Week 2: Data Visualization with ggplot

*Making pretty pictures!*

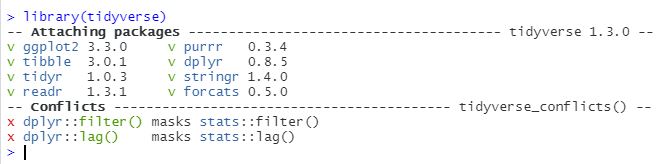
This week, we’re diving right into making plots with ggplot**,** a member of the tidyverse that we installed last time. Now, if you do a bit of googling about data visualization, you’ll quickly find that people have a LOT to say about it. There are lots of different philosophies, best practices, arguments… it’s fascinating stuff, but I’m not going to even try to go into all that. Coming from a scientific background, I have my own opinions about what makes a “good” visual, and I’m sure you all have your own thoughts. The authors of *R4DS* have included a link to an article about the theory of ggplot (<http://vita.had.co.nz/papers/layered-grammar.pdf>). If you’re interested, go ahead and give it a look! But for now, I’m going to focus on how to actually use ggplot.

A quick note before we get started: there’s going to be some stuff we do that isn’t going to make a ton of sense, at least not at first. That’s the downside of starting with data vis versus building the foundations first… but with some googling we can figure most everything out. Now, onward!

To do anything, we’ll need to load up the tidyverse like before, since that’s where ggplot lives:

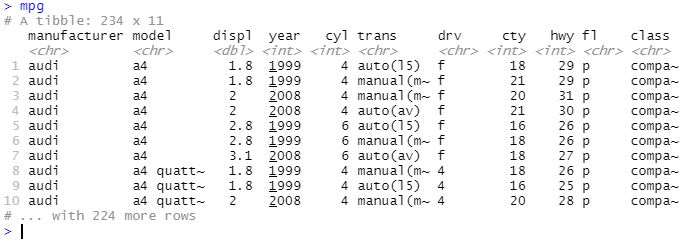
> library(tidyverse)

If you have it installed correctly, you’ll see this:



(The “conflicts” part gave me pause, but this is normal apparently).

Now we can start asking some questions! While we will presumably be learning to load and clean outside data in future chapters, for now we will take advantage of that fact that ggplot comes pre-loaded with some clean practice dataframes. I think about dataframes as kind of like spreadsheets or tables. They have two dimensions, with columns of variables and rows of observations. The first dataframe we’re working with is the mpg dataset. To see what it looks like, just type mpg into the console, and this is what you should see:



There are 11 variables, or fields, and 234 observations in this dataset. If you type ?mpg into the console, you can get some more information on each field.

The first plot *R4DS* has us make begins with typing this line into our console:

> ggplot(data = mpg) + geom\_point(mapping = aes(x = displ, y =

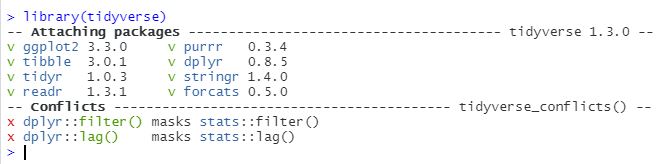
hwy))

This week, we’re diving right into making plots with ggplot**,** a member of the tidyverse that we installed last time. Now, if you do a bit of googling about data visualization, you’ll quickly find that people have a LOT to say about it. There are lots of different philosophies, best practices, arguments… it’s fascinating stuff, but I’m not going to even try to go into all that. Coming from a scientific background, I have my own opinions about what makes a “good” visual, and I’m sure you all have your own thoughts. The authors of *R4DS* have included a link to an article about the theory of ggplot (<http://vita.had.co.nz/papers/layered-grammar.pdf>). If you’re interested, go ahead and give it a look! But for now, I’m going to focus on how to actually use ggplot.

Every time we want to use the tidyverse, which we do because that’s where ggplot is, we’ll need to “load” it. Yep, every time you open RStudio and want to use ggplot you’ll need to type this:

> library(tidyverse)

If you have it installed correctly, you’ll see this:



(The “conflicts” part gave me pause, but this is normal apparently).

So, our first research question! This is the whole point right? We’re learning to explore data so that we can answer questions and find new ones.

*“Do cars with big engines use more fuel than cars with small engines?”*

Without looking ahead, try to think of what information we might need to answer this question (I’ll do the same thing).

Ready yet?

We basically need two pieces of information (or, really, many iterations of two categories of information). We need a list of cars and, or each car, the amount of fuel it consumes and the size of its engine. Let’s see if we’re right!

Conveniently, ggplot actually contains a dataframe called “mpg” that should contain the information we need.

Wait though, hold up- let’s talk about what a *dataframe* is. At its most simple, I think of a dataframe like a spreadsheet or table. It has two dimensions, with columns of variables and rows of observations.

With that out of the way, let’s take a look at the **mpg** dataframe. Type this into RStudio:

> ~~ggplot::mpg~~

> ggplot2::mpg

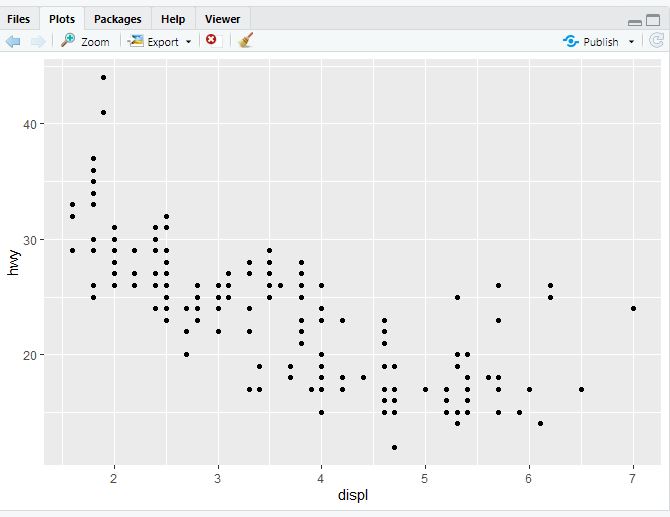
So it looks like there’s a lot of data here: 11 different variables. But we really only need two: engine size and the fuel use. In this dataframe, the engine size is in *displ* column. Why is it called that? I have no idea. Luckily *R4DS* just gives us the answer for now. The fuel use, or fuel efficiency, is divided into two variables, city (*cty*) and highway (*hwy*) efficiency. We’ll use highway efficiency for this.

> ggplot(data = mpg) + geom\_point(mapping = aes(x = displ, y =

hwy))

Going through the syntax: ggplot is your function (note that you don’t include the 2 here, the package must be called ggplot2 but the function you call is ggplot… whatever). The part in parentheses (the “argument”) specifies what data you are using- the mpg dataframe in this case. This is the first layer. If you just run ggplot(data=mpg), stuff will happen, but all you’ll see is a gray box. Not super useful. **The plus sign means we’re adding a layer (is this right???).** The geom\_point() function specifics that we’re making a scatterplot. It’s argument is *mapping* (sensing a pattern?), which basically just says “this is where I’m specifying how my data will look.” The next bit, *aes* (or “aesthetic speficifications”) is where you’ll actually *specify* the *aesthetics* (in this example, all we specified is that *displ* is on the x axis and that *hwy* is on the y axis). In fact, as we go forward, we can use this basic template for any geom, or “geometric object,” changing the dataset, geom function, or mappings.

Here’s what you get if you run that code:



Simple right? Try a few different variables and see what you get. If you play around, you’ll notice that while some combinations make sense, others might look a little weird… think about why that may be! For example, does it make sense to make a scatterplot of manufacturer vs. model? Why or why not?

I’ll answer that one for you… it doesn’t make sense, because manufacturer and model are both examples of *categorical* data. Variables like *hwy* and *displ* are *continuous*. That doesn’t mean you can make plots and charts of categorical data… it just means you need to find the right one! Scatterplots are generally used to plot two continuous variables.

**Note: In facet section, make a point of how the grid is oriented can change the major take-away… give an example, then encourage readers to explore.**

<https://rstudio.com/resources/cheatsheets/> (link in book is broken)

* In statistical transformation section, address the “y=depth” issue but don’t go into too much detail-not a statistics blog.
* Use ..name.. syntax to map stat variables to aesthetics.
* In the bar chart proportion section and adding the “group” aesthetic, discuss that it doesn’t matter what value you put there… just having it there will override the default to group by “cut.”