CHAPTER ELEVEN

**Note**: For all my answers, in my R Studio script, you will find the accompanying code present and/or needed to solve.

# 11.2.1

1. To create this plot, we'll use the ggplot2 package to visualize the relationship between engine displacement (displ) and highway fuel economy (hwy) in the mpg dataset. My plot becomes clearer and easier to interpret, especially for an audience who might not be familiar with the data due to the customized labels.
2. For the recreation, we'll use ggplot2 to plot city fuel economy (cty) versus highway fuel economy (hwy), customizing both color and shape based on the drive train. With the different shapes and colors, to match the given plot, it thus makes it easier to identify train types. More so, this plot helps to visualize how the type of drive train influences fuel economy, with each point representing a car's city and highway fuel efficiency.
3. Though similar to #1, this plot is different in the sense that the title and subtitle explain the general trend (larger engines tend to be less fuel-efficient), while the axis labels and legend make the graph easier to interpret. Adding this context helps others understand the data better without needing a detailed explanation.

# 11.3.1

1. To place text at the four corners of a plot using geom\_text(), you can use x and y values that go beyond the limits of your plot’s axes. By setting the x and y coordinates to extreme values, like Inf or -Inf, you can place text at the top-left, top-right, bottom-left, and bottom-right corners. This is helpful for labeling or annotating the boundaries of your plot.
2. With annotate(), you can add a point to your plot without creating a separate dataset (like a tibble). You just specify the geom = "point", choose the coordinates for where you want the point, and customize the shape, size, or color of the point by adding arguments like shape, size, and color. This makes it quick and easy to add a point for emphasis in the middle of your plot, or anywhere else you need it.
3. So, for this question, I used geom\_text() with a subset() to add a label only to the "rear-wheel drive" facet (drv == "r"). This way, I could put a custom label ("Rear-wheel drive") in just that facet while keeping the other facets clean without labels. I used position\_nudge() to move the label a bit up, so it doesn't overlap with the points. I found this a way to correctly add a label to a single facet without messing up the others.
4. For geom\_label(), to make things more interesting, I’ve added labels to the plot using geom\_label() to highlight specific data points, specifically the drive type for each point. The labels are positioned based on the displ and hwy values. I’ve also added a few styling tweaks to make the labels stand out. For example, I used label.padding to add some space around the text inside the label, and label.r to give the label box rounded corners. I also adjusted the label.size to control the thickness of the border around the label and set a light blue fill color with label.fill. The labels have been nudged upwards a bit (nudge\_y) so they don’t overlap with the data points, and the points themselves are a bit transparent (alpha = 0.3) to make the labels pop even more.
5. The four arguments to arrow() are type, length, angle, and ends. The type specifies the shape of the arrowhead (e.g., open or closed), length controls the size of the arrowhead, angle sets the angle of the arrowhead, and ends determine whether the arrow should be drawn on both ends or just one. In my code, you can see how each plot helps demonstrate how adjusting these four arguments can change the appearance of the arrows in my plots.

# 11.4.6

1. The issue here is that geom\_hex() by default uses the fill aesthetic, not color, to represent the hex bin values. So when you try to use scale\_color\_gradient(), it's not affecting the plot because the colors are controlled by fill. To override the default scale, you should use scale\_fill\_gradient() instead of scale\_color\_gradient(). This would then allow you to modify the color gradient for the bins.
2. The first argument to every scale in ggplot2 is typically the aesthetic being modified. For example, scale\_color\_\* is used to change the color aesthetic, scale\_size\_\* changes the size, aesthetic, and so on. These scales map the data values to visual properties (like color or size). On the other hand, labs() is used for labeling the axes, legends, and the title of the plot. While labs() allows you to assign names or labels to these components, scales control how the actual data values are visually represented. In short, scales handle the data-to-visual mapping, while labs() manages the text and labels on the plot.
3. For problems 3a through 3e, I created visualizations using ggplot2 to explore the relationships within the dataset. In 3a, I loaded the necessary libraries (ggplot2 and dplyr) to facilitate plotting and data manipulation. For 3b, I prepared a sample dataset containing variables such as variable1, variable2, and president, which I assumed would be the focus of the analysis. In 3c, I created a scatter plot using geom\_point() to visualize the relationship between variable1 and variable2, and added text labels from the president column using geom\_text() to annotate each point, helping identify each data point’s corresponding president. In 3d, I constructed a boxplot using geom\_boxplot() to compare the distribution of variable1 across different president categories, allowing for a clear visual comparison of how each president's data is spread. Finally, in 3e, I generated a density plot for variable2 using geom\_density(), providing a smooth estimate of the variable’s distribution. All in all, these visualizations aim to facilitate a deeper understanding of the data and its underlying patterns.
4. In this code, the override.aes function allows you to modify the legend's appearance. Consequently, I increased the size and opacity (alpha) of the points in the legend to make it more visible.

# 11.5.1

1. In the first plot, I used the ggplot2 and ggthemes packages to create a density plot of a variable (variable2) from a randomly generated dataset of 100 data points. The plot is enhanced by applying the theme\_economist() from the ggthemes package, which provides a clean, professional aesthetic often used for economic and financial visualizations. The theme\_economist() helps make the plot visually appealing by offering a minimalist design with subdued gridlines and a white background. The plot’s title and axis labels are added using labs(), which clearly labels the plot as "Density Plot of Variable2", with appropriate labels for the x-axis and y-axis, "Variable 2" and "Density", respectively.
2. For the second plot, I further customized the density plot created in Question #1 by adding a styling change to the axis labels. Specifically, I modified the x-axis and y-axis labels to be both blue and bold. This was done using the theme() function, where element\_text(color = "blue", face = "bold") was applied to both the x-axis and y-axis labels. This change makes the axis labels stand out more, improving readability and making the plot more visually striking. The rest of the plot retains the same theme\_economist() styling, ensuring a clean and professional look while adding the desired emphasis to the axis labels.