Notes 04 - Data Visualization with ggplot2

STS 2300 (Spring 2025)

Updated: 2025-02-02

Table of Contents

Reading for Notes 04

Read Chapter 2 of ModernDive to learn about creating graphs with the `ggplot2` package. (Note: Some topics in this chapter, like Section 2.6 will be covered in Notes 05.)

The `ggplot2` cheat sheet will also be helpful as a reference.

Additional optional resources include Chapter 10 of R for Data Science and Chapter 3 of Modern Data Science with R.

For a **much** more in depth look, see the R Graphics Cookbook

---

Learning Goals for Notes 04

- Understand the `ggplot2` grammar of graphics including how to use the `aes()` function.
- Be able to use the `ggplot2` package to create histograms, boxplots, bar graphs, and scatterplots.
- Be able to display information about additional variables in these graphs using color and other features.

STS 2120 Review

In STS 2120, you should have (or will, if currently enrolled) covered the basics of graphs and distributions. We will do a review of some types in STS 2300, but you should be comfortable with the following ideas:

- Basic graph types
    - Histograms
    - Boxplots
    - Bar graphs
    - Pie charts
    - Scatterplots
    - Line graphs
- Distributions of variables
    - Bell-shaped distribution (e.g., t distribution, Normal distribution)
    - Uniform distribution
    - Skewed distributions (right and left skewed)

The ggplot2 package

ggplot2 Cartoon (from https://github.com/allisonhorst/stats-illustrations)

ggplot2 is a package that lets you make nice looking graphs quickly while using consistent syntax. This is in contrast with "base graphics", which are graphing functions pre-installed in R. The letters "gg" stand for "grammar of graphics".

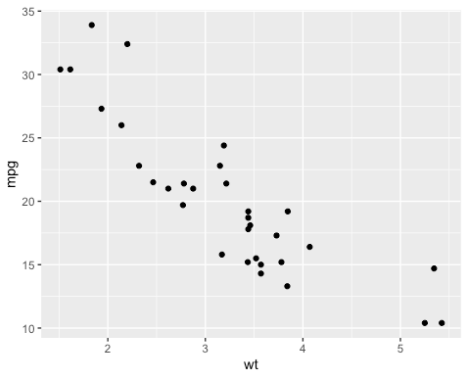
To make graphs using the grammar of graphics for this package, we will

i. start with a data frame
ii. map variables from the data frame to aesthetic features of our graph (like the x-axis, the y-axis, the color of the points, etc.)
iii. and add geometric objects (called geoms) to our graph (like bars on a bar graph or points in a scatterplot).

Below is an example of how we can make a scatterplot:

```
library(ggplot2)

ggplot(data = mtcars) +
  geom_point(aes(x = wt,
                 y = mpg))
```



So how do I understand the code above?

- First, the `ggplot()` function is creating an object for graphing, and I'm specifying that I want to use the mtcars data frame.
- Then I'm adding a "points" geometric object using the `geom_point()` function.
- Lastly, I'm using the `aes()` function inside `geom_point()` to *map* variables in my data frame to specific *aesthetic* elements of my scatterplot. In this case, I'm mapping the weight of the cars (variable `wt`) to the x axis and the mileage per gallon of the cars (variable `mpg`) to the y axis.

Practice Questions

1. Take the code above and run it on your own to get this graph (make sure you've installed and loaded the `ggplot2` package).

2. Try changing the code to make a scatterplot for horsepower (the variable is called `hp`) on the x axis vs. miles per gallon on the y axis.

   See notes04.R.

3. (Bonus) Try using `geom_bar()` (instead of `geom_point()`) to create a bar graph of transmission types. The transmission variable is called `am` (with 0 = automatic and 1 = manual transmission). You will only have *one* variable to map to an aesthetic here. We will cover bar graphs in more detail below, but it will be helpful to think through the process of adapting this code before that.

   See notes04.R.

Types of graphs

There are many different kinds of graphs we can make with the `ggplot2` (and even more variations on those graphs). For example, the R Graph Gallery includes lots of examples. In many cases, there will be more than one way to write code to create these graphs. For now, we will just scratch the surface of what `ggplot2` can do, but we will expand on this in Notes 05, and I encourage you to explore ways to create graphs of interest to you.

Scatterplots

Review: What does a scatterplot graph?

Scatterplots typically display the relationship between two numerical variables.

We saw above that scatterplots are created by adding a point geom to an object created by `ggplot()`. In general, our code will look like this:

```
ggplot(data = dataset_name) +
  geom_point(aes(x = explanatory_var,
                 y = response_var))
```

There are lots of other ways we can customize our scatterplots within `geom_point()`. Below are some examples of other arguments:

- `color` controls the color of the points (default is "black")
- `size` controls how big the points are (default is 1)
- `shape` controls the shape of the points (see here for more options)
- `alpha` controls the transparency of the points (0 is completely see through and 1 is completely solid)

We will put these arguments **outside** of the `aes()` function unless we want to map another variable in our dataset to one of these aesthetic elements (more on this later).

Practice: Take the scatterplot we made above and see if you can make the points purple, bigger than they were before, slightly transparent, and triangles instead of circles.

See notes04.R.

Histograms

Review: What does a histogram graph?

Histograms show the distribution of a single numerical variable. There should be no "gaps" between bars in a histogram (unlike bar graphs, discussed later), because a gap represents no observations in that range.

Below is some generic code for creating histograms. You can replace the template code with a data frame and the quantitative variable you're interested in.

```
ggplot(data = dataset_name) +
  geom_histogram(aes(x = quant_var))
```

Notice that R prints a message telling you that the default number of bins was used and that this may not be ideal. You can fix this by specifying either of the following two arguments within `geom_histogram()` (but outside of `aes()`):

1. `bins` = __ argument to specify how many bins your histogram should have or
2. `binwidth` = __ argument to specify how wide each bin should be.

You may also want to use the `boundary` = __ argument to say where a bin should start (e.g., start at 0). You only need to specify this for one bin because all of the rest will follow from that.

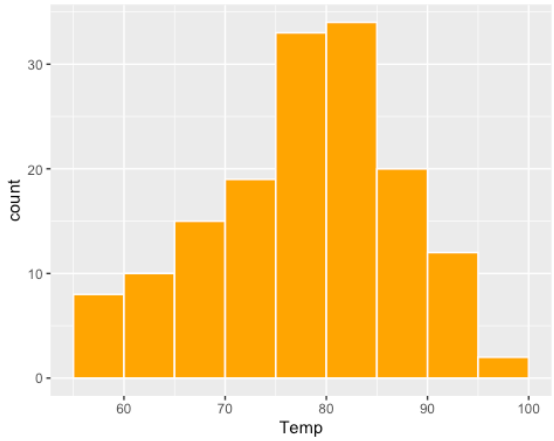
If you want to add color to your graph, you can use

1. `fill` = __ to change the color inside the bars or
2. `color` = __ to change the color of the lines around the bars.

Replace __ with a color name in quotes (like "violet"). Many common color names will work or you can see a complete list of options here.

(Note: Like with scatterplots, the `alpha` argument can be used to make the bars transparent.)

Practice: See if you can use what you've learned to recreate the graph below. It uses the `airquality` data frame that is built into R. Instead of trying to get the graph perfect right away, pick one element at a time to work on.



See notes04.R.

(Side-by-side) Boxplots

Review: What does a boxplot graph?

A single boxplot graphs the distribution of a numerical variable. Side-by-side boxplots allow you to compare the distribution across multiple groups (in a categorical variable).

Boxplots show the distribution by splitting the data into quarters via what we call the **five-number summary**, which consists of the following values:

1. The minimum (bottom of the line)
2. The first quartile, Q1 (the bottom line of the box)
3. The median (the thick line inside the box)
4. The third quartile, Q3 (the top line of the box)
5. The maximum (top of the line)

These numbers are used to split our data into fourths. In other words, 25% of the data is between the minimum and Q1, 25% between Q1 and the median, 25% between the median and Q3, and 25% between Q3 and the maximum.

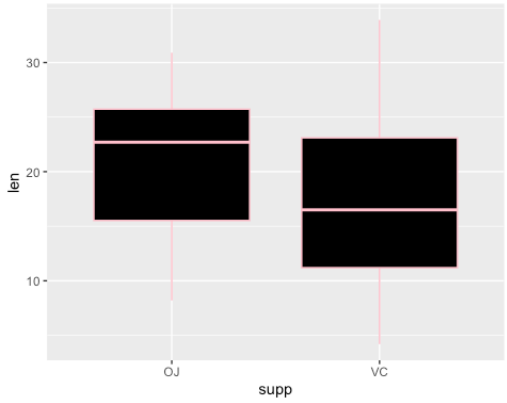
While it's possible to make a single boxplot, these graphs are most often used to compare distributions across multiple groups. Below is some generic code for creating side-by-side boxplots. You can replace the template code with a data frame and your quantitative and categorical variables to make your own.

```
ggplot(data = dataset_name) +
  geom_boxplot(aes(x = cat_var,
                   y = quant_var))
```

(**Note**: Want horizontal boxplots instead of vertical ones? Just switch x and y!)

Like with histograms, the `fill` and `color` arguments can be used to change the inside of the boxes and the lines around the boxes respectively (and `alpha` can be used to make the boxes transparent).

Practice: See if you can use what you've learned to create the following graph that uses the `ToothGrowth` dataset.



See notes04.R.

Question: What does the graph above show?

This graph shows us that the lengths values tend to be longer for the OJ supplement than for the vitamin C supplement. Note: this isn't the same thing as saying OJ is always longer, because there is a lot of overlap still between the boxes.

Bar graphs

Review: What does a bar graph show?

Bar graphs show the distribution of a categorical variable.

There are two different ways to create bar graphs using the `ggplot2` package, and which method we use depends on what our data looks like.

`geom_bar()` for raw data

The first option is for "raw data" where each row in the data represents a person or object. In this situation, we use the `geom_bar()` function to add a bar graph geom to our `ggplot()` object.

Below is some generic code for creating bar graphs using this method. You can replace the template code with a data frame and your categorical variable to make your own.

```r
# Option 1 - Raw data

ggplot(data = dataset_name) +
  geom_bar(aes(x = cat_var))
```

`geom_col()` for summarized data

The second option is for summarized data in which you have one variable for your categories and one corresponding to how many people / objects are in that category. This means each row will represent many people/objects instead of a single observation like the "raw data" case. In this situation, we use the `geom_col()` function to add a bar graph geom to our `ggplot()` object.

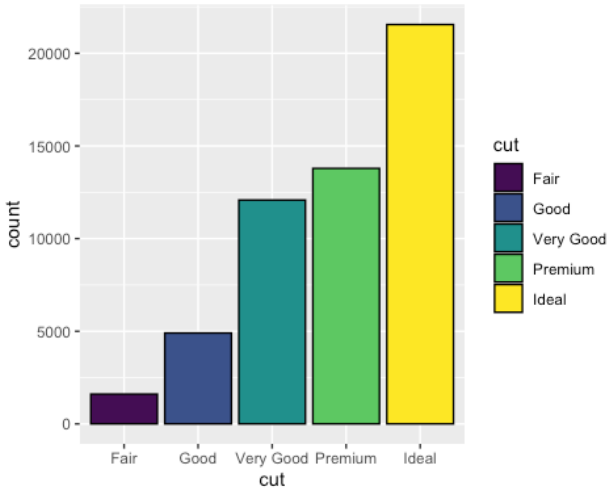
Below is some generic code for creating bar graphs using this method. You can replace the template code with a data frame and your categorical variable to make your own. You will also have a variable that contains the counts of how many people/objects are in each category.

```r
# Option 2 - Summarized data

ggplot(data = dataset_name) +
  geom_col(aes(x = cat_var,
               y = count_var))
```

As with the two previous types of graphs, you can use `fill` and `color` arguments to make the graphs look nicer.

Practice: See if you can create the following graph that uses the `diamonds` dataset in the `ggplot2` package. You may want to look at the help file for the data set to decide if you need to use `geom_bar()` or `geom_col()`. (**Note**: Rather than make each bar correspond to the same color, I mapped a variable to the bar color. How do you think I could do that in my code?)



See notes04.R. Note, if we look at the diamonds data, we can see that this is raw data (each row represents a diamond, not a category). Because of this, we used geom_bar().

Adding additional variables to our graphs

In Activity 04, we will see some examples of how additional variables can be added to specific types of graphs. However, for most arguments that we can specify outside `aes()`, we can also choose to specify them inside `aes()` to map a variable to that element of the graph.

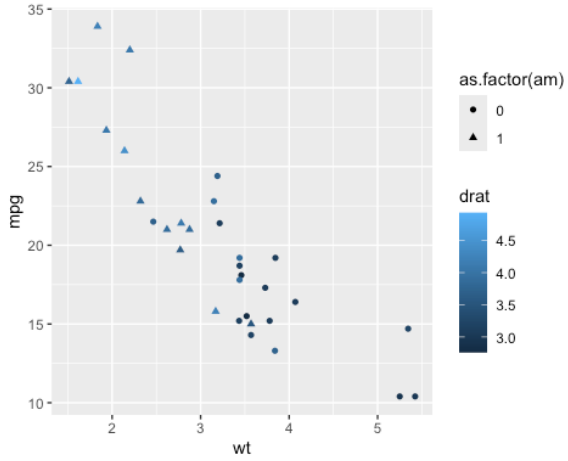
For example, I may want to make the bars of a bar graph have colors corresponding to their category instead of all having the same color. To do that, I assign my x variable to the `fill` argument too (inside of `aes()`).

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

In a scatterplot, I may want the size, color, shape, or transparency of my points to correspond with a different variable in my data. If that variable is treated as categorical, I will have different values for each category. If that variable is treated as numeric, I will get a continuum of values. Below are two examples.

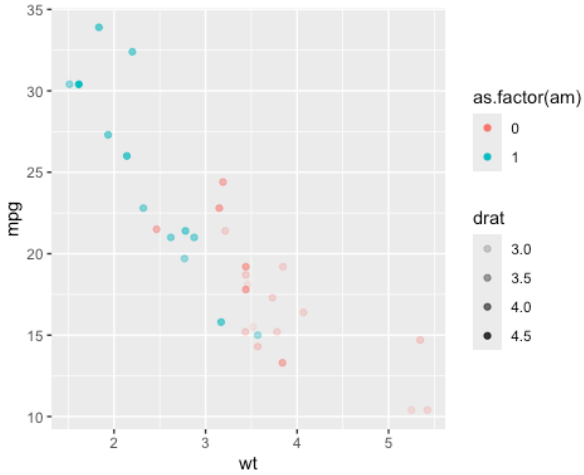
```r
# Continuum of colors but two separate shapes
ggplot(mtcars) +
  geom_point(aes(x = wt,
                 y = mpg,
                 color = drat,
                 shape = as.factor(am)))
```

Color is on a continuous scale because drat is a numeric variable in R. The shape argument requires categorical variables, so we had to use the as.factor() function to convert the numeric am variable into categories.




```r
# Two separate colors but continuum of transparencies
ggplot(mtcars) +
  geom_point(aes(x = wt,
                 y = mpg,
                 color = as.factor(am),
                 alpha = drat))
```

Transparency (alpha) is on a continuous scale because drat is numeric in R. To make sure we have two colors (instead of a continuous scale), we converted am to categorical using as.factor().

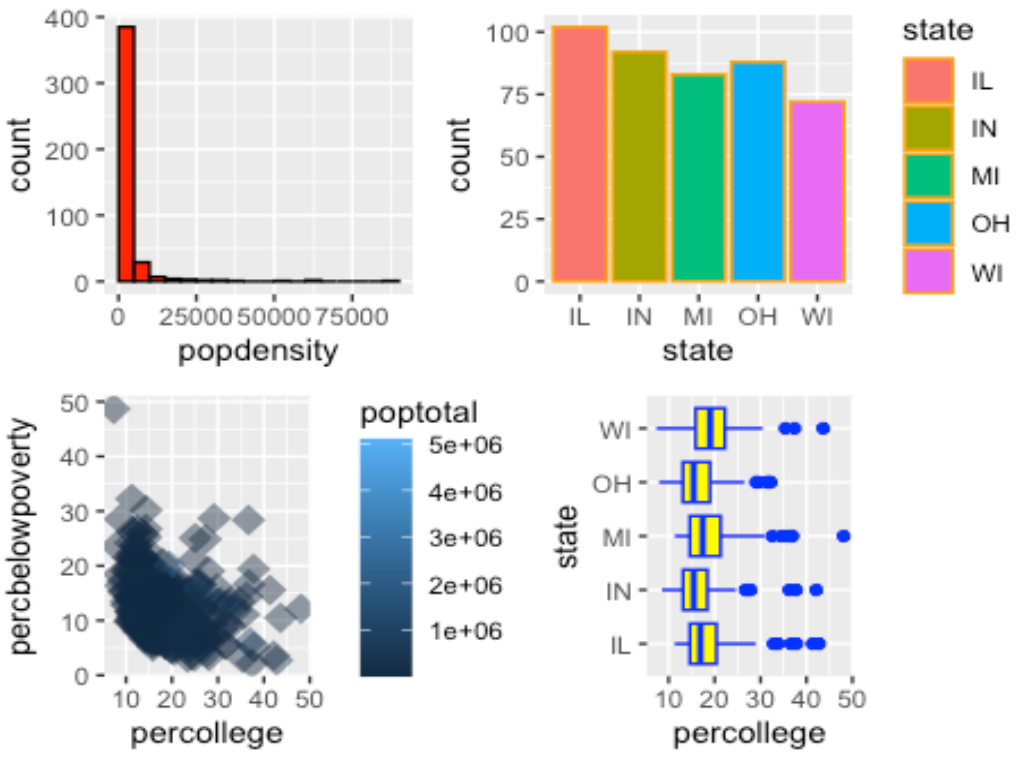


We will continue to explore this in Activity 04. In Notes 05, we will learn how to improve our graphs by editing the theme, labels, color values, and more.

Revisiting the Learning Goals for Notes 04

- Understand the `ggplot2` grammar of graphics including how to use the `aes()` function.
    - Which function do we start each graph with?
    - How do we know whether an argument like `color` should go inside or outside of the `aes()` function?

- Be able to use the `ggplot2` package to create histograms, boxplots, bar graphs, and scatterplots.
    - Attempt to recreate each of the graphs below (that uses the `midwest` data frame from the `ggplot2` package) to the best of your ability.



- Be able to display information about additional variables in these graphs using color and other features.
    - See examples above