Seaborn Workshop

Controlling figure aesthetics

- Drawing attractive figures is important. When making figures for yourself, as you explore a dataset, its nice to have plots that are pleasant to look at.
- Visualizations are also central to communicating quantitative insights to an audience, and in that setting its even more necessary to have figures that catch the attention and draw a viewer in.

Seaborn Workshop

Matplotlib is highly customizable, but it can be hard to know what settings to tweak to achieve an attractive plot. Seaborn comes with a number of customized themes and a high-level interface for controlling the look of matplotlib figures.

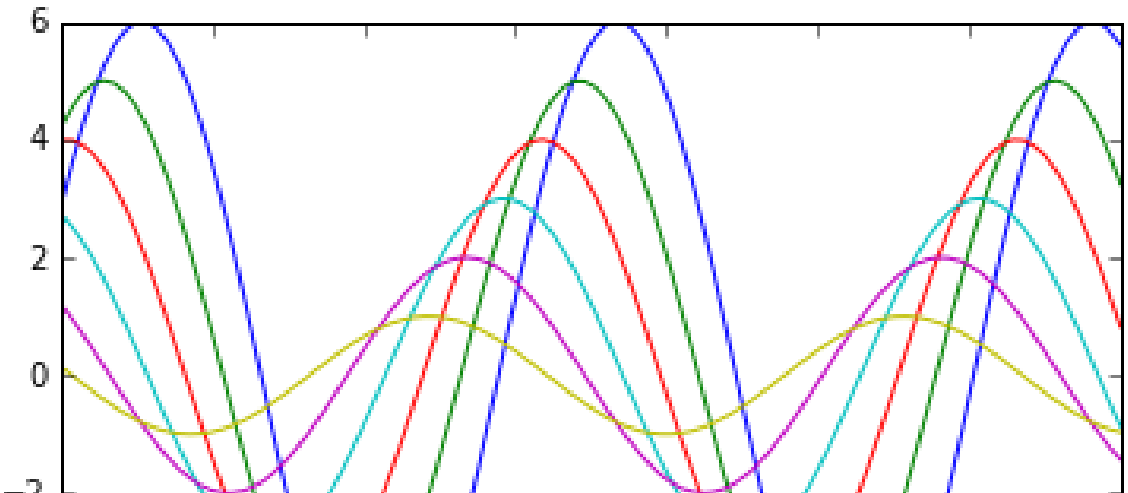
```
%matplotlib inline
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
np.random.seed(sum(map(ord, "aesthetics")))
```

Seaborn Workshop

Lets define a simple function to plot some offset sine waves, which will help us see the different stylistic parameters we can tweak.

```
def sinplot(flip=1):
    x = np.linspace(0, 14, 100)
    for i in range(1, 7):
        plt.plot(x, np.sin(x + i * .5) * (7 - i) * flip
```

This is what the plot looks like with matplotlib defaults:
sinplot()



Seaborn Workshop

To switch to seaborn defaults, simply import the package.

Seaborn Workshop

```
import seaborn as sns
sinplot()
```

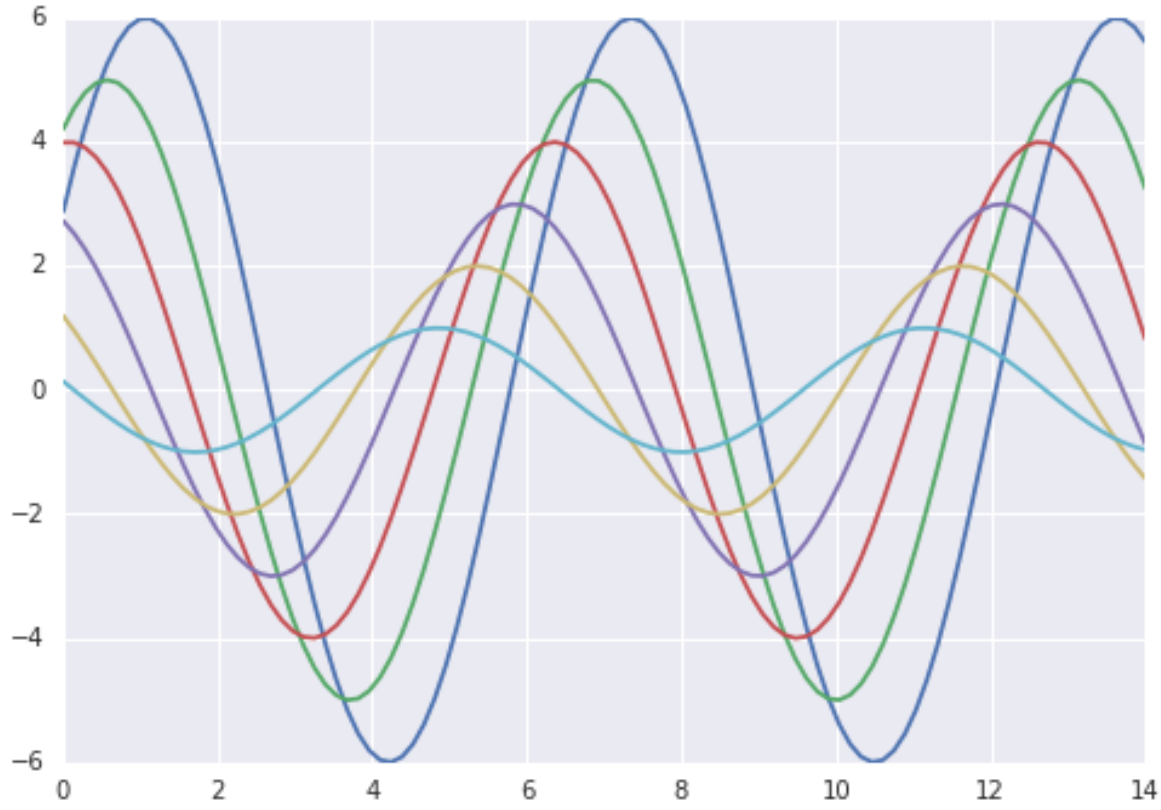


Figure:

Seaborn Workshop

- The seaborn defaults break from the MATLAB inspired aesthetic of matplotlib to plot in more muted colors over a light gray background with white grid lines.
- We find that the grid aids in the use of figures for conveying quantitative information in almost all cases, figures should be preferred to tables.
- The white-on-gray grid that is used by default avoids being obtrusive.
- The grid is particularly useful in giving structure to figures with multiple facets, which is central to some of the more complex tools in the library.

Seaborn Workshop

- Seaborn splits matplotlib parameters into two independent groups.
- The first group sets the aesthetic style of the plot, and the second scales various elements of the figure so that it can be easily incorporated into different contexts.

Seaborn Workshop

- The interface for manipulating these parameters are two pairs of functions.
- To control the style, use the `axes_style()` and `set_style()` functions.
- To scale the plot, use the `plotting_context()` and `set_context()` functions.
- In both cases, the first function returns a dictionary of parameters and the second sets the matplotlib defaults.

Styling figures with `axes_style()` and `set_style()`

- There are five preset seaborn themes: darkgrid, whitegrid, dark, white, and ticks. They are each suited to different applications and personal preferences. The default theme is darkgrid.
- As mentioned above, the grid helps the plot serve as a lookup table for quantitative information, and the white-on grey helps to keep the grid from competing with lines that represent data.
- The whitegrid theme is similar, but it is better suited to plots with heavy data elements:

Seaborn Workshop

```
sns.set_style("whitegrid")
data = np.random.normal(size=(20, 6)) + np.arange(6) /
sns.boxplot(data=data);
```

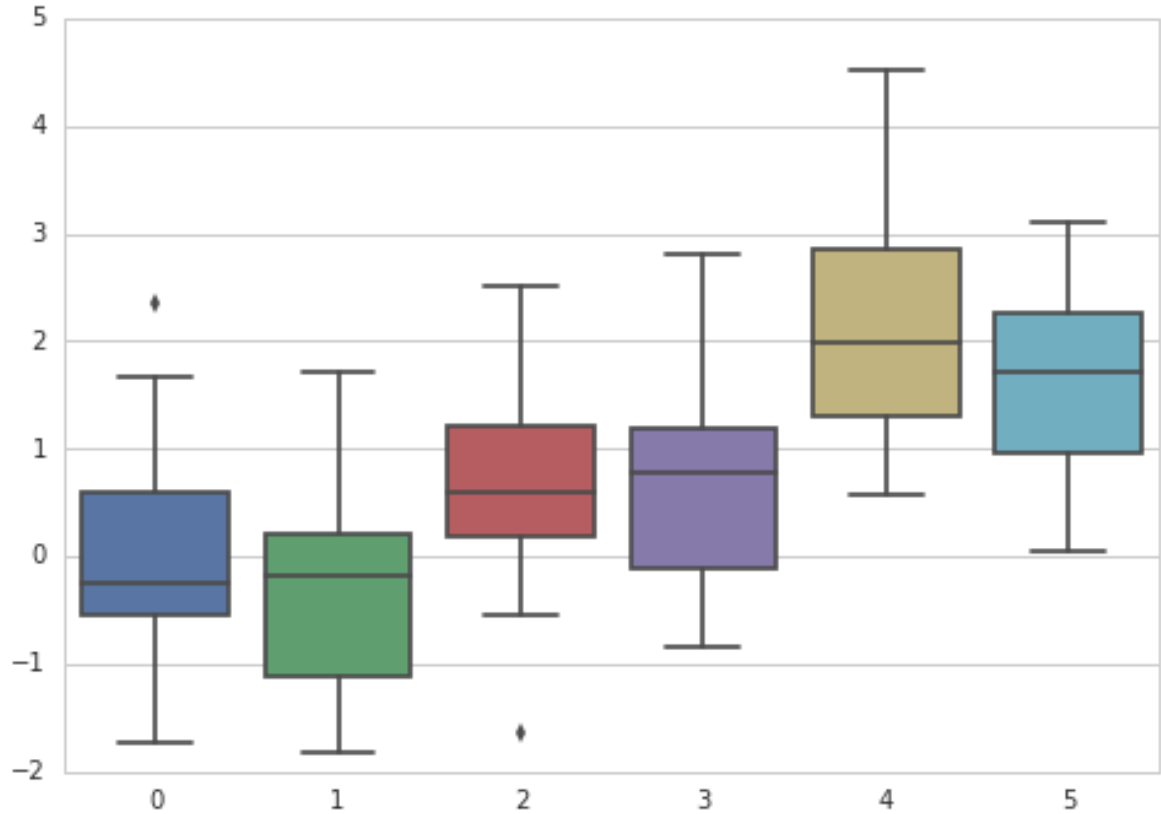


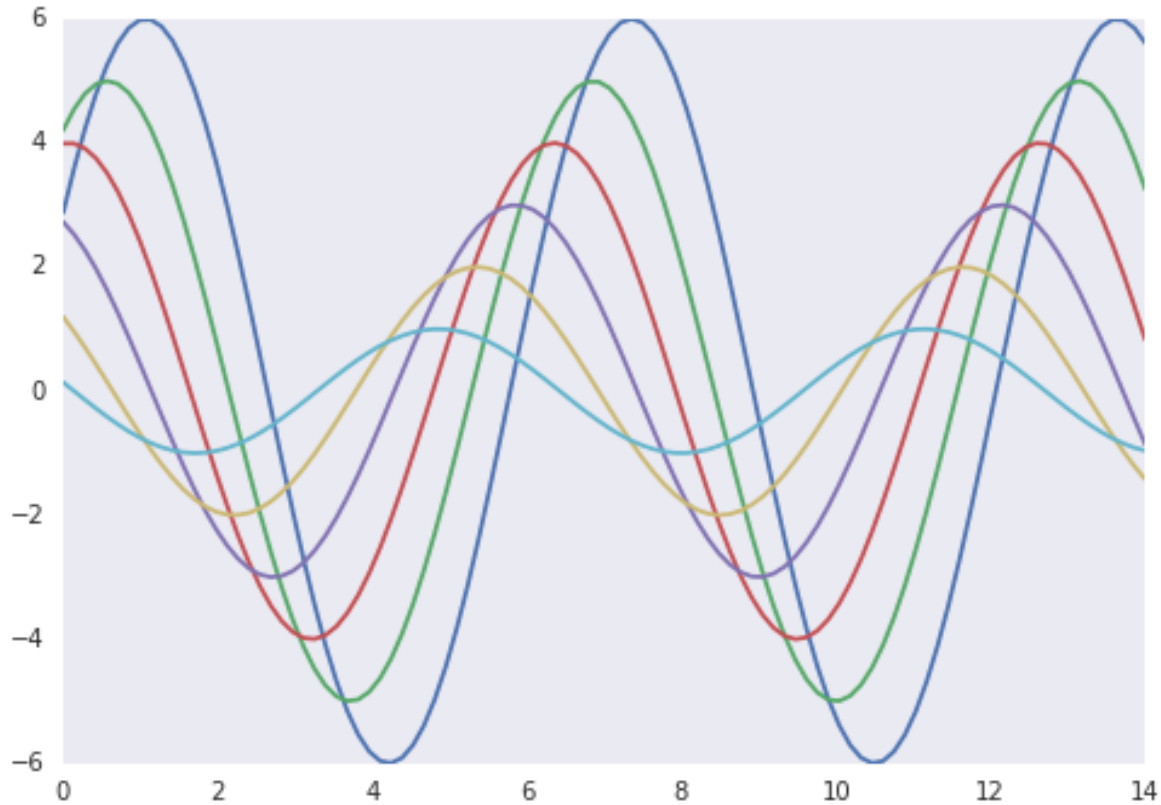
Figure:

Seaborn Workshop

For many plots, (especially for settings like talks, where you primarily want to use figures to provide impressions of patterns in the data), the grid is less necessary.

Seaborn Workshop

```
sns.set_style("dark")
sinplot()
```



Seaborn Workshop

```
sns.set_style("white")
sinplot()
```

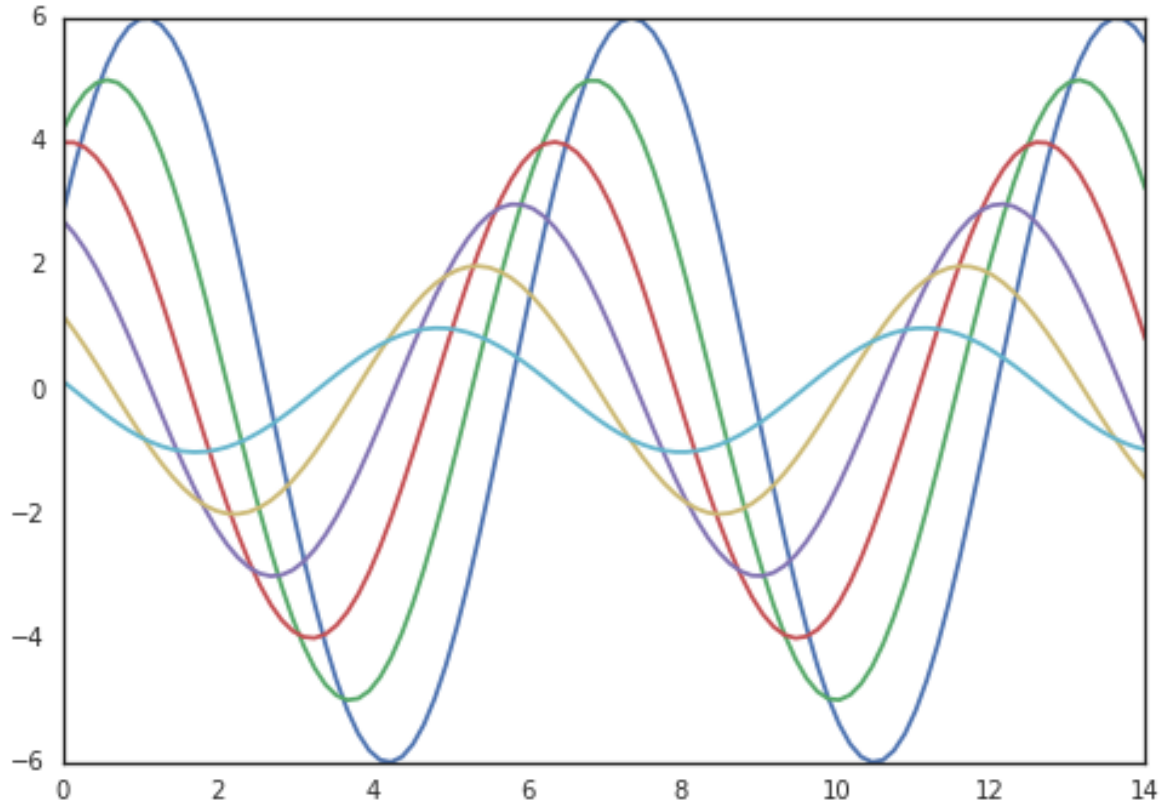


Figure:

Seaborn Workshop

Sometimes you might want to give a little extra structure to the plots, which is where ticks come in handy:

```
sns.set_style("ticks")
sinplot()
```

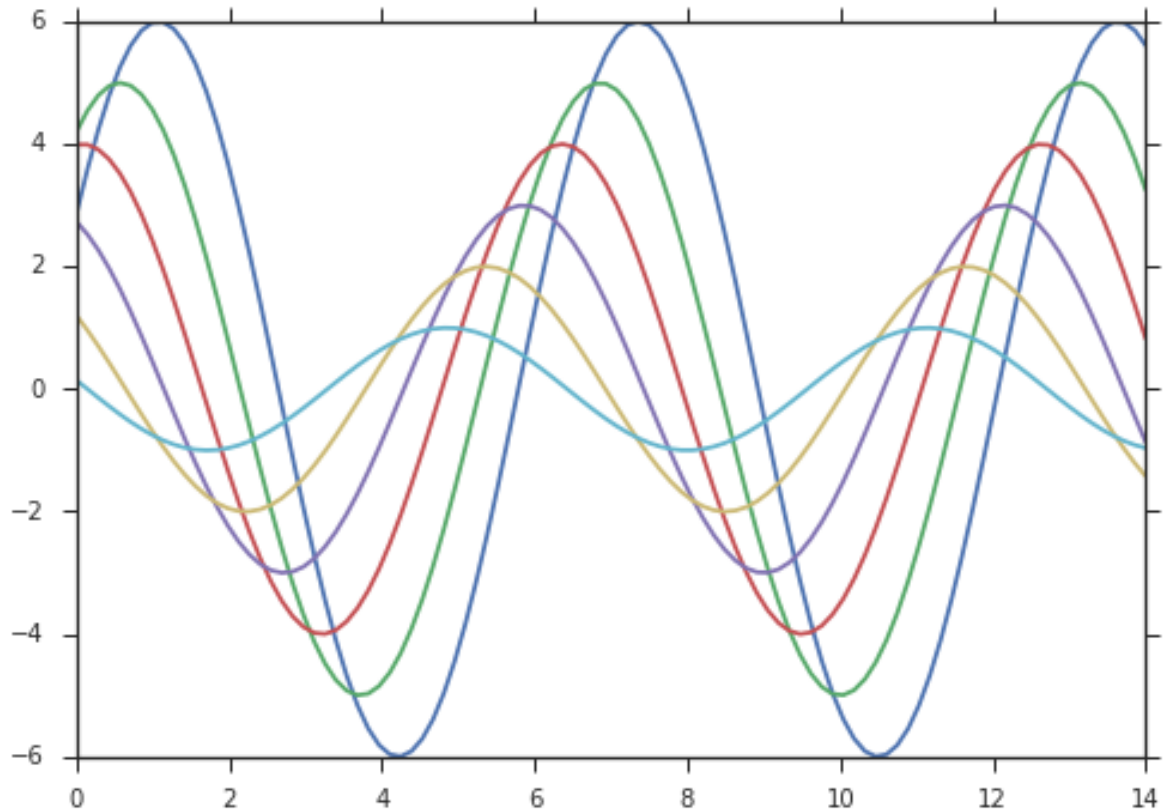


Figure:

Removing spines with despine()

- Both the white and ticks styles can benefit from removing the top and right axes spines, which are not needed.
- Its impossible to do this through the matplotlib parameters, but you can call the seaborn function `despine()` to remove them:

Seaborn Workshop

```
sinplot()
sns.despine()
```

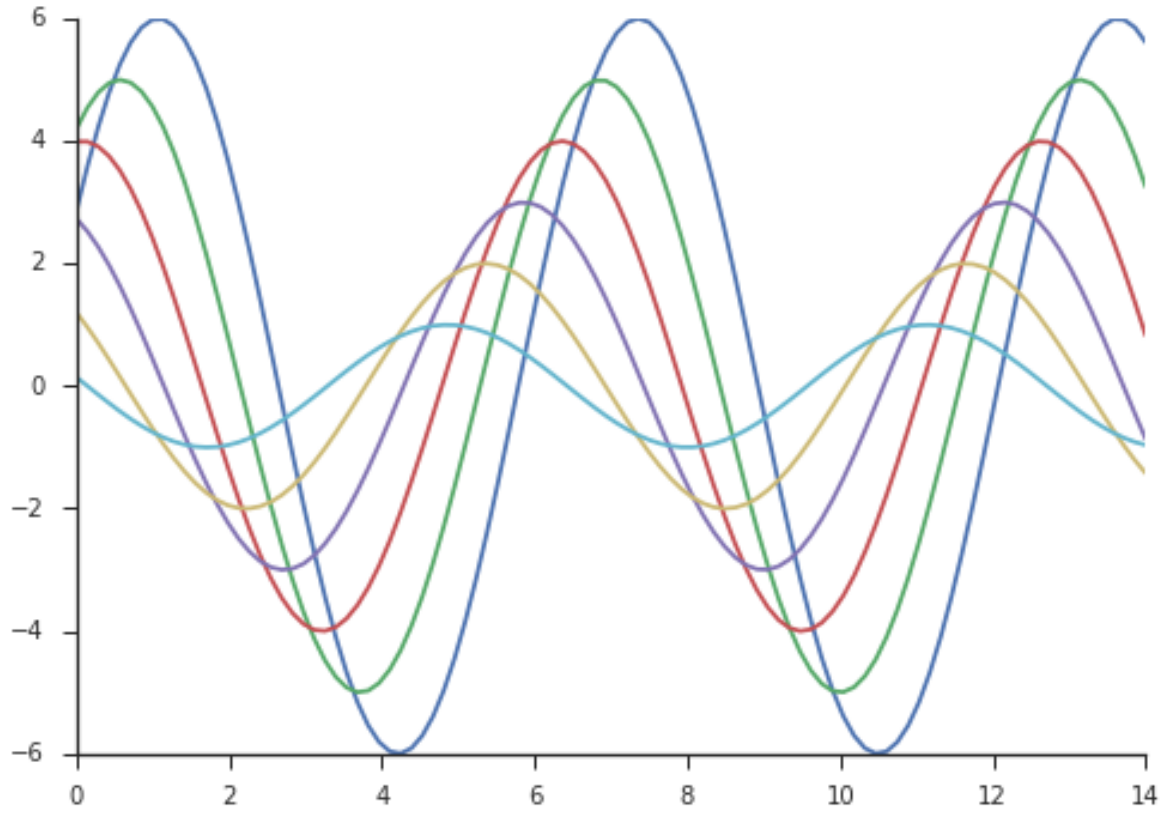


Figure: