HW #1

Interpreting `{ggplot2}` code

Megan Hessel

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.
- **Questions that reference line numbers (e.g., Question #1) refer to the**

line numbers shown in the rendered code chunks. You will need to render the document to view these line numbers.
- 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(ggtext)
5  library(grid)
6  library(magick)
7  library(patchwork)
8  library(scales)
9  library(showtext)
10 library(tidyverse)
11 library(glue)
12
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 alien <- c("#101319", "#28ee85")
17 bg <- alien[1]
18 accent <- alien[2]
19
20 ufo_image <- magick::image_read(path = here::here("images", "ufo.png"))
21
22 sysfonts::font_add_google(name = "Orbitron", family = "orb")
23 sysfonts::font_add_google(name = "Barlow", family = "bar")
24
25 sysfonts::font_add(family = "fa-brands", regular = here::here("fonts", "Font Awesome 6 Brands
26 sysfonts::font_add(family = "fa-solid", regular = here::here("fonts", "Font Awesome 6 Free-S
27
28 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?**

- *Alien* is defined as two colors which are broken down further: alien color 1 as *bg* and color 2 as *accent*. In later ggplot color parameters, these labels show up again. Defining them here is efficient. When graphing, it is easier to reference a pre-made variable label then re-writting the HEX code over and over again. Also, if the author wants to change the color later, they will only have to change it in one place.

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

    - `font_add_google` loads pre-made Google fonts into R. The `name` argument is the font name used in Google Fonts. The `family` argument is the name the author will reference later within `geom_text`.

II. Data wrangling

i. Create `df_pop`

```
df_pop <- places |>
  filter(country_code == "US") |>
  mutate(state = str_replace(string = state,
                             pattern = "Fl",
                             replacement = "FL")) |>
  group_by(state) |>
  summarise(pop = sum(population)) |>
  ungroup()
```

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

    - The `df_pop` dataframe contains 2 columns. The first column specifies each state's postal code (all capitals). The second column is each state's total population.

ii. Create `df_us`

```
df_us <- ufo_sightings |>
  filter(country_code == "US") |>
  mutate(state = str_replace(string = state,
                             pattern = "Fl",
                             replacement = "FL")) |>
  count(state) |>
  left_join(df_pop, by = "state") |>
```

```
  rename(num_obs = n) |>
  mutate(
    num_obs_per10k = num_obs / pop * 10000,
    opacity_val = num_obs_per10k / max(num_obs_per10k)
    )
```

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

    - The `df_us` dataframe has each state's total population and UFO sighting information such as total number of sighting, number of sightings per every 10k people, and the sightings in comparison to other states.

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

    - `opacity_val` is a proportion of number of sightings (per every 10k people) for each state comparatively to the highest number of sightings (per every 10k people). This is calculated to have an easy state by state comparison. Just by `opacity_val`, it is easily observe Montanata has the most number of sightings (per every 10k people), followed by VT, WA, WV, ME, and NH. This also sets up opacity levels (used for alpha values) representing number of sightings (per every 10k people) relative to the maximum for each state.

iii. Create `df_shape`

```
df_shape <-
ufo_sightings |>
  filter(!shape %in% c("unknown", "other")) |>
  count(shape) |>
  rename(total_sightings = n) |>
  arrange(desc(total_sightings)) |>
  slice_head(n = 10) |>
  mutate(
    shape = fct_reorder(.f = shape,
                        .x = total_sightings),
    opacity_val = scales::rescale(x = total_sightings,
                                  to = c(0.3, 1))
    )
```

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

- The `df_shape` dataframe contains the different alien sighting shapes. These shapes have the corresponding total number of sightings and the proportion of the shape's total number of sighting compared to the shape with the maximum observations.

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

   - The `fct_reorder` sorts the shape column by the number of total sightings. If this step was not performed, the dataframe would not be organized in descending order of total sighting.

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

   - The `rescale` function rescales the continous variables from total_sightings between 0.3 and 1. The author choose 0.3 to 1 because they will use it later for alpha values.

iv. Create `df_day_hour`

```
df_day_hour <- ufo_sightings |>
  mutate(
    day = wday(reported_date_time),
    hour = hour(reported_date_time),
    wday = wday(reported_date_time, label = TRUE)
  ) |>
  count(day, wday, hour) |>
  rename(total_daily_obs = n) |>
  mutate(
    opacity_val = total_daily_obs / max(total_daily_obs),
    hour_lab = case_when(
      hour == 0 ~ "12am",
      hour <= 12 ~ paste0(hour, "am"),
      hour == 12 ~ "12pm",
      TRUE ~ paste0(hour - 12, "pm"))
    )

colnames(df_day_hour)
```

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

   - For every day of UFO sightning, there is sighting observations (and the proportional opacity_val information) for every hour.

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

    - The `hour` column shows time ranging from 1-24. The `case_when` statement converts the 1-24 hour column to the typical 1-12 am and pm time slots by saying "if < 12 then its 'am' and if its > 12 its 'pm' ". The `TRUE ~ paste0(hour - 12, "pm"` says "if it is anything else (aka anything greater than 12), take the hour column value (which should be between 13-24) minus it by 12 and add 'pm' ".

III. Prepare text elements

```
quotes <- paste0('"...', str_to_sentence(ufo_sightings$summary[c(47816, 6795, 93833)]), '...'

original <- glue("Original visualization by Dan Oehm:")
dan_github <- glue("<span style='font-family:fa-brands;'>&#xf09b;</span> doehm/tidytues")
new <- glue("Updated version by Sam Shanny-Csik for EDS 240:")
link <- glue("<span style='font-family:fa-solid;'>&#xf0c1;</span> eds-240-data-viz.github.io'
space <- glue("<span style='color:{bg};'>. .</span>")
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?**

    - `paste0()` allows for the mechanical, repetitive structure within this code chunk as it creates 3 different strings of texts. `glue()` is useful because its easy to read and edit. Also, `glue()` can add formatting with HTML tags and embedded variables.

IV. Build plots

i. Build `plot_shape`

```
plot_shape <- ggplot(data = df_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 `geom_text`, the x is 200 which describes the placement of the shape labels on the graph's y axis (200 units from the x-axis origin). In the second `geom_text`, the x is `total_sightings` - 200 which descibes the placement of the shape's total observations on the graph's y axis (200 units away from the end of the bar graph).

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) +
  geom_rect(aes(xmin = 0, xmax = 1, ymin = 0, ymax = 1, alpha = opacity_val),
            fill = accent) +
  geom_text(aes(x = 0.5, y = 0.7, label = state),
            family = "orb",
```

```
            fontface = "bold",
            size = 9,
            color = bg) +
  geom_text(aes(x = 0.5, y = 0.3, label = round(num_obs_per10k, 1)),
            family = "orb",
            fontface = "bold",
            size = 8,
            color = bg) +
  geofacet::facet_geo(~state) +
  coord_fixed(ratio = 1) +
  labs(subtitle = "Sightings per 10k population") +
  theme_void() +
  theme(
    strip.text = element_blank(),
    plot.subtitle = element_text(family = "bar",
                                 size = 40,
                                 color = accent,
                                 hjust = 1,
                                 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 author builds the layers one by one. The `geom_rect()` sets up the opacity_val alpha information. Then, with the `geom_text()`, the author builds labels: State's postal code and observation numbers. The `facet_geo()` and `coord_fix()` set up the orientation and organization of the fill and label values in the US shape. Lastly, the `labs()` and `theme()` are the finishing touches to make the visualization look cohesive.

iii. Build `plot_day`

```
plot_day <- ggplot(data = df_day_hour) +
  geom_tile(aes(x = hour, y = day, alpha = opacity_val),
            fill = accent,
            height = 0.9,
            width = 0.9) +
  geom_text(aes(x = hour, y = 9, label = hour_lab),
            family = "orb",
```

```
            color = accent,
            size = 10) +
  geom_text(aes(x = 0, y = day, label = str_sub(string = wday, start = 1, end = 1)),
            family = "orb",
            fontface = "bold",
            color = bg,
            size = 8) +
  ylim(-5, 9) +
  xlim(NA, 23.55) +
  coord_polar() +
  theme_void() +
  theme(
    plot.background = element_rect(fill = bg, color = bg),
    legend.position = "none"
  )

plot_day
```

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

    - In the label parameter, the author told R to just put the first letter of the day column with `str_sub(string = wday, start = 1, end = 1)`

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.**

    - The `ylim()` hollows out the cicle graph by creating a min and max to the y axis. The `xlim` dictate how many "slices" are in the circle graph by creating limits on the x axis.

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")

quote1
quote2
```

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)?**

    - The author creates all the images and text elements prior in order to use `inset_element()` at the final plot. Breaking the visualization into various, smaller parts allows for easier manipulation.

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.

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

vi. Build `plot_base`

```
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)
  )

plot_base
```

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`)?**

    - `element_markdown()` interprets the text as Markdown/HTLM. Whereas `element_text()` is just for text formatting. Therefore, `element_markdown()` is better for more complex formatting, such as bolding, italics, and line breaks.

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?**

    - The `plot_final` is constructed by using `inset_element()` for all perviously created plots and texts. The most challenging aspect was most likely tediously maneuvering all the pieces to fit perfectly with the parameters `left = #, right = #, top = # bottom = #`. Also, when examining each element, the outputs looked crazy because they were designed for the size and poritions of the `plot_final`. When first creating the data visualization, it was probably difficult to size all the outputs correctely to look adequant in the final plot.

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

    - `plot_base` was used as a foundation for all the other elements to be stacked ontop of in order to create the `plot_final`.

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 viewer must distinguish between shading and saturation for all the plots, especially in figure 2 and 3. Additionally, in the first figure, the viewer must also differentiate between portions.

    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.**

- The author is showing the audience the trends of UFO sightings - light shapes, in the early mornings, and in Virgina and Montaina are the most common sighting traits. In the first figure, the author is comparing the reported shapes. In the second figure, the author is showing sightings' spatial pattens. In the third figure, the author is examing the temporal patterns of the UFO reports.

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?**

- By placing the figures on a black background, it allows the viewers to differienate between the hues/saturations easier. Also the chosen color, has high luminance when the hue is higher, letting the viewer see the trends better.

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

- I love the first bar graph because it is super clear. The observation numbers are on the graph itself and the proportions between the bars are well done. Additionally, I like the USA sightings per state figure. The figure design is visually appealing and geographically intuitive.

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

- .(1) The quotes are confusing. I do not understand the quote themselves, why they are incomplete, and the author's intent behind them. (2) The plot titles need to be larger as they seem smaller than the quotes. The titles get lost in the infographic.

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.**

- As time consuming as this was, I learned tons! Firstly, the data wrangling happening in a couple massive pipes was beautiful, concise, and easy to understand. When looking at the data visualization side, the organization was interesting - making all the elements first, then combining everything at the end. I learned the `glue()` function, and how to construct personalized themes with `theme_void()` + `theme()`.

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?**

- My strategy for this homework was (1) comment the portion of code out and visually see the difference, (2) look up the documentation in R, (3) look up the documentation on Google, and (4) ask AI. I only used AI for the rescaling `opacity_val` question. After using all the documentation, I was still confused on why the author rescalled

0.3 to 1. Documentation for `rescale()` used nine words to losely described *what* the function did. AI could help me help me understand *why* the author choose 0.3 and 1.