Introduction to `ggplot2`

Introduction

The goals of this lab are three-fold: first, we want to start familiarizing ourselves with the various types of *visual summaries* that are used in statistical analysis. This will involve introducing ourselves to the different plots and the types of variables they are intended to represent. Second, we will begin learning the basics of data visualization using R. This is a critical addition to our toolkit, as the ability to create informative plots will allow us to quickly investigate the qualities of single variables and the relationships between two or more of them. While there are a number of useful tools for creating data summaries, the most widely used is a collection of plots and functions included in the `ggplot2` package. We will be using `ggplot2` extensively during the semester, so the third goal of this lab is to provide a handy reference for later in the semester – if you are ever unsure on how to create various graphics, you are always welcome to refer back to the content in this lab.

As you might reasonably assume, data is a critical part of data visualization. To that end, we will be utilizing a number of different datasets for this lab. Before we introduce them, start by creating a new R Markdown document for Lab 2. Name this file `firstname_lastname_ggplot_lab` (using your actual name). Create an R code chunk at the top (Ctrl+Alt+I is the shortcut), paste in the following code, and run the block to load the two datasets into your environment (the pane at the top right).

Note: You can create comments in your R code chunk by typing a new line starting with the ‘#’ symbol.

```r
library(ggplot2)

# Make graphs look a bit nicer for ggplot2
theme_set(theme_bw())

# Load majors data
majors <- read.csv("https://collinn.github.io/data/majors.csv")

# Get mpg dataset from ggplot2 package and modify variables
mpg <- as.data.frame(mpg)
mpg$cyl <- factor(mpg$cyl)
mpg$year <- factor(mpg$year)
```

The first line use the `library()` function which is used to load the functions from the `ggplot2` package. You only need to do this once, but you will have to do it again every time you close out of R and open it again. The line after this, `theme_set()` , is optional as it simply adjusts the default theme of our plots to make them more aesthetically pleasing.

Following this, we read in our first dataset, just like we did in Lab 1 with the `read.csv()` function. This dataset, `majors` contains salary data for various college majors based on the results of a 2022 American Community Survey. You should spend a minute or two familiarizing yourself with the variables in this dataset in the environment pane.

The second dataset, `mpg` , comes as a part of the `ggplot2` package so there is nothing else we need to load it into R. This dataset contains a subset of fuel economy data that the EPA collected for various car models in 1999 and 2008. We won't worry about the details of the three lines we use to modify it; for now, it is sufficient to say that we have changed it to make it more amenable to our purposes. You can learn more about the `mpg` dataset, along with a full description of each of the variables, by typing `?mpg` into your console.

Grammer of Graphics

The `ggplot2` package is intended to follow the "grammar of graphics", a coherent philosophy of independent components that can be added together to build a plot. Every ggplot is made up of three components:

1. Data – the information we wish to visually represent
2. Aesthetics – A mapping between the data and visual properties, i.e., xy axes, color, etc
3. Layers – How the observations in our data our rendered, i.e., scatter plot, box plot, etc.,

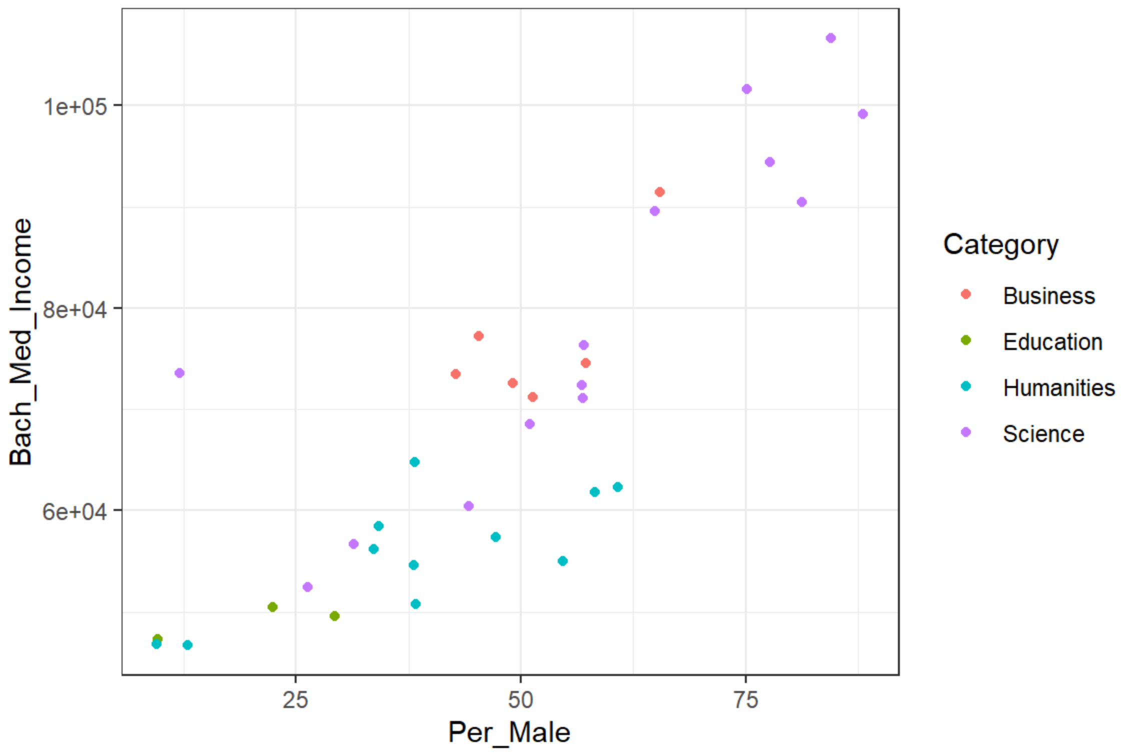
We will walk through each of these components in more detail. As motivation, though, consider how the relationship of three variables from our `majors` dataset, percentage male, median bachelor degree income, and category of major:

```r
## The function head() shows us the first 6 rows of a dataset
head(majors[, c("Per_Male", "Bach_Med_Income", "Category")])
```

```
##   Per_Male Bach_Med_Income Category
## 1     75.1          101600  Science
## 2     81.3           90490  Science
## 3     77.7           94370  Science
## 4     84.5          106600  Science
## 5     88.1           99180  Science
## 6     57.0           76350  Science
```

can be quickly summarized with a plot:

```
## Code to generate a ggplot
ggplot(majors, aes(x = Per_Male, y = Bach_Med_Income, color = Category)) +
  geom_point()
```

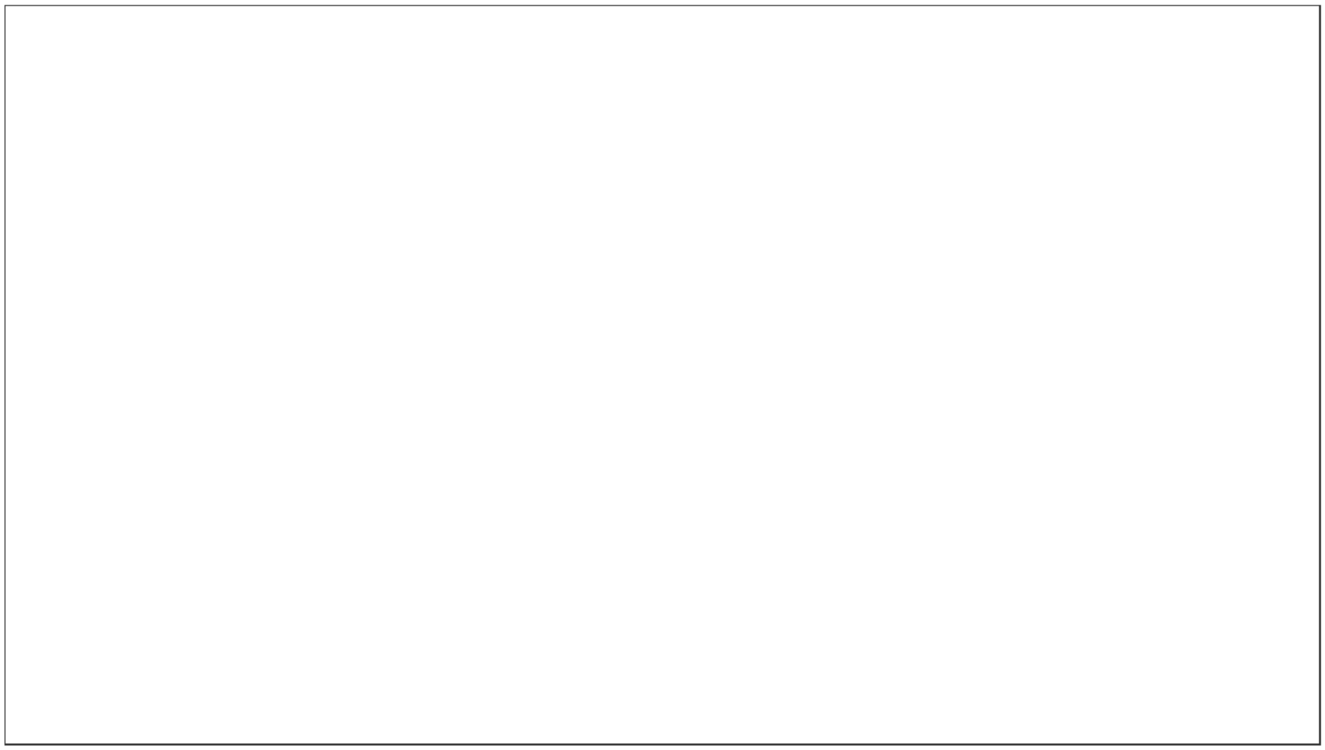


Question 0 In a few short sentences, describe what you see here. How are the different variables represented in this plot? Are there any relationships that stand out to you?

Aesthetics

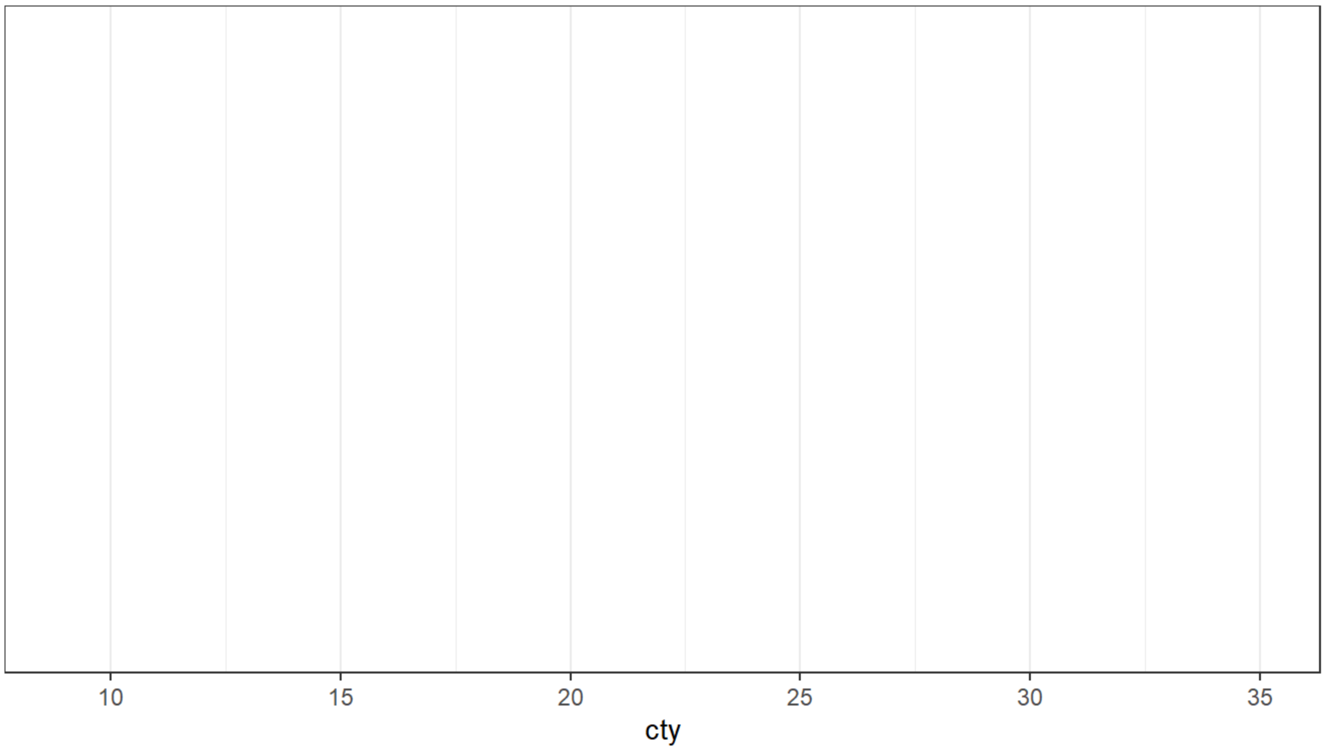
The aesthetics of a `ggplot2` plot (which we simply call a “ggplot”) establish the most general properties relating our collected data to the visual representation we wish to create. Specifically, the aesthetics create a link *from the variables in the data* to visual properties of the graph. This will be an important point to keep in the back of our minds as our aesthetics becoming increasingly more detailed. The very first argument to the `ggplot()` function is simply the dataset from which we intend to generate our graphics. Without specifying any aesthetics, however, it will simply render a blank plot

```
## Calling the ggplot function with no aesthetics
ggplot(mpg)
```



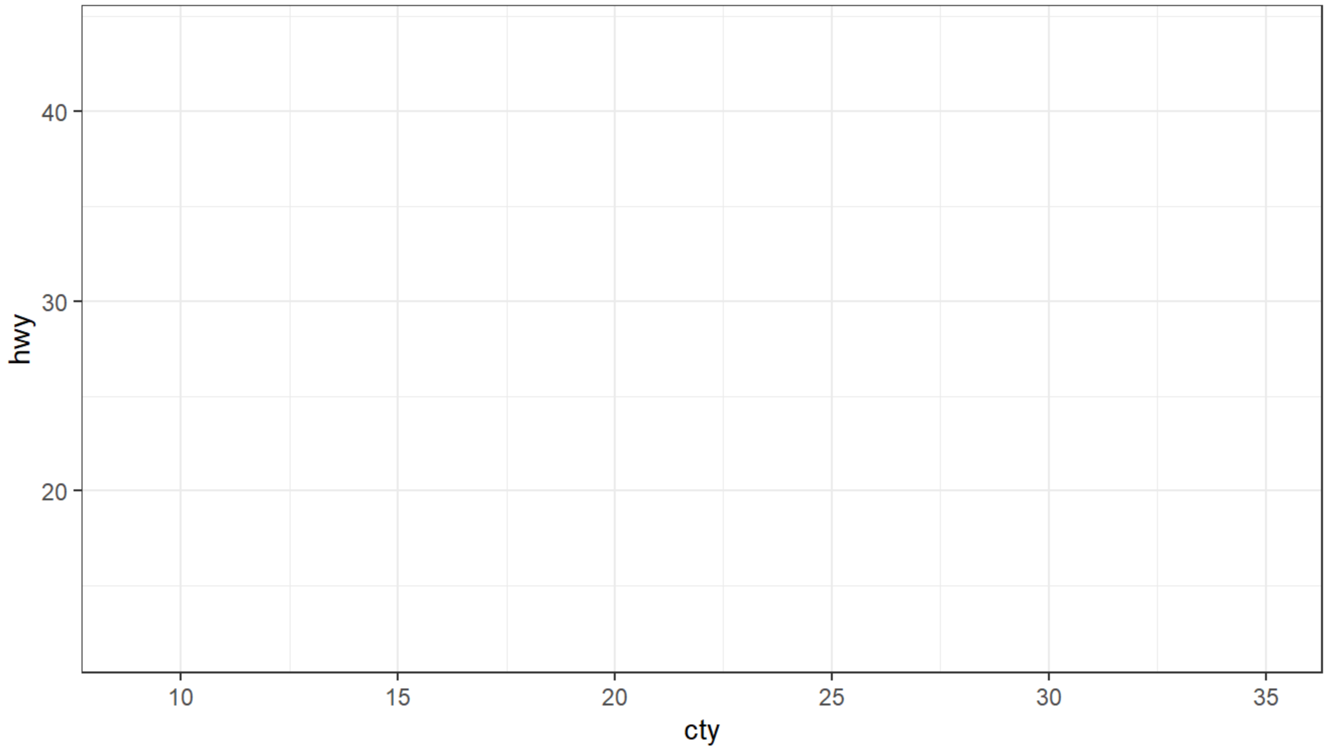
We create a mapping from the variables in our dataset to the visual properties of the plot using the `aes()` function (short for aesthetics), which is the second argument to `ggplot()`. For example, to specify that we want the variable `cty` (city mpg) to be represented on the x-axis, we do by specifying that the `x` aesthetic is equal to `cty`

```
ggplot(mpg, aes(x = cty))
```



We now see that our blank box has been updated to include an x-axis, the scale of which is automatically determined by the `cty` variable. We can then add another variable to the y-axis in the exact same way. Here, we include `hwy`, or highway mpg

```
ggplot(mpg, aes(x = cty, y = hwy))
```



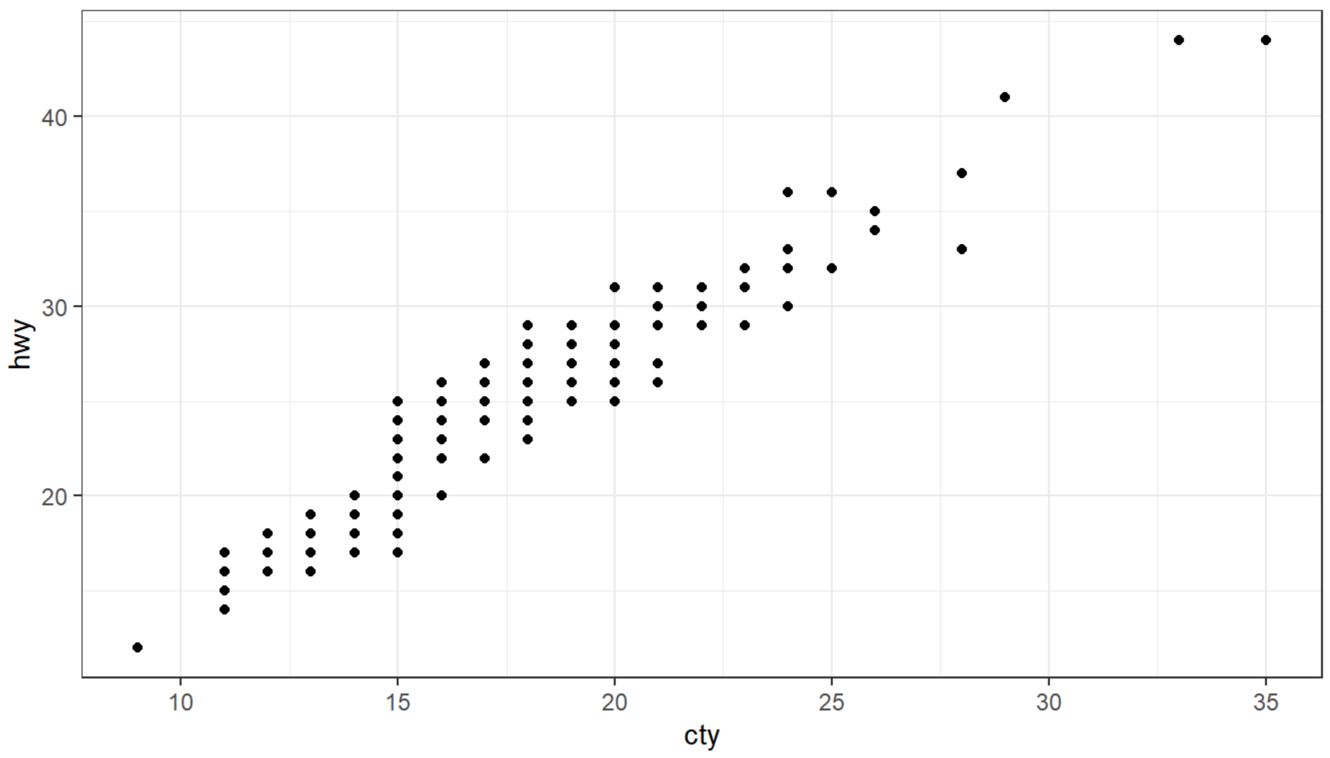
Every ggplot will begin this way: we have the name of the function (`ggplot()`), the data that is being used (`mpg`), and a set of variables that we wish to include in the aesthetics (`aes(x = cty, y = hwy)`). Recall that the names of the variables in a dataset can be found in the Environment Panel. Of course, not every plot will require both axes – a histogram, for example, only needs an x-axis.

There will be more than we can do with aesthetics soon, but for now let's consider the next component, layers. Layers will dictate exactly how the observations in our dataset are rendered visually.

Layers

Once the data has been specified, along with the aesthetics, we are ready to add *layers*. Literally, as it turns out, as layers are added to an existing ggplot with `+`. Layers in ggplot are often referred to as "geoms" (for geometries). There are a number of geoms that we may be interested in including, but for now we will add points to our plot with the layer `geom_point()`, giving us the following *scatterplot*:

```
ggplot(mpg, aes(x = cty, y = hwy)) +
  geom_point()
```



Note that this is the same plot that we had created above, except now each of the observations in our dataset is now included as a point.

Most (but not all) layers that we might add in ggplot begin with `geom_` to help indicate that they can be thought of as the actual, geometric elements (lines, points, etc.,) that you will see on the plot. In addition to `geom_point()` , there are a few others that are worth being aware of:

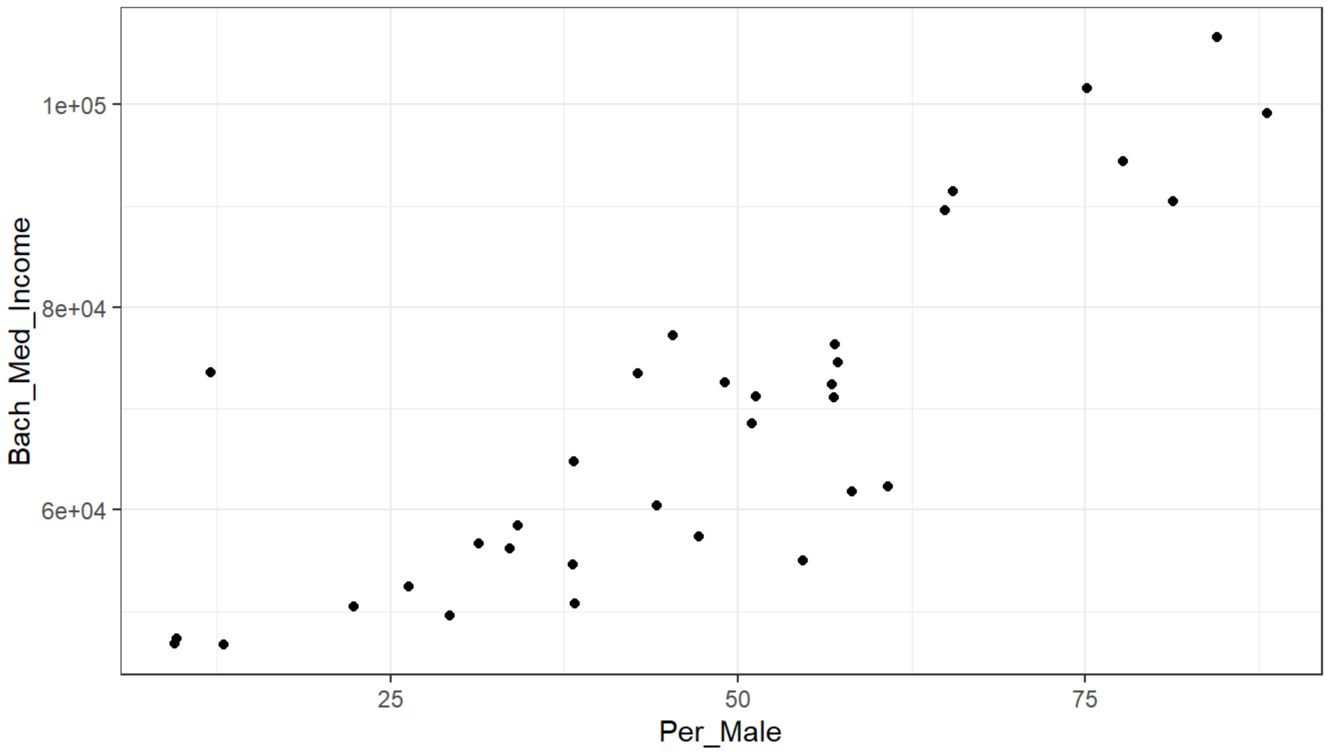
- `geom_point` - scatter plots
- `geom_jitter` – scatter plot with "jittered" coordinates
- `geom_bar` - bar graph for categorical data
- `geom_smooth` - smoother for data
- `geom_boxplot` - box plot
- `geom_histogram` - histograms
- `geom_violin` - violin plot

While there are many to chose from, we will primarily be using only a handful, so don't worry about knowing them all now.

Additional Aesthetics

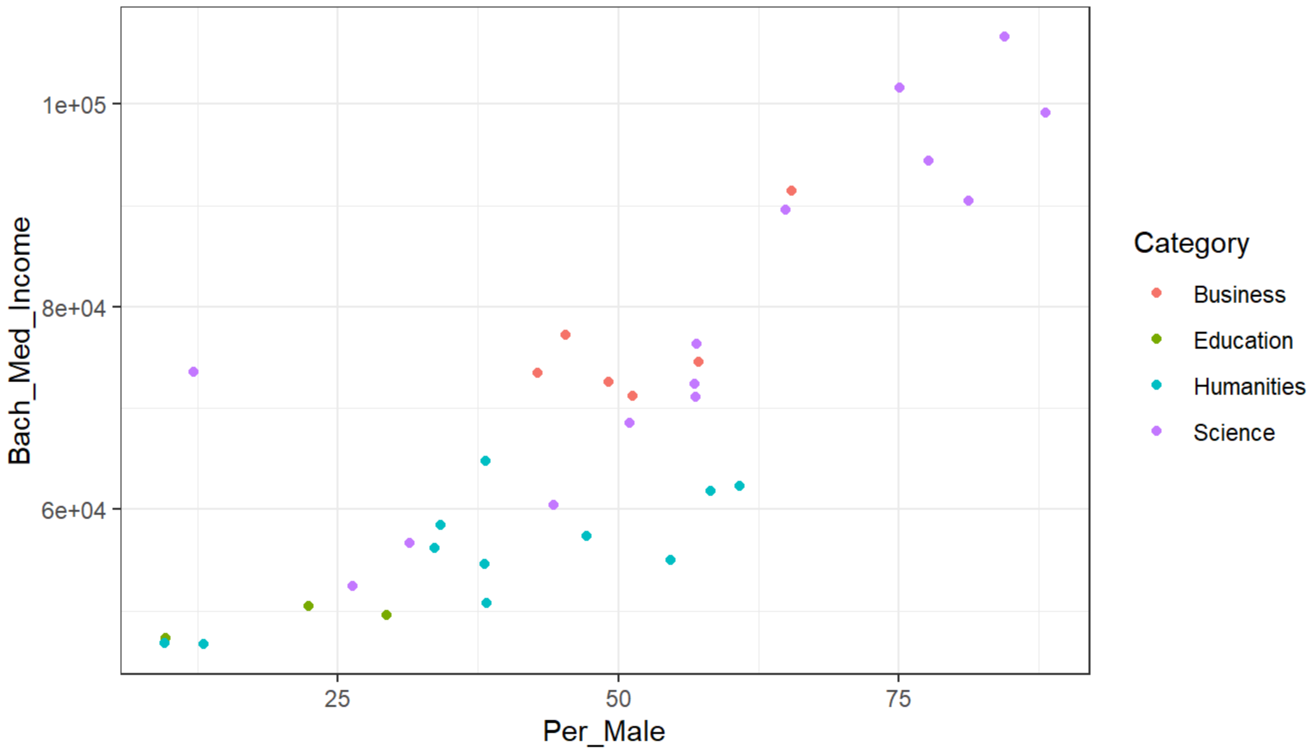
While the x and y axes are the most common aesthetics, we can take advantage of the mapping between data and other visual properties. Here, we consider the `majors` dataset as an example, plotting the relationship between percentage male in a field against the median income for a bachelor's degree:

```
ggplot(majors, aes(Per_Male, Bach_Med_Income)) +
  geom_point()
```



By default, the first and second aesthetics are always `x` and `y`, so we do not need to specify them directly. In addition to these, we can also include a `color` aesthetic, which allows us to take anotherh variable, in this case, `Category`, and represent it in the plot as a distinct color:

```
## Adding color = Category
ggplot(majors, aes(Per_Male, Bach_Med_Income, color = Category)) +
  geom_point()
```



This is an example of a *multivariate* plot; it allows us to succinctly describe the relationship between 3 or more variables.

Question 1: Looking at the multivariate plot above, what additional information is gleaned by including `Category` as a color as opposed to what we saw when we only had the x- and y-axes? (Similar to Question 0, but here pay attention specifically to the difference between the two.)

This rest of this lab will be divided into three sections: creating univariate plots, creating bivariate plots, and then exploring multivariate plots. At the end of the lab is a (mostly) comprehensive collection of examples. These examples include different *arguments* that we can add to a ggplot to change, for example, how the bar charts are organized or to modify the number of bins in a histogram.

Types of Plots

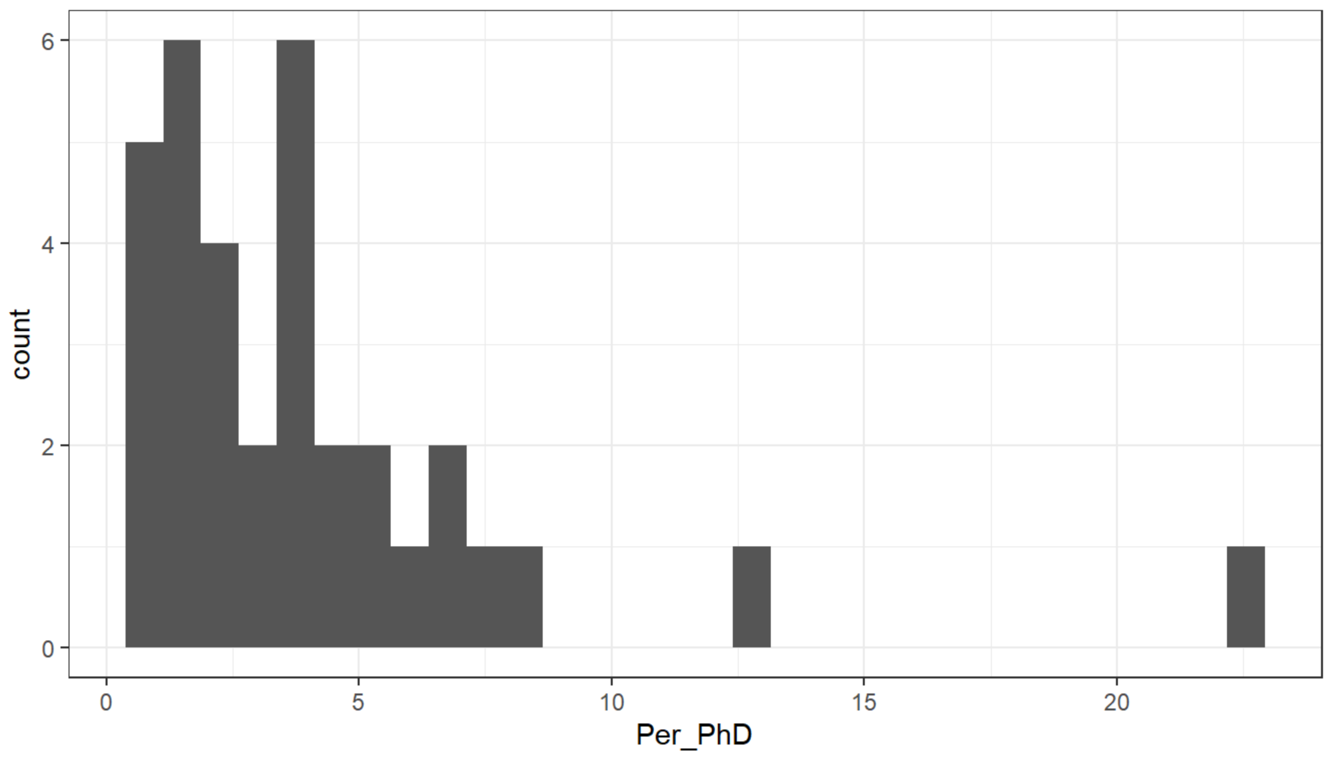
Univariate Plots

Univariate plots are those that provide a visual summary of a single variable. Variables are either quantitative, dealing with numerical quantities and typically taking any number within a range of values, or they are quantitative/categorical, describing the number of objects that have a particular property. As will typically be the case, the types of variables we are working with will dictate the appropriate types of plots to use.

Quantitative

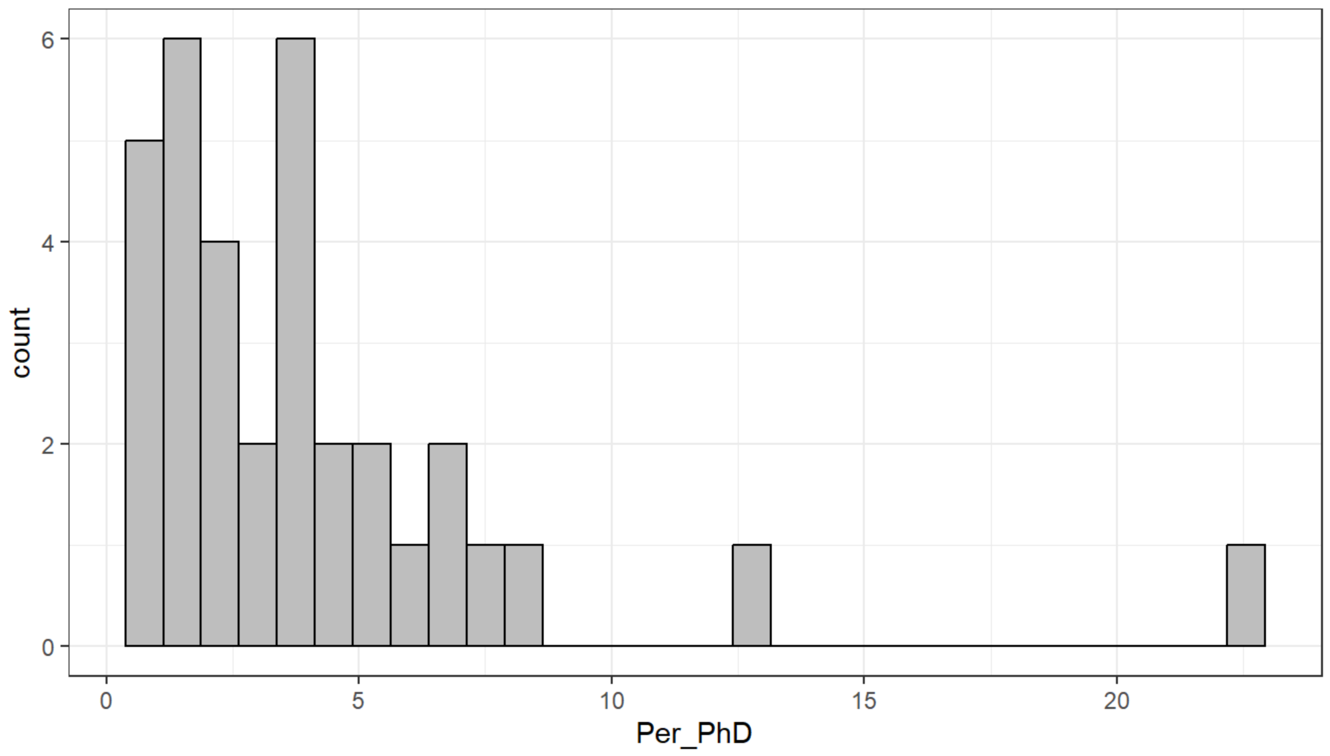
For univariate quantitative variables, we will typically be interested in *histograms*, which show the relative frequency of numerical values over a particular domain. Consider the `majors` dataset, which contains a variable indicating the percentage within each college major that has obtained a PhD. As this is only a single variable, we only need one aesthetic, and because we wish to generate a histogram, we will use the layer/geom associated with histograms:

```
ggplot(majors, aes(x = Per_PhD)) +
  geom_histogram()
```



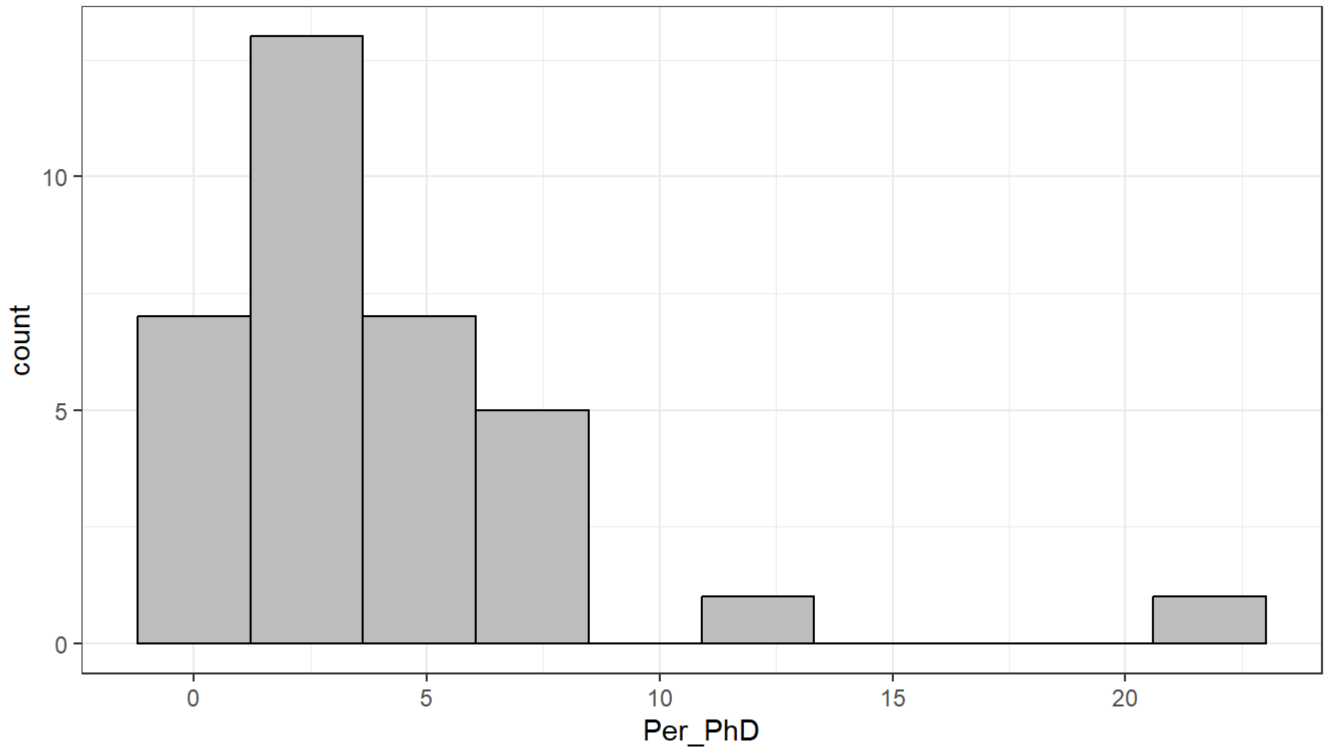
Typically with layers, we can add additional arguments that change their behavior. Here, I replicate the exact same histogram, this time modifying the fill and color variables to make it easier to see:

```
ggplot(majors, aes(x = Per_PhD)) +
  geom_histogram(color = 'black', fill = 'gray')
```



Histograms work through a process called *binning* in that it breaks a range of numbers (here, from 0 to about 25) into a number of equally sized “bins”. For each of our observations that fall into a bin, its associated bar increases by one. While there are few hard and fast rules associated with binning, it is helpful to try and modify the number of total bins to give us a better pictures of the distribution of the data. We do this with the `bins` argument in `geom_histogram()`

```
ggplot(majors, aes(x = Per_PhD)) +
  geom_histogram(bins = 10, color = 'black', fill = 'gray')
```

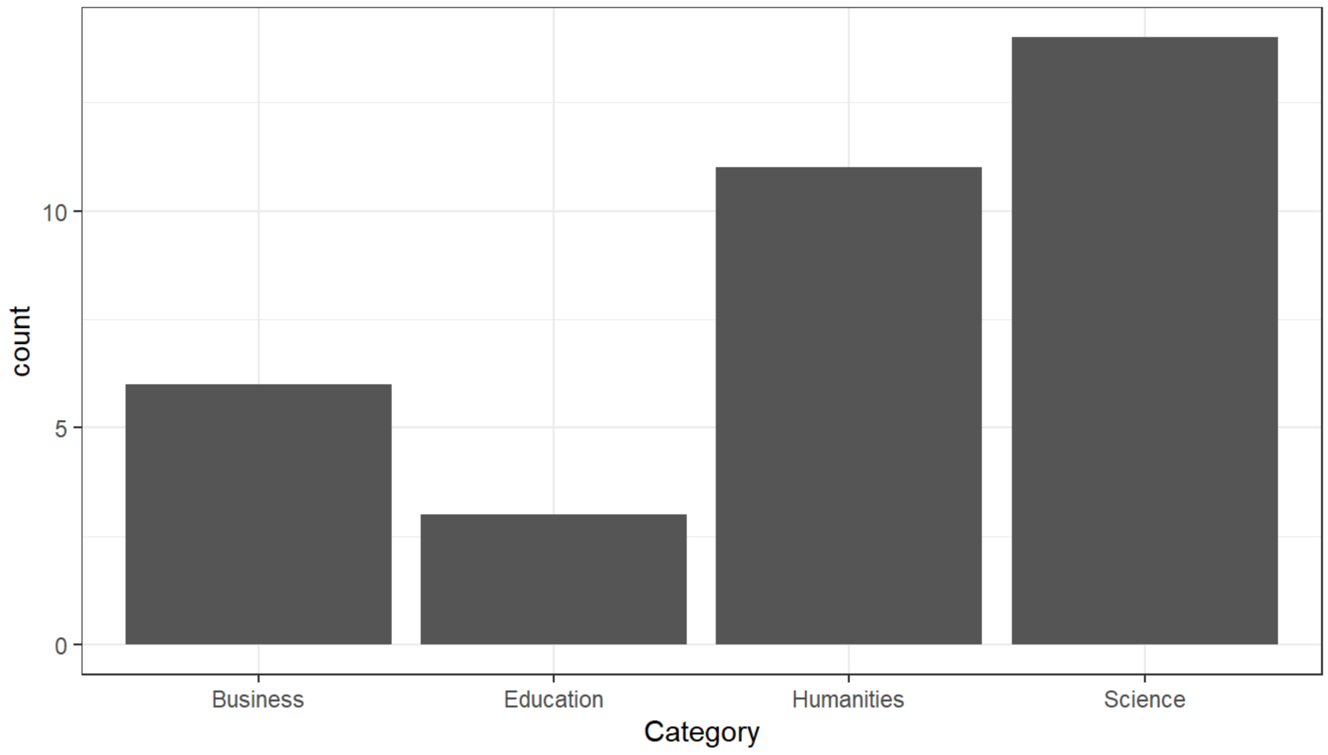


Using the example above, play around with various numbers. Between 10, 15, and 20, which of these do you think best illustrates the data we have?

Categorical

For categorical data, we are typically interested in either *counts* or *proportions*. These are represented with *bar plots*. Bar plots ostensibly are similar to histograms in that they have the appearance of vertical bars (read: these are often confused), but it is important to recognize that they have different purposes. Similar to histograms, however, they are added with an additional geom, `geom_bar()`

```
ggplot(majors, aes(x = Category)) +
  geom_bar()
```



It is also possible to use the y aesthetic instead. Try recreating the plot above assigning Category to y instead of x. How does your plot change?

Bivariate Plots

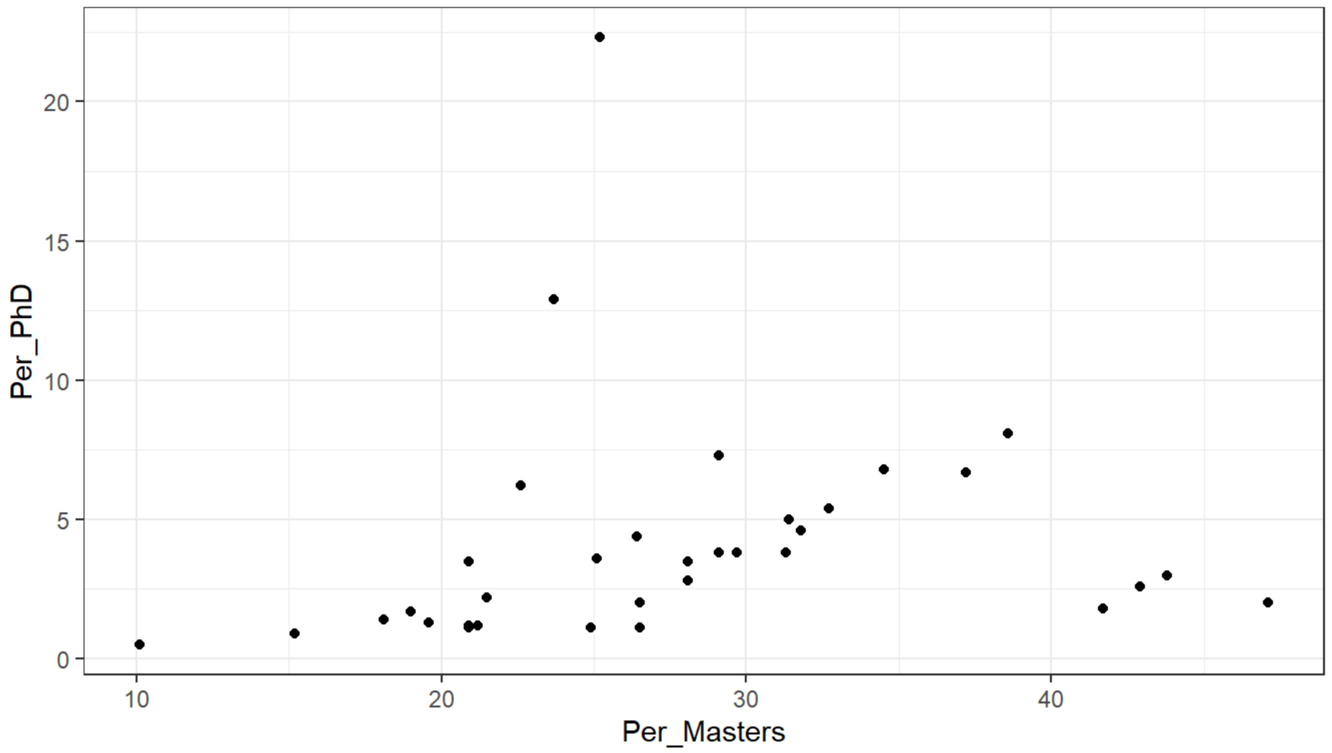
Bivariate plots, as the name suggests, are plots presenting the relationship with two variables. As before, the nature of these variables will dictate the type of plot used. For two variables, we have the following possible arrangements:

1. Two quantitative
2. One quantitative, one categorical
3. Two categorical

Two Quantitative

The relationship between two quantitative variables is described with a *scatterplot* (Is it one word or two? Nobody knows). We have seen this already in this lab, and it is added with the layer `geom_point()`

```
ggplot(majors, aes(Per_Masters, Per_PhD)) +
  geom_point()
```

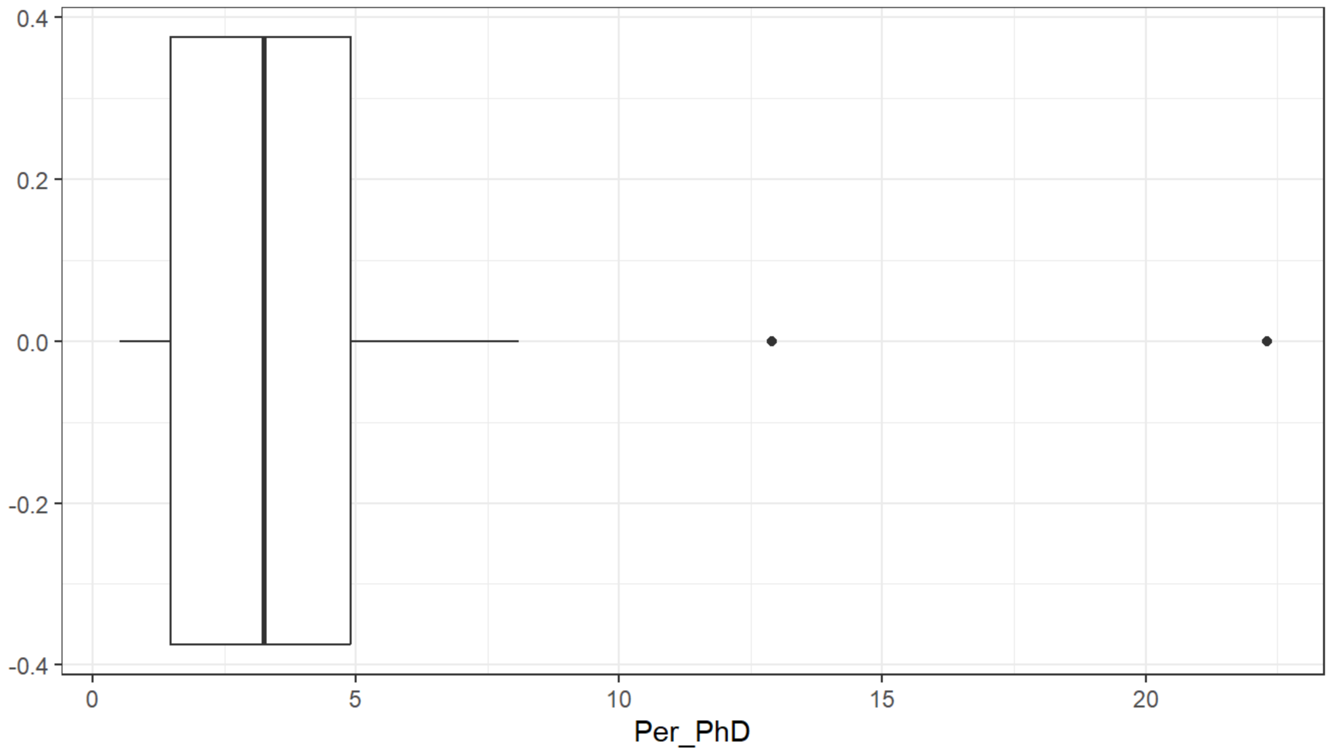


We will have much to say on scatterplots later in the course, but for now it is sufficient to know how to make one.

One Quantiative, One Categorical

In the case with a single quantitative variable, we were primarily interested in seeing the *distribution* of a single value within our dataset. This idea extends naturally with the addition of a categorical variable, where now we are interested in seeing the distribution of a quantitative variable *within each category* of a categorical variable. This is done with a *boxplot* (boxplot? box plot? Again – nobody knows). A boxplot extends the idea of a numerical, five-figure summary (which we will cover in class) into something visual, giving indications of the median, quantiles, and range of a quantitative variable. The typically are not used with a single quantitative variable alone, but there is nothing stopping us from doing so. They are created with the layer `geom_boxplot()`.

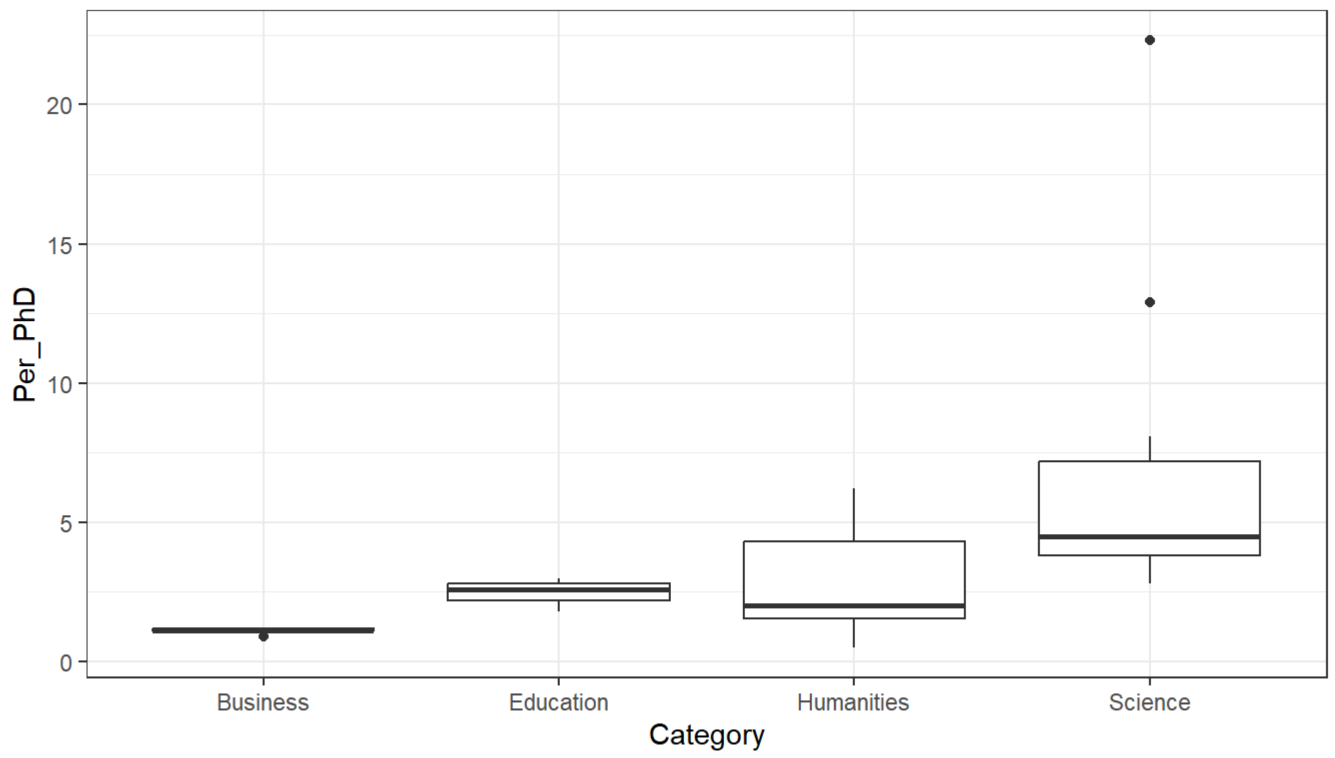
```
ggplot(majors, aes(Per_PhD)) +
  geom_boxplot()
```



The large box indicates where the middle fifty percent of the data lies, with the horizontal line inside of the box representing the median. Additionally, the “whiskers” give us a sense of the range. The individual points represent *outliers*.

More useful would be to see this distribution across a range of variables. As this involves mapping an additional variable to our plot, we indicate this addition by adding an additional aesthetic. It can be either x or y, depending on how we would like our plot oriented

```
ggplot(majors, aes(Per_PhD, x = Category)) +
  geom_boxplot()
```



From this, we see that there is a significant difference in the distribution of percentage PhDs across each of the categories. We also see, with this additional granularity, that the outliers from before fall within the "Science" category.

Two Categorical

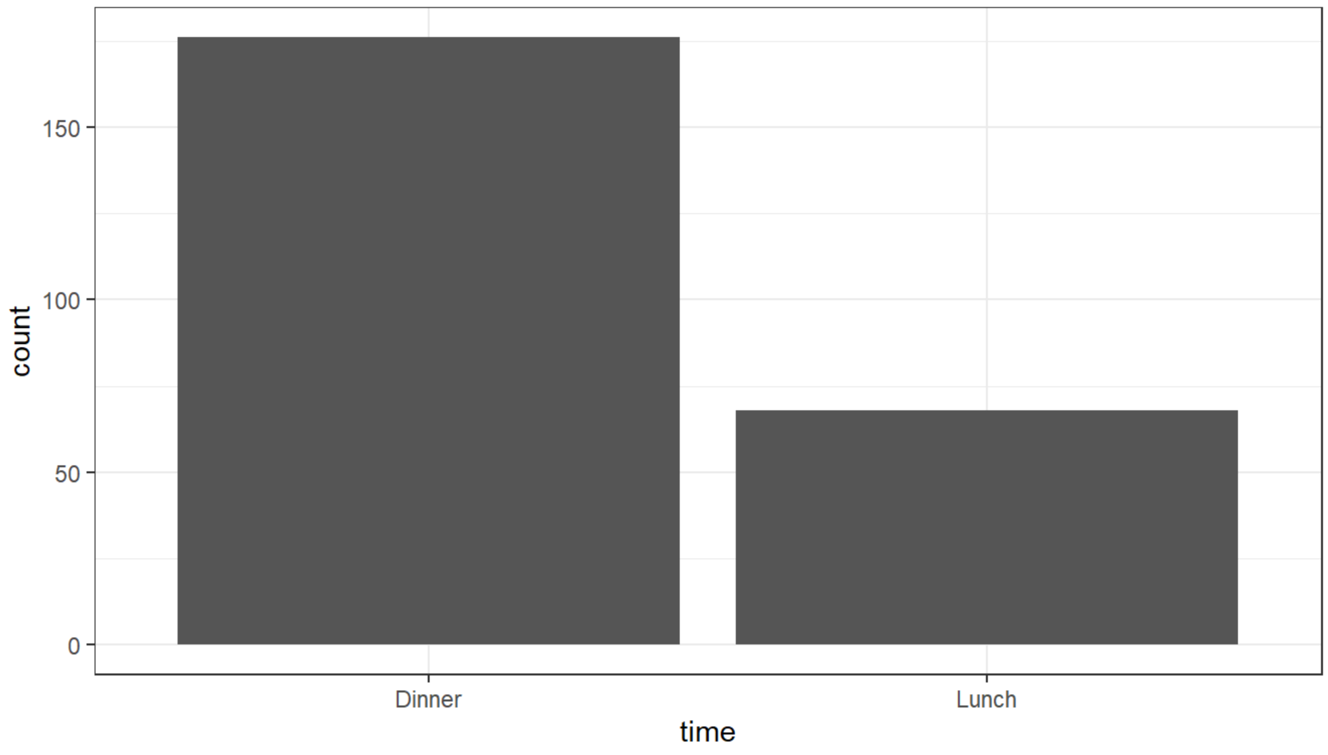
To showcase situations with two categorical variables, we will use the `tips` dataset below, containing data on bill information for various meals at a restaurant. It may be worthwhile to take a minute or two to investigate the variables included in this data.

```
tips <- read.csv("https://collinn.github.io/data/tips.csv")
```

The case for presenting two categorical variables in a plot is slightly more involved than were either of the other two bivariate scenarios, largely on account of the different types of relationships we may be interested in displaying. We don't need to worry about this in too much detail, as it will be covered in far greater depth later in the course. For now, we will simply investigate the two primary ways we may be interested in viewing these relationships.

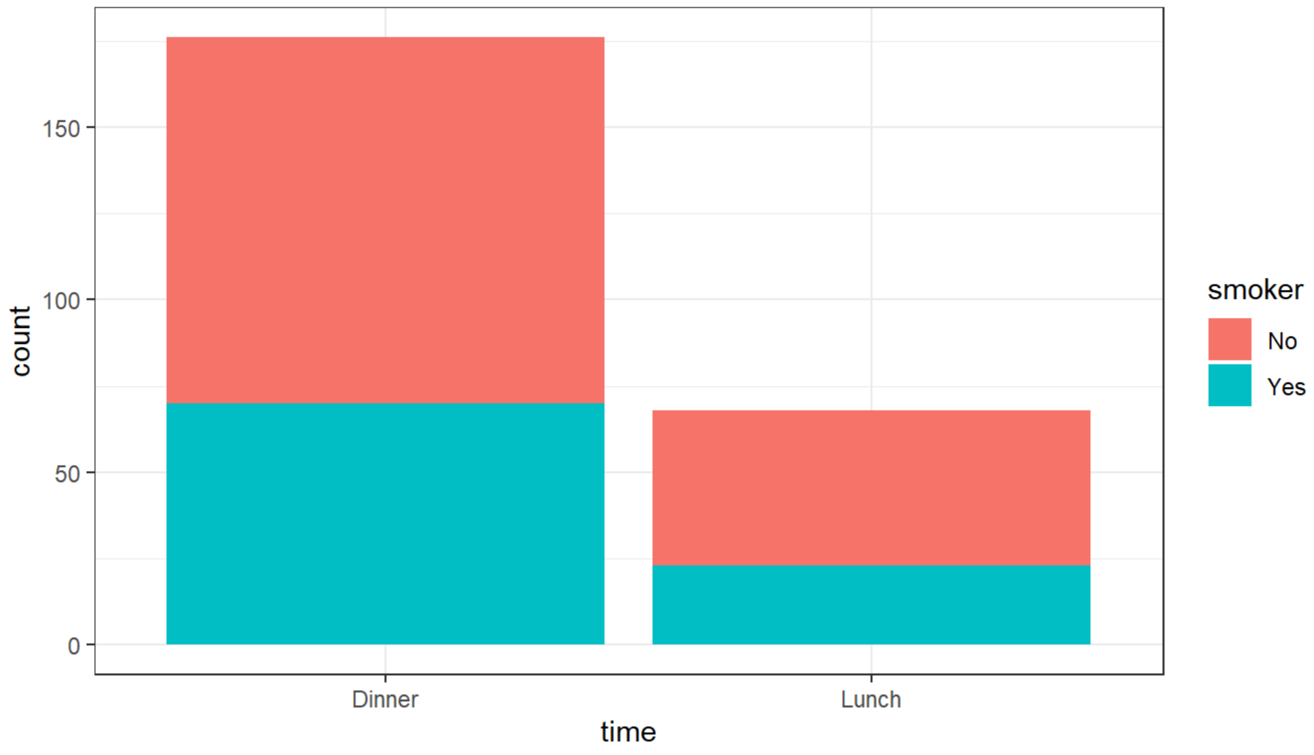
First, let's look at a bar chart showing the frequency of meals that were served either for Lunch or Dinner with the variable `time`

```
ggplot(tips, aes(time)) + geom_bar()
```



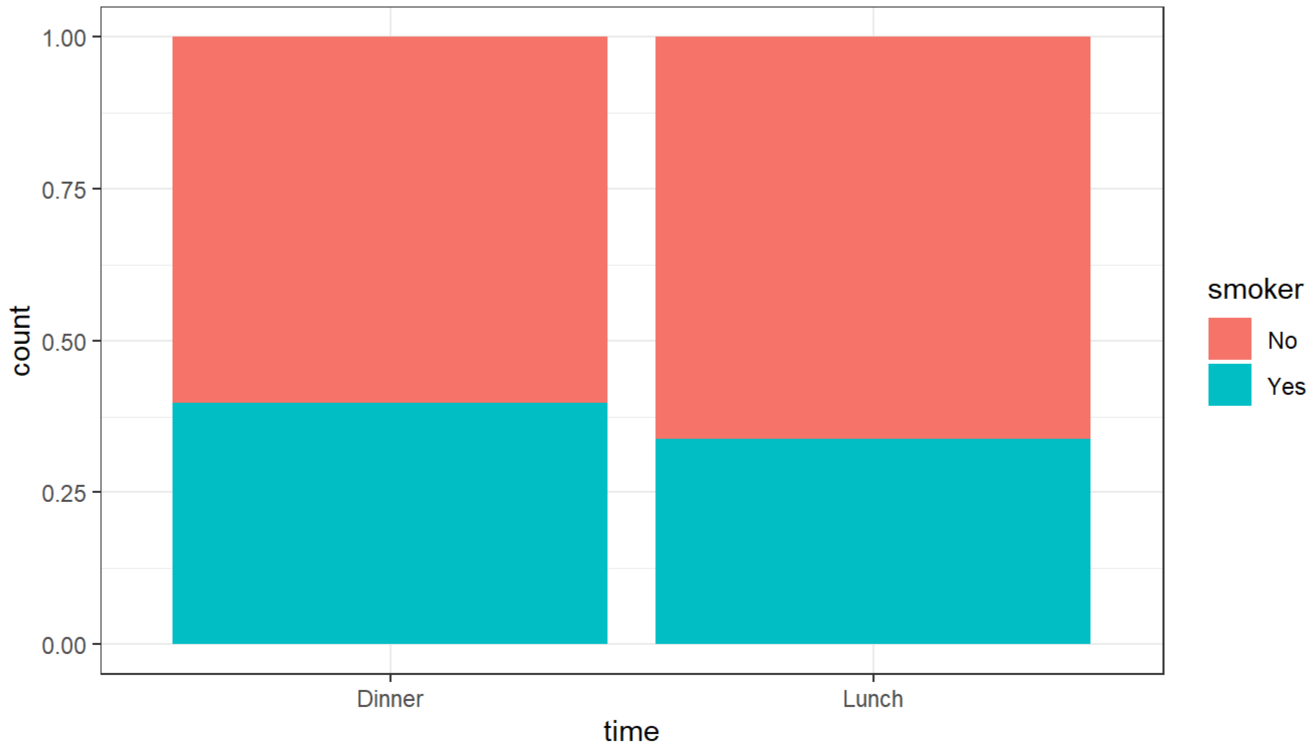
Simple enough. If we wish to further break this down by an additional categorical variable, say, by smoking status, we can do so by adding an additional aesthetic `fill` (as in, fill this box with color) and assigning it the new variable:

```
ggplot(tips, aes(time, fill = smoker)) +
  geom_bar()
```



This is great for showing us absolute counts (more people had dinner than lunch), but it is difficult to assess how the *proportions* differ within each group. Similar to the case with histograms, we can modify the layer itself by passing in additional arguments. In this case, we can use the `position` argument with the value “fill” to create a bar chart that shows us proportions

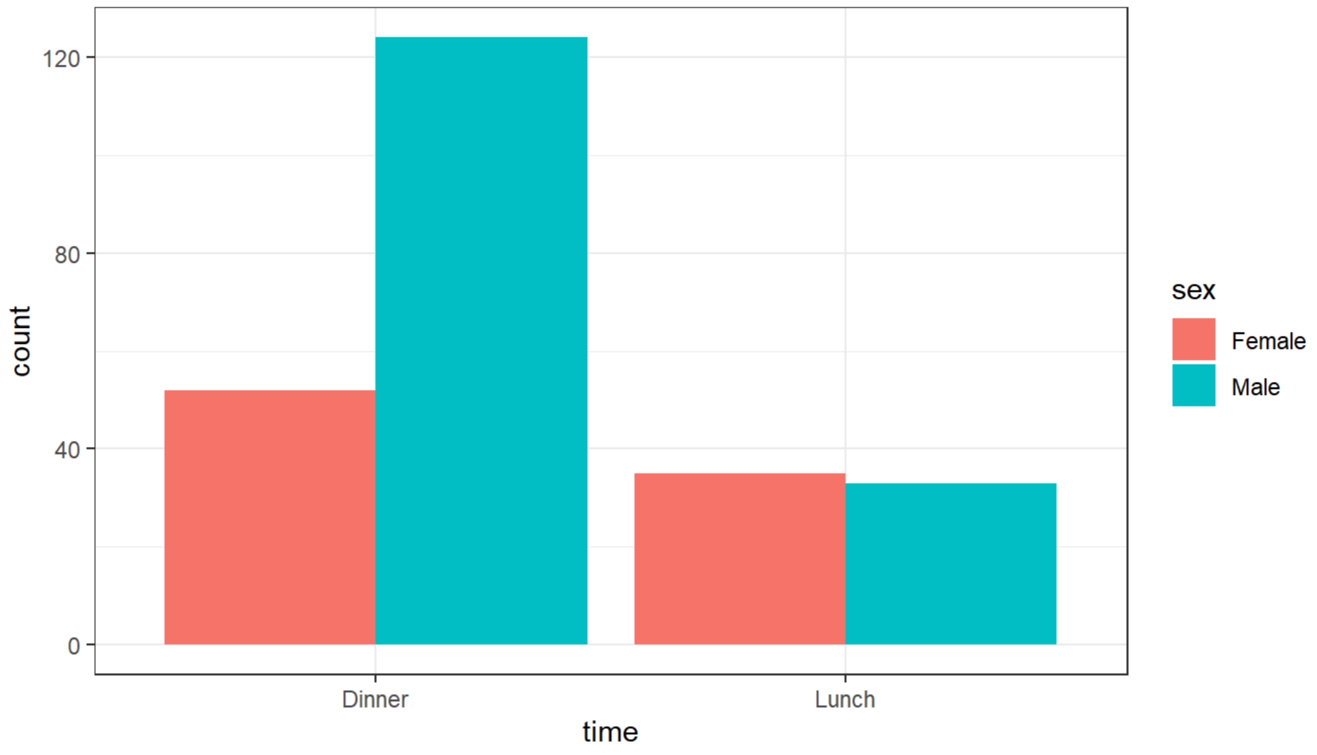
```
ggplot(tips, aes(time, fill = smoker)) +
  geom_bar(position = "fill")
```



Here, we can now see that a slightly higher proportion of dinner diners were smokers, compared with those at lunch. As the number of unique values within a category increases, we will see that which variable is mapped to which aesthetic plays a large role in what relationships are being communicated.

The bar charts that we have seen until now are called “stacked” and “proportional” bar charts, respectively. Often however, we would prefer to see each of the partitioned categories on its own to compare, and we can do this by modifying the position argument in `geom_bar()` :

```
ggplot(tips, aes(time, fill = sex)) +
  geom_bar(position = "dodge")
```



There are a number of variations on each of these plots that can be employed, some of which are included in the “Examples” section below. We will look at these in more detail a little later in the course.

Multivariate Plots

Finally we come to multivariate plots which visually represent three or more variables in a dataset. While there are a wide variety of ways to do this (including with the use of size or shape), we will primarily limit ourselves to the use of two: color and faceting.

For our multivariate plots, we will be using a subset dataset containing attributes and outcomes for all primarily undergrad institutions in the US with at least 400 full-time students for the year 2019. Our particular subset will consists of college in Iowa and a few neighboring states

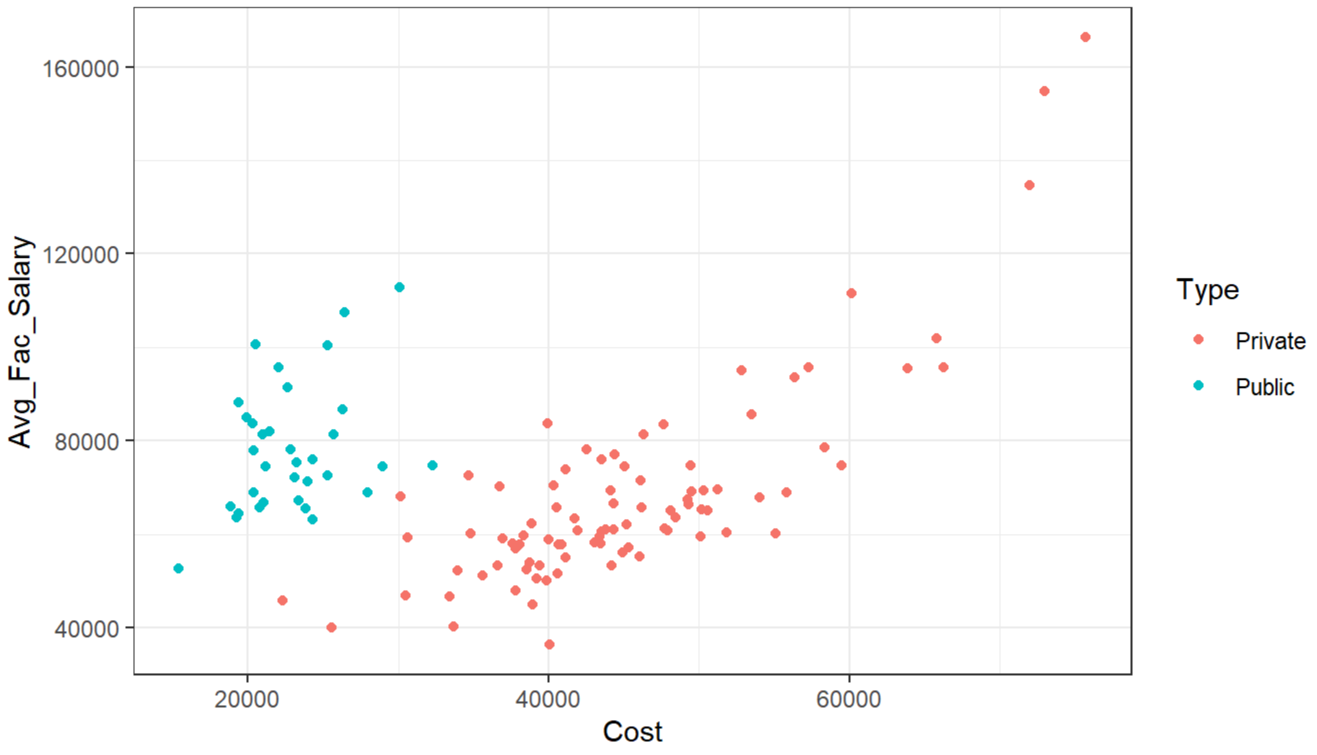
To load the data, simply copy and run the following:

```
college <- read.csv("https://collinn.github.io/data/college2019.csv")
midwest <- subset(college, State %in% c("IA", "MN", "IL", "MO"))
```

Color

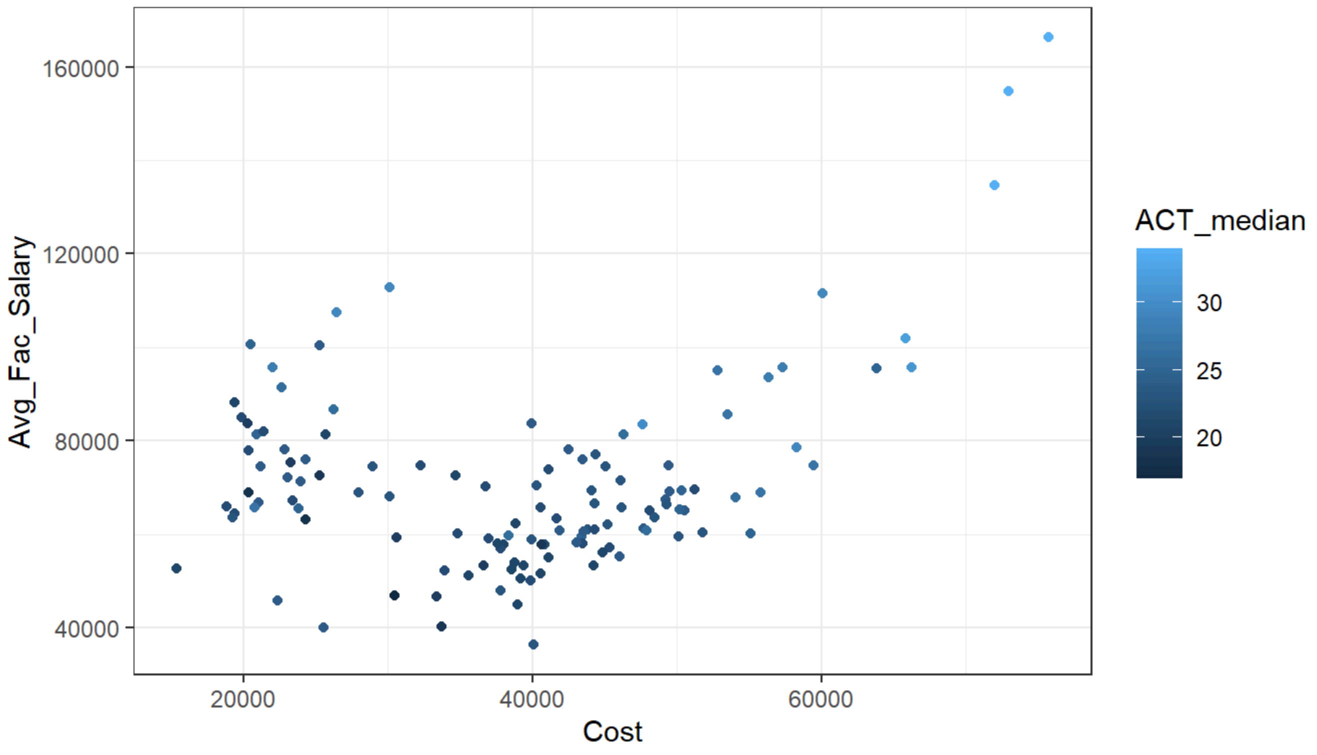
As we have seen previously in this lab, the color aesthetic can be used to include an additional variable in our plots. How the color aesthetic maps will be dependent on the type of variable included. For example, consider a scatterplot demonstrating the relationship between the cost of tuition and the average faculty salary when a *categorical* variable is included in the color aesthetic.

```
ggplot(midwest, aes(Cost, Avg_Fac_Salary, color = Type)) +
  geom_point()
```



We see that a legend is created to the right of the plot, with distinct colors provided for each of the values within the category. Contrast this with the case in which a *quantitative* variable is included in the color aesthetic:

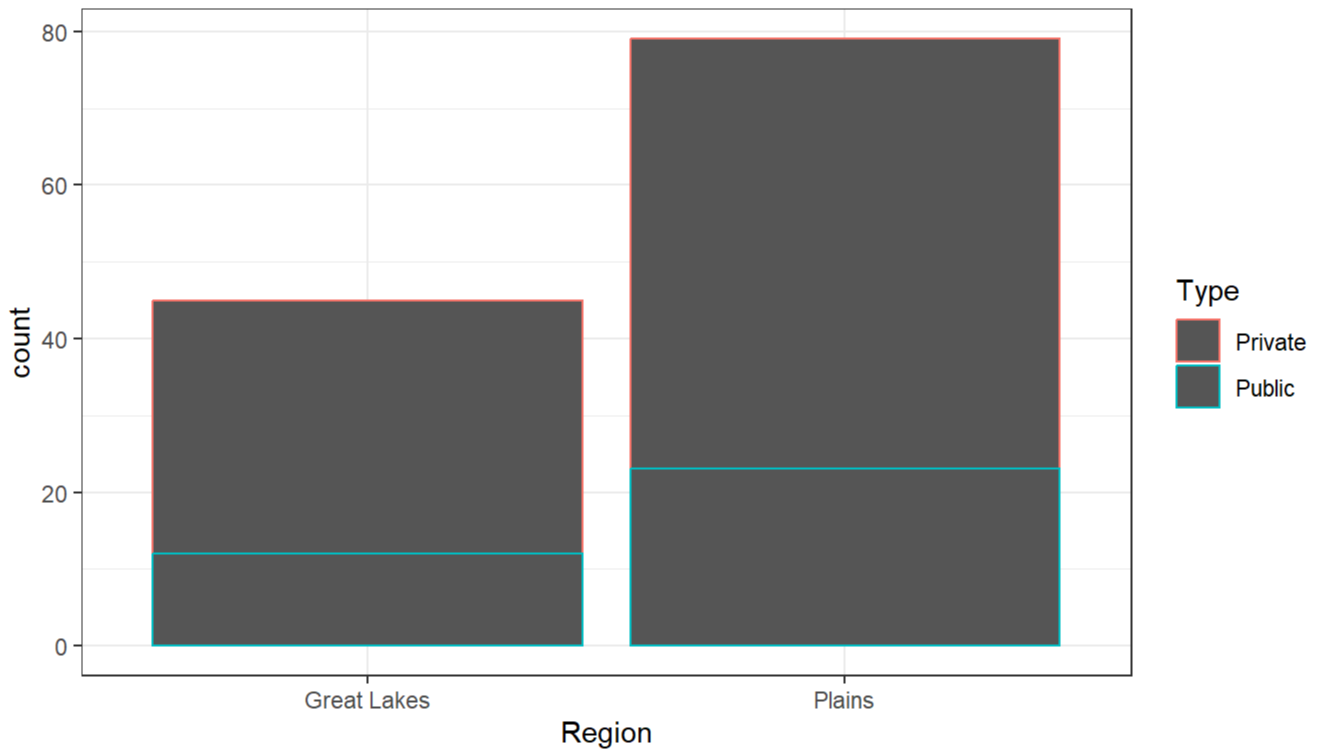
```
ggplot(midwest, aes(Cost, Avg_Fac_Salary, color = ACT_median)) +
  geom_point()
```



How was this changed?

A quick note should be made with regards to adding color to a bar chart. The `color` aesthetic in this case refers to the outline of the bars, rather than the bars themselves. You'll realize pretty quickly if you have specified this when you didn't intend to

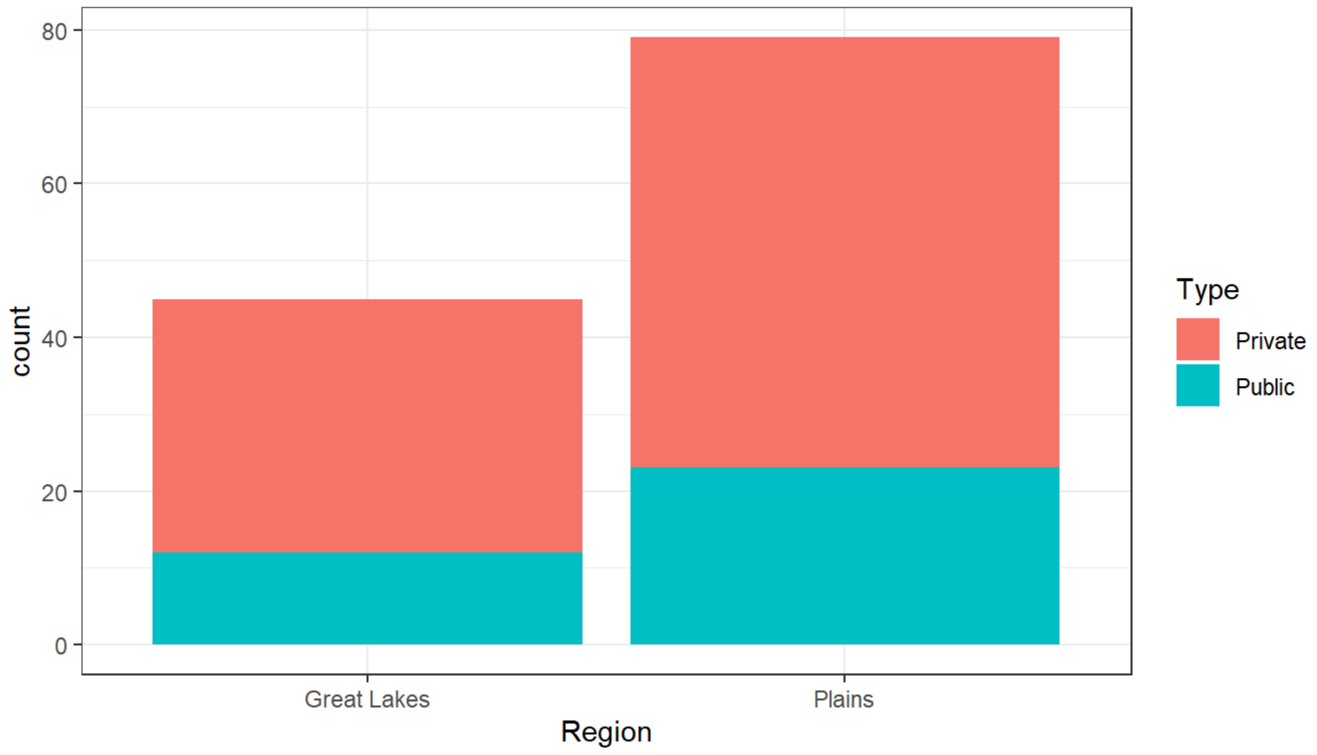
```
ggplot(midwest, aes(x = Region, color = Type)) +
  geom_bar()
```



Eww.

Instead, we want to use the closely related `fill` aesthetic which "fills" the bars with color

```
ggplot(midwest, aes(x = Region, fill = Type)) +
  geom_bar()
```



Note that this color/fill distinction applies when using boxplots as well.

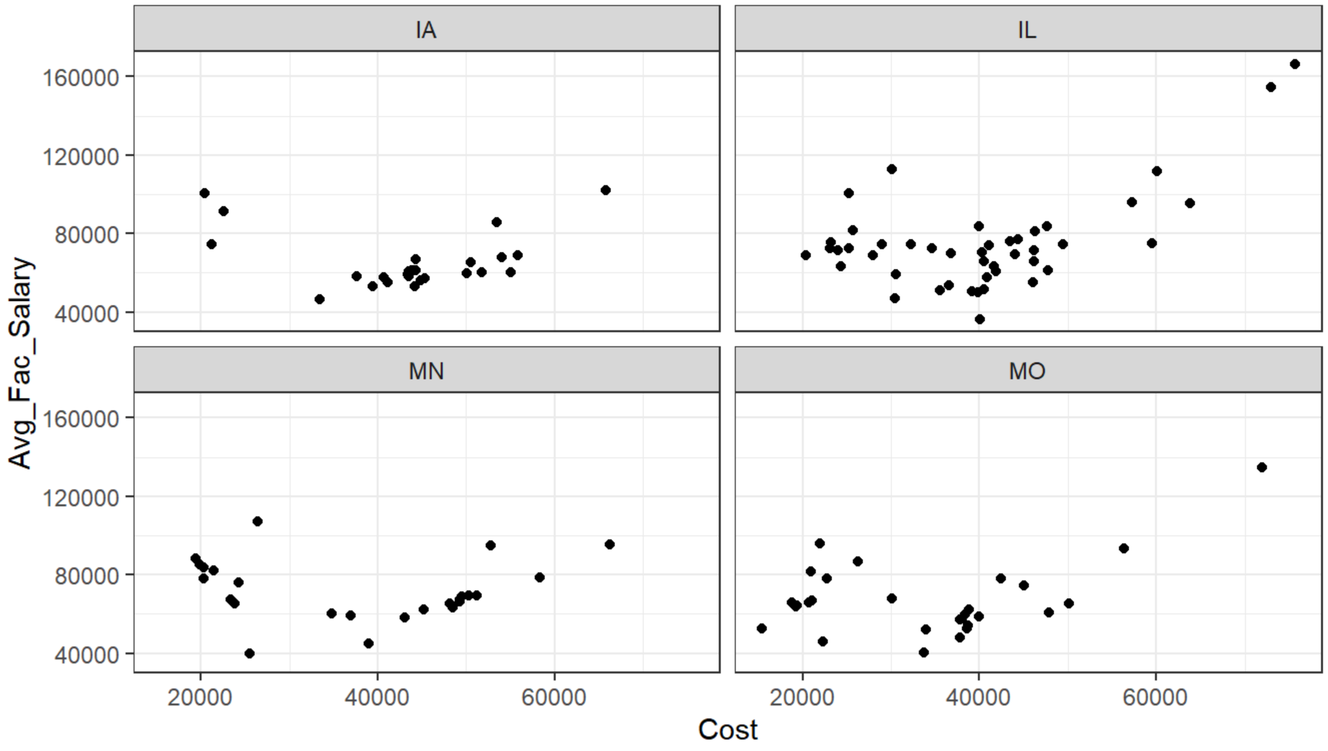
There are many things we can do with the colors of our plots, but making adjustments ourselves will rarely be required in this course. For those interested, there are a few examples included in the Examples section at the bottom or a slightly more comprehensive set of examples in an old (read: unedited) lab from STA-230 (https://collinn.github.io/teaching/2023/labs/lab4.html#Colors). Again, none of this is required.

Faceting

While color allows us to add an additional variable within a single plot, *faceting* permits us to separate a plot into pieces based on the values of a category. This is helpful, for example, if we wish to see if various trends or relationships appear to be similar across groups.

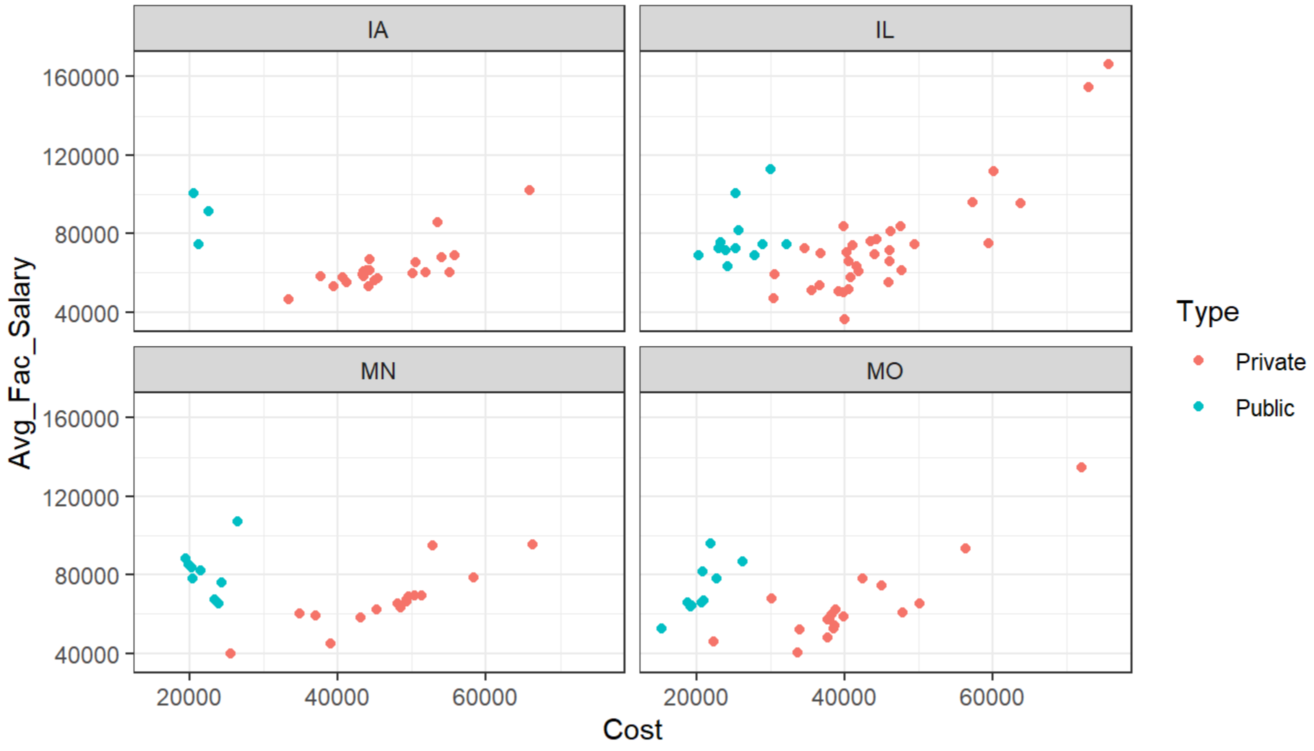
Just as we used `+` to add a layer for our geoms, we use `+` again to add faceting. This syntax is a little bit different than we have seen generally, but it is of the form `facet_wrap(~variable)`, where `variable` is the name of the variable in the dataset we wish to facet. For example, to facet the above scatter plot relating cost to faculty salary, we can facet by State to view this for each of the different states

```
ggplot(midwest, aes(Cost, Avg_Fac_Salary)) +
  geom_point() +
  facet_wrap(~State)
```



Of course, there is nothing from stopping us from limiting our multivariate plots to three variables; we can do four quite easily by adding a color aesthetic to our faceted plot:

```
ggplot(midwest, aes(Cost, Avg_Fac_Salary, color = Type)) +
  geom_point() +
  facet_wrap(~State)
```



As we can see, plots allow us to very quickly and concisely summarize the relationships between all different types of variables in our datasets. Throughout the semester we will continue to focus on creating and presenting effective visual data summaries which will involve both identifying the correct types of plots to create, given the variables we wish to summarize, as well as utilizing faceting and the various aesthetics to best capture the associations and relationships we wish to present.

Problem Sets

Part 1 - Univariate Plots

Question 2: The `class` variable in the `mpg` dataset details the “type” of car for each observation (pickup, SUV). Create the appropriate graph to demonstrate this distribution of this variable and comment on which class appears to be the most frequent.

Question 3: The `cty` variable in the `mpg` dataset details the average fuel economy for the vehicle when driven in the city. Create the appropriate graph to represent this variable. Does this variable appear to be symmetric or skewed?

Part 2 - Bivariate Plots

Question 4: Using the `mpg` dataset, we are interested in determining if either front wheel drive, rear wheel drive, or 4 wheel drive have better highway fuel economy. Create the appropriate plot to show the distribution of fuel economy for each of these types of vehicles. Which appears to have the best fuel economy?

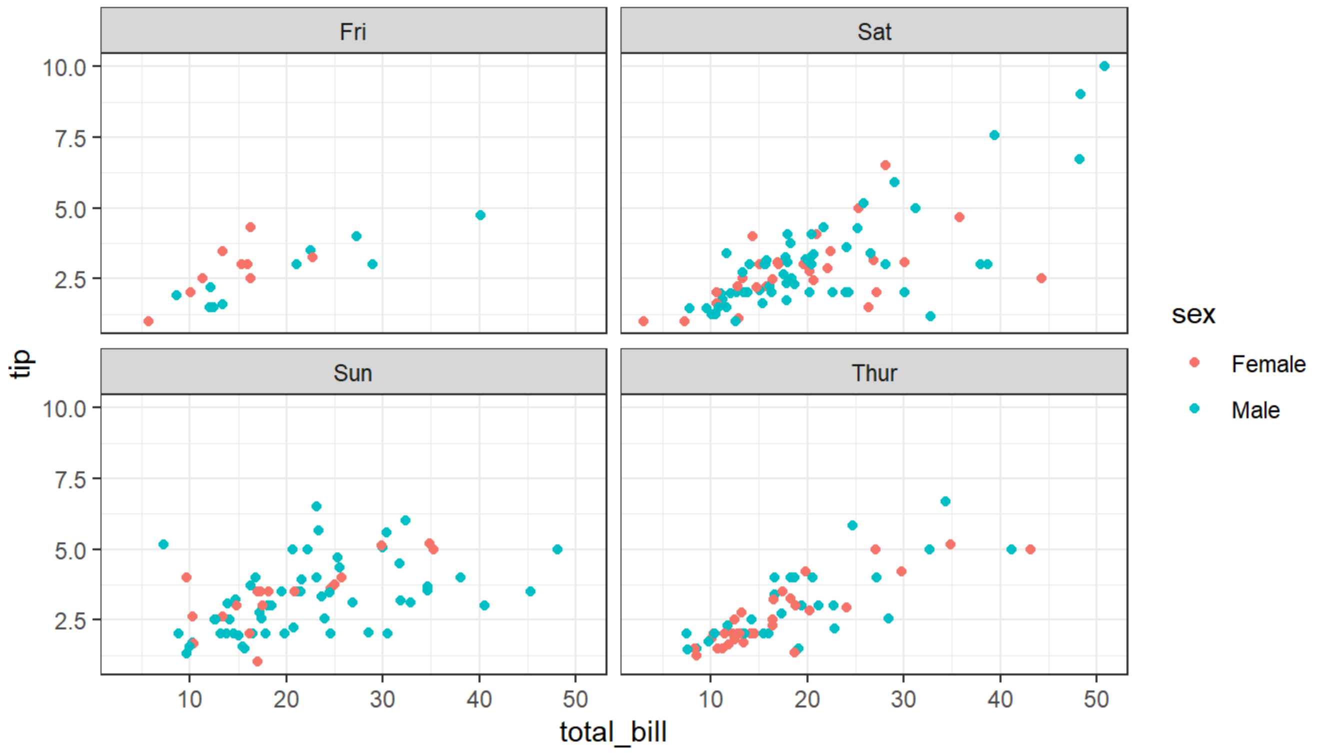
Question 5: In the `majors` dataset, the variable `Per_Masters` describes the percentage of the workforce within a particular field whose highest degree is a master’s degree. Create a graph to assess whether or not there appears to be a relationship between the percentage of individuals who hold a master’s degree and the percent that is unemployed. What do you find?

Question 6: Using the `mpg` dataset, create a plot to find which vehicle `class` produces the largest number of vehicles with 6 cylinder (`cyl`) engines. Which produces the largest proportion of 6 cylinder engines?

Part 3 - Multivariate Plots

Question 7 Below is code that reads in `tips` data that are used in the Example section of this lab. First identify the four variables from the `tips` data that are used to make this plot and then write code to reproduce it. (This plot uses *faceting* which has examples at the end of the lab)

```
tips <- read.csv("https://collinn.github.io/data/tips.csv")
```



Question 8 Using the full version of the `college` dataset (provided again below), create two different plots box plots with a `fill` aesthetic to illustrate the relationship between region, type, and admission rate. Which of your plots would be more useful in answering the question, “Do private or public schools generally have a higher admission rate?”

```
college <- read.csv("https://collinn.github.io/data/college2019.csv")
```

Question 9 The code below will load a data set containing 970 Hollywood films released between 2007 and 2011, then reduce these data to only include variables that could be known prior to a film’s opening weekend. The data are then simplified further to only include the four largest studios (Warner Bros, Fox, Paramount, and Universal) in the three most common genres (action, comedy, drama). You will use the resulting data for this question.

```
## Read in data
movies <- read.csv("https://collinn.github.io/data/hollywood.csv")

## Simplify data
movies <- subset(movies, LeadStudio %in% c("Warner Bros", "Fox", "Paramount", "Univ
ersal") &
                               Genre %in% c("Action", "Comedy", "Drama"),
                       select = c("Movie", "LeadStudio", "Story", "Genre","Budget",
                                  "TheatersOpenWeek","Year","OpeningWeekend"))
```

Your goal in this question is to create a graphic that effectively differentiates movies with high revenue on opening weekend from those with low revenue on opening weekend (the variable `OpeningWeekend` records this revenue in millions of US dollars). In other words, using the plotting methods included in this lab, we want to create a visual summary of the data that answers the question: which trends or attributes seem to predict a film having low or high opening weekend revenue.

Your plot should include *at least* three variables from the dataset, either by including them in the axes, through color or fill, or through faceting. Finally, using the graph you have created, write 2-3 sentences explaining detailing what you have found.

Examples

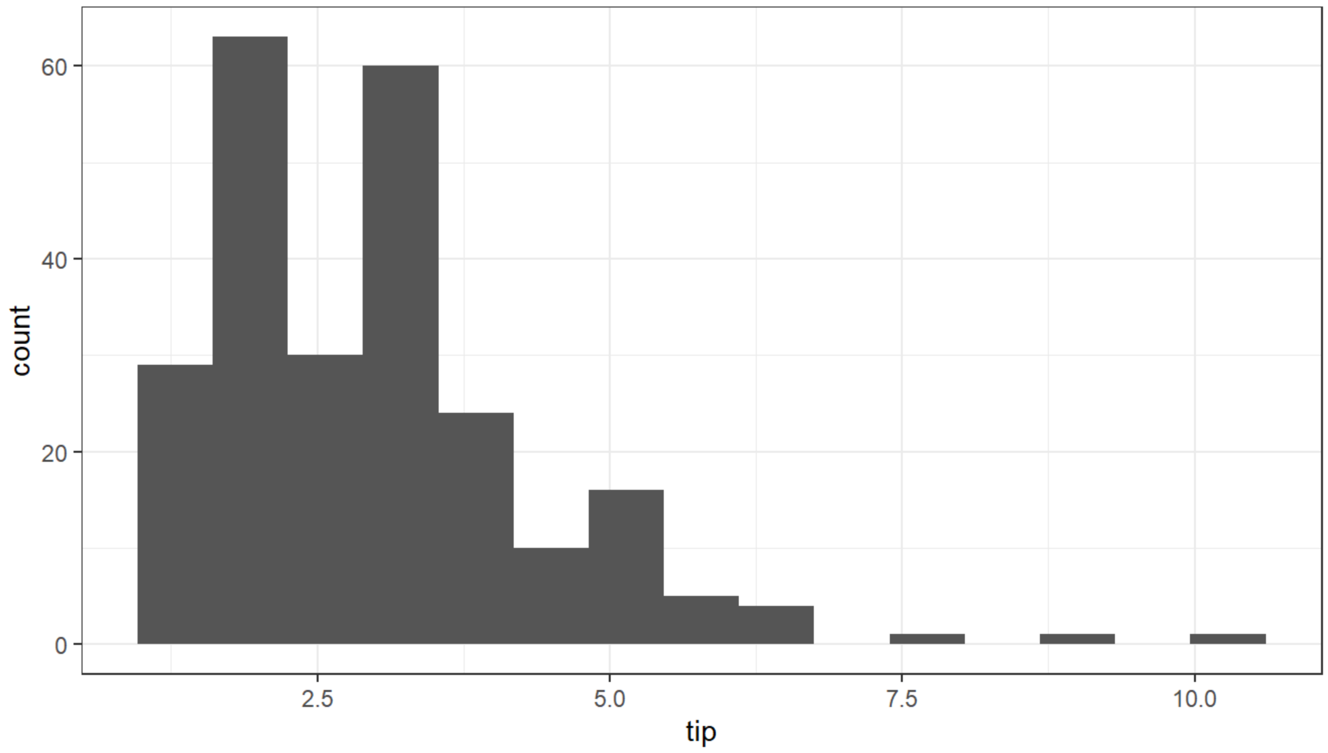
These examples will be done with the `tips` dataset, loaded in below:

```r
## Read in the "Tips" data
tips <- read.csv("https://collinn.github.io/data/tips.csv")
```

Histograms

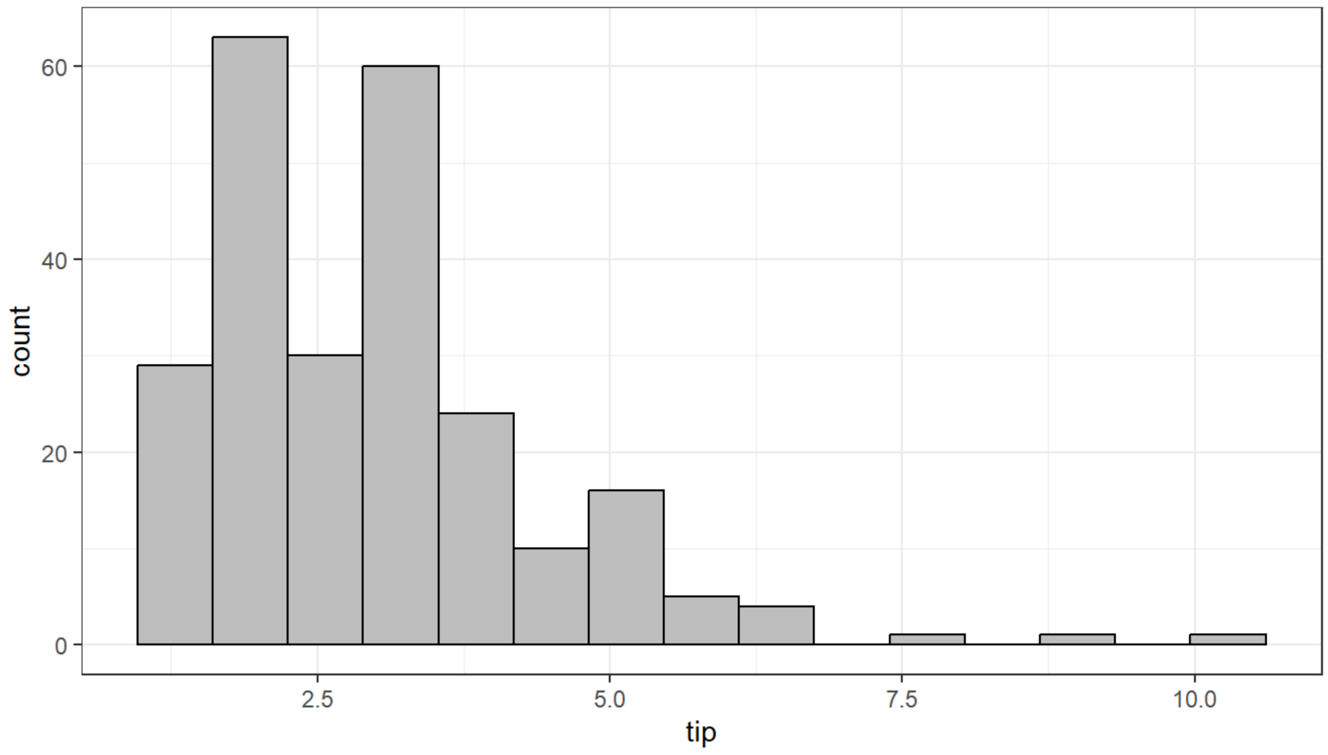
We can change the number of bins with `bins` argument. Feel free to play around with different numbers until you get what you want

```r
ggplot(tips, aes(x = tip)) + geom_histogram(bins = 15)
```



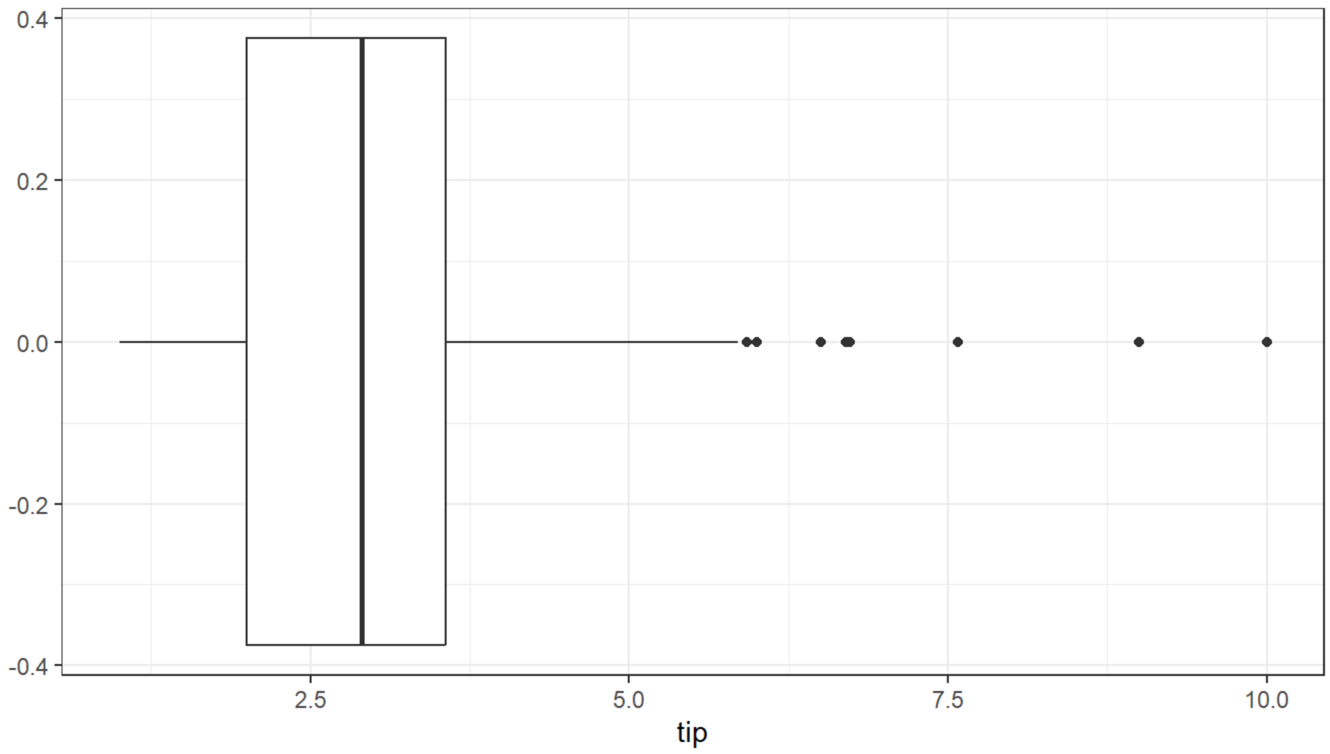
I think the default colors here are incredibly ugly, so I usually add lines and a slightly less offensive color to histograms in one variable

```r
ggplot(tips, aes(x = tip)) + geom_histogram(bins = 15, color = 'black', fill = 'gra
y')
```



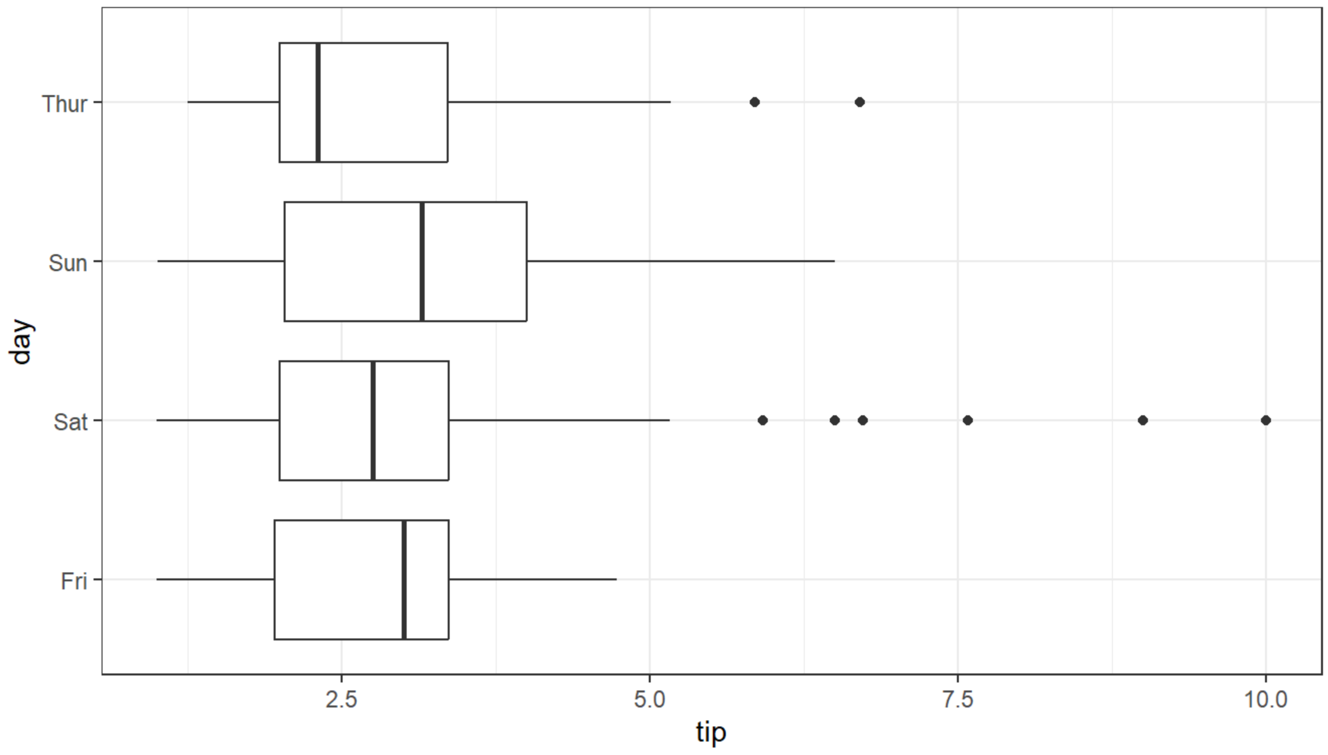
Boxplots

```
ggplot(tips, aes(x = tip)) + geom_boxplot()
```



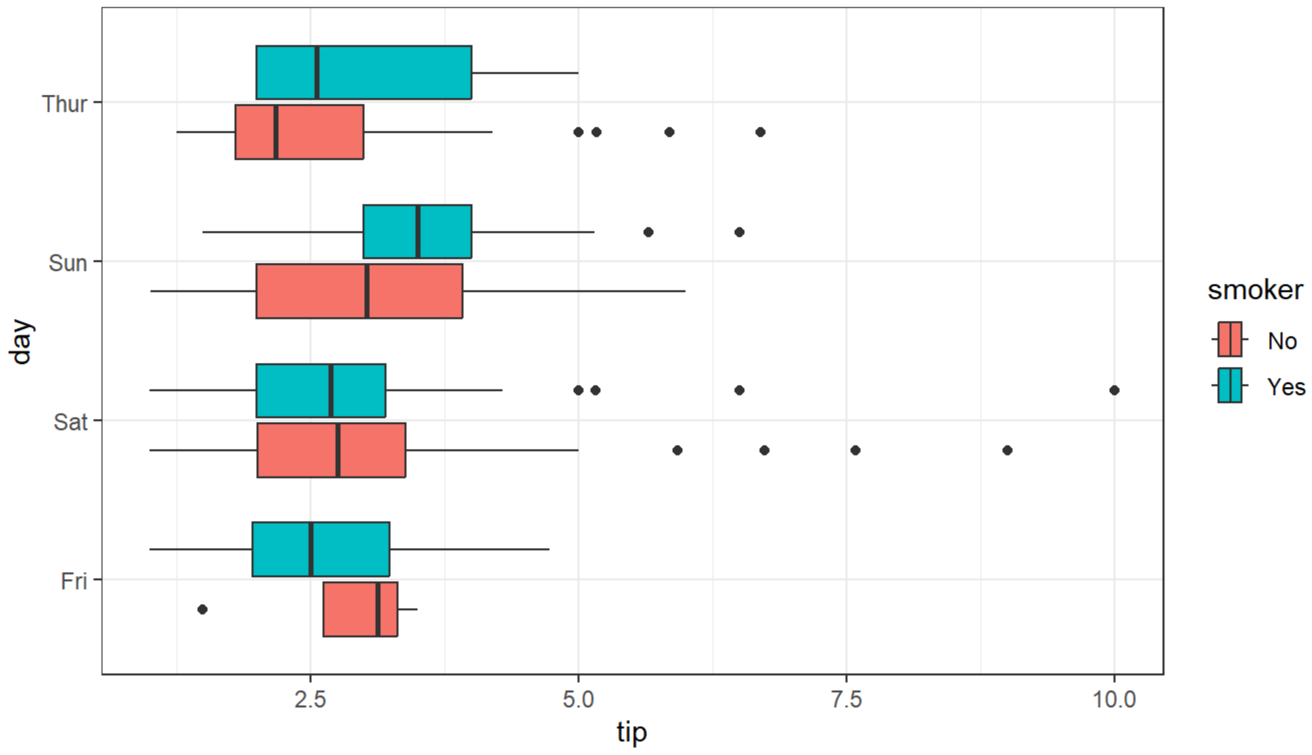
We can add groups by passing an extra categorical argument to `aes()`

```
ggplot(tips, aes(tip, y = day)) + geom_boxplot()
```

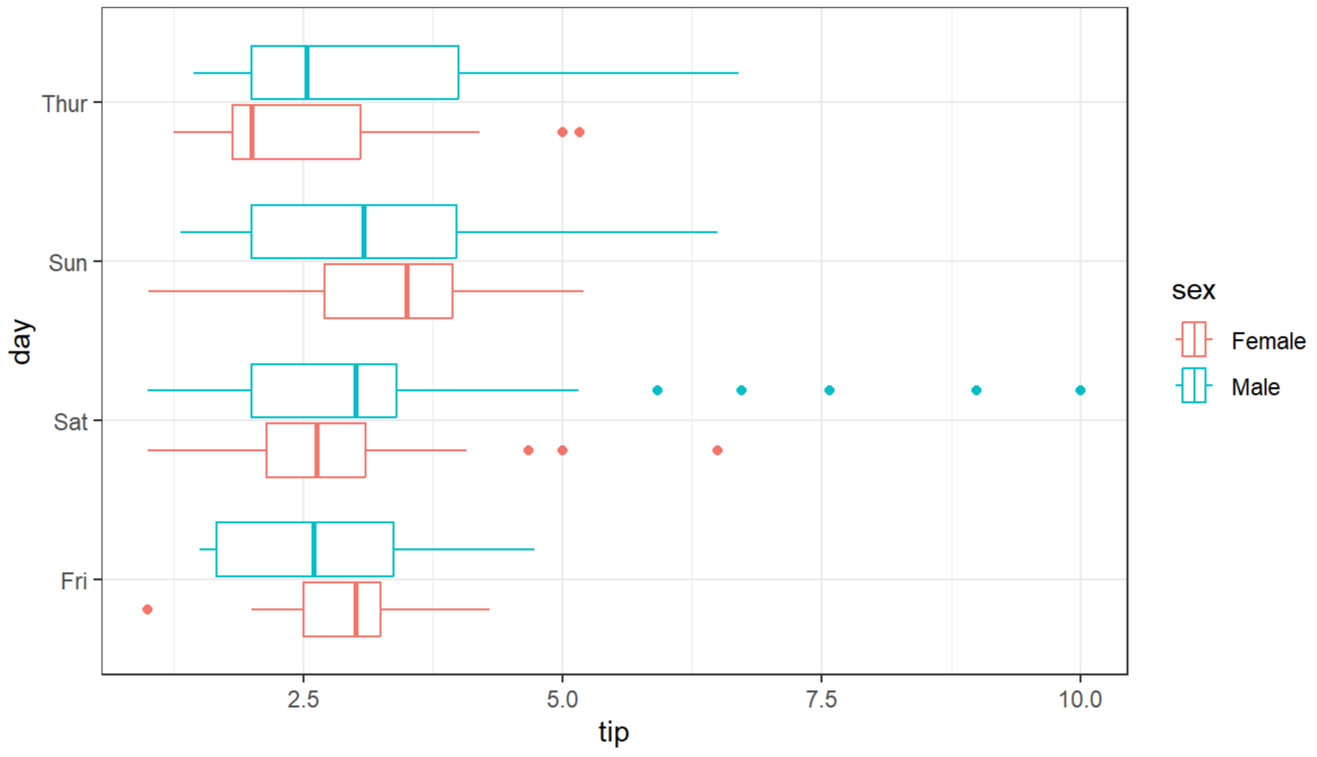


We can add more groups by including a color or fill aesthetic:

```
ggplot(tips, aes(tip, y = day, fill = smoker)) + geom_boxplot()
```



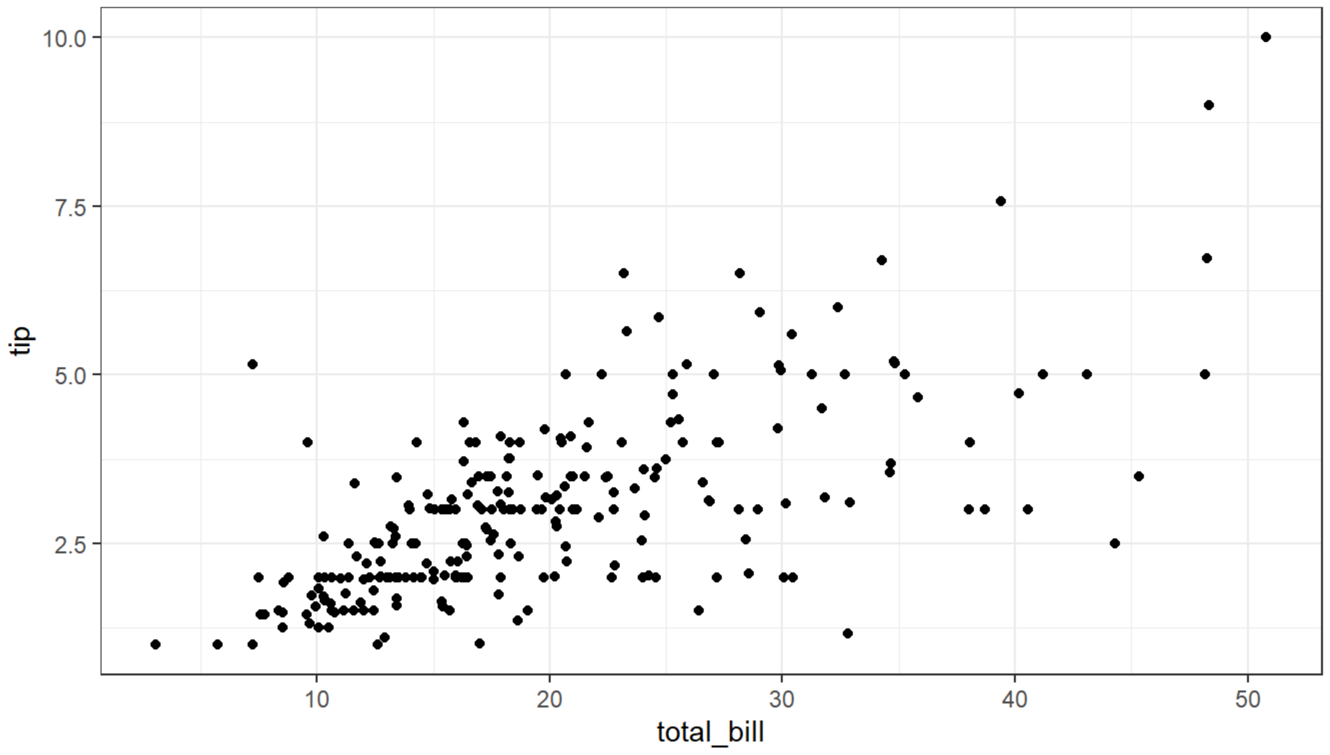
```
ggplot(tips, aes(tip, y = day, color = sex)) + geom_boxplot()
```



Scatter plots

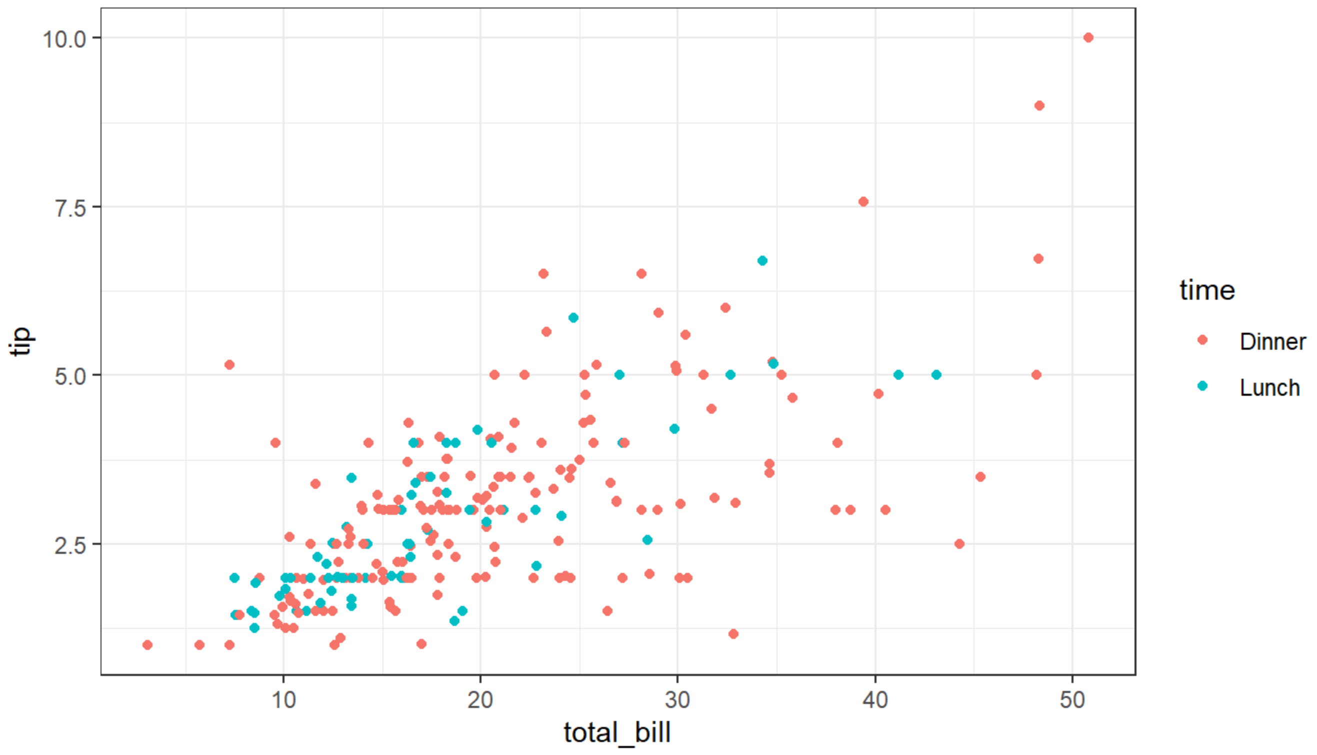
A basic scatter plot requires only two quantitative variables

```
ggplot(tips, aes(total_bill, tip)) + geom_point()
```

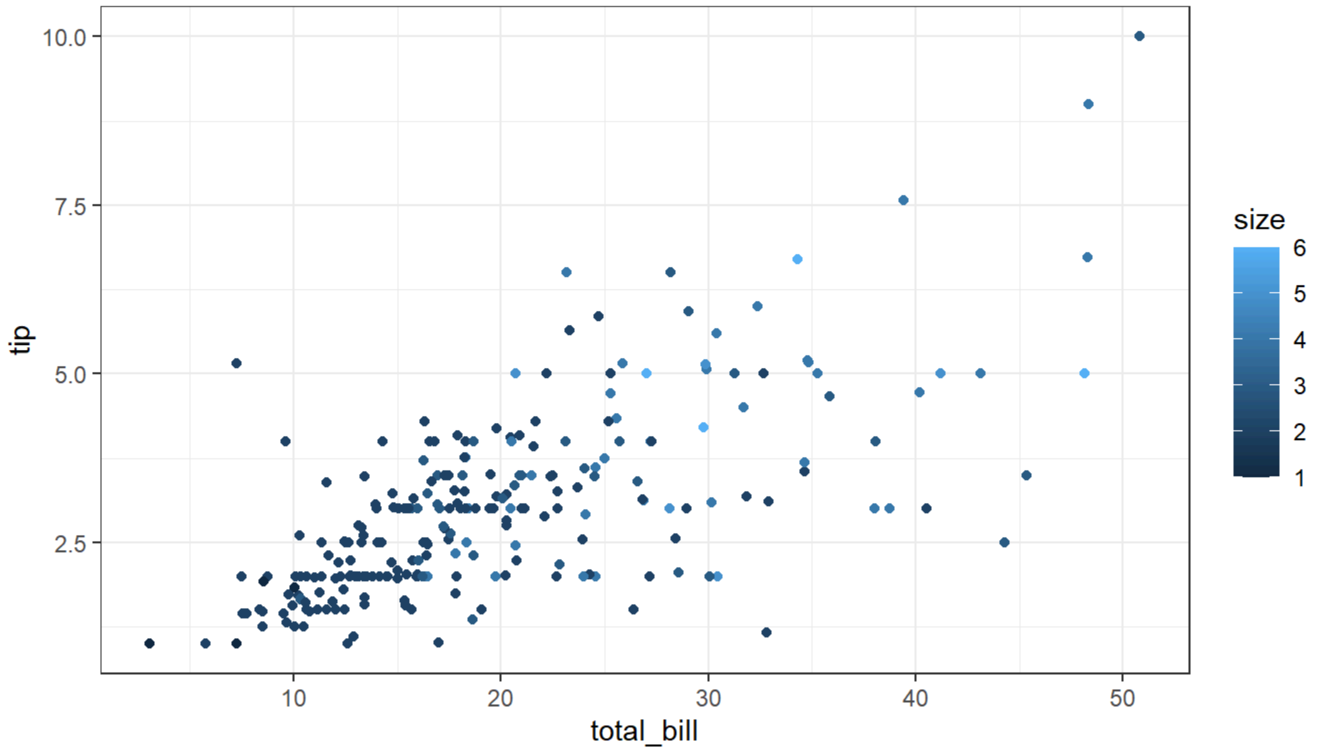


We can add color to this plot to represent a third variable: note that categorical variables will create a collection of “discrete” colors, while adding a quantitative variable will give us color on a scale

```
# Categorical variable added
ggplot(tips, aes(total_bill, tip, color = time)) + geom_point()
```



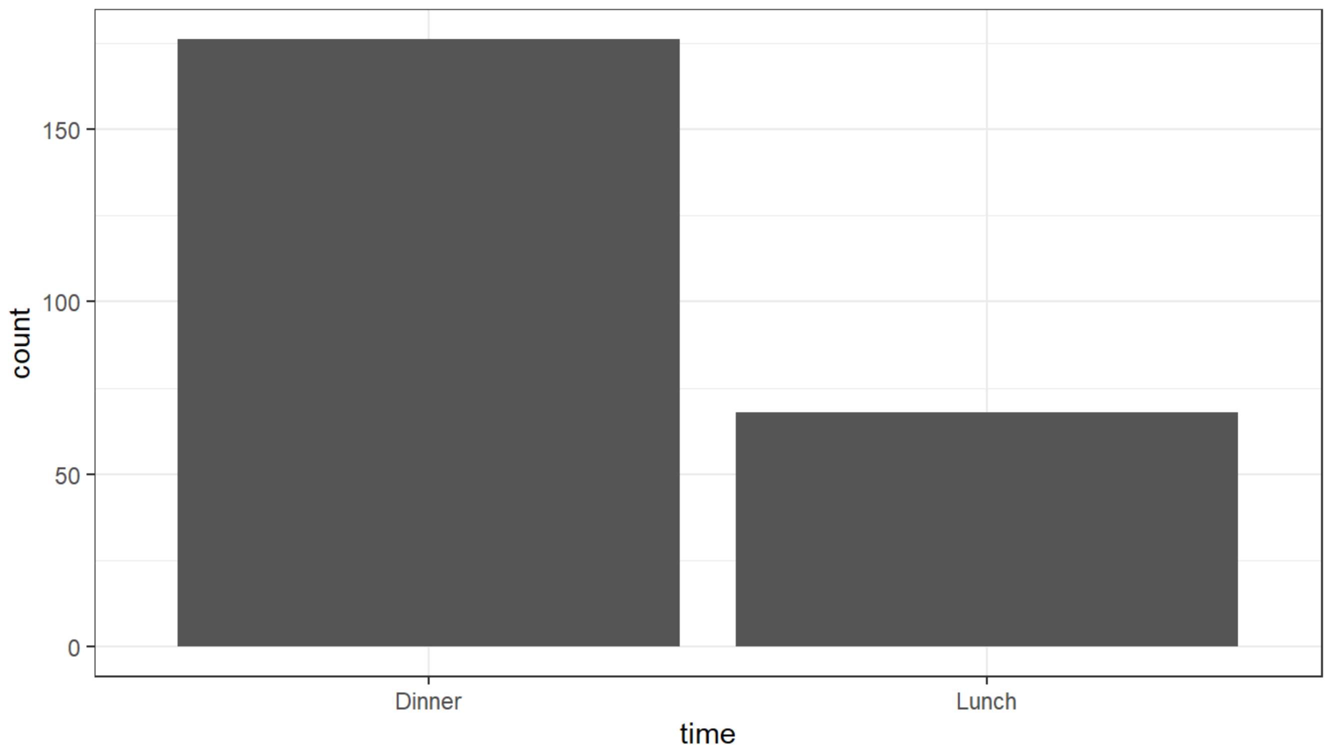
```
# Quantitative variable added
ggplot(tips, aes(total_bill, tip, color = size)) + geom_point()
```



Bar plots

The simplest box plot returns the distribution of a single variable

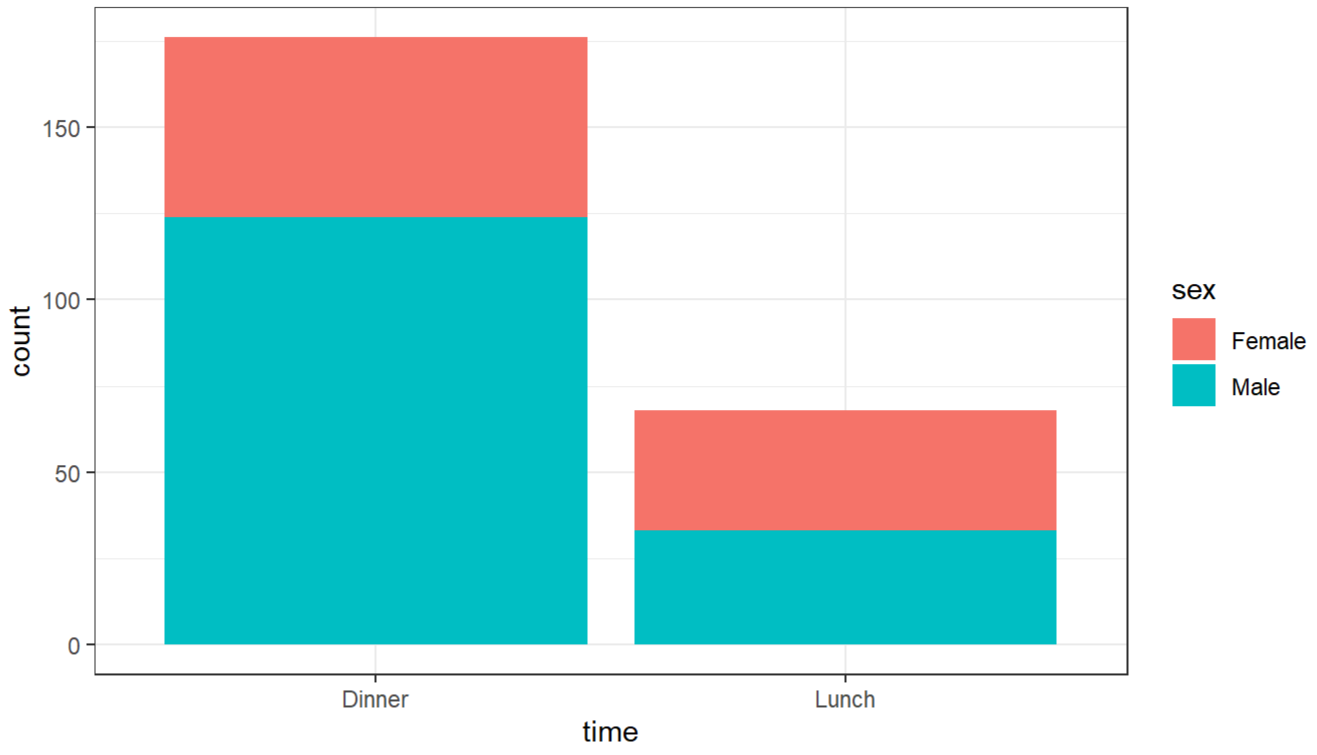
```
ggplot(tips, aes(time)) + geom_bar()
```



Stacked

When adding additional categorical variables, the default behavior is to give a stacked bar chart

```
ggplot(tips, aes(time, fill = sex)) + geom_bar()
```

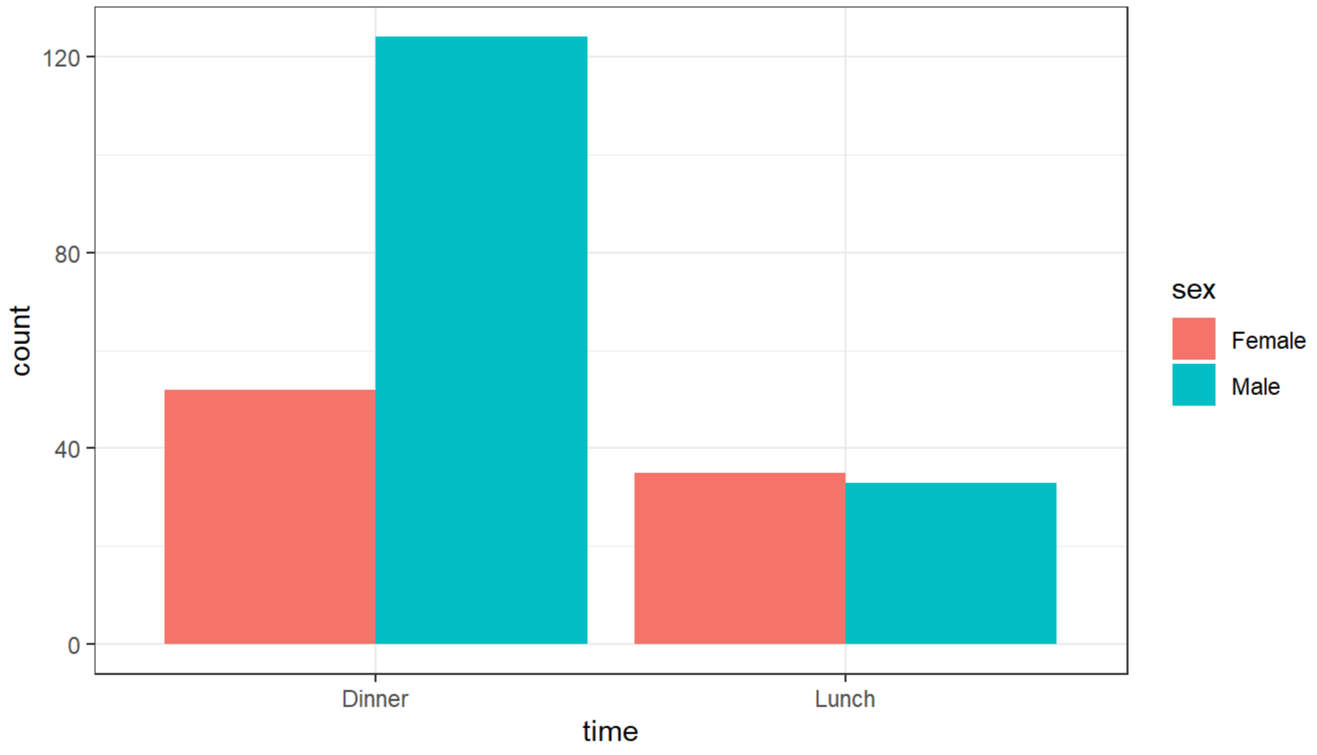


Note that for bar charts our second aesthetic will almost always be `fill =`. This tells ggplot to “fill in” with the color of the group. Try doing `color = Sex` to compare

Dodge

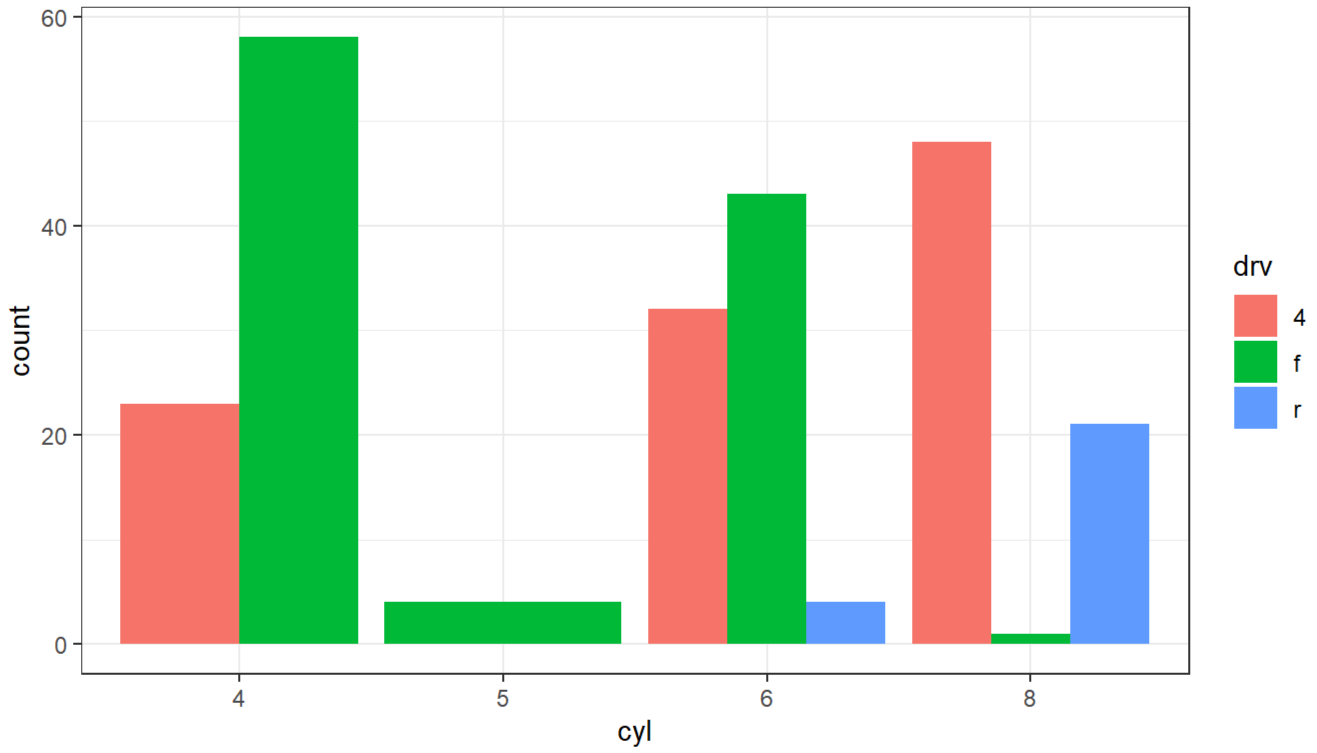
We can change to a dodged bar chart by passing an argument to `geom_bar()`

```
ggplot(tips, aes(time, fill = sex)) + geom_bar(position = "dodge")
```



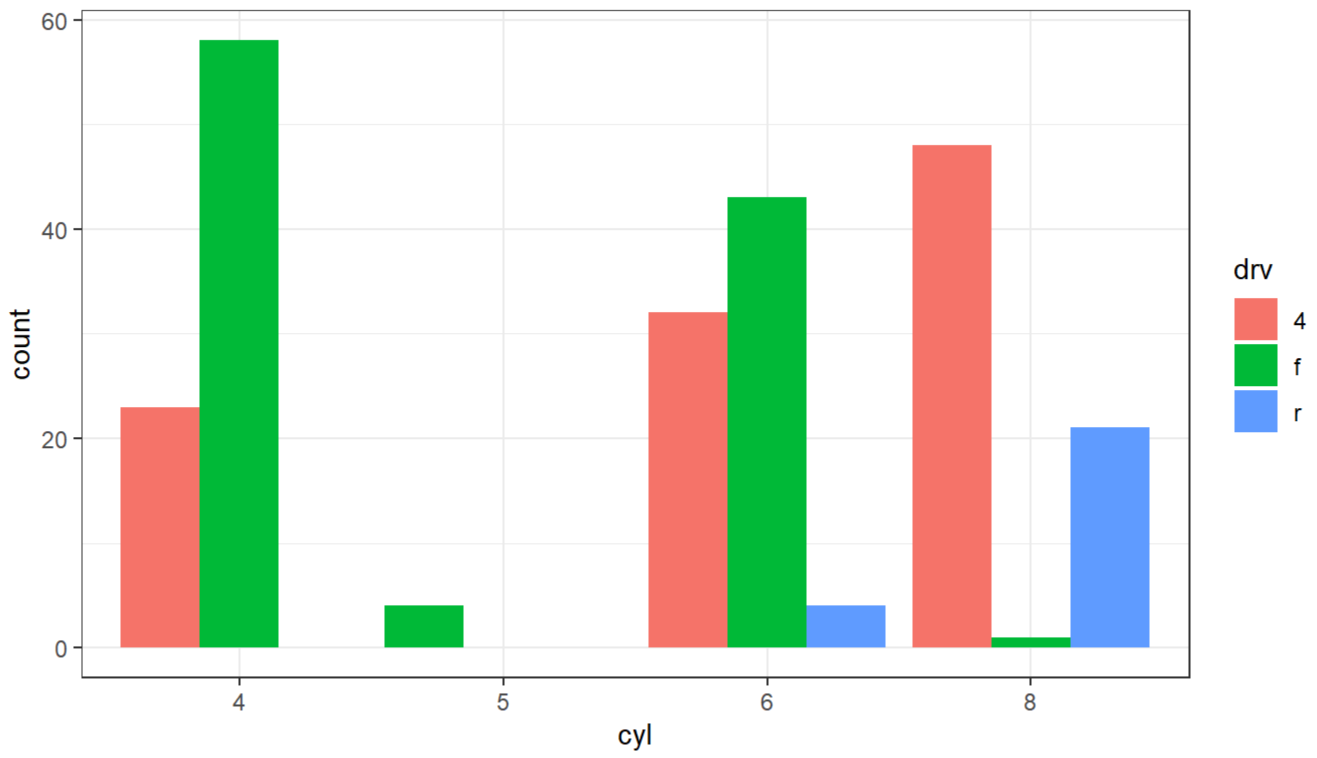
If there is a case in which one group has zero observations from another group, the default behavior for dodge looks weird. Here's an example from the `mpg` dataset:

```
ggplot(mpg, aes(cyl, fill = drv)) + geom_bar(position = "dodge")
```



Compare the width of the bars for 4 cylinder, 5 cylinder, and 8 cylinder; the default behavior is to fill in the missing groups by making the bars wider. It's a bit more typing to avoid this behavior, but it can be done

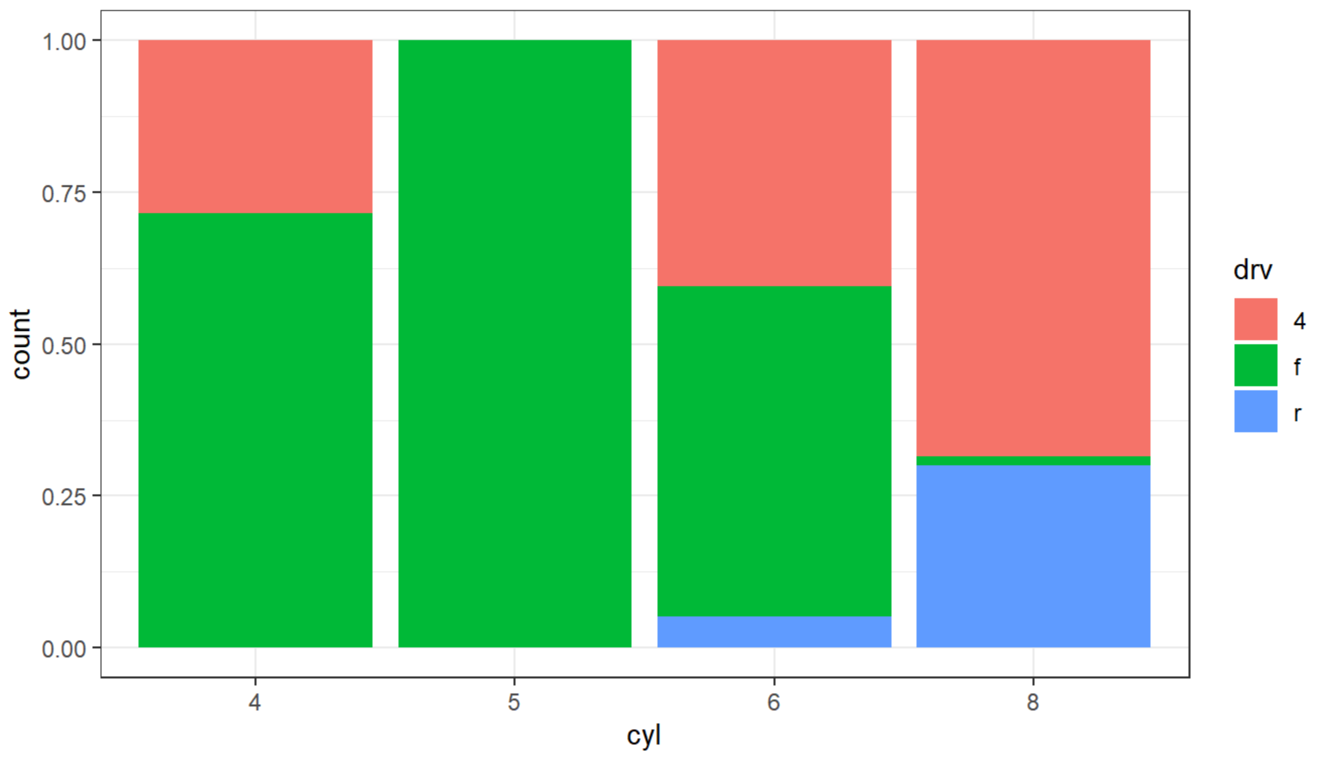
```
ggplot(mpg, aes(cyl, fill = drv)) + geom_bar(position = position_dodge(preserve =
"single"))
```



Filled charts (conditional)

Finally, we can get the filled/conditional charts by using `position = "fill"`

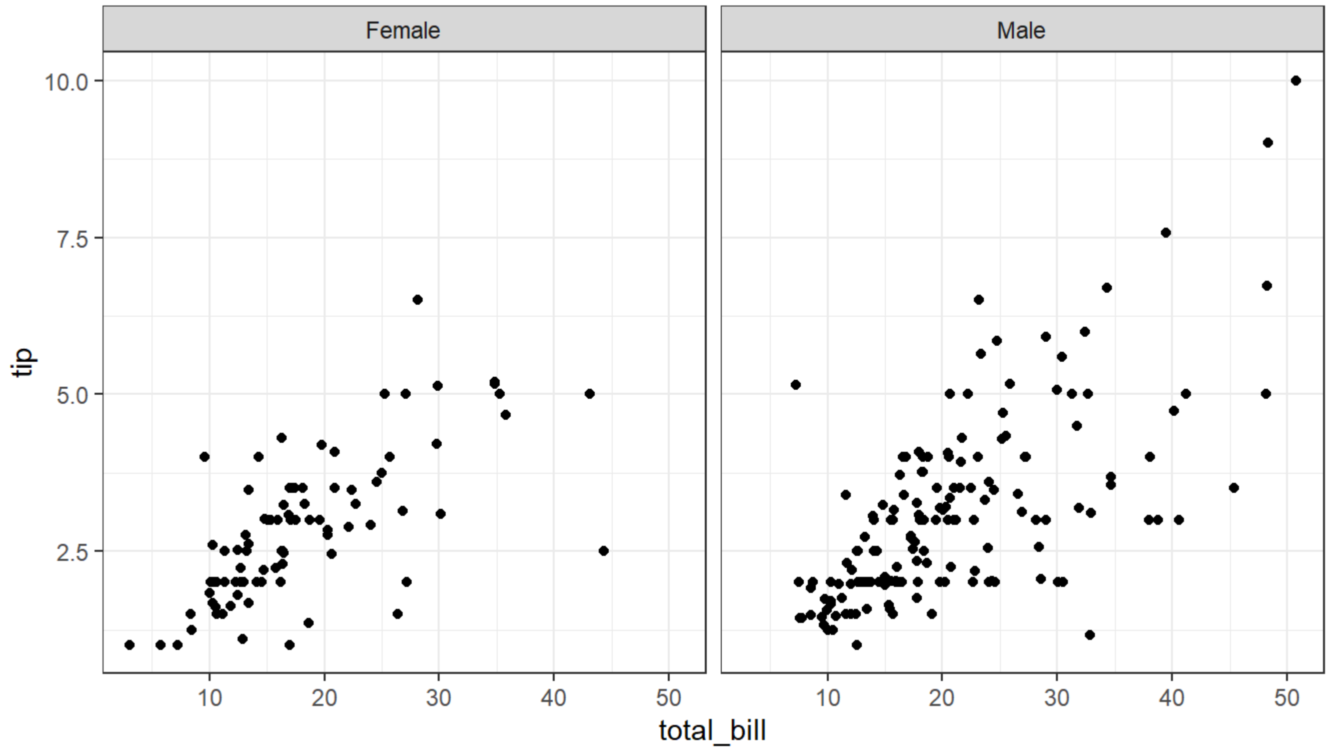
```
ggplot(mpg, aes(cyl, fill = drv)) + geom_bar(position = "fill")
```



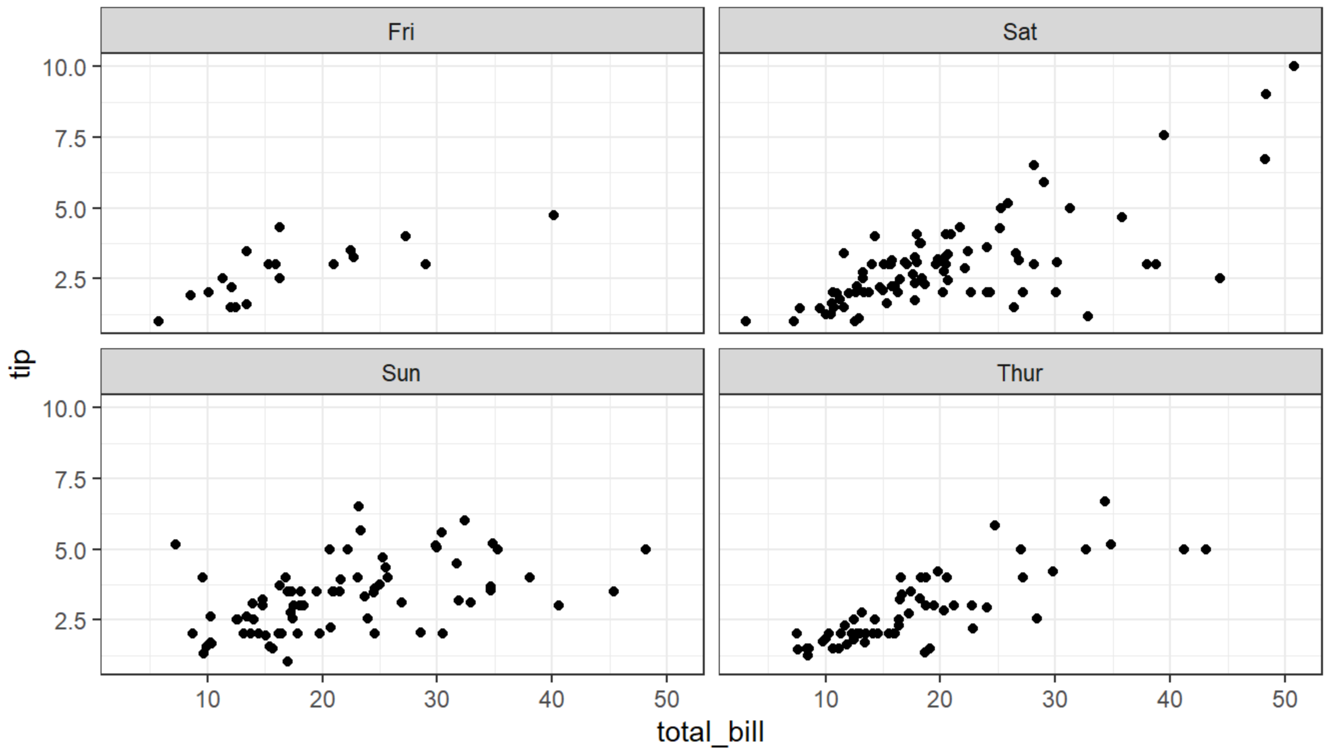
Faceting

“Facetting” is a process whereby we split our plots up by a group or categorical variable, allowing us to see two (or more) side-by-side plots. This can be a handy way to view the relationship between two variables separately, conditioned on a third

```
ggplot(tips, aes(total_bill, tip)) +
  geom_point() +
  facet_wrap(~sex)
```



```
ggplot(tips, aes(total_bill, tip)) +
  geom_point() +
  facet_wrap(~day)
```



The syntax is `facet_wrap(~var_name)` and can be added with `+` at the end of the plot

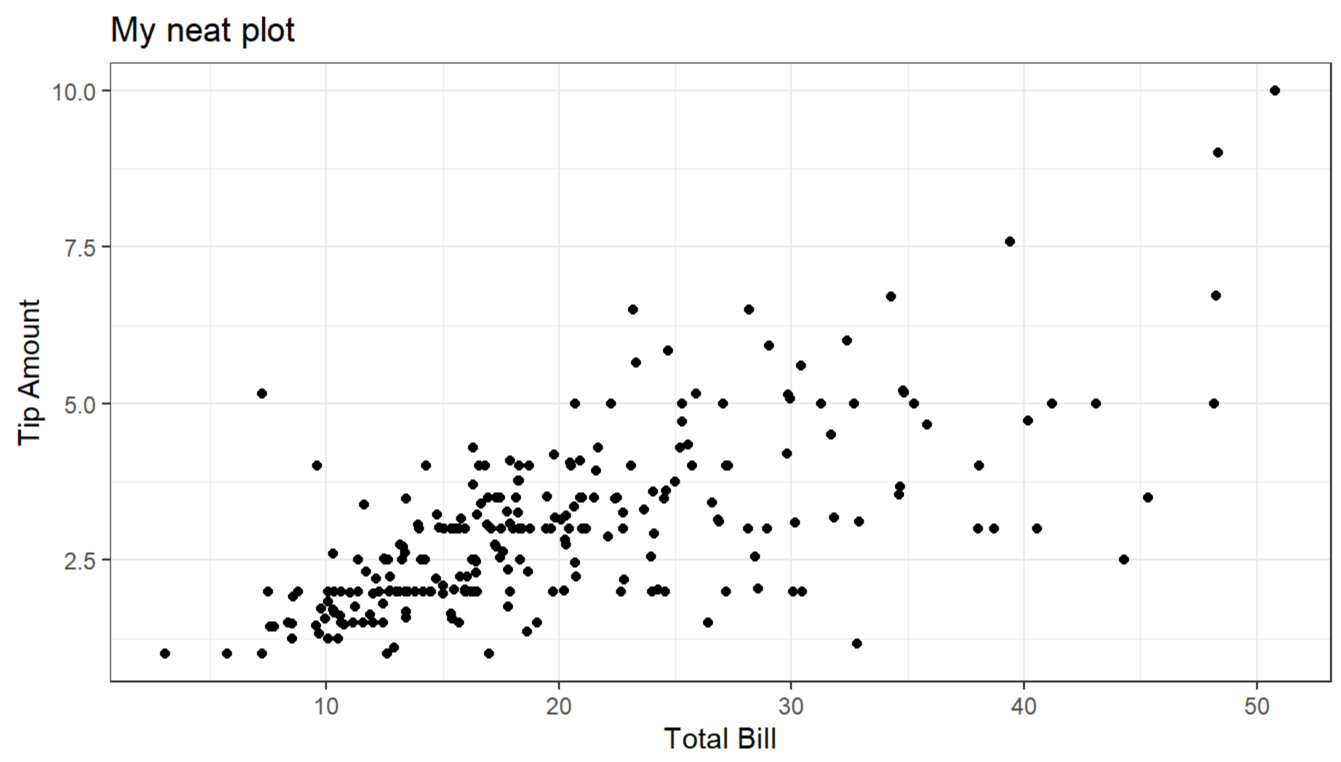
Extra Details

Here are some extra examples for things you may be intereseted in modifying in your plot

Labels and title

We can add labels and a title with the `labs()` function (short for labels). The arguments to change labels are shown below

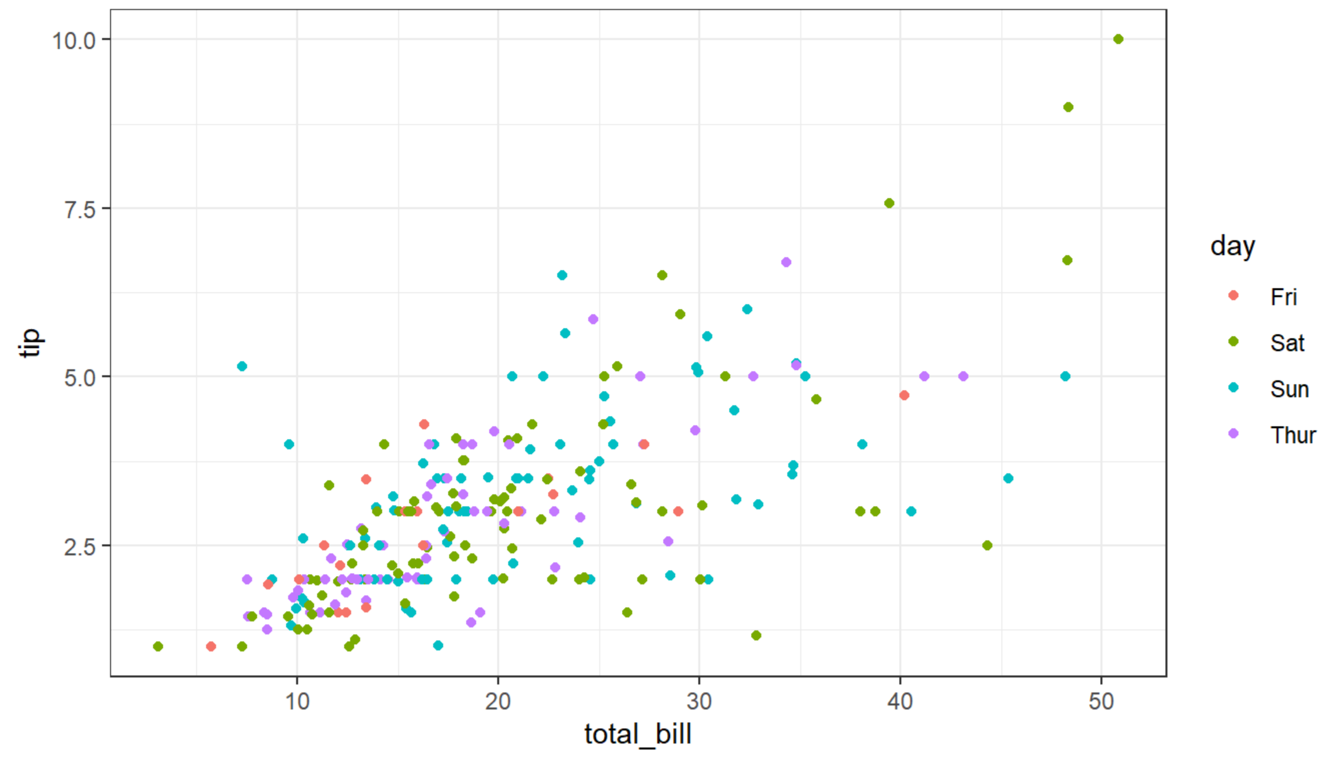
```
ggplot(tips, aes(total_bill, tip)) +
  geom_point() +
  labs(x = "Total Bill", y = "Tip Amount", title = "My neat plot")
```



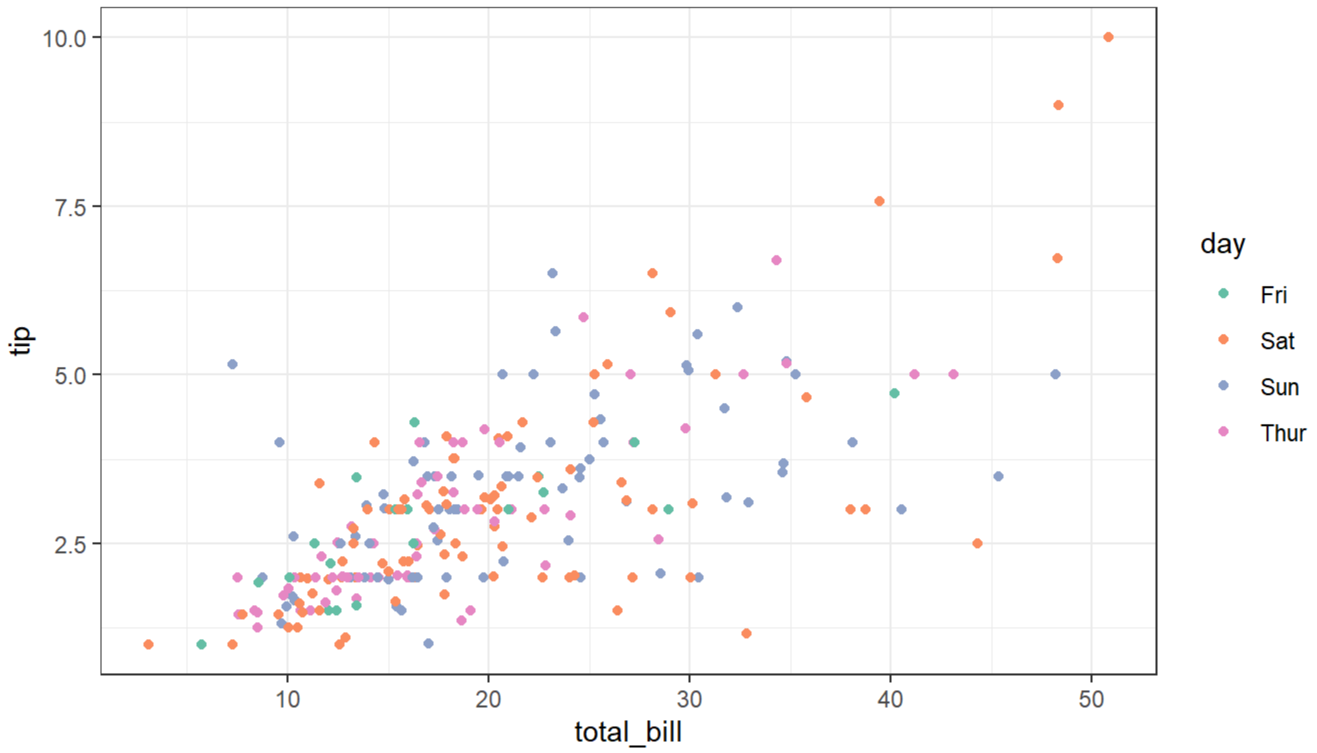
Colors

This is (mostly) just for fun. If we use either a color or fill aesthetic in our plots, we can change the colors

```
# Original
ggplot(tips, aes(total_bill, tip, color = day)) +
  geom_point()
```

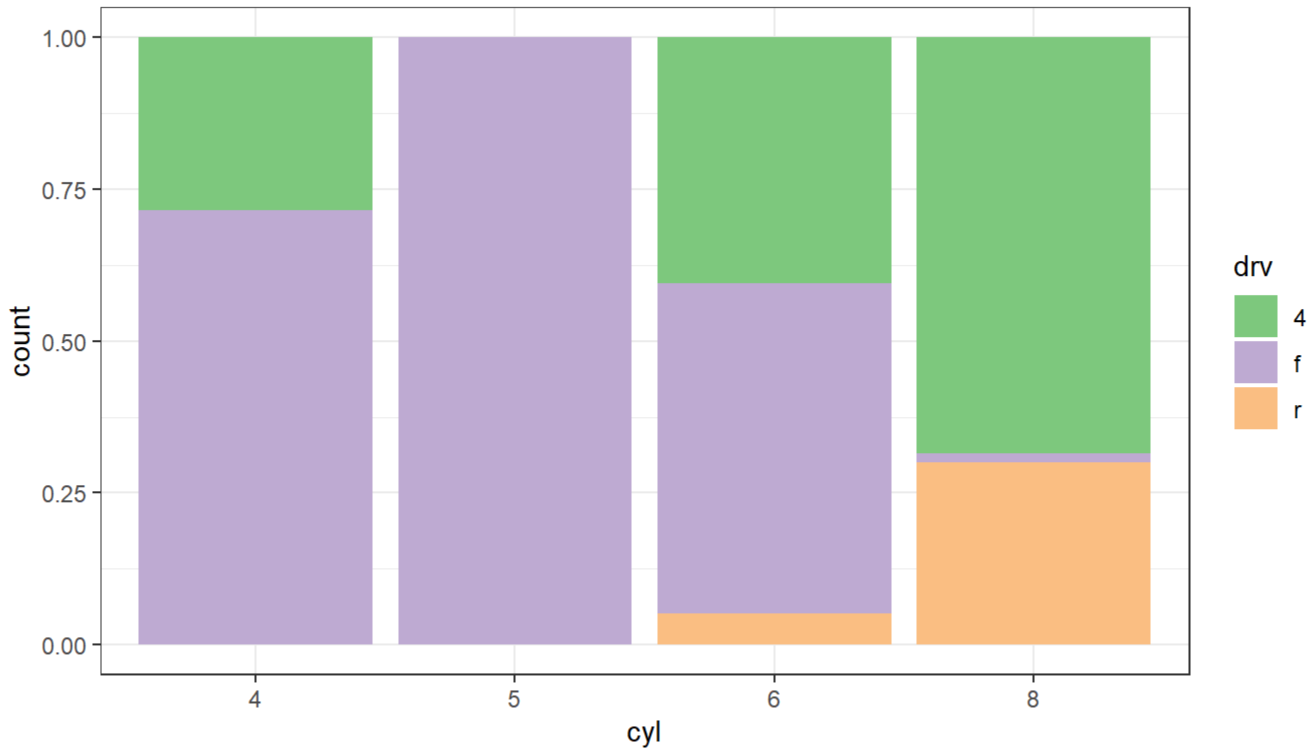


```
# Using scale_color_brewer
ggplot(tips, aes(total_bill, tip, color = day)) +
  geom_point() +
  scale_color_brewer(palette = "Set2")
```



The syntax is nearly identical for fill using `scale_fill_brewer()`

```
ggplot(mpg, aes(cyl, fill = drv)) +
  geom_bar(position = "fill") +
  scale_fill_brewer(palette = "Accent")
```



A list of additional palettes can be found with `?scale_fill_brewer()` or `?scale_fill_color()`