Basic graphics with tidy data, or: why starting tidy makes life in R palatable

Elizabeth Wickes, wickes1@illinois.edu, @elliewix

Spreadsheet activity

Handout for the "Creating Tidy Data" workshop of the 2016 Midwest Data Librarian Symposium (MDLS). GitHub Repository: https://github.com/elliewix/but-i-just-want-to-make-a-graph

Open up the Excel file included within the GitHub repository. Take a few moments to explore it, the data included, structure, etc. Now, try to think about how you could reorganize it into a single data file. Work with 1-2 people sitting next to you. Determine what the new headers should be and write them down below.

Create a new tab and begin recreating the new data file if you have time remaining.

R graphing activity

You should follow along in RStudio with these commands. You will be using a data file called `cleanedsurvey.csv` located within the GitHub repository.

Load the library

You should have installed this with `install.library("ggplot2")`. You call `library()` with the module name, without quotes, to load the library.

```
library(ggplot2)
```

Load the data

Use Tools -> Import Dataset -> From text file...

Select the files, and accept all the default options in the wizard. You can play around with these options and view the changes in the preview box.

```
cleanedsurvey <- read.csv("cleanedsurvey.csv")
```

This creates an object in memory, called `cleanedsurvey`, that represents the data.frame containing the data.

Make a basic histogram

This will be using ggplot2 to create a histogram. They usually start with a `ggplot()` statement that declares where the data frame is, then you use `+` to add graphic elements to the plot. You can layer these elements as desired.

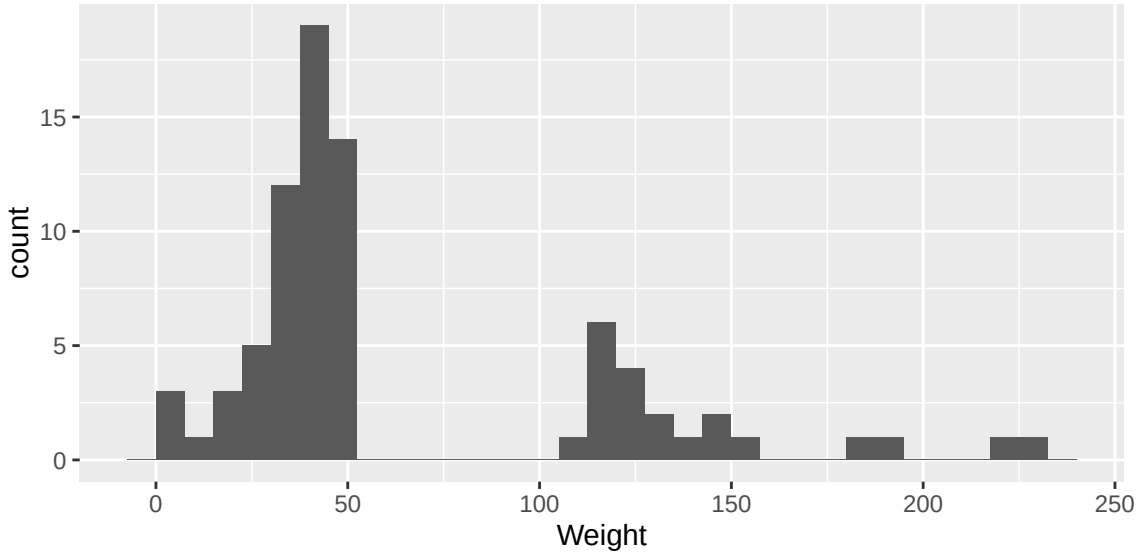
You can see in the code below that we are declaring `data = cleanedsurvey` in `ggplot()`, but not selecting any of the data columns yet. Just providing the data frame.

`+` then adds a `geom_histogram()` graphical layer, which is a histogram. We tell it that we want `x = Weight`. `Weight` is the column name without quotes. The case must also match what appears in the data.frame it is contained within. There are many types of graphical elements (http://docs.ggplot2.org/current/) within the ggplot2 library.

You'll see below that the plot is giving us a message that it is using a default bin value.

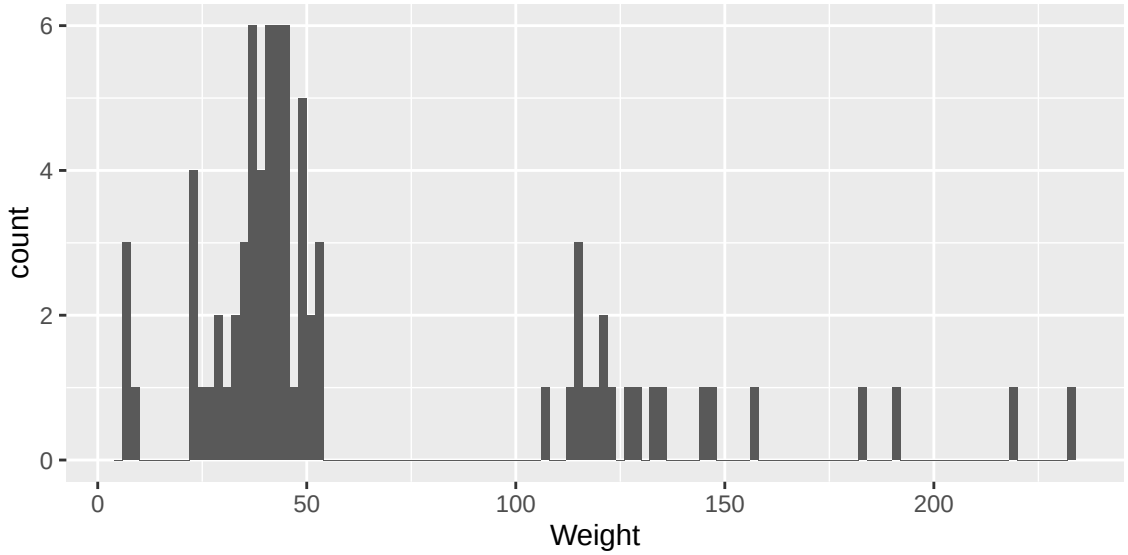
```
ggplot(data = cleanedsurvey) + geom_histogram(aes(x = Weight))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



As that error message hinted, `geom_histogram()` has additional options that you can play with. Place your curser just inside the last set of `()` press tab to see them. Let's play with binwidth.

```
ggplot(data = cleanedsurvey) + geom_histogram(aes(x = Weight), binwidth = 2)
```



Now you

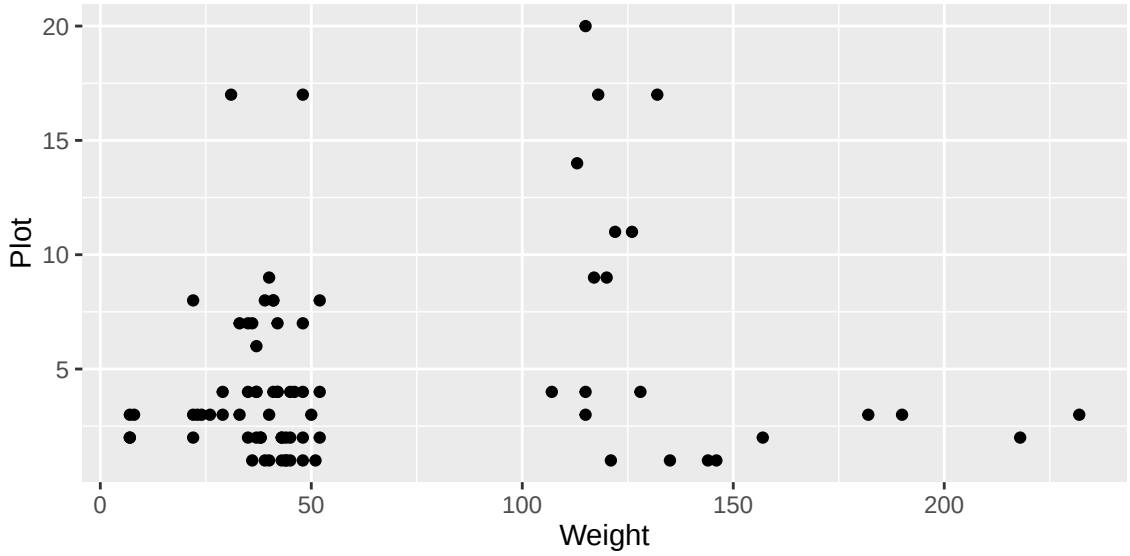
- Change some of the binwidth values to find the right balance for the data you are looking at
- Change the `x =` value to other columns and see what works and what doesn't.
- Which columns worked? Which columns didn't?
- What sort of errors did you get when you tried a column that didn't work?
- What other sort of errors did you see in this process?

Scatterplots and grouping

We can also use scatterplots to look at numerical data.

You can see that the syntax is already changing on us, but at least the x and y labels are showing our nice column headers.

```
ggplot(cleanedsurvey, aes(x = Weight, y = Plot)) + geom_point()
```

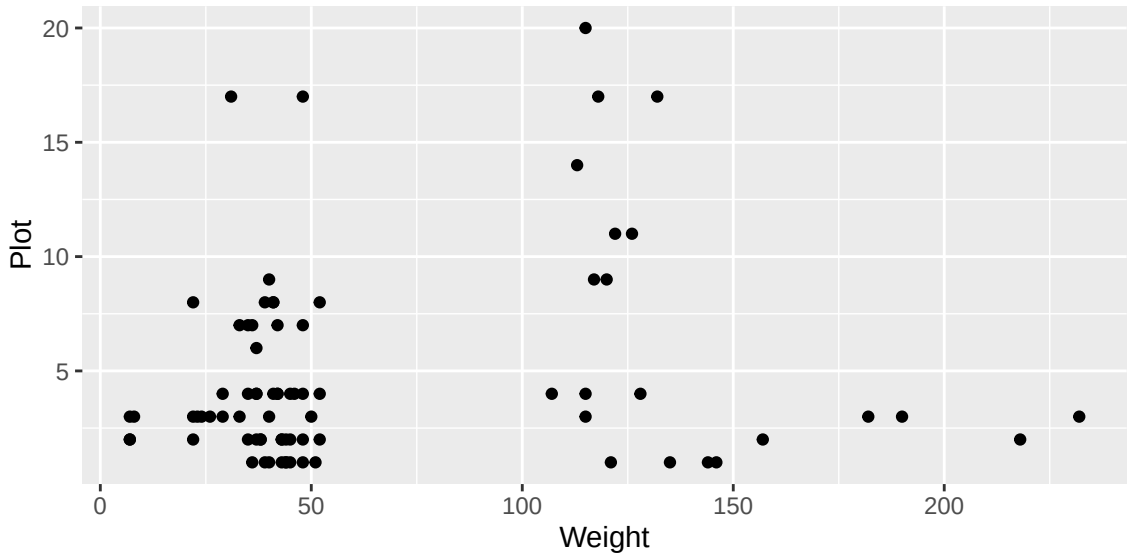


This is nice, but hard to tell a bit of what's going on. Let's group this in a few ways.

Because our code is getting long, we can save our basic ggplot code as a variable and reference it. We're going to call our ggplot() object `p`. Nothing will happen when we first create `p` because we must call p to make it appear. Calling `p` alone will show a blank graph. We need to add our graphical element layer to see the actual chart.

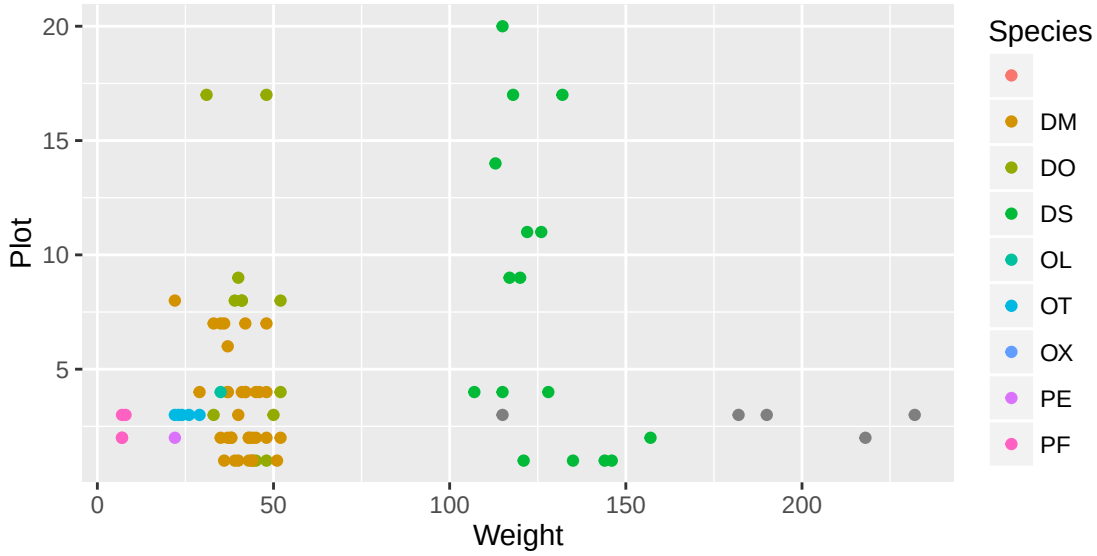
```
p = ggplot(cleanedsurvey, aes(x = Weight, y = Plot)) # nothing will be plotted yet

p + geom_point() # the plot will appear when you evaluate it with a geometry element
```



Now we're seeing the power of having our data already in tidy format. Because all our labels are in a single column, associated with each observation, we can just point R to that column and have nice labels and colors without too much work.

```
p + geom_point(aes(color = Species)) # color by the Species column
```



Now you

- Try changing the color variable to other columns within the data file.
- What worked? What didn't?
- What were the errors that you encountered?