Quite often when you just want to whip out some quick plots, I like to assign the colours by colour names. Some of my go-to colours are… tomato,darkorange,turquioise,darkgrey,lightgrey.

Recently I realized that I can get all of built-in colour names available in R to use by typing colors() or colours(). So I thought I’ll do bit of studying on colour names.

After staring at colour names for bit, it really made me wonder who has came up with these selection of colours in first place. I liked that there’s colour name like “lemon chiffon”, “peach puff”, “papaya whip”, “mint cream”, “chocolate” – although chocolate didn’t seem like the colour I’d typically associate with chocolate myself.

In general, built-in colour names looked quite random, aside from the fact that some colours have variations with suffix 2,3,4,5 etc. There are pale, light, medium, dark, deep colour variations too, but it didn’t seem like not every colour comes with those variations either.

So I wanted to see if there’s anyway I can visualize the built-in colours.

## Here's list of packages I'm going to use.

library(tidyverse) ## For everything...

library(tidygraph) ## Goew well with ggraph :)

library(ggraph) ## Been wanting to use this!

library(ggforce) ## Another amazing package I've been wanting to test

library(scales) ## Making Rescaling Easy

library(tidystringdist) ## stringdist but tidy!

library(data.tree) # To make tree from data frame

## display 15 random colours out of 657 colour names.

colors() %>% sample(size=15)

## [1] "gray73" "grey15" "grey91" "grey9"

## [5] "cornsilk4" "darkgoldenrod3" "mintcream" "grey"

## [9] "grey3" "gold2" "gray26" "white"

## [13] "salmon2" "gray9" "slategray4"

**R-Built In Colours – Abstract Plot**

First, I’ve decided to make nice tibble from the vector of colour names, then plot them using ggplot2. I have really wanting to test out [ggforce package](https://ggforce.data-imaginist.com/).

## color\_df will contain all of colours that are built-in with extra info about the colour.

color\_df <- tibble(

name = colors(distinct=F)

) %>%

mutate(idx = row\_number(), # original sorting order

rgb = map(name,col2rgb), #color name can be converted to rgb

hsv = map(rgb, rgb2hsv), #rgb can then be converted to hsv (I like using hue)

h = map\_dbl(hsv,1), # get the hue from hsv

s = map\_dbl(hsv,2), # get the saturation from hsv

v = map\_dbl(hsv,3), # get the lightness? darkness? from hsv

hue\_angle = rescale(h,to=c(0,2\*pi), from=c(0,1)), # convert hue value to angle

name\_parent = str\_remove(name,"\\d+"), # some colours have numeric variation of same color

name\_num = as.numeric(replace\_na(str\_extract(name,"\\d+"),0))) %>%

add\_count(name\_parent, name="n\_children") %>%

select(-rgb,-hsv) %>% ## just discarding from the tibble to make it smaller.

filter(n\_children<10) ## this will remove many shades of grey|gray family - there's over 100 of them.. So I'll just exclude them for this colour study.

## Also noticed that typically there's colour name with typical variations.

color\_df <- color\_df %>%

mutate(modifier=str\_extract(name,"(light|pale|medium|dark|deep)"),

name\_grandparent = str\_remove(name\_parent,"(light|pale|medium|dark|deep)"))

## Using hue & saturation to set x-coordnate, using hue & value for y-coordinate

color\_df <- color\_df %>%

mutate(x\_pos = dense\_rank(s\*cos(hue\_angle)),

y\_pos = dense\_rank(v\*sin(hue\_angle)))

## Colour Plots

base\_plot <-color\_df %>%

ggplot(aes(x=x\_pos,y=y\_pos)) +

# Connect the colours that is same family with dotted line

geom\_line(aes(group=name\_parent, color=name\_parent), linetype=3) +

geom\_point(aes(color=name, size=(name\_num+1)/n\_children)) +

# I want to put another circle around colour that are numeric variation of some colour.

geom\_point(aes(color=name, size=n\_children+1), shape=21,

data=. %>% filter(name!=name\_parent)) +

scale\_color\_identity() +

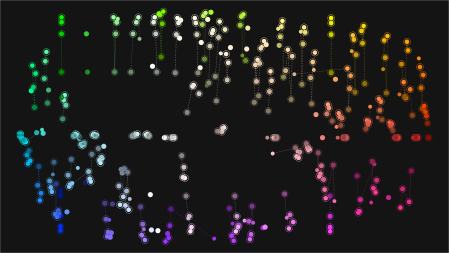
scale\_size\_continuous(guide="none", range=c(4,8)) +

theme\_void() +

theme(plot.background=element\_rect(fill="grey8"),

plot.title = element\_text(family="Roboto Condensed", color="white"))

base\_plot



Then I wanted to try geom\_mark\_\* function from ggforce ever since I saw some examples on twitter I’ve been wanting to try it myself.

annotate\_color <- function(x) {

cap\_text <- ifelse(x==1,"Colour with No Family", str\_glue("Colour with {x} Colours in a Family"))

base\_plot +

geom\_mark\_rect(aes(group=name\_parent, color=name\_parent,

label=str\_c(name\_parent)),

label.margin=margin(0,0,0,0,"mm"),

label.buffer=unit(1,"mm"),

label.fontsize=13, label.family="Roboto Condensed",

label.fill="#000000ae", con.colour="#ffffffae",

data = . %>% filter(n\_children==x),

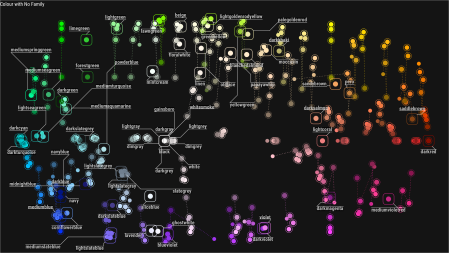
label.colour="#ffffffae",

con.cap=unit(0,"mm")) +

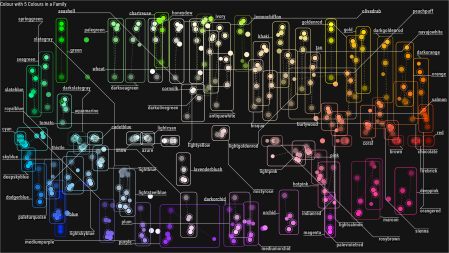
labs(title=cap\_text)

}

annotate\_color(1)



annotate\_color(5)



**Network Visualization of Built-in R Colours**

I recently saw the annoucement of updated [package ggraph](https://www.data-imaginist.com/2019/1-giraffe-2-giraffe-go/), and after seeing bunch of example plots inspired me to do below. Really amazing package!

I thought it would be really neat to put nodes the colour close together if colour names are similar. This has lead me to another great package called [tidystrtingdist](https://cran.r-project.org/package=tidystringdist)!

## first create tibble that list all of combination of colour names using tidy\_comb\_all function from tidystringdist package.

color\_comb <- tidy\_comb\_all(color\_df$name)

## then compute string distances using different method. q for qgram, jaccard, cosine, p for jw.

## I couldn't figure out which method is more suitable... Still trying to put my head around for the different string distance metrics.

color\_comb\_dist <- tidy\_stringdist(color\_comb,

method=c("jw","osa","jaccard","cosine","qgram","lv"),q=1,p=0)

## 1 take a peak at the table - What's similar to colour name tomato?

color\_comb\_dist %>% filter(V1=="tomato") %>% arrange(jw) %>% head(5) %>% knitr::kable()

| **V1** | **V2** | **jw** | **osa** | **jaccard** | **cosine** | **qgram** | **lv** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| tomato | tomato1 | 0.047619 | 1 | 0.2 | 0.0465374 | 1 | 1 |
| tomato | tomato2 | 0.047619 | 1 | 0.2 | 0.0465374 | 1 | 1 |
| tomato | tomato3 | 0.047619 | 1 | 0.2 | 0.0465374 | 1 | 1 |
| tomato | tomato4 | 0.047619 | 1 | 0.2 | 0.0465374 | 1 | 1 |
| tomato | whitesmoke | 0.400000 | 8 | 0.7 | 0.5435645 | 10 | 8 |

#color\_comb\_dist %>% filter(V1=="darkorange") %>% arrange(jw) %>% head(5)

## How is distance metrics distributed?

color\_comb\_dist %>%

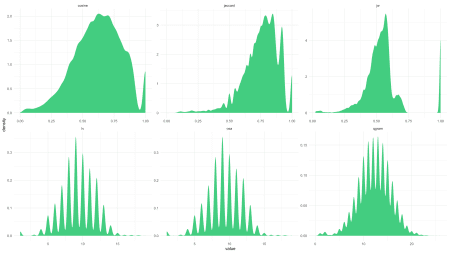
gather(key="method", value="value",-V1,-V2) %>%

ggplot(aes(x=value)) +

stat\_density(fill="seagreen3") + #I'm going to use some colour that I learned that exist!

facet\_wrap(~method, scales="free") +

theme\_minimal()



# Cosine, Jaccard, Jaro - distances is expressed between 0 and 1, 0 being exact match, 1 totally different.

# lv, osa, qgram - integers value, smaller the value, similar the strings.

I’ve decided to use Jaro distance for below. I didn’t want all of nodes (colours) connected to every other colours in network visualization, so I’ve chosen threshold of 0.25 to filter out the edges I don’t want to be present in network visualization.

## Creating Edges & Nodes Data Frame First

edges <-color\_comb\_dist %>%

select(from=V1, to=V2, jw) %>%

filter(jw<0.25)

nodes <- color\_df %>% select(label=name,x=x\_pos,y=y\_pos, everything()) ## I don't need everything, but thought I'd just keep extra attributes about nodes to play around.

g <- tbl\_graph(nodes=nodes, edges=edges, directed=F) %>%

mutate(deg = centrality\_degree(),

hub = centrality\_hub(),

comm = group\_edge\_betweenness(),

is\_parent = (label==name\_parent)) ## if colour name does't have variation then name shoudld be same.

g %>%

ggraph(layout="fr") + ##graphopt, kk, drl, fr , dh - i like fr for this graph

geom\_edge\_fan(alpha=0.25, color="white") +

geom\_node\_point(aes(color=label, size=deg)) +

geom\_node\_text(aes(filter=(is\_parent), label=label, color=label),

repel=T, family="Roboto Condensed", size=3, angle=10) +

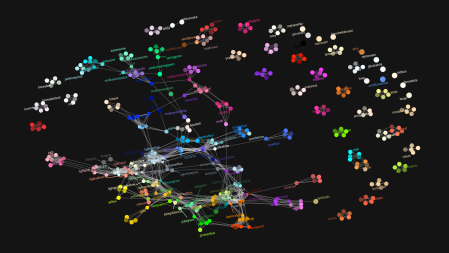
#geom\_mark\_ellipse(aes(x=x,y=y,group=comm), color="snow", linetype=3) + ## if you want to see how community is formed.

scale\_color\_identity() +

scale\_size\_continuous(range=c(6,2), guide="none") +

theme\_graph() +

theme(plot.background=element\_rect(fill="gray8"))



### You can always save the png...

## ggsave("~NetworkViz\_R\_BuiltInColour.png", width=16\*1.5, height=9\*1.5)

After studying the above network visualization, I realized the ones clustered together closely are the ones that has word “light”,“pale”,“deep”,“dark”, “medium” in general, but not all colours comes in all these variations. It also made me think the differences between light vs pale, dark vs deep too.

**Extra: Still trying to demysitify the Colours…**

library(data.tree)

## light, pale, medium, dark, deep... when are these word used?

color\_df <- color\_df %>%

mutate(modifier=str\_extract(name,"(light|pale|medium|dark|deep)"),

name\_grandparent = str\_remove(name\_parent,"(light|pale|medium|dark|deep)"))

## Make it into tree format

color\_df\_unique <- color\_df %>%

mutate(pathString=paste("color\_r",name\_grandparent,

ifelse(name\_parent!=name\_grandparent,name\_parent,""),

ifelse(name!=name\_parent,name,""),sep="/"))

color\_tree <- as.Node(color\_df\_unique)

color\_tree\_g <- as\_tbl\_graph(color\_tree) %>% ## there's no nodes table at this stage...

as\_tbl\_graph() %>% ## this seems to do what i want?

mutate(color=ifelse(name %in% colors(),name,"#ffffff"),

is\_rcolor=(name %in% colors())) %>%

left\_join(color\_df %>% select(name, h,s,v, modifier,n\_children), by="name") %>%

arrange(h,s,v)

tree\_a <-color\_tree\_g %>%

ggraph(layout="partition") +

geom\_node\_tile(aes(filter=is\_rcolor, fill=color), color="#ffffff10", size=0.1) +

scale\_fill\_identity() +

theme\_graph() +

coord\_polar()

tree\_b <- color\_tree\_g %>%

ggraph(layout="treemap") +

geom\_edge\_fan(color="grey33", alpha=0.3) +

geom\_node\_point(aes(filter=is\_rcolor, color=color), size=3, shape=16) +

scale\_color\_identity() +

theme\_graph() +

coord\_fixed()

tree\_c <-color\_tree\_g %>%

ggraph(layout="tree", circular=T) +

geom\_edge\_fan(color="grey33", alpha=0.3) +

geom\_node\_point(aes(filter=is\_rcolor, color=color), size=3, shape=21) +

scale\_color\_identity() +

theme\_graph() +

coord\_fixed()

library(patchwork) ## makes it so easy to put ggplots together side by side

tree\_a + tree\_b + tree\_c

