instead, as with less saturated colors, they choose the random palette which tends to be a brighter and more vibrant. Previous studies found a preference for brighter palettes [OAH11]. Our data supports these findings.

These two results indicate that the preferred palette does not contain colors that are too similar. A desaturated color will generate a desaturated theory palette. The palette appears muted because its colors are not visually distinct. The same effect can be observed with a dark starting color. The

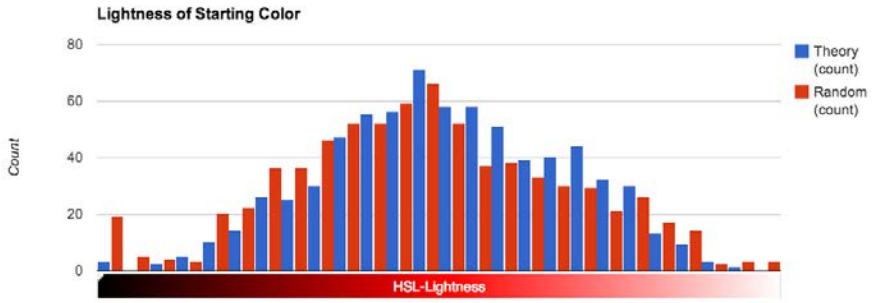

Figure 5.3: The theory palette is more popular with lighter colors while the random palette is preferred for darker colors.

random palette is a reasonable choice in these cases because it is the only algorithm that generates a palette with more variety. O'Donovan et al. found that palettes were more highly rated when they contained colors that weren't too distinct [OAH11]. If this is true, our data would suggest that the ideal palette is neither too distinct nor too similar.

We attribute most of our data to two web development and design communities online. We reasonably conclude that these findings are representative of the palettes most useful individuals of these professions. The value in a tool like Colormoo is that it defines a set of colors that can be used with a base color. Oftentimes designers work with clients that designate the use of a color or colors that are unique to their brand. Our data can offer guidance for successful use of those colors. For example, if a mandated color is a dark color, our data suggests incorporating brighter colors. Alternatively, if the mandated color is highly saturated, it might be safer to use a hue template rather than eye-balling it with an unrelated color.

Chapter 6

Future Work

There are several ways in which Colormoo can be improved. There are also different avenues in which these concepts can be applied outside the context of graphic design.

6.1 Refinement

Gregor Aisch actually advises against using equidistant colors with the HSL color space [Ais11]. The problem is that, though the colors are mathematically the same distance from each other when plotted out in the transformed color space, the colors may not necessarily be perceived to be equidistant by the human eye. In some cases the jump from one color to the next may seem larger than the subsequent jump. Aisch recommends using a more perceptually uniform color space like CIE LAB. While we experimented with it, this approach was omitted from Colormoo because CIE LAB contains gaps to account for the intricacies of human vision. It was difficult to develop a good set of heuristics and account for these gaps such that we could consistently offer a fourth palette. Future work might explore this concept a bit further. Perhaps the equidistant palette performed the worst because of the minor perceptual inconsistencies associated with HSL.

With the initial data on palette preference for different color values, we can further explore how adding heuristics might improve the experience. It might also help to consider the different ways in which these palettes are used. For example, in design work, it is not very often that each color is used the same amount. To generate a more useful palette, it might be worth exploring a way to determine which colors in the palette should be the principal colors and which colors should be secondary or only used as

accents.

6.2 Industry Application

A popular feature in hardware e-commerce is to allow the consumer to customize their product with the colors of their choice. With the growing production capabilities and increasing interest in 3D printing, it is not inconceivable that manufacturers will soon be able to produce a product in any color. While vendors want to leverage the utility that consumers derive from this feature, they do not necessarily want to allow consumers to choose any color that they want. Doing so has the potential to damage the brand. Instead, they might allow users to select one color for customization purposes and then make recommendations for additional colors that could also be used. As this trend becomes more widely accepted, the problem of finding compatible colors will become more of a concern and there will be a demand for algorithms similar to those used throughout this project.

Chapter 7

Conclusion

There are many ways to derive a set of colors that are compatible with one another. Color experts across a variety of professions each have their own solutions to this problem. We wanted to measure the circumstances under which each technique was most successful.

We built Colormoo to evaluate three algorithms for generating color palettes based off of a starting color. We found that the palette based on color theory was most popular with highly saturated and light colors. When the starting color is less saturated or dark, users prefer the random palette. This may be because it produces palettes with more visually distinct colors. Designers can take advantage of these findings by ensuring that their work incorporates colors that are compatible, but not too similar.

We also note a general preference for red and blue colors as they were the ones most frequently input as the starting color. More extensive research needs to be done to determine how this might fit into a solution to the problem of generating color palettes.

Our work provides a starting point for future exploration of the topic. We believe that with refined heuristics and a better understanding of the way humans perceive color, these algorithms can be improved and extended such that they can be applied beyond the field of graphic design.

Appendix A

Selected Source Code

Included here is the main python file for generating the three color palettes used on Colormoo. The only dependency is the python Grapefruit color conversion library. This code is provided "as is" without warranties or conditions of any kind.

```
import sys
import random
import grapefruit
import pdb
import ast
import math

random.seed(542)

color1 = grapefruit.Color.NewFromHsl(30, 1, .5)

def generateRandomColor(mix):
    """ Averages the RGB values of a random color and the input to
        effectively 'mix' the two colors
    """
    red = random.uniform(0,255)
    green = random.uniform(0,255)
    blue = random.uniform(0,255)

    red = (red + mix.rgb[0]) / 2
    green = (green + mix.rgb[1]) / 2
    blue = (blue + mix.rgb[2]) / 2
```

```
    newColor = grapefruit.Color.NewFromRgb(red, green, blue)
    return newColor

def RGBconvertToHTML(color):
    """ Takes a grapefruit color in RGB and converts it to HTML"""
    r = color.rgb[0] / 255
    g = color.rgb[1] / 255
    b = color.rgb[2] / 255
    return grapefruit.Color.RgbToHtml(r,g,b)

def averaged(myColor):
    # create dictionary of colors
    d = {}
    d["one"] = RGBconvertToHTML(generateRandomColor(myColor))
    d["two"] = RGBconvertToHTML(generateRandomColor(myColor))
    d["three"] = RGBconvertToHTML(generateRandomColor(myColor))
    d["four"] = RGBconvertToHTML(generateRandomColor(myColor))
    d["five"] = RGBconvertToHTML(generateRandomColor(myColor))
    return d

def generateBlendedColor(myColor):
    red = random.uniform(0,255)
    green = random.uniform(0,255)
    blue = random.uniform(0,255)
    randomColor = grapefruit.Color.NewFromRgb(red, green, blue)
    newColor = myColor.Blend(randomColor, 0.1)
    return newColor

def blended(myColor):
    # create dictionary of colors
    d = {}
    d["one"] = RGBconvertToHTML(generateBlendedColor(myColor))
    d["two"] = RGBconvertToHTML(generateBlendedColor(myColor))
    d["three"] = RGBconvertToHTML(generateBlendedColor(myColor))
    d["four"] = RGBconvertToHTML(generateBlendedColor(myColor))
    d["five"] = RGBconvertToHTML(generateBlendedColor(myColor))
    return d

def theory(myColor, myMode='rgb'):
```

```
    a1, a2 = myColor.AnalogousScheme(angle=30, mode=myMode)
    t1, t2 = myColor.TriadicScheme(angle=120, mode=myMode)
    c = myColor.ComplementaryColor(mode=myMode)

    # create dictionary of colors
    d = {}
    d["one"] = a1.html
    d["two"] = a2.html
    d["three"] = t1.html
    d["four"] = t2.html
    d["five"] = c.html
    return d

def blendedTheory(myColor, myMode='rgb'):
    a1, a2 = myColor.AnalogousScheme(angle=30, mode=myMode)
    t1, t2 = myColor.TriadicScheme(angle=120, mode=myMode)
    c = myColor.ComplementaryColor(mode=myMode)

    a1b = myColor.Blend(a1, 0.07)
    a2b = myColor.Blend(a2, 0.07)
    t1b = myColor.Blend(t1, 0.07)
    t2b = myColor.Blend(t2, 0.07)
    cb = myColor.Blend(c, 0.07)

    # create dictionary of colors
    d = {}
    d["one"] = a1b.html
    d["two"] = a2b.html
    d["three"] = t1b.html
    d["four"] = t2b.html
    d["five"] = cb.html
    return d

################################################
#           HSL COLOR SPACE CODE               #
################################################

# 360/250 = 1.44
hue = [x*1.44 for x in range(250)]
sat = [x*0.4 for x in range(250)]
```

```
def distance(x0,y0,x1,y1):
    return math.sqrt((float(x1-x0)**2) + ((y1-y0)**2))

def slope(x0,y0,x1,y1):
    if (x1-x0) == 0: return float("inf")
    return float(y1-y0)/(x1-x0)

def midpoint(x0,y0,x1,y1):
    return (x0+x1)/2, (y0+y1)/2

def newRandomColor(mix):
    """ Generates a new color with the lightness of the input color
    """
    h,s,l = mix.hsl
    hue = random.uniform(0,360)
    sat = random.random()
    lineColor = grapefruit.Color.NewFromHsl(hue, sat, l)
    return lineColor

def validColor(original, chosen):
    """ Returns true if the chosen color meets the distance and slope requirements
        relative to the original color
    """
    oh, os, ol = original.hsl
    ch, cs, cl = chosen.hsl
    dist = distance(oh, os, ch, cs)
    m = slope(oh, os, ch, cs)
    if dist >= 180:
        if -1 < m < 1:  return True
        else:  return False
    else: return False

def genNewColor(myColor):
    """ Returns a valid random color relative to the input color
    """
    lineColor = newRandomColor(myColor)

    while not (validColor(myColor, lineColor)):
        lineColor = newRandomColor(myColor)
```

```
    return lineColor

def fiveColors(myColor):
    endColor = genNewColor(myColor)
    oh, os, ol = myColor.hsl
    ch, cs, cl = endColor.hsl

    # find the midpoint color
    midHue, midSat = midpoint(oh, os, ch, cs)
    middleColor = grapefruit.Color.NewFromHsl(midHue, midSat, ol)

    # find the second color (midpoint of first and the newly generated midpoint)
    sndHue, sndSat = midpoint(oh, os, midHue, midSat)
    sndColor = grapefruit.Color.NewFromHsl(sndHue, sndSat, ol)

    # find the fourth color
    frthHue, frthSat = midpoint(midHue, midSat, ch, cs)
    frthColor = grapefruit.Color.NewFromHsl(frthHue, frthSat, ol)

    # create dictionary of colors
    d = {}
    d["one"] = myColor.html
    d["two"] = sndColor.html
    d["three"] = middleColor.html
    d["four"] = frthColor.html
    d["five"] = endColor.html
    return d

def generate(hue, sat, light):
    myColor = grapefruit.Color.NewFromHsl(hue, sat, light)
    d = {}
    d["original"] = myColor.html
    d["averaged"] = averaged(myColor)
    d["blended"] = blended(myColor)
    d["theory"] = theory(myColor)
    d["blendedTheory"] = blendedTheory(myColor)
    d["equidistant"] = fiveColors(myColor)
    return d
```

Bibliography

[Ais11] Gregor Aisch. How to avoid equidistant hsv colors. http://vis4.net/blog/posts/avoid-equidistant-hsv-colors/, 2011.

[Col12] Tiger Color. Color harmonies. http://www.tigercolor.com/color-lab/color-theory/color-harmonies.htm, 2012.

[HB03] Mark Harrower and Cynthia A. Brewer. Colorbrewer.org: An online tool for selecting colour schemes for maps. *Cartographic Journal*, 40(1):27 – 37, 2003.

[HLT09] Y-C. Hu, M-G. Lee, and P. Tsai. Colour palette generation schemes for colour image quantization. *Imaging Science Journal*, 57(1):46 – 59, 2009.

[Inc00] Adobe Systems Incorporated. Color models: Cielab. http://dba.med.sc.edu/price/irf/Adobe_tg/models/cielab.html, 2000.

[Jac13] Mathieu Jacomy. I want hue. http://tools.medialab.sciences-po.fr/iwanthue/theory.php, 2013.

[Mac09] Bruce MacEvoy. The geometry of color perception. http://www.handprint.com/HP/WCL/color2.html, 2009.

[OAH11] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. Color compatibility from large datasets. *ACM Trans. Graph.*, 30(4):63:1–63:12, July 2011.

[Sch11] Erica Schoonmaker. My secret for color schemes. https://dribbble.com/shots/166246-My-Secret-for-Color-Schemes, 2011.

[Vir14] Virtuosoft. jquery color picker sliders. http://www.virtuosoft.eu/code/jquery-colorpickersliders/, 2014.