HW #1

Interpreting `{ggplot2}` code

Assigned Wed 01/07/2026 | Due Wed 01/14/2026

AUTHOR
Your Name

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

```
library(colorspace)
library(geofacet)
library(ggtext)
library(glue)
library(grid)
library(magick)
library(patchwork)
library(scales)
library(showtext)
library(tidyverse)

ufo_sightings <- read_csv('https://raw.githubusercontent.com/rfordatascience/tidytue
places <- read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/ma

alien <- c("#101319", "#28ee85")
bg <- alien[1]
accent <- alien[2]

ufo_image <- magick::image_read(path = here::here("images", "ufo.png"))

sysfonts::font_add_google(name = "Orbitron", family = "orb")
sysfonts::font_add_google(name = "Barlow", family = "bar")

sysfonts::font_add(family = "fa-brands", regular = here::here("fonts", "Font Awesome
sysfonts::font_add(family = "fa-solid", regular = here::here("fonts", "Font Awesome
```

```
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?**

   The author is defining graphical aesthetics that will remain consistent throughout the project. By making these pre-defined in the set-up for the project, the author can reference them as they build their data visualizations, streamling the process and eliminating repetitive code.

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 is a function that searches Google adds fonts to your plots. The `name` argument is the Google Fonts name that will be searched and `family` specifies the font that R can understand.

II. Data wrangling

i. Create `df_pop`

```
1  df_pop <- places |>
2    filter(country_code == "US") |>
3    mutate(state = str_replace(string = state,
4                               pattern = "Fl",
5                               replacement = "FL")) |>
6    group_by(state) |>
7    summarise(pop = sum(population)) |>
8    ungroup()
```

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

   - After importing the data and establishing plot aesthetics, the author proceeds to wrangle the data. The author proceeds to filter the `places` data frame to just the US region, and modifies the state column with the `str_replace` function to modify the state string pattern Aa to be both capitalized. The author then groups by state to obtain a sum of the population per state. In order to prevent `df_pop` from being grouped by state, the author ungroups. This produces a dataframe with two columns; one column for the state and another column for the population per state.

ii. Create `df_us`

```
1  df_us <- ufo_sightings |>
2    filter(country_code == "US") |>
3    mutate(state = str_replace(string = state,
4                               pattern = "Fl",
5                               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.**

   - After producing a dataframe with the total population per state, the author proceeds to create a dataframe with the `ufo_sightings` data. The author filters this data to the US and uses `str_replace` to update the state column similar to the `df_pop` dataframe. This dataframe contains a row for state, number of ufo observations per state, the population per state, and the number of observations per 10,000 inviduals which is used to calculate the opacity value column.

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

   - The author renames the `n` column to be called `num_obs` and creates two new columns with the `mutate()` function; one column is the number of observations per population and the other column uses the `num_obs_per10k` to produce an opacity value based on the maximum of the `num_obs_per10k` column. This value is calculated because when plotted, it controls the opacity based on the number of observations per 10,000 people.

iii. Create `df_shape`

```
df_shape1 <- ufo_sightings |>
  # Filter df for any shape not unknown or other
  filter(!shape %in% c("unknown", "other")) |>
  # count the number of shapes
  count(shape) |>
  # rename "count" column
  rename(total_sightings = n) |>
  # Arrange total sightings in order
  arrange(desc(total_sightings)) |>
  # only keep the top 10
  slice_head(n = 10) |>
  # reorder the shape column so that it is ordered by total sightings
  mutate(
    shape = fct_reorder(.f = shape,
                        .x = total_sightings),
    # Create column using total sightings for opacity, rescale
    opacity_val = scales::rescale(x = total_sightings,
                                  to = c(0.3, 1))
    )
```

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

- This dataframe contains three columns. One column contains the top ten most common shape of ufos, another column contains the total sightings per shape, and another column contains the opacity value to color the shapes by.

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 function `fct_reorder` reorders the shape columnso that it is ordered by total sightings. If this step is not performed, then the dataframe will not be order in descending order of sightings.

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

   - Rescaling `opacity_val` ensures that the opacity value is not too intense for plotting. The opacity value is derived from the `total_sightings` columns, and this does not ensure that the opacity will be greater than 1 (too bright) or less than 0.3 (not enough brightness). By rescaling, you can adjust to the correct opacity, which can be altered based on what works with the data.

iv. Create `df_day_hour`

```
df_day_hour <- ufo_sightings |>
  # Get days from a date time
  mutate(
    day = wday(reported_date_time), # Extract day from column with wday function
    hour = hour(reported_date_time),  #Extract hour from column
    wday = wday(reported_date_time, label = TRUE) # Extract week day from colum as a
  ) |>
  # Sum the totals of the three columns
  count(day, wday, hour) |>
  # Rename the count to be total daily obs
  rename(total_daily_obs = n) |>
  # Create an opacity value based on the total daily observations
  mutate(
    opacity_val = total_daily_obs / max(total_daily_obs),
    # Label the hours into three categories
    hour_lab = case_when(
      hour == 0 ~ "12am", # hour 0 = 12 am
      hour <= 12 ~ paste0(hour, "am"), # hours less than or equal to 12 = hour "am"
      hour == 12 ~ "12pm", # hour 12 = 12 pm
      TRUE ~ paste0(hour - 12, "pm"))
    )
```

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

   - This data frame contains three columns pertaining to daily observations of ufo sightings. It has four time-related columns including; a column called `day` containing the weekday number, a column called `wday` with the actual character day of the week, an hour column for the 24 hour

clock, and a column called `hour_lab` that is the 12-hr clock label. Additionally, it has a column for total ufo sightings per day, and an `opacity_val` column for plotting.

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

    - The `case_when()` function is used to vectorize if-else statements. In order to produce the `hour_lab` column, the hour column is filtered and "binned" into specific cases if they apply to it. The last statement ensures that any hours that do not satisfy the above statement are binned as "pm" with their respective times subtracted from 12.

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/tidyt
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.g
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?**

    - The `paste0` function is an efficient way to concatenate character strings (e.g., qoutes) without specifically specifying the `sep = ""` argument in `paste()`. The `glue()` function can integrate code within statements with braces, similar to the f-string in Python. This is helpful when annoting plots.

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_sigh
            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"
  )
# View
plot_shape
```

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

    - Within the `geom_text()` argument, x represents the positioning along the x axis. The first `geom_text` defines where to put the label for each bar with the shape name, with x = 200 placing the bar slightly within the left edge of the bar. The second `geom_text` defines where to place the text, placing thise one 200 units to the right.

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) +
  # Plot rectangles colored by the opacity value of df_us from 0 to 1 on x and y axi
  geom_rect(aes(xmin = 0, xmax = 1, ymin = 0, ymax = 1, alpha = opacity_val),
            fill = accent) + # fill with customized color
  # Add a label foreach state, specify where on each individual square the label sho
  geom_text(aes(x = 0.5, y = 0.7, label = state),
            family = "orb",
            fontface = "bold",
            size = 9,
            color = bg) +
  # Add text label for number of observations per state
  geom_text(aes(x = 0.5, y = 0.3, label = round(num_obs_per10k, 1)),
            family = "orb",
            fontface = "bold",
            size = 8,
            color = bg) +
```

```
  # Facet by state using the geofacet package
  geofacet::facet_geo(~state) +
  # Ensures map is a 1:1 ratio
  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 orders them by first establishing one single rectangle, specifying its position and aesthetics. The author then layers on text for state and ufo sightings. Since ggplot layers on features, we need to establish the background shape and text and then use `geofacet::` package to facet by state on a map of the U.S.

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"
  )
```

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 is done with line 314 of the code where the author uses `str_sub` to extract one letter with the start and end arguments.

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.**

    - In a ggplot, the `ylim()` and `xlim()` features can adjust for the limits to the x and y axis. For this particular plot, these coordiantes are used before the `coord_polar` feature (used for pie charts), which makes y the radius of the plot. Since the y axis is day, the limit specifies how the days should line up along the radius, and the x-axis corresponds to the hour, the numbers controls the way the tiles wrap around the pie.

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
quote3
```

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 decided to convert the text elements into ggplot objects in order to plot them on the base plot in lines 468-470. By converting them into ggplot objects, they can be adjusted with more details and placed in specific locations on the base plot with `inset_element`.

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`)?**

    - The author uses `element_markdown()` to render more caption features that are not used in the `element_text()` feature of ggplot. Additionally, using `element_markdown` is helpful for captions because they might already contain some markdown styling and often include different styles in one line, which this feature can perform.

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?**

    - `plot_final` is assembled by first initializing a base plot with `plot_base()`. Then, the shapes are positioned by defining their position on the base plot with the left, right, and bottom arguments. The plot is then filled with a background color from the pre-defined aesthetics. The most challenging aspect of arranging all components into a single plot is specifying the arguments correctly and layering the ggplot elements correctly.

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

    - By adding the `inset_element` to the plot base, this allows for you to add layers onto the base like painting a canvas. These pieces for the `plot_final` and can be adjusted and iterable when needed.

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 final visualization is easy to understand because the author balances these percetual tasks. For instance, the bar chart of top ten most commonly reported shapes is an example of great positioning, where the viewer judges the length of the bars to obtain information. Additionally, the author utilizing hue and shade to display patterns in the data. For instance, the map of sightings per 10k population is easy to understand because it utilizes hue to show which states have the most ufo sightings.

    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 was interested in showing individual values between ufo shapes, state sightings, and time of day sightinings, however they also wanted to compare these trends within each respective plot. For the “ten most commonly reported shapes” graph, the author was interested in showing trends across reported shapes. The inclusion of the data values for each shape was very intentional and also allows you to compare the individual values across shapes. For the “sightings per 10k population” graph, the author is comparing individual state values for ufo sightings, and due to the nature of the graph being a map, it also is including a spatial element to the message. For the pie chart graph of the most common hours, this graph is clearly displaying big picture trends in ufo sightings on a temporal scale.

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?**

- One caveat is that the when glancing at the graph, you can easily draw conclusions solely based on the way the elements are arranged. For the “10 most commonly reported shapes”, the data most clearly shows that light is the most reported shape with the inclusion of the data value. By making this bar the strongest hue, it clearly displays the message.

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

- I think that the use of hue and the inclusion of the map of the US are visually-pleasing. I find that using hue against the black base plot is eye-catching and also easy to understand. I think the inclusion of the map of the US was very smart, as you could just show the top ten most common states with ufo sightings, but by including a map you get to see the trends spatially which is interesting.

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

- I am personally not a fan of the bottom left graph because of the lack of title and the visual is less intiutive. You can infer that the hours 1 am to 4 am are peak hours for ufo sightings, however the hours around the chart coupled with the grid-like style add too much visual cognitive load.

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 the use of pre-defnining graphing aesthetics before graphing. I learned new data wrangling techniques with `fct_reorder`. I learned about the `geofacet` package.

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 utilized ChatGPT for a few questions for this assignment. I found that when prompting it with the code and asking for it to add comments, that was a quick way to break down the complexities. What I found more helpful was looking up the documentation in the console. Since this is how I originally learned ot code, I found that this interface is easy to understand especially when viewing what arguments do what in each function.