Data Visualization using ggplot2

2024-01-09

This is adapted from content in the book *R for Data Science*.
https://r4ds.hadley.nz/data-visualize.html

Getting started with ggplot

Note: This walk-through is to give you a taste of data visualization tools available to you in R. Many items which are given explicitly in the online textbook will be presented verbally or skipped entirely. If you wish to review, or for a more in-depth presentation with exercises, please use the online text whose url is above. An internet search will locate the solutions to said exercises should you get stuck!

The "gg" in ggplot2 stands for "grammar of graphics", a coherent system for describing and building graphs. We will use the `palmerpenguins` package and its data to explore ggplot capabilities.

First: `install.packages("palmerpenguins")` and `install.packages("ggthemes")` if you don't already have these packages.

```
library(tidyverse)
library(ggthemes)
library(palmerpenguins)
```

You can type ?penguins or ?palmerpenguins::penguins to get info on the data set.

In this data set, a variable refers to an attribute of all the penguins, and an observation refers to all the attributes of a single penguin.

Type the name of the data frame in the console and R will print a preview of its contents. Note that it says tibble on top of this preview. In the tidyverse, we use special data frames called tibbles.

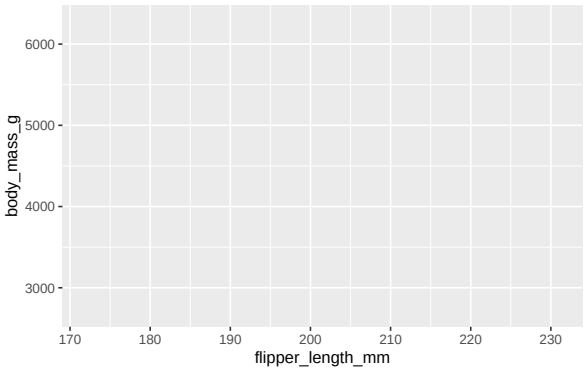
Our goal today is to slowly build up a fun graphic while learning how the grammar works to put it all together for us.

```
ggplot(data = penguins)
```


At first, you shouldn't get anything exciting, because you're just taking out a blank canvas to start playing with.

We need "aesthetic" properties to get us started in the right direction.

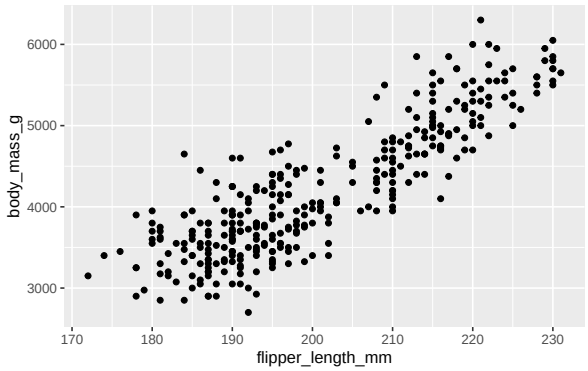
```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm, y = body_mass_g)
)
```



That's still a little underwhelming.

Let's add a "geom" or geometric object. This one will produce a basic scatter plot.

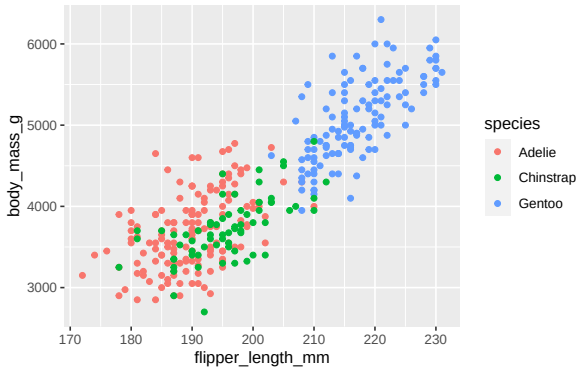
```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm, y = body_mass_g)
  )+
  geom_point()
```



Note: Make sure the + is at the end of the line.

Now, let's add some color!

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm, y = body_mass_g, color = species)
) +
  geom_point()
```

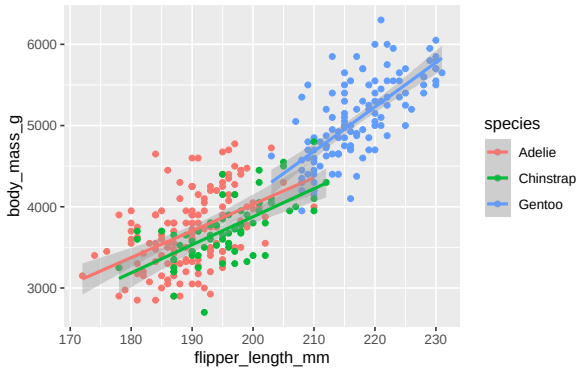


Note: Try to map a continuous variable to color, size, and shape. See what happens! These aesthetics behave differently for categorical versus continuous variables.

You can easily add best-fit lines.

(Also try this code without the "lm". What happens? Why?)

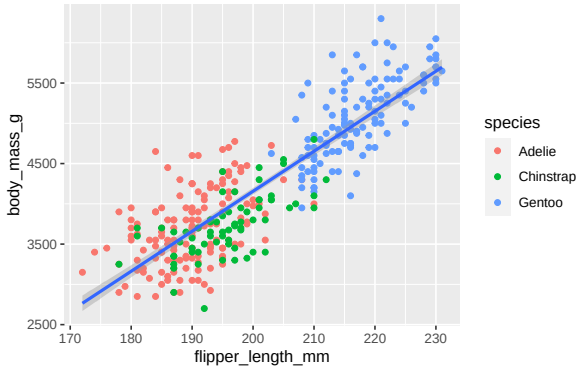
```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm, y = body_mass_g, color = species)
) +
  geom_point() +
  geom_smooth(method = "lm")
```



This might be precisely what you intended to do, but maybe it's not.

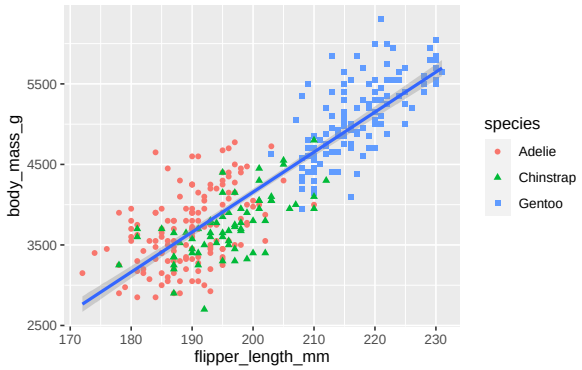
This time, let's break this into two separate plot layers, one on top of the other, so we can do a best-fit on the overall set.

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm, y = body_mass_g)
) +
  geom_point(mapping = aes(color = species)) +
  geom_smooth(method = "lm")
```



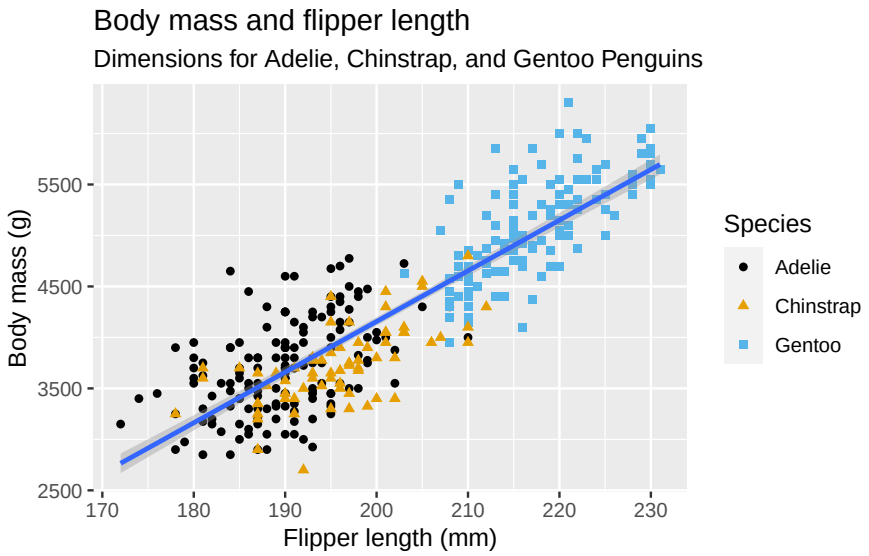
Artistically, we can do better, though. We can have both color and shape to guide our eyes!

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm, y = body_mass_g)
) +
  geom_point(mapping = aes(color = species, shape = species)) +
  geom_smooth(method = "lm")
```



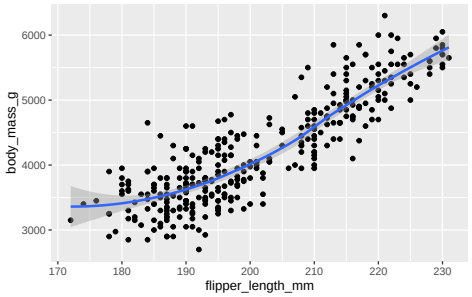
There are a lot of additional upgrades we can make to this simple scatter plot. Compare the code with the output and discuss what changed and why/when it's better this way.

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm, y = body_mass_g)
) +
  geom_point(aes(color = species, shape = species)) +
  geom_smooth(method = "lm") +
  labs(
    title = "Body mass and flipper length",
    subtitle = "Dimensions for Adelie, Chinstrap, and Gentoo Penguins",
    x = "Flipper length (mm)", y = "Body mass (g)",
    color = "Species", shape = "Species"
  ) +
  scale_color_colorblind()
```

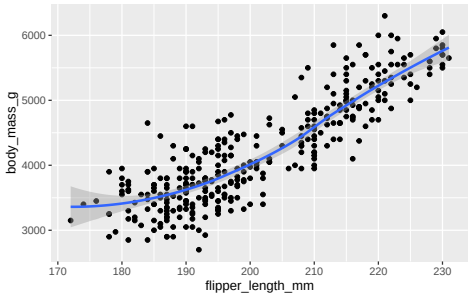


Compare the two pieces of code and determine if the resulting plots are the same or not.

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm, y = body_mass_g)
) +
  geom_point() +
  geom_smooth()
```

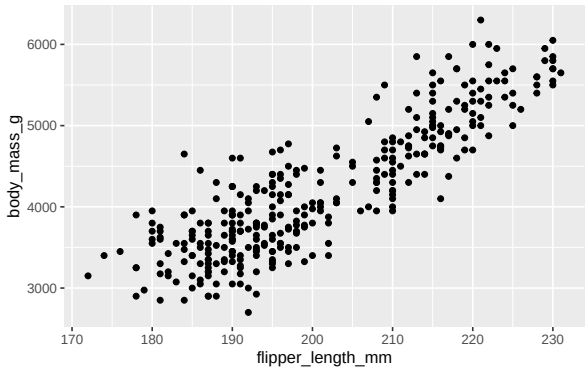



```
ggplot() +
  geom_point(
    data = penguins,
    mapping = aes(x = flipper_length_mm, y = body_mass_g)
  ) +
  geom_smooth(
    data = penguins,
    mapping = aes(x = flipper_length_mm, y = body_mass_g)
  )
```

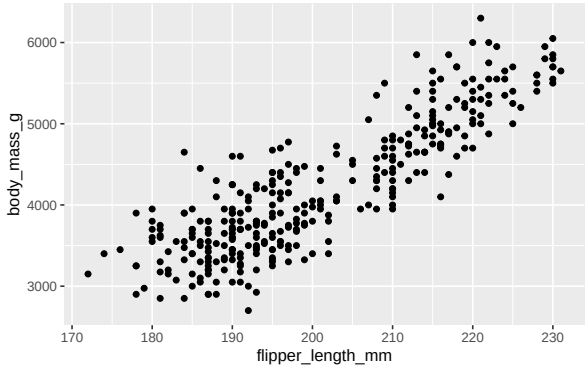


Are the following two plots the same or different, and why?

```
ggplot(
  data = penguins,
  mapping = aes(x = flipper_length_mm, y = body_mass_g)
) +
  geom_point()
```

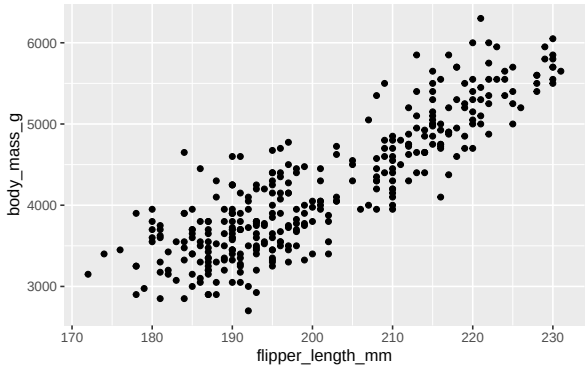



```
ggplot(penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point()
```

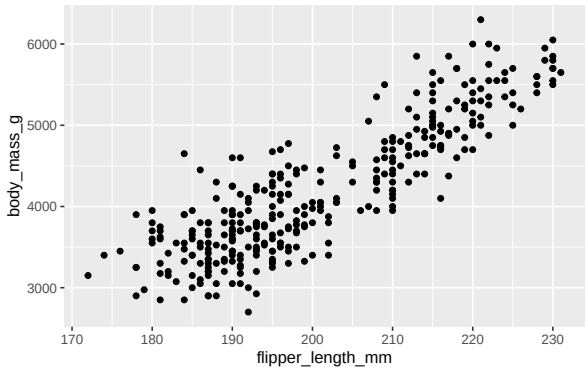


What about these two?

```
penguins |>
  ggplot(aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point()
```

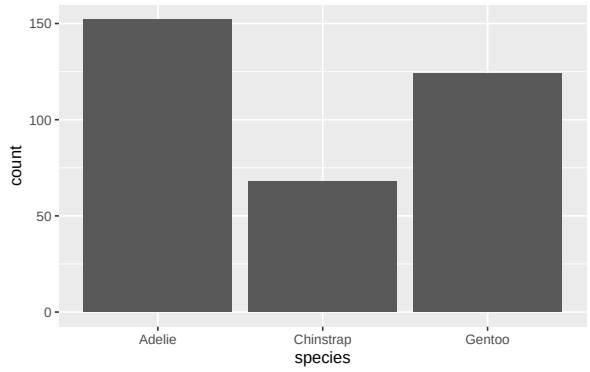



```
penguins %>%
  ggplot(aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point()
```

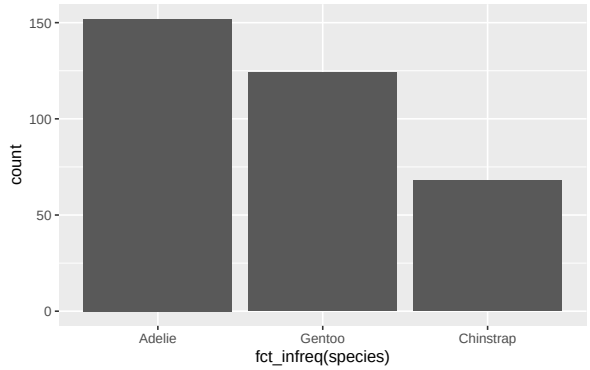


Barplots for categorical variables

```
ggplot(penguins, aes(x = species)) +
  geom_bar()
```



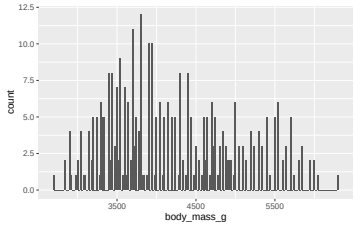

```
ggplot(penguins, aes(x = fct_infreq(species))) +
  geom_bar()
```



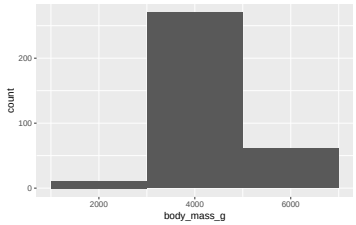
Frequency histograms for bins of numerical values

```
ggplot(penguins, aes(x = body_mass_g)) +
  geom_histogram(binwidth = 200)
```


```
ggplot(penguins, aes(x = body_mass_g)) +
  geom_histogram(binwidth = 20)
```




```
ggplot(penguins, aes(x = body_mass_g)) +
  geom_histogram(binwidth = 2000)
```

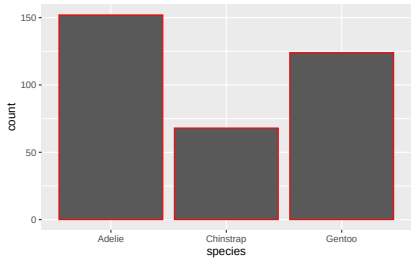


A curve can be fit also

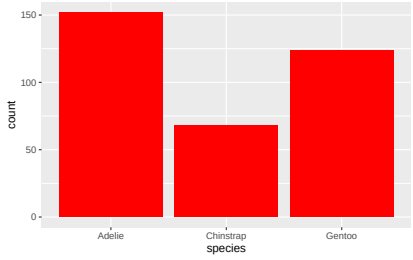
```
ggplot(penguins, aes(x = body_mass_g)) +
  geom_density()
```

Compare and discuss:

```
ggplot(penguins, aes(x = species)) +
  geom_bar(color = "red")
```

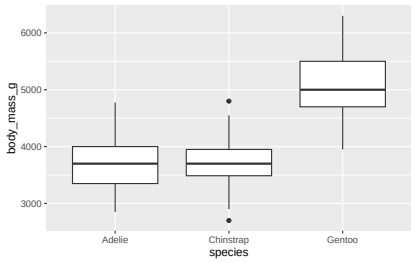



```
ggplot(penguins, aes(x = species)) +
  geom_bar(fill = "red")
```



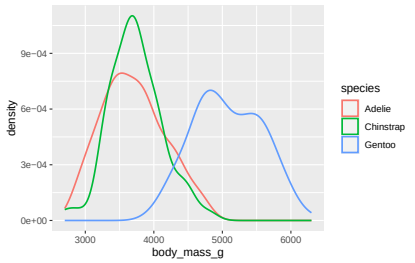
Boxplots - a categorical with a numerical

```
ggplot(penguins, aes(x = species, y = body_mass_g)) +
  geom_boxplot()
```

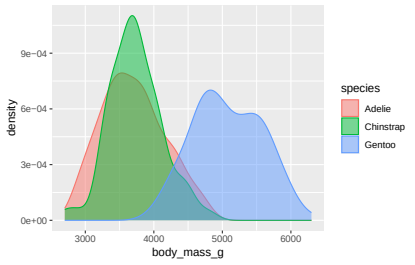


Density Plots

```
ggplot(penguins, aes(x = body_mass_g, color = species)) +
  geom_density(linewidth = 0.75)
```

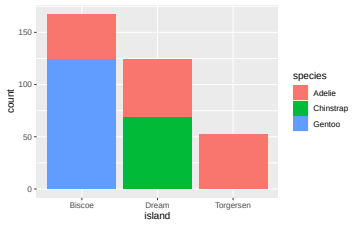



```
ggplot(penguins, aes(x = body_mass_g, color = species, fill = species)) +
  geom_density(alpha = 0.5)
```

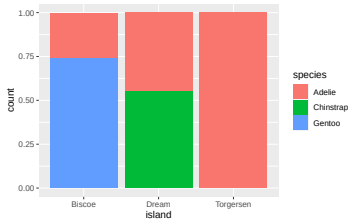


Two categorical variables

```
ggplot(penguins, aes(x = island, fill = species)) +
  geom_bar()
```

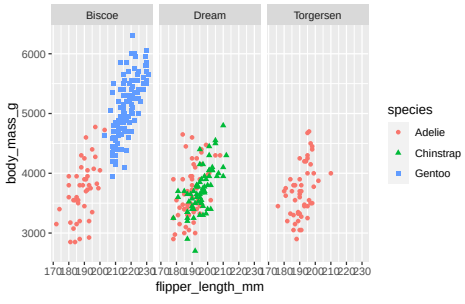



```
ggplot(penguins, aes(x = island, fill = species)) +
  geom_bar(position = "fill")
```



Facets

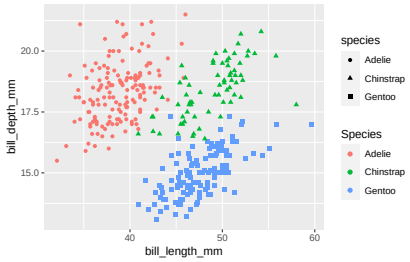
```
ggplot(penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(aes(color = species, shape = species)) +
  facet_wrap(~island)
```



Question

How can we repair the two legends so we get only one legend?

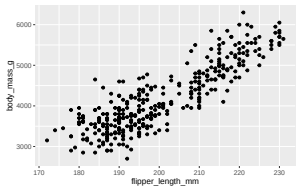
```
ggplot(
  data = penguins,
  mapping = aes(
    x = bill_length_mm, y = bill_depth_mm,
    color = species, shape = species
  )
) +
  geom_point() +
  labs(color = "Species")
```



Saving graphics for use elsewhere

Once you've made a plot, you might want to get it out of R by saving it as an image that you can use elsewhere. That's the job of ggsave(), which will save the plot most recently created to disk:

```
ggplot(penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point()
```




```
ggsave(filename = "penguin-plot.png")
```

Hadley recommends Quarto for authoring living documents which incorporate code to create diagrams with text. R-Markdown can also be used for this purpose as well, but Quarto is more versatile and is becoming the more favored choice. Either way, you can always export your artwork for use in any software.