Creating and placing custom glyphs

Scott Spencer

2020 April 6

Contents

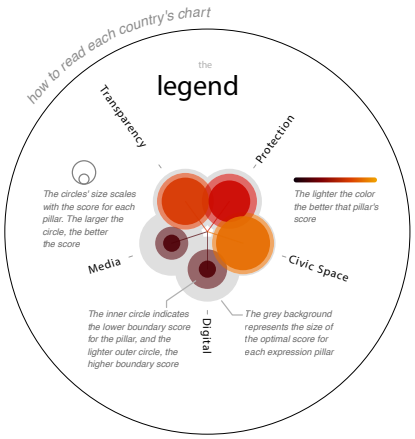
In this tutorial, we explore creating and placing custom glyphs by studying Nadieh Bremer's graphic for Article19: *The Freedom of Expression*.

1 Reference graphic

Before continuing, review this reference graphic on her website: https://www.visualcinnamon.com/portfolio/the-freedom-of-expression.

1.1 Reference individual glpyh encodings

In the graphic, each glyph represents a specific country and five measures related to the freedom of expression, civic, digital, media, protection, and transparency:



Each measure is encoded as a colored circle with their centers evenly spaced around a unit circle, *i.e.*, a circle with center at (0,0) and radius of 1 unit. As a circle's circumference spans 360 degrees or $2 \cdot \pi$, each measure would be spaced at multiples of $\frac{2 \cdot \pi}{5}$.

Each measure's data is scaled from 0 to 1. We find several layers in the glyph. The first, back, layer, is colored light gray and its size encoded as the ideal measure of expression. The next layer forward encodes the high value of that measure for the given country as both size and color along a gradient. The third layer encodes the low value of the same measure, again encoding the value as both size and color. These layers are grouped visually around a unit circle where, for each measure, a "stem" (line segment) spans from the origin of the unit circle (0,0) along the specified angle for that measure ending on the circle's circumference.

Finally, above the glyph, text encodes the name of the country.

1.2 Reference glyph placements

The complete graphic contains an above glyph for each country, over 160 in all, wherein their placement — left to right — encodes average freedom value relative to the other countries. Those with high average values are to the left, those with low values to the right. Along with axis, glyphs with similar values are spread vertically.

2 Tools

We'll be using the following packages:

```
library(knitr)      # create tables and knitting this document
library(kableExtra) # create tables
library(dplyr)      # organize data
library(ggplot2)    # create graphics
library(ggforce)    # ggplot2 extension to draw or encode data as circles
library(jsonlite)   # gather json data from website
```

3 Collecting the data

First, let's get the data, which comes from Varieties of Democracy (V-Dem): https://www.v-dem.net. We can gather this from Nadieh's website as a JSON file using the function `fromJSON` from the R package `jsonlite`.

```
rawdata <-
  jsonlite::fromJSON(paste0("https://nbremer.github.io/",
                            "article19/data/",
                            "article_19_country_data.min.json"))
```

The rawdata comes in as a list, and for teaching purposes, we'll just pull the 2016 data of the five measures into a dataframe:

```
d16 <- rawdata$xpa_data[[11]]
```

Here are the first few rows of the data:

country_name	country_id	civic	civic_low	civic_high	digital	digital_low	digital_high	media	media_low	media_high	protection	protection_low	protection_high	transparency	transparency_low	transparency_high	avg_value
Afghanistan	AFG	0.363	0.288	0.448	0.516	0.429	0.616	0.499	0.399	0.582	0.396	0.282	0.515	0.337	0.243	0.441	0.366
Albania	ALB	0.606	0.519	0.683	0.659	0.553	0.749	0.460	0.375	0.547	0.754	0.643	0.849	0.658	0.552	0.745	0.586
Algeria	DZA	0.189	0.120	0.223	0.202	0.149	0.274	0.241	0.175	0.309	0.292	0.200	0.408	0.347	0.265	0.457	0.258
Angola	AGO	0.367	0.294	0.444	0.568	0.469	0.662	0.520	0.425	0.604	0.467	0.353	0.570	0.217	0.147	0.287	0.369
Argentina	ARG	0.789	0.729	0.848	0.856	0.789	0.902	0.787	0.716	0.848	0.781	0.662	0.858	0.799	0.699	0.868	0.788

4 Reconstruction

Perhaps the most efficient approach is to first encode the individual glyph, and then *translate* or move each glyph to a particular place relative to the others. First we build the glyph.

4.1 Coding the glyph

4.1.1 Preparing the basic structure

We'll use R and a few of its packages, `ggplot2` for most of the graphic functions and `ggforce` for its function to create circles (`geom_circle()`).

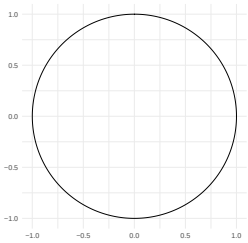
```
library(ggplot2); library(ggforce)
```

Let's start by setting up the general graphic and a coordinate system where the x and y axis are equally scaled. Eventually we will want to place each glyph at various x and y coordinates, so we'll add two variables to the data but, for now, we'll set them all to the origin.

```
d16$x <- 0
d16$y <- 0

d16_first <- d16[1,] # to test glyph, start with first observation

p <-
  ggplot(d16_first) +
  theme_minimal() +
  coord_equal() +
  labs(x = '', y = '')
```

Next, we just draw a unit circle with its center at the origin.

```
p +
  geom_circle(aes(x0 = x,
                  y0 = y,
                  r  = 1),
              color = "#000000")
```



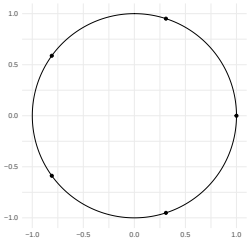
We will need five of these, one for each measure, and their centers should be set along the unit circle's circumference at equally-spaced angles. To make it easier to place each type, let's create a function to place it around the circle's circumference:

```
# create function to calculate the angle for each type
petal_loc <- function(i) {
  c(cos(2 * pi / 5 * i), sin(2 * pi / 5 * i))
}
```

```
xy_civic        <- petal_loc(1)
xy_digital      <- petal_loc(2)
xy_media        <- petal_loc(3)
xy_protection   <- petal_loc(4)
xy_transparency <- petal_loc(5)
```

Let's first check to see that our numbers are correct, by including a point for each measurement type around the unit circle we graphed above where the type's circle center will be located. Notice that for each point, we add the origin, or data, to the location around the circle circumference. Doing this makes the location of the glyph components relative to the data:

```
p +
  geom_circle(aes(x0 = x,
                  y0 = y,
                  r  = 1),
              color = "#000000") +
  geom_point(aes(x = x + xy_civic[1],        y = y + xy_civic[2]) ) +
  geom_point(aes(x = x + xy_digital[1],      y = y + xy_digital[2]) ) +
  geom_point(aes(x = x + xy_media[1],        y = y + xy_media[2]) ) +
  geom_point(aes(x = x + xy_protection[1],   y = y + xy_protection[2]) ) +
  geom_point(aes(x = x + xy_transparency[1], y = y + xy_transparency[2]) )
```

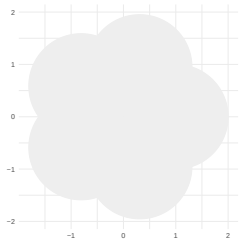


Each point will be where we place the circle centers for each measurement type. In our actual glyph, we will not draw either the unit circle or the five points. Next, we begin encoding our actual glyph, layer-by-layer, to the data.

4.1.2 First layer: gray petals encoding idea measures

So far so good. Now, we can arrange five circles (aka "petals") around the circle:

```
p <- p +
  geom_circle(aes(x0 = x + xy_civic[1],
                  y0 = y + xy_civic[2],
                  r  = 1),
              fill = "#eeeeee",
              color = "#eeeeee") +

  geom_circle(aes(x0 = x + xy_digital[1],
                  y0 = y + xy_digital[2],
                  r  = 1),
              fill = "#eeeeee",
              color = "#eeeeee") +
  geom_circle(aes(x0 = x + xy_media[1],
                  y0 = y + xy_media[2],
                  r  = 1),
              fill = "#eeeeee",
              color = "#eeeeee") +
  geom_circle(aes(x0 = x + xy_protection[1],
                  y0 = y + xy_protection[2],
                  r  = 1),
              fill = "#eeeeee",
              color = "#eeeeee") +
  geom_circle(aes(x0 = x + xy_transparency[1],
                  y0 = y + xy_transparency[2],
                  r  = 1),
              fill = "#eeeeee",
              color = "#eeeeee")
```



That finishes our first, back, layer for a single glyph. Next, let's layer on the "high" values for each type.

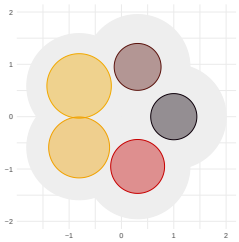
4.1.3 Second layer: colored and sized petals encoding high measures

Here, we'll start to encode actual data. Those encodings include data as color, for the petals, so we set up color and fill scales:

```
p <- p +
  scale_color_gradientn(colors = c("#0D000A", "#ca0000", "#ED8B00", "#F2A900"),
                        values = seq(0, 1, length.out = 4) ) +
  scale_fill_gradientn(colors = c("#0D000A", "#ca0000", "#ED8B00", "#F2A900"),
                       values = seq(0, 1, length.out = 4) ) +
  theme(legend.position = '')
```

Let's overlay the next layer, encoding each measure's average value for that country to color, and encoding the high value of the range to size of the circle's radius:

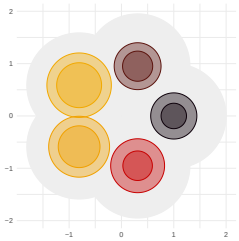
```
p <- p +
  geom_circle(aes(x0 = 0 + xy_civic[1],
                  y0 = 0 + xy_civic[2],
                  r  = civic_high,
                  color = civic,
                  fill = civic),
              size = 0,
              alpha = 0.4) +
  geom_circle(aes(x0 = 0 + xy_digital[1],
                  y0 = 0 + xy_digital[2],
                  r  = digital_high,
                  color = digital,
                  fill = digital),
              size = 0,
              alpha = 0.4) +
  geom_circle(aes(x0 = 0 + xy_media[1],
                  y0 = 0 + xy_media[2],
                  r  = media_high,
                  color = media,
                  fill = media),
              size = 0,
              alpha = 0.4) +
  geom_circle(aes(x0 = 0 + xy_protection[1],
                  y0 = 0 + xy_protection[2],
                  r  = protection_high,
                  color = protection,
                  fill = protection),
              size = 0,
              alpha = 0.4) +
  geom_circle(aes(x0 = 0 + xy_transparency[1],
                  y0 = 0 + xy_transparency[2],
                  r  = transparency_high,
                  color = transparency,
                  fill = transparency),
              size = 0,
              alpha = 0.4)
```



4.1.4 Third layer: colored and sized petals encoding low measures

Great, let's add the third, low value, layer, just setting the low values to, say, 0.6:

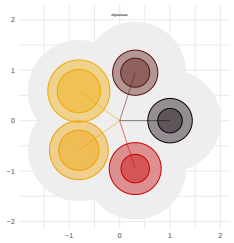
```
p <- p +
  geom_circle(aes(x0 = 0 + xy_civic[1],
                  y0 = 0 + xy_civic[2],
                  r  = civic_low,
                  color = civic,
                  fill = civic),
              size = 0,
              alpha = 0.4) +
  geom_circle(aes(x0 = 0 + xy_digital[1],
                  y0 = 0 + xy_digital[2],
                  r  = digital_low,
                  color = digital,
                  fill = digital),
              size = 0,
              alpha = 0.4) +
  geom_circle(aes(x0 = 0 + xy_media[1],
                  y0 = 0 + xy_media[2],
                  r  = media_low,
                  color = media,
                  fill = media),
              size = 0,
              alpha = 0.4) +
  geom_circle(aes(x0 = 0 + xy_protection[1],
                  y0 = 0 + xy_protection[2],
                  r  = protection_low,
                  color = protection,
                  fill = protection),
              size = 0,
              alpha = 0.4) +
  geom_circle(aes(x0 = 0 + xy_transparency[1],
                  y0 = 0 + xy_transparency[2],
                  r  = transparency_low,
                  color = transparency,
                  fill = transparency),
              size = 0,
              alpha = 0.4)
```



4.1.5 Fourth layer: adding stems and text label for country

We have two layers left, adding the stems and the country name. Let's do all five together with the name, so that will mean adding five line segments and a text label to the glyph. Following the reference, we'll encode average measure values as color, and use some transparency:

```
p <- p +
  geom_segment(aes(x = 0,
                   y = 0,
                   xend = 0 + xy_civic[1],
                   yend = 0 + xy_civic[2],
                   color = civic),
               size = 0.6,
               alpha = 0.5) +
  geom_segment(aes(x = 0,
                   y = 0,
                   xend = 0 + xy_digital[1],
                   yend = 0 + xy_digital[2],
                   color = digital),
               size = 0.6,
               alpha = 0.5) +
  geom_segment(aes(x = 0,
                   y = 0,
                   xend = 0 + xy_media[1],
                   yend = 0 + xy_media[2],
                   color = media),
               size = 0.6,
               alpha = 0.5) +
  geom_segment(aes(x = 0,
                   y = 0,
                   xend = 0 + xy_protection[1],
                   yend = 0 + xy_protection[2],
                   color = protection),
               size = 0.6,
               alpha = 0.5) +
  geom_segment(aes(x = 0,
                   y = 0,
                   xend = 0 + xy_transparency[1],
                   yend = 0 + xy_transparency[2],
                   color = transparency),
               size = 0.6,
               alpha = 0.5) +
  geom_text(aes(x=x, y = y + 2.1, label = country_name), size = 4/.pt)
```



4.2 Placing glyphs on the plot relative to each other

With our glyph setup and mapped to data, we now need to change the default x and y locations from (0, 0) to something meaningful. Looking back to the reference graphic, we

see that the glyphs are arranged left to right from high average values to low average values. So we need to assign each glyph to a place based on its average value relative to the others.

We can set this up by arranging our observations in order of average value. There are 160 observations (countries). We can arrange 160 values as a matrix or grid with a size of, say, 20 by 8. Now, doing this would mean the result is completely symmetrical whereas the reference grid has more variation. So let's try moving each grid location by a small random amount to create a similar idea. Each location on the 20 x 8 grid needs to be a good bit wider than the glyph itself. So along each axis we can just setup a sequence of numbers and multiply them to add space. The following code arranges the countries by average value `avg_value`, then assigns its x and y values like so,

```
d16 <- d16 %>%
  arrange((avg_value)) %>%
  mutate(y = rep(seq(20), each = 8) * 6 + rnorm(160, 0, 0.8),
         x = rep(c(0, 1, -1, 2, -2, 3, -3, 4), 20) * 6 + rnorm(160, 0, 0.8))
```

To use the new `x` and `y`, we just place the full dataframe `d16` into the above code. Altogether,

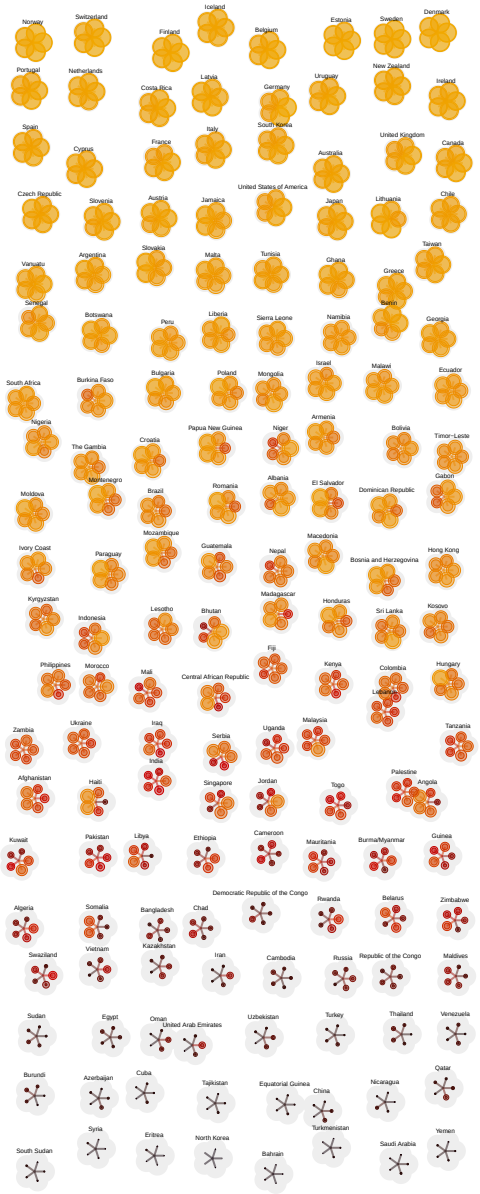
```
p <-

  # base graphic
  ggplot(d16) +
  theme_void() +
  coord_equal() +
  labs(x = '', y = '') +

  # scales
  scale_color_gradientn(colors = c("#0D000A", "#ca0000", "#ED8B00", "#F2A900"),
                        values = seq(0, 1, length.out = 4) ) +
  scale_fill_gradientn(colors = c("#0D000A", "#ca0000", "#ED8B00", "#F2A900"),
                       values = seq(0, 1, length.out = 4) ) +
  theme(legend.position = '') +


  # add gray layer
    geom_circle(aes(x0 = x + xy_civic[1],
                    y0 = y + xy_civic[2],
                    r = 1),
                fill = "#eeeeee",
                color = "#eeeeee") +

  geom_circle(aes(x0 = x + xy_digital[1],
                  y0 = y + xy_digital[2],
                  r = 1),
              fill = "#eeeeee",
              color = "#eeeeee") +
  geom_circle(aes(x0 = x + xy_media[1],
                  y0 = y + xy_media[2],
                  r = 1),
              fill = "#eeeeee",
              color = "#eeeeee") +
  geom_circle(aes(x0 = x + xy_protection[1],
                  y0 = y + xy_protection[2],
                  r = 1),
              fill = "#eeeeee",
              color = "#eeeeee") +
  geom_circle(aes(x0 = x + xy_transparency[1],
                  y0 = y + xy_transparency[2],
                  r = 1),
              fill = "#eeeeee",
```

```
            color = "#eeeeee") +


# add high value layer
  geom_circle(aes(x0 = x + xy_civic[1],
                  y0 = y + xy_civic[2],
                  r  = civic_high,
                color = civic,
                fill = civic),
            size = 0,
            alpha = 0.4) +
geom_circle(aes(x0 = x + xy_digital[1],
                y0 = y + xy_digital[2],
                r  = digital_high,
                color = digital,
                fill = digital),
            size = 0,
            alpha = 0.4) +
geom_circle(aes(x0 = x + xy_media[1],
                y0 = y + xy_media[2],
                r  = media_high,
                color = media,
                fill = media),
            size = 0,
            alpha = 0.4) +
geom_circle(aes(x0 = x + xy_protection[1],
                y0 = y + xy_protection[2],
                r  = protection_high,
                color = protection,
                fill = protection),
            size = 0,
            alpha = 0.4) +
geom_circle(aes(x0 = x + xy_transparency[1],
                y0 = y + xy_transparency[2],
                r  = transparency_high,
                color = transparency,
                fill = transparency),
            size = 0,
            alpha = 0.4) +

# add low value layer
  geom_circle(aes(x0 = x + xy_civic[1],
                  y0 = y + xy_civic[2],
                  r  = civic_low,
                color = civic,
                fill = civic),
            size = 0,
            alpha = 0.4) +
geom_circle(aes(x0 = x + xy_digital[1],
                y0 = y + xy_digital[2],
                r  = digital_low,
                color = digital,
                fill = digital),
            size = 0,
            alpha = 0.4) +
geom_circle(aes(x0 = x + xy_media[1],
                y0 = y + xy_media[2],
                r  = media_low,
                color = media,
                fill = media),
            size = 0,
            alpha = 0.4) +
```

```
geom_circle(aes(x0 = x + xy_protection[1],
                y0 = y + xy_protection[2],
                r  = protection_low,
                color = protection,
                fill = protection),
            size = 0,
            alpha = 0.4) +
geom_circle(aes(x0 = x + xy_transparency[1],
                y0 = y + xy_transparency[2],
                r  = transparency_low,
                color = transparency,
                fill = transparency),
            size = 0,
            alpha = 0.4) +

# add stems and text labeels
  geom_segment(aes(x = x,
                   y = y,
                   xend = x + xy_civic[1],
                   yend = y + xy_civic[2],
                  color = civic),
             size = 0.6,
             alpha = 0.5) +
geom_segment(aes(x = x,
                 y = y,
                 xend = x + xy_digital[1],
                 yend = y + xy_digital[2],
                 color = digital),
             size = 0.6,
             alpha = 0.5) +
geom_segment(aes(x = x,
                 y = y,
                 xend = x + xy_media[1],
                 yend = y + xy_media[2],
                 color = media),
             size = 0.6,
             alpha = 0.5) +
geom_segment(aes(x = x,
                 y = y,
                 xend = x + xy_protection[1],
                 yend = y + xy_protection[2],
                 color = protection),
             size = 0.6,
             alpha = 0.5) +
geom_segment(aes(x = x,
                 y = y,
                 xend = x + xy_transparency[1],
                 yend = y + xy_transparency[2],
                 color = transparency),
             size = 0.6,
             alpha = 0.5) +
geom_text(aes(x = x,
              y = y + 2.1,
              label = country_name),
          size = 4/.pt)
```

And here are our finished encodings:

Study of custom glyph construction and placement in reference graphic



Reference design by Nadieh Bremer.
Data source: v−Dem.