Simple tips for prettier and customised plots in Seaborn (Python)

In this tutorial, we will look at simple ways to customise your plots to make them aesthetically more pleasing. I hope these simple tricks will help you get better-looking plots and save you time from adjusting individual plots.

Baseline plot

Let's use Seaborn's built-in dataset on penguins as our sample data:

```
# Import packages
import matplotlib.pyplot as plt
import seaborn as sns

# Import data
df = sns.load_dataset('penguins').rename(columns={'sex': 'gender'})
df
```

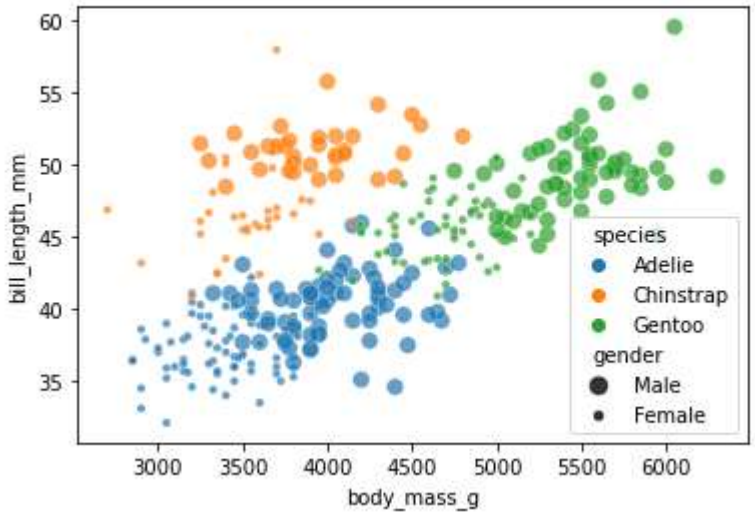
	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	gender
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	Male
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	Female
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	Female
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	Female
...	...	...	...	...	...	...	...
339	Gentoo	Biscoe	NaN	NaN	NaN	NaN	NaN
340	Gentoo	Biscoe	46.8	14.3	215.0	4850.0	Female
341	Gentoo	Biscoe	50.4	15.7	222.0	5750.0	Male
342	Gentoo	Biscoe	45.2	14.8	212.0	5200.0	Female
343	Gentoo	Biscoe	49.9	16.1	213.0	5400.0	Male

344 rows × 7 columns

We will build a standard scatter plot with the default chart settings to use it as our baseline:

```
# Plot
sns.scatterplot(data=df, x='body_mass_g', y='bill_length_mm', alpha=0.7, hue='species',
size='gender')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2017845cdc0>
```



We will see the journey of how this plot alters with each tip.

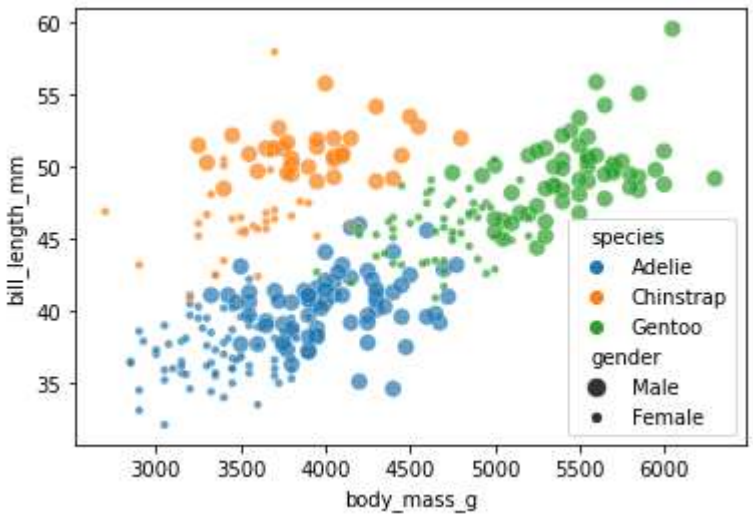
Tips

You will see that the first two tips are for individual plots whereas the remaining four tips are for changing the default settings for all charts.

Tip #1: Semicolon

Did you notice the text output right above the chart in the previous plot? A simple way to suppress this text output is to use ; at the end of your plot.

```
# Plot
sns.scatterplot(data=df, x='body_mass_g', y='bill_length_mm', alpha=0.7, hue='species', size='gender');
```

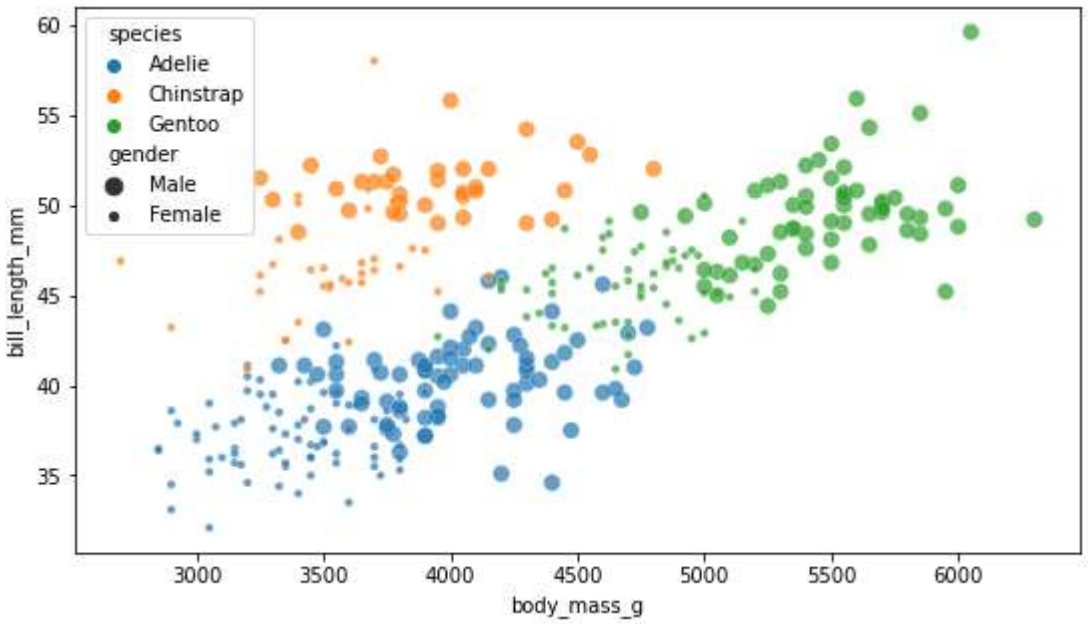


By only adding ; at the end of our code, we get a cleaner output.

Tip #2: plt.figure()

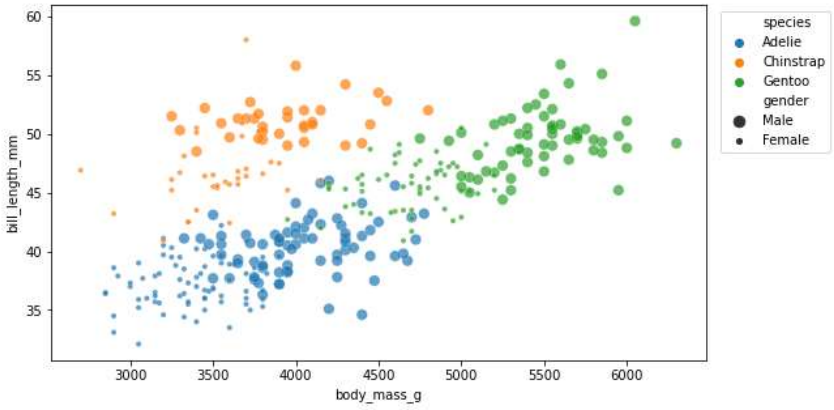
Plots can often benefit from resizing. If we wanted to resize, this is how we would do it:

```
# Plot
plt.figure(figsize=(9, 5))
sns.scatterplot(data=df, x='body_mass_g', y='bill_length_mm', alpha=0.7, hue='species',
size='gender');
```



When we resized, the legend moved to the upper left corner. Let's move the legend outside the chart so that it doesn't accidentally cover data points:

```
# Plot
plt.figure(figsize=(9, 5))
sns.scatterplot(data=df, x='body_mass_g', y='bill_length_mm', alpha=0.7, hue='species',
size='gender')
plt.legend(loc='upper right', bbox_to_anchor=(1.2, 1));
```



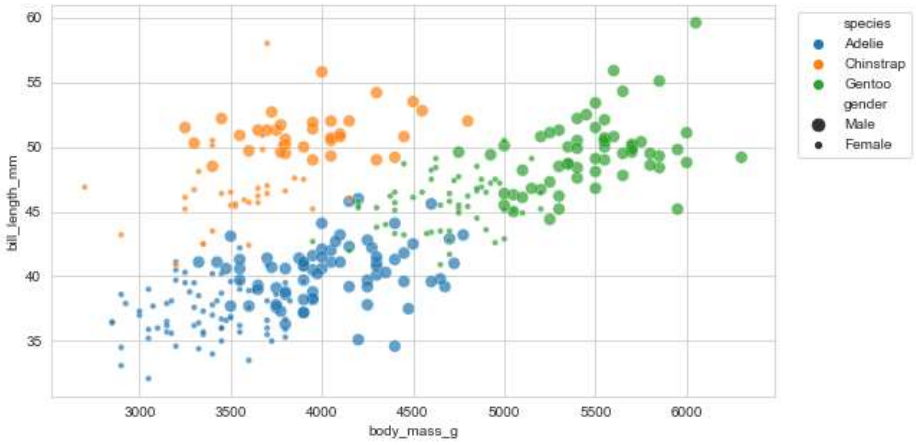
If you are wondering how to know what number combinations to use for figsize() or bbox_to_anchor(), you will need to trial and error which numbers work best for the plot.

Tip #3: sns.set_style()

This function helps to change the overall style of the plots if we don't like the default style. This includes things like the aesthetics of the axis colours and background. Let's change the style to *whitegrid* and see how the plot appearance changes:

```
# Change default style
sns.set_style('whitegrid')

# Plot
plt.figure(figsize=(9, 5))
sns.scatterplot(data=df, x='body_mass_g', y='bill_length_mm', alpha=0.7, hue='species',
size='gender')
plt.legend(loc='upper right', bbox_to_anchor=(1.2, 1));
```



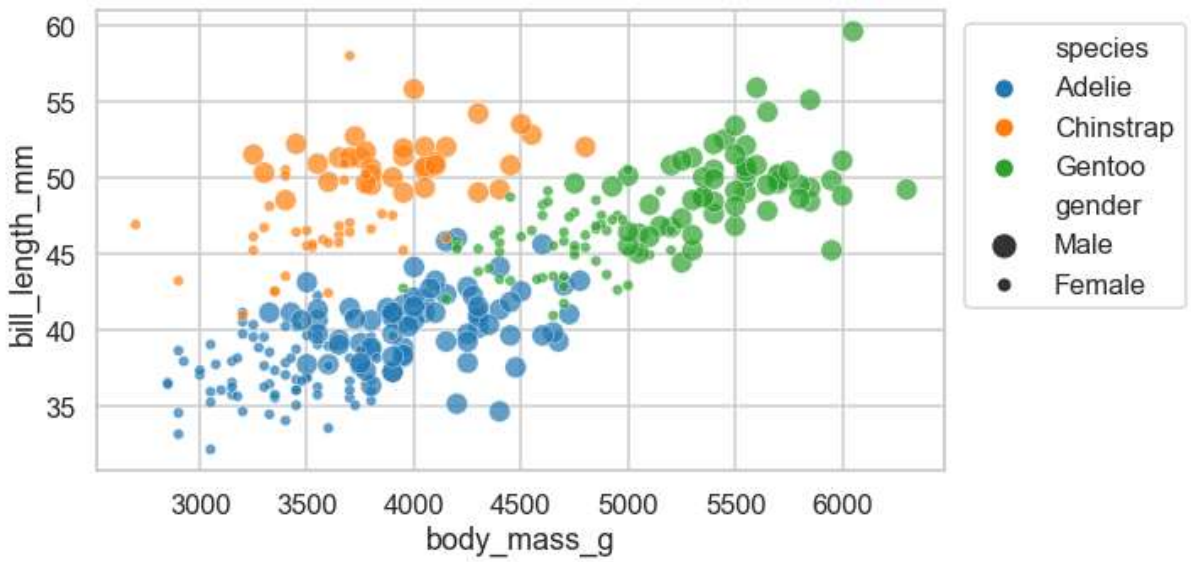
Here are some more other options to try out: 'darkgrid', 'dark' and 'ticks' to find the one you fancy more.

Tip #4: sns.set_context()

The label sizes look quite small in the previous plot. With sns.set_context(), we could change the context parameters if we don't like the default settings. I use this function mainly to control the default font size for labels in the plots. By changing the default, we can save time from not having to tweak the font size for different elements (e.g. axis label, title, legend) of individual plots. Let's change the context to 'talk' and look at the plot again:

```
# Change default context
sns.set_context('talk')

# Plot
plt.figure(figsize=(9, 5))
sns.scatterplot(data=df, x='body_mass_g', y='bill_length_mm', alpha=0.7, hue='species',
size='gender')
plt.legend(loc='upper right', bbox_to_anchor=(1.3, 1));
```

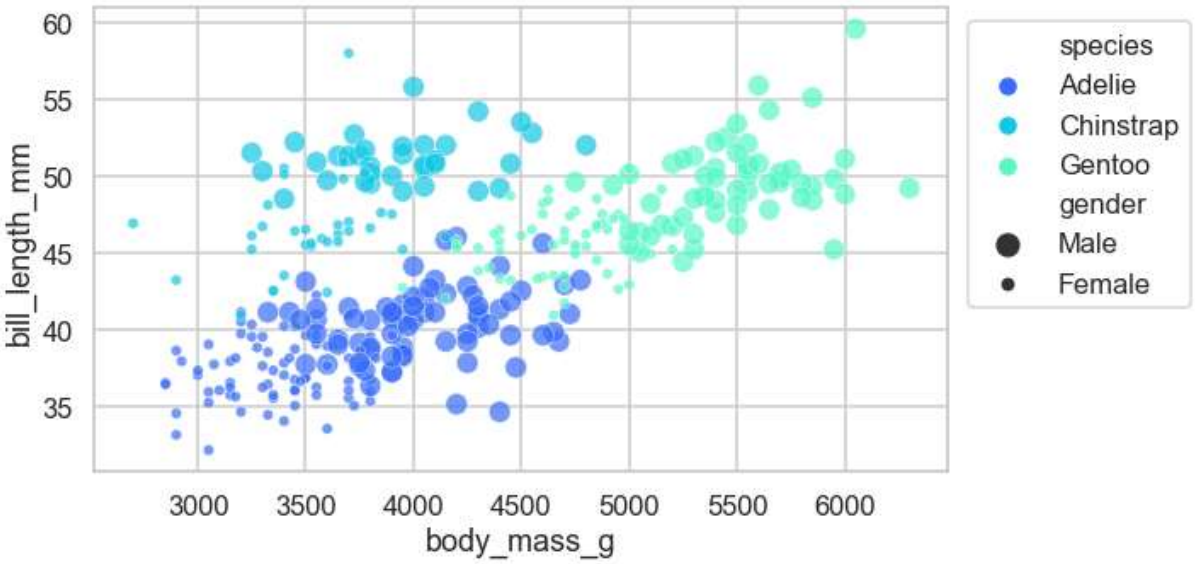


It's more easily legible, isn't it? Another option to try out is: 'poster' which will increase the default size even more.

Tip #5: sns.set_palette()

If you ever want to customise the default colour palette to your preferred colour combinations, this function comes in handy. We can use colourmaps from Matplotlib. Let's change the palette to 'rainbow' and look at the plot again:

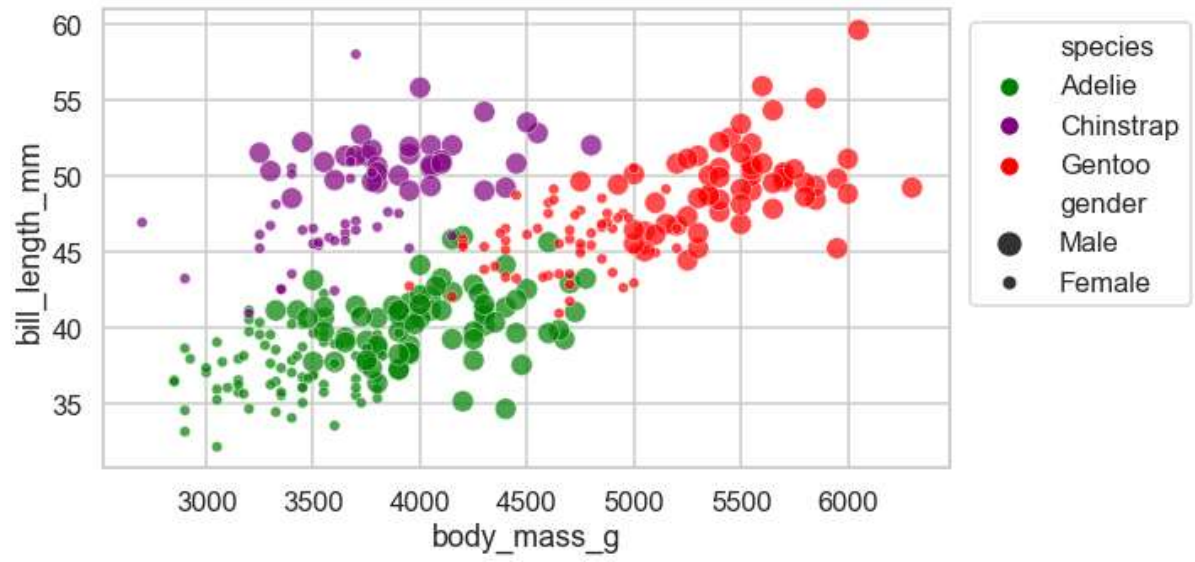
```
# Change default palette
sns.set_palette('rainbow')# Plot
plt.figure(figsize=(9, 5))
sns.scatterplot(data=df, x='body_mass_g', y='bill_length_mm', alpha=0.7, hue='species',
size='gender')
plt.legend(loc='upper right', bbox_to_anchor=(1.3, 1));
```



If you can't find a Matplotlib colourmap that you like, you can hand pick colours to create your own unique colour palette. One way to create your own palette is to pass a list of colour names to the function like in the example below.

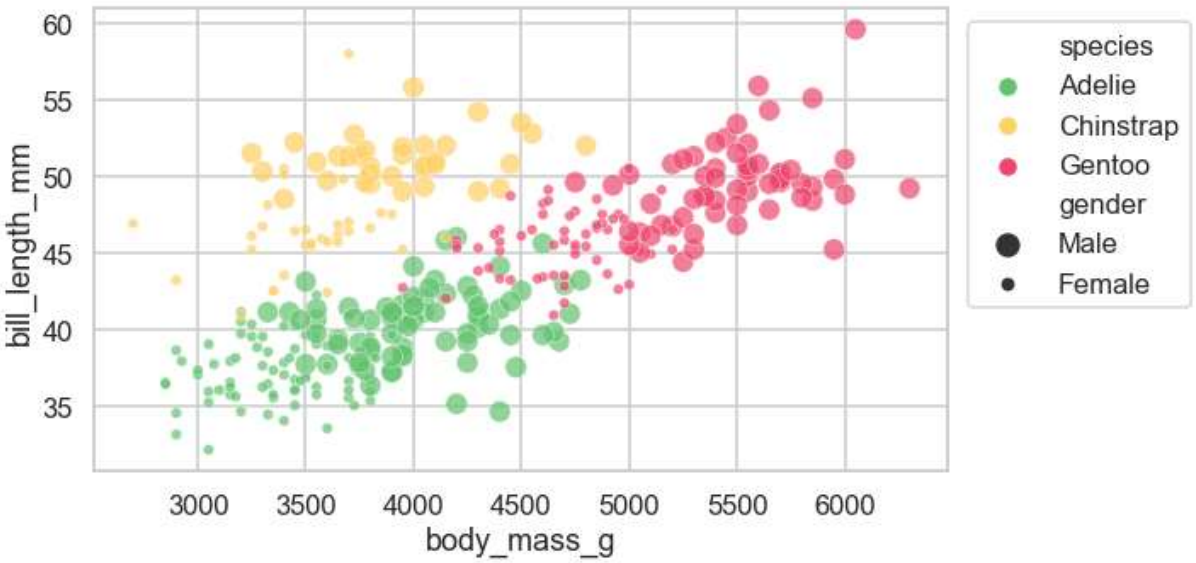
```
# Change default palette
sns.set_palette(['green', 'purple', 'red'])

# Plot
plt.figure(figsize=(9, 5))
sns.scatterplot(data=df, x='body_mass_g', y='bill_length_mm', alpha=0.7, hue='species',
size='gender')
plt.legend(loc='upper right', bbox_to_anchor=(1.3, 1));
```



If the colour names don't quite capture what you are after, you can build your own palette using hexadecimal colours to access a wider range of options (over 16 million colours!). Let's see an example:

```
# Change default palette
sns.set_palette(['#62C370', '#FFD166', '#EF476F'])# Plot
plt.figure(figsize=(9, 5))
sns.scatterplot(data=df, x='body_mass_g', y='bill_length_mm', alpha=0.7, hue='species',
size='gender')
plt.legend(loc='upper right', bbox_to_anchor=(1.3, 1));
```



Tip #6: sns.set()

From the previous three tips, I hope you will find your favourite combination (in some cases, it could be leaving the default as is). If we were to update chart default settings, it's better to do it just after importing the visualisation packages. This means we will have a snippet like this at the beginning of our scripts:

```
# Import packages
import matplotlib.pyplot as plt
import seaborn as sns# Change defaults
sns.set_style('whitegrid')
sns.set_context('talk')
sns.set_palette('rainbow')
```

Updating multiple defaults like above can be done more succinctly with sns.set(). Here's the succinct version of the same code:

```
# Import packages
import matplotlib.pyplot as plt
import seaborn as sns# Change defaults
sns.set(style='whitegrid', context='talk', palette='rainbow')
```

These were the six tips. These are the comparison of the plots before and after tweaking: