HW #1

Interpreting {ggplot2} code

Jaslyn Miura

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 ufo_sightings <- read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/main
13 places <- read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/main/data/
14
15 alien <- c("#101319", "#28ee85")
16 bg <- alien[1]
17 accent <- alien[2]
18
19 ufo_image <- magick::image_read(path = here::here("images", "ufo.png"))
20
21 sysfonts::font_add_google(name = "Orbitron", family = "orb")
22 sysfonts::font_add_google(name = "Barlow", family = "bar")
23
24 sysfonts::font_add(family = "fa-brands", regular = here::here("fonts", "Font Awesome 6 Brand
25 sysfonts::font_add(family = "fa-solid", regular = here::here("fonts", "Font Awesome 6 Free-S
26
27 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?**

   - In lines 15-17 the author is first defining a color palette, with two colors, using their hex codes. Then the author is defining the first color in the alien palette as "bg" or a background color and the second color as "accent" or as an accent color.

These defined variables are then used consistently in the creation of the various plots within the code. By defining the values initally at the beginning of the script, this ensures that for any further steps throughout the entire script a consistent color palette is used, and the author can indicate the value name such as `bg` or `accent` rather than long hex codes.

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()` searches the Google Fonts Repository. The arguement `name` is the name used to search the Google Fonts Repository. The arguement `family` is a name that refers to the searched font and is used in the R plotting functions.

II. Data wrangling

i. Create `df_pop`

```r
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 data frame takes the observations from the `places` dataset, where the country code is "US", and groups the observations by state, to find the total sum of the population of each staete. Including the District of Columbia.

ii. Create `df_us`

```r
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 data frame takes the observations from the `ufo_sightings` dataset, where the country code is "US", and groups the observations by state, to find the count of ufo sightings in each state. Then by joining the ufo counts to the `df_pop` dataframe by the state column the data frame then contains information about the count of ufo sightings and population size of eachs state. Then the author created columns that calculate the number of observations per 10,000 people. The author also created an `opacity_val`, by taking each calculated number of observations per 10,000 people and dividing it by the maximum calculated the number of observations per 10,000 people. Over all the date frame contains, state abreviation, number of ufo observations, population size, number of observations per 10,000 people, and the opacity value.

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

   - `opacity_val` is calculated to demonstrate the intensity of the sightings on 0-1 scale. This is done so that the sighting for each state can be more comparable amongst each other.

iii. Create `df_shape`

```r
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 `shape` column contains the type of shape observed. The `total_sightings` column contains the count of observations for each shape. The `opacity_val` column contains a value related to opacity/transparency for the observation. Again, this is a way to demonstrate the intensity of the sightings on 0-1 scale.

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

    - `fct_reorder` when applied to the `shape` variable reorders the categorical shape variable in order based off of the `total_sighting` value. If this step was not performed then the `shape` column would be alphabetically ordered, which could make it harder to compare the values to each other, as descending order based on the `total_sightings` variable makes it easier to see which shapes were reported the most.

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

    - The purpose of rescaling `ocaity_val` is to have a defined scale that the `alpha` arguement in plotting can use to display the intensity of the sightings, based on transparency/opacity. By rescaling from 0.3 to 1, the shpae with the lowest reported sightings will be represented by a transparency scale of 0.3, to ensure that the observation is still visible.

iv. Create `df_day_hour`

```r
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"))
    )
```

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

   - The `day` column contains a value that represents the numbered day of the week, where Sunday is the first day (1). The `wday` column contains a string, that is the abbreviation of the day of the week. The `hour` column contains a value that represents the hour of the day, these values range from 0 to 23. The `total_daily_obs` column contains the number of total observed ufo sightings for each hour. The `opacity_val` is a value that will demonstrate intensity of the sightings on 0-1 scale, through transparency/opacity.

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

    - This line of code takes the hour column, subtracts 12, to obtain a standard hour time, then pastes "pm" after the hour.

III. Prepare text elements

```r
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()` converts objects into character vectors first before pasting the given character string together. `glue()` concatenates strings provide in " ". `{}` can be used within `glue()` to insert R objects.

IV. Build plots

i. Build `plot_shape`

```r
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`).**

    - The values in the `x` arguement of the aesthetics for the `geom_text` indicate at what position the text will be located. All the `shape` labels are aligned at the 200 mark on the x axis for each of their respected bars. Then the values for `total_sightings` are aligned at the value of `total_sightings` minus 200. This is done so that there is a bit of space between the labels and the start/end of the bars.

ii. Build `plot_us`

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

```r
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 order of the layers first creates the rectangles for each state, coloring it by the defined accent color and with transparency value determined by `opacity_val`. The next layer adds the label for each state, which is it's abbreviation. The next layer adds the value for the number of observations per 10,000 people. Within these three layers, it appears that all the rectangles are concentrated in one area of the plot, to fix this the author then facets the graph by state, using `facet_geo()`, which then arranges each rectangle into a position that resembles a map (where each state is where it would lie on a map). The following layers then add a title and adjust the theme of the plot.

iii. Build `plot_day`

```r
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"
  )
```

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?**

    - This was done through `geom_text()`, line 295. Within the arguement, by using `str_sub()`, the string stored in wday is extracted. Since the start and end arguements are defined as 1, only the first letter of the string is extracted, therefore allowing us to only plot one-letter labels for each day of the week.

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()` function positions each day so that it aligns between the values 1-7 on the y-axis. Sunday being positioned at 1. This also provides a bit of extra space between Sunday and -5. By ignoring this step, when the plot is transformed into a circular/polar chart, the bar associated with Sunday can encompass the entire circle. Rather than being confined to a small space at the center of the circle. The `xlim()` function limits the bar to the hour 23.55, to prevent overlap of the bars, when the plot is transformed into a circular/polar chart.

iv. Build `quote*s`

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

```r
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)?**

    - The author chose to convert the text elements into ggplot objects to make styling and positioning of the object easier, the text elements now acts as and can be treated as a plot. In following steps, these object can be simply added to the final plot using the insert_element() function.

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`)?**

    - The author renders `plot.caption` using `ggtext::element_markdown()` so that the text uses markdown text. This ensures that the text can be bolded, italicized, styled, and include clickable links when rendered.

V. Assemble & save

```r
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?**

    - `plot_final` is rendered through insterting and arranging all elements of the plot onto a singular base, using the `insert_element()` function. The most challenging aspect of arranging all components into a single plot is likely identifying the correct positioning coordinates for element. This probably takes much trial and error, before being satisfied where the elements are positioned.

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

    - By seperating the construction of `plot_base` and `plot_final` there are less elements that need to be added to the `plot_final`. This also ensures that key elements, such as the title, subtitle, and captions are in their fixed positions, also making them a helpful reference point when using `insert_element()` for `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 main "task" is differentiating the transparency of the objects to determine the ranking/order of the observations. Transparency/opacity is a feature used for all the figures, to represent the differences between observations. The least transparent objects being the observations with the highest values/most importance. Another "task" is determining the difference in bar lengths, for the bar graph.

    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 bar chart, the author likely wanted to show the significant differences in reported shapes, where light is clearly the most reported shape. This is easily identified by determining the difference in bar lengths. For the sightings per 10k population figure, the author likely arranged the values into a shape that resembles a map of the United States so that the viewer can identify any patterns of where there might be more sightings. This is also a much easier spatial pattern for viewers to read, rather than if the values were organized in alphabetical order based on state name. Even though the final figure doesn't have a title or caption indicating what it is portraying, it is still fairly easy to determine that it reports what time of day and which day of the week the the most sightings were reported. This is due to it's circular shape, which resembles a clock. The author applied transparency features to all the figures to highlight the important observations of each figure, by doing so our eyes are drawn to the objects on the entire figure that are the easiest to read, which are highlighted through the lowest transparency setting.

    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?**

    - The author applied transparency features to all the figures to highlight the important observations of each figure, by doing so our eyes are drawn to the objects on the entire figure that are the easiest to read, which are highlighted through the lowest transparency setting. Shading falls lower on the "hierarchy of perceptual tasks", however, with this specific color scheme the shading creates a highlighting effect that is easy to identify; than if a larger color palette was used.

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

    - I really like the use of transparency/opacity to highlight the important aspects of the figure. My eyes are immediately drawn to the objects that are colored the most. The color palette is also very on theme with ailens/ufos. I also really like the format of the Sightings per 10k population figure. Since the observations are arranged in a map formation, it is intuitive to see that there are more sightings per 10k population in the west and east coasts of the US.

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

    - With the background coloring, I think the observations with greater transparency are very washed out. Though these values may not be as significant, if they are included in the figure they should still be readable for the viewer. I think that the hour with the most reported sightings could be presented differently. Perhaps as a bar chart that indicates the number of sightings per hour between a certain range of hours. By looking at the graph, the transparency values appear the same so it's hard to determine if there is a real difference between hours and days.

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.**

    - It was helpful to learn about `sysfonts::font_add_google()`. I plan to use this function to add different fonts to the figures I make. Though it may seem like additional code I thought it was helpful to see the author convert the text elements into ggplot objects. It seems like it's easier to make adjustments to the text elements and once we're satisfied convert it to a ggplot object to avoid long lines of code in our final object. Using the `insert_element()` function demonstrates how easy it can be to apply a fixed element to the final object using one line of code. It was also helpful to see how the author first created a base, with some elements of the final figure, rather than just doing everything in one step. Though it may seem easier, this process allowed the auther to fiddle with the sizing of the title, caption, and subtitle before adding additional elements.

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?**

- Some of the documentation is very vauge, such as `element_markdown()`, so in this case using AI tools to further explain what the function does was helpful. For instance, the documentation says "Theme element that enables markdown text". Using AI, I learned that this means that bolded, italicized, and linked text is enabled, which was helpful to know, as the authour applied this function because the text they used had a link included. AI was also helpful in providing further explanations for the arguements within the function, since again, it can be vauge.