Using themes in `ggplot2`

Data Science for Biologists, Fall 2020

Stephanie J. Spielman

> WORD TO THE WISE: You will often be tempted to google how to use `ggplot2`. This is absolutely the right thing to do, but be careful!! Not all internet posts are created equally, and *many webpages or blogs may use DIFFERENT VERSIONS of* `ggplot2`. When asking the internet for help, always make sure you can locally (on your computer/in RStudio Cloud) run and understand little code tidbits you find. Remember: just because a page looks helpful doesn't mean it is.

We use the term *theme* to refer to all non-data components of a plot, including things like "where is the title placed?", "how large is the axis text?", "does the plot have a grid and how spaced out is the grid?", "is there a background fill to the plot canvas itself?", etc.

Themes are comprised of several different types of elements:

- **`element_text()`: How do certain text elements appear?**
  - For example, `axis.title` (axes titles) is an `element_text()`.
  - Among other options, you can change their:
    - `color`
    - `size`
    - `face`: "plain", "italic", "bold", or "bold.italic)
    - `angle`
    - `hjust`: "horizontal justification" where 0 is far left, 0.5 is middle, and 1 is far right, and anything in between!
    - `vjust`: "vertical justification" where 0 is bottom, 0.5 is middle, and 1 is top, and anything in between!)

- **`element_line()`: How do certain line elements appear?**
  - For example, `axis.line` (axis lines themselves) is an `element_line()`.
  - Among other options, you can change their `color`, `size`, `linetype` ("solid", "dashed", "dotted", "dotdash", "longdash", and "twodash"), and `lineend` ("round", "butt",

"square").

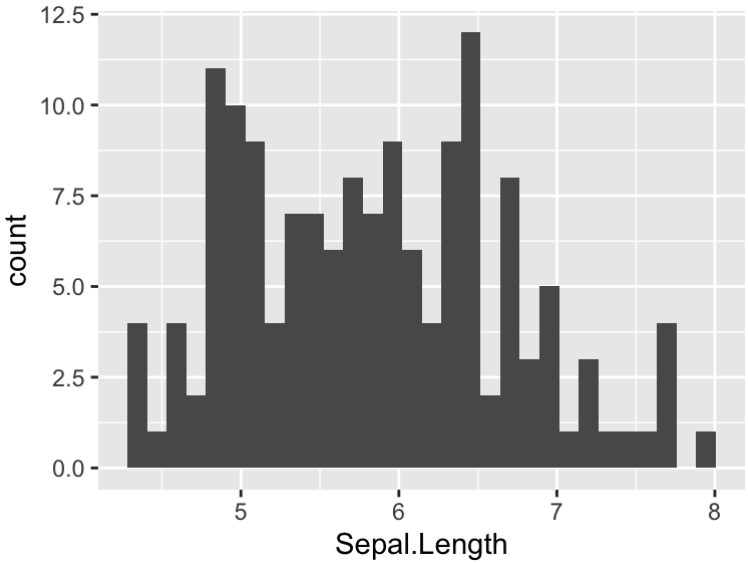
- **`element_rect()` : How do certain *borders and background* elements in the plot appear?**
  - For example, `plot.background` (the background/"canvas" appearance of the plot) is an `element_rect()`.
  - Among other options, you can change their:
    - `color`
    - `fill`
    - `size` (of the outline)
    - `linetype` ("solid", "dashed", "dotted", "dotdash", "longdash", and "twodash")

- **`unit()` :** Some theme components are not strictly elements but are defined in terms of size only and will be termed `unit()` or `grid::unit()`.

All theme components can be fully customized, as is overwhelmingly documented here (https://ggplot2.tidyverse.org/reference/theme.html). As we have previously seen, there are also several existing complete themes (https://ggplot2.tidyverse.org/reference/ggtheme.html) in `ggplot2`. Now we'd like to learn how to further customize our plots with our own themes.

All examples below will modify a version of this plot which uses the default `theme_gray()` with no further customization:

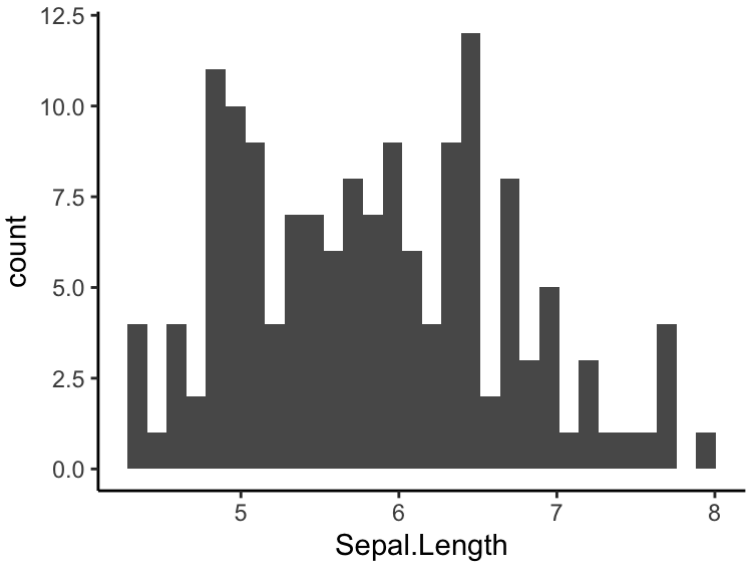
```
ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram()
```



Specifying a complete theme

For a given plot, simply add the theme of interest. For example:

```
ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram() +
  theme_classic() # Set the theme as theme_classic() by adding
```



Customizing a theme

When customizing a theme, you need to:

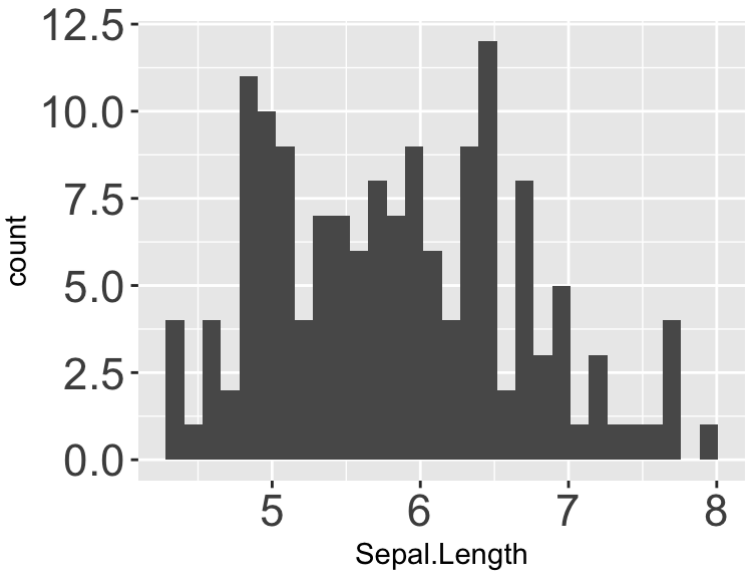
- Search through the documentation (https://ggplot2.tidyverse.org/reference/theme.html) (literally) to find how `ggplot2` refers to the particular plot component you want to change.
- Use the documentation (https://ggplot2.tidyverse.org/reference/element.html) (sensing a pattern?) to determine whether that component goes with `element_text()`, `element_line()`, or `element_rect()`.
    - Or, decide if you want to *entirely remove* that particular plot component. Stay tuned...
- *Add it* to your plot inside the `theme()` function.
    - If you are also specifying a complete theme in your plot code but want to change certain components, make sure to add these components *after* you specify the complete theme. Otherwise, the complete theme settings will *override* your customizations.
- There is a *very very VERY* useful function you can use as part of the code call `rel()` which will help you to change sizes of different elements *relative to the baseline*. `rel(1.1)` means 10% larger than default; `rel(0.9)` means 90% the size of the default.

Quick examples of `element_text()`

Let's see some examples using `axis.text` [1], which controls both X and Y axis *text* (aka the labels along the axes, NOT the title). Customizing `axis.text` must be paired with `element_text()`, as it is text!

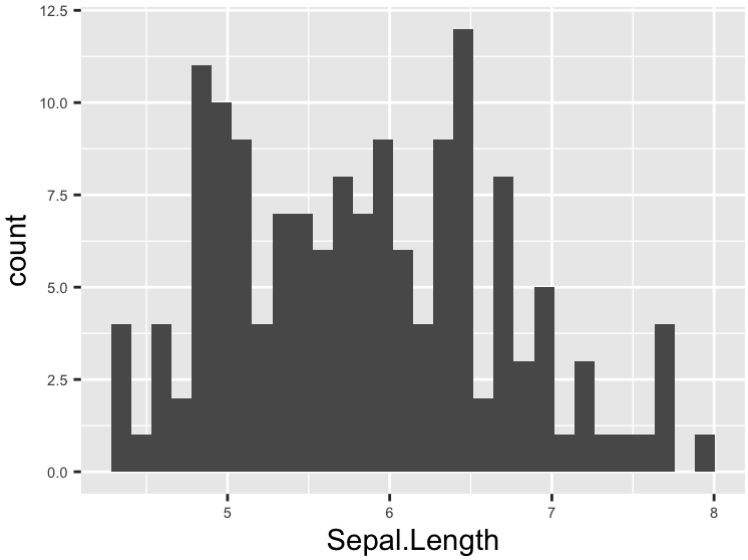
Make the axis text 50% larger than its default:

```
ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram() +
  #theme( plot-thing-to-change = element_<YPE>(things to customize about it) )
  theme(axis.text = element_text(size = rel(1.5)))
```



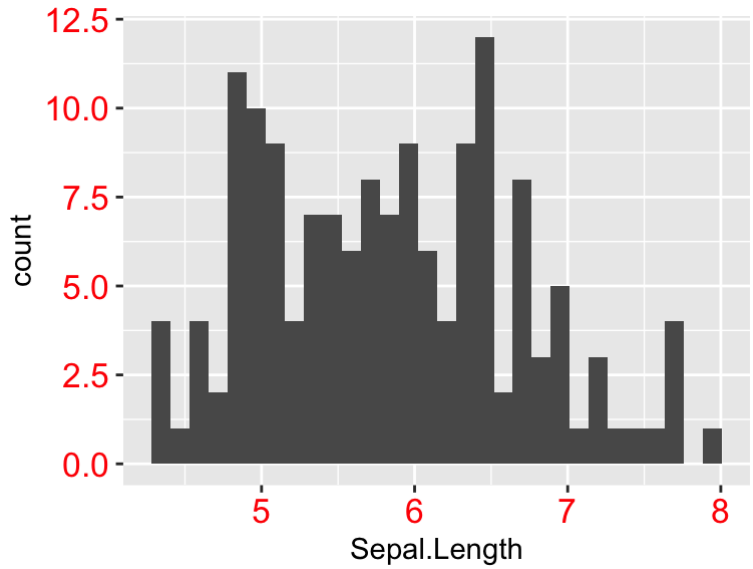
Make the axis text 50% smaller than its default:

```
ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram() +
  theme(axis.text = element_text(size = rel(0.5)))
```



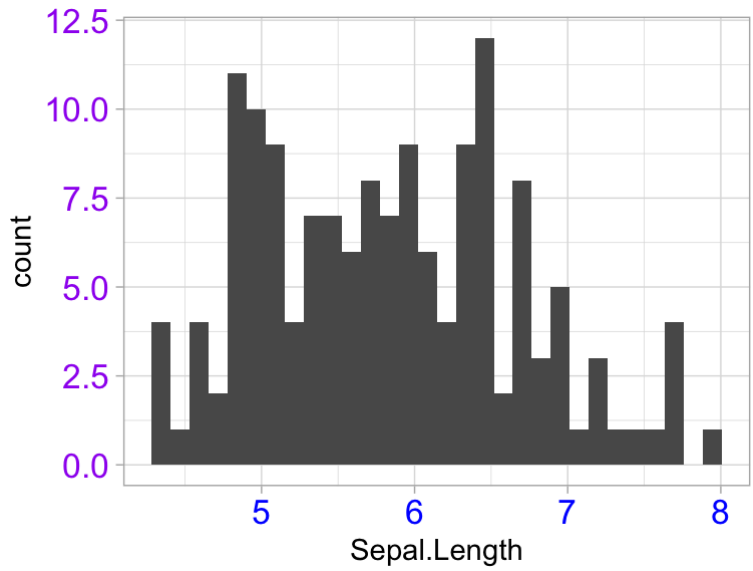
Make the axis text red, *and* 15% larger than default:

```
ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram() +
  theme(axis.text = element_text(color = "red", size = rel(1.15)))
```



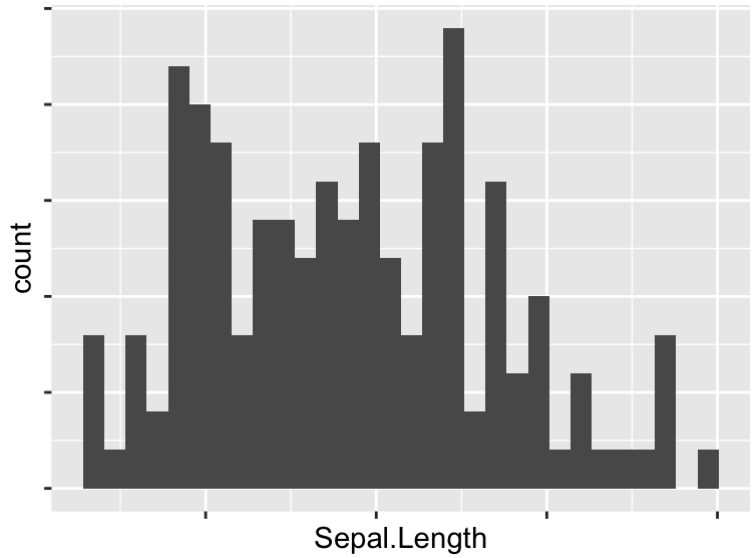
Let's change a whole bunch of things! Change baseline theme to `theme_light()`, make BOTH X- and Y- axis have 15% larger font size, make the X-axis text blue, and make the Y-axis text purple.

```
ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram() +
  theme_light() +
  theme(axis.text = element_text(size = rel(1.15)), # both axes
        axis.text.x = element_text(color = "blue"), # only the x!
        axis.text.y = element_text(color = "purple")) # only the y!
```



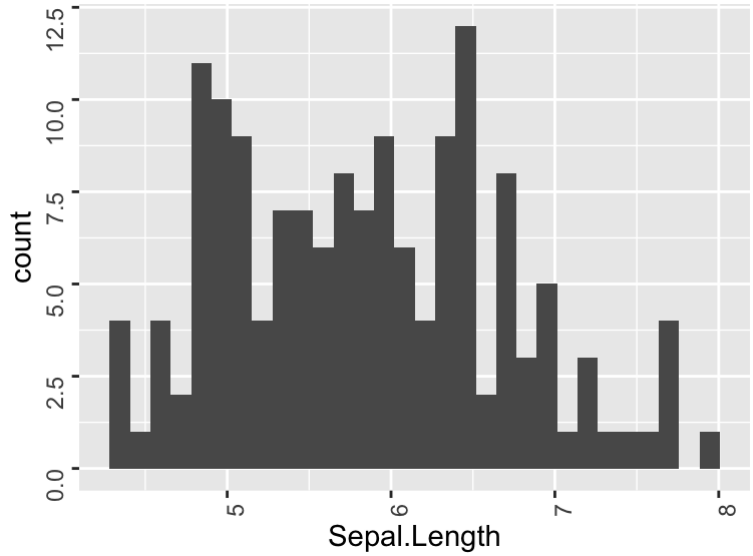
You can **REMOVE** theme elements entirely with `element_blank()`. The example below is obviously bad plot design, but shows how to get the job done if you wanted to!

```
ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram() +
  theme(axis.text = element_blank())
```



Rotate the text by a 90% angle

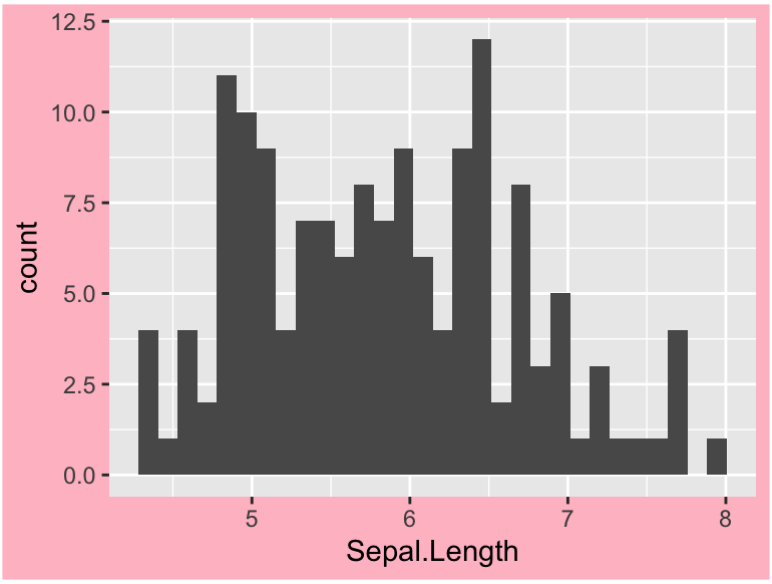
```
ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram() +
  theme(axis.text = element_text(angle = 90))
```



Quick examples of `element_rect()`

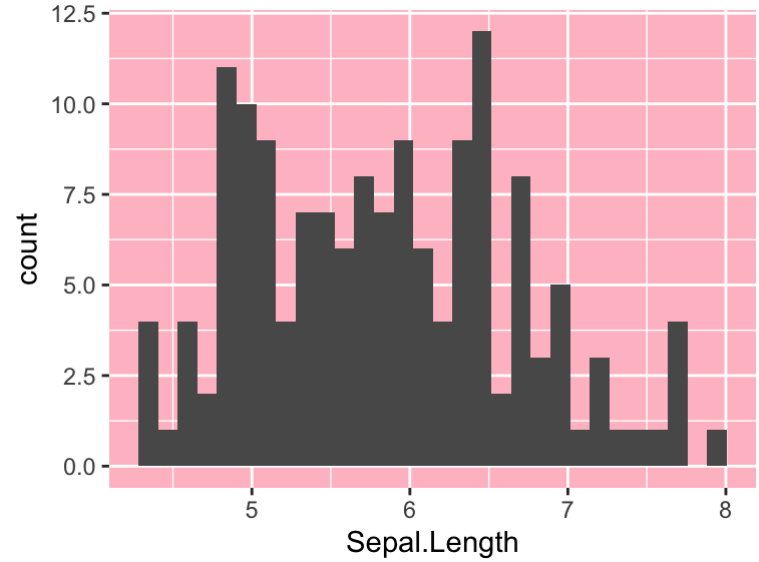
Change the background fill of the plot:

```
ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram() +
  theme(plot.background = element_rect(fill = "pink"))
```



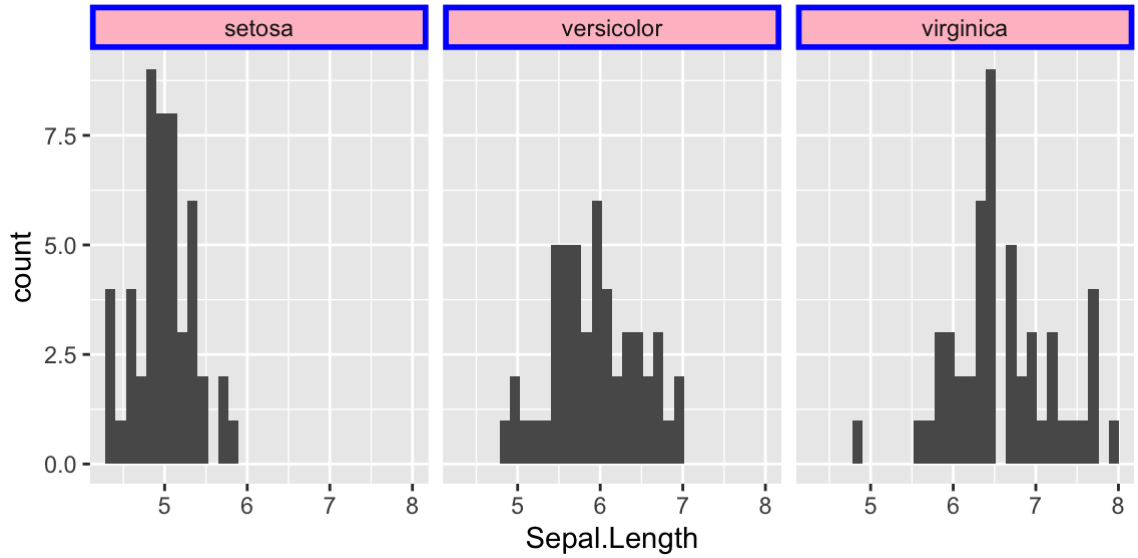
Change the background fill of the plot *PANEL* with `panel.backgroud`:

```
ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram() +
  theme(panel.background = element_rect(fill = "pink"))
```



Change the background fill and color (and use `size` to increase line thickness) of the *facet (panel) strips*:

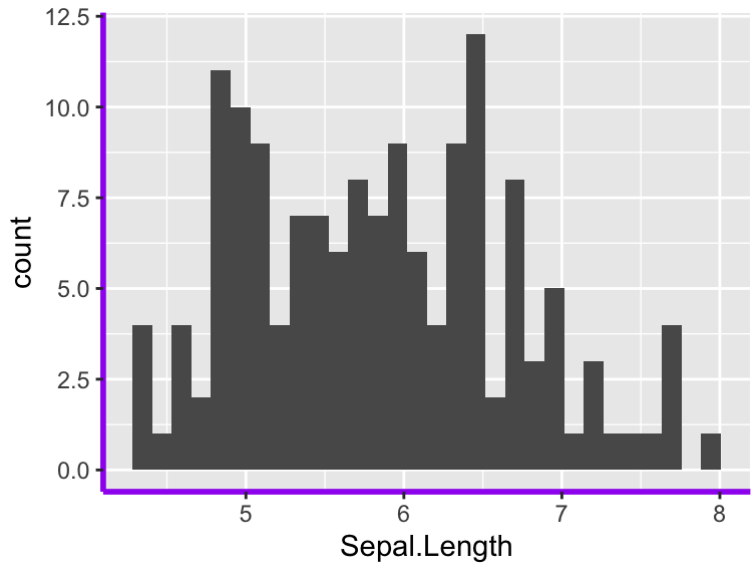
```
ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram() +
  # Add faceting
  facet_wrap(vars(Species), nrow = 1) +
  theme(strip.background = element_rect(fill = "pink",
                                        color = "blue",
                                        size = rel(4)))
```



Quick examples of `element_line()`

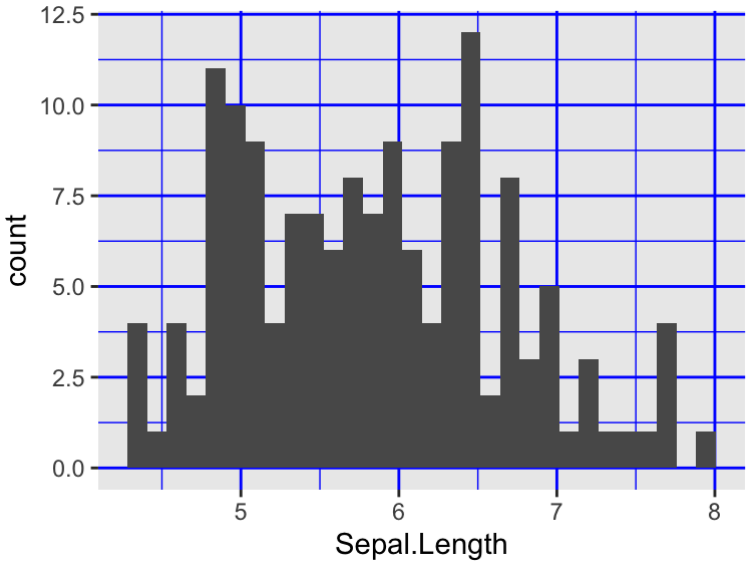
Change the axis thickness (multiply default size by 2) and color, why not!

```
ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram() +
  theme(axis.line = element_line(size = rel(2), color = "purple"))
```



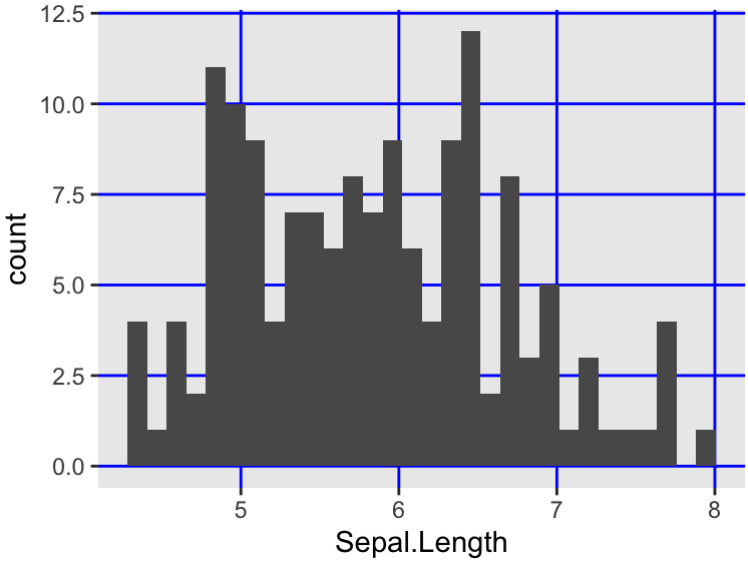
Change the background grid to blue!

```
ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram() +
  theme(panel.grid = element_line(color = "blue"))
```



Remove the *minor grid lines* (still blue to help you see the difference)

```
ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram() +
  theme(panel.grid = element_line(color = "blue"),
        panel.grid.minor = element_blank())
```

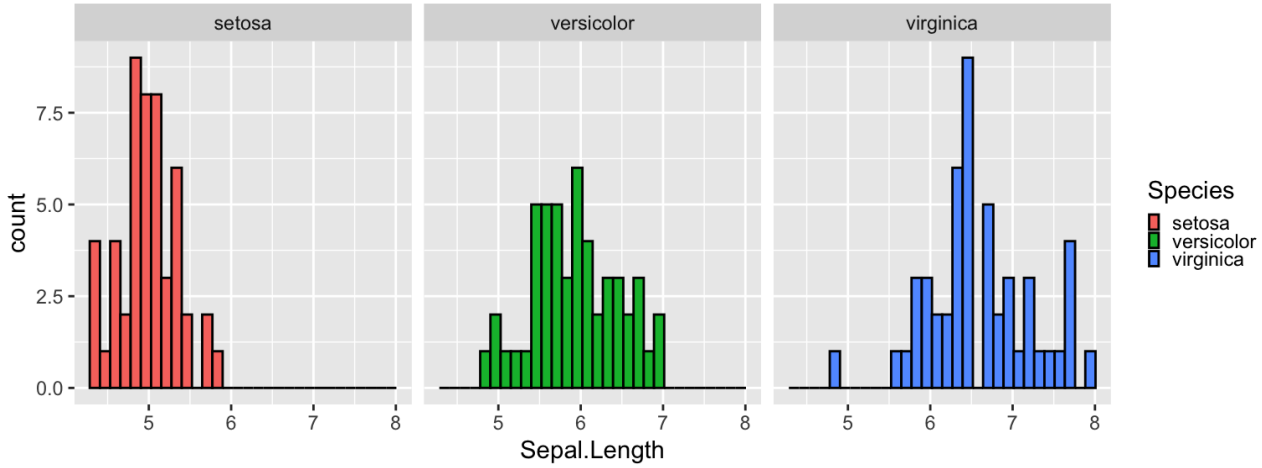


Quick examples of `unit()`

Legend components commonly use `unit()`, which takes two arguments: the target size, and the unit of the target size. There are many units for size, but some popular ones are `"cm"` and `"inches"`!

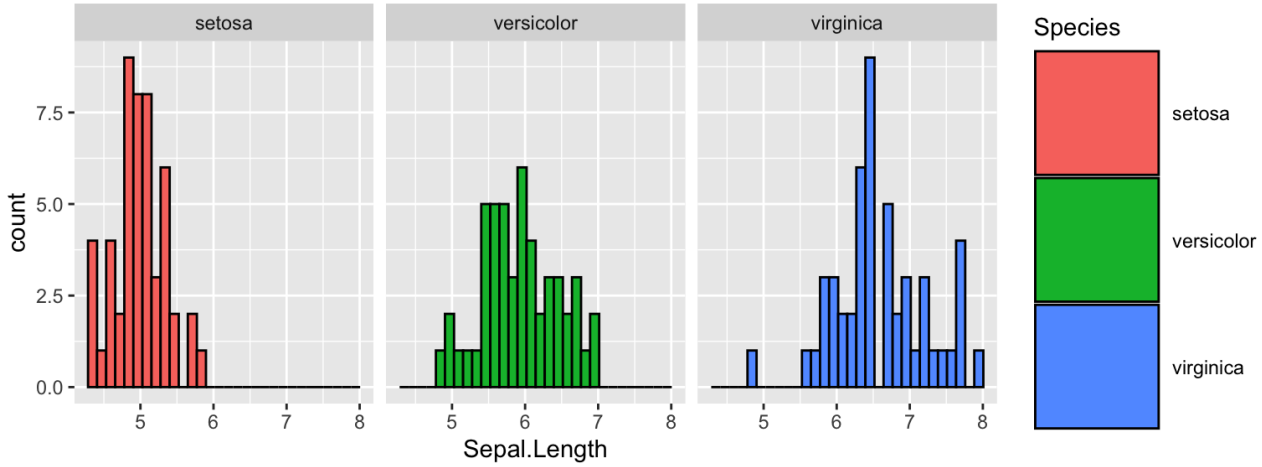
For example, we might want to change the size of the legend keys to make them smaller:

```
ggplot(iris, aes(x = Sepal.Length, fill = Species)) +
  geom_histogram(color = "black") +
  facet_wrap(vars(Species)) +
  theme(legend.key.size = unit(0.2, "cm"))
```



Or larger!

```
ggplot(iris, aes(x = Sepal.Length, fill = Species)) +
  geom_histogram(color = "black") +
  facet_wrap(vars(Species)) +
  theme(legend.key.size = unit(2, "cm"))
```

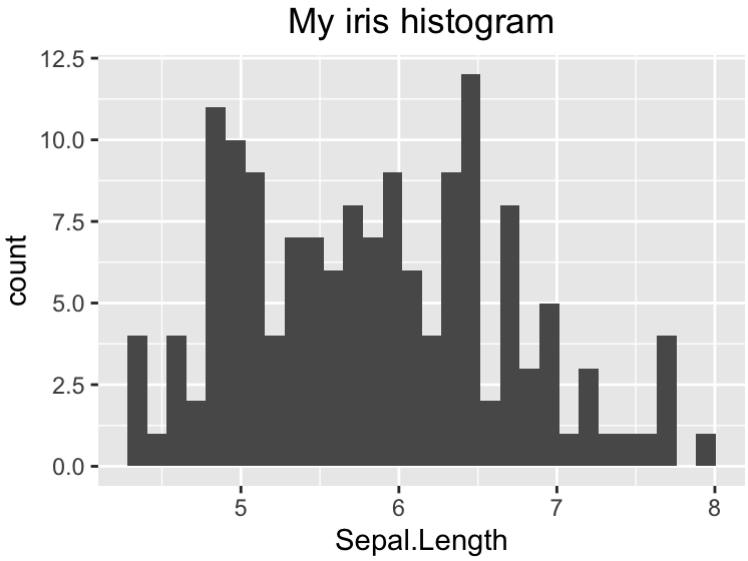


Modifying the theme of an existing plot

You can continue to add features, including theme changes, to a plot once it is created. For example:

```
ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram() +
  labs(title = "My iris histogram") -> iris_histogram # save plot to variable

# Add some theme changes to it, without changing the definition of `iris_histog
ram`
iris_histogram +
  theme(plot.title = element_text(hjust = 0.5)) # moves title to top-MIDDLE of
 the plot instead of the default top-left
```



Setting a theme for your entire session/script

Rather than adding a theme style onto each individual plot, you can use the function `theme_set()` set the theme to use for *all plots made in a given R session*. You can also include command in a script so that all plots made *on lines written AFTER* that command will automatically use that theme.

- To set a complete theme (for example, `theme_classic()` as the default, use:
  `theme_set(theme_classic())`
- To set a customized theme, use:
  `theme_set(theme_classic() + theme(STUFF THAT NORMALLY GOES INTO THEME))`

Want more?

Want more customization?! There are even more ways to tweak plots, specifically legends and keys, using `guides` . Learn more in the documentation (https://ggplot2.tidyverse.org/reference/index.html) (scroll to section “Guides: axes and legends”)!

1. As you will notice when perusing the documentation, there is also `axis.text.x` and `axis.text.y` which can be used to separately customize X and Y axis text.↩