Shade Ranking
MANUSCRIPT

William F. Barnes
*

March 4, 2024

Contents

1 Introduction

Modern displays implement at least $255^3 \approx 1.7 \times 10^7$ unique shades in RGB color space, which is an order of magnitude greater than the number of words in the English language. One may suspect, then, that a 'named' color occupies more than one location in the RGB volume.

For instance, we agree that the combination $(255, 0, 0)$ is unequivocally *red*. However, by consulting any[1] mainstream RGB color-name tool, one also finds that variations such as $(239, 4, 5)$, $(235, 14, 15)$, and many others - also classify as 'red'.

1.1 Database Uniqueness Problem

Suppose you are tasked with gathering a comprehensive database of color records, with each record consisting of a label and the RGB triplet representing that color. For this study, we work with $\approx 10^4$ records traced to fair-use publication.

When merging color databases, a problem arises among colors whose label is the *same* among sources, but whose RGB values are *different*. To show some real data, the following values for 'red' may end up in the database:

```
"red", 131, 43, 41
"red", 238, 0, 0
"red", 196, 2, 51
"red", 237, 41, 57
"red", 205, 0, 0
"red", 237, 28, 36
"red", 242, 0, 60
"red", 254, 39, 18
"red", 255, 0, 0
"red", 238, 32, 77
"red", 139, 0, 0
"red", 229, 0, 0
```

While each of the above may technically be 'red', with some appearing quite close to the 'pure' tone $(255, 0, 0)$, we acknowledge the list is completely unordered.

[1] https://www.color-blindness.com/color-name-hue/

The collision of labels poses a new problem: assuming the finished database is scanned from top to bottom by a program, it may be required that $(255, 0, 0)$ occur at the top of the list. That is, there is no natural reason to define 'red' as whatever record occurs first. Priority ought to be given to $(255, 0, 0)$.

Moreover, whatever solution is applied to the 'red' should do something to sort all other contenders, for if $(255, 0, 0)$ was not present, the next-best choice should be discerned.

The so-called *uniqueness problem* problem is present for not only red, but also for green, blue, cyan, magenta, yellow, and any other shade that has duplicated definition.

Of course, one may manually rearrange the database so as to prioritize pure tones. It would be more elegant, though, to come up with a tool that can be applied to an unsorted database, sending all records with overlapping color names through the same pipeline. The result must have like-named colors in arranged descending order with the 'pure' tones, such as $(255, 0, 0)$, or $(0, 255, 255)$, occurring at the top of their respective groups.

1.2 Notion of Rank

Reaching for the standard tools of color space, it's worth trying to use luminance, saturation, chroma, etc., as a sorting tool to prioritize pure tones. After a bit of experimentation (not reviewed here), none of these turn out to be general enough to handle the full database. A new quantity is needed.

For lack of an existing term, let us associate *rank* with the notion of 'sorting like-named colors in descending order with the pure tone occurring first'. The top of the list has the highest rank.

2 Prototype Ranking Function

To advance on the problem, consider any given shade $C = (r, g, b)$ and normalize by dividing 255 from each component:

$$C_0 = \left(\frac{r}{255}, \frac{g}{255}, \frac{b}{255} \right) = (r_0, g_0, b_0)$$

2.1 Delta Compliment

Next, construct the difference between each pair of normalized components:

$$\begin{aligned} \tilde{b} &= |r_0 - g_0| \\ \tilde{r} &= |g_0 - b_0| \\ \tilde{g} &= |b_0 - r_0| \end{aligned}$$

The tilde-variables $\tilde{r}$, $\tilde{g}$, $\tilde{b}$ are defined as the respective absolute value of each difference pair, and can qualitatively be regarded as 'delta compliments' to their r_0, g_0, b_0 counterparts. For example, the pure red tone $(1, 0, 0)$ minimizes $\tilde{r}$, but maximizes $\tilde{b}$, $\tilde{g}$. Meanwhile though, the full cyan $(0, 1, 1)$ bears the same result, namely a minimum for $\tilde{r}$ and maxima for $\tilde{b}$, $\tilde{g}$. Similar statements apply for the pair blue/yellow, and also for green/magenta.

2.2 Element-wise Product

Starting with a normalized color vector

$$C_0 = (r_0, g_0, b_0) \ ,$$

let us seek to transform C_0 into a new vector C_0' by rescaling each component in the least assuming way, i.e.

$$C_0' = (r_0', g_0', b_0') = (\alpha r_0, \beta g_0, \gamma b_0) \ ,$$

where the coefficients α, β, γ carry of the information of how each component r_0, g_0, b_0 is modified.

The combination $(\alpha r_0, \beta g_0, \gamma b_0)$ has a name called the *element-wise product*, thus we can separate the color components from the unknown coefficients using the notation

$$C_0' = C_0 \circ Y = (r_0, g_0, b_0) \circ (\alpha, \beta, \gamma) \ ,$$

where the vector Y contains the components α, β, γ:

$$Y = (\alpha, \beta, \gamma)$$

2.3 Ranking Coefficients

Now we must decide what the *ranking coefficients* α, β, γ actually *do*. Using only the data on hand, and knowing that When components begin to mix, the ranking coefficients tend to decrease, let us try

$$\begin{aligned} 2\alpha &= \tilde{g} + \tilde{b} = |b_0 - r_0| + |r_0 - g_0| \\ 2\beta &= \tilde{b} + \tilde{r} = |r_0 - g_0| + |g_0 - b_0| \\ 2\gamma &= \tilde{r} + \tilde{g} = |g_0 - b_0| + |b_0 - r_0| \ , \end{aligned}$$

where α 'leans heavy' on r_0, and a similar comment applies to the pairs β, g_0 and γ, b_0.

Of course, there are many ways to pose the ranking coefficients, but the decision we've made is perhaps one of the simplest nontrivial ones.

2.4 Computing the Rank

Using the rank-aware vector $C_0' = C_0 \circ Y$, the next task is to turn this into a single number N, the color's final rank. One's first instinct may be to calculate square of the vector's magnitude, i.e.

$$C_0' \cdot C_0' = r_0' r_0' + g_0' g_0' + b_0' b_0' \,.$$

Note though that the terms on the right now involve the square of the ranking coefficients.

A less aggressive result is attained by projecting C_0' onto the unaltered vector C_0 such that

$$\begin{aligned} N &= C_0 \cdot C_0' = C_0 \cdot C_0 \circ Y \\ &= r_0^2 \alpha + g_0^2 \beta + b_0^2 \gamma \,, \end{aligned}$$

which can be framed by standard scalar products between vectors:

$$\begin{aligned} N &= \left(r_0^2, g_0^2, b_0^2\right) \cdot (\alpha, \beta, \gamma) \\ &= \frac{1}{2} \left(r_0^2, g_0^2, b_0^2\right) \cdot \left(\tilde{g} + \tilde{b}, \tilde{b} + \tilde{r}, \tilde{r} + \tilde{g}\right) \end{aligned}$$

Finally, note that the rank N can be uniformly scaled by a constant k without affecting the relative ranking between any two shades. For our answer then, let us export the final rank as the number:

$$N = \frac{k}{2} \left(r_0^2, g_0^2, b_0^2\right) \cdot \left(\tilde{g} + \tilde{b}, \tilde{b} + \tilde{r}, \tilde{r} + \tilde{g}\right) \,, \quad (1)$$

where k shall be called the *ranking constant*.

2.5 Implementation

The role of rank enters the database sorting problem as equation (1).

Supposing some program has prepared an isolated array of color records, each competing for the same label, such as 'red', we send that array to a standard *QuickSort* routine and receive the rank-sorted result. In pseudocode notation, the code that does this work is listed below.

```
Rank (rgb)
    (rgb0) = (rgb)/255
    drg = Abs(r0 - g0)
    dgb = Abs(g0 - b0)
    dbr = Abs(b0 - r0)
    return k * .5 * (
        r0^2 * (drg+dbr) +
        g0^2 * (dgb+drg) +
        b0^2 * (dgb+dbr) )

QSort (Low, High, dat())
    If (Low < High)
        piv = Part(Low, High, dat())
        QSort(Low, piv - 1, dat())
        QSort(piv + 1, High, dat())

Part (Low, High, dat())
    piv = Rank(dat(High).Clr)
    i = Low - 1
    For j = Low To High - 1
        tmp = Rank(dat(j).Clr) - piv
        If (tmp >= 0)
            i = i + 1
            Swap dat(i), dat(j)
    Swap dat(i + 1), dat(High)
    return i + 1
```

2.6 Results

We check for efficacy by sampling the outputted database. Following are the top few records for the RGB-CMY hexlet, which have emerged sorted in accordance with our wishes.

```
"red", 255, 0, 0
"red~2", 238, 32, 77
"red~3", 237, 41, 57
"red~4", 254, 39, 18
"red~5", 242, 0, 60

"green", 0, 255, 0
"green~2", 102, 176, 50
"green~3", 28, 172, 120
"green~4", 0, 238, 0
"green~5", 0, 168, 119

"blue", 0, 0, 255
"blue~2", 99, 45, 233
"blue~3", 31, 117, 254
"blue~4", 2, 71, 254
"blue~5", 0, 135, 189

"cyan", 0, 255, 255
"cyan~2", 0, 255, 254
"cyan~3", 0, 205, 205
"cyan~4", 0, 183, 235
"cyan~5", 0, 238, 238

"magenta", 255, 0, 255
"magenta~2", 255, 0, 254
"magenta~3", 255, 0, 144
"magenta~4", 208, 65, 126
"magenta~5", 202, 31, 123

"yellow", 255, 255, 0
"yellow~2", 255, 254, 0
"yellow~3", 205, 205, 0
"yellow~4", 255, 239, 0
"yellow~5", 255, 211, 0
```

3 Grand Ranking Function

Now we address an oversight of the ranking apparatus developed above, which is to admit that the ranking function solves the uniqueness problem for colors based on hue alone, but happens to ignore white, black, or any shade of grey.

3.1 Hue Factor

Start by reclassifying the rank equation (1) to signify its emphasis on hue, relabeling the rank number N to *hue factor* $N_{\mathcal{H}}$,

$$N_{\mathcal{H}} = \frac{1}{2}\left(r_0^2, g_0^2, b_0^2\right) \cdot \left(\tilde{g} + \tilde{b}, \tilde{b} + \tilde{r}, \tilde{r} + \tilde{g}\right)$$

with ranking constant $k = 1$.

3.2 Black-White Factor

Next, introduce a *black-white factor* N_B that ranks all occurences of 'white' under the value $(255, 255, 255)$, while simultaneously ranking all 'black' under the value $(0, 0, 0)$.

For a normalized input $C_0 = (r_0, g_0, b_0,)$, we first jot down the normalized average intensity

$$I_A = \frac{r_0 + g_0 + b_0}{3} \ ,$$

and then divine the following form for N_B:

$$N_B = \left|\cos\left(2\pi I_A\right)\right|^q$$

Any suitable N_B, not necessarily limited to the cosine function, must exhibit maxima at $I_A = 0$ and $I_A = 1$. The 'sharpness' of the function is determined by q, tuned at $q = 0.5$ for this study.

3.3 Greyscale Factor

The black-white factor makes a mess out of any database colors related to grey, as shades resembling black *or* white each achieve high rank.

The fix for this is a *greyscale factor*, which aims to emphasize a 'quintessential grey' corresponding to $I_A = 0.5$, and then ranking downward from there. To this end, we introduce the term N_G and write:

$$N_G = 1 - \left|2I_A - 1\right|$$

3.4 Grand Rank Factor

With three contributions to the rank in hand, namely N_H, N_B, N_G, the next job is combine these into a single *grand rank*. The calculation must 'weigh' each of the contributing factors that lets each thrive in its regime.

For instance, when ranking a list of 'yellow' shades, we want N_H to do the heavy lifting, as N_B and N_G will produce no meaningful input. Meanwhile, when sifting through greys, we don't need N_H getting in the way, and so on.

Proceed by introducing three simultaneous weight factors defined as

$$\begin{aligned} W_H &= \left(1 - N_B\right)\left(1 - N_G\right) \\ W_B &= \left(1 - N_G\right)\left(1 - N_H\right) \\ W_G &= \left(1 - N_H\right)\left(1 - N_B\right) \ , \end{aligned}$$

along with the numeric offset N_0:

$$N_0 = \left(1 - N_H\right)\left(1 - N_B\right)\left(1 - N_G\right)$$

At this stage, it's important to point out that each of N_H, N_B, N_G are defined to be already normalized, meaning their outputs must be confined between zero and one, inclusive.

The grand rank factor $N_{\mathcal{G}}$ is

$$N_{\mathcal{G}} = k\left(\frac{W_H N_H + W_B N_B + W_G N_G - N_0}{W_H + W_B + W_G}\right) \ , \quad (2)$$

rectifying the shortcomings of equation (1). The output is normalized by construction.

Ranking Constant

As done with the prototype rank function N_H, t's harmless to throw a global constant k onto the result does doesn't affect the relative rank. The ranking constant k is like an auxiliary dimension of its own, which has the qualitative effect of redirecting the contour of maximal purity.

3.5 Results: Black, White, Grey

In addition to the pure tones, the grand ranking function properly handles black, white, grey:

```
"black", 0, 0, 0
"black~2", 1, 1, 1
"black~3", 20, 20, 20
"black~4", 30, 30, 30
"black~5", 35, 35, 35

"white", 255, 255, 255
"white~2", 254, 254, 254
"white~3", 245, 245, 242
"white~4", 235, 235, 242
"white~5", 237, 237, 237
```

```
"grey", 128, 128, 128
"grey~2", 127, 127, 127
"grey~3", 126, 126, 126
"grey~4", 130, 130, 130
"grey~5", 125, 125, 125

"grey~6", 3, 3, 3
"grey~7", 252, 252, 252
"grey~8", 5, 5, 5
"grey~9", 250, 250, 250
"grey~10", 133, 133, 133
```

3.6 Results: Total Database

A qualitative look at the the sorted database is worth having. To establish a point of comparison, Figure 1 shows each color in the database listed alphabetically by label. Beyond their labels being alphabetized, there is no 'global' rule governing the overall pattern in the Figure.

For a few points of comparison, Figure 2 shows the same database sorted by HSV-saturation, with the most saturated colors at the top. Similar plots in Figures 3, 4 show the color database sorted by HCY-chroma, and luminosity, respectively.

With the eye trained on Figures 1-4, we may contrast these to depictions of the rank-sorted database. Trying the 'prototype' rank, represented by equation (1), produces Figure 5. Immediately we see this does not resemble any previous Figure, hinting that rank is not redundant to the 'usual' color space dimensions. Toward the bottom of Figure 5 are shades of black, white, and grey dancing around without guidance.

Finally depicted in Figure 6 is the grand rank-sorted database according to equation (2)which we take as the 'final answer' to the sorting problem. At the top of the sorted list are pure tones for red, green, blue, cyan, magenta, yellow, white, black, and grey. There is *some* trend going downward in Figure 6, somewhat akin to HSV-saturation or HCY-chroma, however the differences outnumber the similarities.

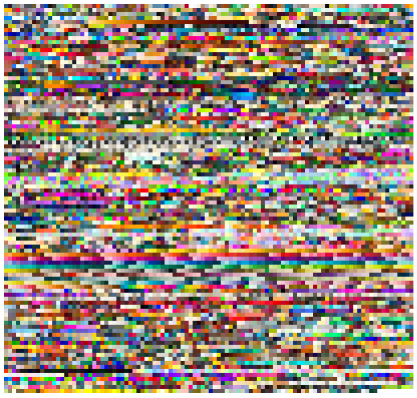

Figure 1: Color database unsorted.

Figure 2: Color database sorted by HSV-saturation.

Figure 3: Color database sorted by HCY-chroma.

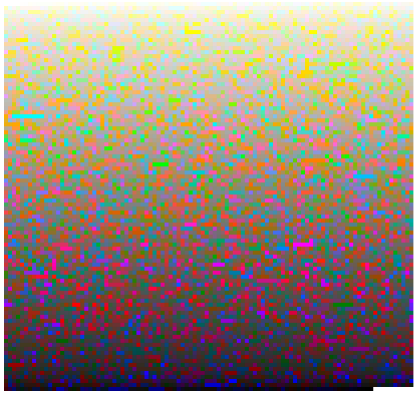

Figure 4: Color database sorted by luminosity.

Figure 5: Color database sorted by hue-rank $N_{\mathcal{H}}$.

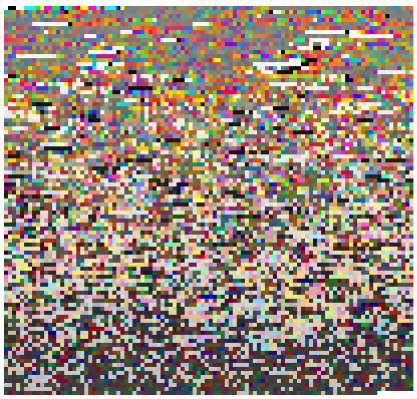

Figure 6: Color database sorted by grand rank $N_{\mathcal{G}}$.

4 Generalized Rank

Having completed its main job, the grand rank factor represented by equation (2) can be modified or altogether replaced for various ends.

4.1 Vividness

Colors that are bright, bold, vibrant, etc., tend to stand out as being *vivid*. These terms don't tend to map to one color dimension alone, namely saturation, chroma, or luminosity. In a sense, vividness is captured by all three of these.

The ranking apparatus has its own opinion on 'vividness'. Introduce the rank factor

$$N_{\mathcal{V}} = k\left(\frac{\tilde{r}+\tilde{g}+\tilde{b}}{r_0+g_0+b_0}\right) \tag{3}$$

as a replacement to equation (1) or equation (2). Sorting the color database on this leads to Figure 7.

Figure 7: Color database sorted by vividness $N_{\mathcal{V}}$.

4.2 General Rank Factor

In building equation (2) , the factors representing the RGB-CMY hexlet, along with those weighing the presence of black/white and grey, can be generalized into arbitrary terms $\{N_j\}$. In total, there can be not just three, but any number M of these factors.

The generalization of equation (2) thus reads

$$N_{\mathcal{R}} = k\left(\frac{\sum_{j=1}^{M} W_j N_j - N_0}{\sum_{j=1}^{M} W_j}\right) , \tag{4}$$

where

$$N_0 = \prod_{j=1}^{M} 1 - N_j \, ,$$

and

$$W_j = \prod_{n=1}^{M} 1 - N_{n\neq j} \, .$$

Generalized Rank Factors

The rank factors $\{N_j\}$ appearing in equation (4) are defined such that

$$0 \leq N_j \leq 1 \, ,$$

where $j = 1, \ldots, M$. Beyond this, each N_j can take any form one desires.

4.3 Palette-Based Ranking

A trick made easy by the ranking apparatus is database sorting by *palette*. For instance, suppose we sample $M = 3$ shades from *The Starry Night* by Vincent van Gogh, (1889). Choosing three shades that

represent the whole painting (Figure 8), we take:

$$\begin{aligned} Q_1 &= (219, 144, 28) \\ &= \textit{Carrot orange} \\ Q_2 &= (72, 136, 200) \\ &= \textit{Cyan-blue azure} \\ Q_3 &= (23, 54, 121) \\ &= \textit{St. Patrick's blue} \end{aligned}$$

Figure 8: *The Starry Night*, Vincent van Gogh, (1889).

To proceed, we need to construct $N_{1,2,3}$ such that a given database color C is compared to $Q_{1,2,3}$ and assigned the appropriate rank. The 'closer' C is to any of $Q_{1,2,3}$, the higher the rank. To this end, let N_j take the form

$$N_j = 1 - \frac{\sqrt{(\Delta R_j)^2 + (\Delta G_j)^2 + (\Delta B_j)^2}}{255\sqrt{3}},$$

where:

$$\begin{aligned} \Delta R_j &= C_R - Q_{jR} \\ \Delta G_j &= C_G - Q_{jG} \\ \Delta B_j &= C_B - Q_{jB} \end{aligned}$$

Sorting the color database this way gives rise to Figure 9. Ranked highest are shades mostly likely to occur in the original work, whereas those unlikely to occur are ranked lowest.

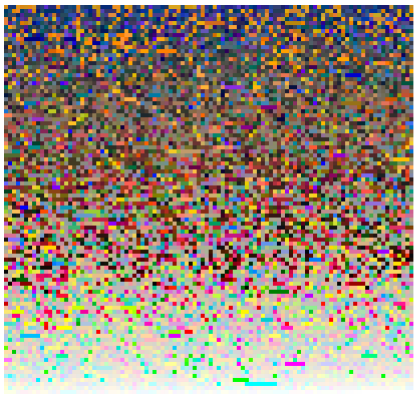

Figure 9: Color database sorted by likeness to palette used in *The Starry Night*.