Understanding color scales in `ggplot2`

Data Science for Biologists, Fall 2020

Stephanie J. Spielman

Whenever we **map** color or fill as an aesthetic, `ggplot2` uses a default color scheme. We will call this the *color or fill SCALE.* These default color/fill scales can be overridden if you can provide your own color scale. There are several types of popular scales that are brought to you by `colorbrewer` (https://colorbrewer2.org/) (a set of color scales originally devised for maps) and `viridis` (https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-viridis.html) (a set of color scales developed for plotting purposes). **This document considers ONLY those templated color schemes which are color-blind friendly.**

Quick Reference

> NOTE: In coding demos or templates, it is common to see `<>` used as a placeholder. This does not imply you should type the `<>`! It implies, text inside `<>` can be customized as stated.

Mapped data type	Scale type	Command
Discrete	`colorbrewer`	`scale_<color/fill>_brewer(palette = <name of palette>)`
Continuous	`colorbrewer`	`scale_<color/fill>_distiller(palette = <name of palette>)`
Discrete	`viridis`	`scale_<color/fill>_viridis(option = <name of palette>, discrete=TRUE)`
Continuous	`viridis`	`scale_<color/fill>_viridis(option = <name of palette>)`
Discrete	Custom	`scale_<color/fill>_manual(values = c(....))`
Continuous, sequential	Custom	`scale_<color/fill>_gradient(low = <low color>, high = <high color>)`
Continuous, diverging	Custom	`scale_<color/fill>_gradient2(low = <low color>, high = <high color>, mid = <mid color>, midpoint = <number>)`

Types of scales and their defaults

Qualitative scales contain a discrete set of distinct colors with no implied order. These scales may be used for discrete data mappings.

Sequential scales convey the ordering or magnitude of an effect. These scales may be used for discrete or continuous data mappings. When used for continuous data, we would refer to the scale as a *gradient.*

default sequential gradient (shown as continuous scale)

Diverging scales convey the ordering or magnitude of an effect *in two directions*. These scales may be used for discrete or continuous data mappings. When used for continuous data, we would refer to the scale as a *gradient2* (2 for 2 colors!)

default divergent gradient2 (shown as continuous scale)

Customizing qualitative scales for discrete data

The colorbrewer scales are all called via `scale_<color/fill>_brewer(palette = "name")` , where name is the name of the palette, e.g. palette = "Dark2". You can also include the `name` argument to change the name of the legend that appears associated with the mapping. **Colorblind friendly options are given below** (though there are many more palettes).

Dark2

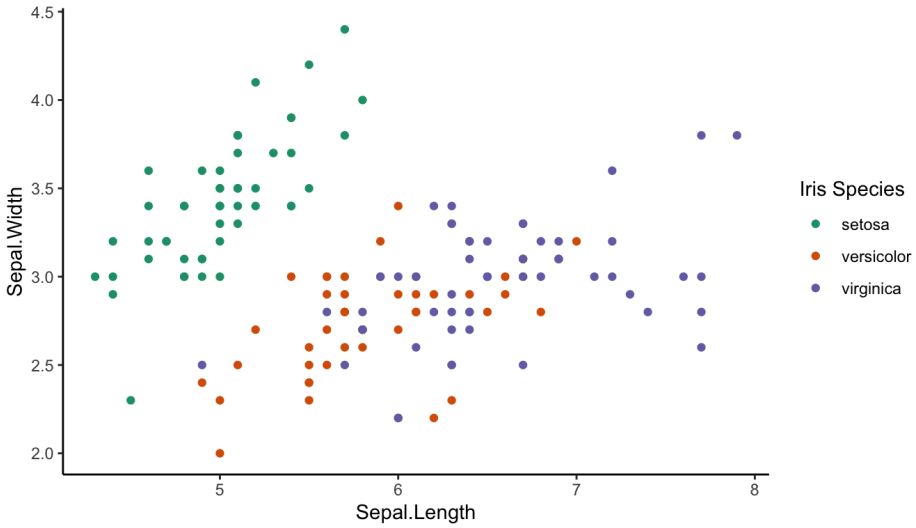
Paired

Set2

Examples:

```
ggplot(iris, aes(x = Sepal.Length, fill = Species)) +
  geom_density(alpha = 0.6) +
  # scale_FILL_brewer accompanies a FILL aesthetic
  scale_fill_brewer(palette = "Dark2", name = "Iris Species")
```


```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
  geom_point() +
  # scale_COLOR_brewer accompanies a COLOR aesthetic
  scale_color_brewer(palette = "Dark2", name = "Iris Species")
```

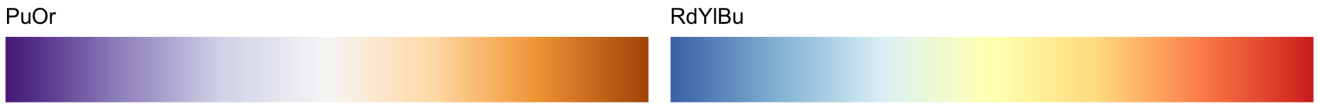


Customizing sequential and diverging scales: Continuous data mappings

The **colorbrewer sequential scales**, when used for continuous data mappings, are all called via `scale_<color/fill>_distiller(palette = "name")`, where name is the name of the palette, e.g. palette = "Blues". Again, you can also include the `name` argument to change the name of the legend that appears associated with the mapping. To flip the direction of these scales, use the argument `direction=1` (default is -1, just to confuse you!).

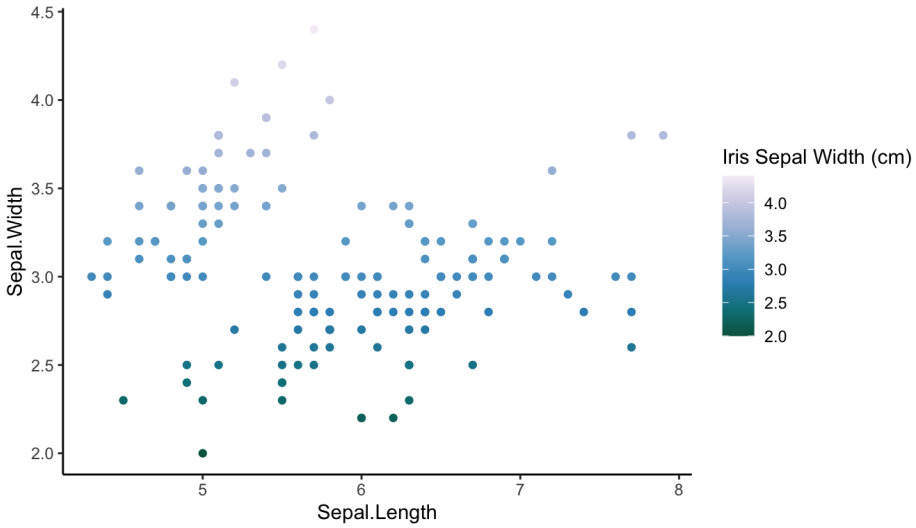
Sequential Gradients

Diverging gradients

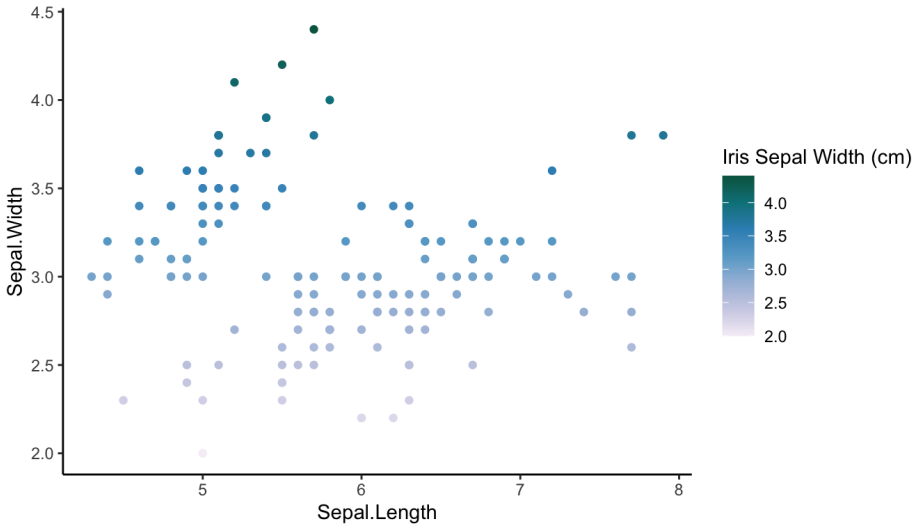


Examples:

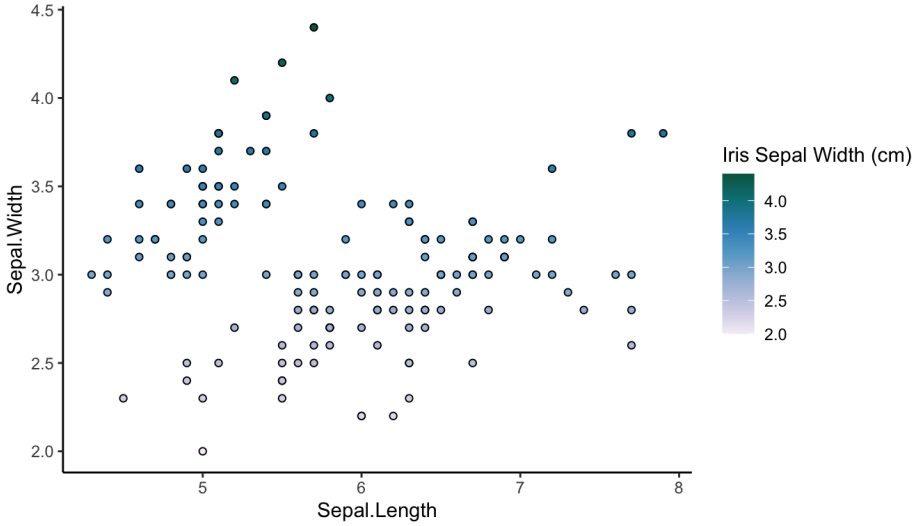
```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Sepal.Width)) +
  geom_point() +
  # scale_COLOR_distiller accompanies a COLOR aesthetic
  scale_color_distiller(palette = "PuBuGn", name = "Iris Sepal Width (cm)")
```



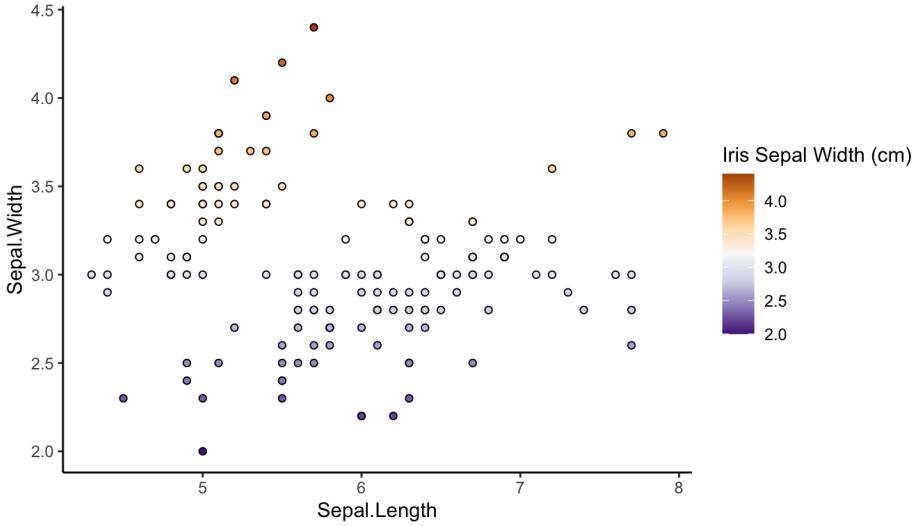

```
# Want the gradient going the opposite way? use `direction = 1` (default is -1)
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Sepal.Width)) +
  geom_point() +
  # scale_COLOR_distiller accompanies a COLOR aesthetic
  scale_color_distiller(palette = "PuBuGn", name = "Iris Sepal Width (cm)", direction = 1)
```



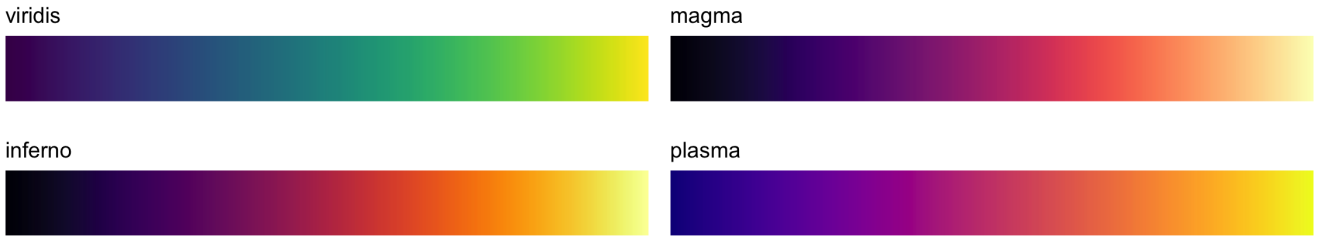

```
# Protip! These points are hard to see. Maybe a filled point could be more helpful?
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, fill = Sepal.Width)) +
  geom_point(shape = 21, color = "black") +
  # scale_FILL_distiller accompanies a FILL aesthetic
  scale_fill_distiller(palette = "PuBuGn", name = "Iris Sepal Width (cm)", direction = 1)
```




```
## DIVERGING gradient:
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, fill = Sepal.Width)) +
  geom_point(shape = 21, color = "black") +
  # scale_FILL_distiller accompanies a FILL aesthetic
  scale_fill_distiller(palette = "PuOr", name = "Iris Sepal Width (cm)")
```

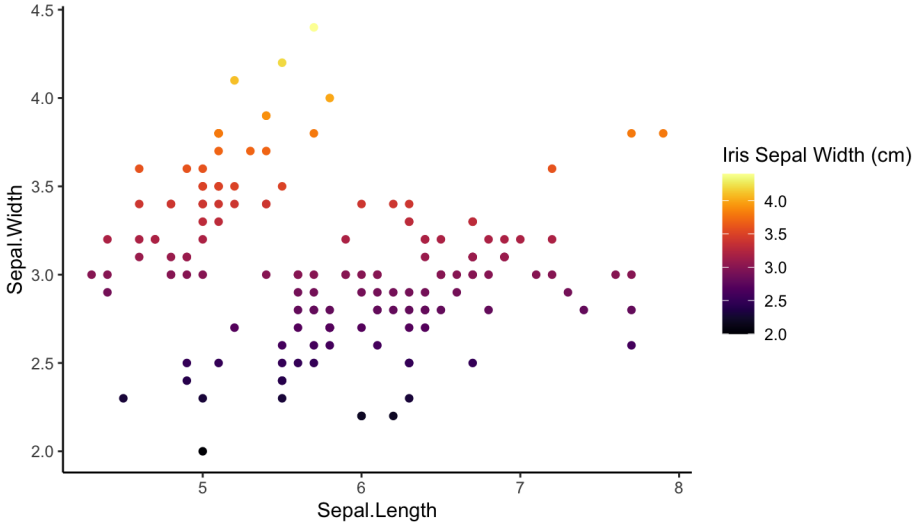


The **viridis sequential scales**, when used for continuous data mappings, are all called via `scale_<color/fill>_viridis(option = "name")`, where "name" is the name of the palette, e.g. "magma". Again, you can also include the `name` argument to change the name of the legend that appears associated with the mapping. *To use these palettes you must load the viridis library:* `library(viridis)`. To flip the direction of viridis scales, use the argument `direction=-1`.



Examples:

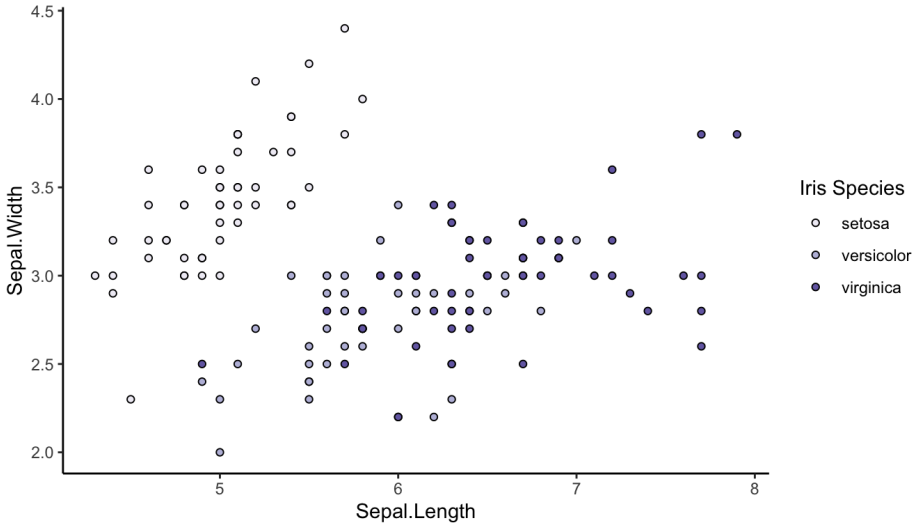
```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Sepal.Width)) +
  geom_point() +
  # scale_COLOR_distiller accompanies a COLOR aesthetic
  scale_color_viridis(option = "inferno", name = "Iris Sepal Width (cm)")
```



Customizing sequential and diverging scales: Discrete data mappings

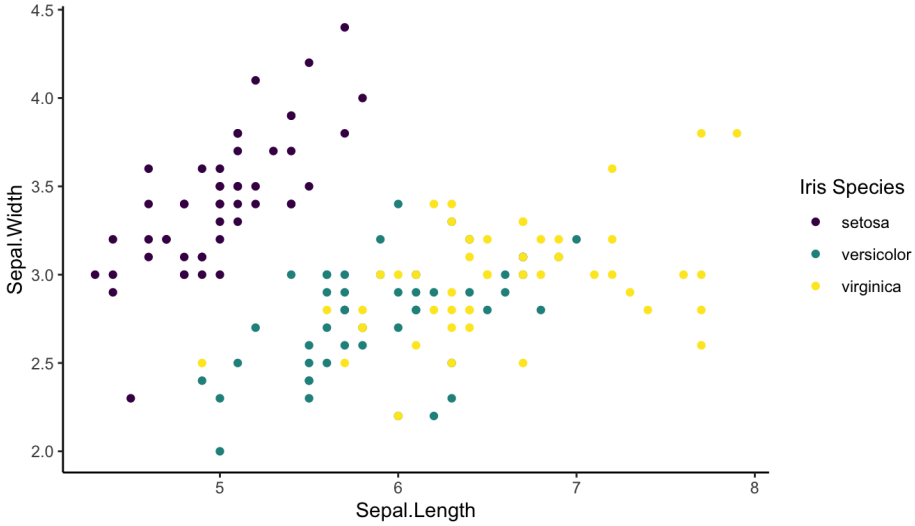
The colorbrewer sequential scales, when used for discrete data mappings, are all called via `scale_<color/fill>_brewer(palette = "name")`, where "name" is the name of the palette, e.g. "Blues". The palette options are the same as given above. For example,

```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, fill = Species)) +  # Species is DISCRETE
  geom_point(shape = 21, color = "black") +
  scale_fill_brewer(palette = "Purples", name = "Iris Species")
```



The viridis sequential scales, when used for continuous data mappings, are all called via `scale_<color/fill>_viridis(option = "name", discrete=TRUE)`, where "name" is the name of the palette, e.g. "magma". **The `discrete=TRUE` argument allows you to use these palettes for discrete mappings!** The palette options are the same as given above. For example,

```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +  # Species is DISCRETE
  geom_point() +
  # Note: if you don't include `option`, uses the default viridis palette
  scale_color_viridis(name = "Iris Species", discrete=TRUE)
```



Creating your own color scales

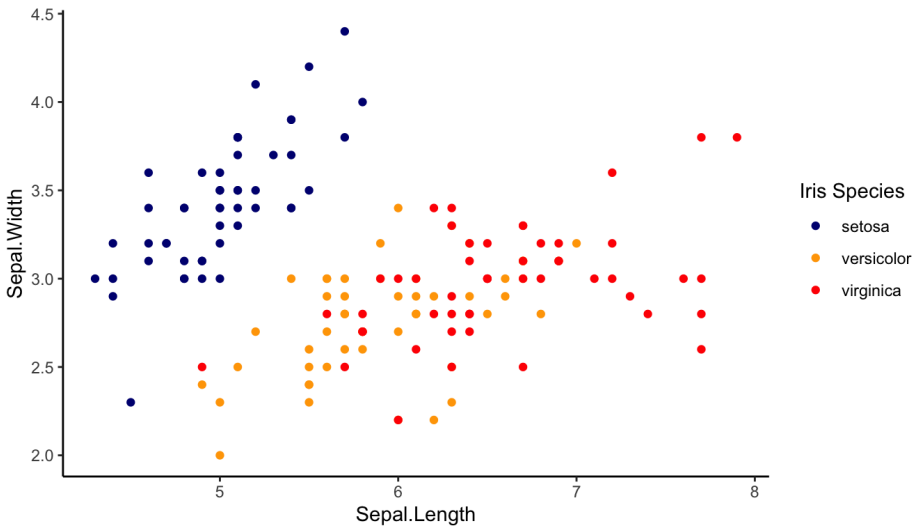
It is also possible to specify your own colors!

To specify a *discrete* color/fill scale, use `scale_<color/fill>_manual(values=c(...))` where you provide a *correct-length* list of colors to the `values` argument:

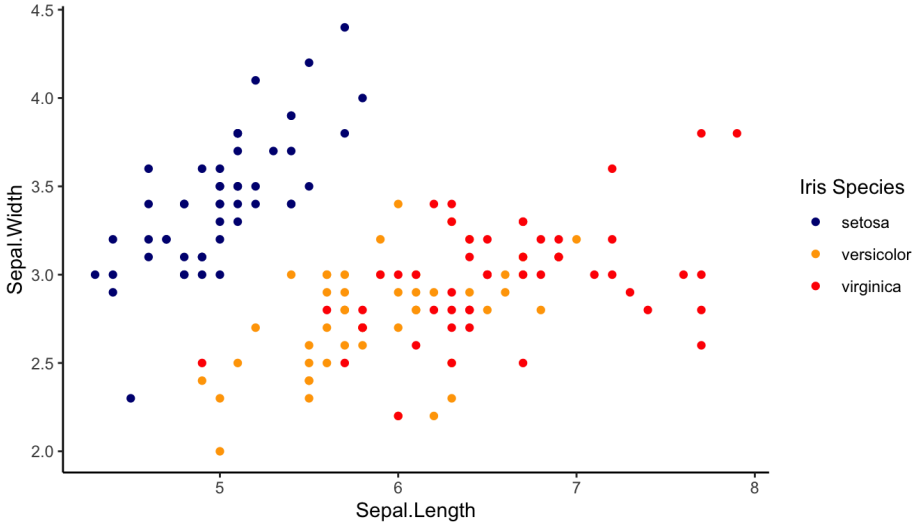
```
# First, check how many mappings? There are THREE species, so we need THREE colors
length(levels(iris$Species))
```

```
## [1] 3
```

```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +  # Species is DISCRETE with THREE LEVELS
  geom_point() +
  scale_color_manual(values = c("navy", "orange", "red"), name = "Iris Species")
```

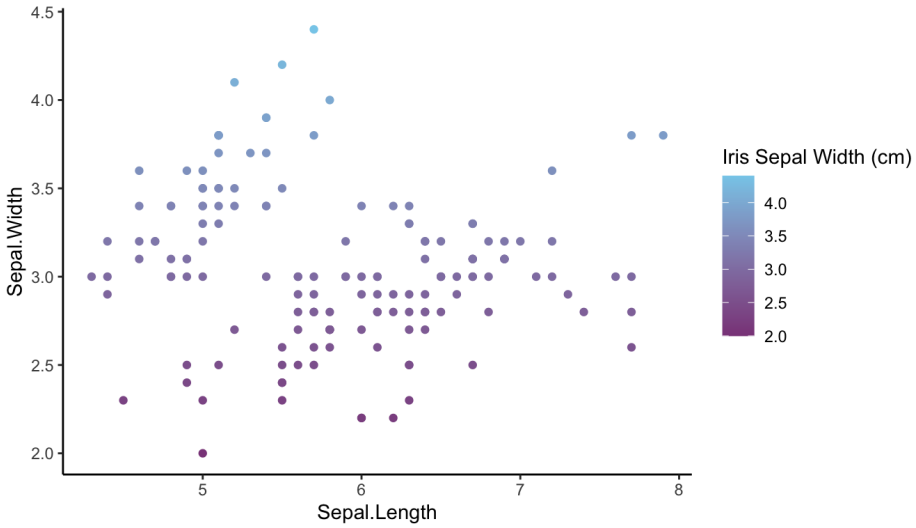



```
# Get fancier with variables?!?!?!! :)
my_three_colors <- c("navy", "orange", "red")
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
  geom_point() +
  scale_color_manual(values = my_three_colors, name = "Iris Species")
```



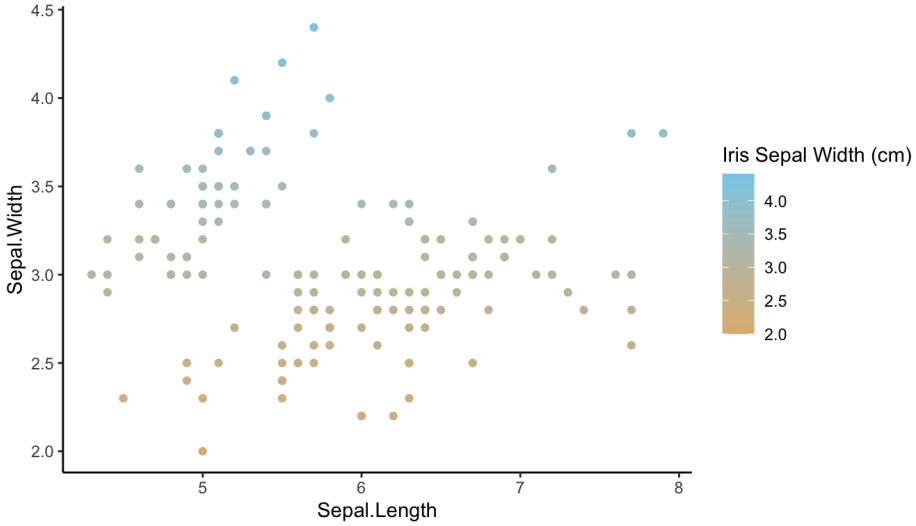
To specify a *continuous sequential* color/fill scale, use `scale_<color/fill>_gradient(low = <low>, high = <high>)` where you provide a color for the low-end and high-end of the custom gradient:

```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Sepal.Width)) +
  geom_point() +
  scale_color_gradient(low = "orchid4",
                       high = "skyblue",
                       name = "Iris Sepal Width (cm)")
```

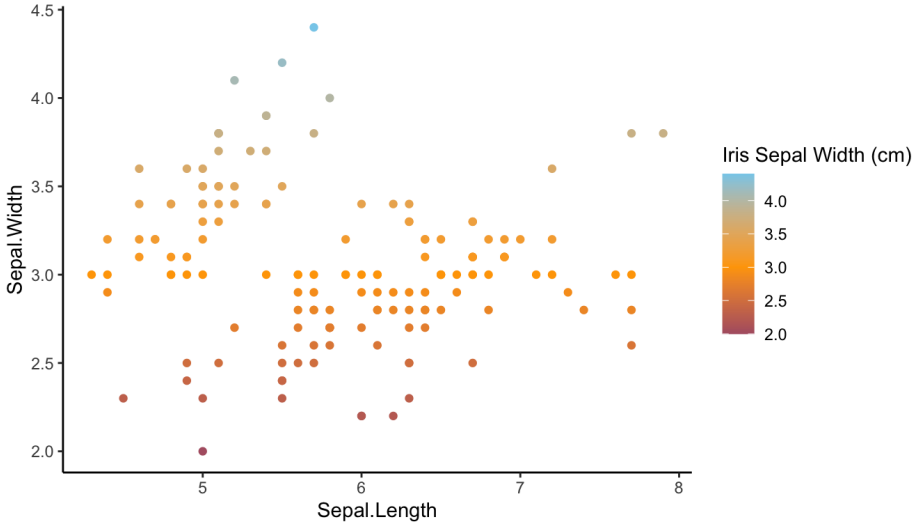


To specify a *continuous diverging* color/fill scale, use `scale_<color/fill>_gradient2(low = <low>, high = <high>, mid = <mid>)` where include a middle color for the gradient to pass through:

```
# Where is orchid4??!?! Since function assumes midpoint is 0, we never hit it.
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Sepal.Width)) +
  geom_point() +
  scale_color_gradient2(low = "orchid4",
                        high = "skyblue",
                        mid = "orange",
                        name = "Iris Sepal Width (cm)")
```




```
# Add a midpoint argument:
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Sepal.Width)) +
  geom_point() +
  scale_color_gradient2(low = "orchid4",
                        high = "skyblue",
                        mid = "orange",
                        midpoint = 3,
                        name = "Iris Sepal Width (cm)")
```



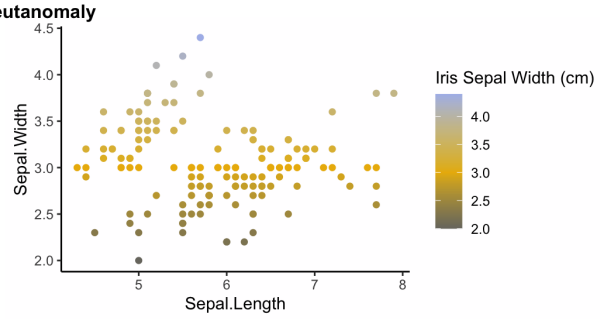
Checking if your figure is colorblind friendly

To check any figure, you will need the package `colorblindr` (https://github.com/clauswilke/colorblindr). *This package is installed for you on RStudio Cloud*, but you need to load it to use in every session with `library(colorblindr)`.

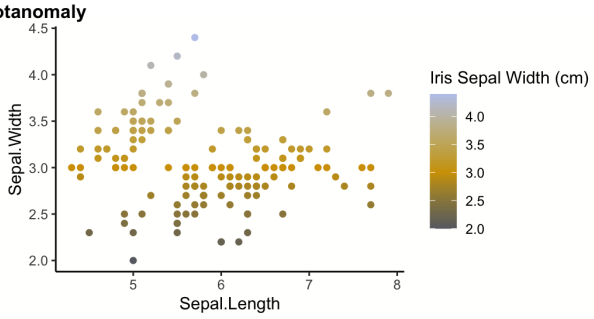
For example, using the customized gradient2 figure with the midpoint of 3 shown above:

```r
library(colorblindr) # Make sure this has been run!

# Save figure to a variable, here `iris_plot`
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Sepal.Width)) +
  geom_point() +
  scale_color_gradient2(low = "orchid4",
                        high = "skyblue",
                        mid = "orange",
                        midpoint = 3,
                        name = "Iris Sepal Width (cm)") -> iris_plot

# Use the function `cvd_grid()` to interactively check your figure's accessibility.
# This one is not terrible, but the color scale still could be a lot better (esp for Tritanomaly)
cvd_grid(iris_plot)
```

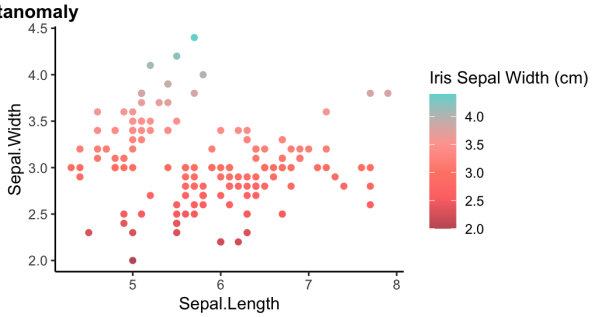
Deutanomaly



Protanomaly



Tritanomaly



Desaturated