Intro to Data Visualization

This Section

You can take an entire class on **data visualization**. Sadly, the guiding principles and theory of data visualization is beyond the scope of our current class.

So what will you learn in this section?

1. (This lecture) Reminder of the basics of `ggplot2` & some new things
2. Changing the gestalt/look/feel/mood of your plot
    - Color palettes
    - Themes
    - Manually changing very specific things
3. Breaking down and adding to plots
    - Make a multi-panel figure from your data
    - Combine separate plots into a single multi-panel figure
    - Add things like lines, text etc.
4. Going further
    - Making scatterplots more readable
    - Combine multiple types of plot into a single, informative figure

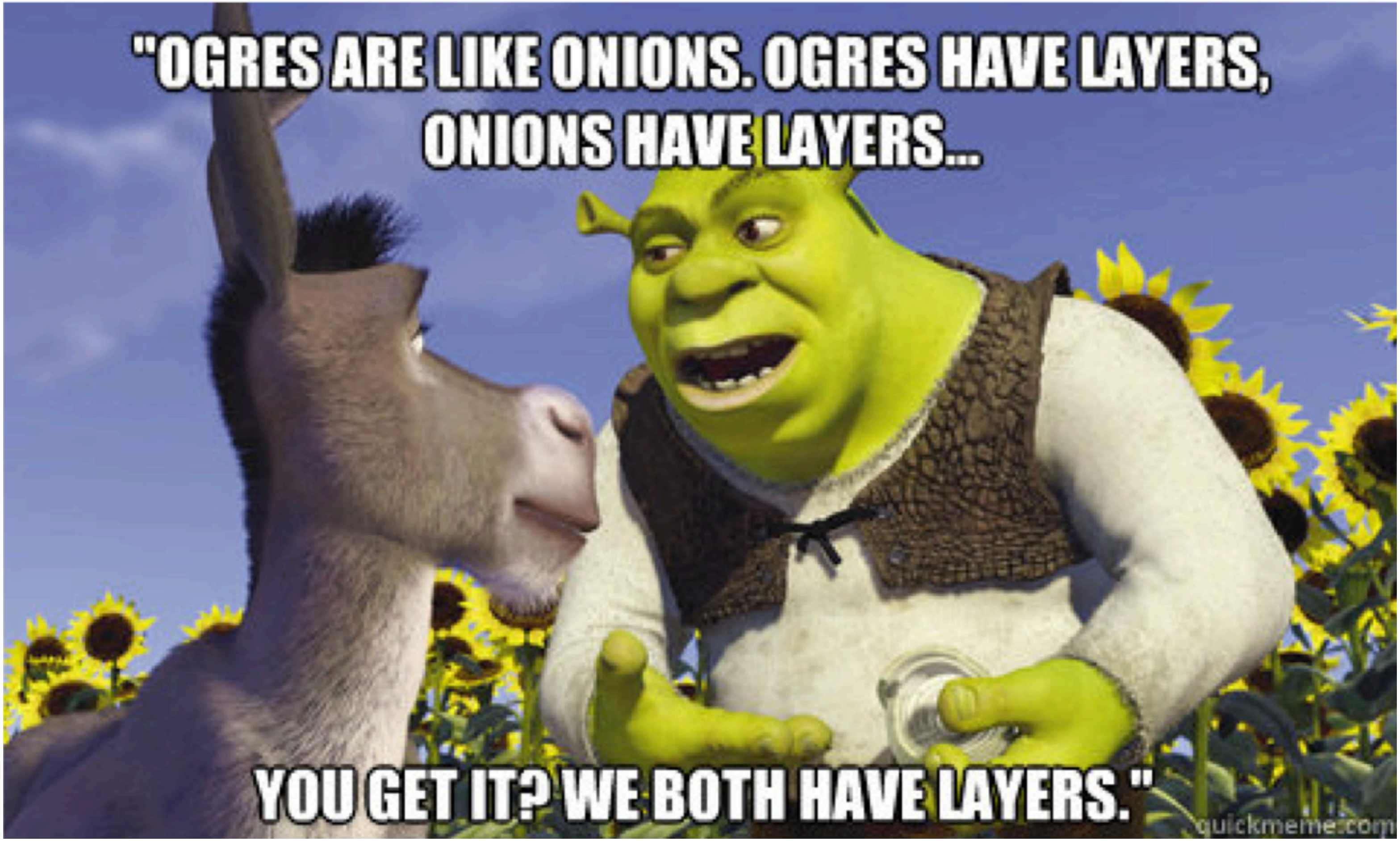
What you'll need

Please make sure the following packages are installed:

- `tidyverse` (this includes `dplyr` and `ggplot2`)
- `RColorBrewer`
- `ggpubr`
- if you like Wes Anderson movies (The Royal Tennenbaums, Life Aquatic, Isle of Dogs, Darjeeling Limited, Rushmore, The Grand Budapest Hotel, and more!), install the `wesanderson` package

Remember: do NOT memorize anything! You will always have these slides to refer back to, and Google is your friend. Looking things up is a good thing!

Recap of `ggplot2`



Recap of `ggplot2`

- The `gg` in `ggplot2` stands for *"the grammar of graphics"*
- `ggplot2` has the following structure:

```
ggplot(things that impact the entire plot) +
  geom_something(things that impact just the something)
```

Plotting with `ggplot2`

`ggplot2` has the following structure:

```
ggplot(things that impact the entire plot) +
  geom_something(things that impact just the something)
```

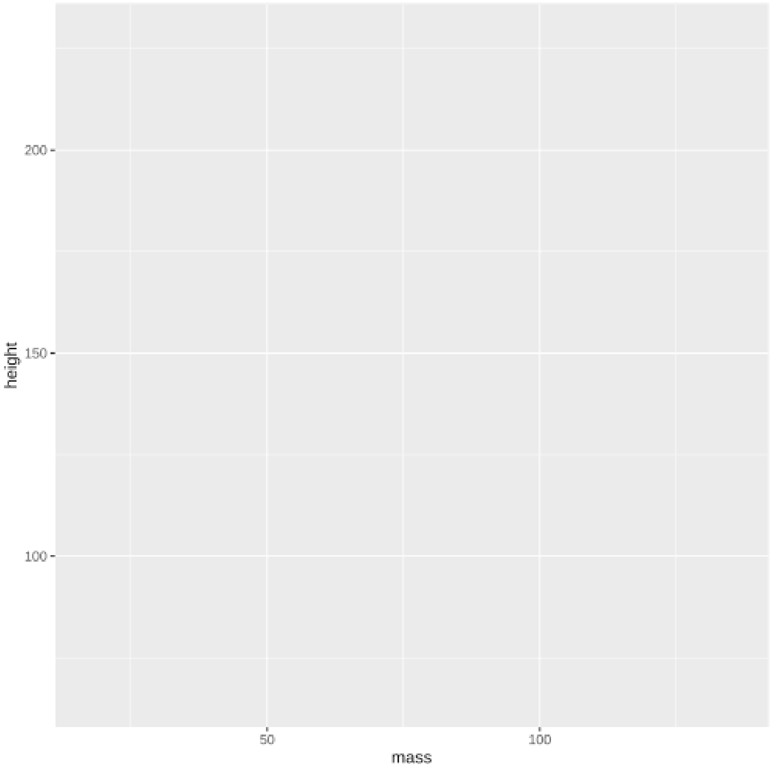
Things like:

- data.frame used for plotting
- defining your x & y axes

Layer 1: `ggplot`

What happens if you only input your data and what you want your x- and y-axes to be?

```
ggplot(data = empire, aes(x = mass, y = height))
```



Layer 2: `geom_`

`ggplot2` has the following structure:

```
ggplot(things that impact the entire plot) +
  geom_something(things that impact just the something)
```

`geom_` typically means **shape**. What shapes do you want to use to represent your data in the plot?

- `geom_histogram` -- histogram
- `geom_density` -- distributions
- `geom_violin` -- distributions
- `geom_point` -- scatter plot
- `geom_col` -- bar plot

These are **functions** that are searchable in the help pages!

Plotting with `ggplot2`

The functions `ggplot()` and `geom_()` can take on different **aesthetics** as an argument, using `aes()`.

Aesthetics are how you control what you want your plot to look like; how can you make it pretty? Examples:

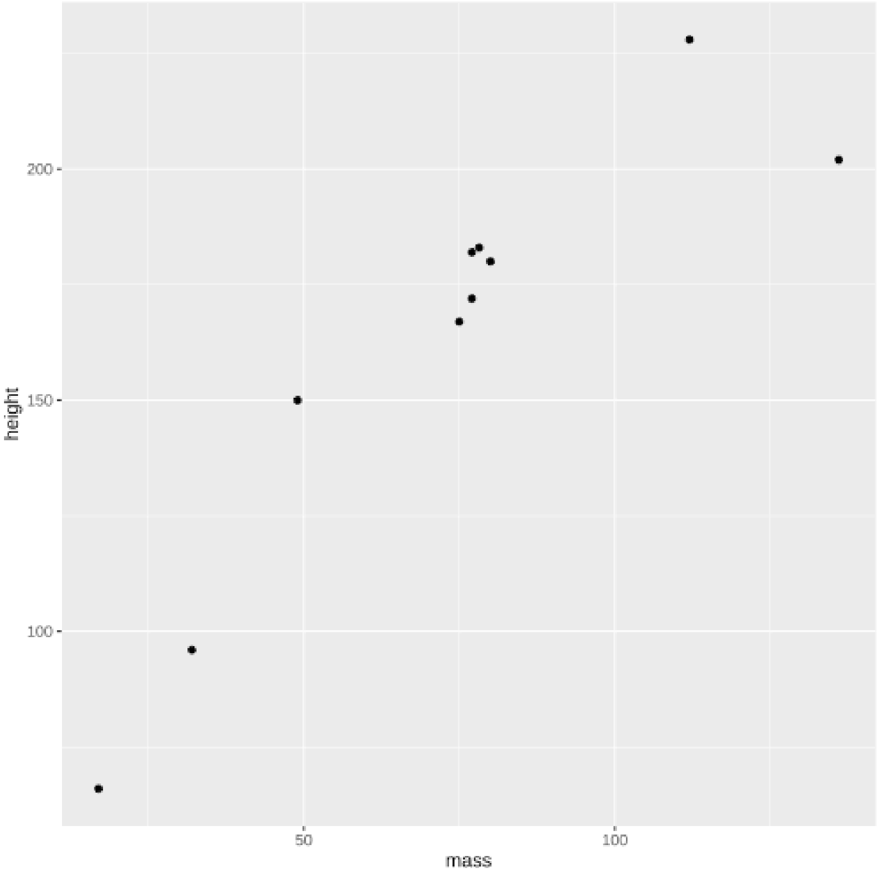
- Which variables are the `x-` and `y-` axes?
- `color` (should you color the plot by some variable?)
- `fill` (very similar to `color`, should you fill the plot in somehow; used for bar graphs and boxplots)
- `shape` (do you want groups to have different shaped points?)
- `size` (how big should plotted data be?)

When to use aesthetics

Let's say you want to change something about your plot...

- If the thing you want to change ***is based on the data***, put it inside an aesthetics `aes()`
- If it is ***not based on the data***, keep it outside the `aes()`

```
ggplot(data = empire,
       aes(x = mass,
           y = height)) +
  geom_point()
```

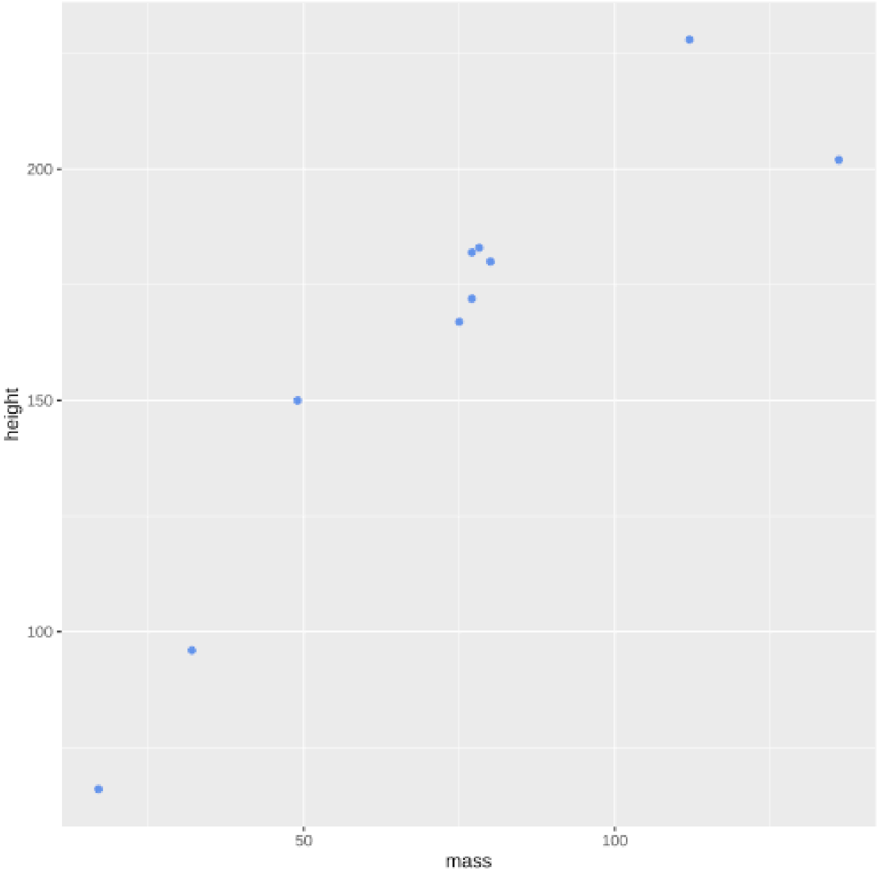


When to use aesthetics

Let's say you want to change something about your plot...

- If the thing you want to change ***is based on the data***, put it inside an aesthetics `aes()`
- If it is ***not based on the data***, keep it outside the `aes()`

```
ggplot(data = empire,
       aes(x = mass,
           y = height)) +
  geom_point(color = "cornflowerblue")
```

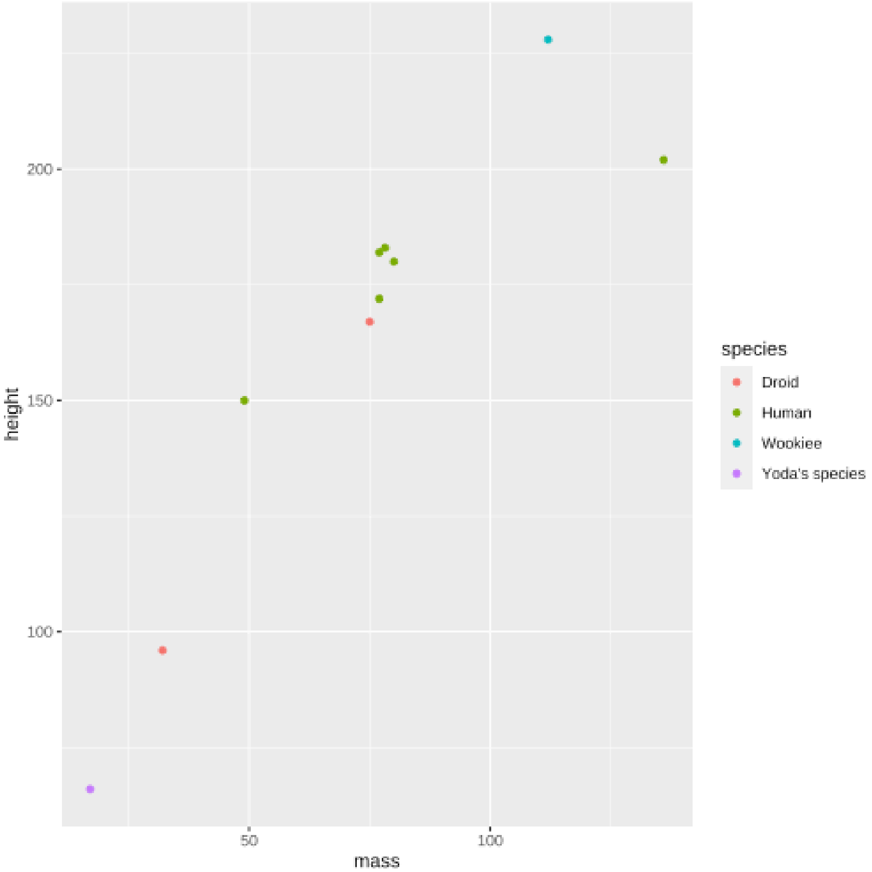


When to use aesthetics

Let's say you want to change something about your plot...

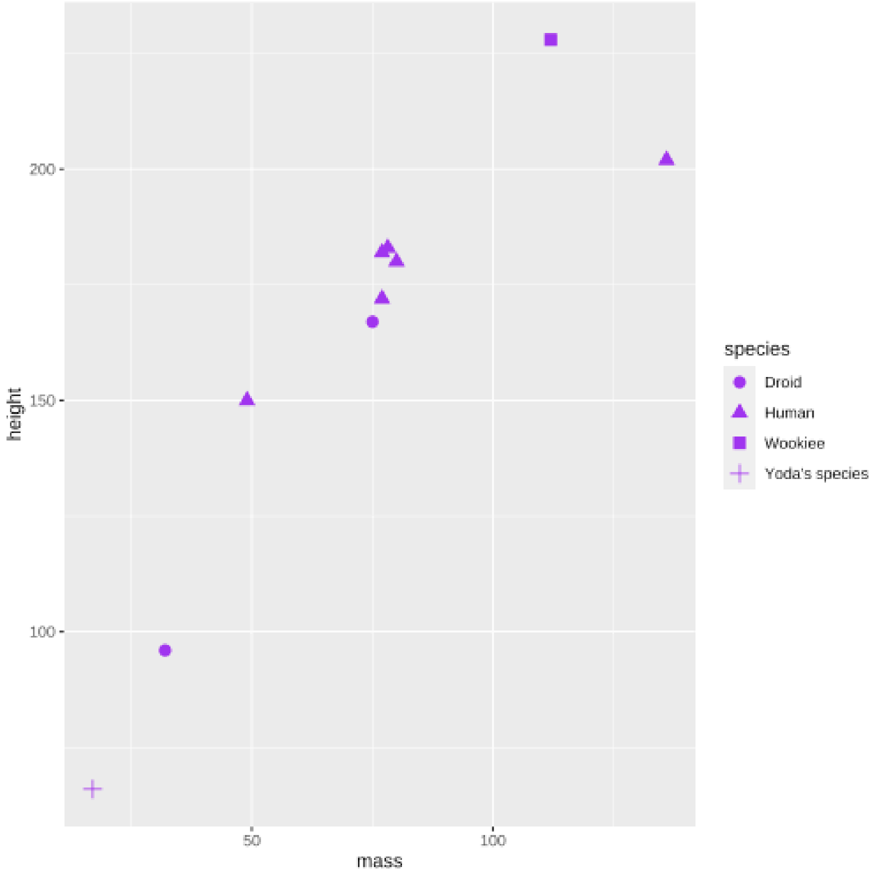
- If the thing you want to change ***is based on the data***, put it inside an aesthetics `aes()`
- If it is ***not based on the data***, keep it outside the `aes()`

```
ggplot(data = empire,
       aes(x = mass,
           y = height)) +
  geom_point(aes(color = species))
```



Mix & Match

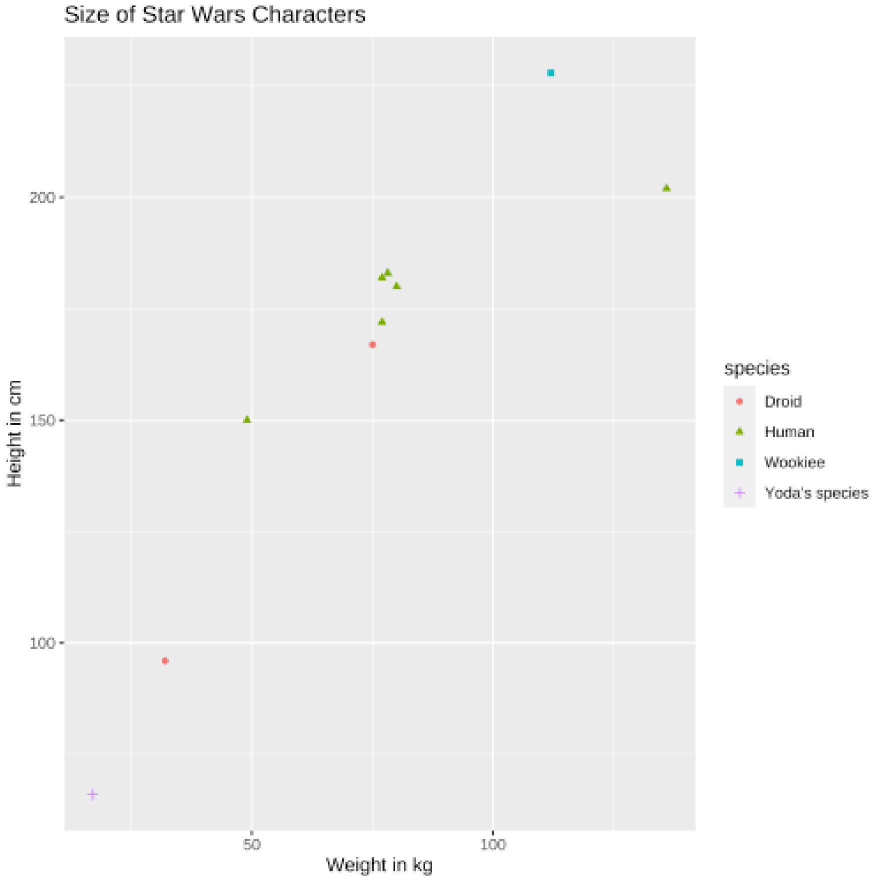
```
ggplot(data = empire,
       aes(x = mass,
           y = height)) +
  geom_point(color = "purple",
             aes(shape = species),
             size = 3)
```



Layer 3: `labs`

- Layers on layers on layers!
- To add labels to your plot, add a new layer that uses the `labs()` function

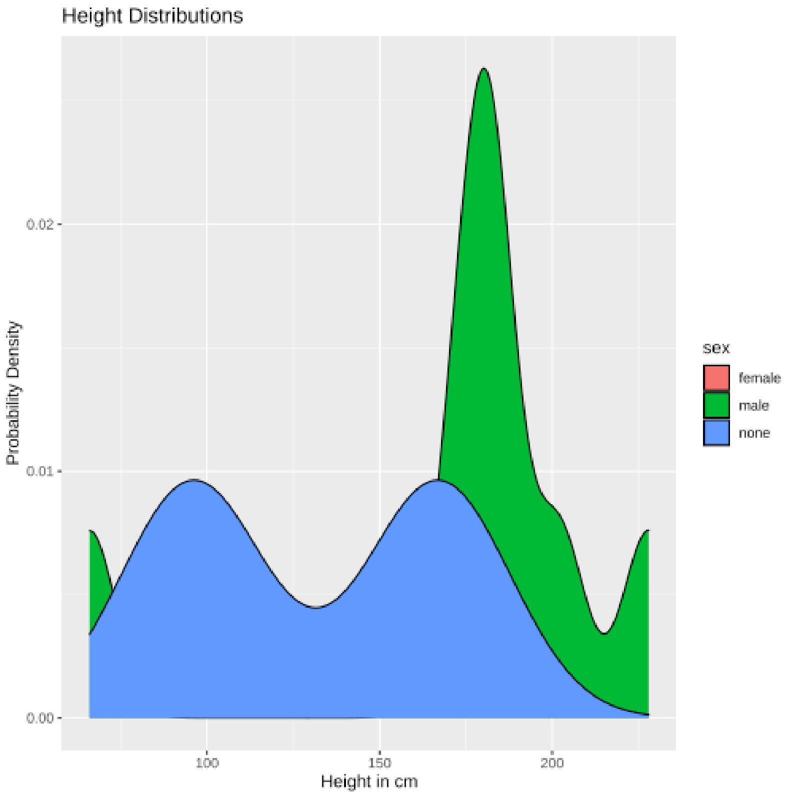
```
ggplot(data = empire,
       aes(x = mass,
           y = height)) +
  geom_point(aes(color = species,
                 shape = species)) +
  labs(x = "Weight in kg",
       y = "Height in cm",
       title = "Size of Star Wars Character
```



Overlapping Data

- `alpha =` is an argument that controls transparency (1 = opaque, 0 = transparent)
- *Without* setting the `alpha` parameter...

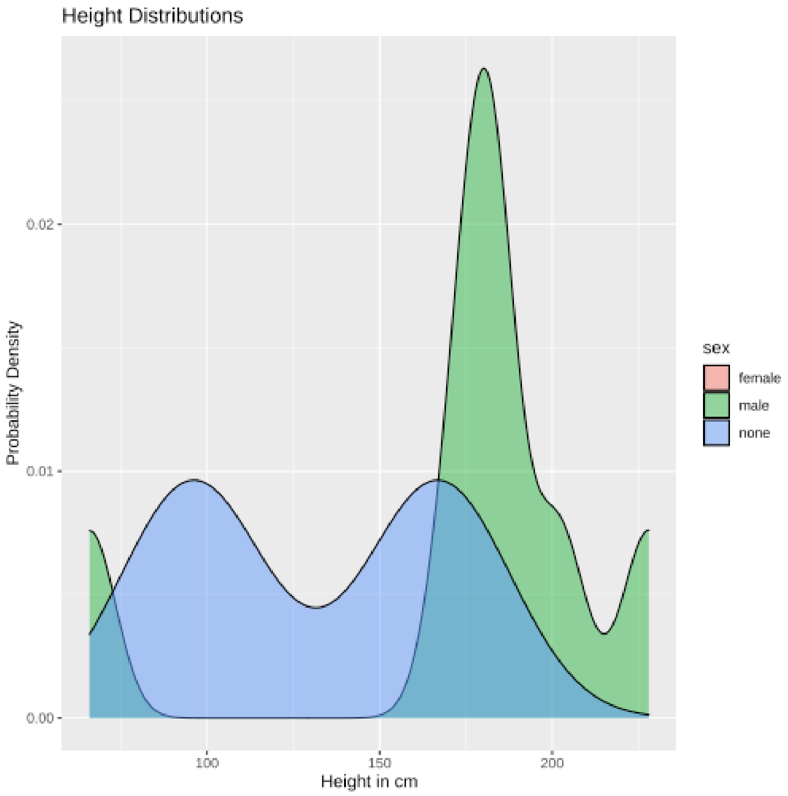
```
ggplot(data = empire,
       aes(x = height)) +
  geom_density(aes(fill = sex)) +
  labs(x = "Height in cm",
       y = "Probability Density",
       title = "Height Distributions")
```



Overlapping Data

- `alpha =` is an argument that controls transparency (1 = opaque, 0 = transparent)
- *With* setting the `alpha` parameter...

```
ggplot(data = empire,
       aes(x = height)) +
  geom_density(aes(fill = sex),
               alpha = .5) +
  labs(x = "Height in cm",
       y = "Probability Density",
       title = "Height Distributions")
```



Re-organizing Factors

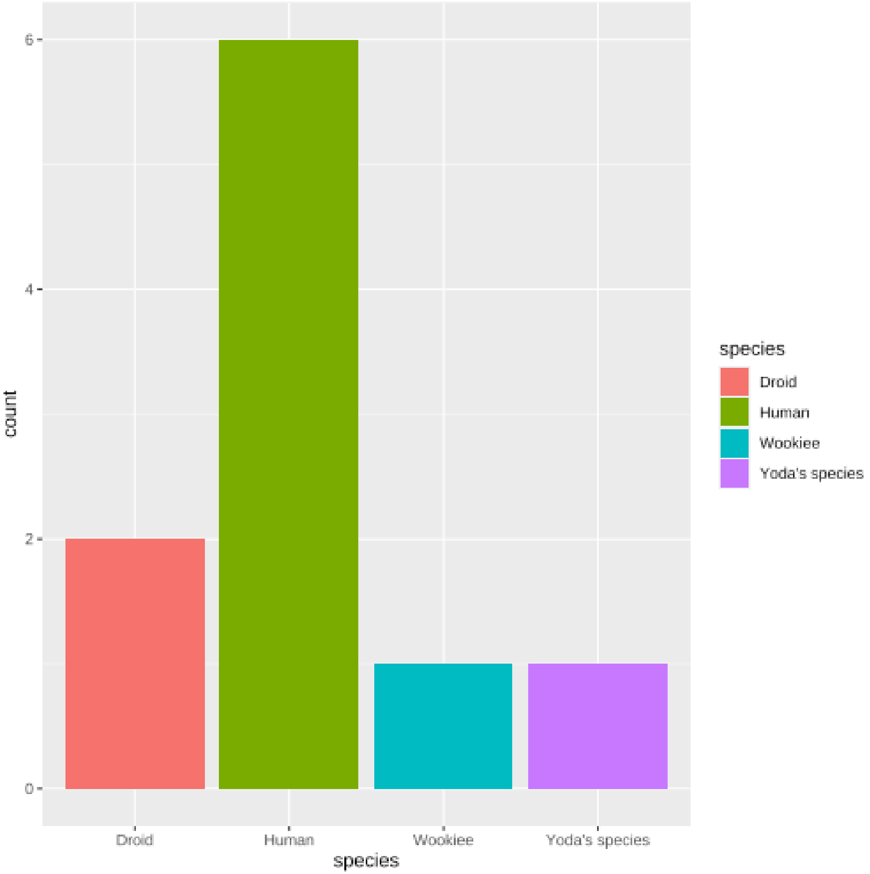
- In R, the levels of each factor are based on alphabetical order
- This means when you go to plot, your factors will show up in the same order.
- For example:
    - `geom_bar()` uses `count` as default for the y-axis
    - That means, it gives you the same information as `table()`

```
table(empire$species)
```

```
## 
##          Droid          Human        Wookiee Yoda's species 
##              2              6              1              1
```

Re-organizing Factors

```
ggplot(data = empire,
       aes(x = species)) +
  geom_bar(aes(fill = species))
```



Re-organizing Factors

Let's say we want the order (from left to right) to be Human, Yoda's species, Wookiee, Droid.

- To do this, we need to re-order the `species` factor like so:

```
empire$species <- factor(empire$species,
                         levels = c("Human",
                                    "Yoda's species",
                                    "Wookiee",
                                    "Droid"))
```

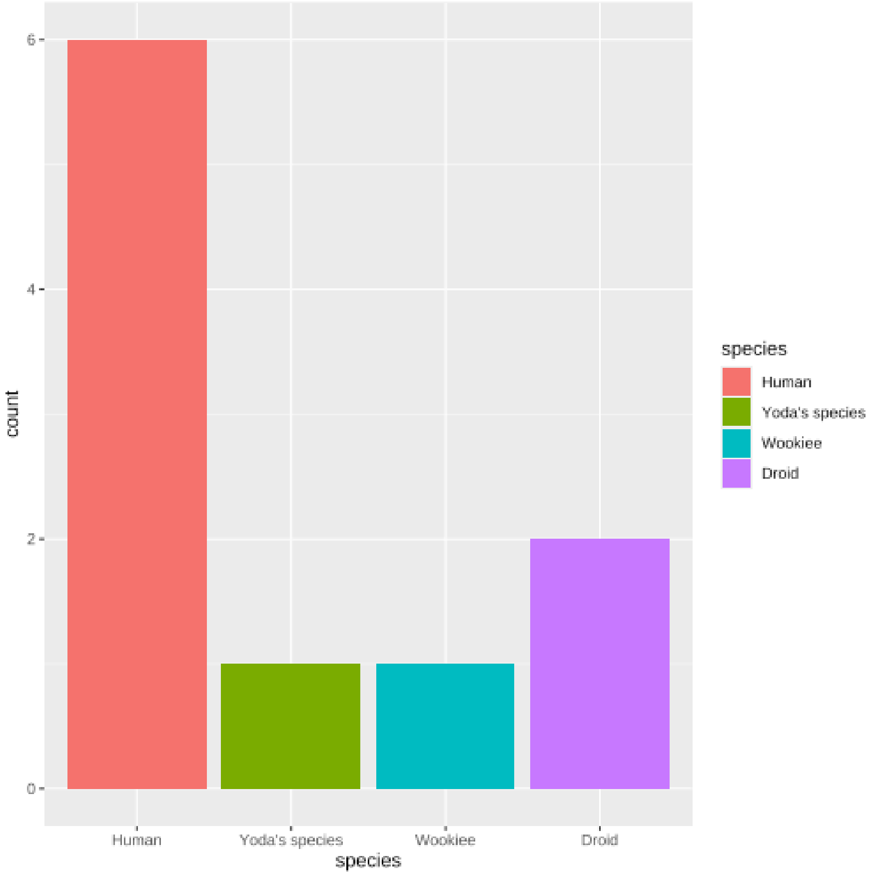
Now, if we simply go the tables of counts again, it should match our new ordering:

```
table(empire$species)
```

```
## 
##          Human Yoda's species        Wookiee          Droid 
##              6              1              1              2
```

Re-organizing Factors

```
ggplot(data = empire,
       aes(x = species)) +
  geom_bar(aes(fill = species))
```



Next time...

- How to assign certain colors to certain factor levels (e.g., make "Droid" blue and "Yoda's species" maroon)
- Color palettes
- Themes
- Changing the background of your plot