[Tableau](https://www.tableau.com/) has chart type called “[Packed Bubble Chart](https://onlinehelp.tableau.com/current/pro/desktop/en-us/buildexamples_bubbles.htm)”, while I haven’t really utilized packed bubble chart much, I always thought they are fun and beautiful. I wanted to try creating same chart using R, and I came across package called Packcircles.

**Creating Bubble Packed Abstract Art…**

I didn’t really have data sets handy to use it for this type of chart, so I’ve decided to play around with image. Initially I wanted to create images used to [test colour blindness](https://www.colour-blindness.com/colour-blindness-tests/ishihara-colour-test-plates/), but I wasn’t sure what colour palettes should be used to create these images.

Since I currently love [Memoji](https://support.apple.com/en-ca/HT208986) on my iPhone, so I’ve decided I’d use Memoji image as base to create abstract art.

**Steps**

* Step 1. Import image and convert image to data frame, so you can extract colour value (RGB)
* Step 2. Genearate circle packing layout using circleProgressiveLayout function. The resulting data frame here contains center points of circle (x,y) and its radius.
* Step 3. Convert x & y coordinate from data frame in step 2 so that you can figure out what colour to fill the circle. i.e. Data Frame from Step 1 and Step 2 should be joined, so you need to adjust the scaling.
* Step 4. Create data frame using circleLayoutVertices function for plotting with ggplot2. The resulting data frame now have specified amount of points per indivisual circle so that you can use geom\_path or geom\_polygon to draw.
* Step 5. Join data from Step 4 with colour value from Step 3, so that you can use geom\_polygon with fill value to colour the circle!

library(tidyverse) ## I need tidyverse for everything :)

library(imager) ## to create data frame from image

library(scales) ## rescale function is so handy!

library(packcircles) ## making circle packing easy!

## Step 1

im <- load.image("https://farm5.staticflickr.com/4868/45503751845\_948f121563\_z.jpg") #memoji2

## if you want to take a look at image.. :)

#plot(im)

## Convert Image into Data Frame

im.df.colour <- im %>%

as.data.frame(wide="c") %>% ## so that rgb value is in separate column.

rename(im\_x=x,im\_y=y) %>%

mutate(hex=rgb(c.1,c.2,c.3))

## Step 2 using circleProgressiveLayout function.

## Generate circle packing layout using rbeta distribution as size of circles

pack\_layout <- circleProgressiveLayout(rbeta(2000,1,2), sizetype='area') %>%

## Step 3 - I want to figure out what colour to use, so I want layout & image df to have same scaling.

mutate(im\_x=floor(rescale(x,to=range(im.df.colour$im\_x))),

im\_y=floor(rescale(y,to=range(im.df.colour$im\_y))),

## also generate id, so i can join the data frame easily later!

id=row\_number()) %>%

inner\_join(im.df.colour %>% select(im\_x,im\_y,hex), by=c("im\_x","im\_y"))

## Step 4

## Using the layout above create data frame using circleLayoutVertices function so that you can plot circle using ggplot2

data\_gg <- circleLayoutVertices(pack\_layout) %>%

inner\_join(pack\_layout %>% select(id,hex), by=c("id"))

## Step 5

data\_gg %>%

ggplot(aes(x=x,y=y,group=id)) +

geom\_polygon(aes(fill=hex)) +

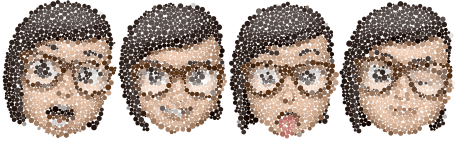
scale\_fill\_identity() +

coord\_equal() +

scale\_y\_reverse() + ## you need to reverse y-axis

theme\_void()





Just few more of these…

Good image to use is square image with above, but it’s fun turning into logo & images!!!

Below are just few more things I’ve experimented with circle packing technique…

**Experimenting with RGB Colour…**

## Generate layout from 500 uniformly distributed number 0 to 1 as area.

pack\_layout1 <- circleProgressiveLayout(runif(n=500), sizetype='area')

## I want to colour each circle with different rgb value, so I'll append data

pack\_layout1 <- pack\_layout1 %>%

mutate(hex\_r=rgb(1,rescale(x),rescale(y),rescale(radius)),

hex\_g=rgb(rescale(x),1,rescale(y),rescale(radius)),

hex\_b=rgb(rescale(x),rescale(y),1,rescale(radius)),

id = row\_number()) ## append id, so you can join this table later.

## pack\_layout1 contains data where center of circle should be placed with its radius.

## Now generate data so that you can actually draw circle using ggplot2

data\_gg1 <- circleLayoutVertices(pack\_layout1, npoints=25)

## notice now you have for each circle, you have 25 x and y coordinates to draw circle!

## Since the colour I want to use for each circle is retained in pack\_layout1 data frame, I want to combine the info. Also I want to create 3 sets of different colouring. I want to make long table.

data\_gg1 <- data\_gg1 %>% inner\_join(pack\_layout1 %>% select(-x,-y), by=c("id"))

## I want to create 3 different coloured variations, so convert above table to long format.

data\_gg\_long <- data\_gg1 %>%

gather(key="colour\_group",value="hex",hex\_r:hex\_b) %>%

mutate(colour\_group = factor(colour\_group,levels=c("hex\_r","hex\_g","hex\_b"),

labels=c("keeping red value constant\nmore green to the right - more blue to the top\nsmaller circle has more transparency",

"keeping green value constant\nmore red to the right - more blue to the top\nsmaller circle has more transparency",

"keeping blue value constant\nmore red to the right - more green to the top\nsmaller circle has more transparency")))

## Now the fun part!

data\_gg\_long %>%

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

geom\_polygon(aes(group=id),fill="#ffffff") + ## first draw all circle white.

geom\_polygon(aes(group=id, fill=hex)) + ## then colour with value with some transparency

coord\_equal() +

theme\_void() +

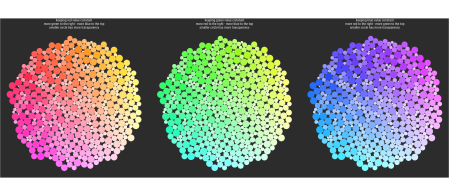
scale\_fill\_identity() +

scale\_y\_reverse() +

facet\_wrap(~colour\_group) +

theme(plot.background=element\_rect(fill="#000000de"),

strip.text=element\_text(family="Roboto Condensed", color="#ffffffde"))



**Drawing Smaller Circles Inside of Circles**

## Instead of using uniform distribution, used beta distribution this time!

pack\_layout2 <- circleProgressiveLayout(rbeta(1000,1,1), sizetype='area')

## This time I want to fill circle using hue value...

pack\_layout2 <- pack\_layout2 %>%

mutate(r = sqrt(x^2 + y^2), ## calculate distance from 0,0 coordinate

angle\_t = atan2(y,x), ## The arc-tangent of two arguments atan2(y, x) returns the angle between the x-axis and the vector from the origin to (x, y)

angle = rescale(angle\_t, from=c(-pi,pi)), ## convert theta value to value betwwen 0 and 1

hex = hsv(h=angle, s=rescale(r), v=0.8),

id = row\_number())

## use circleLayoutVertices function to generate data frame for ggplot2 & bring colour info.

data\_gg2 <- circleLayoutVertices(pack\_layout2,npoints=25) %>%

inner\_join(pack\_layout2 %>% select(-x,-y), by=c("id"))

## Now create data for inner circles!! But I'm sampling so that NOT all circle has inner circles! I want to pick more bigger circles than smaller circle, so using raidus as weight to sample.

data\_gg2\_1 <- circleLayoutVertices(pack\_layout2 %>%

sample\_n(800, weight=radius) %>%

mutate(radius=0.7\*radius), npoints=25) ## I want to draw smaller circle, so shrink the radius

data\_gg2\_2 <- circleLayoutVertices(pack\_layout2 %>%

sample\_n(700,weight=radius) %>%

mutate(radius=0.5\*radius),npoints=25)

data\_gg2\_3 <- circleLayoutVertices(pack\_layout2 %>%

sample\_n(900,weight=radius) %>%

mutate(radius=0.3\*radius),npoints=25)

## Draw Black and White Version

bw <-data\_gg2 %>%

ggplot(aes(x=x,y=y, group=id)) +

geom\_path(data=data\_gg2, size=0.5, color="#00000090") +

geom\_path(data=data\_gg2\_1,size=1, color="#00000090") +

geom\_path(data=data\_gg2\_2,size=0.5, color="#00000090") +

geom\_path(data=data\_gg2\_3,size=0.5, color="#00000090") +

scale\_fill\_identity() +

scale\_color\_identity() +

theme\_void() +

coord\_fixed()

## Draw colourful version

hue <-data\_gg2 %>%

ggplot(aes(x=x,y=y, group=id)) +

geom\_polygon(aes(fill=hex)) +

geom\_path(data=data\_gg2, size=0.5, color="#ffffff90") +

geom\_path(data=data\_gg2\_1,size=1, color="#ffffff90") +

geom\_path(data=data\_gg2\_2,size=0.5, color="#ffffff90") +

geom\_path(data=data\_gg2\_3,size=0.5, color="#ffffff90") +

scale\_fill\_identity() +

scale\_color\_identity() +

theme\_void() +

coord\_fixed()

library(patchwork)

bw + hue

