Notes 05 - Customzing ggplot2 Graphs

STS 2300 (Spring 2025)

Updated: 2025-02-25

Table of Contents

Reading for Notes 05

Chapter 11 of R for Data Science is a good resource for this section. In particular:

- section 11.2 talks about adding labels to your graph,
- section 11.4 goes into more detail about controlling scales, and
- section 11.5 includes information about customizing themes.

Learning Goals for Notes 05

- Be able to customize graph titles, captions, and variable labels.
- Be able to edit the scales of your graphs to improve the visual appeal and to better convey information.
- Be able to customize color schemes and aspects of a graph's theme
- Be able to create images containing multiple graphs (using facetting or by combining separate graph objects)

Customizing graph titles, captions, and variable labels

Recall that when we use the `ggplot2` pacakge, we add layers to our graph as we create the final product (e.g. `ggpplot()` + `geom_histogram()` + `theme_bw()`). If we want to add or edit titles, captions, axis labels, etc., we can add a layer using the `labs()` function. Within this function, we have arguments for title, subtitle, caption, x, y, and any variable mapped to a part of our graph with `aes()`.

The general syntax will look like:

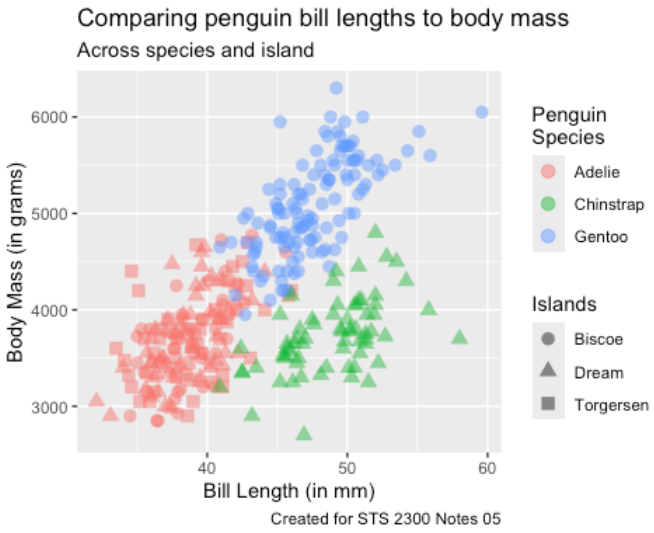
```
ggplot(_____________________, aes(_____________________________________))
+
  geom_xxxxx() +
  labs(title = "Title of your graph",
       subtitle = "Subtitle for your graph (right under the title)",
       caption = "Caption for your graph (bottom right corner)",
       x = "Label for your x-axis (consider including units)",
       y = "Label for your y-axis (consider including units)",
       ______ = "_____________________________")
```

Below is an example of what this looks like for a graph created using the `penguins` dataset from Activity 04.

```
library(ggplot2)
library(palmerpenguins)

graphA <- ggplot(penguins, aes(x = bill_length_mm,
                               y = body_mass_g,
                               color = species,
                               shape = island)) +
  geom_point(size = 3,
             alpha = 0.5) +
  labs(title = "Comparing penguin bill lengths to body mass",
       subtitle = "Across species and island",
       caption = "Created for STS 2300 Notes 05",
       x = "Bill Length (in mm)",
       y = "Body Mass (in grams)",
       shape = "Islands",
       color = "Penguin \nSpecies")

graphA
```



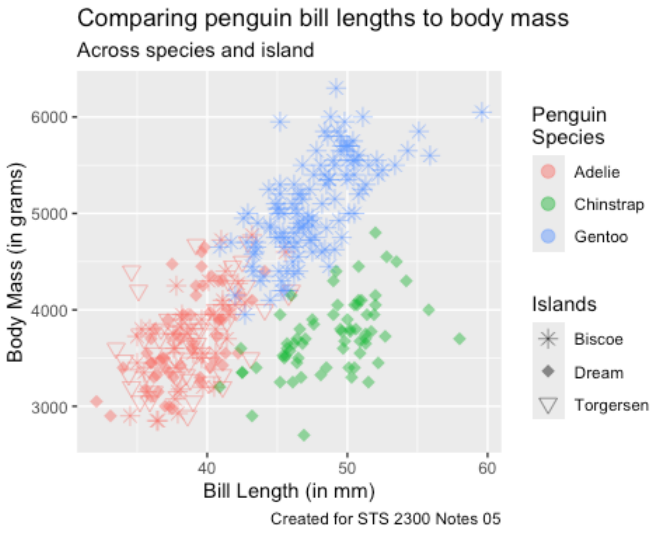
Notice how because I mapped `species` to color and `island` to shape there were legends created for both of those aesthetic elements. By default, the legend title would be the variable name. I was able to use `shape` and `color` arguments inside `labs()` to change these labels. (Note: By adding `\n` as part of my `color` label, I was able to tell R to add a line break there.)

Editing graph scales

There are a number of functions in the `ggplot2` package that follow the format: `scale_xx_yy` where xx is an aesthetic element like `color`, `fill`, `shape`, etc. and yy is a word like `continuous`, `discrete`, or `manual`.

Suppose that I didn't care for the three shapes that were used in my graph above. I could add `scale_shape_manual()` and choose those shapes using the list of shape values from Notes 04.

```
graphA +
  scale_shape_manual(values = c(8, 18, 25))
```



We could also do this with `scale_color_manual()` to change the colors.

Practice: Try using the example above to add a layer to your graph with `scale_color_manual()` that changes the colors of our three penguin species. You will still use the `values` argument to do this. Pick whichever three colors you prefer.

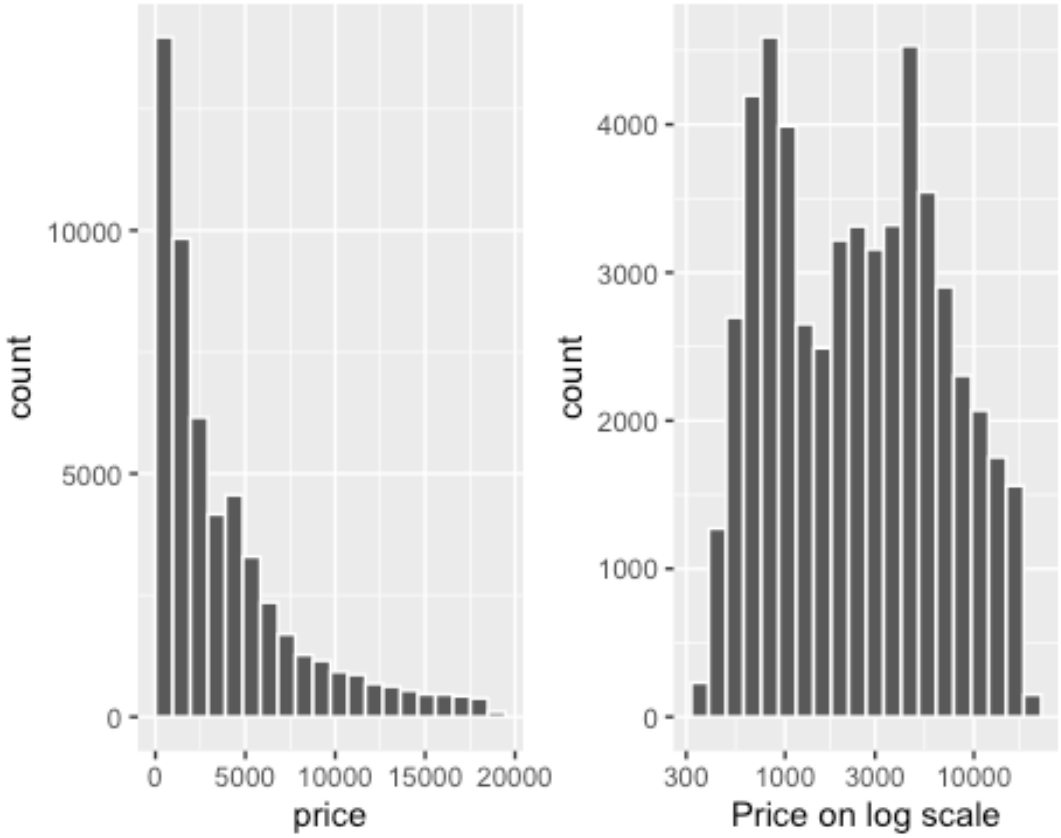
See notes05.R.

With aesthetic elements x and y, there are additional options for the part after the last underscore. For example, I may want to convert a numeric variable to the log (or square root) scale. Below is an example using the diamonds data. The graph on the left does not use a scale function. The graph on the right adds `scale_x_log10()`.

```
library(patchwork)

## Warning: package 'patchwork' was built under R version 4.3.3

left <- ggplot(diamonds, aes(x = price)) +
  geom_histogram(color = "white", bins = 20, boundary = 0)
right <- left +
  scale_x_log10() +
  labs(x = "Price on log scale")
left + right
```

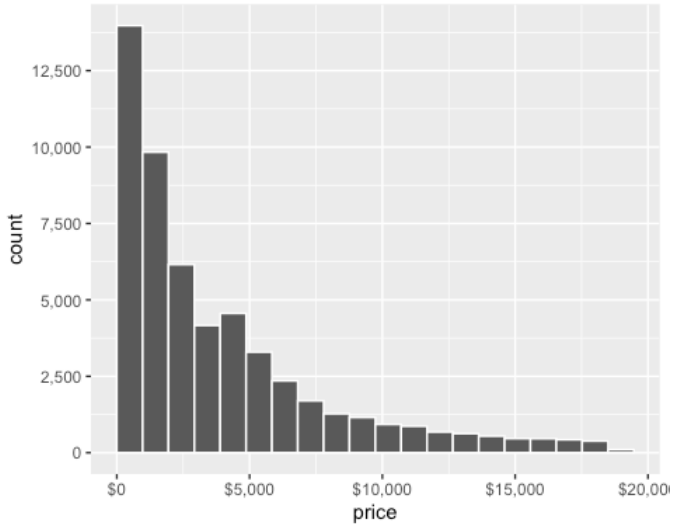


Question: How is the log scale transforming the way the data is displayed? In other words, what does it mean to put data on a log scale?

Answer: Before the transformation, our axis jumps in groups of $5000 (e.g., 5000 to 10,000 to 15,000). After the transformation, our axis jumps in multiples (e.g., 300 to 3000 is the same distance as from 1000 to 10,000).

Within these functions, we can tell R to reformat our axis values by adding commas in longer numbers, adding dollar signs to values that represent money, or converting proportions to percentages. Let's do two of those for our diamond price graph (commas for our y-axis counts and dollar signs for our x-axis prices). I've also demonstrated how `breaks` can be used to change which values are displayed on our axes.

```
left +
  scale_x_continuous(labels = scales::label_dollar()) +
  scale_y_continuous(labels = scales::label_comma(),
                     breaks = seq(0, 15000, 2500))
```



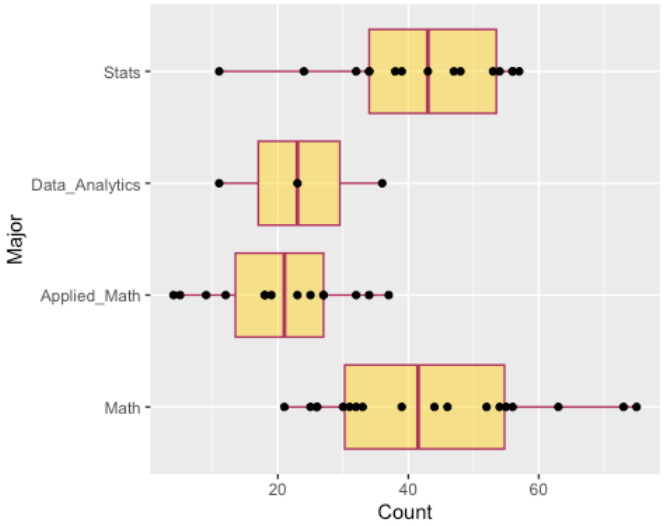
Question: What do you think the `seq()` function is doing here? What's another way we could have written our code for that argument? (Hint: It uses the shortest function name we've learned)

Answer: seq() is short for sequence. We are using this to get a vector of numbers from 0 to 12,500 by increments of 2500. We could have done this using the c() function, but we would have had to write out six numbers to do it.

With categorical data, you may want to re-order the categories on your graph. For example, if I make a boxplot of majors using the MTH/STS Department majors data, the default is to plot the majors in alphabetical order (Applied_Math, Data_Analytics, Math, Stats). If I want to group the two math majors together and the two data majors together, I can do this using the `limits` argument.

```
majors <- read.csv("https://raw.githubusercontent.com/nbussberg/STS2300-Spring2025/refs/heads/main/Data/MTH_STS_majors_long.csv")

ggplot(majors, aes(y = Major, x = Count)) +
  geom_boxplot(fill = "gold", color = "maroon", alpha = .5) +
  geom_point() +
  scale_y_discrete(limits = c("Math", "Applied_Math", "Data_Analytics", "Stats"))
```



The limits argument can be used to order legends for `color`, `fill`, etc. as well.

Question: What else did I do with this graph that we haven't seen before? Why might this be useful?

Answer: We are using two geoms on the same graph to combine features of boxplots and scatterplots. In this case, it allows us to see the general shape of the data (boxplot) along with specific values (scatterplot).

Color schemes and graph themes

We saw above that we can use `scale_color_manual()` or `scale_fill_manual()` to choose our own color schemes for the `color` or `fill` aesthetics. There are also **lots** of schemes others have created that we can choose from. One popular option is the `scale_xx_viridis_yy` series where `xx` is either `color` or `fill` and `yy` is either `c` (for continuous, aka quantitative) or `d` (for discrete, aka categorical). Within these functions is an argument called `option` that can take a letter from "A" to "H" as its value. Each of these is a different color-blind friendly color scheme that will also be readable when printed in black and white.

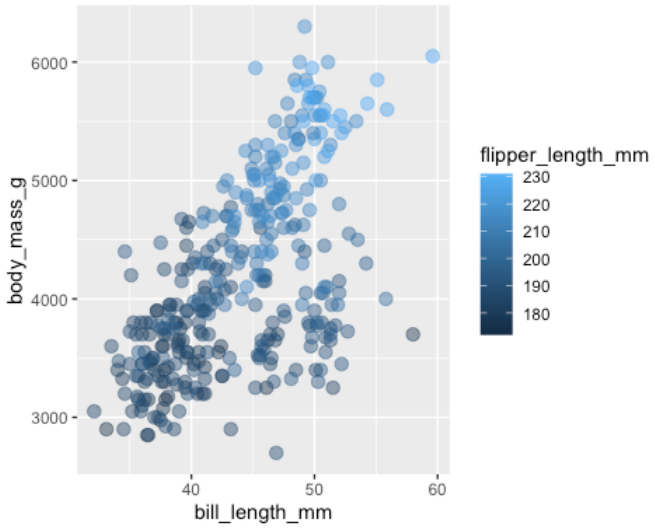
Practice: Try adding the appropriate one of these functions to our penguin scatterplot at the very top. Then choose an option you like.

See notes05.R.

Practice: Add the appropriate function to the graph below and choose an option you like. (Bonus: Edit the x and y axis labels, add commas to the y-axis, and change the values shown in the legend)

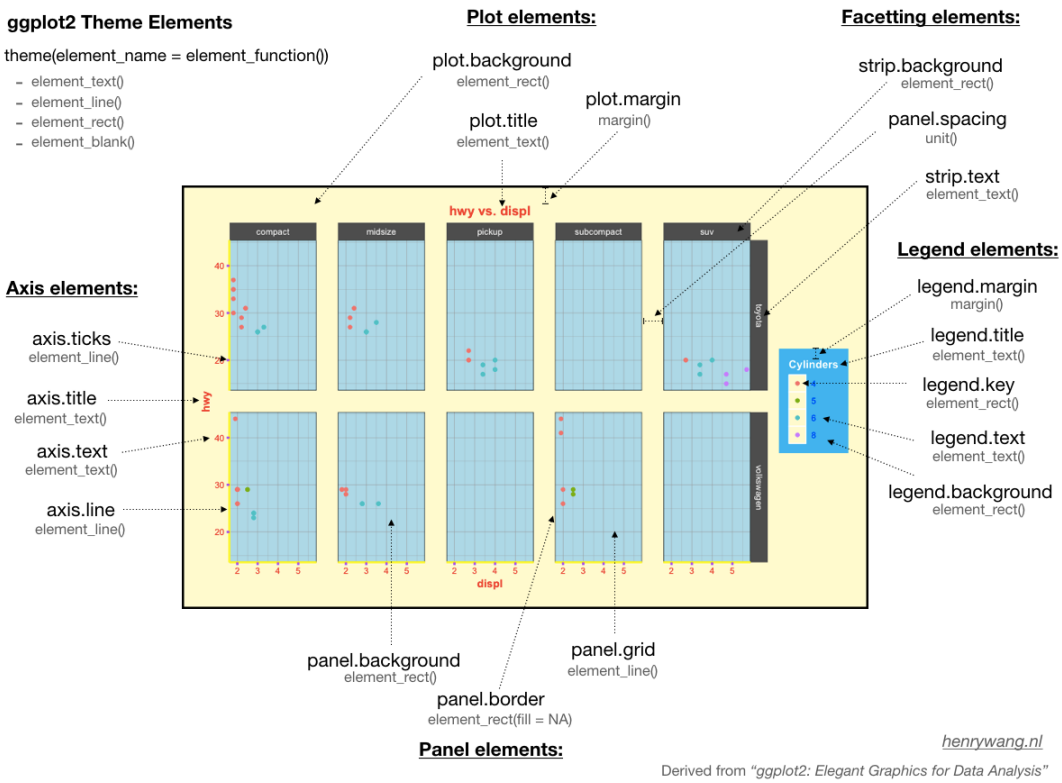
See notes05.R.

```
ggplot(penguins, aes(x = bill_length_mm, y =
                       body_mass_g,
                     color = flipper_length_mm)) +
  geom_point(size = 3, alpha = 0.5)
```



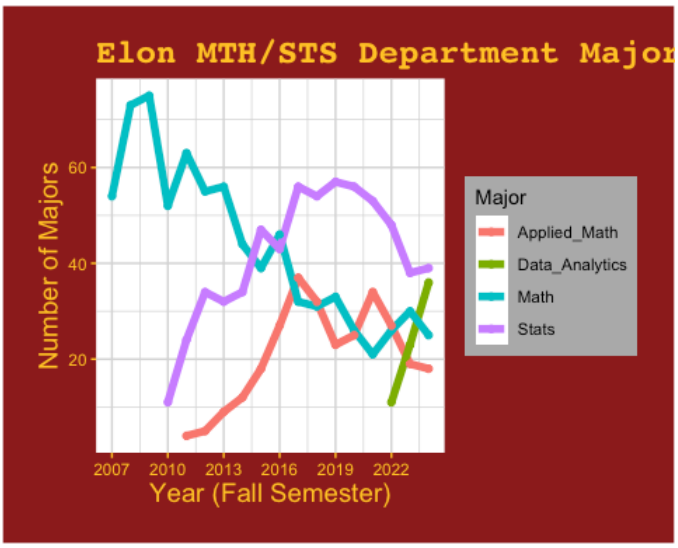
In Activity 04, we saw that we could add themes to our graph using `theme_xx()` where xx is one of the options from here. However, we may also want to customize our own themes. You can do this using the arguments in the `theme()` function.

This website contains a nice example of the elements of a theme that we can control. The picture below is from that site.



Suppose I want to create a theme I can add to graphs that uses Elon's colors. Below is an example to illustrate possibilities for a few of these arguments using our data on MTH/STS Department Majors. (Note: This theme could certainly be improved, but the goal here is to demonstrate many different ways to change elements of a graph's theme.)

```
ggplot(majors, aes(x = Year, y = Count, color = Major)) +
  geom_line(linewidth = 2) +
  geom_point() +
  labs(title = "Elon MTH/STS Department Majors",
       x = "Year (Fall Semester)",
       y = "Number of Majors") +
  scale_x_continuous(breaks = seq(2007, 2024, 3)) +
  theme(plot.background = element_rect(fill = "firebrick4"),
        panel.background = element_rect(fill = "white"),
        panel.grid = element_line(color = "lightgray"),
        plot.title = element_text(color = "goldenrod1",
                                  size = 18,
                                  face = "bold",
                                  family = "Courier"),
        axis.title = element_text(color = "goldenrod1",
                                  size = 14),
        axis.text = element_text(color = "goldenrod1"),
        axis.ticks = element_line(color = "goldenrod1"),
        legend.background = element_rect(fill = "darkgray"),
        plot.margin = margin(20, 20, 20, 20))
```



Images with multiple graphs

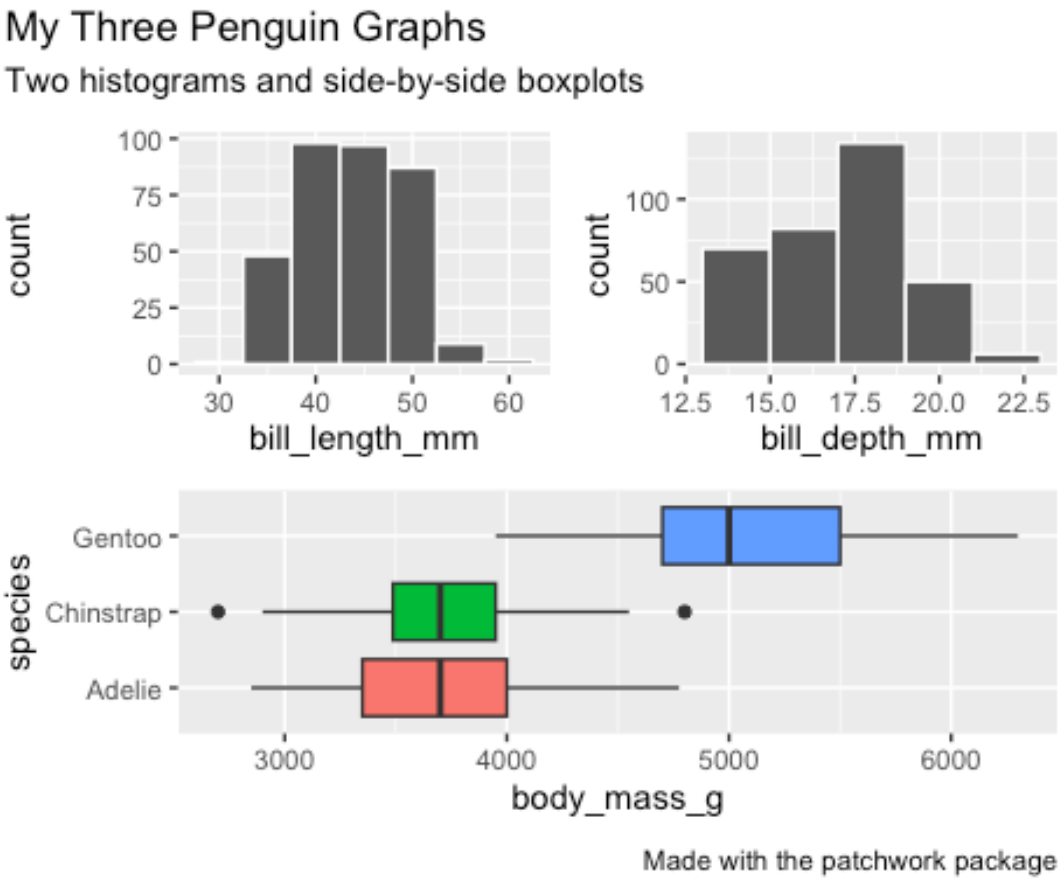
In Activity 04, you saw how you could use `facet_wrap()` or `facet_grid()` to create multiple graphs of the same type that are separated across levels of variables. In other instances, you may want to combine other graphs into a single image. This can be done with the `patchwork` package. To use this package, we assign our graphs to objects in our environment (like I did at the top of this document for the penguins graph stored as `graphA`). Then we can combine these objects into one image using `+` to put them side-by-side or `/` to stack them. We can also combine these two features.

For example, below I make three graphs with the `penguins` data and then combine them into a single image. Notice how I can use the `plot_annotation()` packages similar to how I've used `labs()` if I want to add a title, subtitle, and caption to the combined plots rather than to individual plots.

```
library(patchwork)

A <- ggplot(penguins, aes(x = bill_length_mm)) +
  geom_histogram(binwidth = 5, color = "white")
B <- ggplot(penguins, aes(x = bill_depth_mm)) +
  geom_histogram(binwidth = 2, color = "white")
C <- ggplot(penguins, aes(x = body_mass_g, y = species)) +
  geom_boxplot(aes(fill = species),
               show.legend = FALSE)

(A + B) / C +
  plot_annotation(title = "My Three Penguin Graphs",
                  subtitle = "Two histograms and side-by-side boxplots",
                  caption = "Made with the patchwork package")
```



Note: Notice that I was able to remove the legend for the boxplots using a `show.legend = FALSE` argument. This is a good idea for a graph like this because the fill argument is already mapped to labels on the y-axis, so the legend wouldn't be adding any new information.

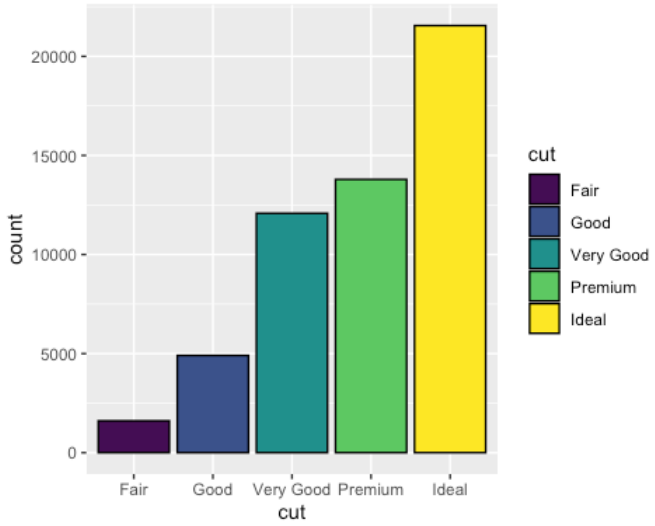
Practice: Choose a dataset to explore. Create at least three graphs (with at least one using scale and theme options) and combine them in a different arrangement than we did above. Add an overall title.

Answers will vary.

Revisiting the Learning Goals for Notes 05

- Be able to customize graph titles, captions, and variable labels.
    - Update this graph from Notes 04 to include an appropriate title. Change the color legend to say “Diamond Cut”. Add a caption saying “Made by __” with your name in the blank.

```
ggplot(data = diamonds) +
  geom_bar(aes(x = cut,
               fill = cut),
           color = "black")
```



- Be able to edit the scales of your graphs to improve the visual appeal and to better convey information.
    - Add commas to the y-axis values.
- Be able to customize color schemes and aspects of a graph’s theme
    - Choose a new color scheme for the fill of the bars using an appropriate function. Create a custom theme that changes at least three elements of the graph (e.g. panel background, plot background, panel grid, axis text, etc.)
- Be able to create images containing multiple graphs (using facetting or by combining separate graph objects)
    - Create a second graph that shows a histogram of diamond price facetted by cut. Then combine your new graph and the previous bar graph into a single image using the patchwork library. Add an overall title to this new image.