MODULE 4: Communicating your data using Rmarkdown!

Partial Answer Key for Comparison to R Script

Author: Kimberly Komatsu

Goal:

The ***goal*** of this file is to facilitate a comparison between an .R and .Rmd script.

Contents:

This file is an R Markdown Notebook. The script contains the same information as the first two sections of the 7_plotAgain R script answer key. Refer back to the .R file (on canvas) to compare and contrast the two versions.

How to use this file:

When you execute code within the notebook, the results appear beneath the code.

You can execute a chunk of code by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter* or *ctl+Shift+Enter* as in an R script.

See below for notes (in **bold italics**) related to specific tips on text formatting or code chunk settings.

When you have read through this entire file and understand the syntax to format text and add chunks of R code, try knitting the file as each of the three main document types (html, word, and pdf). Then open each file to see differences among them, as well as to understand what the code below has created.

Note here, the use of a backslash \ to specify line breaks in the document. Without these, the segments of text in your final document might be closer together than you'd prefer. Additionally, paragraphs can be indicated by leaving a blank line between segments of text or by adding a double space to the end of a line.

Objective:

1. To develop a "grammar of graphics".
2. To become comfortable using ggplot2 for exploratory data visualization.

In the above text, note the use of the sub-header formatting with a double ##, as well as the creation of a numbered list.

Set Up:

REMINDER: In RStudio, open the Rproject for our course (UNCG_DataWrangling), pull any changes from GitHub, and navigate to the branch you created for yourself (just a single branch for the whole semester) in RStudio's Git tab. Open this week's assignment in this new branch to complete and submit on GitHub. Ideally, you'll commit your answers a bit at a time as you work through this assignment, rather than committing all at once at the end.
In the above text, note the use of a single astrisk to create italicized font.

Reminder of some helpful resources for graphing in R:
- ggplot Cookbook
- Official ggplot2 Book
- Themes in ggplot2
In the above text, note the creation of a bullet-point list.
Additionally, note the inclusion of hyperlinks by surrounding the desired text with [] and the link with ().

In ggplot2, aesthetic means "something you can see". Each aesthetic is a mapping between a visual cue and a variable.
Examples include:
- position (i.e., on the x and y axes)
- color ("outside" color)
- fill ("inside" color)
- shape (of points)
- line type
- size

Each type of geometric object (geom) accepts only a subset of all aesthetics. Refer to the geom help pages to see what mappings each geom accepts. Aesthetic mappings are set with the aes() function.

TASK: We will be using the ggplot2 package for graphing. Fortunately, ggplot2 is nested within the tidyverse package (along with many others). Start by writing code to load the tidyverse library. *HINT:* see the end of assignment #1 if you forgot how to load a package.

In the code below, note the use of "‘ to start and end the code chunk. A common mistake is forgetting to end your code chunk! So if you're getting error messages when knitting, that might be the first thing to check for.

Further note that we can label the code chunk by including text following the R within the curly brackets. This label is only helpful to us in the RStudio environment, where we can use it to navigate easily among chunks. Otherwise chunks of code will just be given numbers, which can be hard to navigate among.

Finally, note that we include `message=FALSE` and `warning=FALSE` in the chunk options so that the extra info you get in the console when you lead a package doesn't show up on our word document. If you want, try removing these options and knitting again to see the difference.

```
library(tidyverse)
```

TASK: Write code below to set your theme to black and white and both your major AND minor gridlines to element_blank for all plots you'll be making today.
HINT: Check back to last week's assignment section 1.9 for setting themes for all plots.

```
theme_set(theme_bw())
theme_update(panel.grid.major = element_blank(),
             panel.grid.minor = element_blank())
```

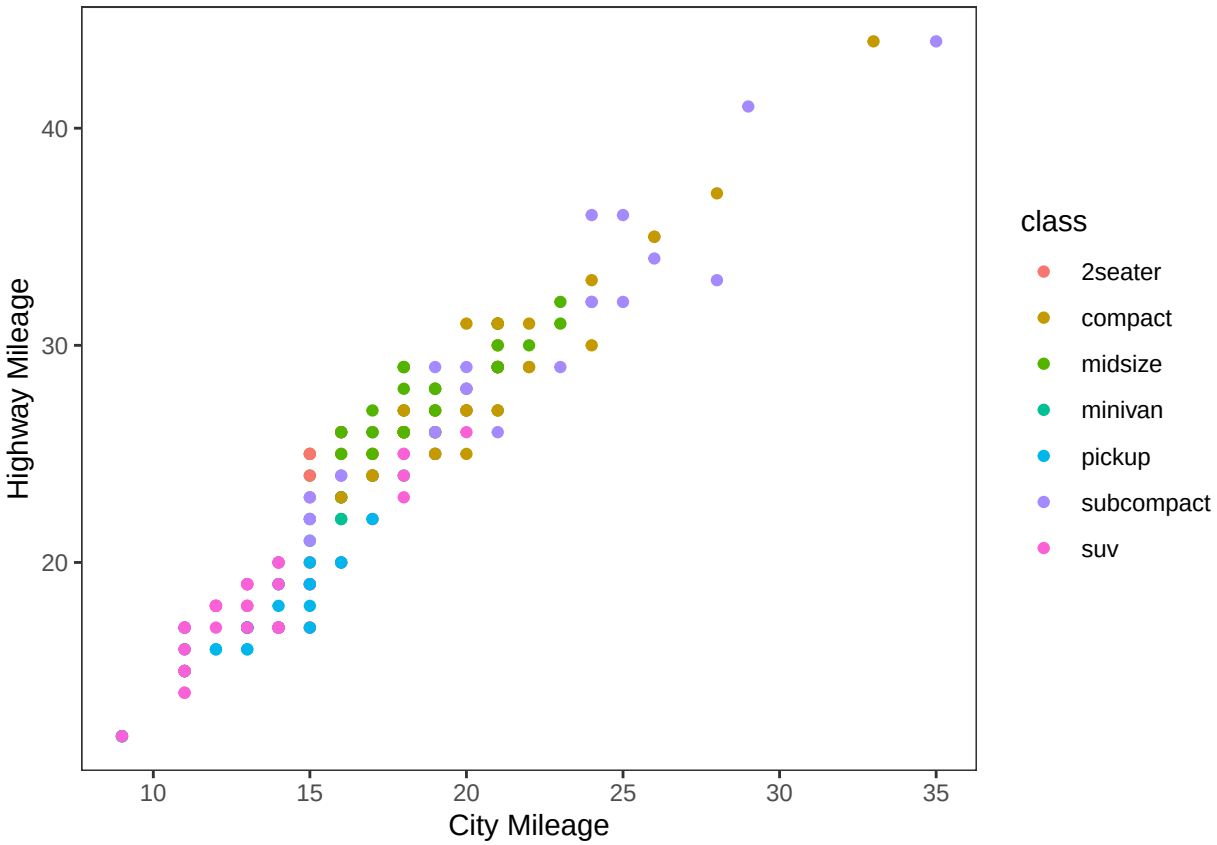
1.0 Correlation

We learned how to make correlations last week, but it never hurts to get more practice! Let's start by using a data set built into ggplot2. The mpg dataset looks at the gas efficiency of different cars. Run the following code to load the data: `data(mpg, package = "ggplot2")`
Note the use of a single ‘ in the text above, which tells Rmarkdown that this is a small bit of code that it should denote as such in the output file

```
data(mpg, package = "ggplot2")
```

TASK: Create a scatterplot (*i.e.*, a dot plot) to relate city and highway mileage. Color the points by the class of car (class column) and label the x and y axis to be more informative (City Mileage (MPG) vs Highway Mileage (MPG)).
HINT: Refer back to last week's assignment or the ggplot help resources if you forget how to make a scatterplot.

```
ggplot(data=mpg, aes(x=cty, y=hwy, color=class)) +
  geom_point() +
  xlab('City Mileage') + ylab('Highway Mileage')
```



Looks alright, but the graph may be hiding some information. . .

QUESTION: How many data points are in the mpg dataframe?
ANSWER: 234

QUESTION: Approximately how many dots are in the graph you just made? How does that compare to the number of observations in the dataframe?
ANSWER: 77–that's a lot LESS than the number of observations in the dataframe

Try another correlation-focused geom that addresses this problem by running the following code: `ggplot(data=mpg, aes(x=cty, y=hwy)) + geom_jitter()`

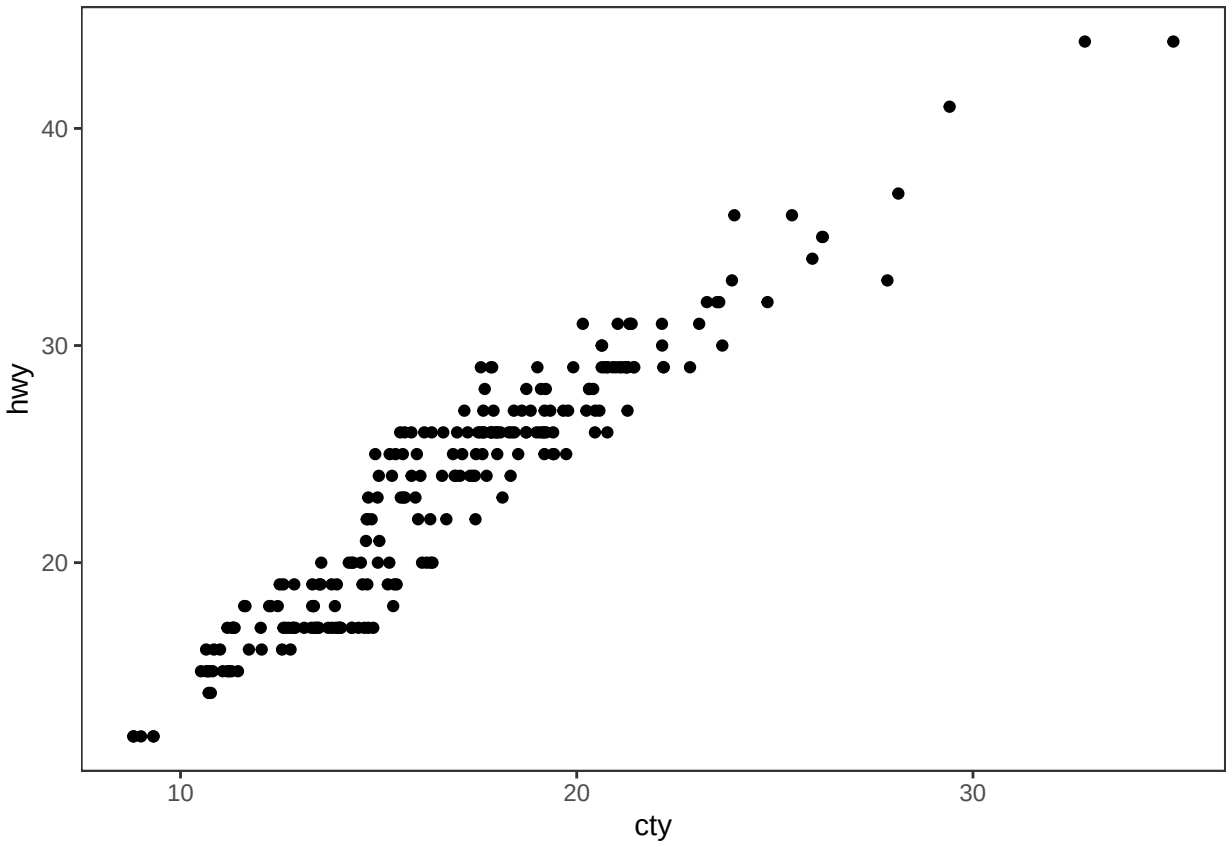
QUESTION: What happened when you created the plot with geom_jitter?
ANSWER: It shifted the points a bit so they weren't overlapping.

QUESTION: Run the code to create a plot using geom_jitter a second time. Then run it again and again. What happens each time? Why is this happening?
ANSWER: The points shift around each time you run it. Because they are randomly being shifted, so it changes exact position each time.

TASK: The default in geom_jitter is to jitter (or slightly move) the points away from each other in both the x and y directions. Check the help file for geom_jitter and write code below to make a graph where you

jitter points in only the x-dimension by 0.5.

```
ggplot(data=mpg, aes(x=cty, y=hwy)) +
  geom_jitter(width=0.5, height=0)
```



1.1 DETOUR! COLORS, COLORS, COLORS

Our world is so colorful, and our graphs can be too! But we want to be mindful of our color choices to not only be beautiful, but also informative and accessible.

TASK: Create a new dataframe that filters our mpg data down to only information for car classes compact, midsize, and suv using an `%in%` statement. Name your new dataframe mpgSubset.
HINT: Refer back to the Transform assignment if you want help with `%in%` (or try googling!)

```
mpgSubset <- mpg %>%
  filter(class %in% c('compact', 'midsize', 'suv'))
```

ggplot has lots of nice (and some not so nice) built-in color palettes that we can use to fill our bars with color. Try running the following code: `ggplot(data=mpgSubset, aes(x=cty, y=hwy, color=class)) + geom_jitter()`

The code above supplies us with ggplot's default color palette, which I find not so pleasing. Let's try to change this!

We can add a statement specifying the scale of colors we would prefer. One way to do this is to use other pre-set palettes from ggplot. Run the code below to ask ggplot to use the color brewer's palette called Set 1. `ggplot(data=mpgSubset, aes(x=cty, y=hwy, color=class)) + geom_jitter() + scale_color_brewer(palette="Set1")`

Alternatively, we could use the "RdBu" palette: `ggplot(data=mpgSubset, aes(x=cty, y=hwy, color=class)) + geom_jitter() + scale_color_brewer(palette="RdBu")`

QUESTION: Investigate the color brewer's range of palettes at this website: https://r-graph-gallery.com/38-rcolorbrewers-palettes.html What do you notice about the colors chosen from each of the palettes that we used above? (i.e., does it use the first three colors in the palette? The last three? Some other combination?)
ANSWER: It picks the first three for Set1 and somewhere on the left, the center, and the right of the palette for RdBu.

We could also pick out EXACTLY which colors we want for our figure. There are 4 main ways to specify colors in R:
1. by name
2. by number
3. by Hex code
4. with the rgb() function

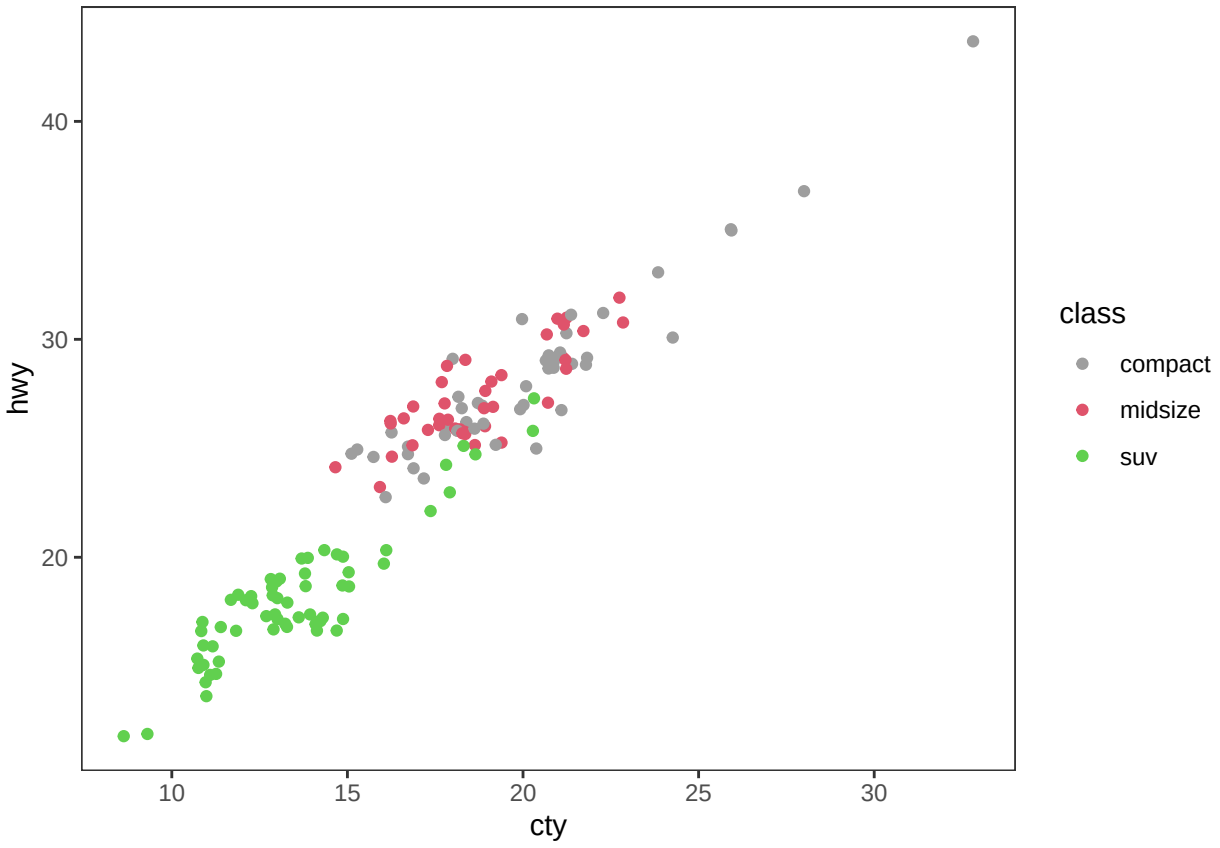
By Name: R allows you to call more than 600 colors by name! Run the following code to see a list: `colors()`

We can put in the colors we chose for each variable by adding the `scale_color_manual()` function to our plot. Try it out by running the following code: `ggplot(data=mpgSubset, aes(x=cty, y=hwy, color=class)) + geom_jitter() + scale_color_manual(values=c('midnightblue', 'cornflowerblue', 'tomato3'))`

You can also chose colors by number, which are assigned in R.

TASK: Copy and paste the code to make our scatterplot and replace the color names with three numbers of your choice (between 1 and 657). How does your new figure look?
HINT: remember to remove the quotation marks when calling numbers.

```
ggplot(data=mpgSubset, aes(x=cty, y=hwy, color=class)) +
  geom_jitter() +
  scale_color_manual(values=c(576, 378, 267))
```

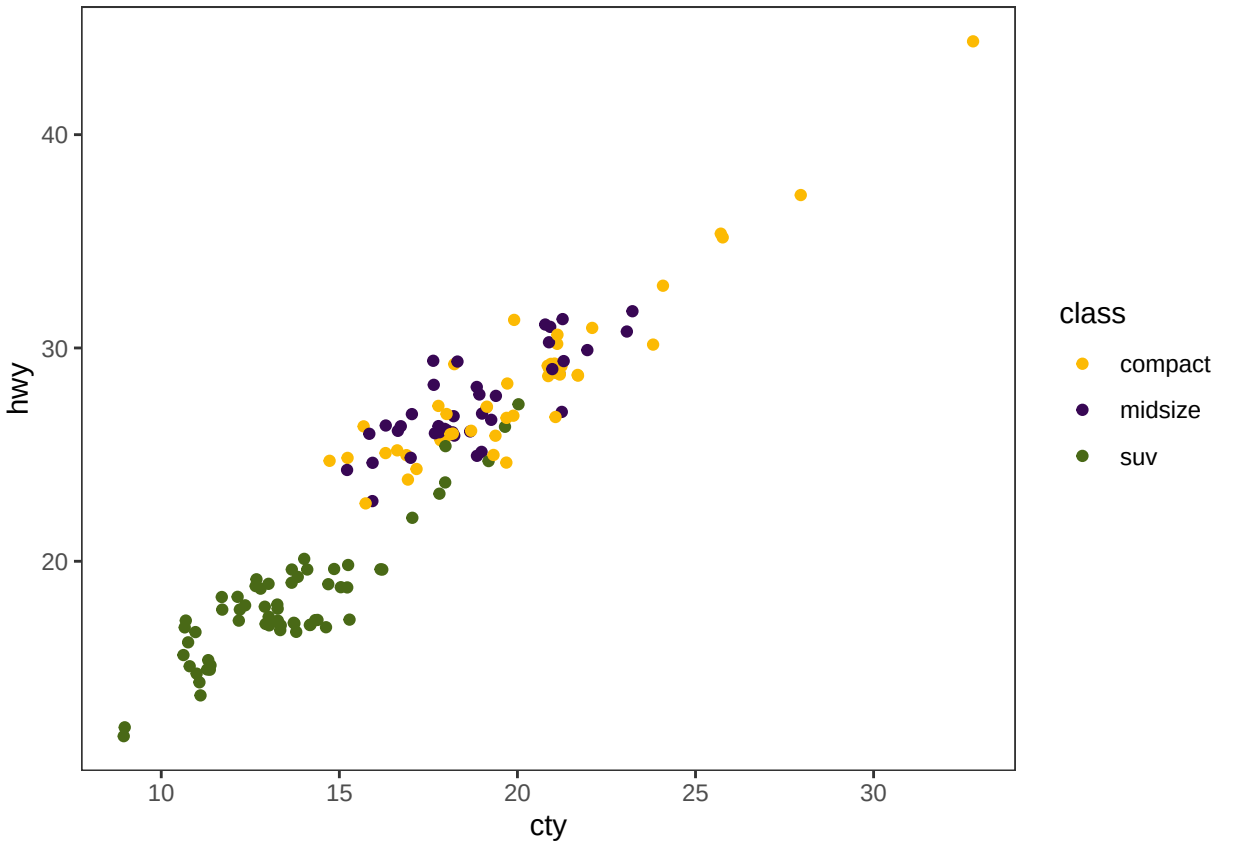


QUESTION: How do you think you could figure out which color name belongs to each color number?
HINT: Try creating a dataframe from color() by passing it into the as.data.frame() function.
`color <- as.data.frame(colors())`
ANSWER: The number is the position of the color name in the vector of colors.

You can also chose colors by Hex code. A Hex color code is a 6-symbol code made of up to three 2-symbol elements (6 symbols in length all together). Each of the 2-symbol elements represents a color value in the Red-Green-Blue (RGB) color scale from 0 to 255. Hex codes are written with a # in front of them. Let's try out calling our colors using hex code. `ggplot(data=mpgSubset, aes(x=cty, y=hwy, color=class)) + geom_jitter() + scale_color_manual(values=c('#191970', '#6495ED', '#FF6347'))`
There are two great things about hex codes. First, you can call ANY color you want with a hex code. That is, you're not limited to the 657 colors that are in ggplot. By specifying the color by hex code, you can pick literally anything!

TASK: Google the search term "color picker". This should bring up google's color picker (in addition to a billion other color picking websites). Use this color picker to generate the hex codes for three new colors of your choice. Then copy and paste the above code, replacing the hex codes with your color choices.

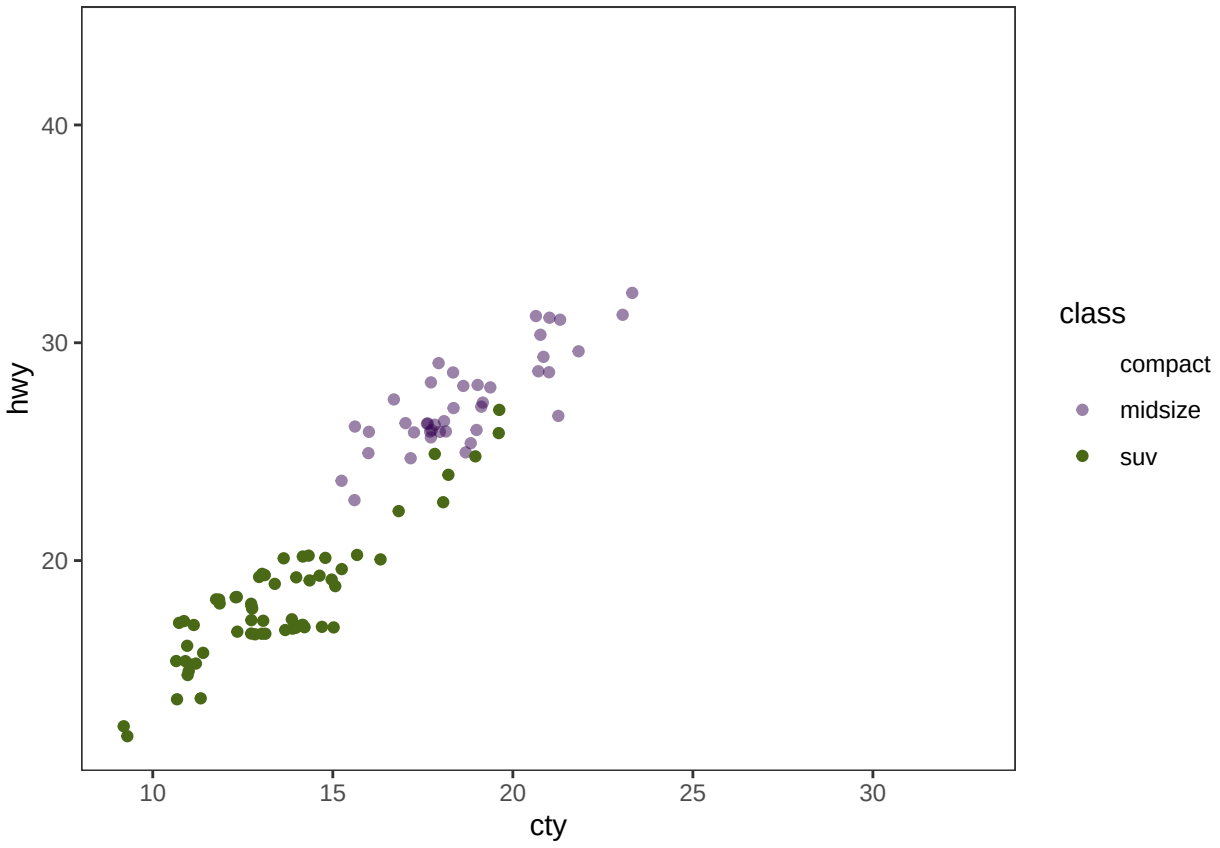
```
ggplot(data=mpgSubset, aes(x=cty, y=hwy, color=class)) +
  geom_jitter() +
  scale_color_manual(values=c('#FCBA03', '#380754', '#496916'))
```



The second great thing about hex codes is that you can control the transparency of your colors. Transparency is set in a hex code by adding two extra symbols to the end, which together make up the "alpha" element. Adding 00 to the end of your hex code will make your color completely transparent, while adding FF to the end will make it completely opaque. Any number/letters in-between will result in partial transparency. You can find a complete list of transparencies between 0-100% here: https://gist.github.com/lopspower/03fb1cc0ac9f32ef38f4

TASK: Try out transparency by copying and pasting your graph code below and adding the alpha element to the end of each of your hex codes to make your first color 0% transparent, your second color 50% transparent, and your third color 100% transparent.

```
ggplot(data=mpgSubset, aes(x=cty, y=hwy, color=class)) +
  geom_jitter() +
  scale_color_manual(values=c('#FCBA0300', '#38075480', '#496916FF'))
```

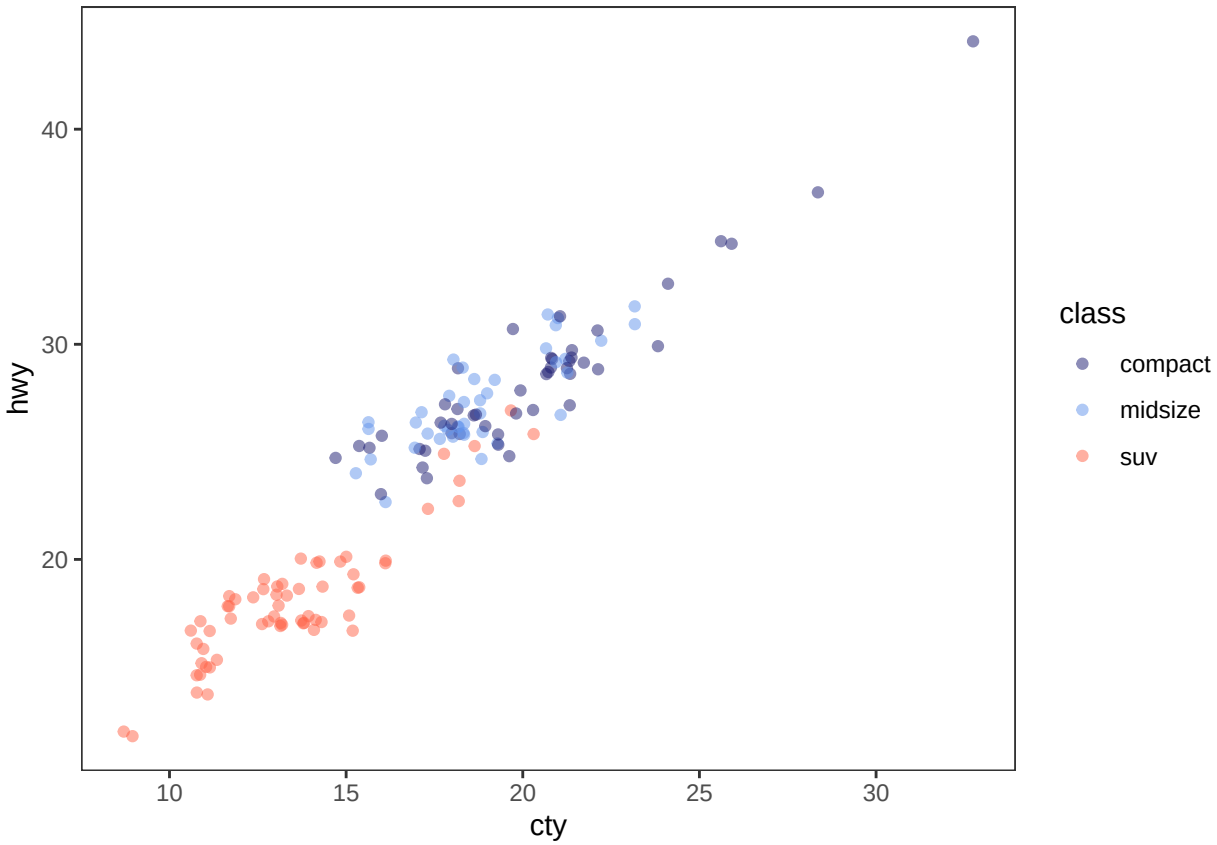


QUESTION: What happened to the point that you set to 100% transparent?
ANSWER: they disappear!

Finally, we can set our colors using the rgb() function. This operates very similarly to the hex code, where you can pick exactly the color and transparency you want. Try it out by running the following code: `ggplot(data=mpgSubset, aes(x=cty, y=hwy, color=class)) + geom_jitter() + scale_color_manual(values=c(rgb(.10, .10, .44, 1), rgb(.39, .58, .93, 1), rgb(1.0, .39, .28, 1)))`

TASK: Modify the above code to make all of your points 50% transparent.

```
ggplot(data=mpgSubset, aes(x=cty, y=hwy, color=class)) +
  geom_jitter() +
  scale_color_manual(values=c(rgb(.10, .10, .44, .5), rgb(.39, .58, .93, .5), rgb(1.0, .39, .28, .5)))
```



There are so many inventive and artistic people in the world who have expanded the offerings for colors in ggplot. Check out some notable ones listed below when choosing colors for this assignment.
-Color-blind friendly palettes
-Color palettes from vintage National Parks posters
-Color palettes based on Wes Anderson movies
-Color palettes from famous Dutch paintings