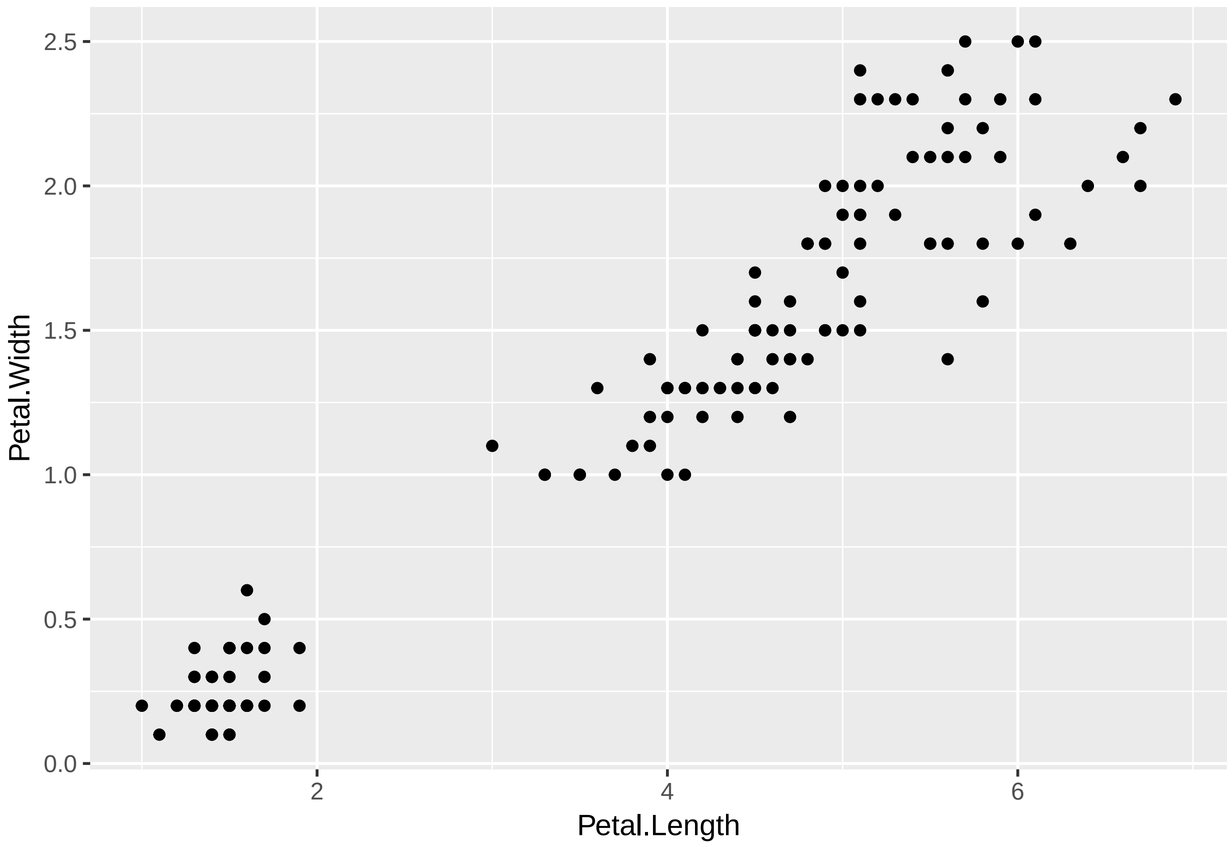
Plotting with `ggplot()`

The ultimate goal of any analysis is to *communicate* your results and understandings. Plotting your data is the most effect way to communicate a summary of it to an audience. Today will focus on the basics on how to do that in R, using the tidyverse package ggplot, with all examples with the built-in iris dataset.

Basic ggplot syntax

You have to specify a data table, at least one column from the table, and a geometry.

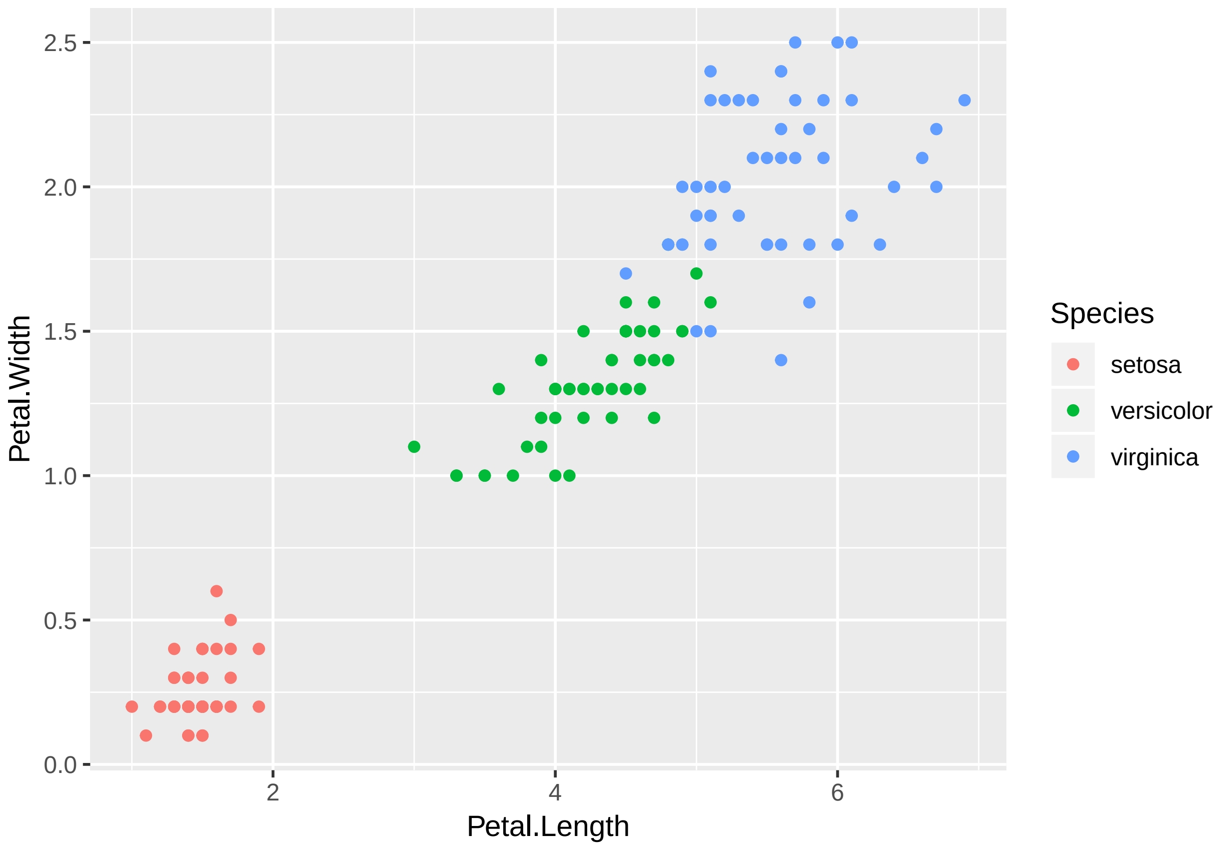
```r
ggplot(iris, aes(x = Petal.Length, y = Petal.Width)) + geom_point()
```



The scale of the axes are automatically determined from the data and they're labelled witht the column name.

Adding an aesthetic like color modifies all the points and automatically adds a legend.

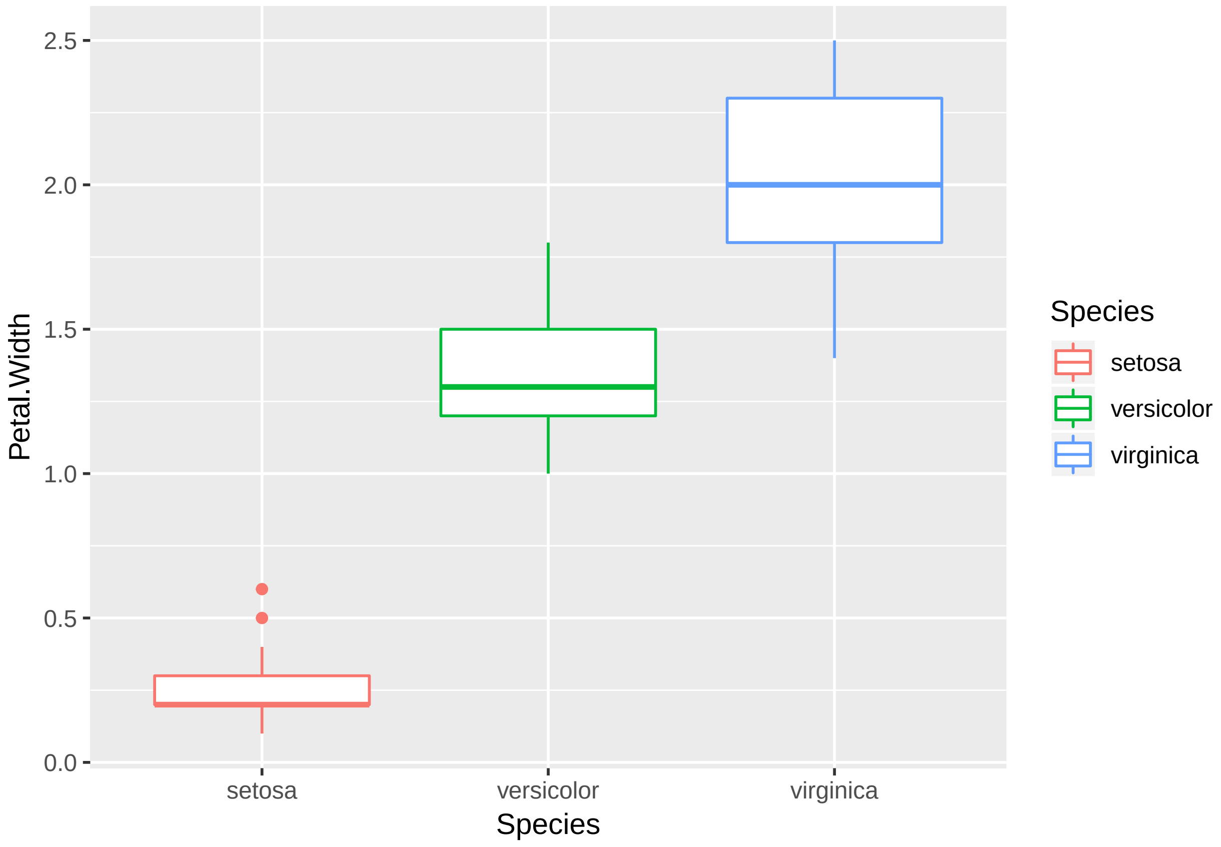
```r
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, color = Species)) + geom_point()
```



The legend title, like the axes labels, is the name of the column given to `color =` and the legend labels are whatever is in that column. For a continuous scale `ggplot()` will have a bar showing the range of colors and what values they correspond to.

You just have to change the geom to change the plot to another geometry/type

```r
ggplot(iris, aes(x = Species, y = Petal.Width, color = Species)) + geom_boxplot()
```



Geometries (aka Types of Plots)

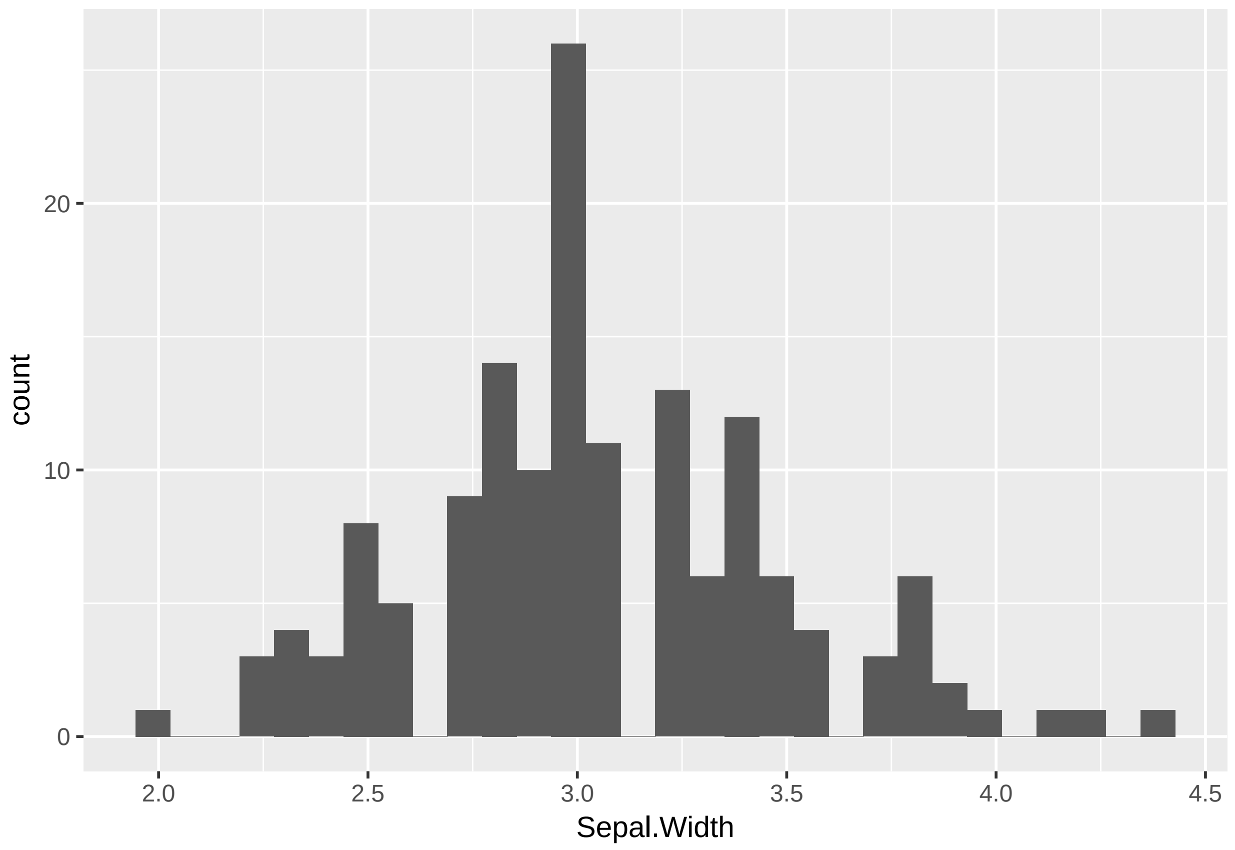
One Variable

Continuous

Histogram

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

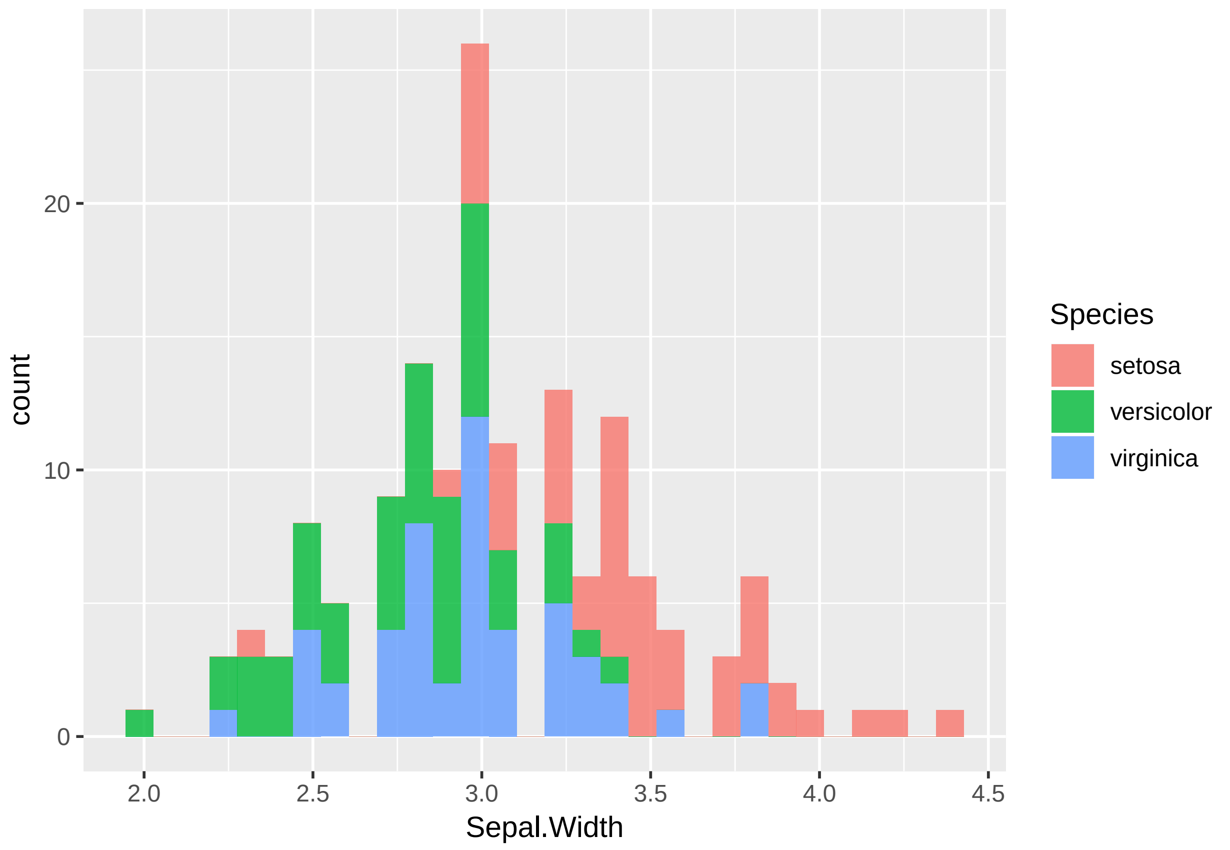
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Be careful with histograms! By default, when you add fill, color, etc, `geom_histogram()` stacks the different bars on top of each other. For *overlapping* histograms, which are more effective visually and what people are used to seeing, use `position = 'identity'` inside of `geom_histogram()`

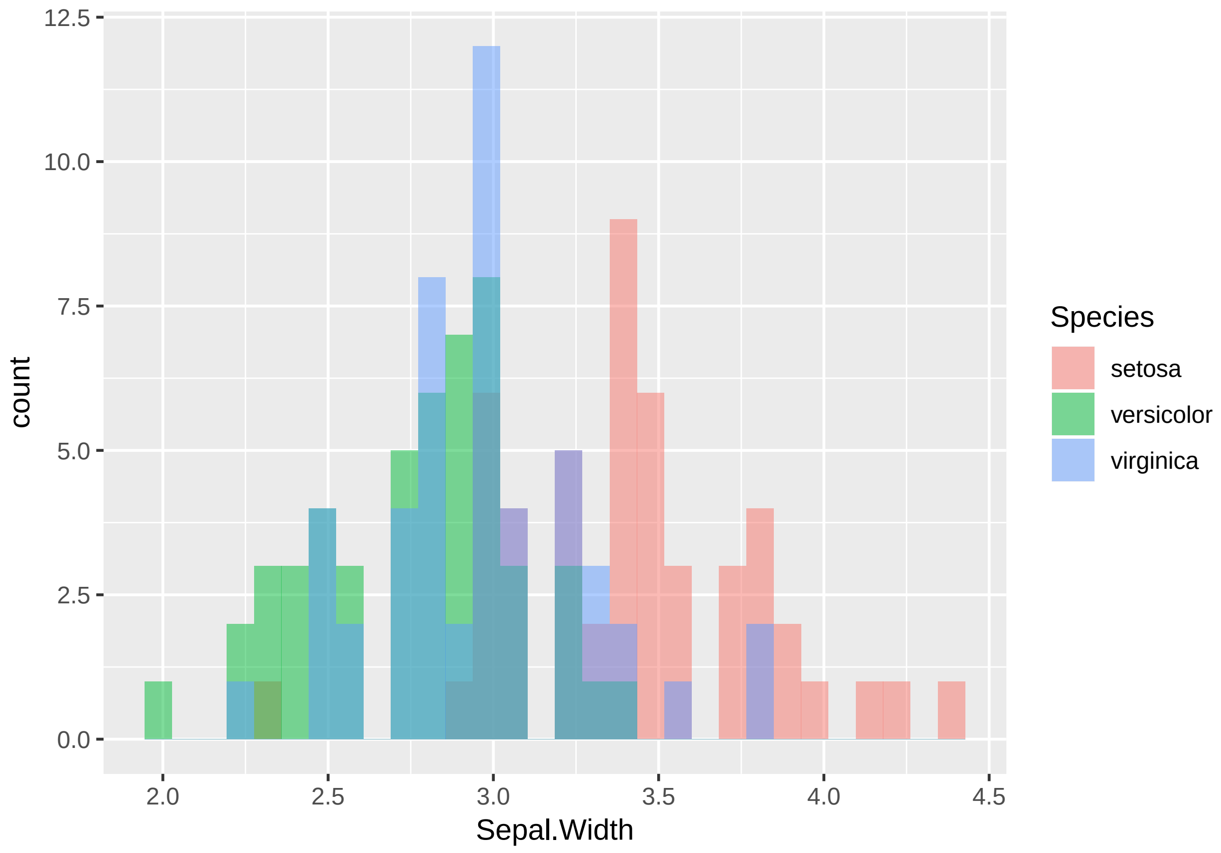
```
# default histogram
ggplot(iris, aes(x = Sepal.Width, fill = Species)) + geom_histogram(alpha = 0.8)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



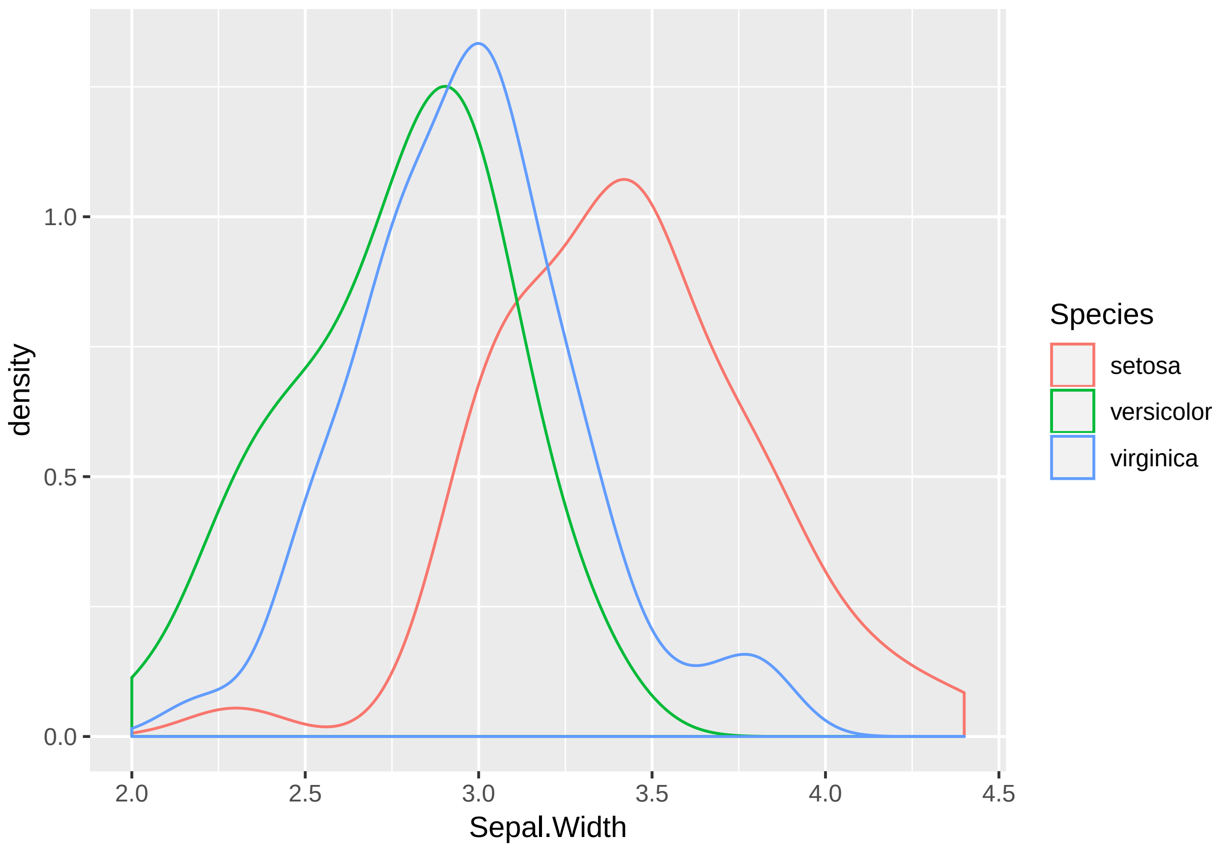
```
# use position = 'identity' for overlapping histograms
ggplot(iris, aes(x = Sepal.Width, fill = Species)) + geom_histogram(position = 'identity', alpha = 0.5)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Density plot

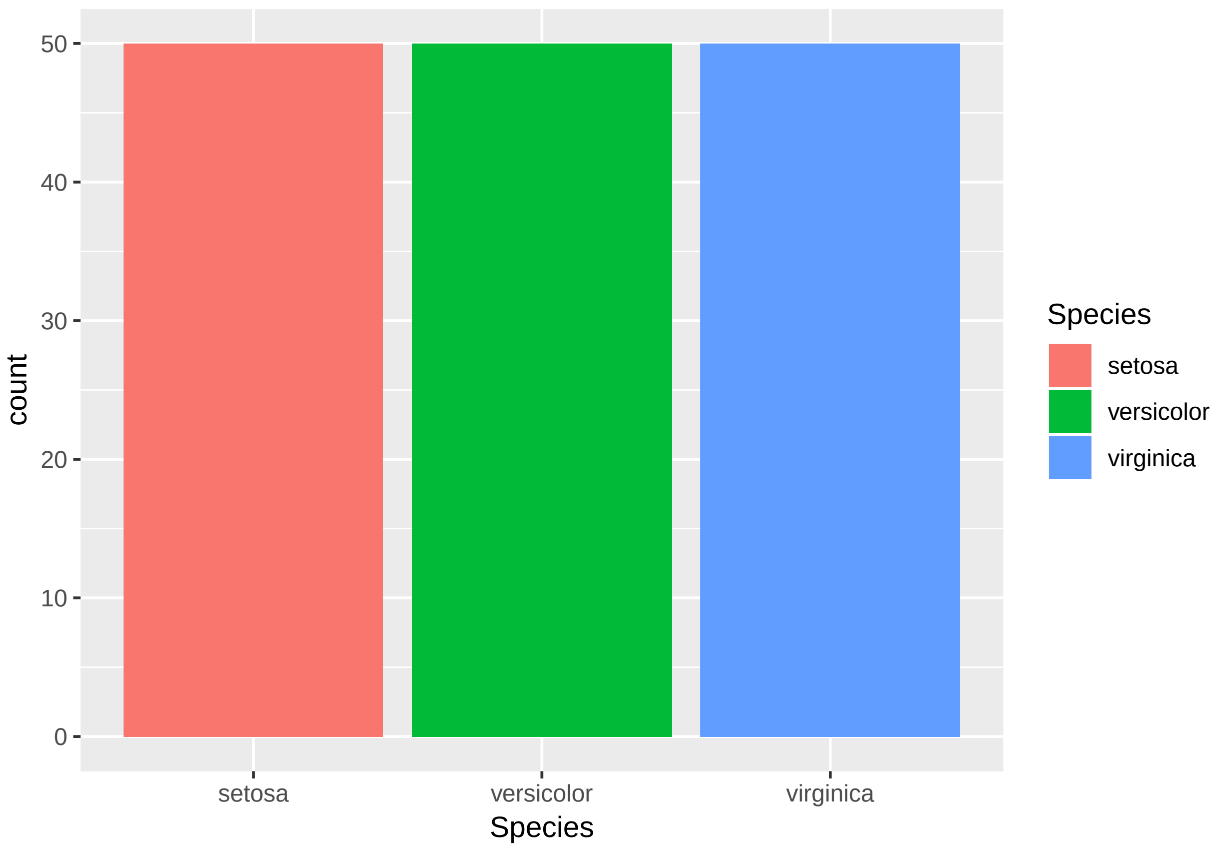
```
ggplot(iris, aes(x = Sepal.Width, color = Species)) + geom_density()
```



Discrete

Bar plot

```
ggplot(iris, aes(x = Species, fill = Species)) + geom_bar()
```

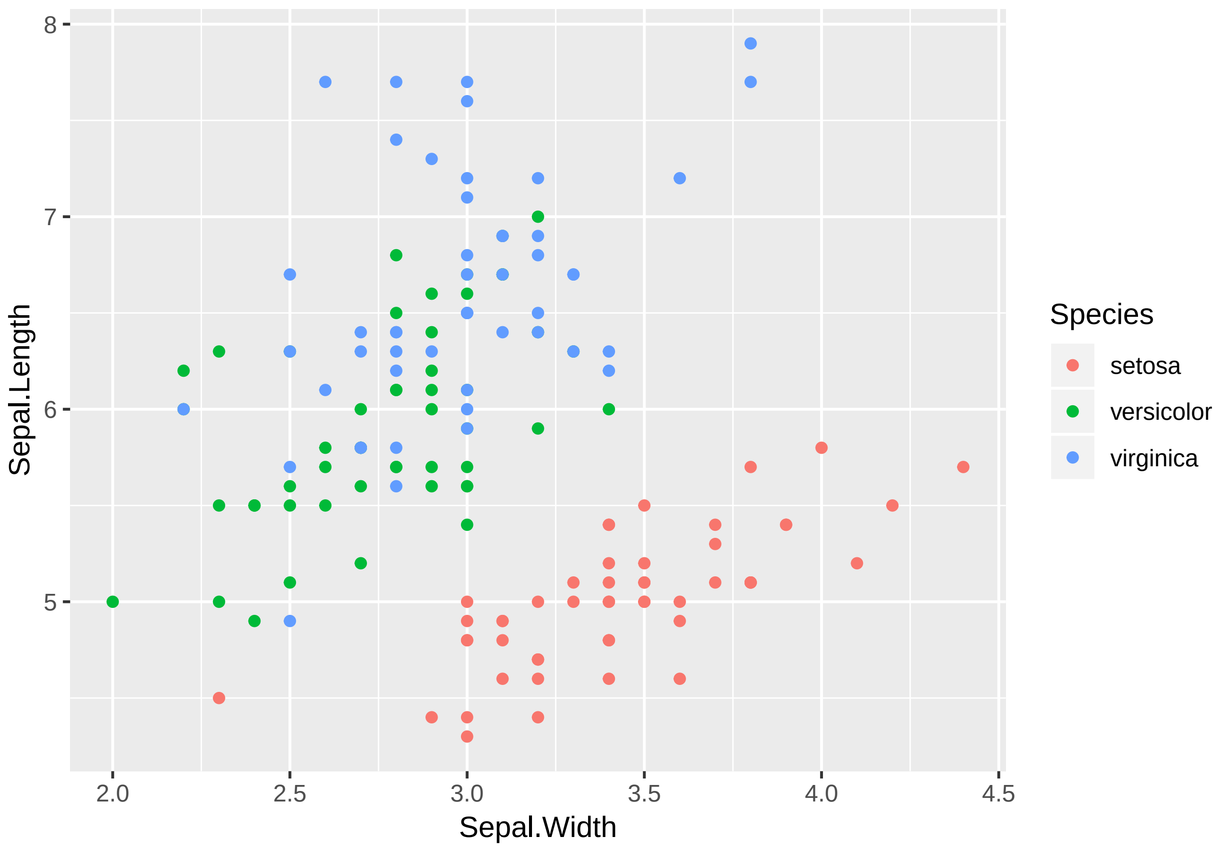


Two Variables

Continuous X, Continuous Y

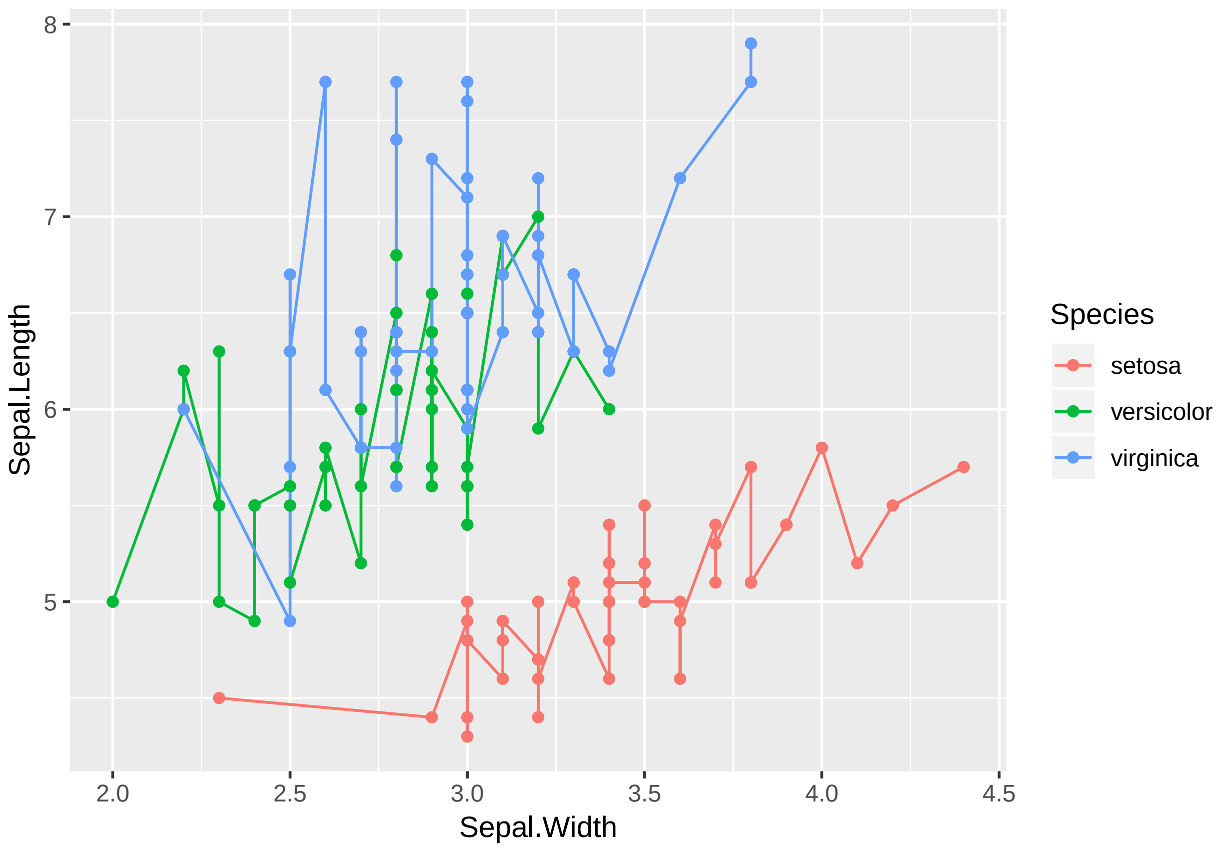
Scatter plot

```
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) + geom_point()
```



Line plot

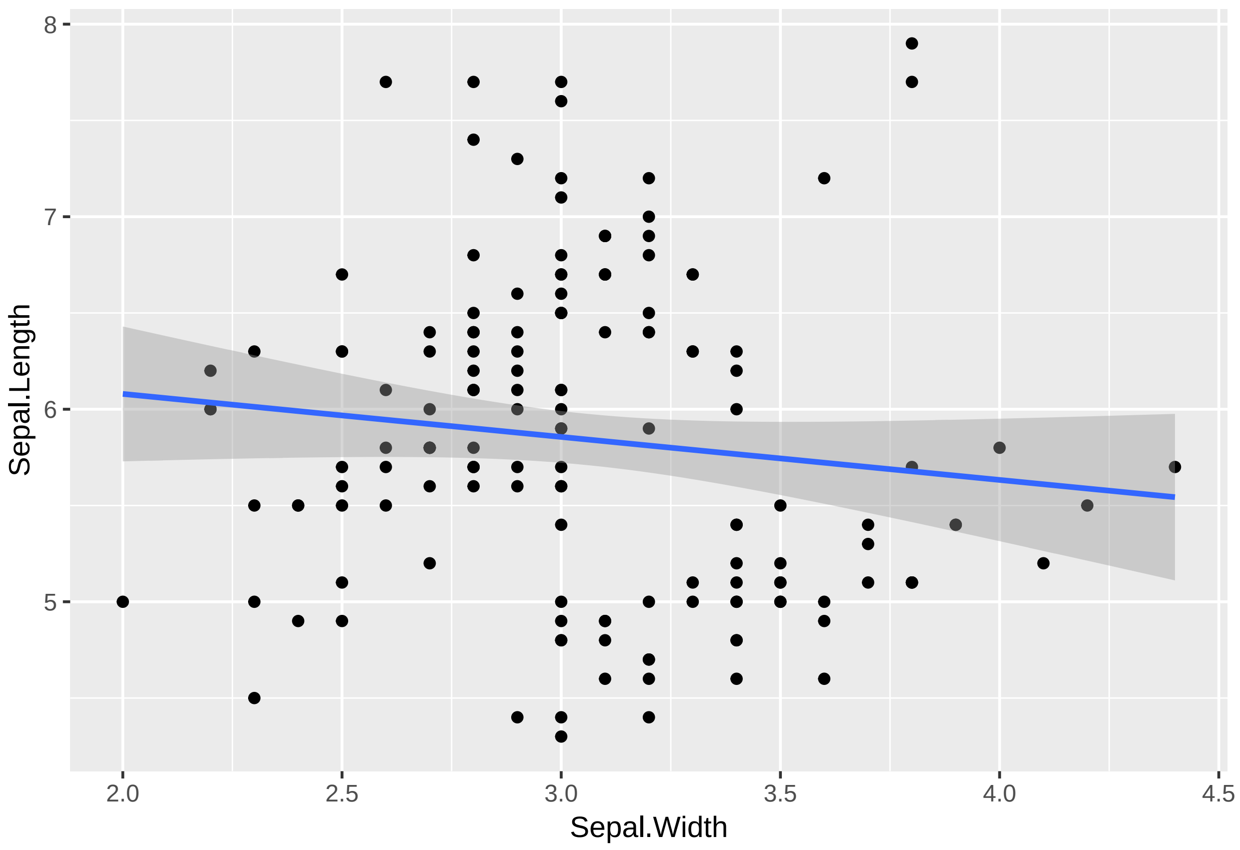
```
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) + geom_line() + geom_point()
```



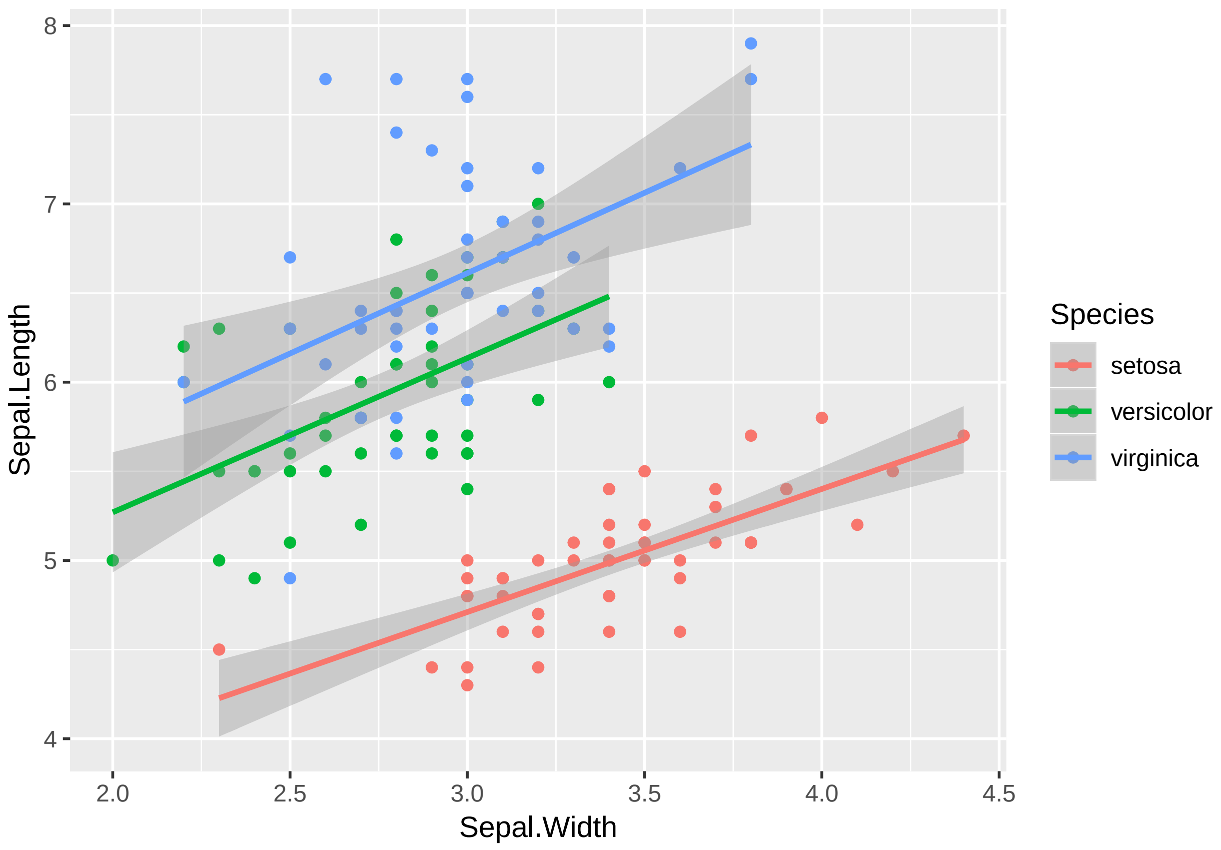
Adding a line of best fit

If you want to fit a straight line (or other type of fit) to your scatter plot, add on `geom_smooth()`. Use `method = lm` inside it to add a line (see the `geom_smooth()` documentation for other fit methods). Also, it automatically supplies confidence intervals with whatever is fit.

```r
# with line of best fit
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length)) + geom_point() + geom_smooth(method = lm)
```



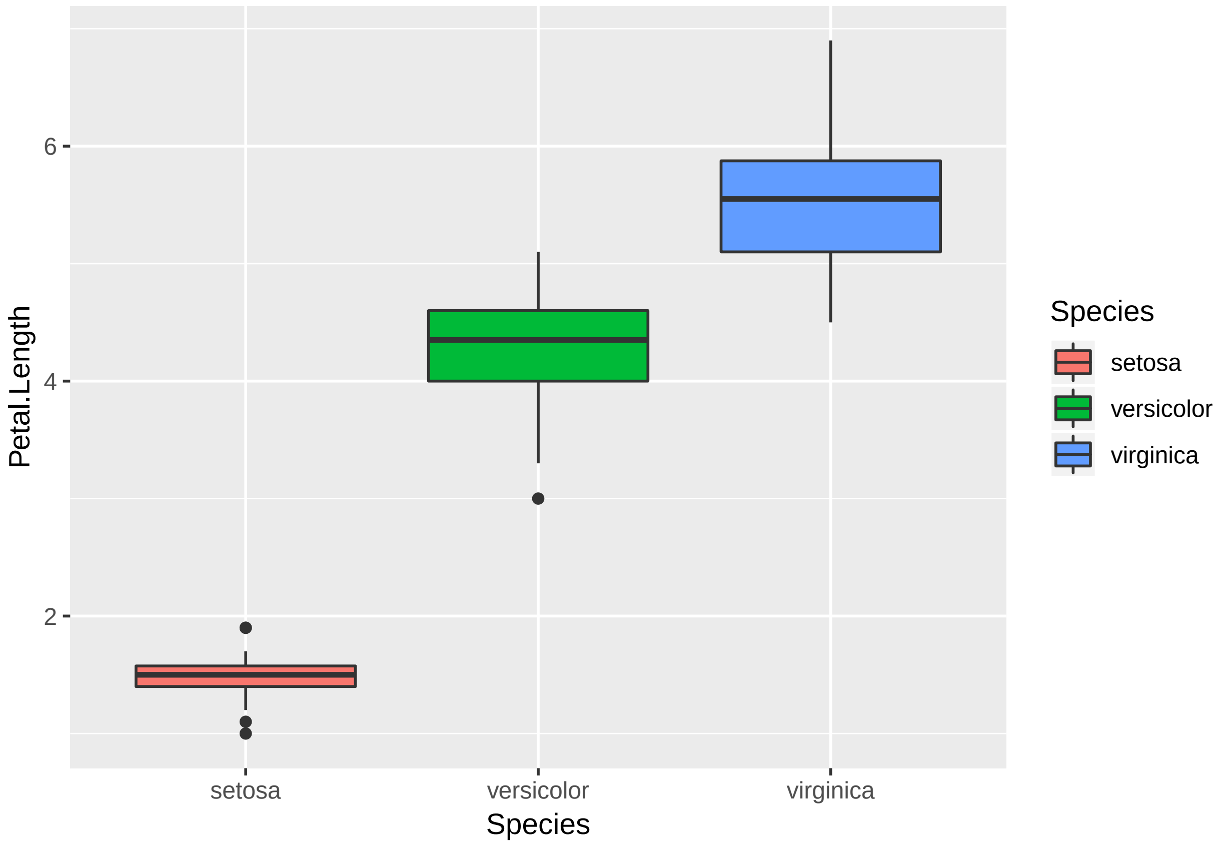
```
# if you color by something, each subgroup will automatically get it's own line
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) + geom_point() + geom_smooth(meth
```



Discrete X, Continuous Y

Boxplot

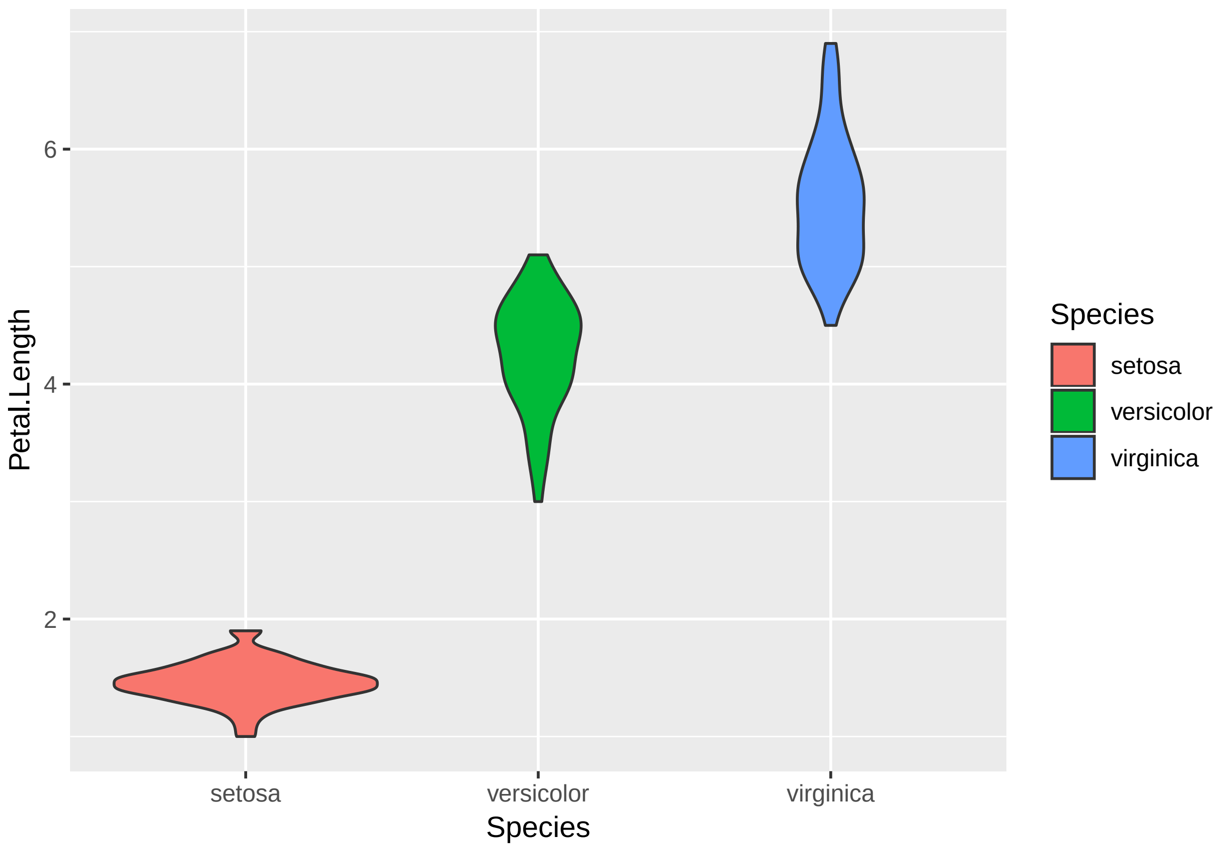
```
ggplot(iris, aes(x = Species, y = Petal.Length, fill = Species)) + geom_boxplot()
```



Violin plot

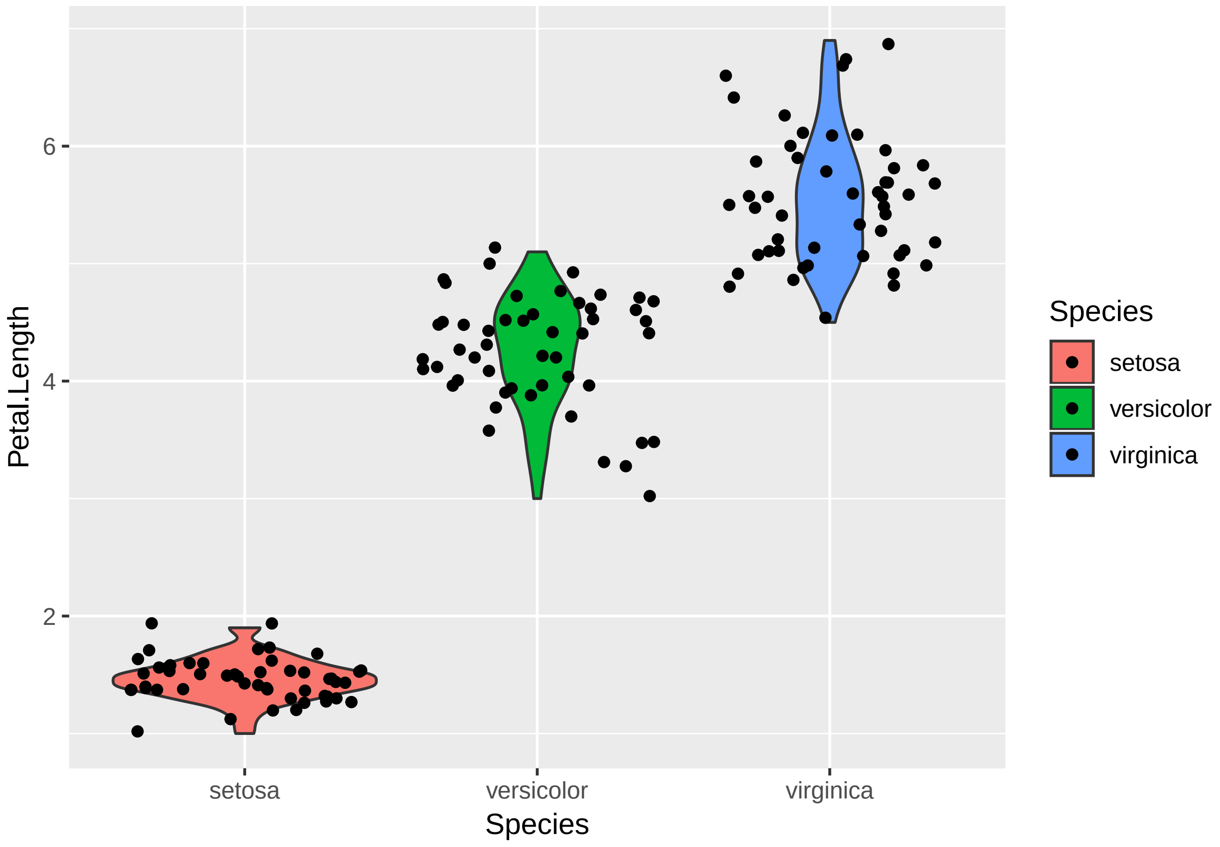
A violin plot is a mirrored density plot displayed like a boxplot. It gives you a better sense of the distribution of the underlying data than the five number summary of the boxplot.

```
ggplot(iris, aes(x = Species, y = Petal.Length, fill = Species)) + geom_violin()
```



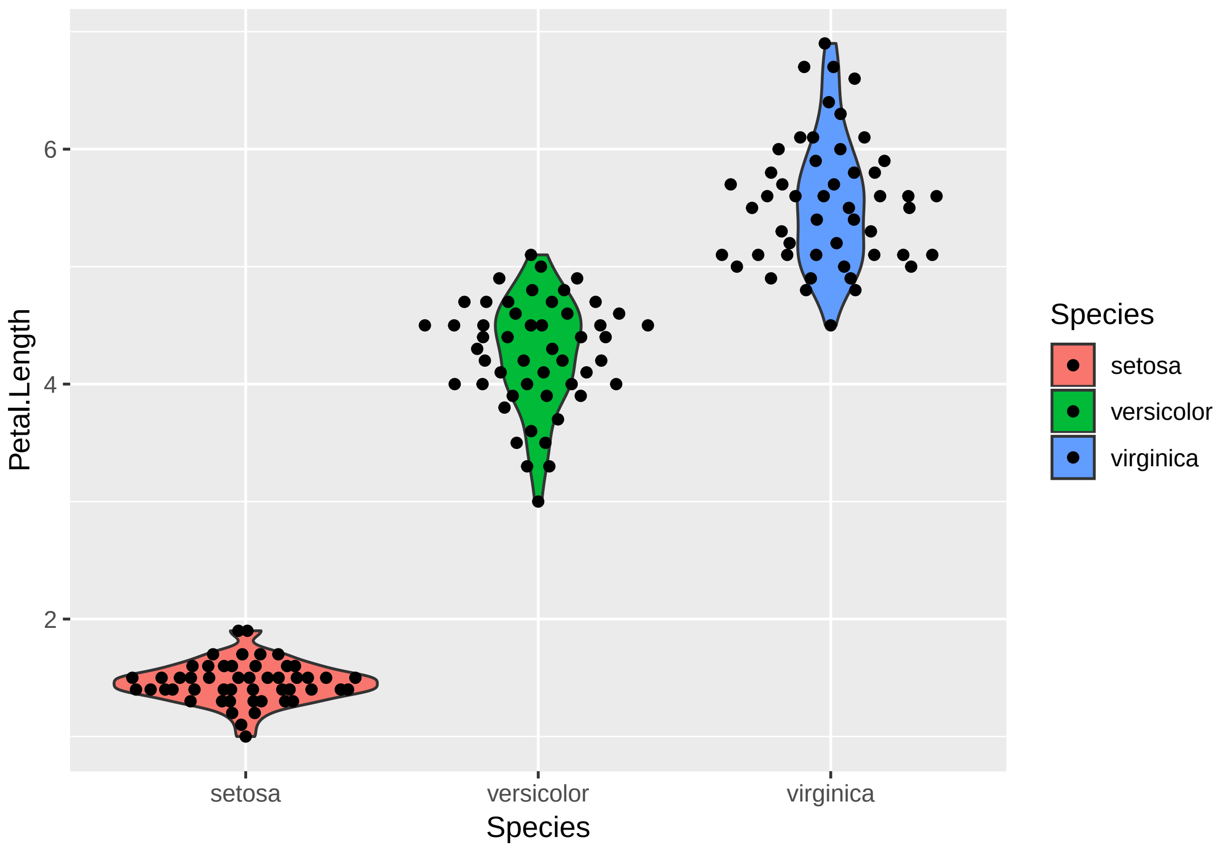
If you want to overlay a bar/boxplot/violin plot with the actual data points, you can add `geom_jitter()` on

```
ggplot(iris, aes(x = Species, y = Petal.Length, fill = Species)) + geom_violin() + geom_jitter()
```

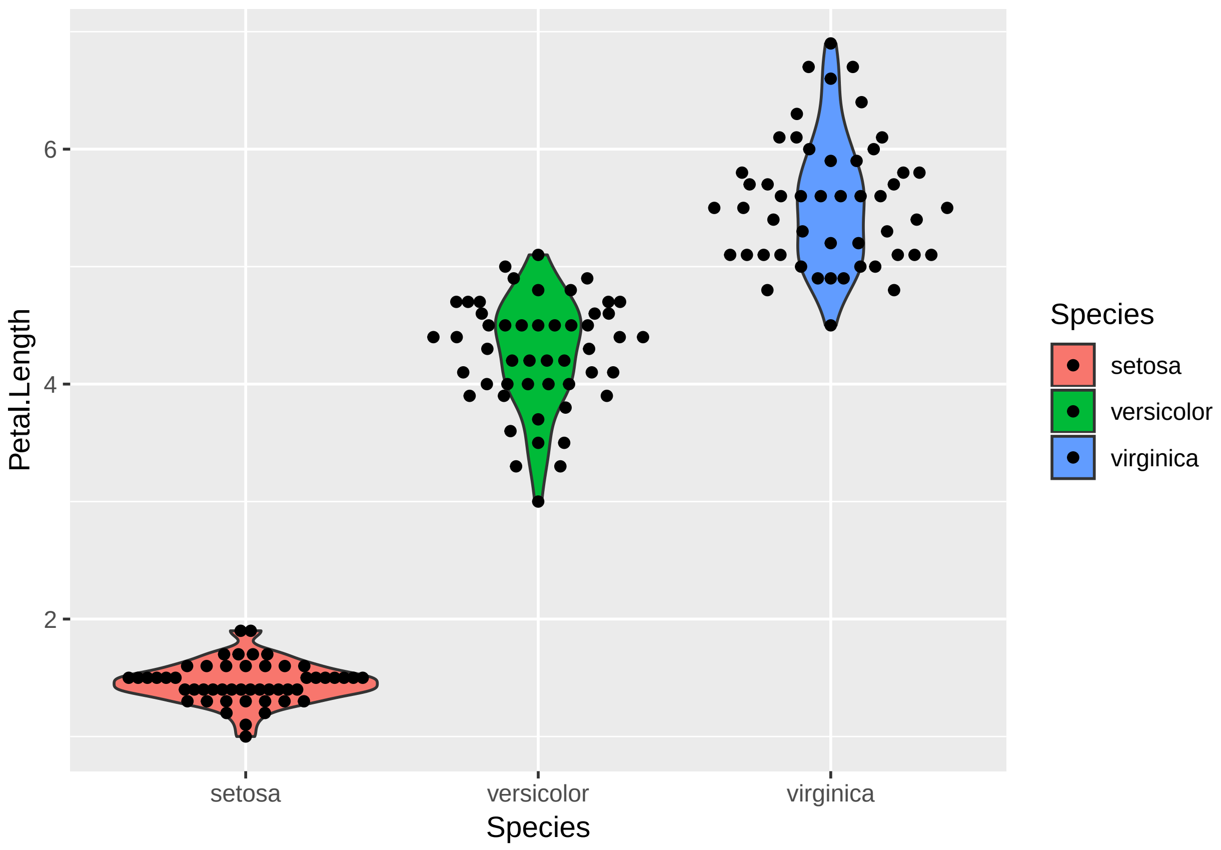


Even better than geom_jitter, there's the ggbeeswarm package, which has different geoms for overlaying jittered points according to the underlying distribution. See the GitHub README for more documentation/examples https://github.com/eclarke/ggbeeswarm

```
library(ggbeeswarm)

# example 1
ggplot(iris, aes(x = Species, y = Petal.Length, fill = Species)) + geom_violin() + geom_quasirandom()
```



```
# example 2 with modified method
ggplot(iris, aes(x = Species, y = Petal.Length, fill = Species)) + geom_violin() + geom_quasirandom(met
```

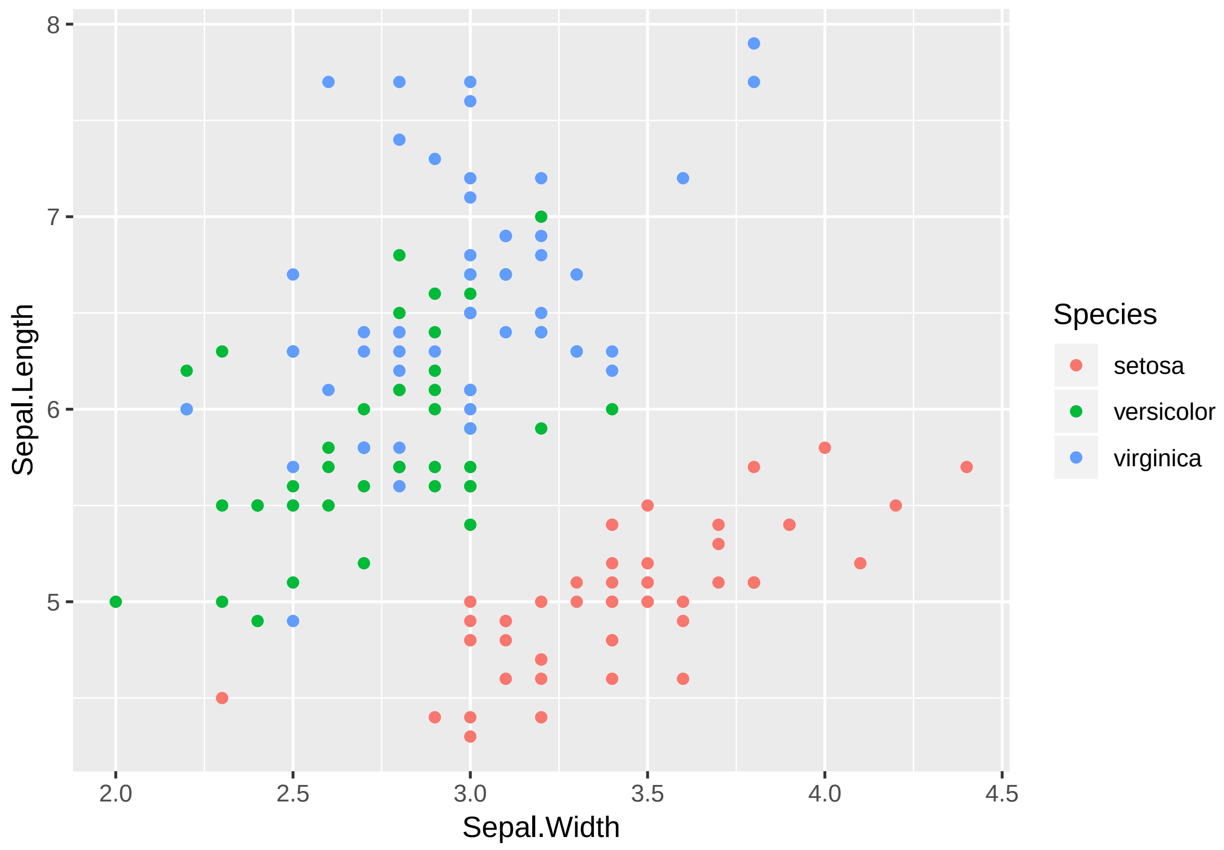


Aesthetics

Basic three

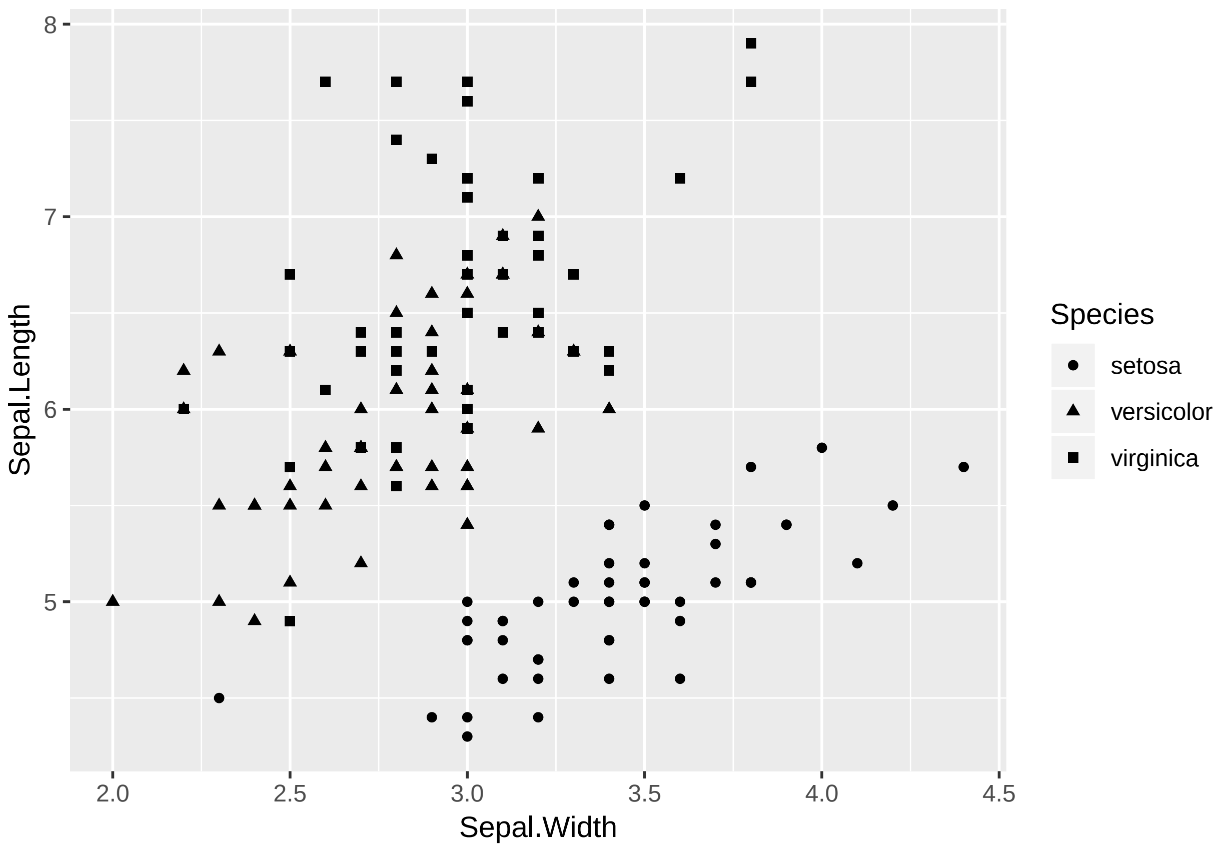
Color

```r
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) + geom_point()
```



Shape

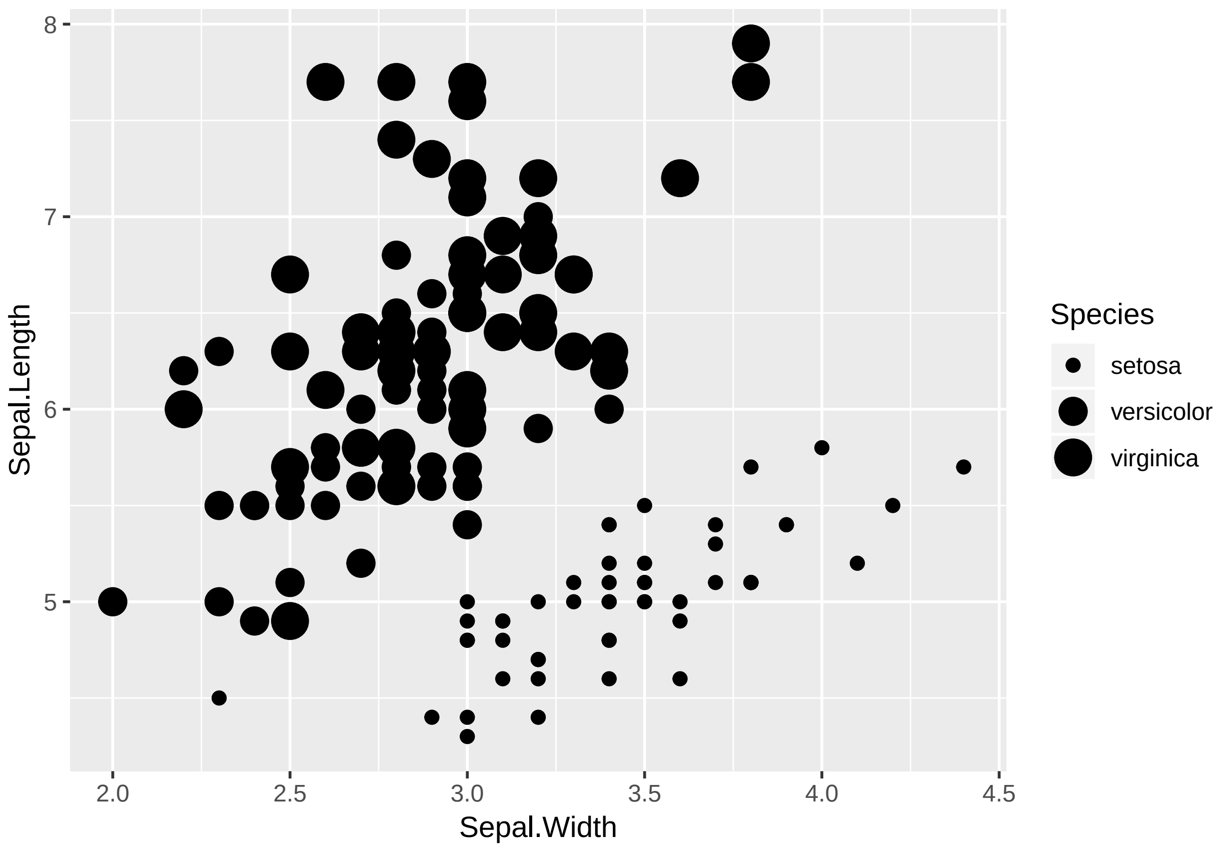
```r
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, shape = Species)) + geom_point()
```



Size

```
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, size = Species)) + geom_point()
```

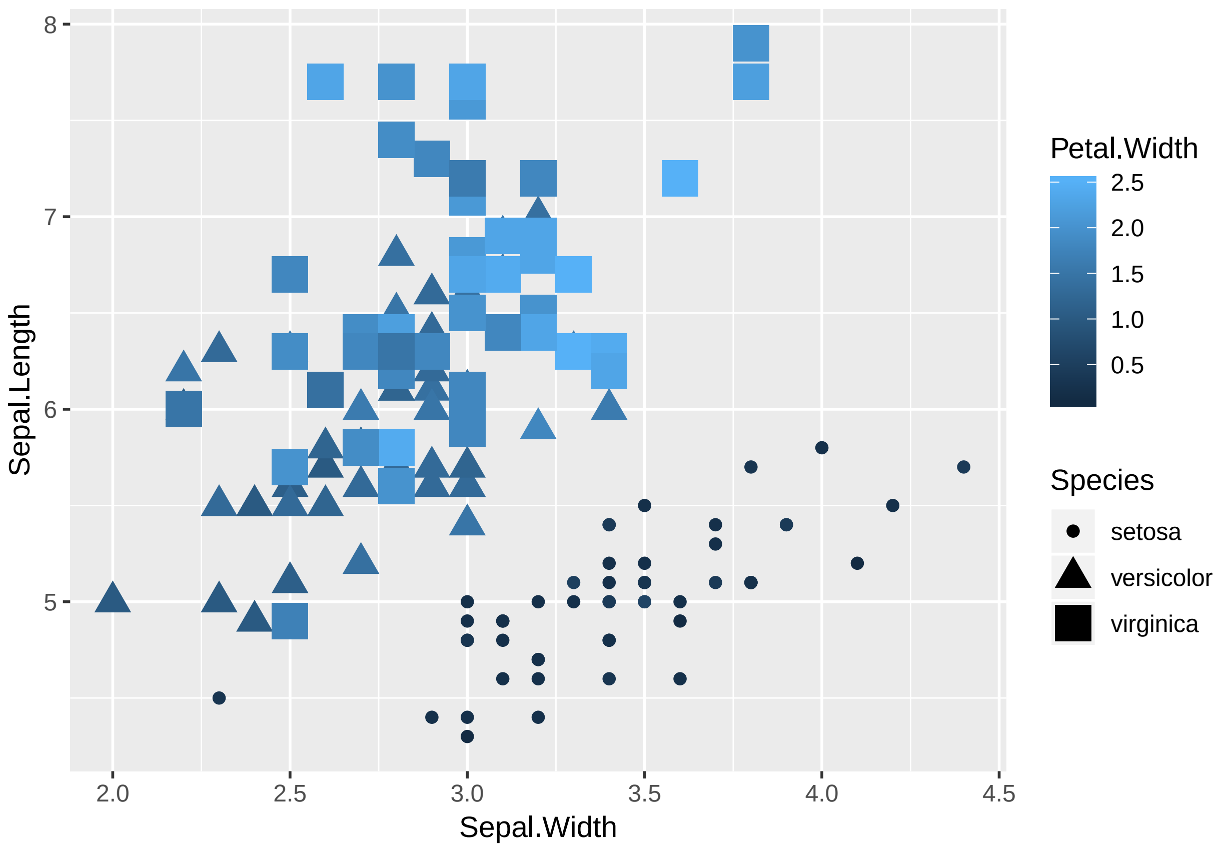
```
## Warning: Using size for a discrete variable is not advised.
```



Combine as many as you want

```
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Petal.Width, shape = Species, size = Specie
```

```
## Warning: Using size for a discrete variable is not advised.
```

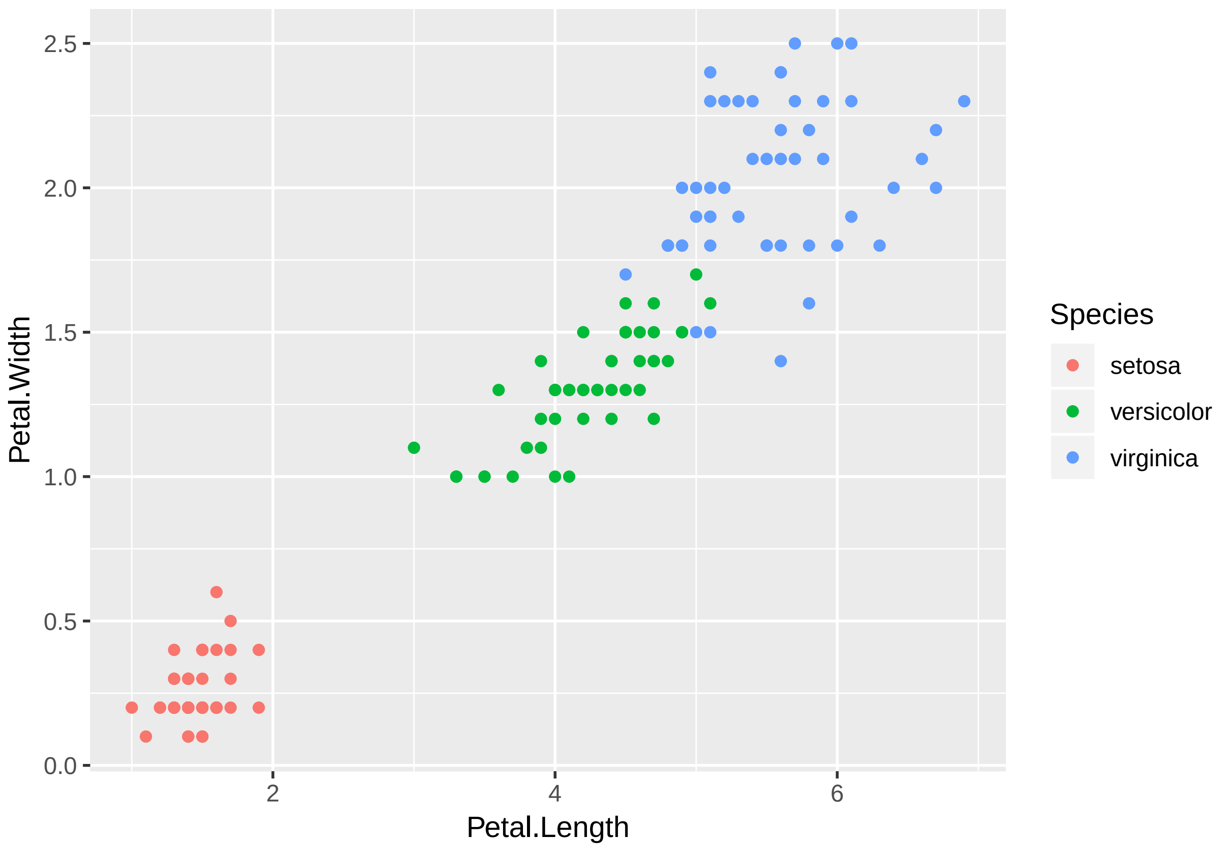


Miscellaneous Useful Features

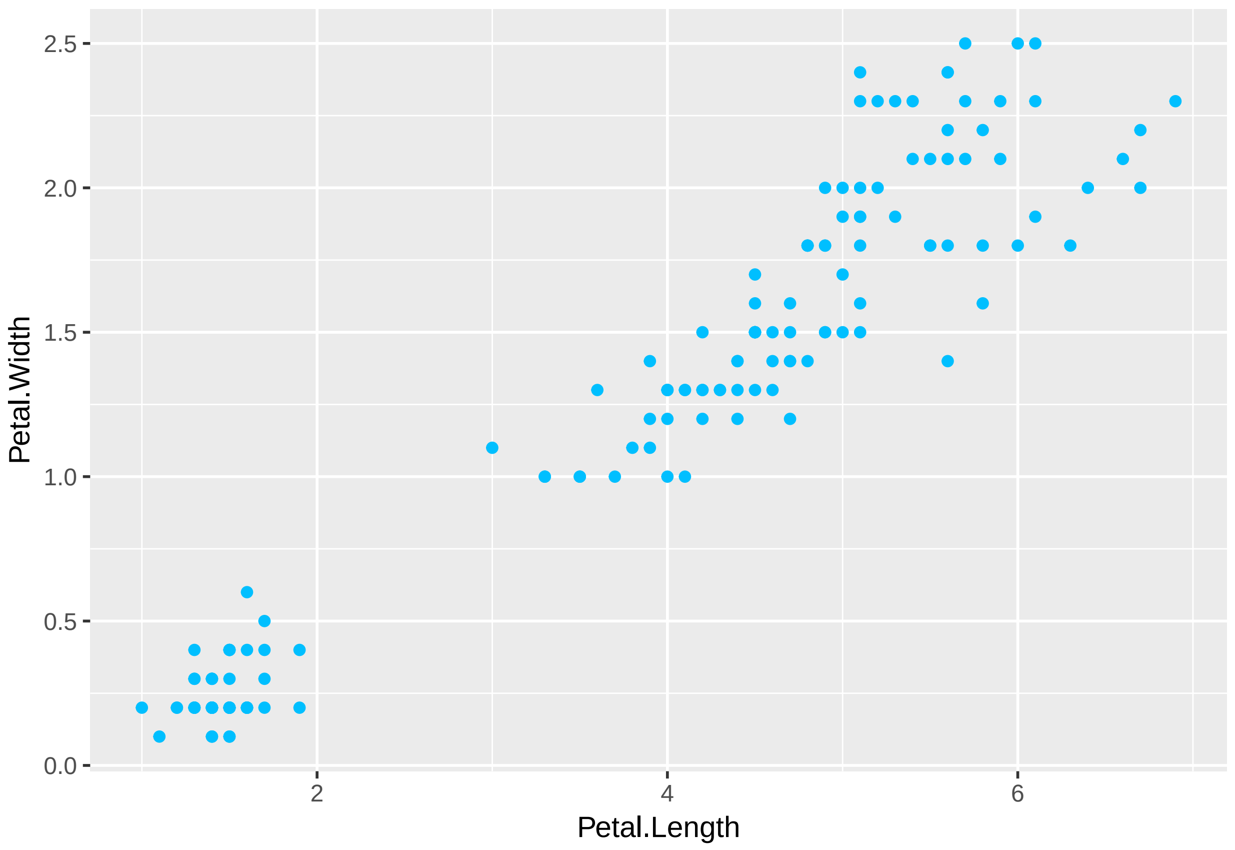
More about `aes()`

Whether you add color (or shape, size, etc.) inside or outside of 'aes()' has a different outcome. Inside `aes()` ggplot modifies points differently, but outside `aes()` it applies the same thing to all points.

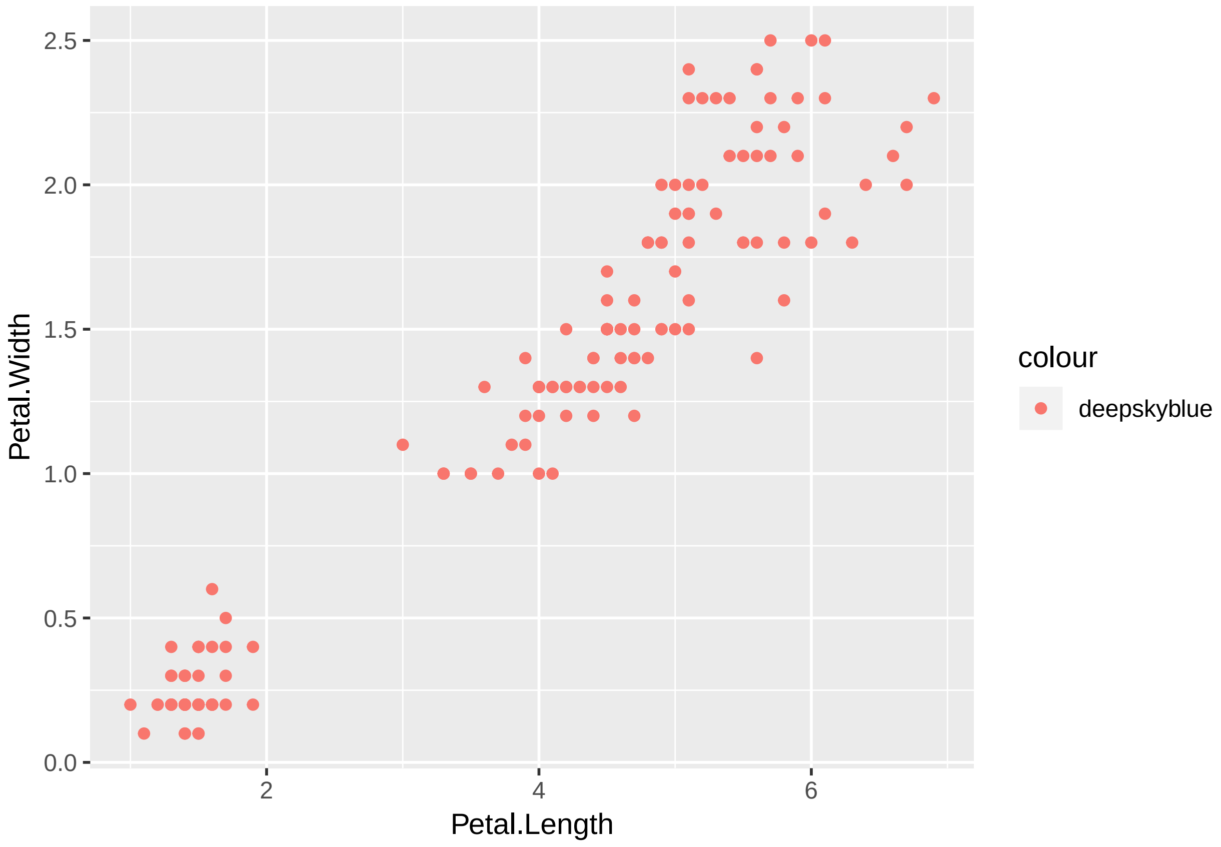
```r
# where we've had it
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, color = Species)) + geom_point()
```



```r
# I just want my points to be another color
ggplot(iris, aes(x = Petal.Length, y = Petal.Width)) + geom_point(color = 'deepskyblue')
```



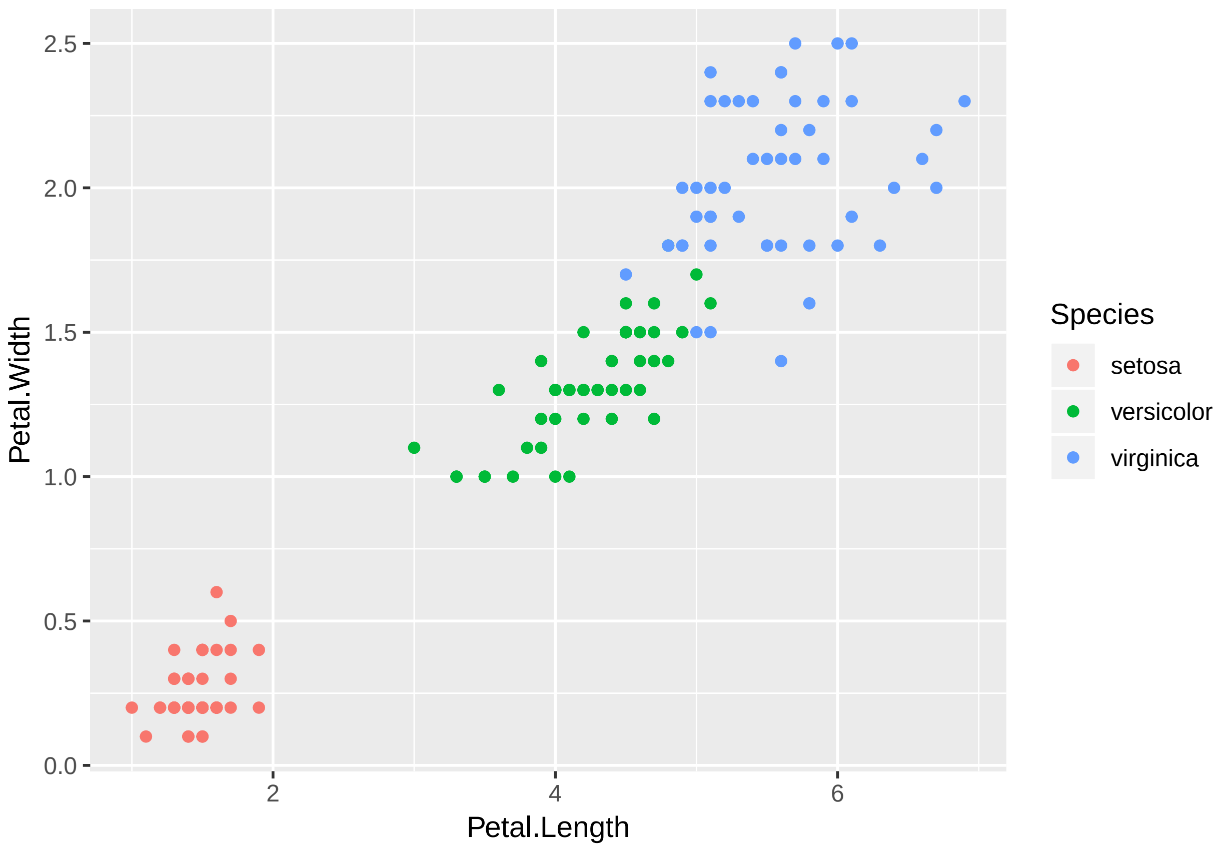
```
# BUT if I put color = 'deepskyblue' instead an aes() it won't work; ggplot thinks it's a data feature
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, color = 'deepskyblue')) + geom_point()
```



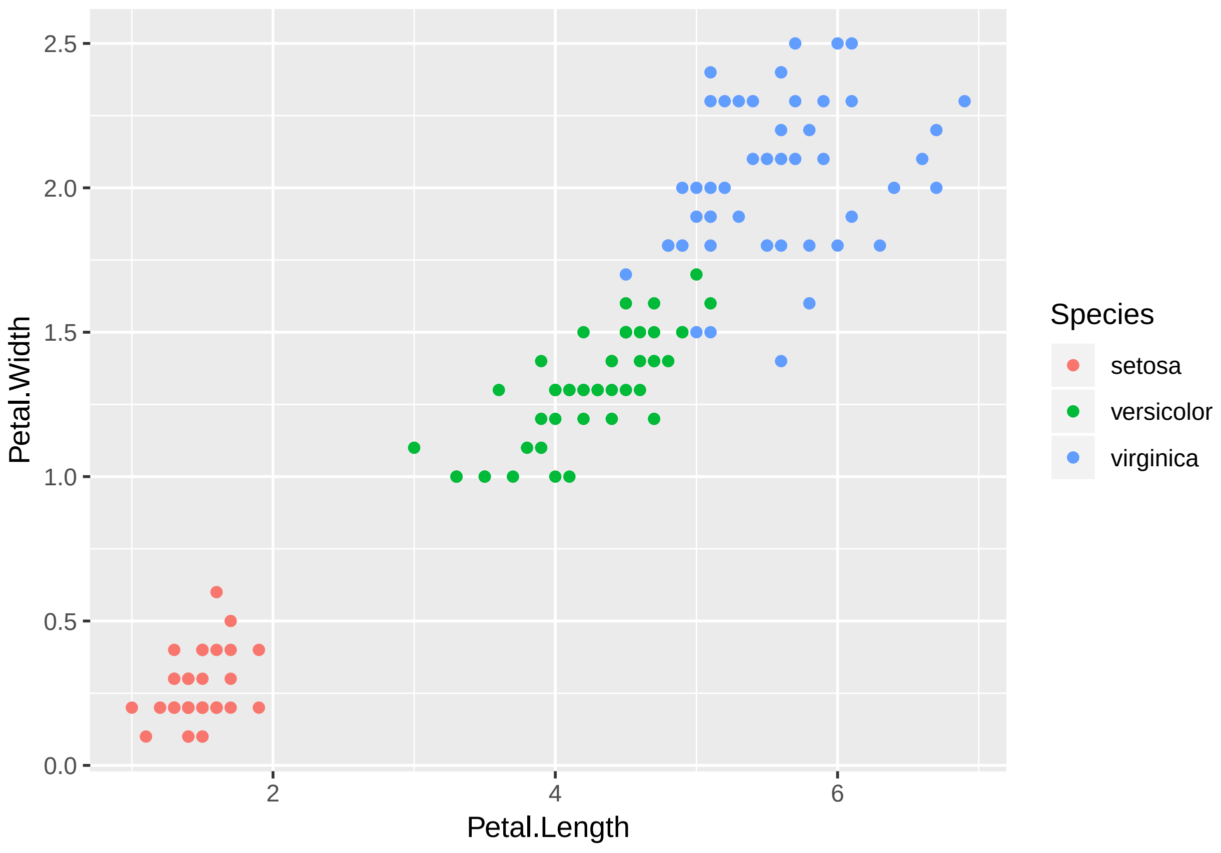
```
# ALSO if I DON'T put color = Species inside aes() it throws an error
# uncomment the line below and try
#ggplot(iris, aes(x = Petal.Length, y = Petal.Width)) + geom_point(color = 'Species')
```

You can also put `aes()` inside either `ggplot()` or whatever geom you pick

```
# inside ggplot()
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, color = Species)) + geom_point()
```



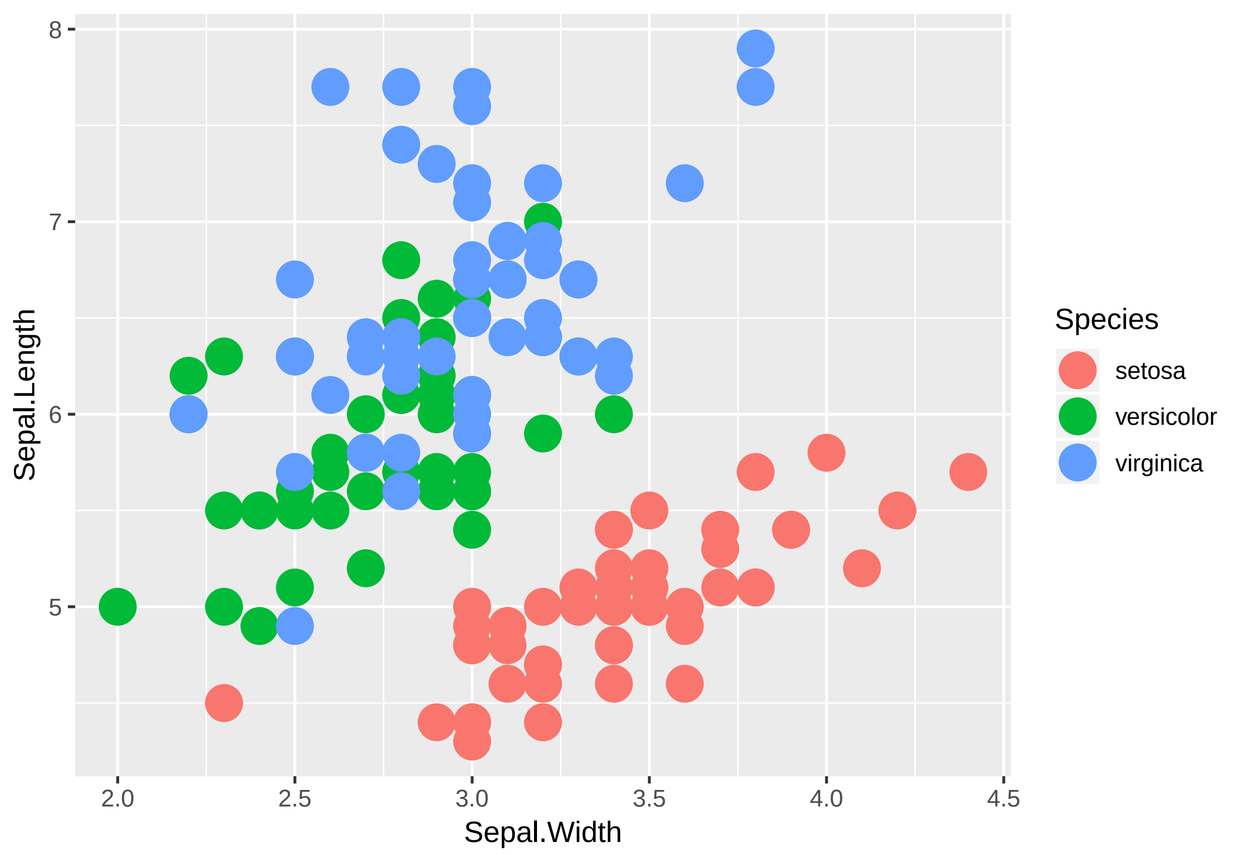
```r
# inside geom_point()
ggplot(iris) + geom_point(aes(x = Petal.Length, y = Petal.Width, color = Species))
```



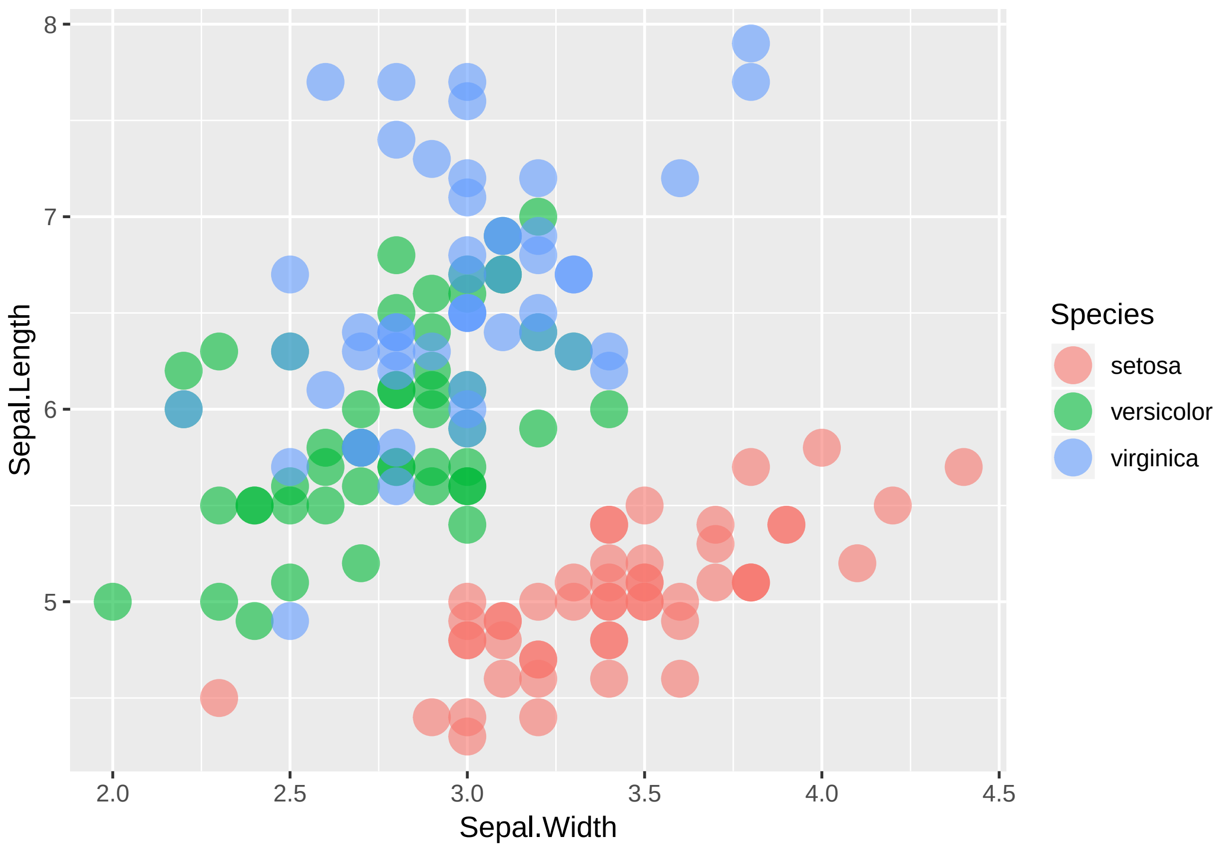
Alpha

The `alpha =` parameter sets the transparency of the plot. Alpha ranges from 0 to 1, where 0 is completely transparent and 1 is completely solid. Making your plots partially transparent is helpful when you have overplotting or any overlapping. (I made the points bigger to make the difference in alpha easier to see)

```
# without transparency
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) + geom_point(size = 6)
```



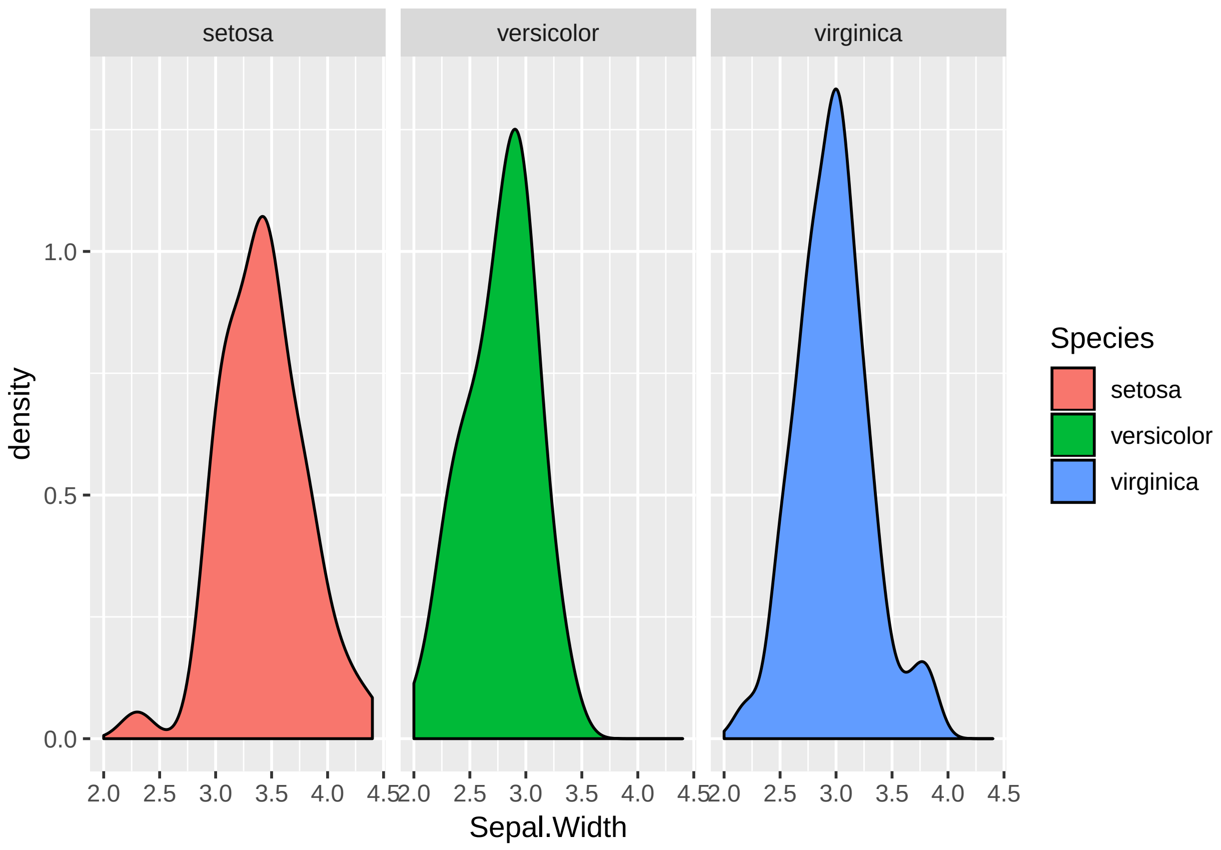
```
# with transparency
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) + geom_point(size = 6, alpha = 0.
```



facets

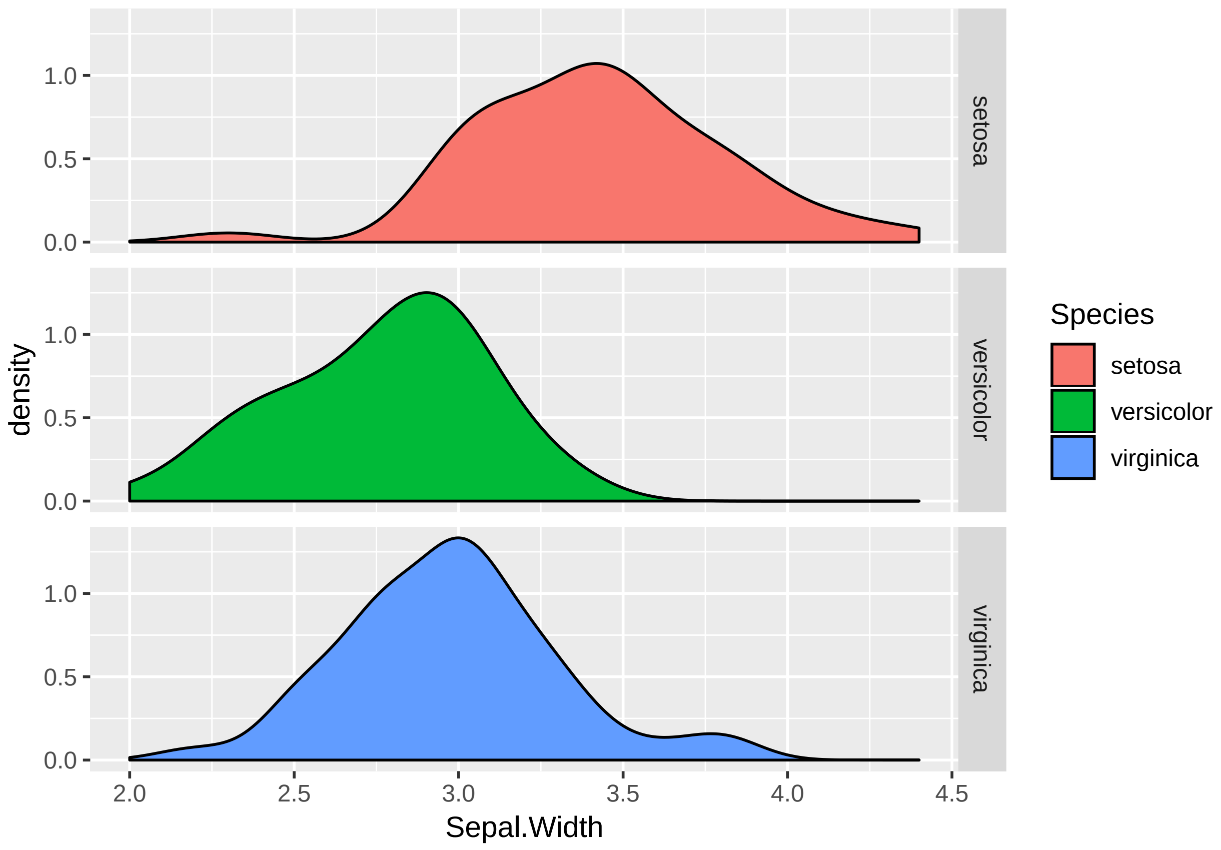
Sometimes you might want to make multiple plots based on an element in your data like: significant/not significant, sample, phenotype, etc. If it's a label in your table, you can add a facet to automatically split it.

```
ggplot(iris, aes(x = Sepal.Width, fill = Species)) + geom_density() + facet_grid(. ~ Species)
```



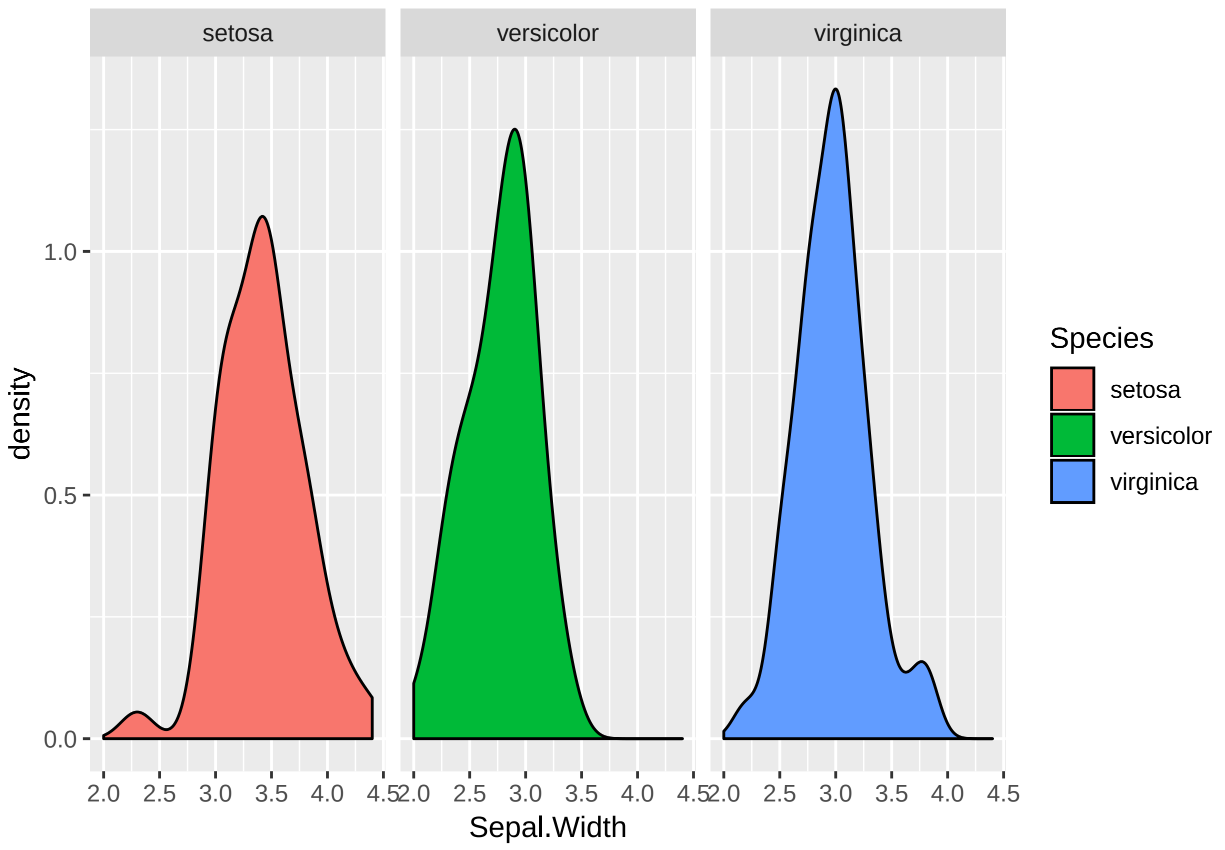
You can also make the facets line up vertically by switching the syntax in `facet_grid()`

```
ggplot(iris, aes(x = Sepal.Width, fill = Species)) + geom_density() + facet_grid(Species ~ .)
```



You can also use `facet_wrap()` instead, which will automatically wrap your facets (this isn't an issue with the iris examples here, but as you add 5, 10, 15 facets, I prefer this)

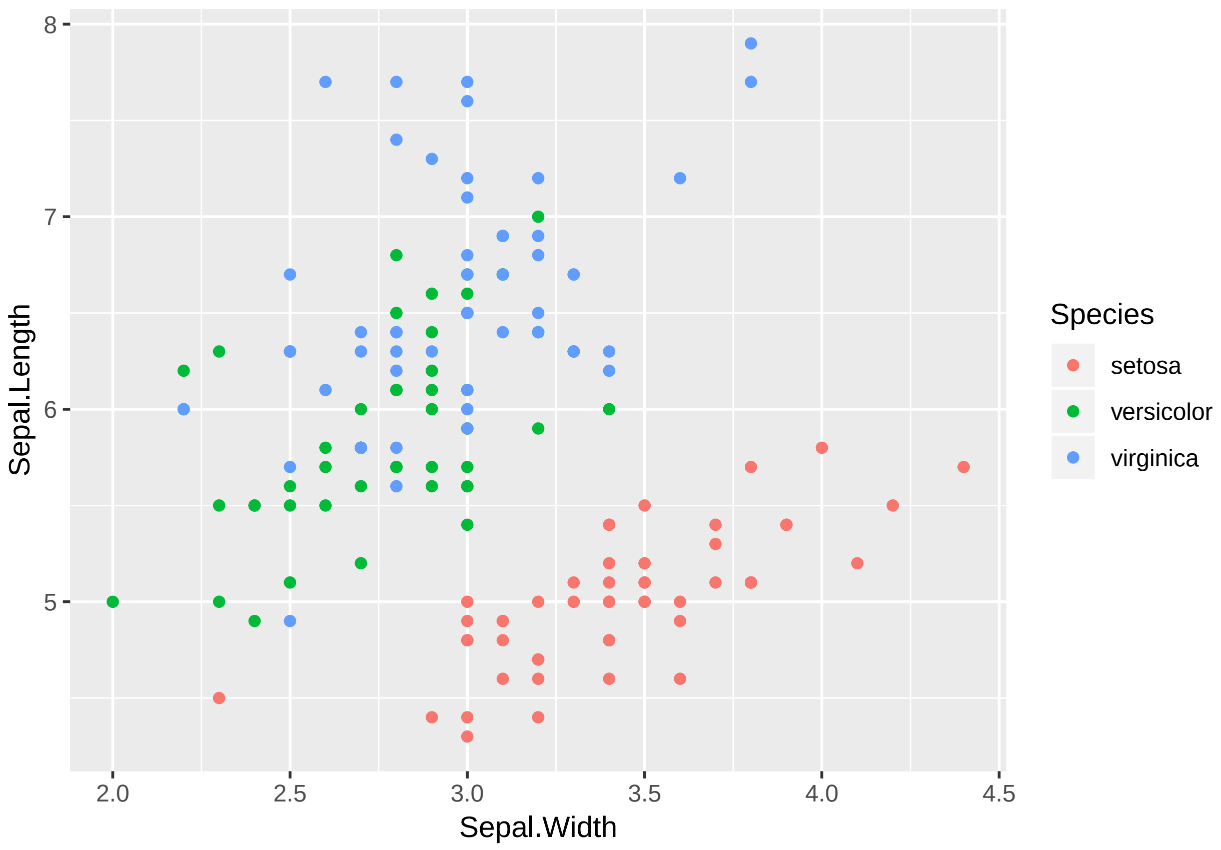
```
ggplot(iris, aes(x = Sepal.Width, fill = Species)) + geom_density() + facet_wrap(. ~ Species)
```



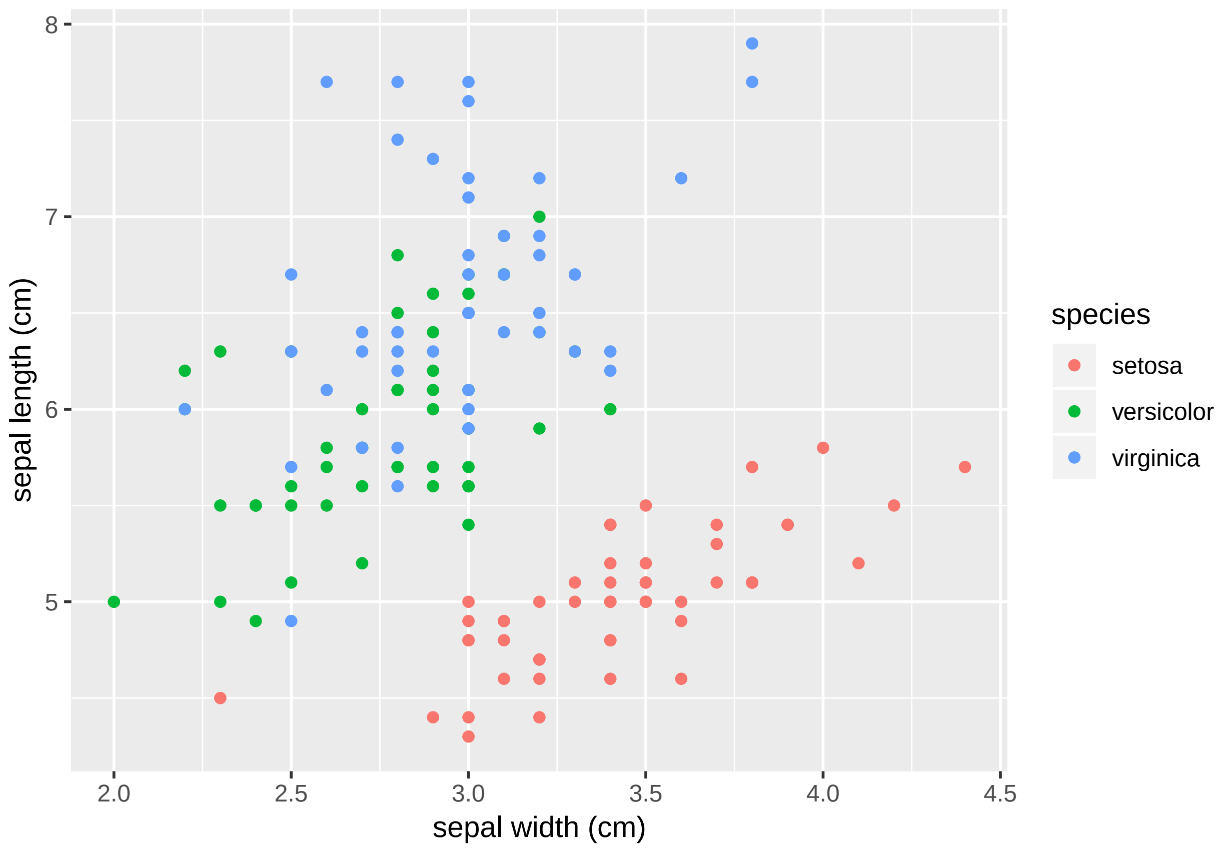
labels

The best practice is to have your column names and labels in your data table formatted nicely so you can plot and not think about it. But someimes that isn't possible or you don't think about it and you need to rename axis, legend, etc. The easiest, with `labs()`

```r
# plot before labelling
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) + geom_point()
```



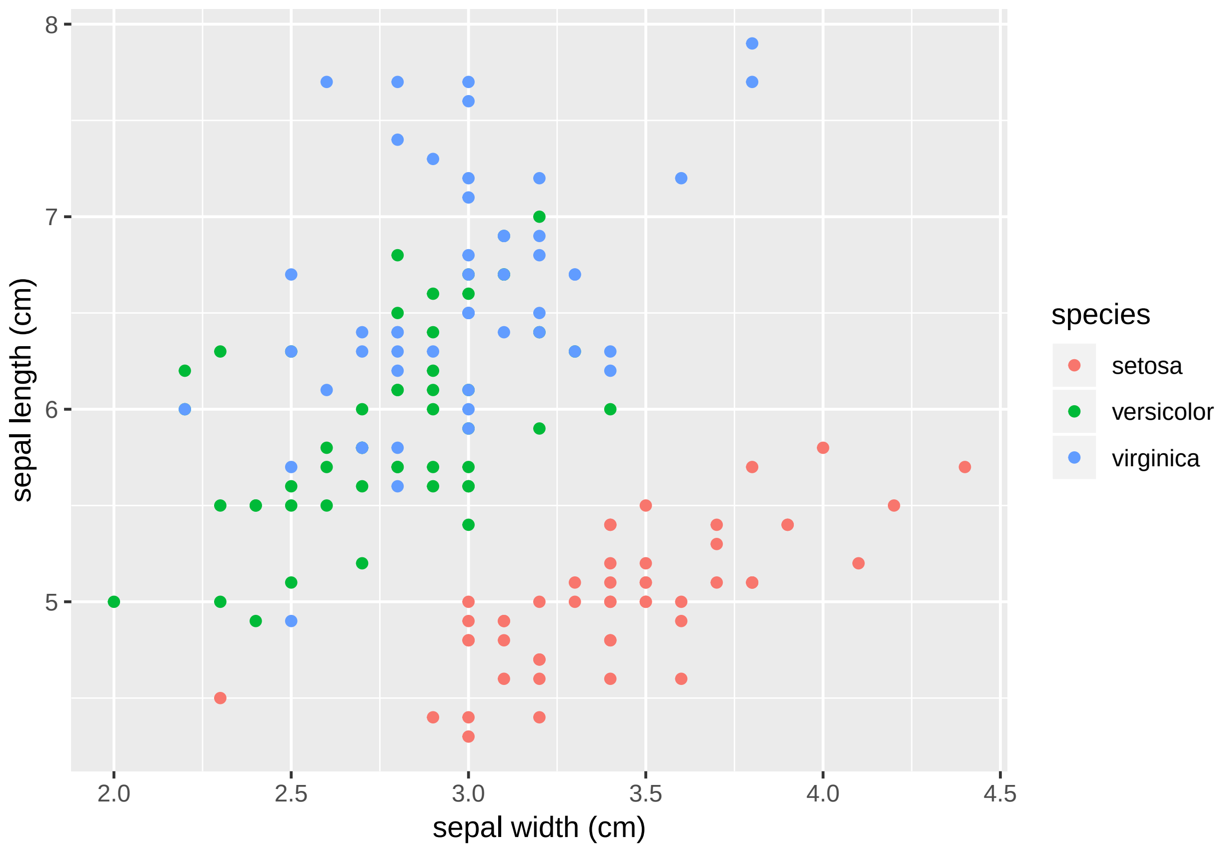
```r
# with labels added
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) + geom_point() +
  labs(x = 'sepal width (cm)', y = 'sepal length (cm)', color = 'species')
```



You can also modify the labels (and make more extensive modifications but only labels are shown here) with 'scale_??()‘. *The syntax is scale +* + plot part to modify + type of data (discrete or continuous mainly). These are the scale modifiers you'll use most often:

- scale_x_discrete()
- scale_x_continuous()
- scale_y_discrete()
- scale_y_continuous()
- scale_color_discrete()
- scale_color_continuous()
- scale_color_manual()

```r
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) + geom_point() +
  scale_x_continuous(name = 'sepal width (cm)') +
  scale_y_continuous('sepal length (cm)') +
  scale_color_discrete('species')
```

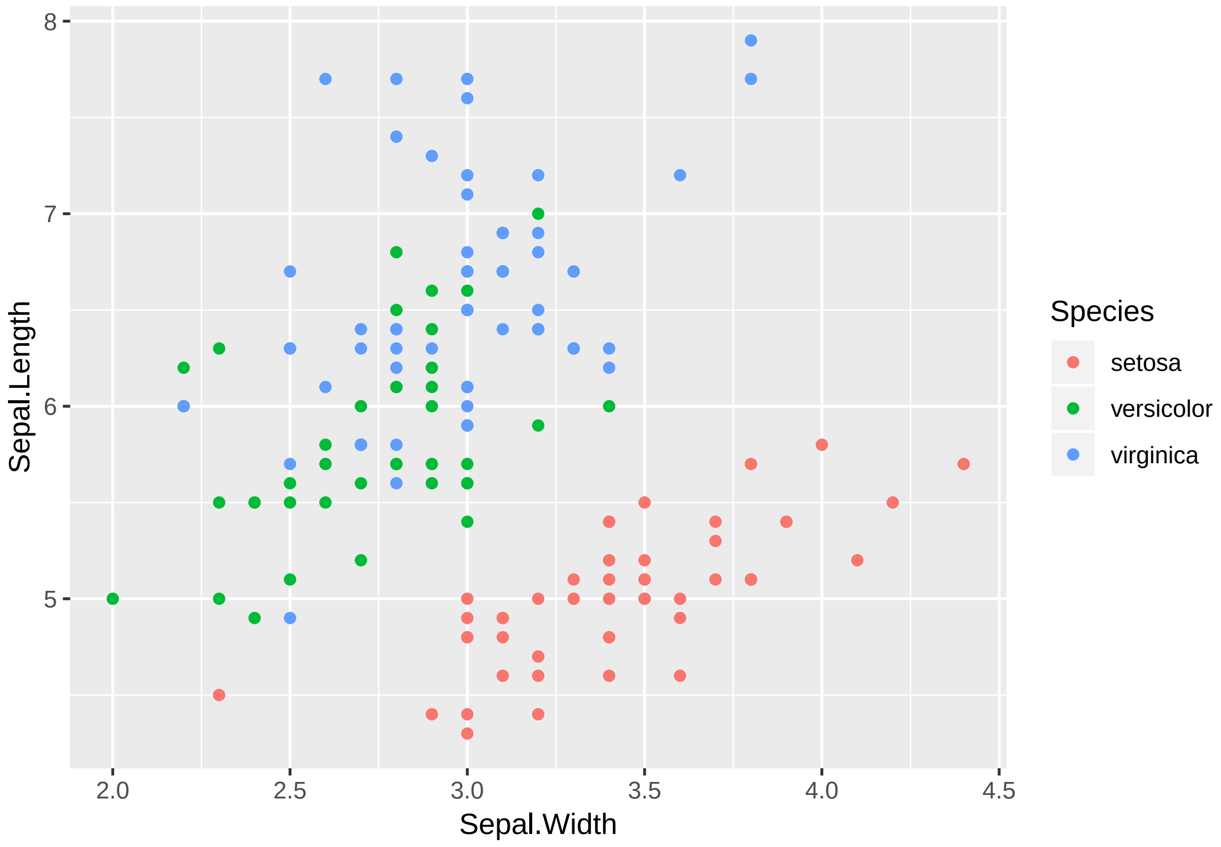


themes

The default theme in ggplot with the light gray background is kind of ugly, so people usually modify it with `theme_*()`. I've previewed the most common 3 below, but they all pop up if you type `?theme_classic()` in the console (or Google ggplot package themes)

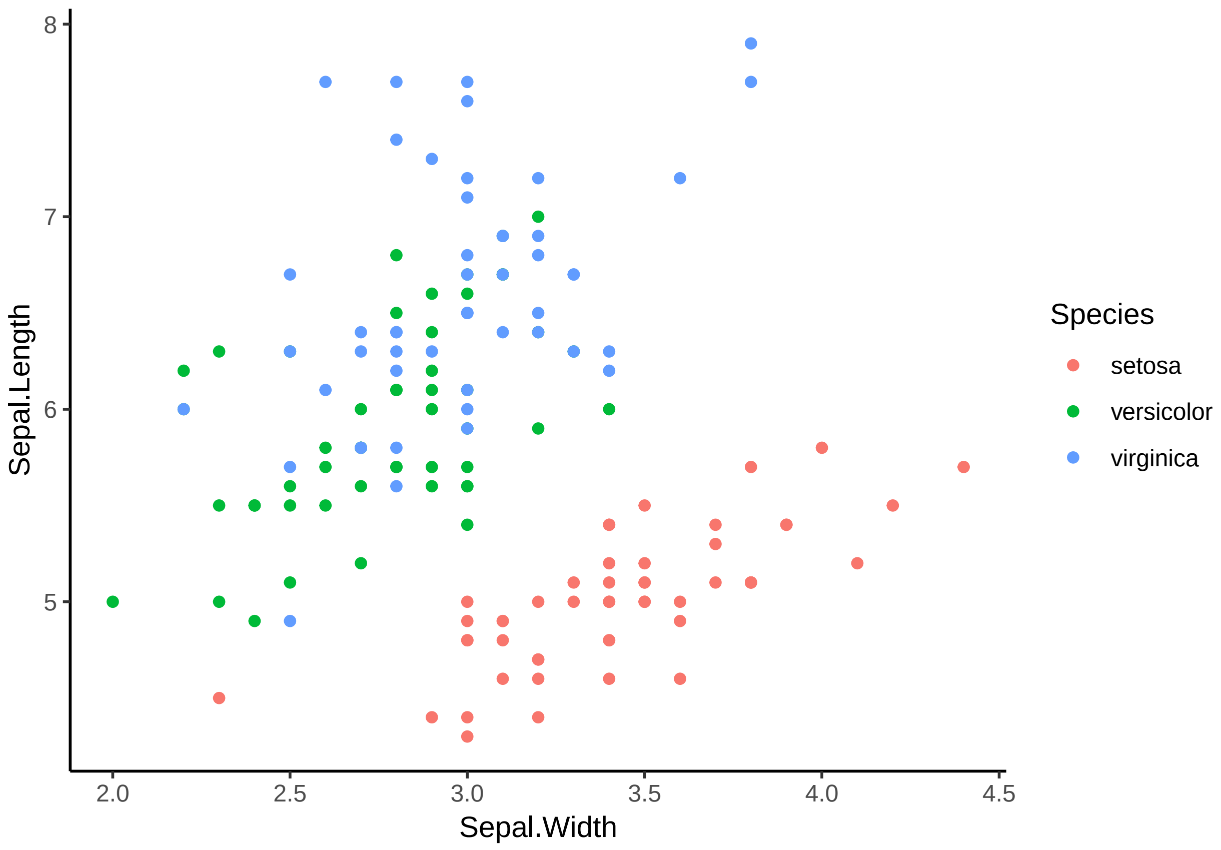
default

```
# default theme
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) + geom_point()
```



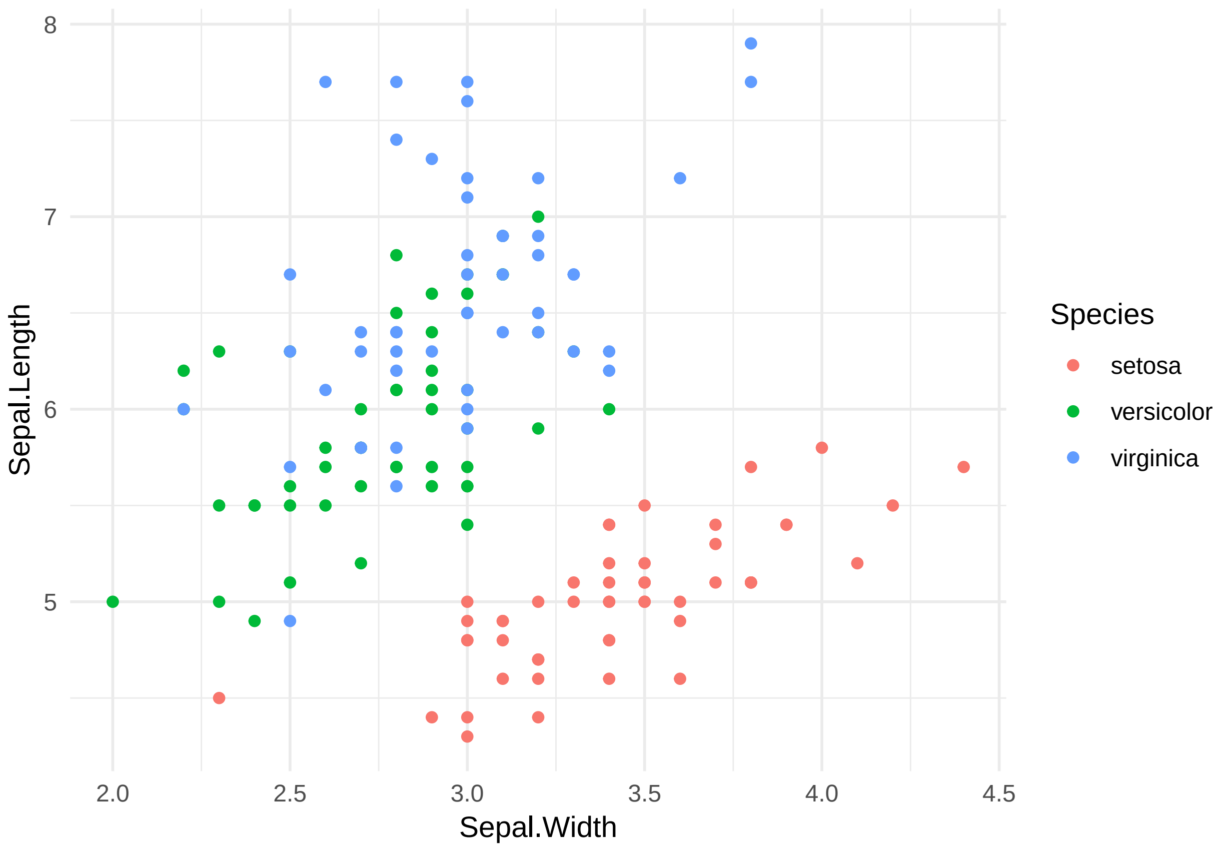
theme_classic() This is the one I use

```
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) + geom_point() + theme_classic()
```



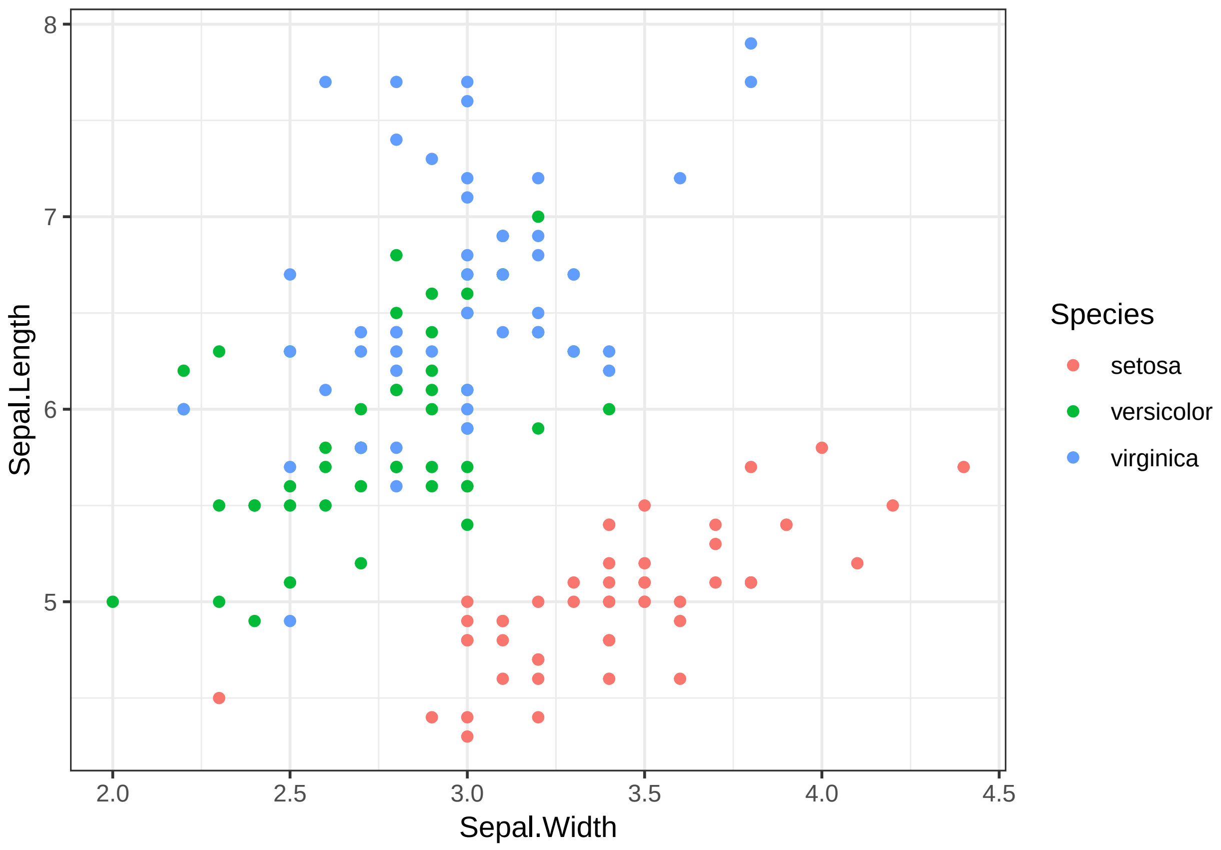
theme_minimal()

```
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) + geom_point() + theme_minimal()
```



theme_bw()

```
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) + geom_point() + theme_bw()
```



For fun, try `theme_void()` on something

viridis

Viridis is a package of color scales https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-viridis.html designed to address several problems. Viridis is

- **colorful:** It has the largest difference possible between the starting color value and the ending color value for to make differences easy to see
- **perceptually uniform:** The change from the starting color value to the ending color happens at the same right so the differences in similar appearing colors are the same across the color scale
- **color blind friendly:** 8.5% of people (at least in the US) have some form of colorblindness. Chances are there's a colorblind person viewing your paper. Viridis is designed so that color blind people will still see contrasts. Also, viridis still has contrast in greyscale, so it works if a figure is printed in black and white or for someone completely colorblind.
- It looks **pretty**

Get started

If it's not already installed, install the package

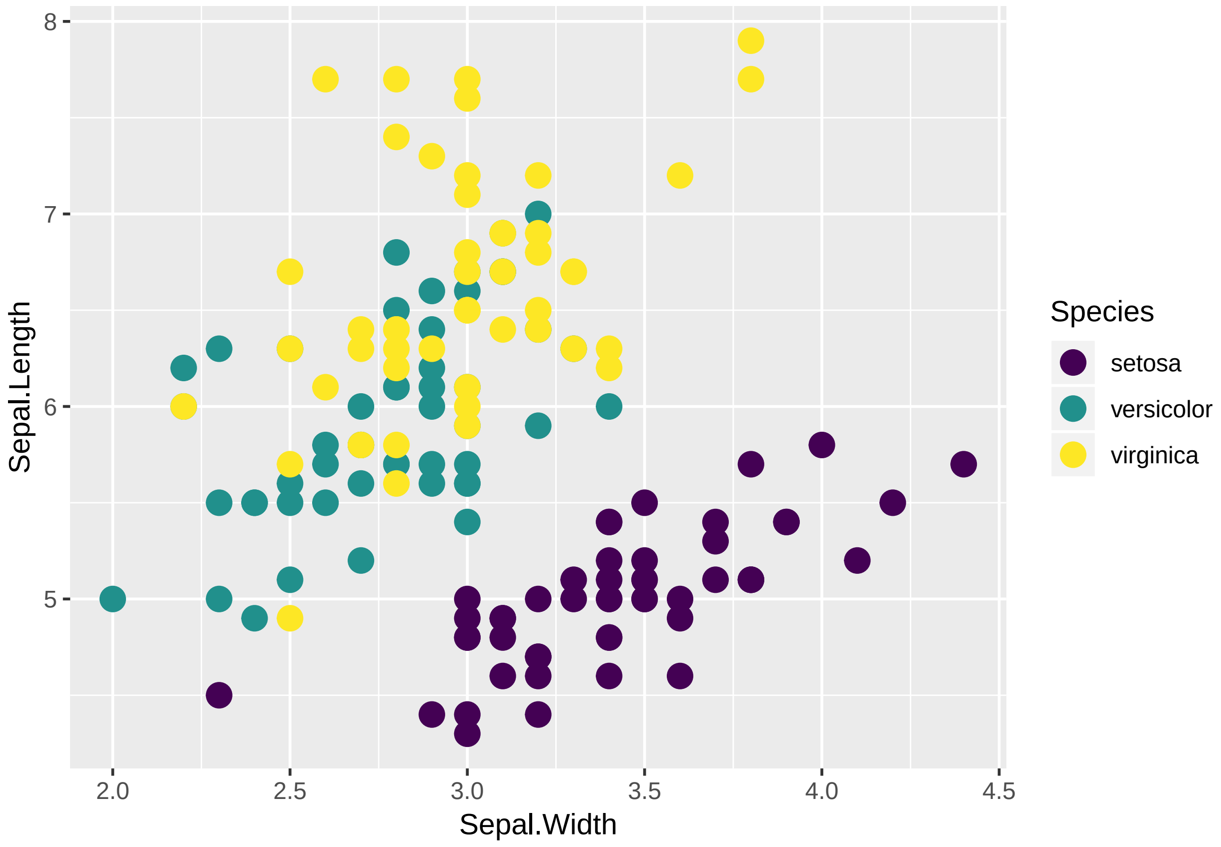
```
#install.packages('viridis')
```

You have to load the package before you can use the color scale

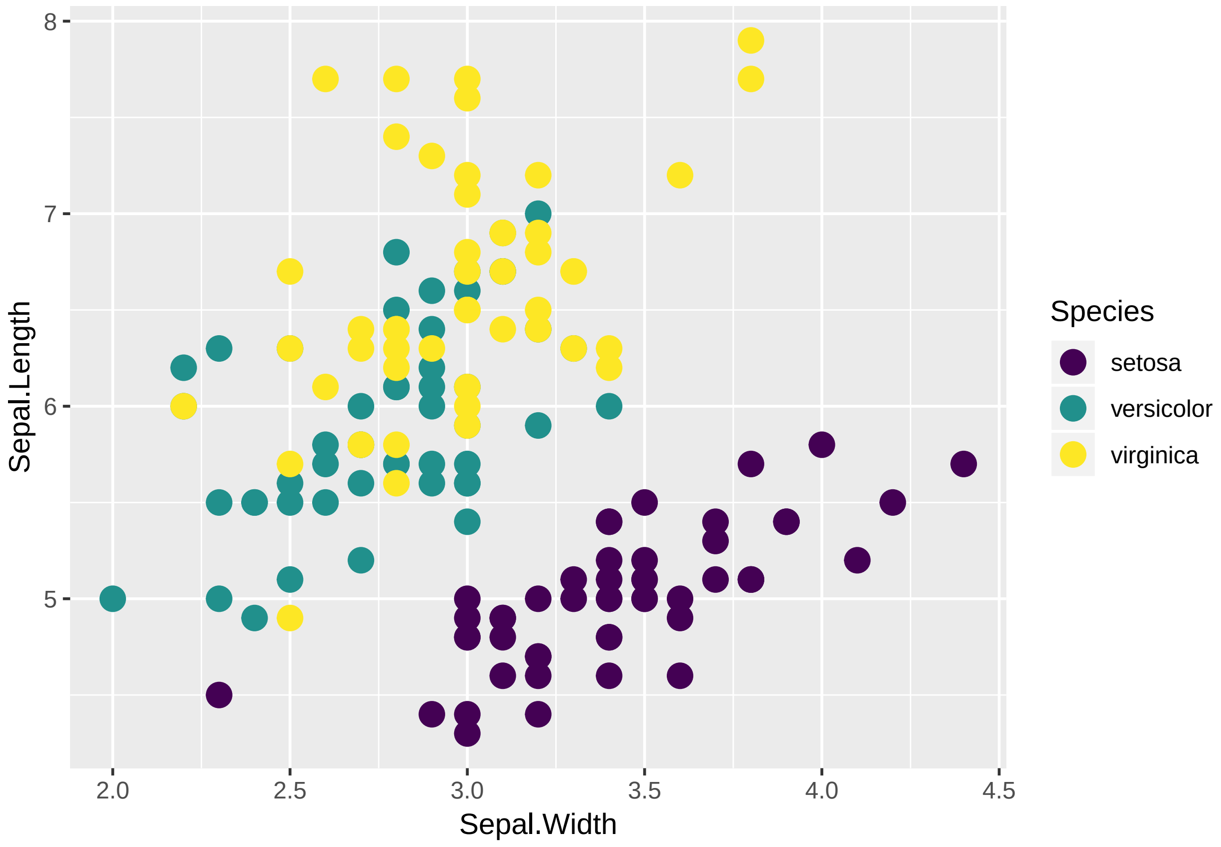
```
library(viridis)
```

Viridis coloring by discrete variable

```
# two different ways of specifying a discrete scale; continuous is the default
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) + geom_point(size = 4) + scale_co
```



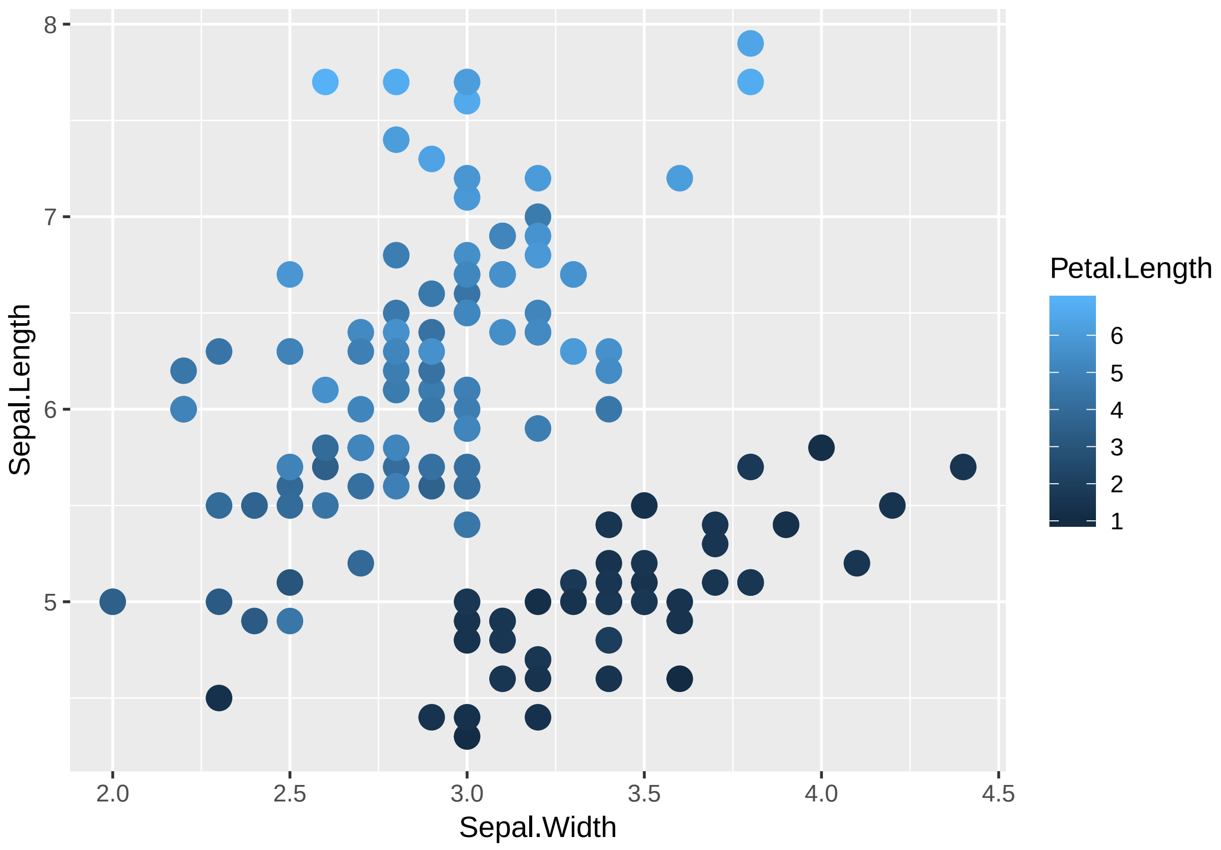
```
### OR

ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) + geom_point(size = 4) + scale_co
```



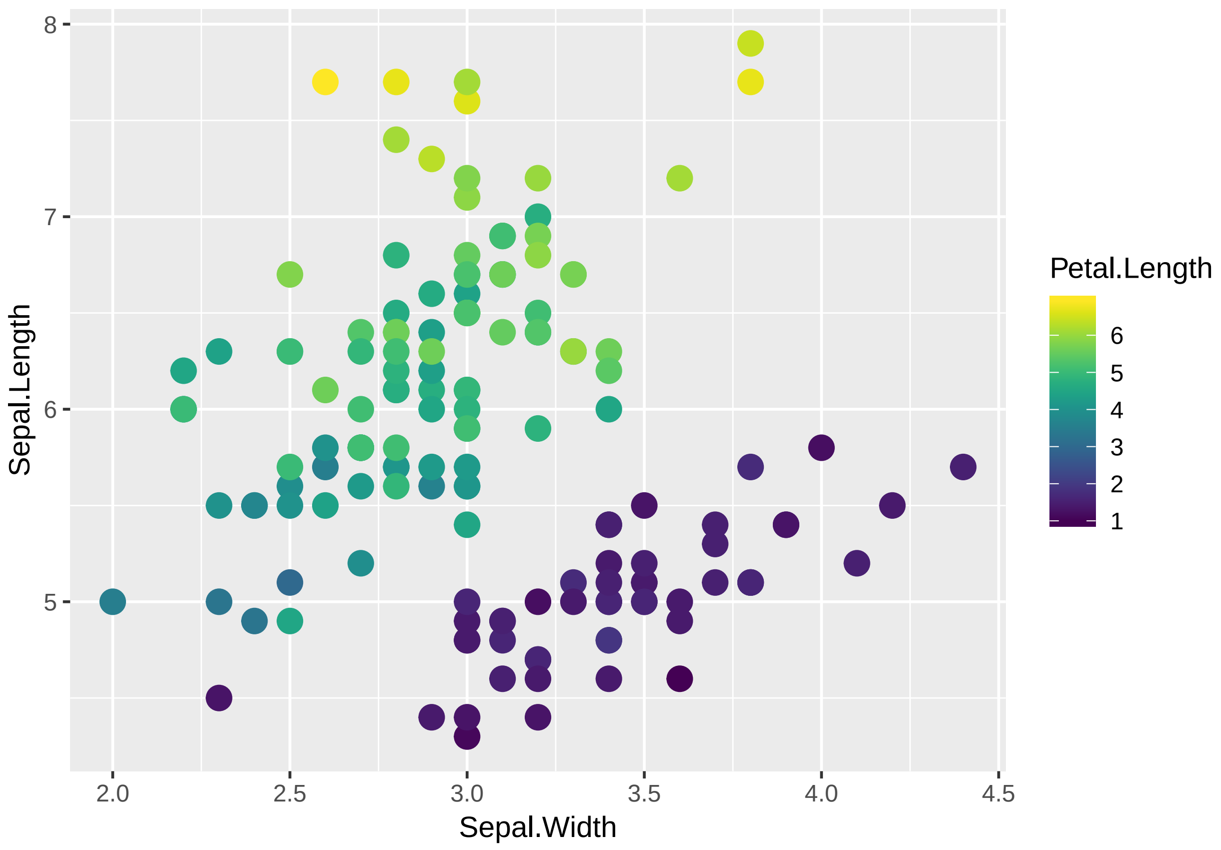
Viridis coloring by a continuous variable

The viridis scale has better contrast than the default ggplot color scale for continuous coloring

```
# default ggplot() continuous color scale
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Petal.Length)) + geom_point(size = 4)
```



```
# viridis continuous color scale
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Petal.Length)) + geom_point(size = 4) + scal
```


Sepal.Length
8
7
6
5
2.0
2.5
3.0
3.5
4.0
4.5
Sepal.Width
Petal.Length
6
5
4
3
2
1