Data Exploration Lesson: Palmer Penguins Wrap-Up

Katie Willi

2025-06-03

Review

In yesterday's lesson, we learned how to use several tidyverse functions to describe our penguins data set. Let's review some of those functions here.

First, we must load in the necessary packages for this lesson:

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(palmerpenguins)
```

Next, we can load in the built-in penguins data set from {palmerpenguins}:

```
data(penguins)
```

We can use the {tidyverse} to answer questions like, "Which penguin species has the longest flipper length?"

```
longest_flippers <- penguins %>%
  group_by(species) %>%
  summarize(mean_flipper_length = mean(flipper_length_mm, na.rm = TRUE)) %>%
  filter(mean_flipper_length == max(mean_flipper_length))
longest_flippers
```

```
## # A tibble: 1 x 2
##   species mean_flipper_length
##   <fct>                 <dbl>
## 1 Gentoo                 217.
```

We can make new columns with `mutate()`:

```
penguins <- penguins %>%
  mutate(flipper_length_cm = flipper_length_mm/10)
```

We can filter rows of the data frame. For example, filtering our penguins dataset to only Adelie penguins on Torgersen Island:

```r
adelie_torgersen <- penguins %>%
  filter(species == "Adelie" & island == "Torgersen")
```

We can select specific columns from the data frame. For example, selecting the species and island columns from the penguins dataset:

```r
penguins_sub <- penguins %>%
  select(island, species)
```

We can also use `mutate()` combined with `if_else()` to create a new variable based on conditions. For example, let's classify penguins as having "large" or "small" body mass based on whether they're above or below the median:

```r
penguins <- penguins %>%
  mutate(body_size = if_else(body_mass_g > median(body_mass_g, na.rm = TRUE),
                             "large",
                             "small"))
```

Visualization

An important part of data exploration includes visualizing the data to reveal patterns you can't necessarily see from viewing a data frame of numbers. Here we are going to walk through a very quick introduction to `ggplot2`, using some code examples from the {palmerpenguins} R package tutorial: https://allisonhorst.github.io/palmerpenguins/articles/intro.html.

{ggplot2} is perhaps the most popular data visualization package in the R language, and is also a part of the {tidyverse}. One big difference about `ggplot` is that it **does not use the pipe %>% operator** like we just learned, but instead threads together arguments with `+` signs (but you can pipe a data frame into the first `ggplot()` argument).

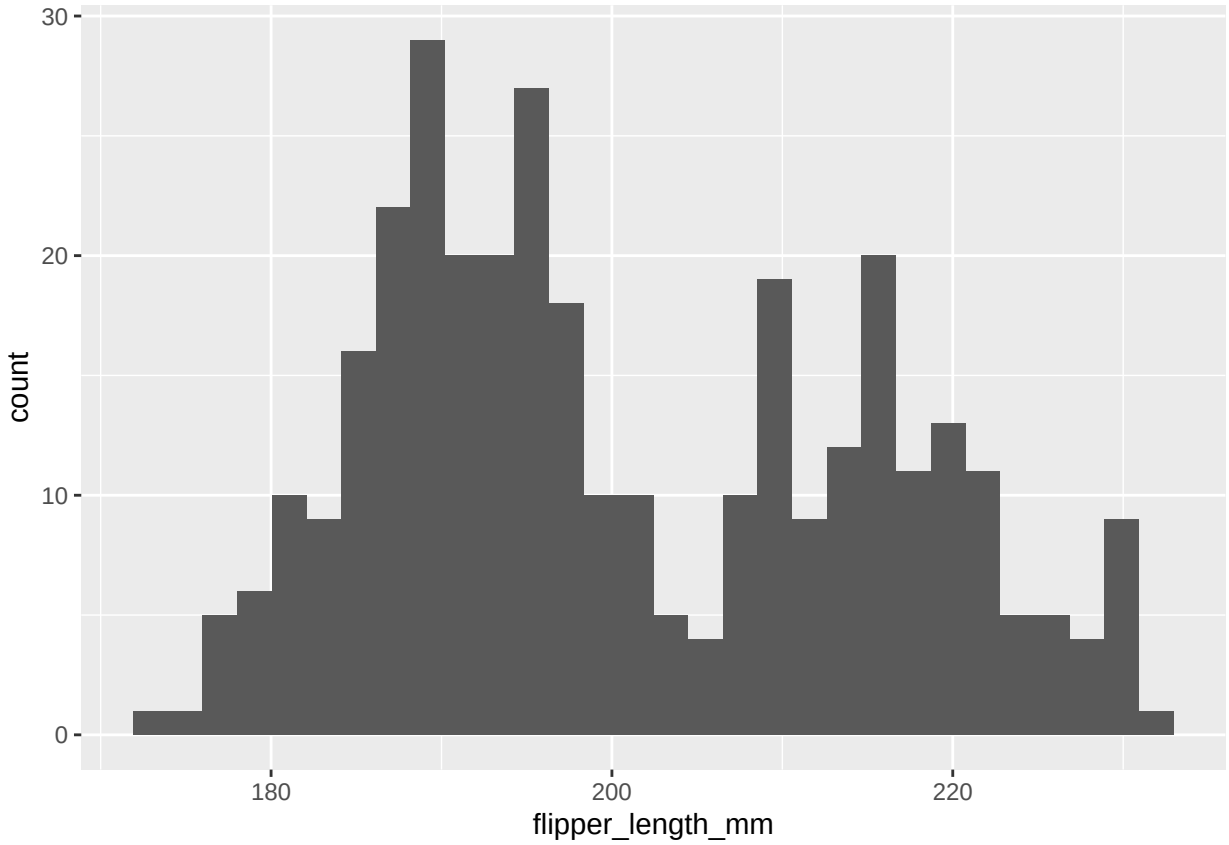
The general structure for ggplots follows the template below. Note that you can also specify the `aes()` parameters within `ggplot()` instead of your geom function, which you may see a lot of people do. The mappings include arguments such as the x and y variables from your data you want to use for the plot. The geom function is the type of plot you want to make, such as `geom_point()`, `geom_bar()`, etc, there are a lot to choose from.

```r
# general structure of ggplot functions
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

Visualize variable distributions with `geom_historgram()`

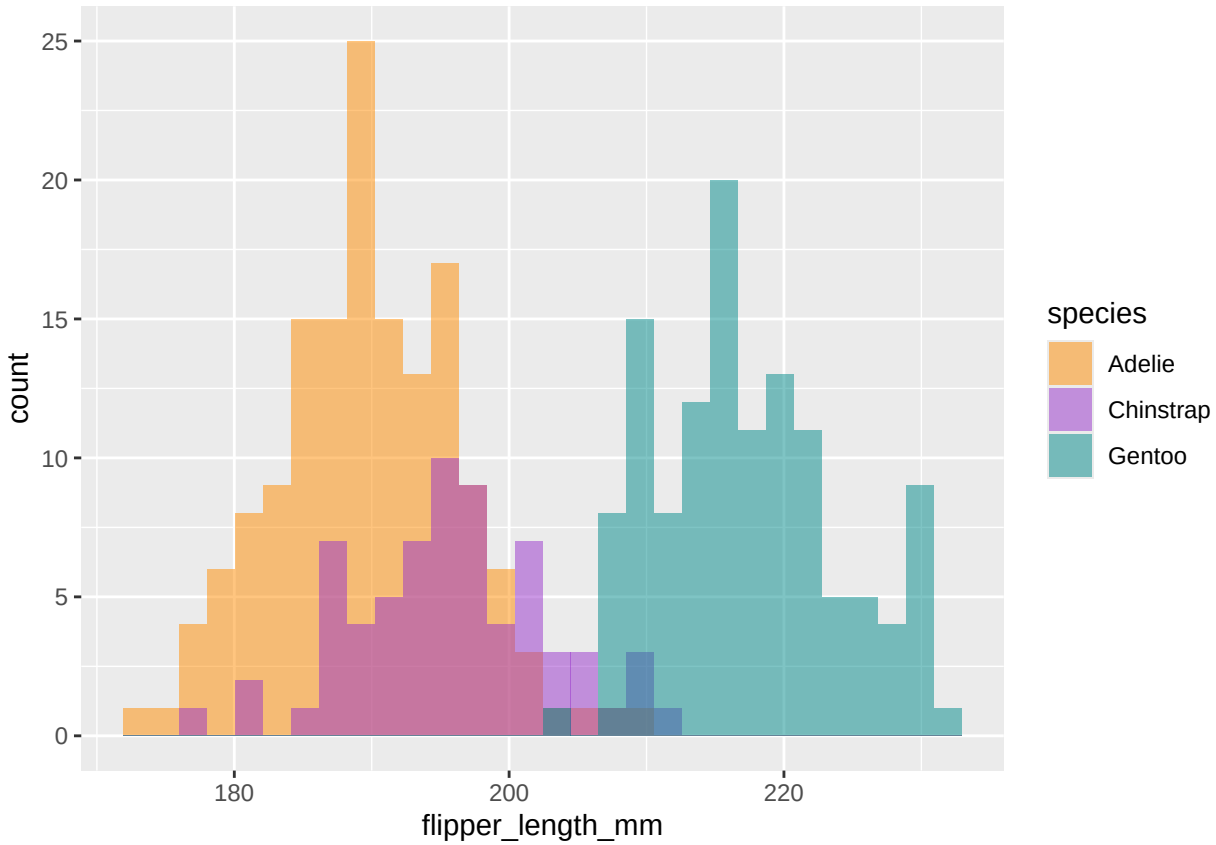
If you plan on doing any statistical analysis on your data , one of the first things you are likely to do is explore the distribution of your variables. You can plot histograms with `geom_histogram()`

```
ggplot(penguins) +
  geom_histogram(mapping = aes(x = flipper_length_mm))
```



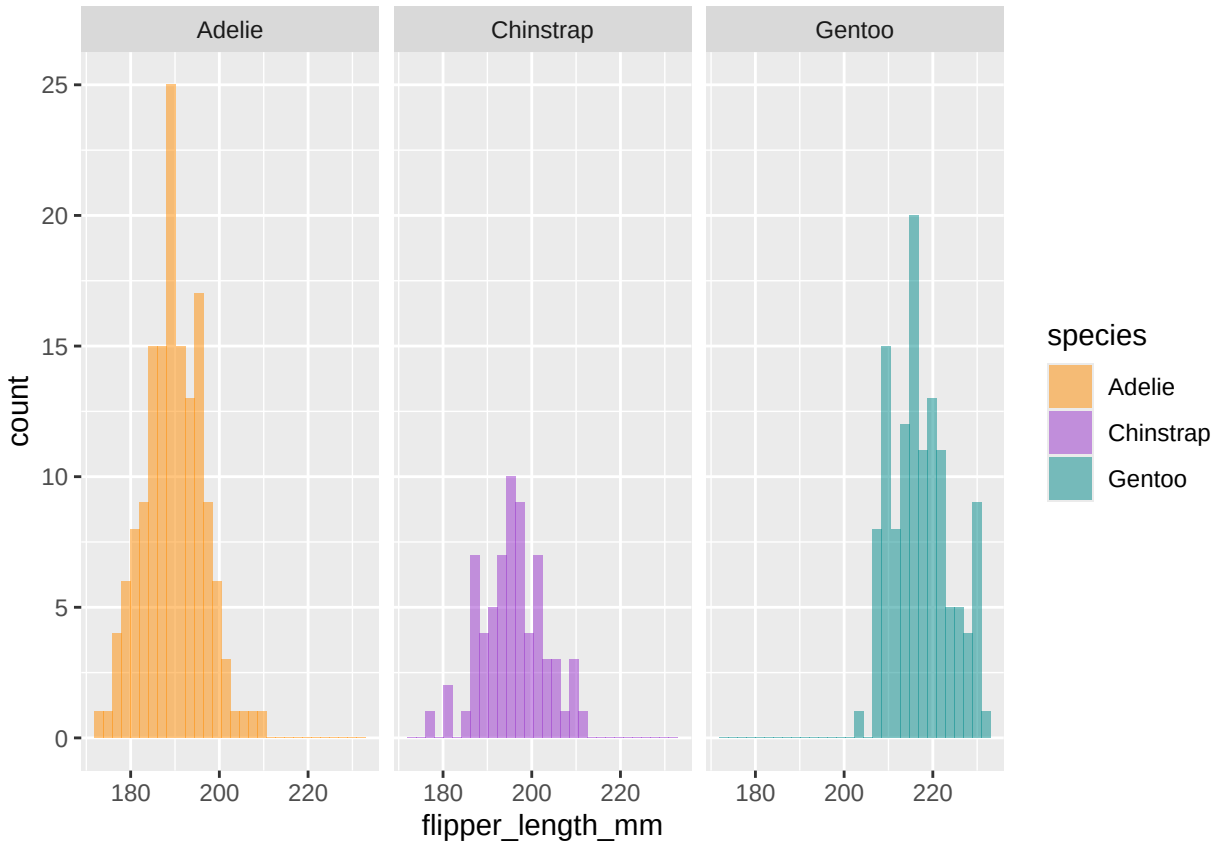
This tells us there may be a lot of variation in flipper size among species. We can use the 'fill =' argument to color the bars by species, and `scale_fill_manual()` to specify the colors.

```
# Histogram example: flipper length by species
ggplot(penguins) +
  geom_histogram(aes(x = flipper_length_mm, fill = species), alpha = 0.5, position = "identity") +
  scale_fill_manual(values = c("darkorange","darkorchid","cyan4"))
```



Cool, now we can see there seems to be some pretty clear variation in flipper size among species. Another way to visualize across groups is with `facet_wrap()`, which will create a separate plot for each group, in this case species.

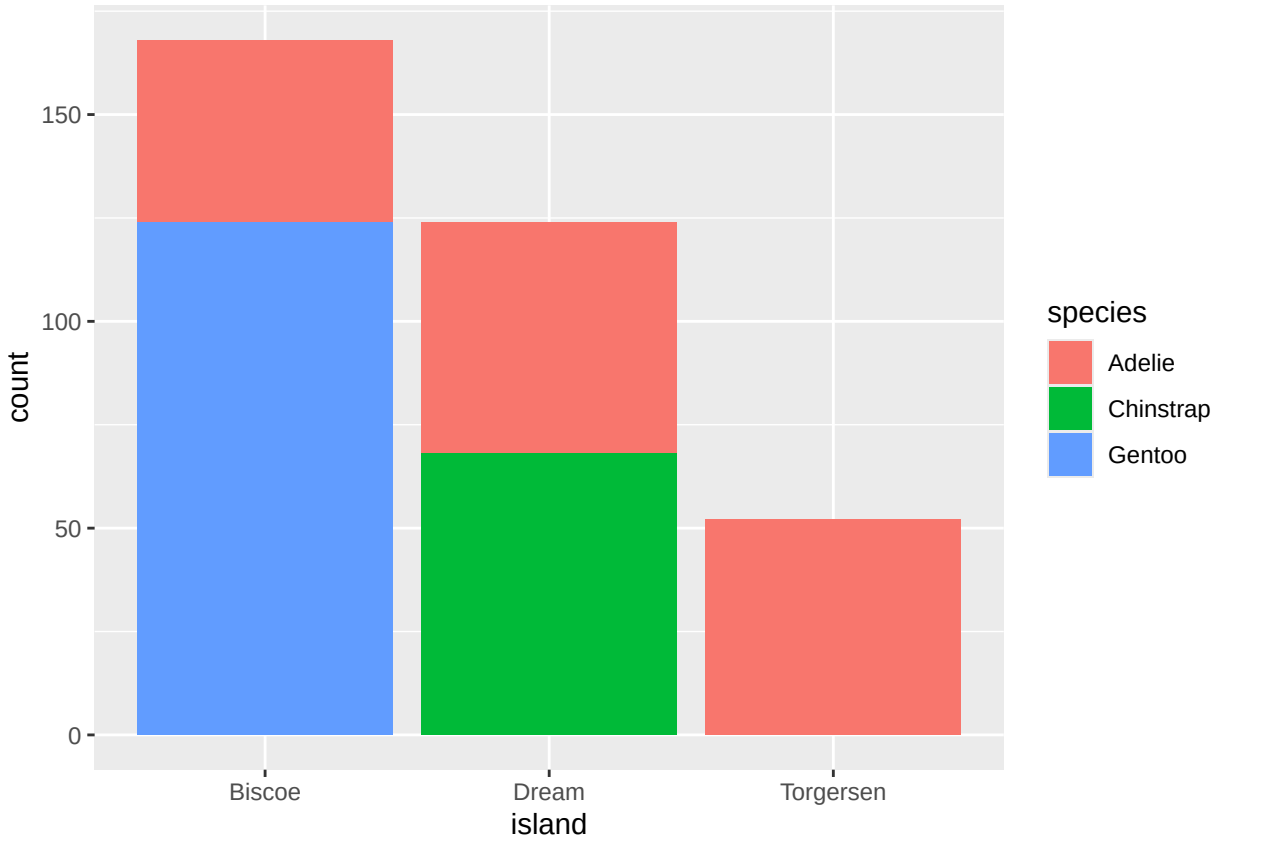
```
ggplot(penguins) +
  geom_histogram(aes(x = flipper_length_mm, fill = species), alpha = 0.5, position = "identity") +
  scale_fill_manual(values = c("darkorange","darkorchid","cyan4")) +
  facet_wrap(~species)
```


Adelie
Chinstrap
Gentoo
count
25
20
15
10
5
0
180
200
220
flipper_length_mm
species
Adelie
Chinstrap
Gentoo

Compare sample sizes with `geom_bar()`

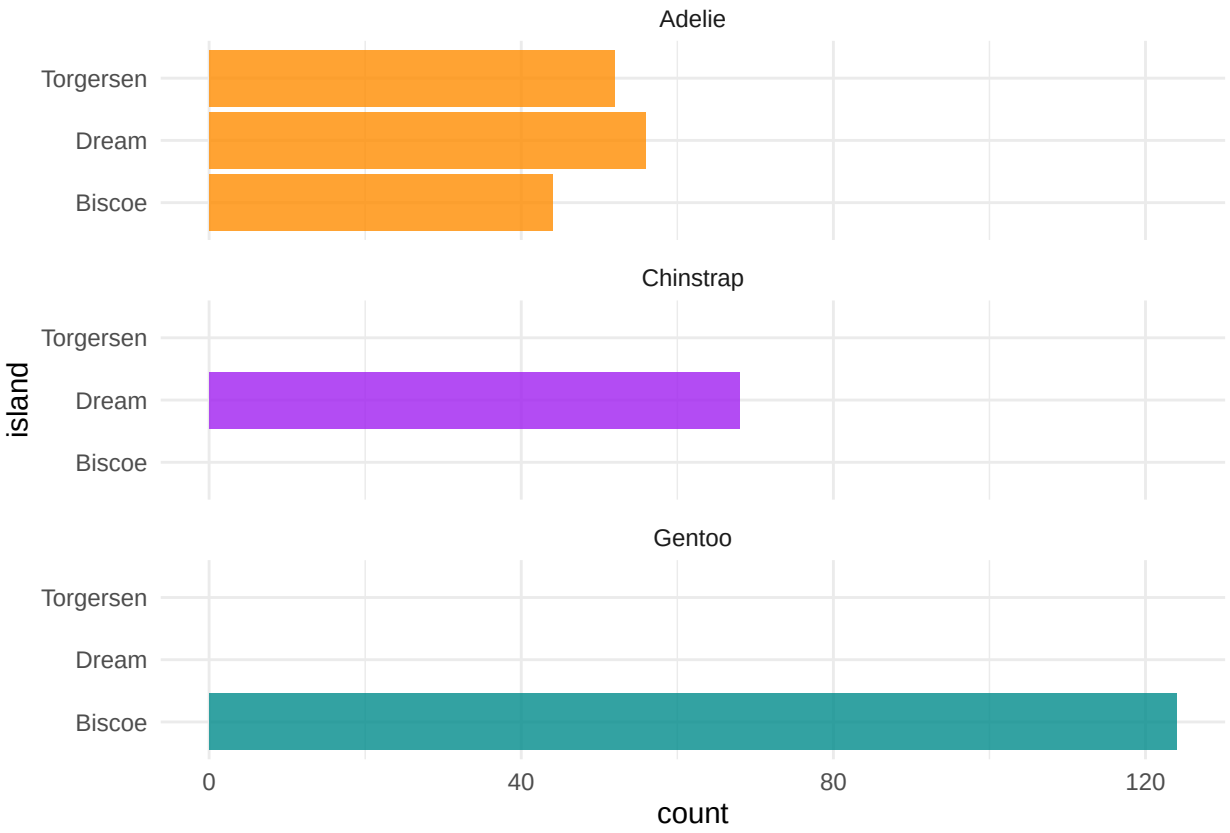
Let's use ggplot to see sample size for each species on each island.

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



As you may have already noticed, the beauty about `ggplot2` is there are a million ways you can customize your plots. This example builds on our simple bar plot:

```
ggplot(penguins, aes(x = island, fill = species)) +
  geom_bar(alpha = 0.8) +
  scale_fill_manual(values = c("darkorange","purple","cyan4"),
                    guide = FALSE) +
  theme_minimal() +
  facet_wrap(~species, ncol = 1) +
  coord_flip()
```

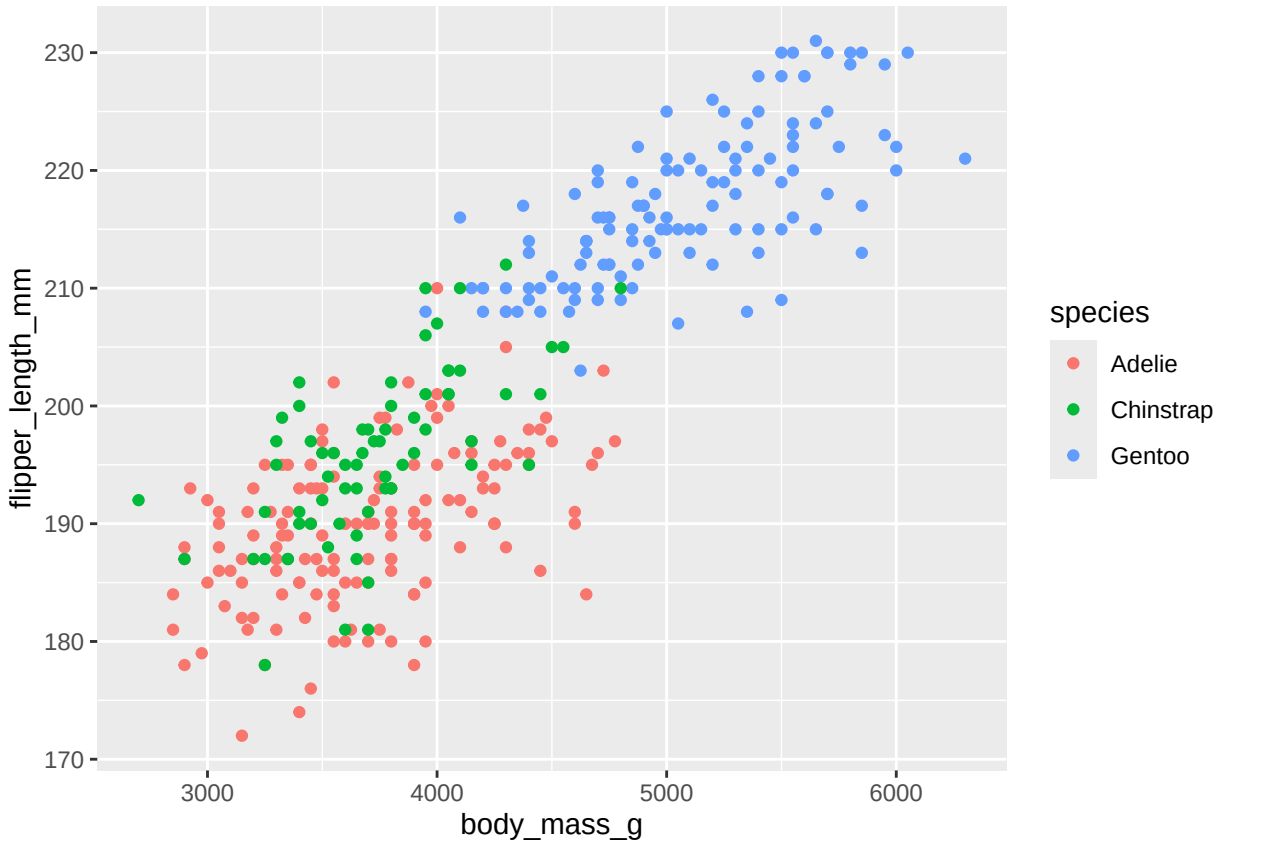


This is important information, since we know now that not all species were sampled on every island, which will have complications for any comparisons we may want to make among islands.

Visualize variable relationships with `geom_point()`

We can use `geom_point()` to view the relationship between two continuous variables by specifying the x and y axes. Say we want to visualize the relationship between penguin body mass and flipper length and color the points by species:

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



Lastly, we can edit the axis labels with `labs()`

```
ggplot(penguins) +
  geom_point(mapping = aes(x = body_mass_g, y = flipper_length_mm, color = species)) +
  labs(x = "Body Mass (g)", y = "Flipper Length (mm)")
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_point()`).
```


Flipper Length (mm)
230
220
210
200
190
180
170
3000
4000
5000
6000
Body Mass (g)
species
Adelie
Chinstrap
Gentoo