KEY_Lesson20_Improving_plots_1

August 28, 2019

1 Improving plots I - Scatterplots

So far you have learned the basics of plotting data. We've seen that data visualization is a very powerful tool that allows us to communicate a lot of information in a clear and concise way. By adding more elements to our plots, we can communicate even more information in a single graph. In this lesson we will focus on using color to enhance our plots.

Up until now, we have focused on using **matplotlib** for our plotting and **seaborn** to load example datasets. However, **seaborn** can also be used for plotting, and allows for easily customizable coloring of points. Here, we will learn about coloring our plots using **seaborn** plotting functions.

[0]:
```
# import the packages we need: numpy, matplotlib, seaborn
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

For our first example, we will utilize the iris dataset from seaborn. Let's load the data and remind ourselves what information is contained in the dataset by looking at the first few rows.

[2]:
```
# load iris and preview the data
iris = sns.load_dataset("iris")
iris.head(10)
```

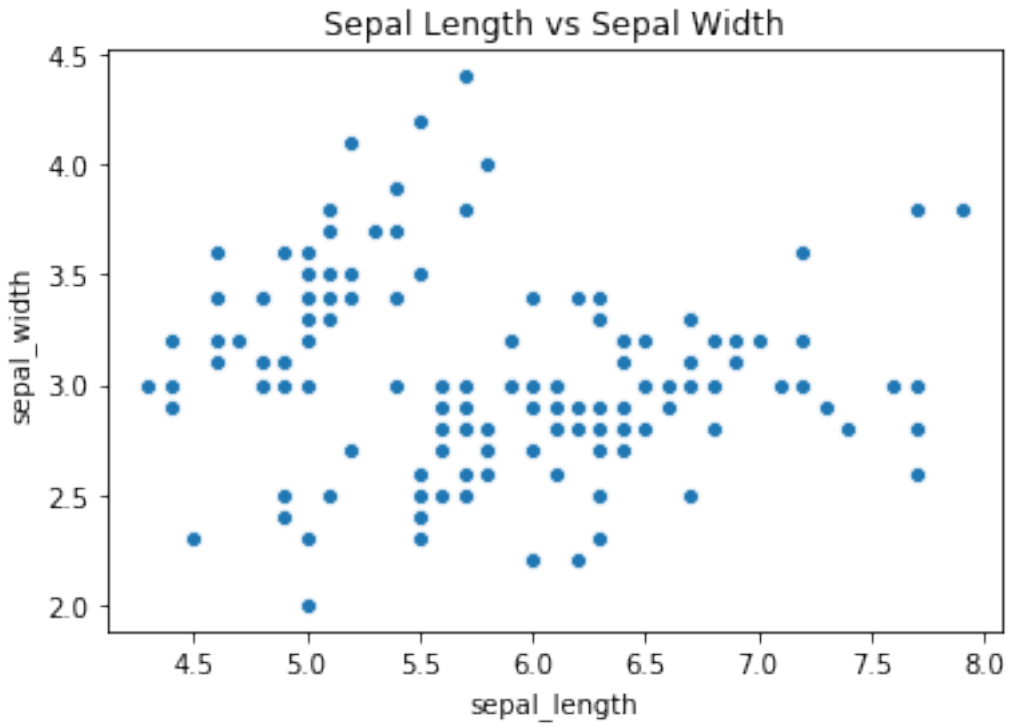
[2]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa

Say we want to look at the relationship between `sepal_length` and `sepal_width` within our dataset. From our previous lessons, we learned that a scatterplot is the best tool to look at the relationship between two variables. We'll use the `sns.scatterplot` function to plot this.

[3]:
```
# plot sepal_length vs sepal_width
sns.scatterplot('sepal_length', 'sepal_width', data=iris)
# dont forget a title!
# We can add a title layer using plt.title as we've done before
plt.title("Sepal Length vs Sepal Width")
```

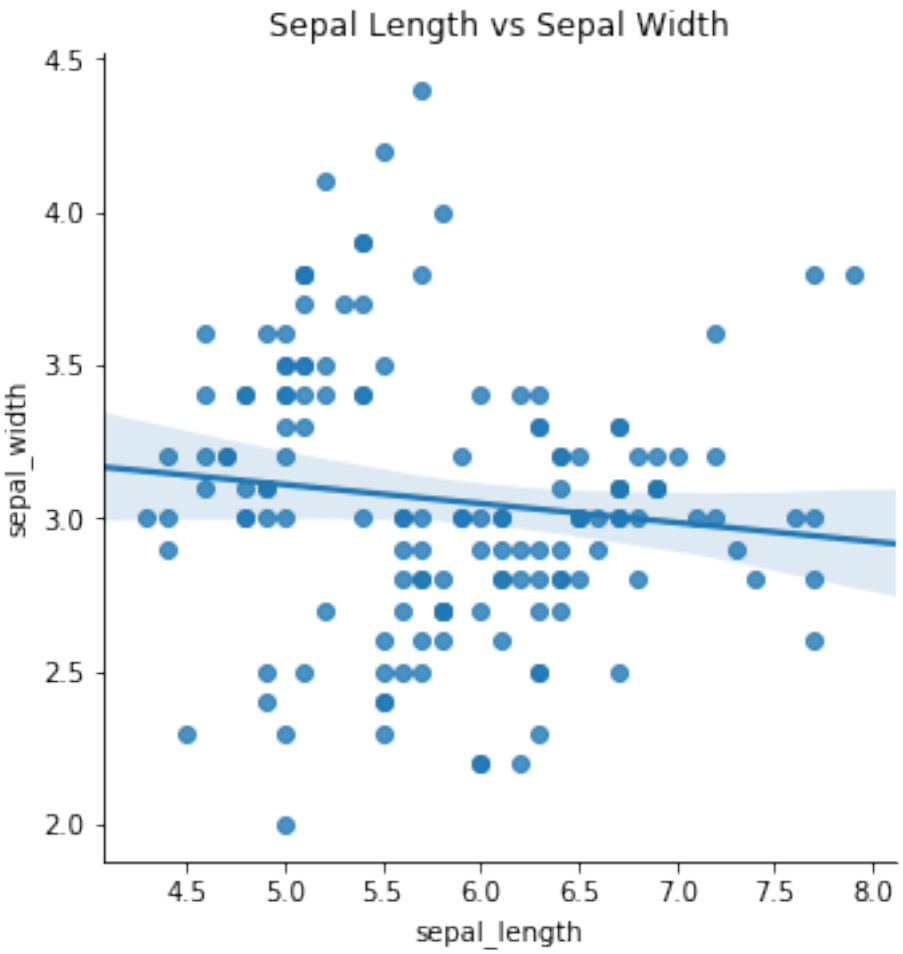
[3]: Text(0.5, 1.0, 'Sepal Length vs Sepal Width')



If we would like to be even more specific, we could use the `sns.lmplot` function, which works similarly to `sns.scatterplot`, but adds a linear trendline to help visualize the relationship between our x and y variables.

[4]:
```
# plot sepal_length vs sepal_width
sns.lmplot('sepal_length', 'sepal_width', data=iris)
# dont forget a title!
# We can add a title layer using plt.title as we've done before
plt.title("Sepal Length vs Sepal Width")
```

[4]: Text(0.5, 1.0, 'Sepal Length vs Sepal Width')



1.1 Coloring by a Categorical Varible

This gives us a general idea of the trend between `sepal_length` and `sepal_width`, but what if we wanted to explore the relationship between these variables on a more granular level? For example - if we wanted to see how this relationship might differ between the different species within our dataset? Let's first determine how many different species we have data for within our dataset

[5]:
```
# find unique values within the species column of iris
np.unique(iris.species)
```

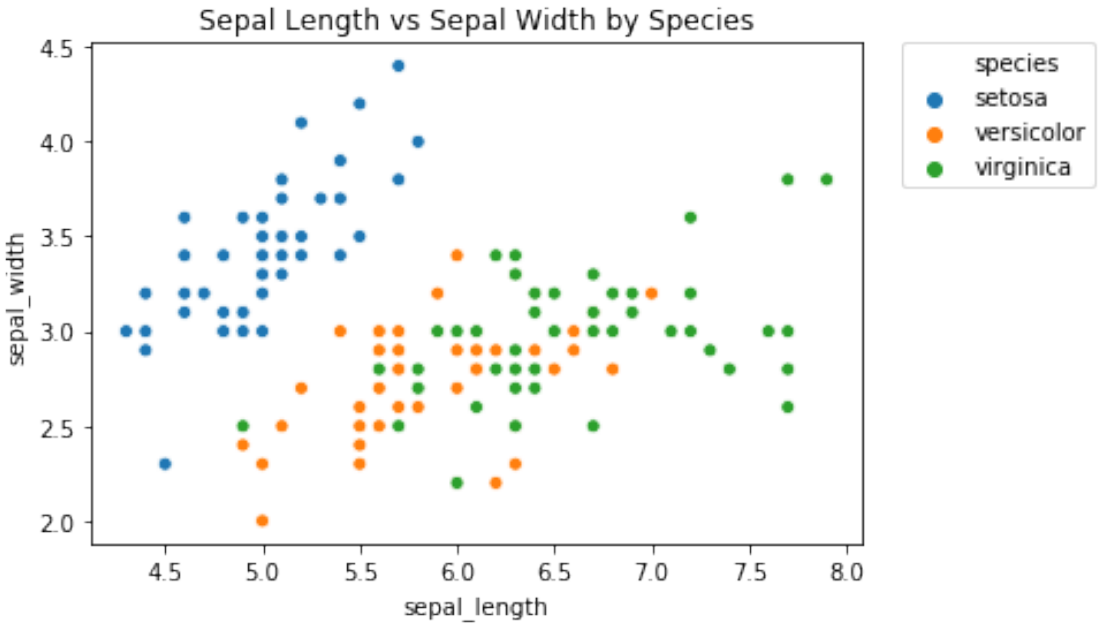
[5]:
```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

We want to highlight each of the three species using different colored points in our scatterplot. We can do this easily using the `hue` parameter.

[6]:
```
# plot sepal_length vs sepal_width colored by species
sns.scatterplot('sepal_length', 'sepal_width', data=iris, hue = 'species')
plt.title("Sepal Length vs Sepal Width by Species")

# the line below moves the legend outside of the plot borders
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

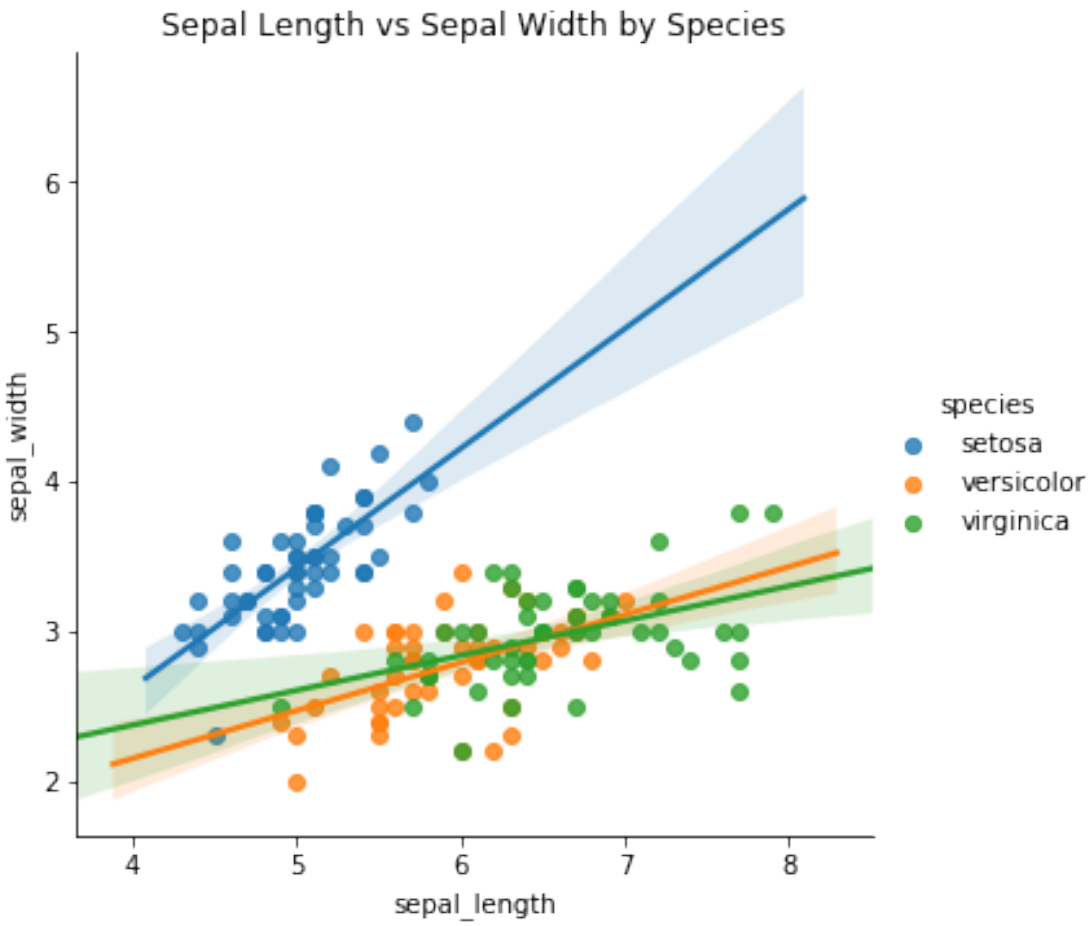
[6]:
```
<matplotlib.legend.Legend at 0x7f916165b550>
```



Similarly, we can use the `sns.lmplot` function to add a linear trendline for each species separately.

[7]:
```
# plot sepal_length vs sepal_width colored by species
sns.lmplot('sepal_length', 'sepal_width', data=iris, hue = 'species')
plt.title("Sepal Length vs Sepal Width by Species")
```

[7]:
```
Text(0.5, 1.0, 'Sepal Length vs Sepal Width by Species')
```



Now we can make some important conclusions about our data. When we looked at sepal length vs. sepal width across all of our data we did not see a strong relationship between the two variables (as evidenced by the nearly-flat trendline). However, when we explor this trend by species we see that the *setosa* species has the strongest relationship between sepal length and sepal width (as evidenced by the strongly sloped trendline), and the relationship between sepal length and sepal width for the *versicolor* and *virginica* species are similarly moderate.

You may also want to change the color palette you are working with. You can do this with the `palette` parameter. You can find built-in seaborn color palettes here: https://seaborn.pydata.org/tutorial/color_palettes.html

[8]:
```
# plot sepal_length vs sepal_width colored by species
sns.lmplot('sepal_length', 'sepal_width', data=iris, hue = 'species',␣
 ↪palette="Set2")
plt.title("Sepal Length vs Sepal Width by Species")
```

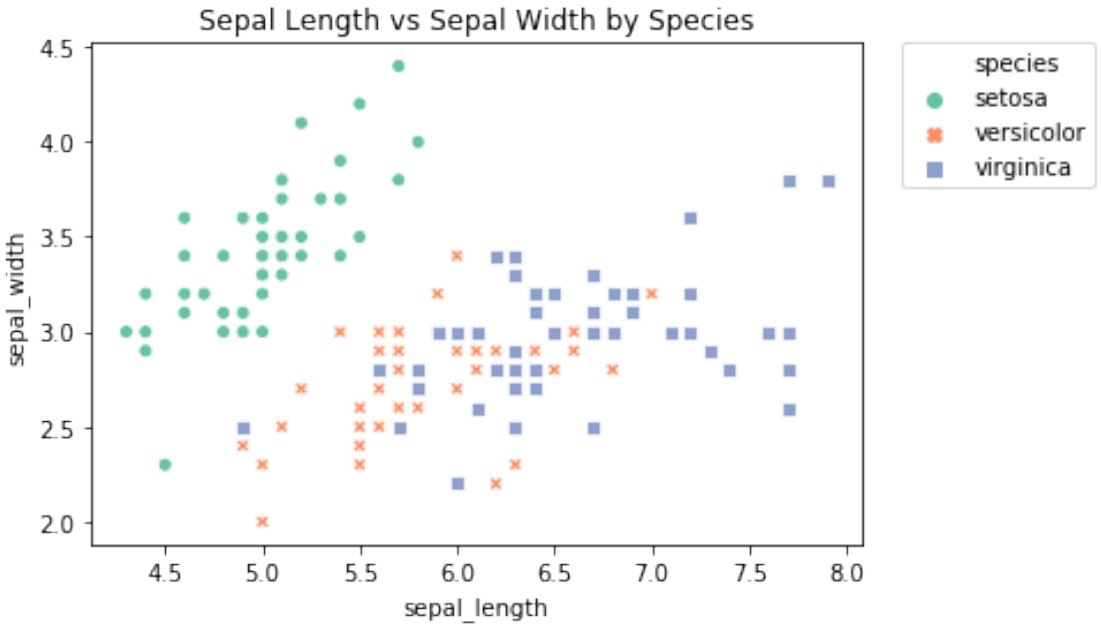
[8]:
```
Text(0.5, 1.0, 'Sepal Length vs Sepal Width by Species')
```

If we want to further emphasize the difference between our datapoints by species, we can also change the point markers, using the `style` parameter.

[9]:
```
# plot sepal_length vs sepal_width colored by species
sns.scatterplot('sepal_length', 'sepal_width', data=iris, hue = 'species',␣
↪style='species', palette = 'Set2')
plt.title("Sepal Length vs Sepal Width by Species")

# the line below moves the legend outside of the plot borders
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

[9]:
```
<matplotlib.legend.Legend at 0x7f91614c8940>
```

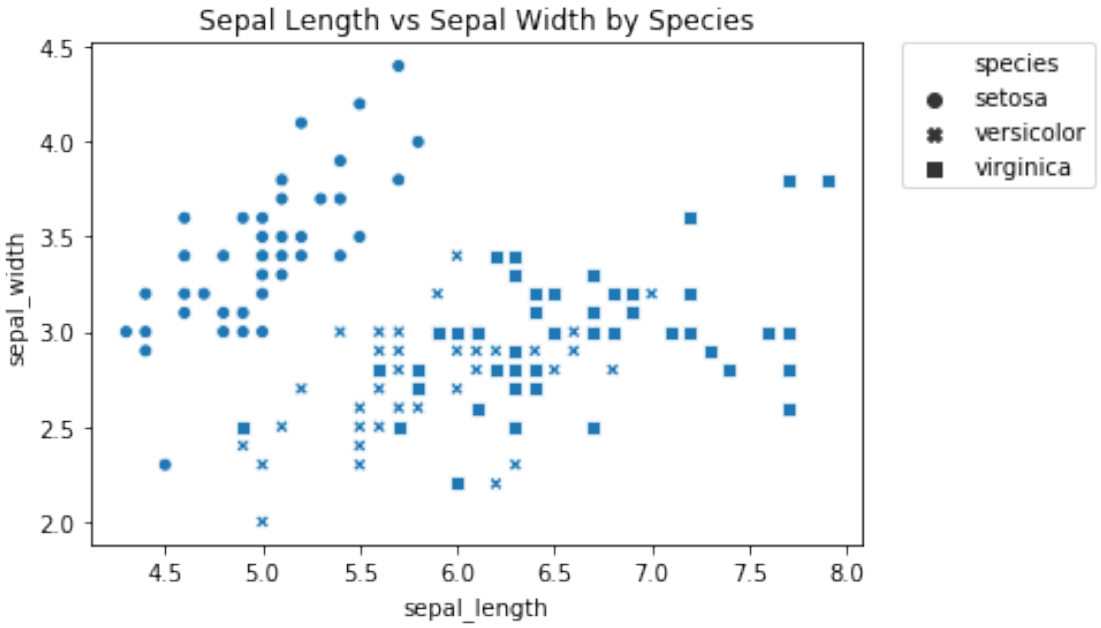


The `style` parameter can be used in conjunction with the `hue` parameter, or on its own.

[10]:
```
# plot sepal_length vs sepal_width colored by species
sns.scatterplot('sepal_length', 'sepal_width', data=iris, style='species')
plt.title("Sepal Length vs Sepal Width by Species")

# the line below moves the legend outside of the plot borders
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

[10]:
```
<matplotlib.legend.Legend at 0x7f916149b668>
```

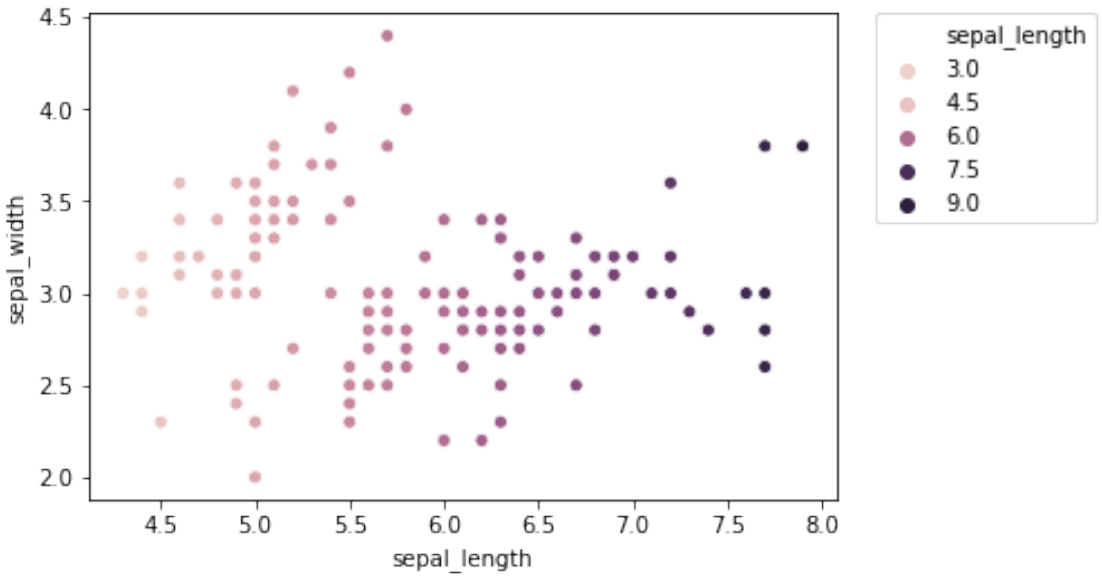


1.2 Coloring by a Continuous Variable

It is sometimes useful to color by a continuous variable rather than a categorical variable. For example, let's recreate our scatterplot from above, but instead of coloring by the `species` column, let's color by the `sepal_length` column.

[11]:
```
sns.scatterplot('sepal_length', 'sepal_width', data=iris, hue='sepal_length')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

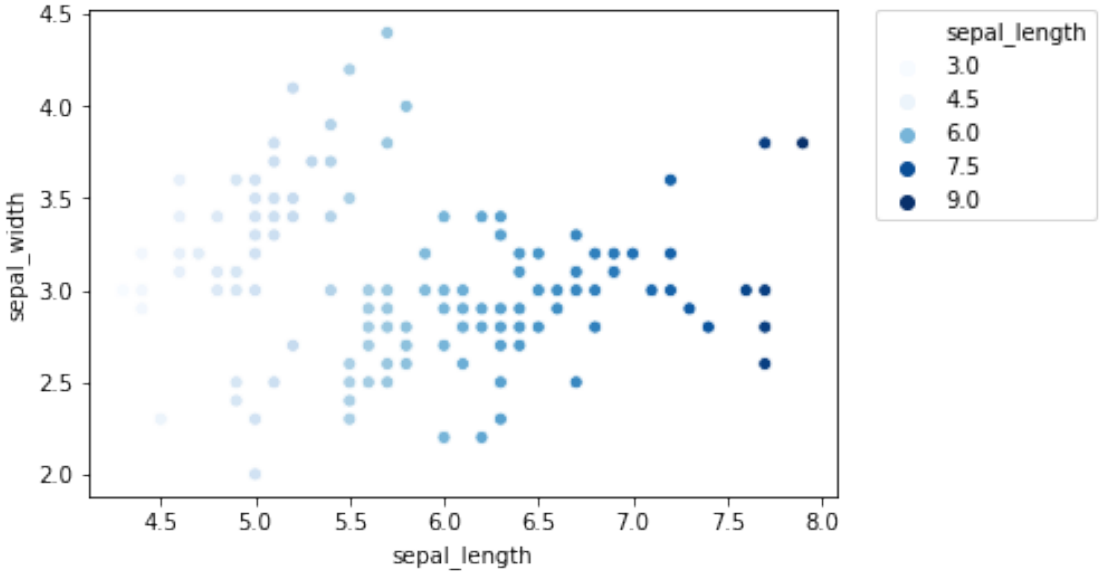
[11]:
```
<matplotlib.legend.Legend at 0x7f91614409b0>
```



Since `sepal_length` is a numeric (i.e. continuous) variable, seaborn automatically assigns a sequential color palette rather than a qualitative or categorical one. Same as before, we can customize the color palette to suit our preferences.

[12]:
```
sns.scatterplot('sepal_length', 'sepal_width', data=iris, hue='sepal_length',␣
↪palette="Blues")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

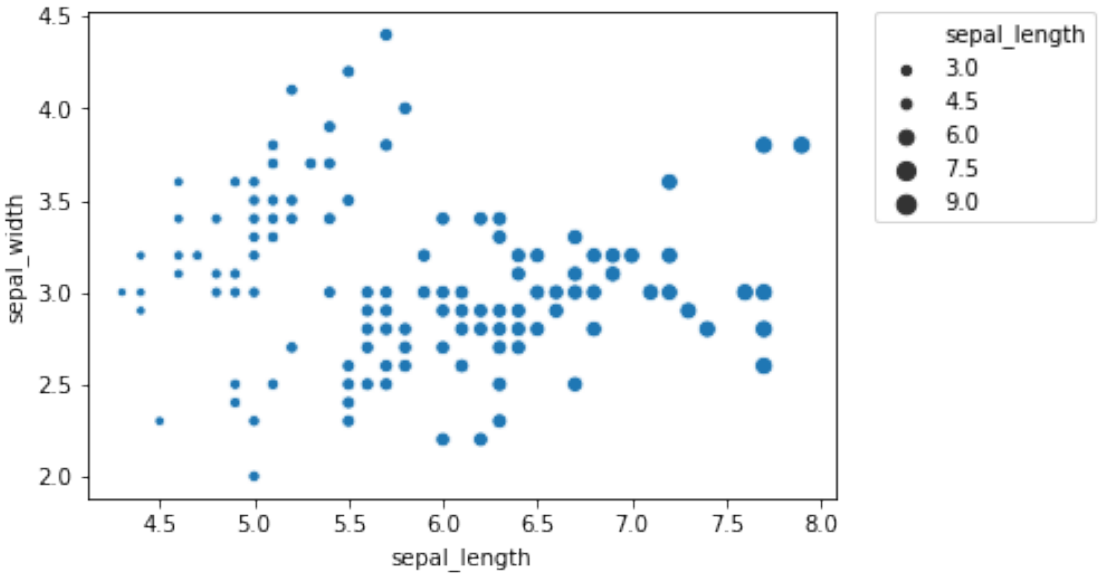
[12]: `<matplotlib.legend.Legend at 0x7f9161387128>`



Another useful way to illustrate a continuous variable in our datapoints is using the `size` parameter.

[13]:
```
sns.scatterplot('sepal_length', 'sepal_width', data=iris, size='sepal_length')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

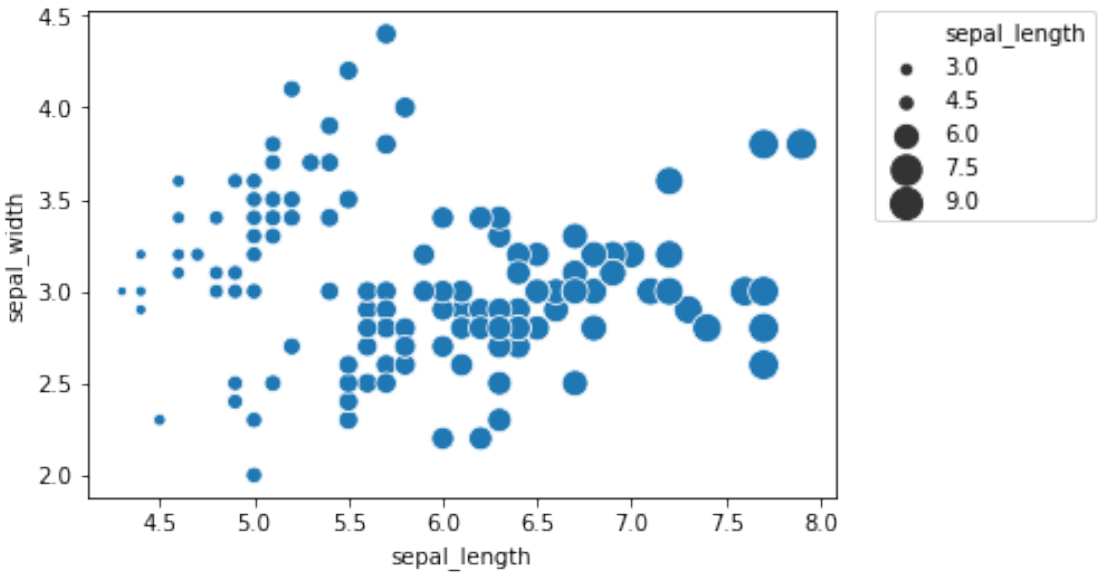
[13]:
```
<matplotlib.legend.Legend at 0x7f916135cba8>
```



We can adjust this range using the `sizes` parameter in addition to `size`

[14]:
```
sns.scatterplot('sepal_length', 'sepal_width', data=iris, size='sepal_length',␣
↪sizes = (20, 200))
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

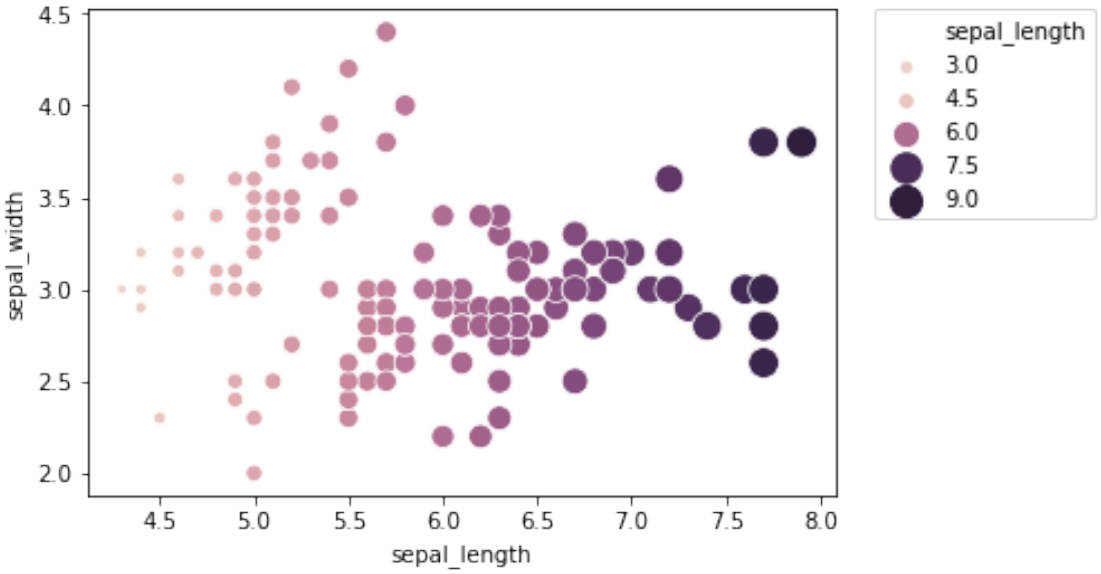
[14]:
```
<matplotlib.legend.Legend at 0x7f91612e69b0>
```



Again, the `size` parameter can be used either on its own or in conjunction with the `hue` parameter.

[15]:
```
sns.scatterplot('sepal_length', 'sepal_width', data=iris, hue='sepal_length',
                size='sepal_length', sizes = (20, 200))
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

[15]: `<matplotlib.legend.Legend at 0x7f9161204438>`

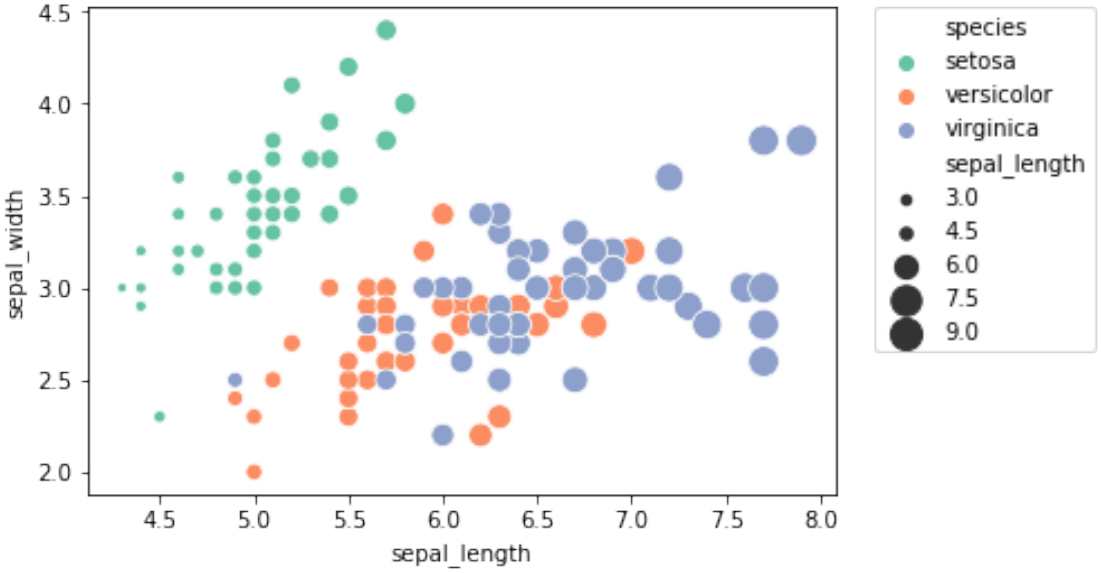


Furthermore, we can mix and match illustrating a categorical variabel with `hue` and a continuous

variable with `size`.

[16]:
```
sns.scatterplot('sepal_length', 'sepal_width', data=iris, hue = 'species',
                size = "sepal_length", sizes = (20, 200), palette = "Set2")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

[16]:
```
<matplotlib.legend.Legend at 0x7f916118b3c8>
```



1.3 Coloring with other plot types

We can similarly use the `hue` parameter with other plot types we've learned about as well. You will practice this in the next few exercises.