HW #1

Interpreting {ggplot2} code

Emily Miller

Table of contents

Some notes before you get started

- **Be sure to install any packages** in the Setup chunk that you don't already have.
- **Leave the code chunk options, `eval: false` and `echo: true`, set as they are.** The final infographic has been intentionally optimized (e.g., text size, spacing) for saving and viewing as a PNG file, not for display in the Plots pane or within a rendered Quarto document. As a result, the text in each individual ggplot may appear too large when viewed in the Plots pane, but will be correctly sized in the exported PNG. We'll talk more about the nuances of saving ggplots (and why these differences occur) in a later lab section.
- Some answers may become clearer once you've looked ahead at the code further

down in the script. **Consider revisiting questions as you go.**

I. Setup

```
1  library(colorspace)
2  library(geofacet)
3  library(ggtext)
4  library(glue)
5  library(grid)
6  library(magick)
7  library(patchwork)
8  library(scales)
9  library(showtext)
10 library(tidyverse)
11
12 # Load ufo data
13 ufo_sightings <- read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/mai
14 places <- read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/main/data/
15
16 # Define background and accent alient color palette
17 alien <- c("#101319", "#28ee85")
18 bg <- alien[1]
19 accent <- alien[2]
20
21 # Load in ufo image
22 ufo_image <- magick::image_read(path = here::here("images", "ufo.png"))
23
24 # Download fonts for use in figures/graphics
25 sysfonts::font_add_google(name = "Orbitron", family = "orb")
26 sysfonts::font_add_google(name = "Barlow", family = "bar")
27
28 sysfonts::font_add(family = "fa-brands", regular = here::here("fonts", "Font Awesome 6 Brands
29 sysfonts::font_add(family = "fa-solid", regular = here::here("fonts", "Font Awesome 6 Free-S
30
31 showtext::showtext_auto(enable = TRUE)
```

1. **What is the author defining in lines 15-17? Where else in the code do these defined variables show up? What advantage(s) is there to defining these values here, as variables, rather than defining the values directly throughout the script?**

The author seems to be defining an alien color palette and then specifying black as the background color and green as the accent. These variables show up throughout the code in theme elements, as fill colors, and text colors. The advantage of defining them as text variables is that if you ever want to change the colors in the palette, you need only change one line of code rather than change every instance. It also helps with readability, `fill = accent` is easier to understand than `fill = "#28ee85"`.

2. **In your own words, explain what the function, `font_add_google()`, does. What's the difference between the two arguments, `name` and `family`?**

   This function downloads Google Fonts files from a particular family for use in rendered graphics like this one. The `name` argument is the exact name of the font as it appears on Google Fonts whereas `family` is how the font will be referenced within the code.

II. Data wrangling

i. Create `df_pop`

```
df_pop <- places |>
  # Filter places to those in the US
  filter(country_code == "US") |>
  # Capitalize Florida's abbreviation
  mutate(state = str_replace(string = state,
                             pattern = "Fl",
                             replacement = "FL")) |>
  # Make a pop column containing the population of each state
  group_by(state) |>
  summarise(pop = sum(population)) |>
  ungroup()
```

3. **Describe what this data frame contains.**

   df_pop contains the total population for each US state.

ii. Create `df_us`

```
df_us <- ufo_sightings |>
  # Filter sightings to those in the US
  filter(country_code == "US") |>
  # Capitalize Florida's abbreviation
```

```
  mutate(state = str_replace(string = state,
                             pattern = "Fl",
                             replacement = "FL")) |>
  # Create a count column, n, containing the number of sightings per state
  count(state) |>
  # Join with df_pop to add population data for each state
  left_join(df_pop, by = "state") |>
  # Rename the count column
  rename(num_obs = n) |>
  mutate(
    # Calcuate the number of sightings per 10k people for each state
    num_obs_per10k = num_obs / pop * 10000,
    # Create a opacity value normalized between 0 and 1
    opacity_val = num_obs_per10k / max(num_obs_per10k)
    )
```

4. **Describe what this data frame contains.**

   df_us contains UFO sighting statistics for each US state including the number of observations, the total population of the state, the number of sightings per 10k people, and an opacity value to help visualize this number.

5. **What does `opacity_val` represent, and why is it calculated?**

   It represents the relative intensity of UFO sightings for each state, normalized to a 0-1 scale by dividing each state's per-10k sighting rate by the maximum per-10k sighting rate. This value can be mapped to a color scale or alpha value in ggplot so that states with more sightings per 10k people will appear colored or more opaque.

iii. Create `df_shape`

```
df_shape <- ufo_sightings |>
  # Filter out sightings with unknown or other shapes
  filter(!shape %in% c("unknown", "other")) |>
  # Count how many sightings occurred for each unique shape value
  count(shape) |>
  # Rename this count column
  rename(total_sightings = n) |>
  # Sort the data frame in descending order by total_sightings
  arrange(desc(total_sightings)) |>
  # Keep only the top 10 most commonly reported shapes
  slice_head(n = 10) |>
```

```
  mutate(
    # Convert shape from character to factor, with factor levels ordered by total_sightings
    shape = fct_reorder(.f = shape, # Variable to reorder
                        .x = total_sightings), # values to order by
    # create an opacity value by rescaling total_sightings to range 0.3-1
    opacity_val = scales::rescale(x = total_sightings,
                                  to = c(0.3, 1))
    )
```

6. **Describe what this data frame contains.**

   df_shape contains the 10 most commonly reported UFO shapes (excluding "unknown" and "other"), with the total number of sightings for each shape, ordered from most to least common, and an opacity value for visualization.

7. **What does `fct_reorder` do when it is applied to the `shape` variable? What would have happened if this step was not performed?**

   It reorders the values of `shape` as a factor based on the values of `total_sightings`. Without this, the shapes would appear in alphabetical order which would make visualizations harder to interpret.

8. **What is the purpose of rescaling `opacity_val`? And why rescale from 0.3 to 1?**

   It helps us visualize the values in context by being mappable to an alpha aesthetic. This opacity value is used in visualizations to set transparency so the lower bound of 0.3 ensures even the least common shape is still visible (not entirely transparent).

iv. Create `df_day_hour`

```
df_day_hour <- ufo_sightings |>
  # Extract date-time components from the reported_date_time column
  mutate(
    day = wday(reported_date_time),
    hour = hour(reported_date_time),
    wday = wday(reported_date_time, label = TRUE)
  ) |>
  # Count the number of sightings for each unique combination of day and hour
  count(day, wday, hour) |>
  # Rename this count column
  rename(total_daily_obs = n) |>
  mutate(
```

```
    # Calculate an opacity value
    opacity_val = total_daily_obs / max(total_daily_obs),
    # Create formatted hour labels
    hour_lab = case_when(
      hour == 0 ~ "12am",
      hour <= 12 ~ paste0(hour, "am"),
      hour == 12 ~ "12pm",
      TRUE ~ paste0(hour - 12, "pm"))
    )
```

9. **Describe what this data frame contains.**

   It contains the count of UFO sightings for each combination of day of the week and hours of the day. It contains both numberic and string versions of the weekday, an opacity value based on frequency, and formatted hour labels.

10. **What is the purpose of the last line inside the `case_when()` statement (`TRUE ~ paste0(hour - 12, "pm")`)?**

    Its a default/catchall condition that will handle military time.

III. Prepare text elements

```
# Extract specific UFO sightings from the dataset
quotes <- paste0('"...', str_to_sentence(ufo_sightings$summary[c(47816, 6795, 93833)]), '...'

# Create text string for the original author credit
original <- glue("Original visualization by Dan Oehm:")

# Create github link with Font Awesome Github icon
dan_github <- glue("<span style='font-family:fa-brands;'>&#xf09b;</span> doehm/tidytues")

# Create text string for the updated version credit
new <- glue("Updated version by Sam Shanny-Csik for EDS 240:")

# Create website link with Font Awesome
link <- glue("<span style='font-family:fa-solid;'>&#xf0c1;</span> eds-240-data-viz.github.io'

# Create invisible spacing using the background color
space <- glue("<span style='color:{bg};'>. .</span>")

# Combine all elements to create the caption
```

```
caption <- glue("{original}{space}{dan_github}
                 <br><br>
                 {new}{space}{link}")
```

11. **In your own words, what is the difference between `paste0()` and `glue()`? Why did the author use `paste0` to construct `quotes` and `glue` to construct the other text elements?**

    They both concatenate strings but `glue()` allows for inline R code evaluation within curly braces and variable substitution. The author used `paste0()` for simple quotes with not complex string formatting requirements, but used `glue()` to include html formatting and include variable substitutions.

IV. Build plots

i. Build `plot_shape`

```
plot_shape <- ggplot(data = df_shape) +
  # Create visualization showing sighting count for each shape
  geom_col(aes(x = total_sightings, y = shape, alpha = opacity_val),
           fill = accent) +
  geom_text(aes(x = 200, y = shape, label = str_to_title(shape)),
            family = "orb",
            fontface = "bold",
            color = bg,
            size = 14,
            hjust = 0,
            nudge_y = 0.2) +
  geom_text(aes(x = total_sightings-200, y = shape, label = scales::comma(total_sightings)),
            family = "orb",
            fontface = "bold",
            color = bg,
            size = 10,
            hjust = 1,
            nudge_y = -0.2) +
  scale_x_continuous(expand = c(0, NA)) +
  labs(subtitle = "10 most commonly reported shapes") +
  theme_void() +
  theme(
    plot.subtitle = element_text(family = "bar",
                                 size = 40,
```

```
                                    color = accent,
                                    hjust = 0,
                                    margin = margin(b = 10)),
    legend.position = "none"
  )
```

12. **Explain the values provided to the x aesthetic for both text geoms (`shape` & `total_sightings`).**

    In the first one, the text is placed 200 units from the left edge, the second positions the text 200 units from the right edge.

ii. Build `plot_us`

HINT: Consider temporarily commenting out / rearranging the `geom_*()` layers to better understand how this plot is constructed

```
plot_us <-  ggplot(df_us) +
  # Create filled rectangles for each state, opacity varies by sighting rate
  geom_rect(aes(xmin = 0, xmax = 1, ymin = 0, ymax = 1, alpha = opacity_val),
            fill = accent) +
  # Add state abbreviation labels in upper portion of each rectangle
  geom_text(aes(x = 0.5, y = 0.7, label = state),
            family = "orb",
            fontface = "bold",
            size = 9,
            color = bg) +
  # Add per-capita sighting numbers in lower portion of each rectangle
  geom_text(aes(x = 0.5, y = 0.3, label = round(num_obs_per10k, 1)),
            family = "orb",
            fontface = "bold",
            size = 8,
            color = bg) +
  # Arrange rectangles in geographic grid matching US state positions
  geofacet::facet_geo(~state) +
  # Maintain square aspect ratio for each state panel
  coord_fixed(ratio = 1) +
  labs(subtitle = "Sightings per 10k population") +
  theme_void() +
  theme(
    strip.text = element_blank(), # Remove default facet labels (using custom text instead)
    plot.subtitle = element_text(family = "bar",
```

```
                                    size = 40,
                                    color = accent,
                                    hjust = 1, # Right-align subtitle
                                    margin = margin(b = 10)),
    legend.position = "none"
  )
```

13. **Consider the order of `geom_*()` layers in the the above plot (`plot_us`). Why did the author order the layers in this way?**

    The layers show up back to front, so the background rectangles have to go first to form the base, then text layers are added on top so that they are not hidden by te background.

iii. Build `plot_day`

```
plot_day <- ggplot(data = df_day_hour) +
  # Create colored tiles for each hour/day combo; opacity = sighting frequency
  geom_tile(aes(x = hour, y = day, alpha = opacity_val),
            fill = accent,
            height = 0.9,  # Slight gaps between tiles
            width = 0.9) +
  # Add hour labels around outer edge at y=9
  geom_text(aes(x = hour, y = 9, label = hour_lab),
            family = "orb",
            color = accent,
            size = 10) +
  # Add single-letter day labels at center x=0
  geom_text(aes(x = 0, y = day, label = str_sub(string = wday, start = 1, end = 1)),
            family = "orb",
            fontface = "bold",
            color = bg,
            size = 8) +
  # Extend y-axis down to -5 for spacing, up to 9 for hour labels
  ylim(-5, 9) +
  # Extend x slightly past 23 to create gap in circular plot
  xlim(NA, 23.55) +
  # Convert to circular/clock layout
  coord_polar() +
  theme_void() +
  theme(
    plot.background = element_rect(fill = bg, color = bg),
```

```
    legend.position = "none"
  )
```

14. **This plot includes one-letter labels for each day of the week. How is ths accomplished when week days are written using their three-letter abbreviations (e.g. `Mon`, `Tue`) in the `df_day_hour` data frame?**

    The 'str_sub(string = wday, start = 1, end = 1) extracts only the first letter from each three letter weekday abbreviation.

15. **What role do the `ylim()` and `xlim()` functions play in shaping a ggplot, and how do they change the visual layout of this particular plot? To better understand their effect, try rerunning the code with each of these lines commented out and observe how the plot's spacing and composition change.**

    They control the x and y axis coordinate ranges, for example, `ylim(-5, 9)` creates extra spece below and above the data which helps with spacing in polor coordinates.

iv. Build `quote*s`

A comment from Dan Oehm's original code: "A bit clunky but the path of least resistance."

```
quote1 <- ggplot() +
  annotate(geom ="text",
           x = 0,
           y = 1,
           label = str_wrap(string = quotes[1], width = 40),
           family = "bar",
           fontface = "italic",
           color = accent,
           size = 16,
           hjust = 0,
           lineheight = 0.4) +
  xlim(0, 1) +
  ylim(0, 1) +
  theme_void() +
  coord_cartesian(clip = "off")

quote2 <- ggplot() +
  annotate(geom = "text",
           x = 0,
           y = 1,
           label = str_wrap(string = quotes[2], width = 25),
```

```
            family = "bar",
            fontface = "italic",
            color = accent,
            size = 16,
            hjust = 0,
            lineheight = 0.4) +
  xlim(0, 1) +
  ylim(0, 1) +
  theme_void() +
  coord_cartesian(clip = "off")

quote3 <- ggplot() +
  annotate(geom = "text",
            x = 0,
            y = 1,
            label = str_wrap(string = quotes[3], width = 25),
            family = "bar",
            fontface = "italic",
            color = accent,
            size = 16,
            hjust = 0,
            lineheight = 0.4) +
  xlim(0, 1) +
  ylim(0, 1) +
  theme_void() +
  coord_cartesian(clip = "off")
```

16. **Why do you think the author chose to convert these text elements (and also in `plot_ufo`, below!) into ggplot objects (you may consider returning to this question after you've worked your way through all of the code)?**

    It allows them to be formatted and layered using patchworks functions, particularly `inset_element()`. All the components need to be ggplot objects to be combined together easily.

v. Build `plot_ufo`

Note: Grob stands for **gr**aphical **ob**ject. Each visual element rendered in a a ggplot (e.g. lines, points, axes, entire panels, even images) is represented as a grob. Grobs can be manipulated individually to fully customize plots.

```r
plot_ufo <- ggplot() +
  annotation_custom(grid::rasterGrob(ufo_image)) +
  theme_void() +
  theme(
    plot.background = element_rect(fill = bg, color = bg)
  )
```

vi. Build `plot_base`

```r
plot_base <- ggplot() +
  labs(
    title = "UFO Sightings",
    subtitle = "Summary of over 88k reported sightings across the US",
    caption = caption
    ) +
  theme_void() +
  theme(
    text = element_text(family = "orb",
                        size = 48,
                        lineheight = 0.3,
                        color = accent),
    plot.background = element_rect(fill = bg,
                                   color = bg),
    plot.title = element_text(size = 128,
                              face = "bold",
                              hjust = 0.5,
                              margin = margin(b = 10)),
    plot.subtitle = element_text(family = "bar",
                                 hjust = 0.5,
                                 margin = margin(b = 20)),
    plot.caption = ggtext::element_markdown(family = "bar",
                                            face = "italic",
                                            color = colorspace::darken(accent, 0.25),
                                            hjust = 0.5,
                                            margin = margin(t = 20)),
    plot.margin = margin(b = 20, t = 50, r = 50, l = 50)
  )
```

17. **Why does the author render `plot.caption` using `ggtext::element_markdown()`, rather than `element_text()` (like he does for rendering `plot.title` and `text`)?**

It interprets HTML/markdown syntax in the text, which is necessary for the icon tags and line breaks. `element_text()` would displace this as literal text.

V. Assemble & save

```
plot_final <- plot_base +
  inset_element(plot_shape, left = 0, right = 1, top = 1, bottom = 0.66) +
  inset_element(plot_us, left = 0.42, right = 1, top = 0.74, bottom = 0.33) +
  inset_element(plot_day, left = 0, right = 0.66, top = 0.4, bottom = 0) +
  inset_element(quote1, left = 0.5, right = 1, top = 0.8, bottom = 0.72) +
  inset_element(quote2, left = 0, right = 1, top = 0.52, bottom = 0.4) +
  inset_element(quote3, left = 0.7, right = 1, top = 0.2, bottom = 0) +
  inset_element(plot_ufo, left = 0.25, right = 0.41, top = 0.23, bottom = 0.17) +
  plot_annotation(
    theme = theme(
      plot.background = element_rect(fill = bg,
                                     color = bg)
    )
  )

ggsave(plot = plot_final,
       filename = here::here("outputs", "ufo_sightings_infographic.png"),
       height = 16,
       width = 10)
```

18. **Explain how `plot_final` is assembled. What do you think is the most challenging aspect of arranging all components into a single plot?**

    Its assembled starting with plot_base which has the title/subtitle/caption and then uses inset_element() to position each element using a normalized coordinate system ranging from 0 to 1. I would think this might be the hardest part, figuring out the exact corridnate values to align and position every component.

19. **Can you think of one reason the author may have chosen to separate the construction of `plot_base` and `plot_final`?**

    Its like saving progress id think, its the canvas, and having this separate makes it so that things like the background color, title, margins etc. are established and constant before the layout its put on.

Answer some final reflective questions

20. **During week 2, we discuss Choosing the right graphic form. Refer to this lecture when answering the sub-questions, below:**

    a. **What "perceptual tasks" (from Cleveland & McGill's heirarchy) must the viewer perform to extract information from these visualizations?**

    The shapes plot requires 'position along a common scale', and the us map, shapes, and time of day visualizations require 'color saturation/opacity', though the last requires understanding position as well.

    b. **What task(s) do you think the author wanted to enable or message(s) he wanted to convey with these visualizations (see lecture 2.1, slide 16 for examples)? Be sure to note at least one task / message for each of the three data viz.**

    For the shape frequency plot, I think he was trying to enable a comparison/ranking task so it would be easy to see that light is the most commonly reported UFO shape by quite a lot, followed by cirlce and triangle. For the us plot I think the author was trying to enable shading and saturation task along with positionality, what stands out to me here is that the more rural states seem to have the higher values. The third graphic is again enabling a shading and saturation task along with positionality, but is also enabling position along common scales. The message highlighted in this visualization seems to be that sightings are most common early morning Sunday (or perhaps after a late night partying on Saterday).

    c. **Name at least one caveat to the "hierarchy of perceptual tasks" that the author employed to achieve a goal(s) you noted in question b?**

    I think that the third representation is very clear though saturation and non-standard cartesian coordinates are not listed as being very accurate, the polar coordinates in this case make the visual take the form of a clock, which viewers are likely quite familiar with interpreting, and it is easy to see the temporal patterns without needing exact values.

21. **Describe two elements of this piece that you find visually-pleasing / easy to understand / intuitive. Why?**

    As mentioned above, I think that the 'clock-face' representation of the time data is really nice. Overall, I like the how the lower values are more transparent, especially in the shape visualization, it feels like the less commonly reported shapes are fading out and it directs the eye to the next visualization.

22. **Describe two elements of this piece that you feel could be better presented in a different way. Why?**

    It took me a second to understand that the second visualization was a map of the US. I feel like I want to see this on a real map, not as representative squares in approximate locations. I am also not sure what the quotes are doing for the visualization, I think they are funny, but their placement feels odd and I would expect them to add something to the message or information presented in the graphic but they don't really.

23. **Describe two new things that you learned by interpreting / annotating this code. These could be packages, functions, or even code organizational approaches that you hadn't previously known about or considered.**

    I learned about building plots modularly, particularly `patchwork::inset_element()` seems like a very useful tool. Also, though I didn't think it added much clarity here, geofacet seems like a very cool tool and I would like to see what other ways it can be used.

24. **How, if at all, did you use AI tools to help you interpret this code? Describe your approach to using these tools for this assignment. In what ways was consulting the documentation more (or less) helpful than using AI?**

    I used AI to generate some of the documentations for the data cleaning code I was familiar with. But other than that I did not use it to interpret the code. I find that if I paste a line of code into and LLM and ask what it does I learn only about the use of any packages/functions in one particular instance whereas when I look up the documentation I get more of a big picture idea of what the package/function can do, its different arguements/parameters, etc. many of which may not be present in the code I am looking at.