CH1-5

2024-03-09

1: The penguins data frame

You can see all variables and the first few observations of each variable by using `glimpse()`.

```
penguins <- palmerpenguins::penguins

glimpse(penguins)
```

```
## Rows: 344
## Columns: 8
## $ species           <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, Adel~
## $ island            <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torgerse~
## $ bill_length_mm    <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.1, ~
## $ bill_depth_mm     <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.1, ~
## $ flipper_length_mm <int> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, 186~
## $ body_mass_g       <int> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 3475, ~
## $ sex               <fct> male, female, female, NA, female, male, female, male~
## $ year              <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007~
```

2: Creating a ggplot

The `mapping` argument of the `ggplot()` function defines how variables in your dataset are mapped to visual properties (*aesthetics*) of your plot. The `mapping` argument is always defined in the `aes()` function, and the `x` and `y` arguments of `aes()` specify which variables to map to the x and y axes.

geom: The geometrical object that a plot uses to represent data. These geometric objects are made available in ggplot2 with functions that start with `geom_`.

People often describe plots by the type of geom that the plot uses. For example, bar charts use bar geoms (`geom_bar()`), line charts use line geoms (`geom_line()`), boxplots use boxplot geoms (`geom_boxplot()`), scatterplots use point geoms (`geom_point()`), and so on.

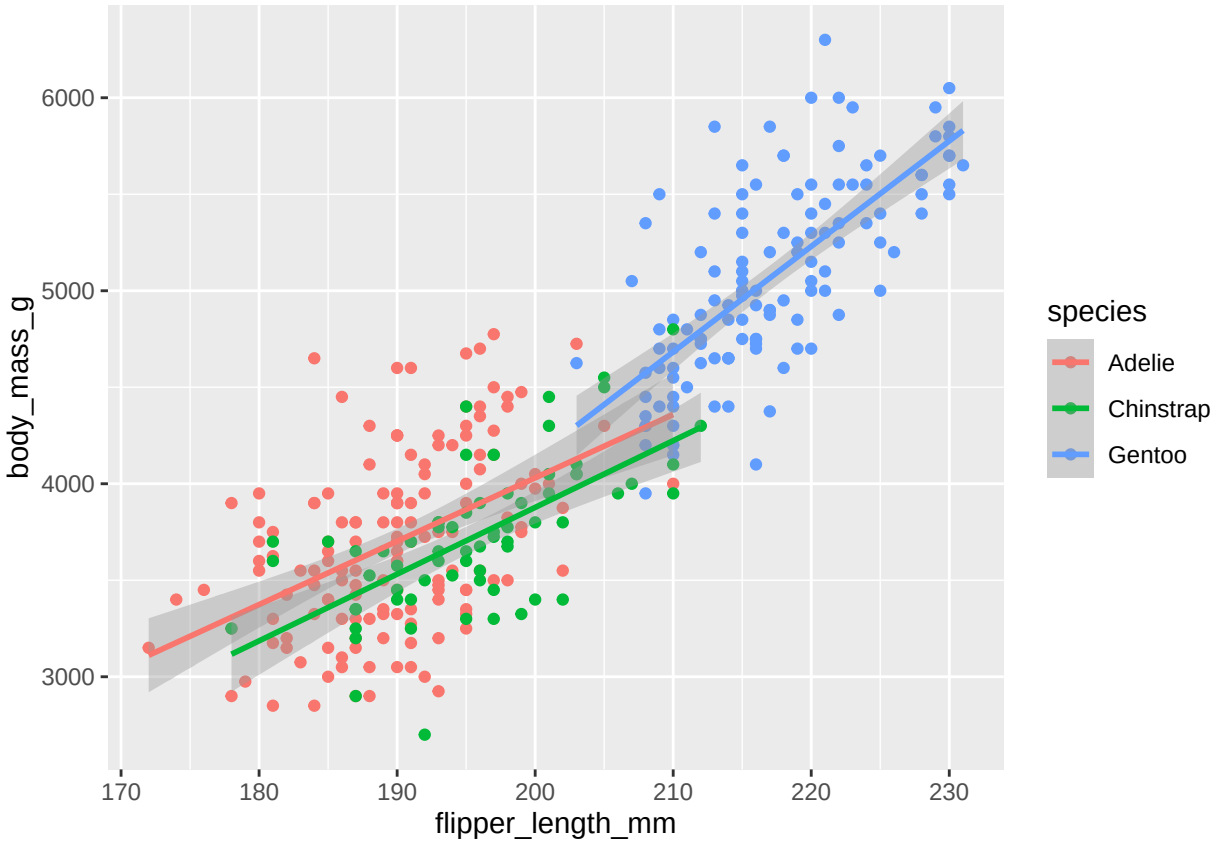
The function `geom_point()` adds a layer of points to your plot, which creates a scatterplot.

When a categorical variable is mapped to an aesthetic, ggplot2 will automatically assign a unique value of the aesthetic (here a unique color) to each unique level of the variable (each of the three species), a process known as *scaling*. ggplot2 will also add a legend that explains which values correspond to which levels.

Now let's add one more layer: a smooth curve displaying the relationship between body mass and flipper length.Since this is a new geometric object representing our data, we will add a new geom as a layer on top of our point geom: `geom_smooth()`. And we will specify that we want to draw the line of best fit based on a `linear model` with `method = "lm"`.

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

```
## `geom_smooth()` using formula = 'y ~ x'
```



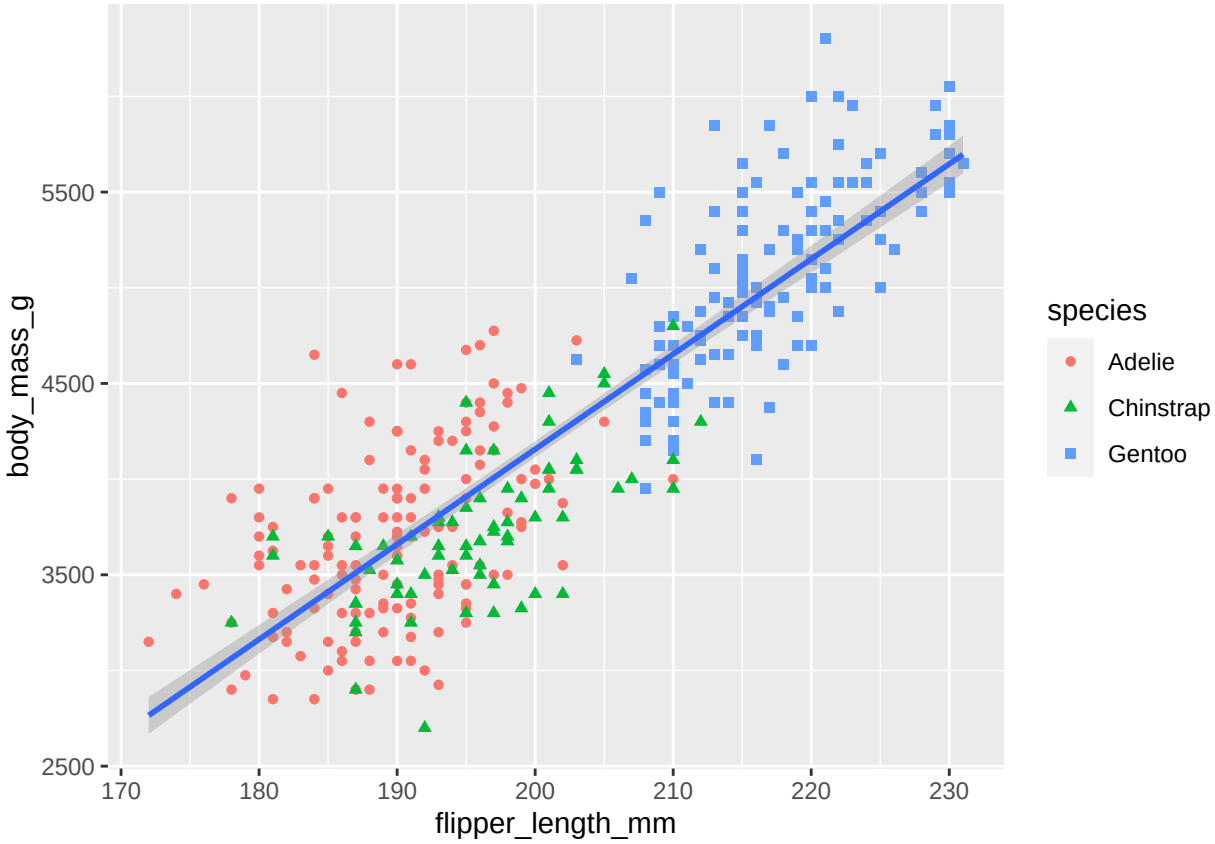
When aesthetic mappings are defined in `ggplot()`, at the global level, they're passed down to each of the subsequent geom layers of the plot. However, each geom function in ggplot2 can also take a `mapping` argument, which allows for aesthetic mappings at the local level that are added to those inherited from the global level.

Since we want points to be colored based on species but don't want the lines to be separated out for them, we should specify `color = species` for `geom_point()` only.

It's generally not a good idea to represent information using only colors on a plot, as people perceive colors differently due to color blindness or other color vision differences. Therefore, in addition to color, we can also map `species` to the `shape` aesthetic.

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

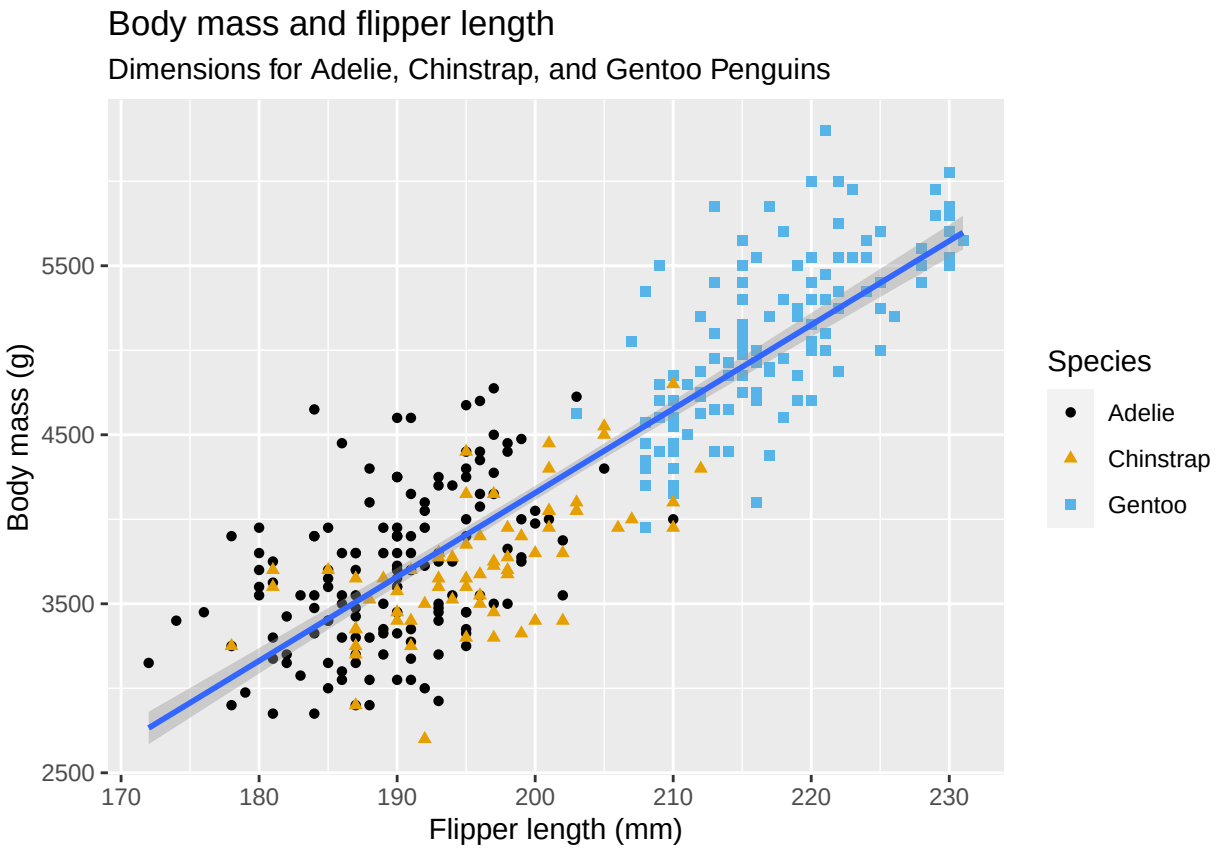
```
## `geom_smooth()` using formula = 'y ~ x'
```



We can improve the labels of our plot using the `labs()` function in a new layer. Some of the arguments to `labs()` might be self explanatory: title adds a `title` and subtitle adds a `subtitle` to the plot. Other arguments match the aesthetic mappings, `x` is the x-axis label, `y` is the y-axis label, and `color` and `shape` define the label for the legend. In addition, we can improve the color palette to be colorblind safe with the `scale_color_colorblind()` function from the `ggthemes` package.

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

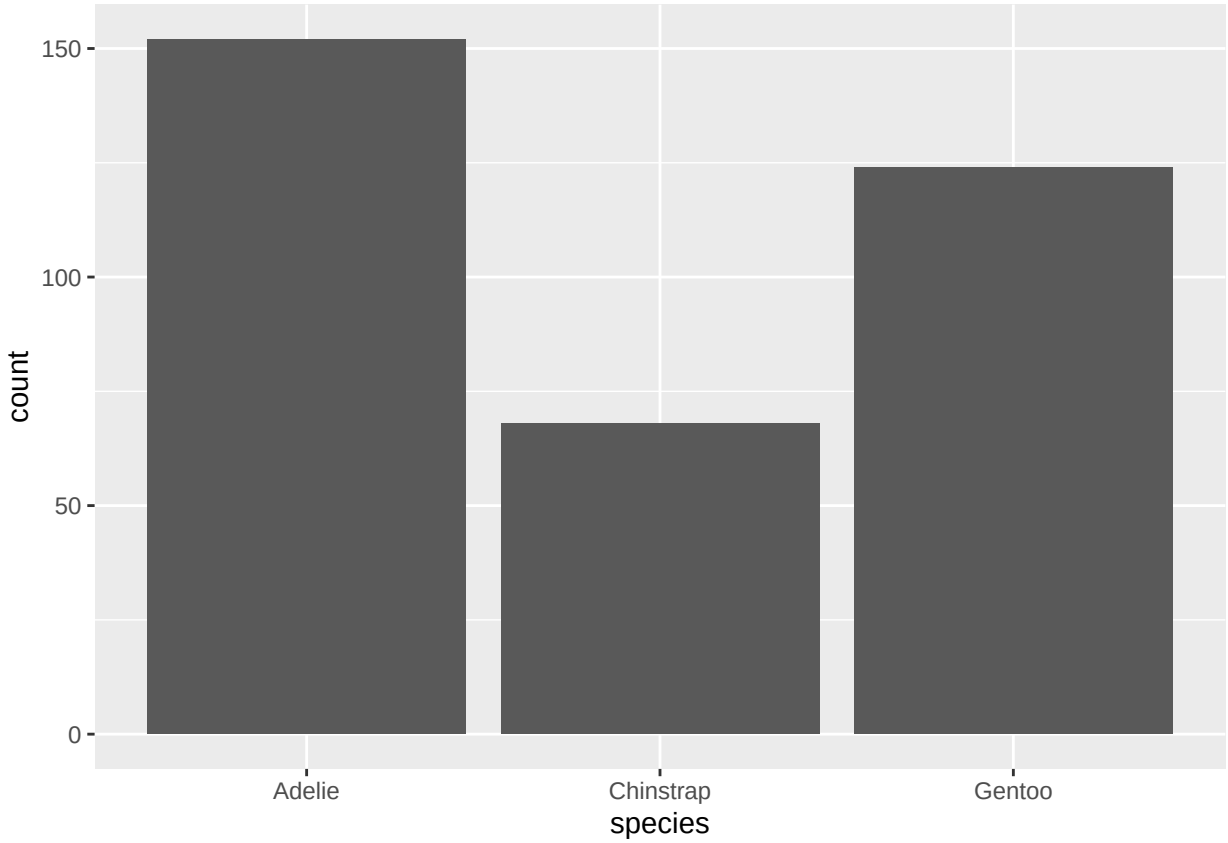
```
## `geom_smooth()` using formula = 'y ~ x'
```



3. Visualizing distributions

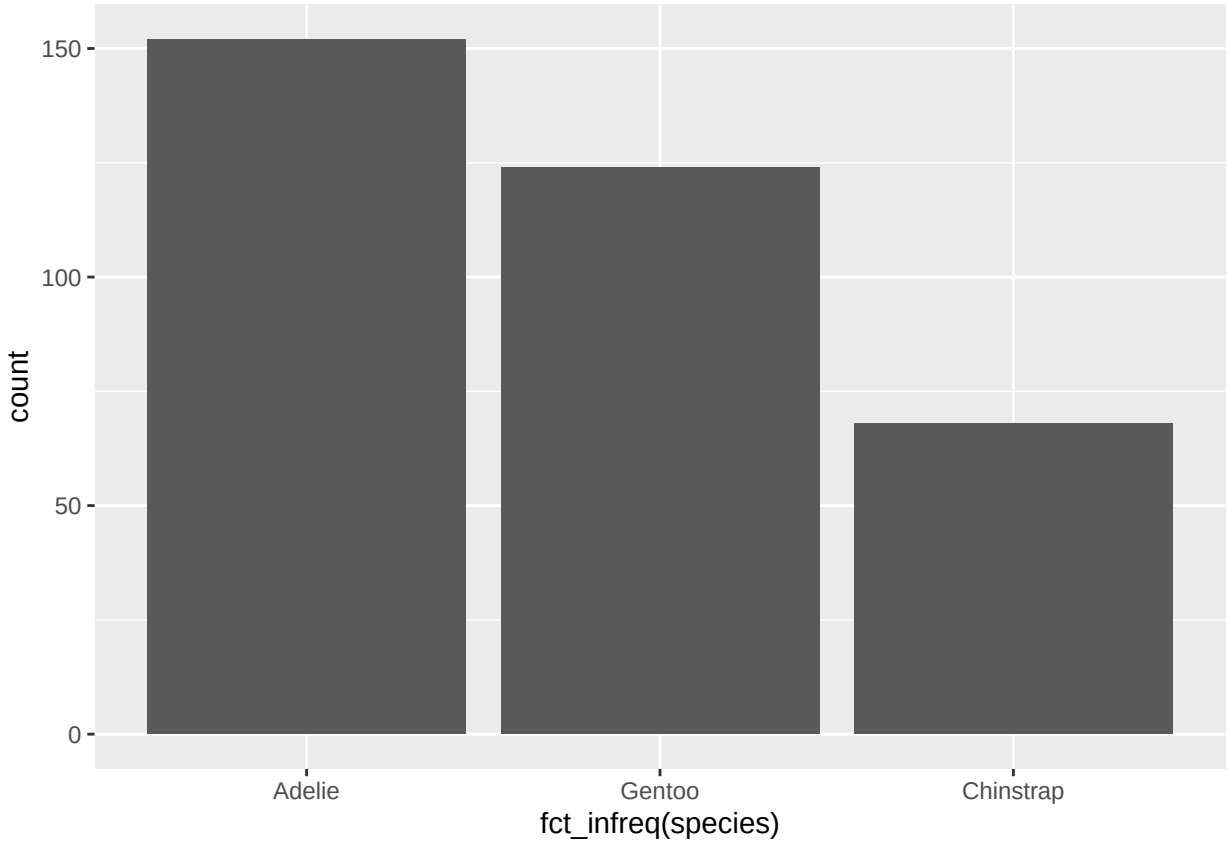
A variable is *categorical* if it can only take one of a small set of values. To examine the distribution of a categorical variable, you can use a bar chart. The height of the bars displays how many observations occurred with each x value.

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



In bar plots of categorical variables with non-ordered levels, like the penguin species above, it's often preferable to reorder the bars based on their frequencies. Doing so requires transforming the variable to a factor (how R handles categorical data) and then reordering the levels of that factor. So, you can use `fct_infreq()`.

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

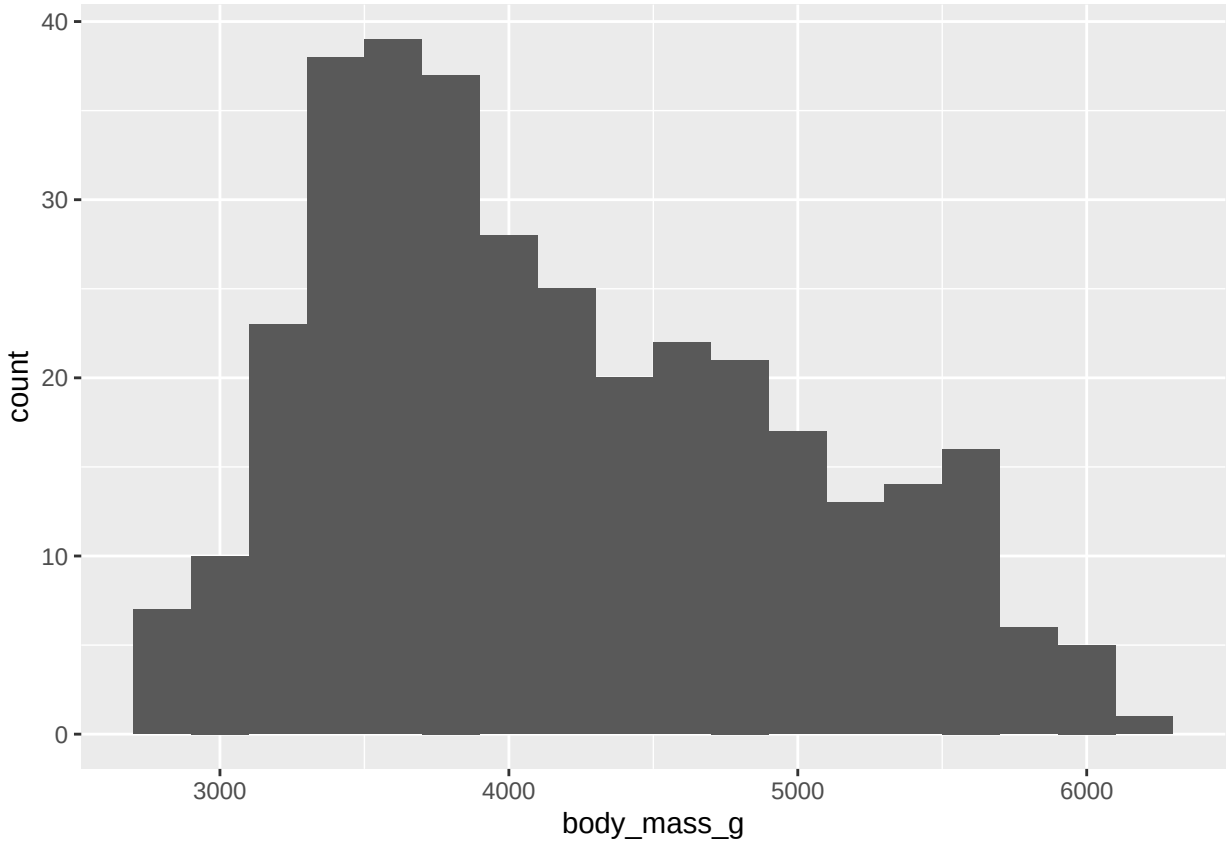


A variable is *numerical* (or quantitative) if it can take on a wide range of numerical values, and it is sensible to add, subtract, or take averages with those values. Numerical variables can be continuous or discrete.

One commonly used visualization for distributions of continuous variables is a *histogram*. A histogram divides the x-axis into equally spaced bins and then uses the height of a bar to display the number of observations that fall in each bin.

You can set the width of the intervals in a histogram with the binwidth argument, which is measured in the units of the x variable.

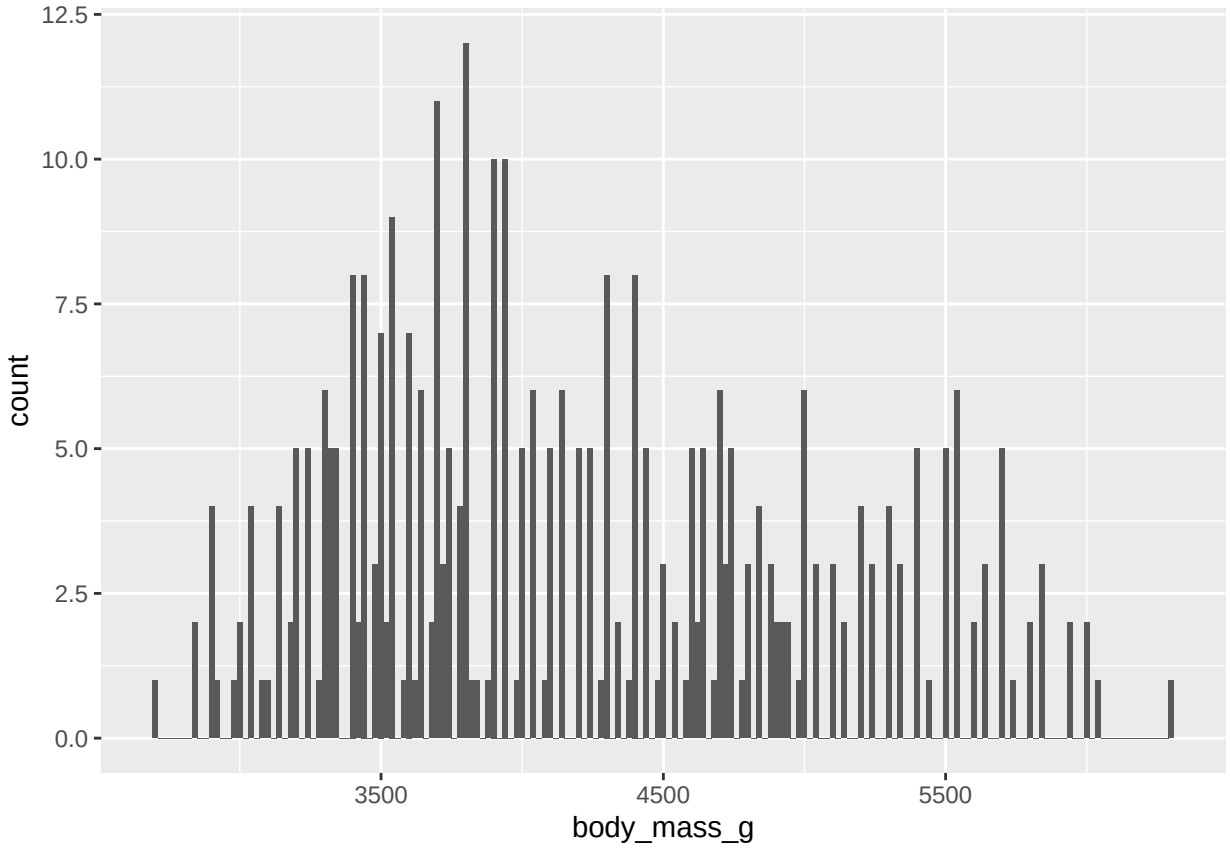
```
ggplot(penguins,
       aes(x = body_mass_g)) +
  geom_histogram(binwidth = 200)
```



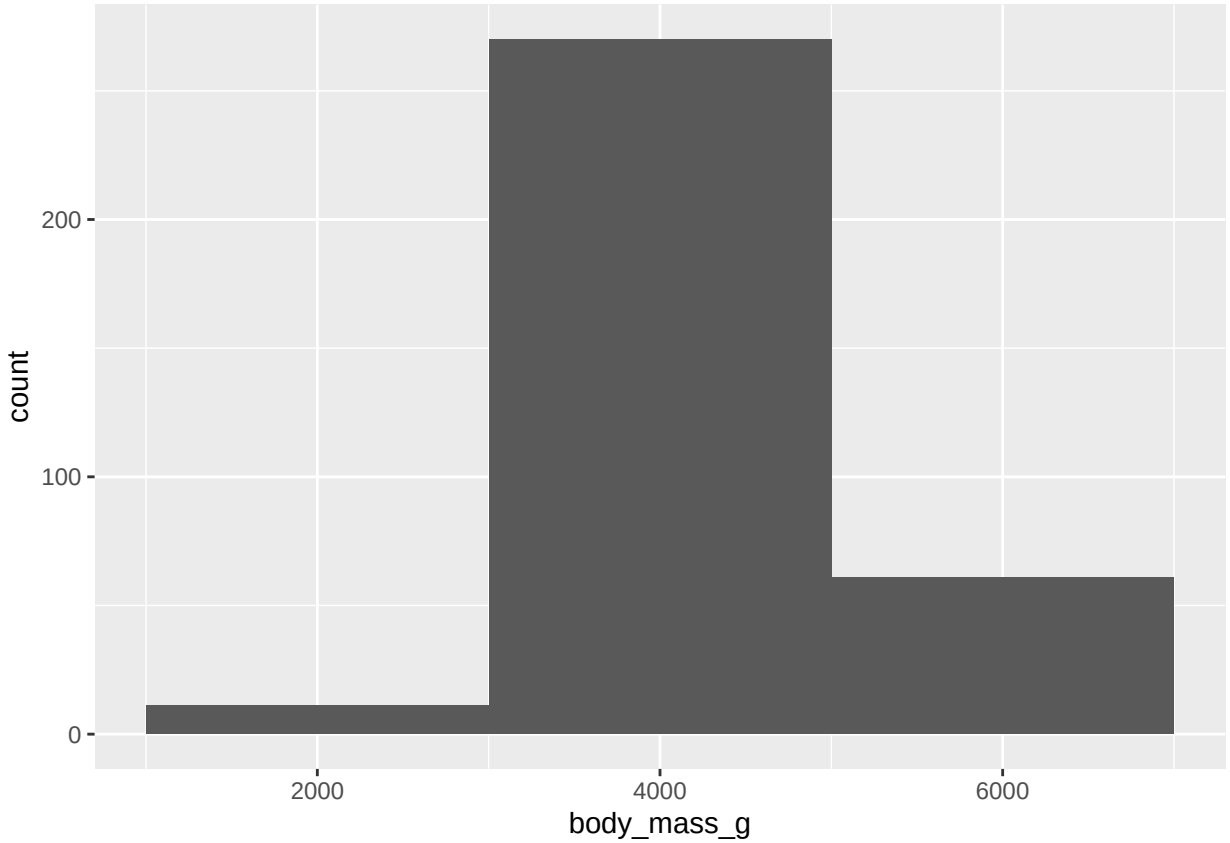
You should always explore a variety of binwidths when working with histograms, as different binwidths can reveal different patterns.

In the plots below a binwidth of 20 is too narrow, resulting in too many bars, making it difficult to determine the shape of the distribution. Similarly, a binwidth of 2,000 is too high, resulting in all data being binned into only three bars, and also making it difficult to determine the shape of the distribution.

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

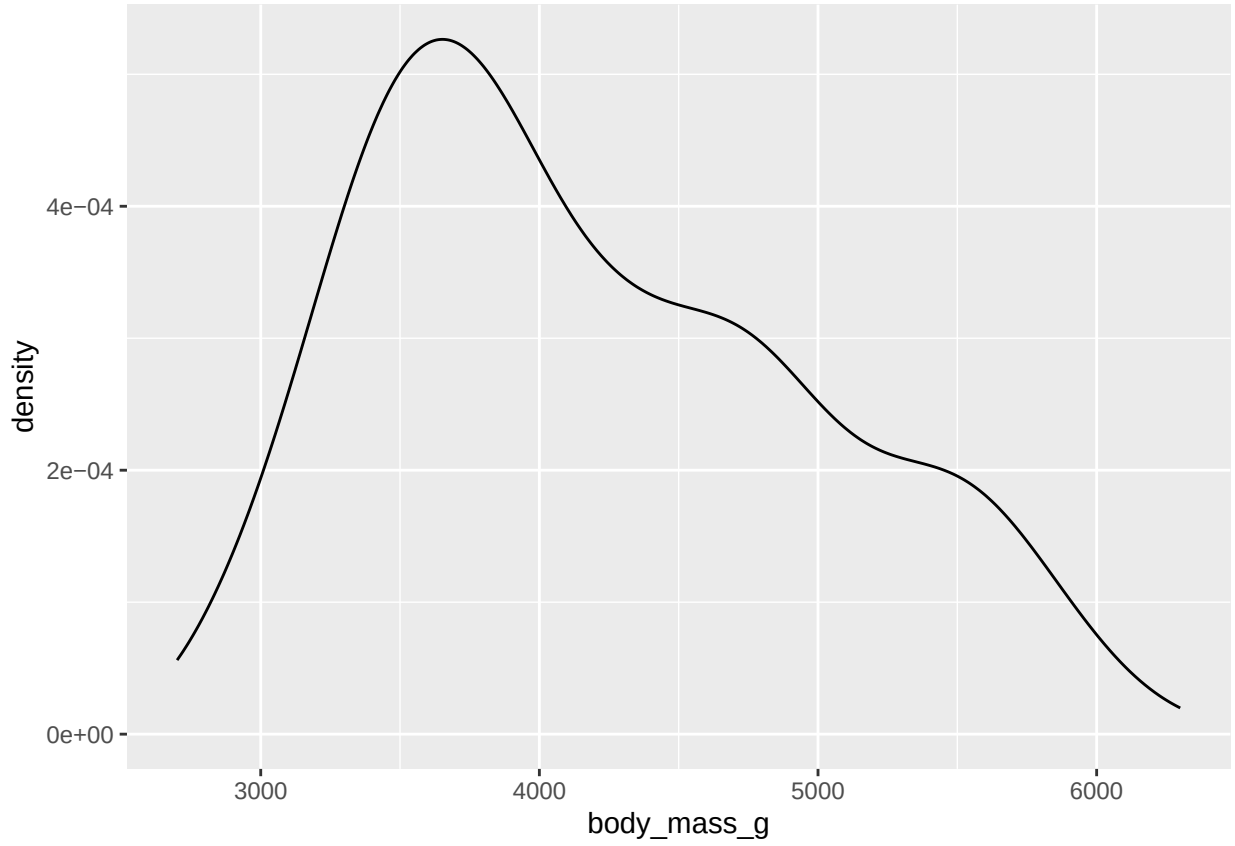



```
ggplot(penguins,
       aes(x = body_mass_g)) +
  geom_histogram(binwidth = 2000)
```



A *density* plot is a smoothed-out version of a histogram and a practical alternative, particularly for continuous data that comes from an underlying smooth distribution.

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



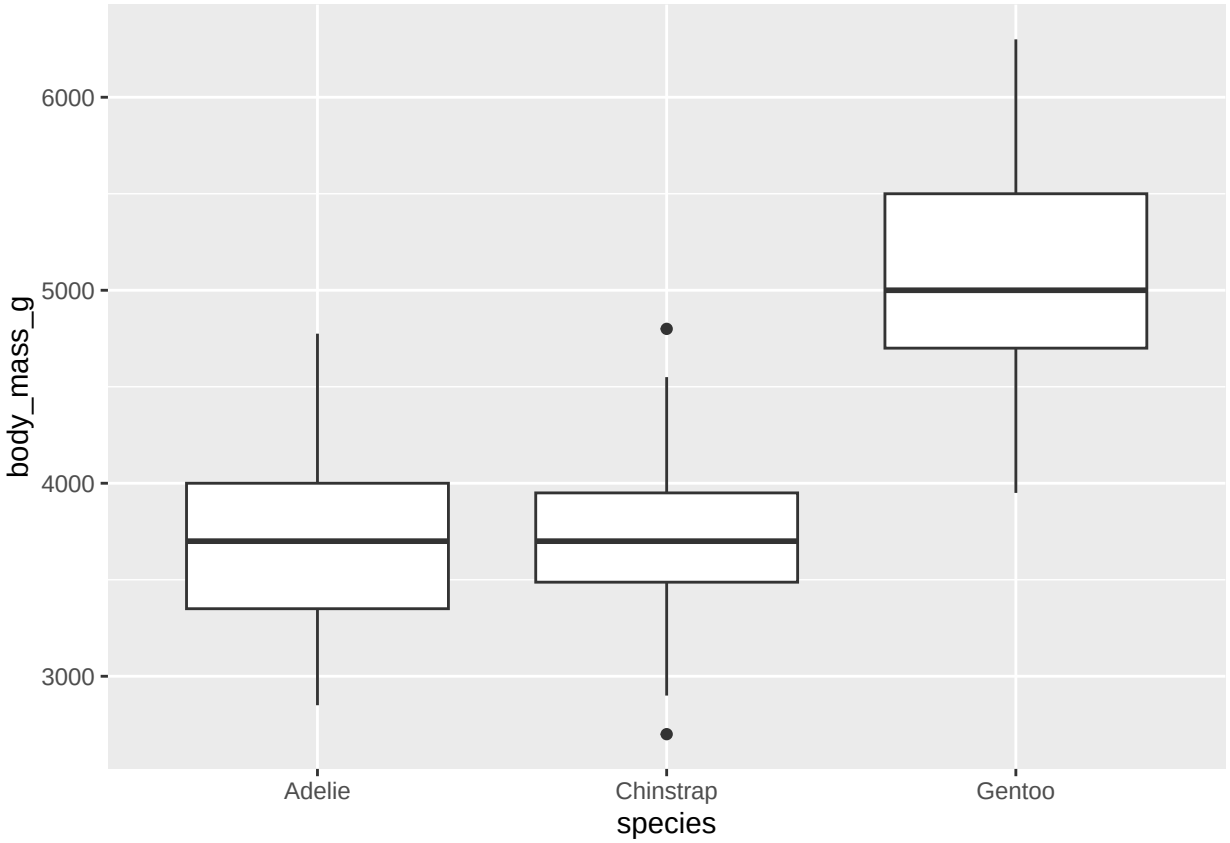
4. Visualizing relationships

To visualize the relationship between a *numerical* and a *categorical* variable we can use side-by-side box plots.

A *boxplot* is a type of visual shorthand for measures of position (percentiles) that describe a distribution. It is also useful for identifying potential outliers.

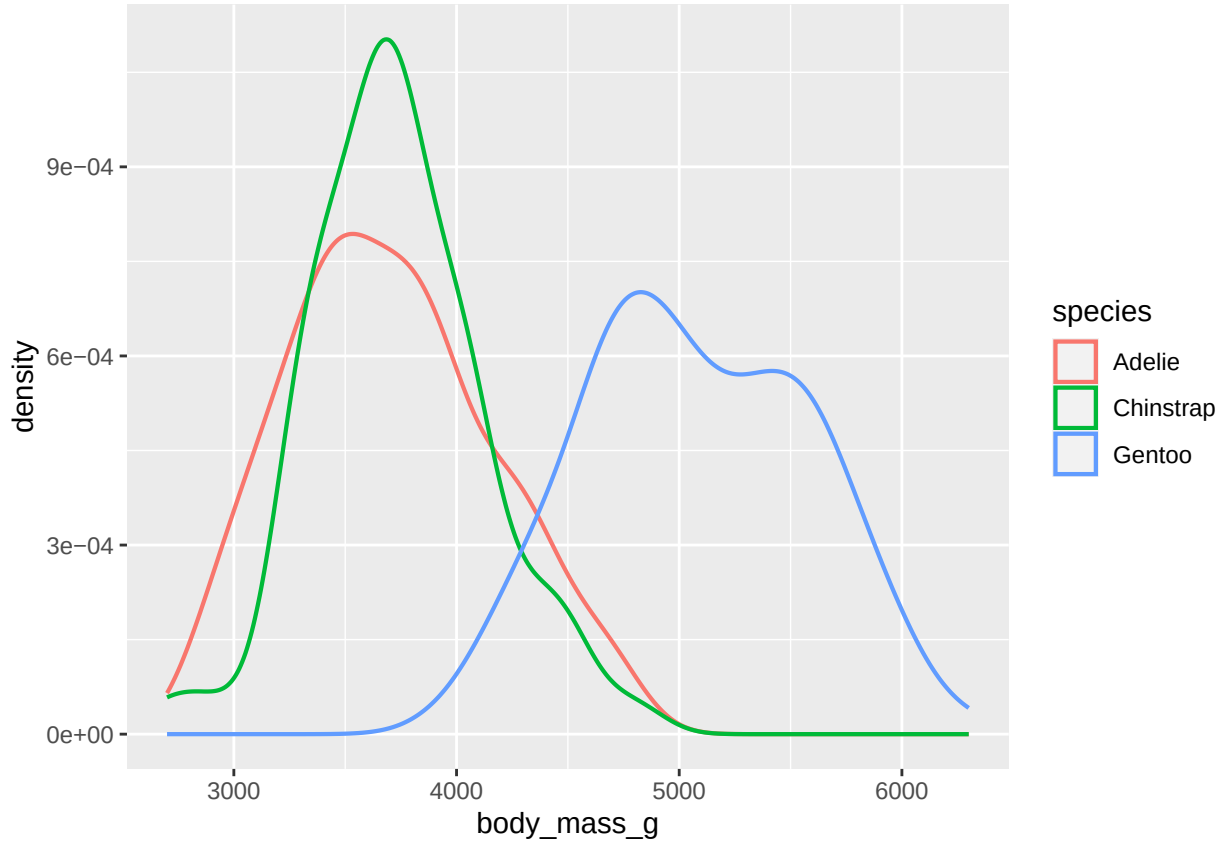
- A box that indicates the range of the middle half of the data, a distance known as the interquartile range (IQR). The 25th, 75th, and the median lines give you a sense of the spread of the distribution and whether or not the distribution is symmetric about the median or skewed to one side.
- A line (or whisker) that extends from each end of the box and goes to the farthest non-outlier point in the distribution.

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



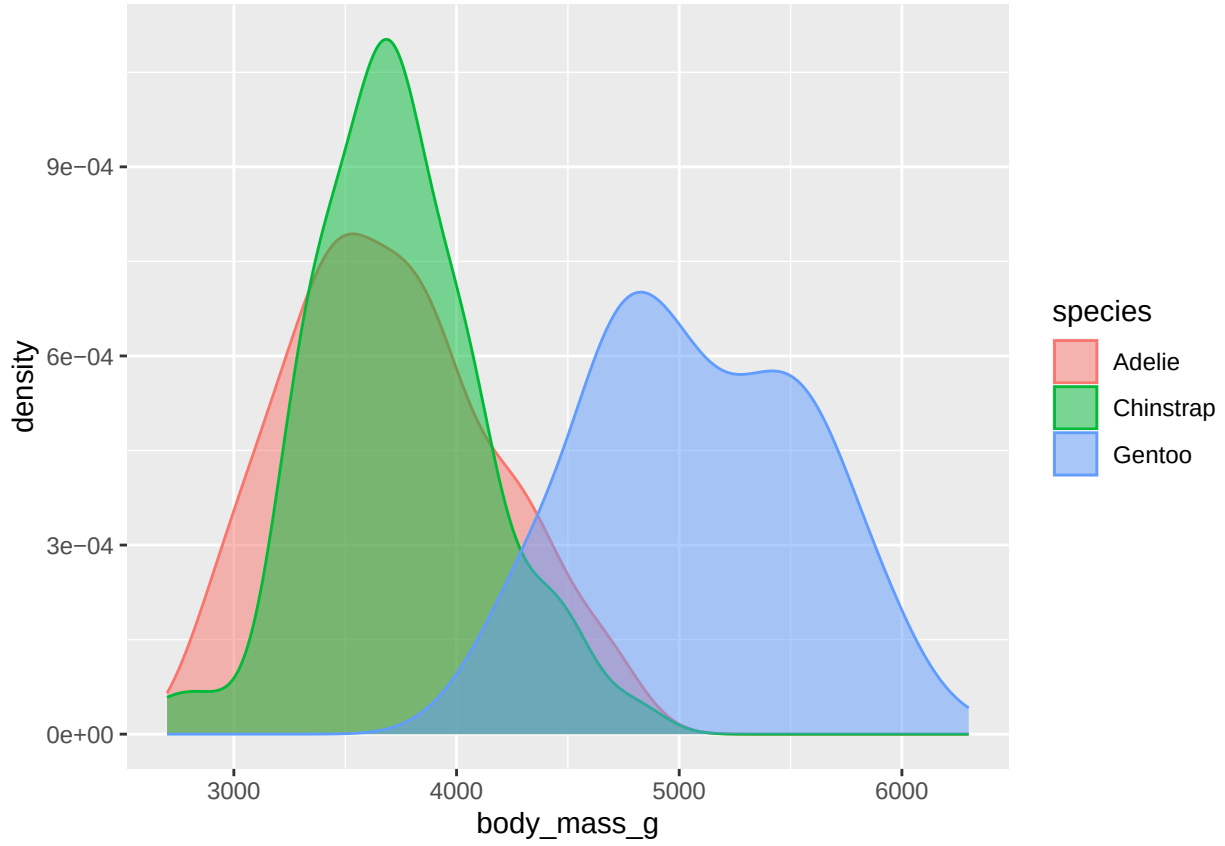
Alternatively, we can make density plots with `geom_density()`. You can customize the thickness of the lines using the `linewidth` argument in order to make them stand out a bit more against the background.

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



Additionally, we can map species to both `color` and `fill` aesthetics and use the `alpha` aesthetic to add transparency to the filled density curves. This aesthetic takes values between 0 (completely transparent) and 1 (completely opaque).

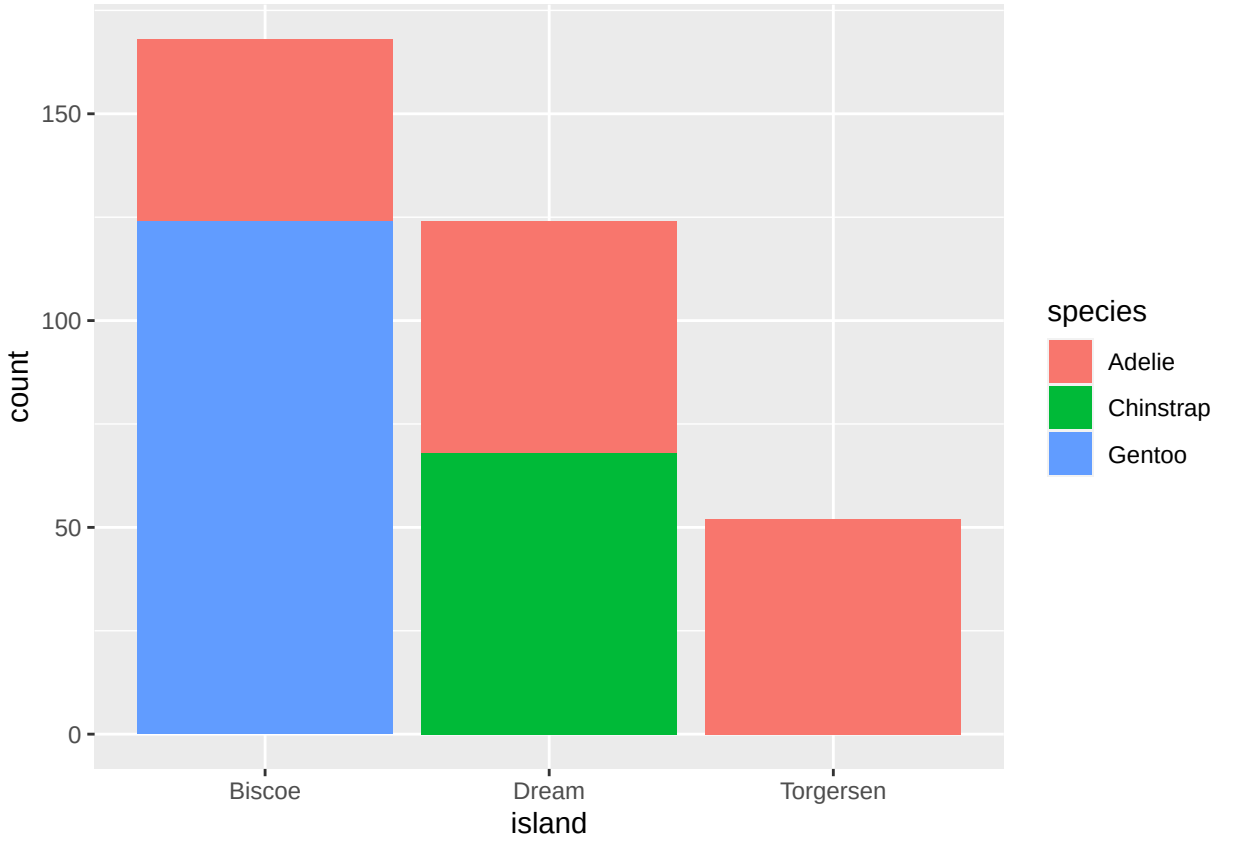
```
ggplot(penguins,
       aes(x = body_mass_g,
           color = species,
           fill = species)) +
  geom_density(alpha = 0.5)
```



We can use `stacked bar plots` to visualize the relationship between two categorical variables.

The first plot shows the frequencies of each species of penguins on each island. The plot of frequencies shows that there are equal numbers of Adelies on each island. But we don't have a good sense of the percentage balance within each island.

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

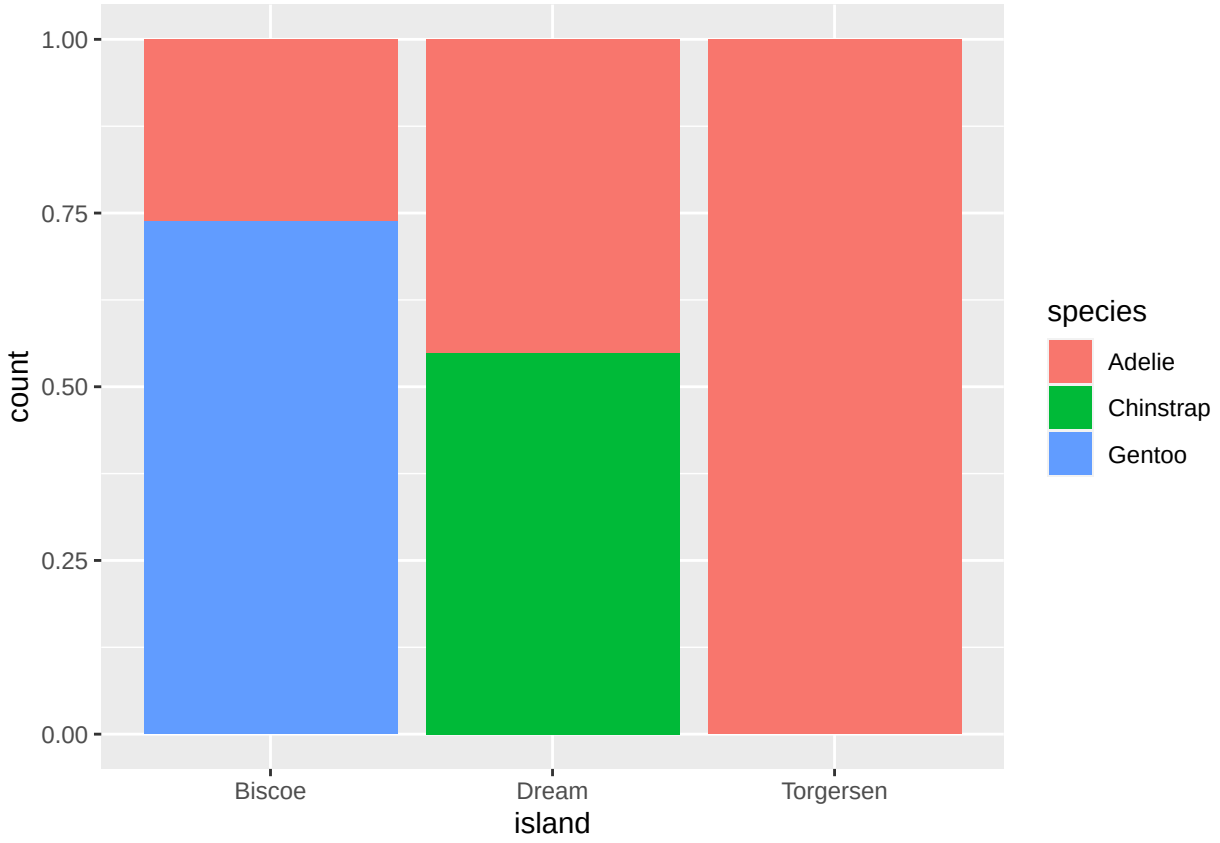


The second plot, a relative frequency plot created by setting `position = "fill"` in the geom, is more useful for comparing species distributions across islands since it's not affected by the unequal numbers of penguins across the islands.

Using this plot we can see that Gentoo penguins all live on Biscoe island and make up roughly 75% of the penguins on that island, Chinstrap all live on Dream island and make up roughly 50% of the penguins on that island, and Adelie live on all three islands and make up all of the penguins on Torgersen.

In creating these bar charts, we map the variable that will be separated into bars to the `x` aesthetic, and the variable that will change the colors inside the bars to the `fill` aesthetic.

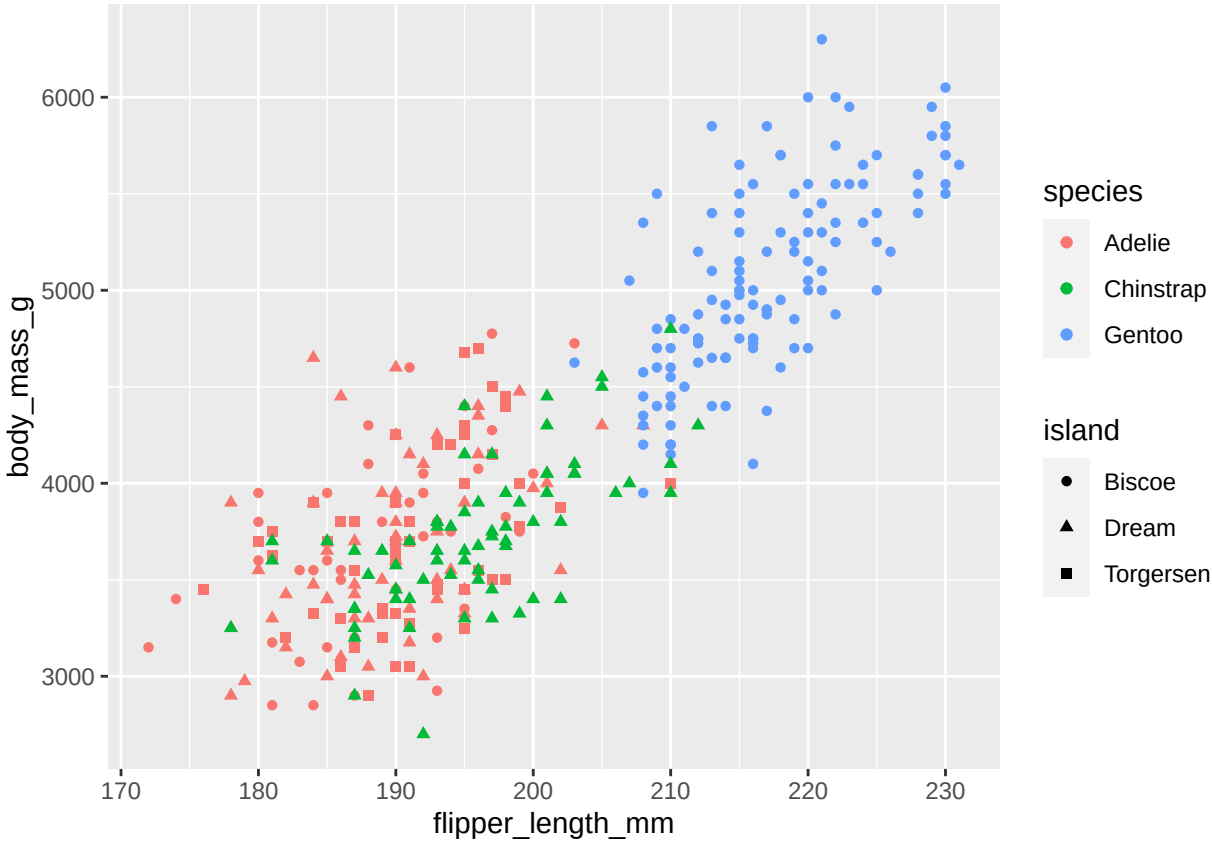
```
ggplot(penguins,
       aes(x = island,
           fill = species)) +
  geom_bar(position = "fill")
```



5. Three or more variables

We can incorporate more variables into a plot by mapping them to additional aesthetics. For example, in the following scatterplot the colors of points represent species and the shapes of points represent islands.

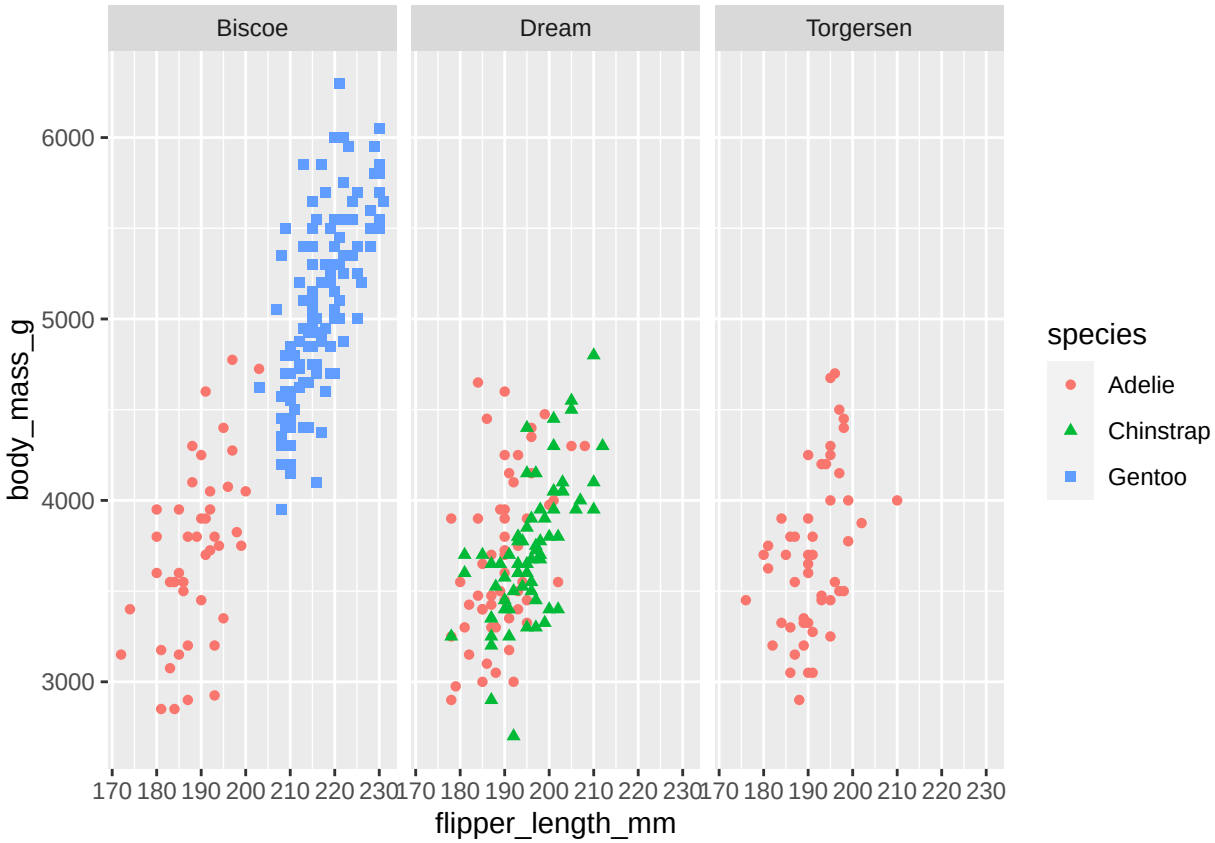
```
ggplot(penguins,
       aes(x = flipper_length_mm,
           y = body_mass_g)) +
  geom_point(aes(color = species,
                 shape = island))
```



However adding too many aesthetic mappings to a plot makes it cluttered and difficult to make sense of. Another way, which is particularly useful for categorical variables, is to split your plot into *facets*, subplots that each display one subset of the data.

To facet your plot by a single variable, use `facet_wrap()`. The first argument of `facet_wrap()` is a formula, which you create with `~` followed by a variable name. The variable that you pass to `facet_wrap()` should be *categorical*.

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


Biscoe
Dream
Torgersen
body_mass_g
6000
5000
4000
3000
170 180 190 200 210 220 230
flipper_length_mm
species
Adelie
Chinstrap
Gentoo