4: Exploring `geom` functions

Ellen Bledsoe

2023-02-23

Plotting in `ggplot2`

Data Visualization Types and When to Use Them

Scavenger Hunt

Head over to data-to-viz.com and do some exploration! In your groups, choose one of the following types of plots. Hunt down as many locations in the flow chart where your plot type is found as you can find.

- histogram: `geom_histogram()`
- density plot: `geom_density()`
- scatter plot: `geom_point()`
- box-and-whisker: `geom_boxplot()` + `geom_jitter()`

If you've finished, click on your plot type and read more about it. Check out the examples that were made in R, including many using `ggplot2`. Take note of the following:

- which `geom` function is being used to create the plot
- you'll see a number of other things in the `ggplot` code, some which will be new. What are they? Any idea what they are doing?

Using `geom` Functions to Make Plots

Let's practice making different kinds of plots with various `geom` functions to see how they work.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
tanks <- read_csv("../data/fish_tank_data.csv")
```

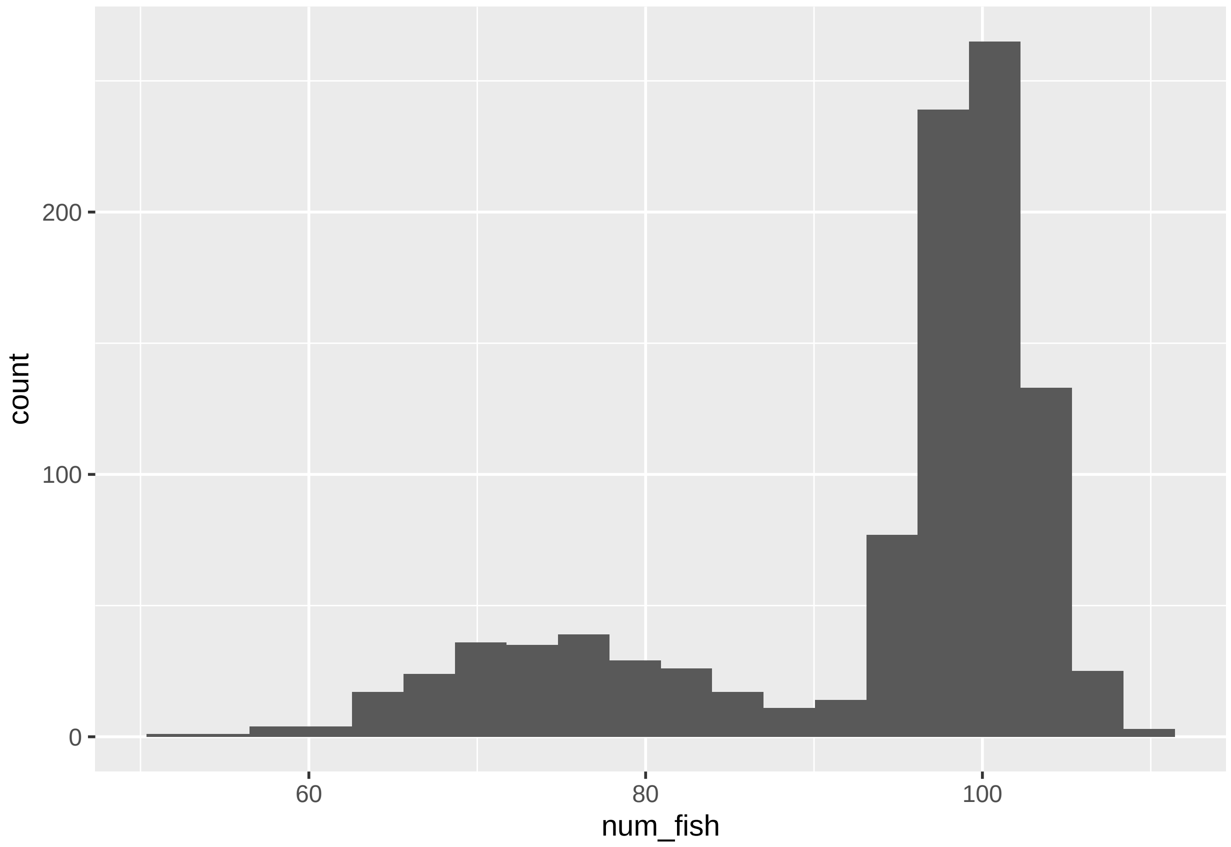
```
## Rows: 1000 Columns: 7

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): species
## dbl (6): tank_id, avg_daily_temp, num_fish, day_length, tank_volume, size_da...
##
## 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.
```

Histograms

As we learned last modules, histograms are plots which let us look at *one continuous variable* and get a feel for the distribution of that data. To make histograms in ggplot2, we use the geom_histogram() function. Let's look at signal distance.

```
ggplot(tanks, aes(num_fish)) +
  geom_histogram(bins = 20)
```

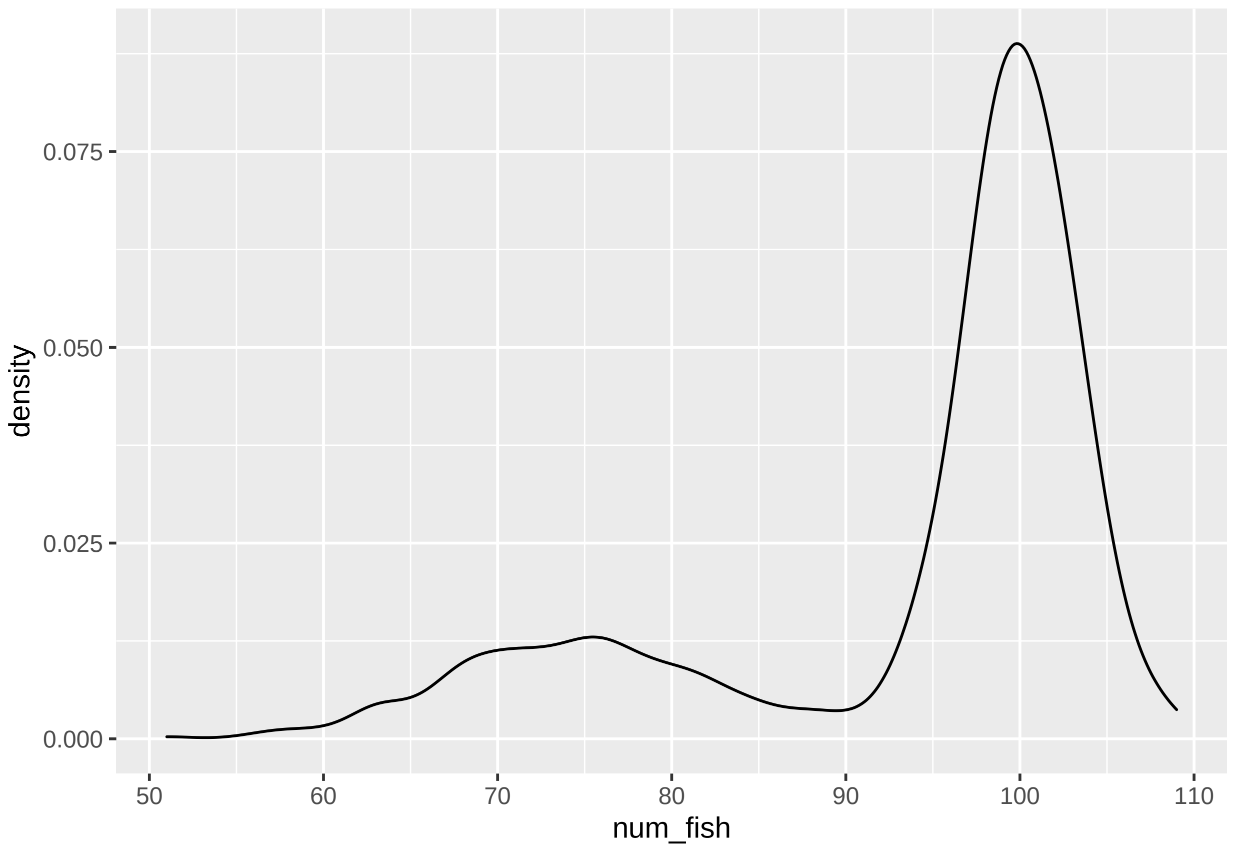


```
# we can change the number of bins (essentially, the number of columns) by modifying the bins argument
```

Density plots

Density plots are similar to histograms in that they show the distribution of a *continuous variable* but they are smoothed and continuous version estimated from the data. For this class, you can think of them as being basically the same thing (and use them interchangeably).

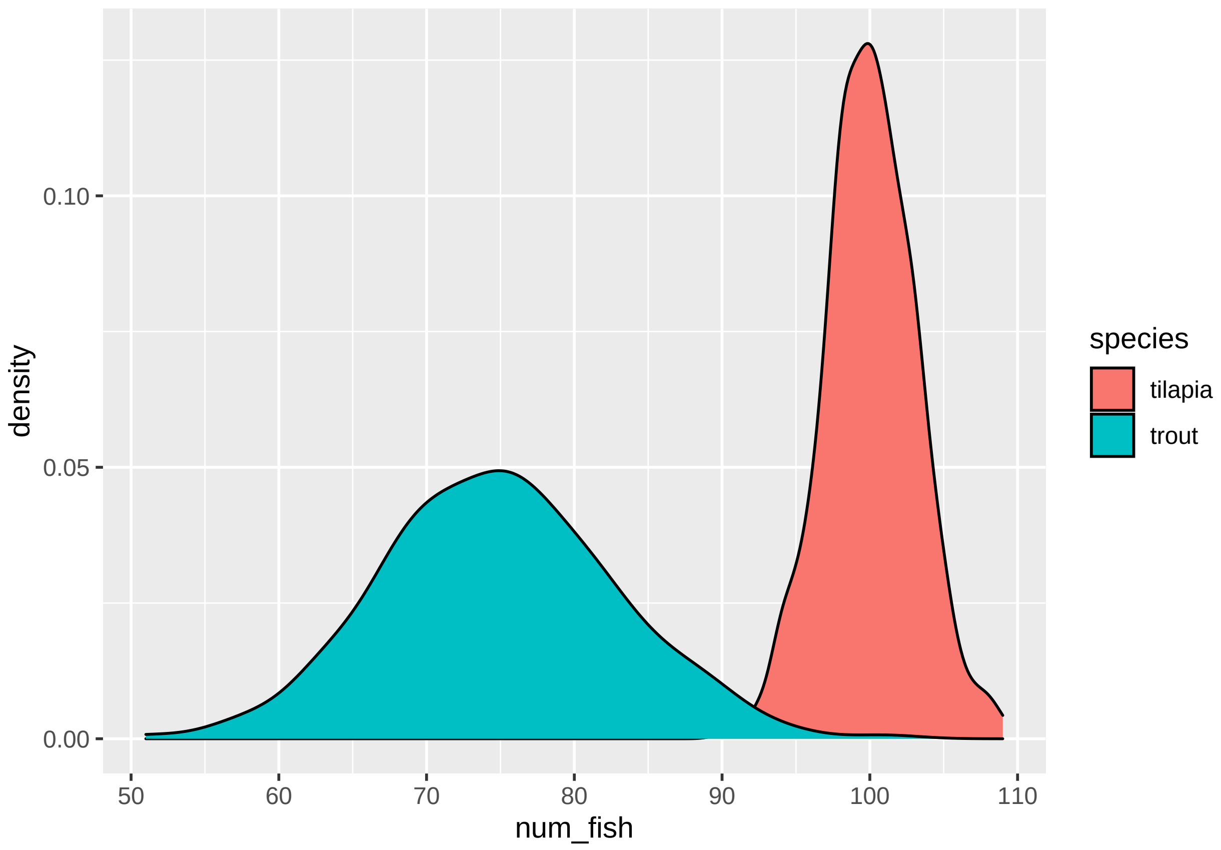
```
ggplot(tanks, aes(num_fish)) +
  geom_density()
```



Multiple Density Plots (or Histograms)

What if we wanted to create a density plot for signal distance for each collar manufacturer? In base R, we would need to filter or make a new dataframe. With `ggplot2`, we can specify that we want the `fill` (or the `color`) to be determined by the maker column.

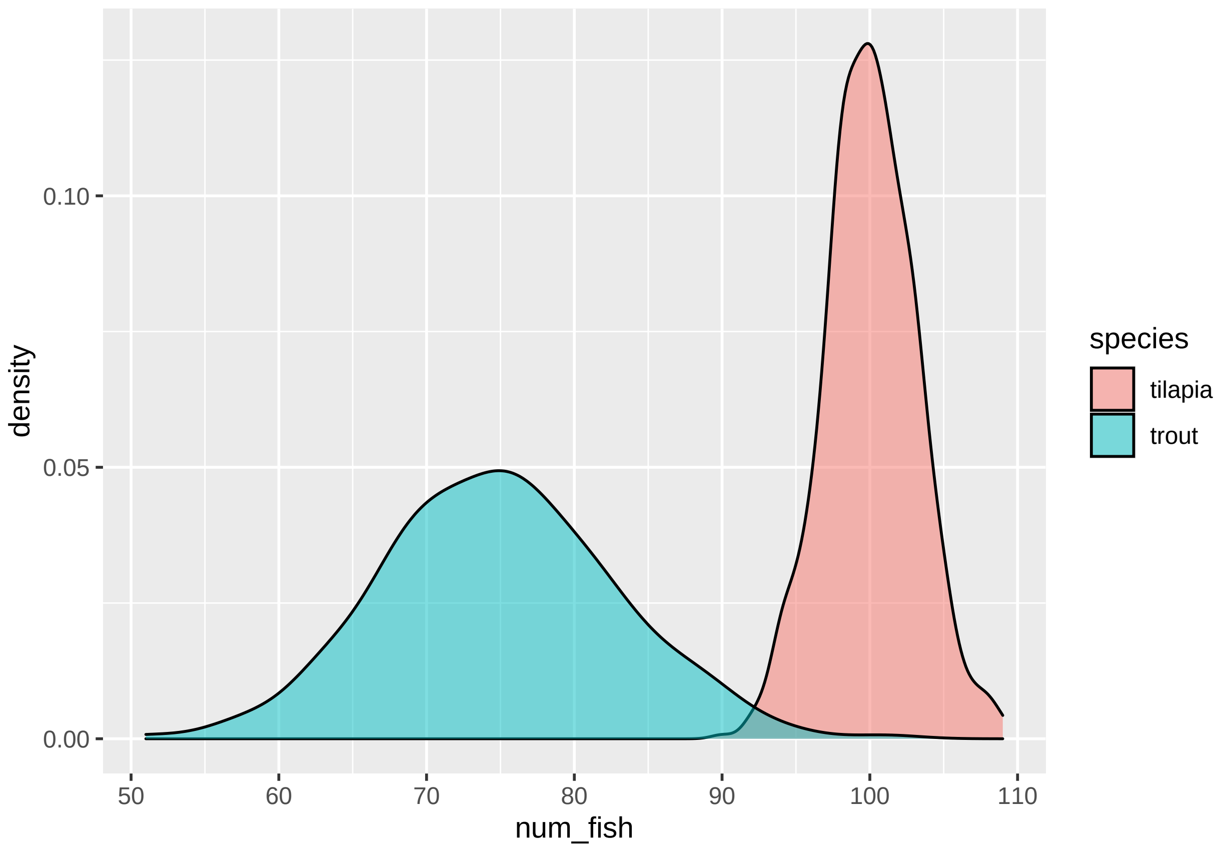
```
ggplot(tanks, aes(num_fish, fill = species)) +
  geom_density()
```



Hooray! But. . . we can't really see the bars for Budget Collars LLC in the lower range of the values because they are hidden behind the Collarium Inc. bars. How might we address this?

To fix this issue, we can use the `alpha` argument, which changes the transparency of the bars. The scale for `alpha` runs from 0 (completely transparent) to 1 (completely solid). Let's try 0.5.

```
ggplot(tanks, aes(num_fish, fill = species)) +
  geom_density(alpha = .5)
```



The powerful and pesky `aes()` function

A quick note about the `aes()` function. It's one of the more confusing bits of `ggplot2`.

When do I put the `color` (or `size` or `linetype` or `fill` or whatever) inside the `aes()` function versus in the `geom` function but outside of `aes()`? When we made our density plots, why did the `color` argument go inside of `aes()` but `alpha` went outside?

Essentially, it boils down to this:

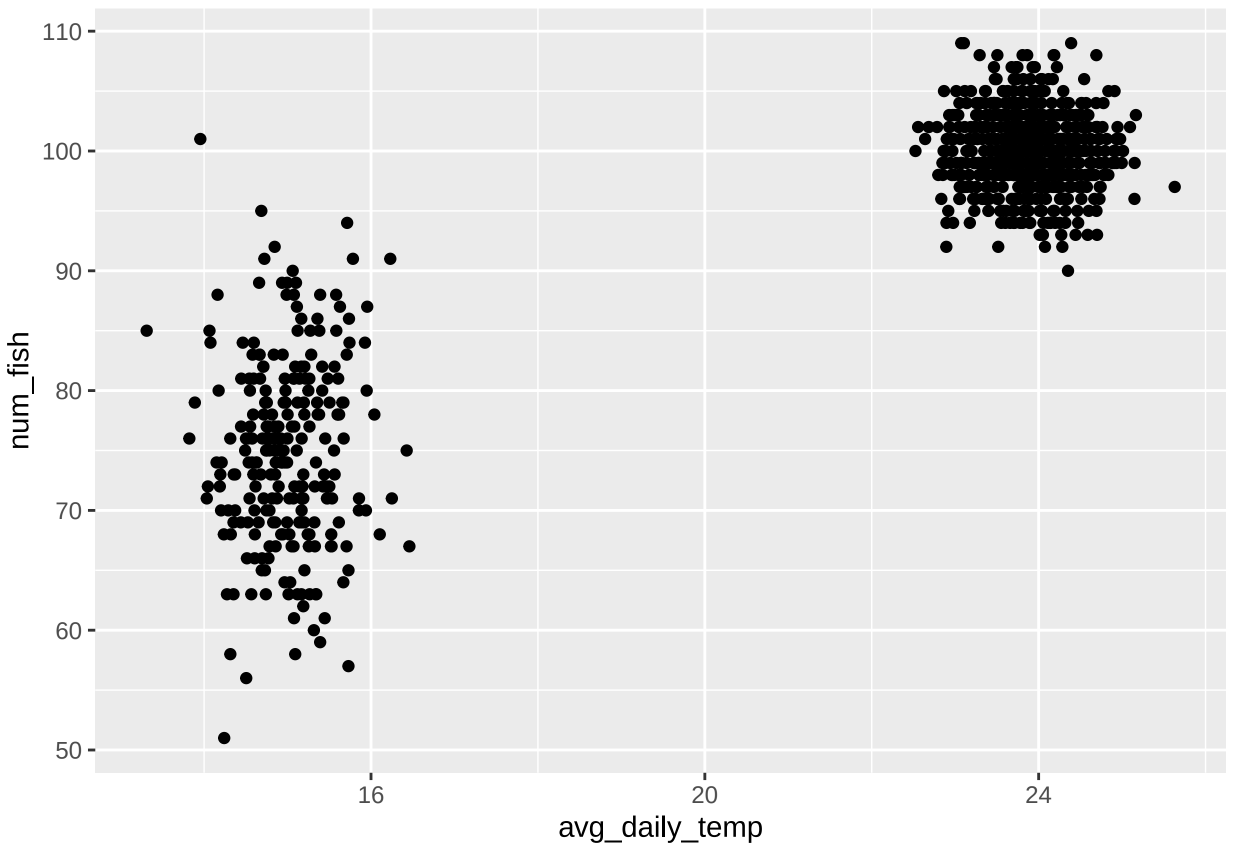
- if you want something (color, size, etc.) on your plot to change based on a **variable** from a data frame, you will want to put the argument *within* the `aes()` function.
- if you want something (color, size, etc.) on the plot to be **constant**, you will specify it *outside* of the function.

For some additional examples and explanation, check out this Stack Overflow page.

Scatter Plot

As a reminder, we use the `geom_point()` function to make a scatter plot of the relationship between *two continuous variables*.

```r
ggplot(tanks, aes(avg_daily_temp, num_fish)) +
  geom_point()
```

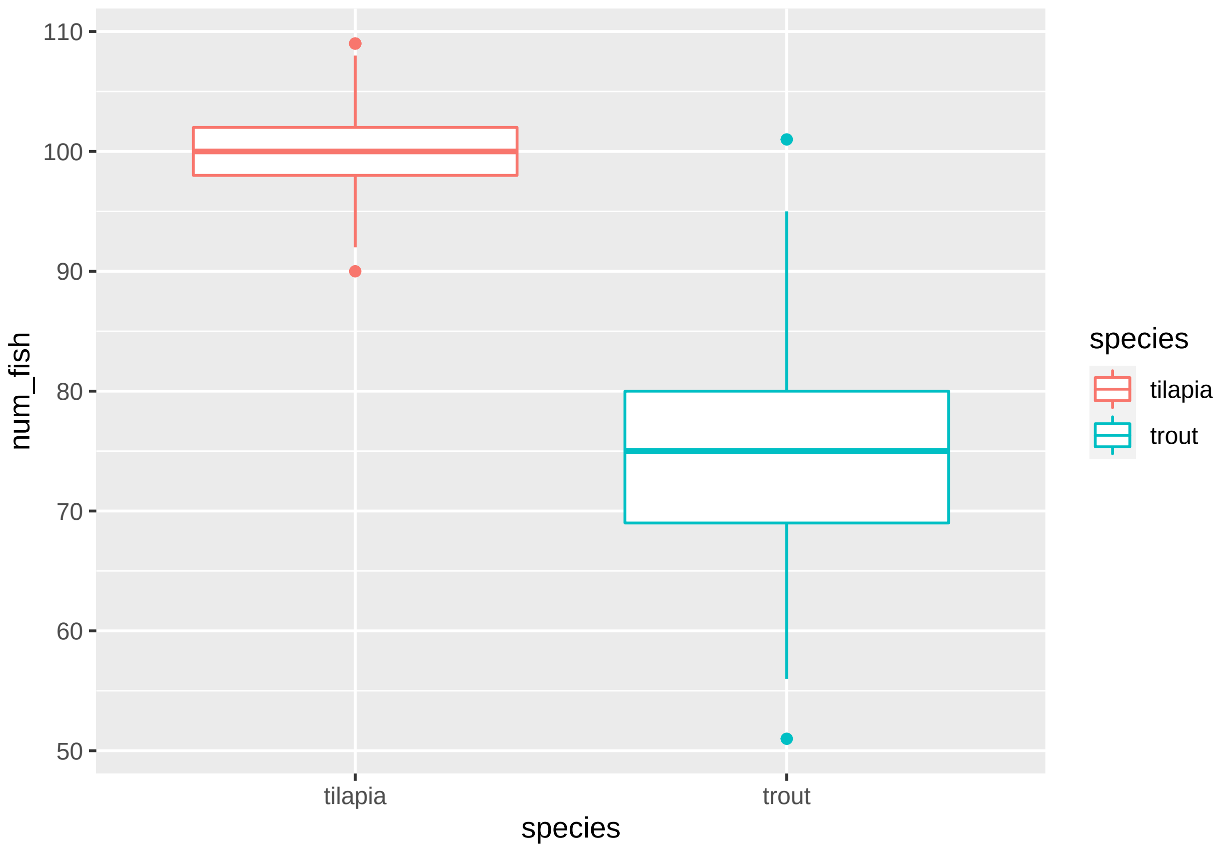


Box-and-Whisker Plots

Box-and whisker-plots (also known as box plots) are another great option for looking at one *continuous* variable and one or more *categorical* variables. They are particularly nice when you want to see measures of central tendency and variation in the same plot.

Let's build one and then talk through what each component means. We use `geom_boxplot` to make these types of plots.

```r
ggplot(tanks, aes(species, num_fish, color = species)) +
  geom_boxplot()
```



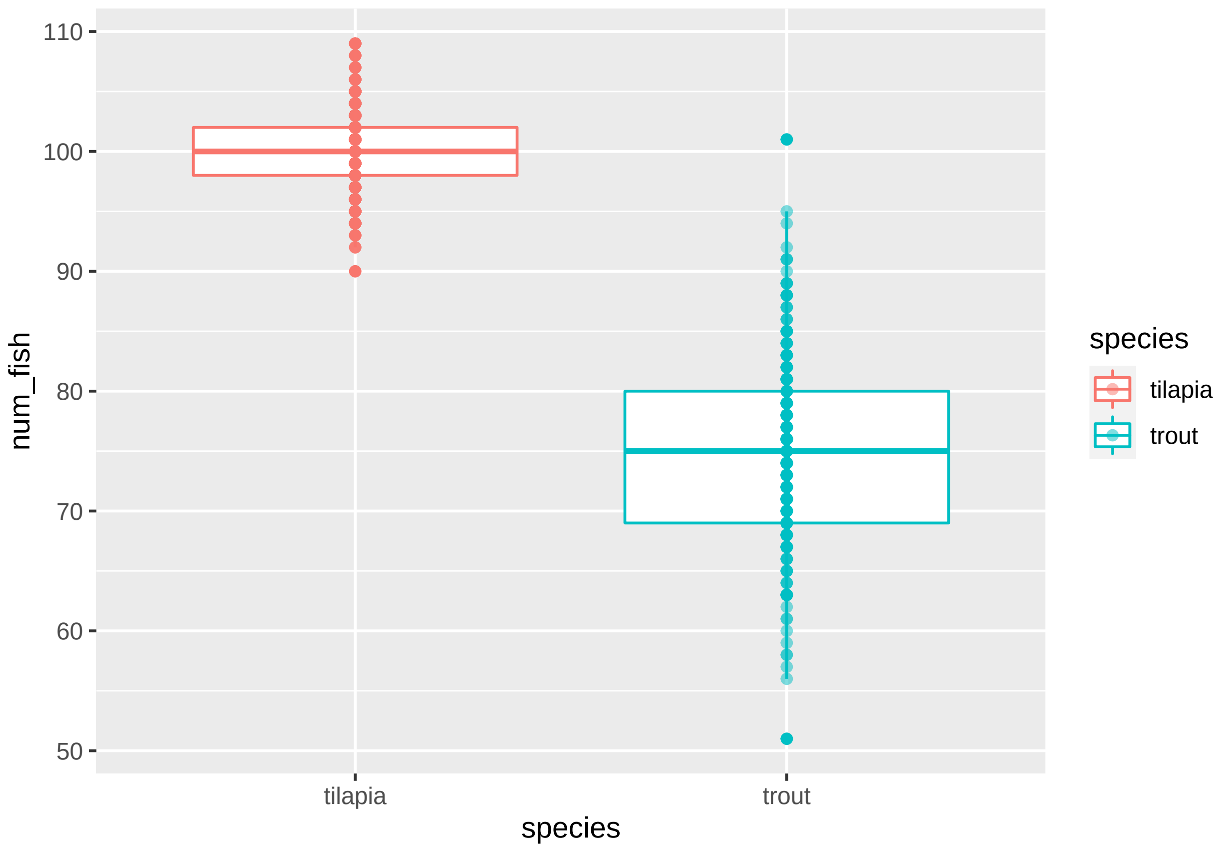
So what does the box represent? And the whiskers?

- the **box** represents the *middle 50% of the values* in the data set.
- the line that runs through the middle of the box represents the *median* ***(middle value)*** of the data
- the **whiskers** represent the *spread of the data* (we won't get into the mathematical details of exactly how they are calculated) and are roughly comparable to 95% confidence intervals (we will cover this in another module)
- values that fall outside of the whiskers can be considered outliers and are plotted individually

Layering

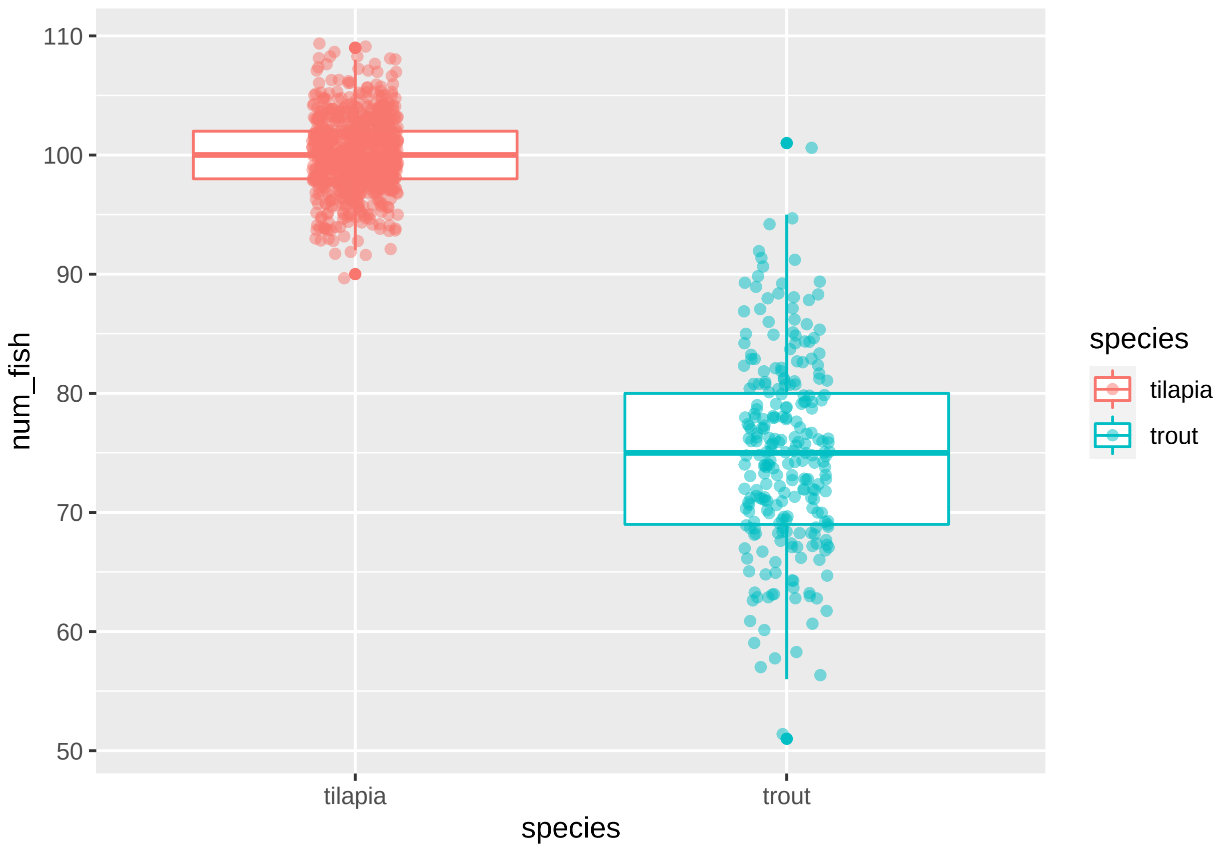
One of the beautiful parts of working with `ggplot2` is that you can add multiple layers to each plot. One of the key things missing from box-and-whisker plots is any indication of how many data points we have. In the plot above, there could be 5 tanks per species or 500 tanks per species. How can we add an indication of how many points there are? We can layer each indiividual data point on top of the boxes!

```r
ggplot(tanks, aes(species, num_fish, color = species)) +
  geom_boxplot() +
  geom_point(alpha = 0.5)
```



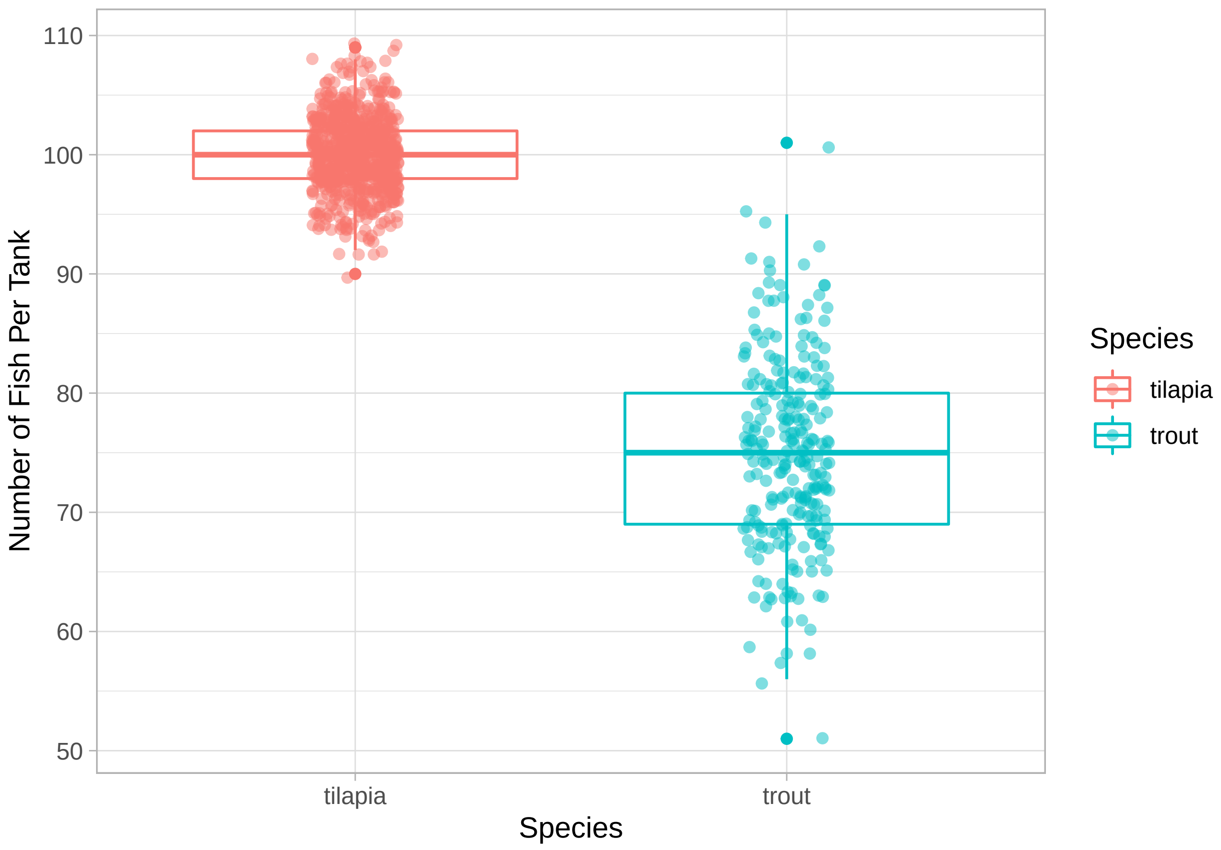
The `geom_jitter()` function is a special version of `geom_point()`. It adds a little bit of randomness to the points (both horizontally and vertically) so that they don't overlap as much. We can control how much randomness we allow with the `width` and/or `height` arguments. I usually ignore `height` and set the `width` argument to 0.1.

```
ggplot(tanks, aes(species, num_fish, color = species)) +
  geom_boxplot() +
  geom_jitter(alpha = 0.5, width = 0.1)
```



That is looking really nice! We can keep improving it, though, with better labels for the axes and the legend as well as a nice `theme`.

```
ggplot(tanks, aes(species, num_fish, color = species)) +
  geom_boxplot() +
  geom_jitter(alpha = 0.5, width = 0.1) +
  labs(x = "Species",
       y = "Number of Fish Per Tank",
       color = "Species") +
  theme_light()
```



Summary

Let's summarize some of what we've learned in this module

Data Visualization Types and When to Use Them

Histogram or Density Plot

Good for looking at the distribution of one continuous variable

- one continuous variable (x-axis)

Multiple Histogram or Multiple Density Plot

Good for looking at differences in the distributions of one continuous variable based on a categorical variable

- one continuous variable (x-axis)
- one categorical variable via the `fill` or `color` argument in the `aes()` function

Scatter Plot

Good for looking for the relationship between two continuous variables

- two continuous variables (x-axis and y-axis)
- can add in a categorical variable via `aes()`, but the main relationship is between the two continuous variables

Box Plot

Good for looking at measures of central tendency and variation for a continuous variable and the differences between those measures between categories

- one continuous variable (y-axis)
- at least one categorical (x-axis and additional via `aes()`)

Layering

We can add multiple layers to ggplots, which is part of what makes them so useful!

- we can add multiple `geom` functions to a single plot
- we use the `labs()` function to rename axes labels and legends
- we use a `theme` function to make the plot more aesthetically pleasing and easier to understand