Module 2: Plotting with `ggplot2`

Ellen Bledsoe

2023-02-22

Plotting with `ggplot2`

So far, we've used the base R plotting syntax. While quick plots in base R can still be really useful ways to do preliminary data exploration and visualization, we often want plots that go beyond the basics without too much additional effort. This is where `ggplot2` comes in and really shines!

Example

Before we get into the nitty-gritty of how `ggplot2` works, Let's run an example using the data about our sick crew members from earlier.

First, we need to load in both the `tidyverse` package and our data. We can remind oursevles what the data look like using the `head()` function.

```
# load package
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0     v purrr   1.0.1
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.5.0
## v readr   2.1.3     v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
# load data
sick <- read_csv("../data/sick_data.csv")
```

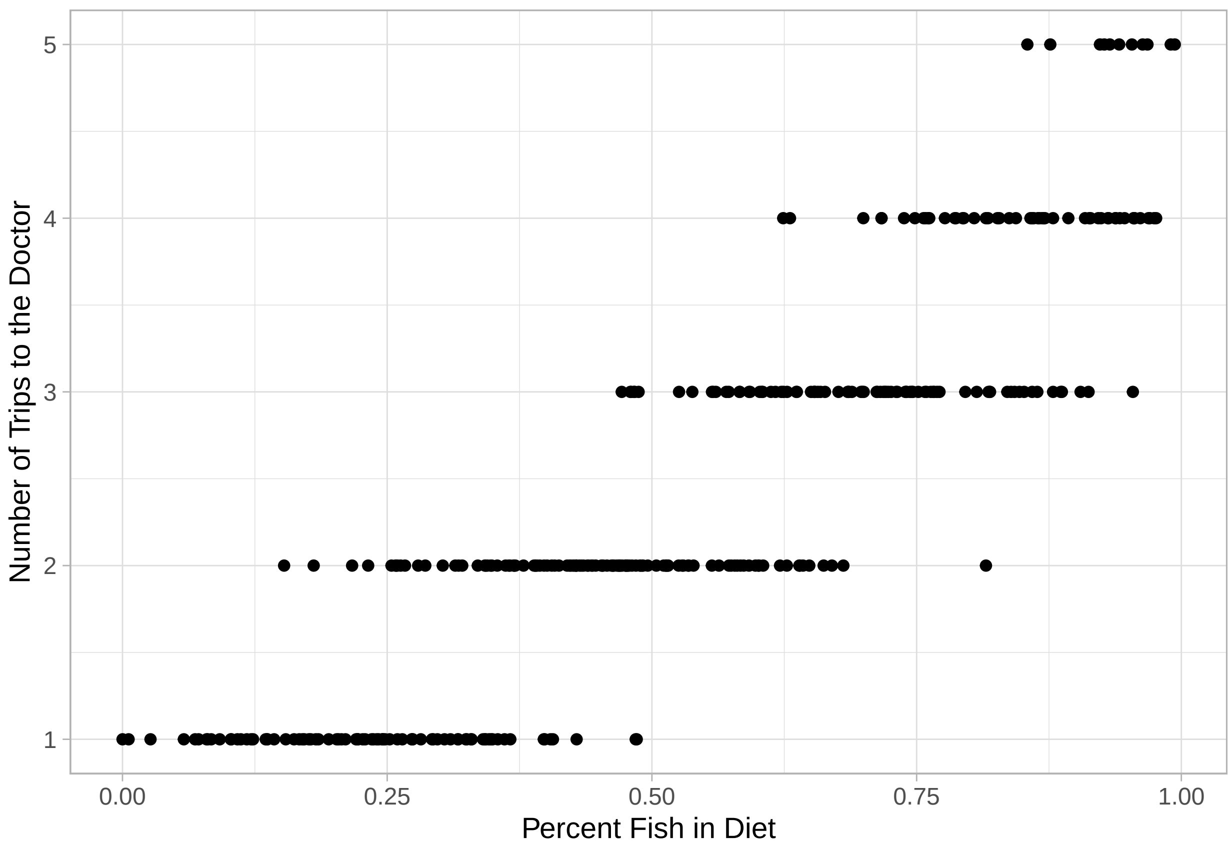
```
## Rows: 349 Columns: 10
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): last, first, sex, specialties
## dbl (6): age, height_cm, weight_kg, perc_fish, perc_plant, doctor_trips
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(sick)
```

```
## # A tibble: 6 x 10
##   last          first sex     age heigh~1 weigh~2 speci~3 perc_~4 perc_~5 docto~6
##   <chr>         <chr> <chr> <dbl>   <dbl>   <dbl> <chr>     <dbl>   <dbl>   <dbl>
## 1 Gonzalez      Ange~ M        35    169.    51.4 Hydrol~   0.994 0.00620       5
## 2 Navratil      John  M        19    112.    96.3 Geneti~   0.297 0.703         1
## 3 Duff          Josh~ M        26    133.    52.1 Hortic~   0.514 0.486         2
## 4 Dottson       Juli~ M        36    140.    52.6 Climat~   0.686 0.314         3
## 5 al-Sultana    Mune~ M        26    194.    52.2 Geology   0.292 0.708         1
## 6 Gallegos Pe~ Rich~ M        29    153.    98.1 Climat~   0.329 0.671         1
## # ... with abbreviated variable names 1: height_cm, 2: weight_kg,
## #   3: specialties, 4: perc_fish, 5: perc_plant, 6: doctor_trips
```

Here is code to make a scatter plot of the relationship between percent fish in diets and how many trips to the doctor.

```
ggplot(sick, aes(x = perc_fish, y = doctor_trips)) +
  geom_point() +
  labs(x = "Percent Fish in Diet",
       y = "Number of Trips to the Doctor") +
  theme_light()
```



Nice, right? In the next few classes, we will really start to see the power of `ggplot`. For now, though, let's focus on how this works.

ggplot2

The package `ggplot2` is part of the `tidyverse`.

Here are some resources you might find helpful now or in the future:

- ggplot2 Book
- UC Business Analytics ggplot2 intro
- R for Data Science Data Visualization chapter

The `gg` in `ggplot2` stands for "Grammar of Graphics." The "grammar" part is based on an idea that all statistical plots have the same fundamental features: data and mapping (and specific components of mapping. The design is that you work iteratively, building up layer upon layer until you have your final plot.

The typical structure looks like this:

```
# ggplot(data = <DATA>, mapping = aes(<MAPPINGS>)) +
# <GEOM_FUNCTION>()
```

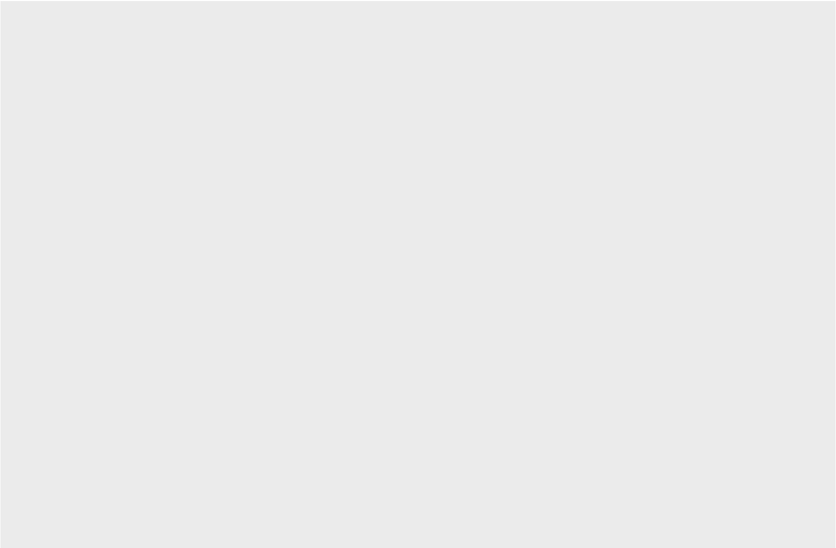
A few things to note:

- we always start with the `ggplot()` function
- we specify the dataset we want to use
- we specify the mappings (x- and y-axes and some other bits) with the `aes()` function
- we use a `+` to add layers
- we specify the type of plot, or `geom` using one of many possible geom functions
- we use the `labs()` function to clean up the labels
- we add a `theme` function to make it look extra pretty

Let's iteratively build up to the plot we have made above:

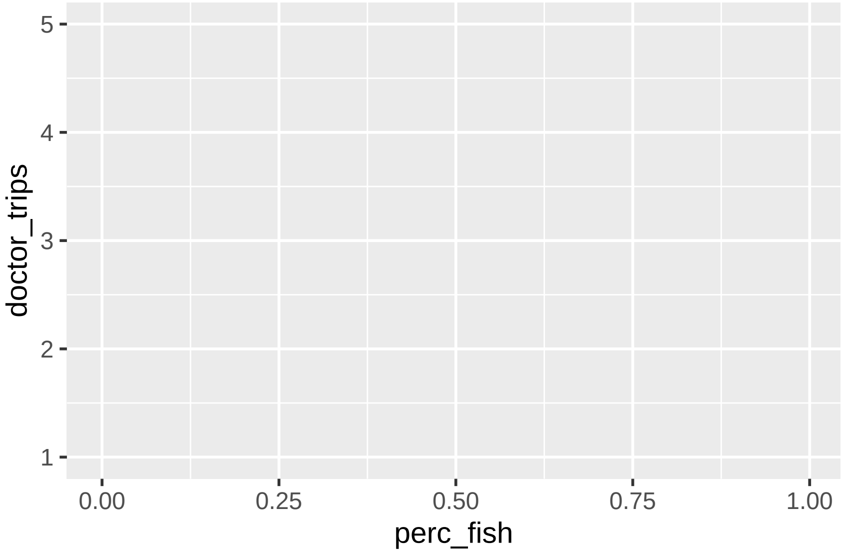
1) Specify the data

```
ggplot(data = sick)
```



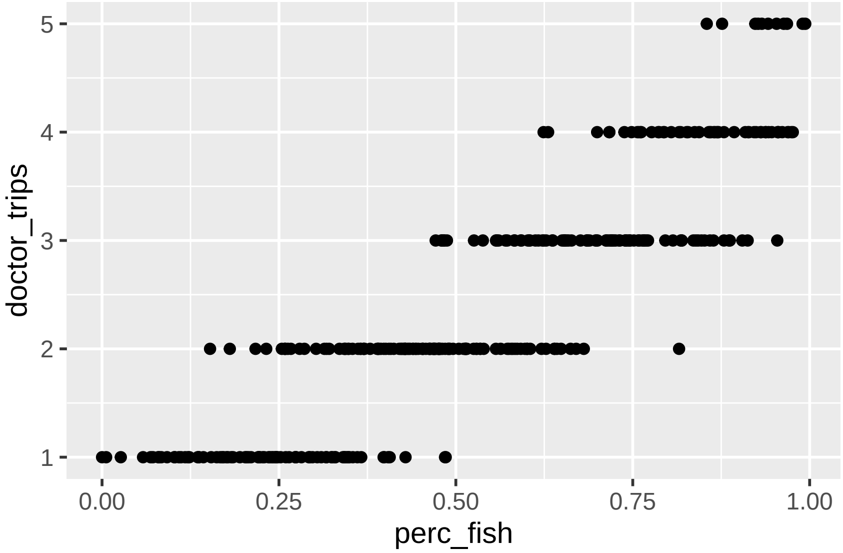
2) Specify the x-axis (horizontal) and the y-axis (vertical) in the `aes()` function.

```
ggplot(data = sick, mapping = aes(x = perc_fish, y = doctor_trips))
```



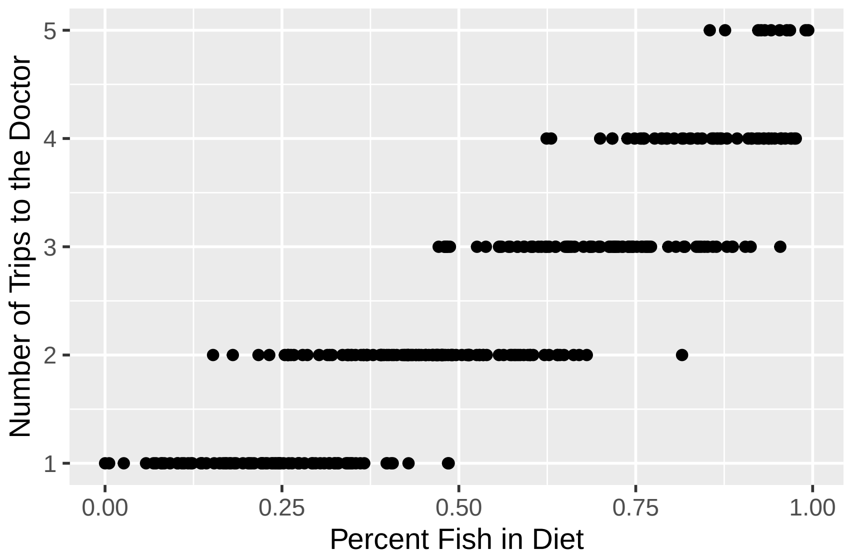
3) Add the type of plot we want using a `geom` function.

```
ggplot(data = sick, mapping = aes(x = perc_fish, y = doctor_trips)) +
  geom_point()
```



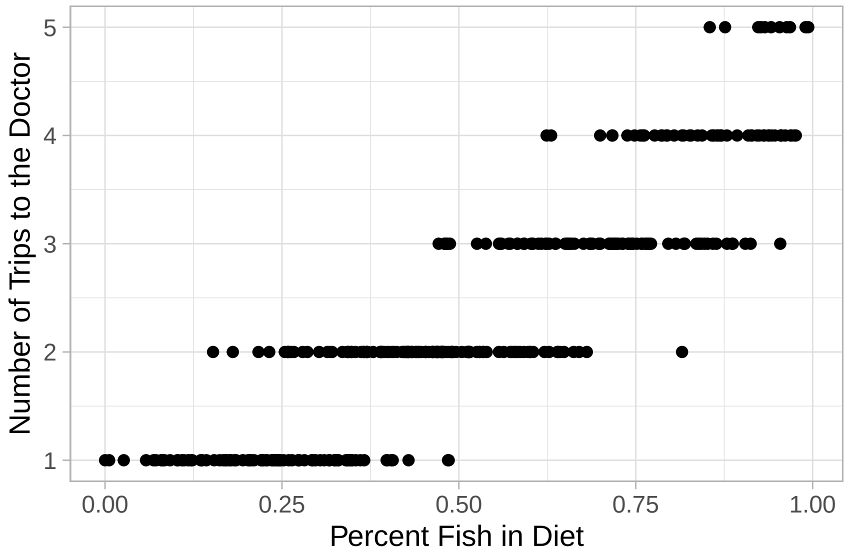
4) Clean up the axis labels with the `lab()` function so they are more easily interpreted.

```
ggplot(data = sick, mapping = aes(x = perc_fish, y = doctor_trips)) +
  geom_point() +
  labs(x = "Percent Fish in Diet",
       y = "Number of Trips to the Doctor")
```



5) Choose a `theme` function to make the plot more aesthetically pleasing.

```
# theme_bw(), theme_classic(), and theme_light() are my favorites
ggplot(sick, aes(x = perc_fish, y = doctor_trips)) +
  geom_point() +
  labs(x = "Percent Fish in Diet",
       y = "Number of Trips to the Doctor") +
  theme_light()
```


Number of Trips to the Doctor
Percent Fish in Diet