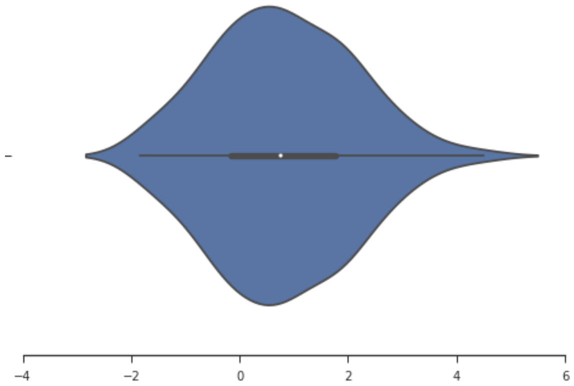
Seaborn Workshop

- Some plots benefit from offsetting the spines away from the data, which can also be done when calling `despine()`.
- When the ticks dont cover the whole range of the axis, the trim parameter will limit the range of the surviving spines.

Seaborn Workshop

```
f, ax = plt.subplots()
sns.violinplot(data)
sns.despine(offset=10, trim=True);
```



Seaborn Workshop

You can also control which spines are removed with additional arguments to despine():

```
sns.set_style("whitegrid")
sns.boxplot(data=data, palette="deep")
sns.despine(left=True)
```

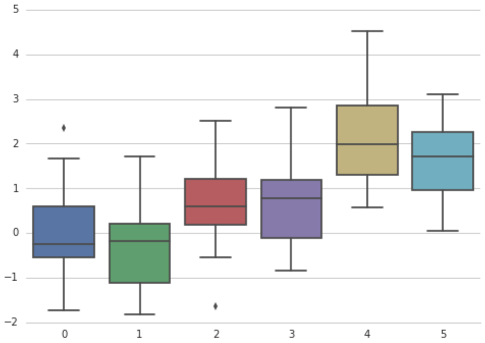


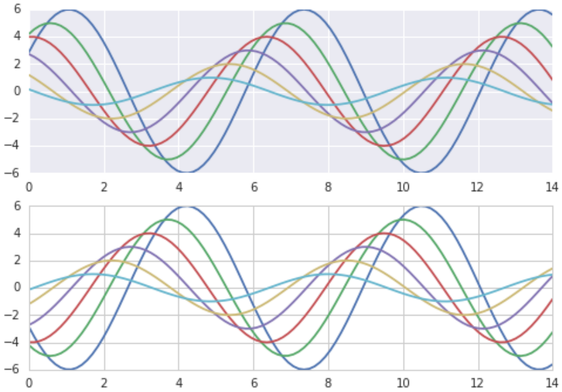
Figure:

Seaborn Workshop

- Although its easy to switch back and forth, you can also use the `axes_style()` function in a with statement to temporarily set plot parameters.
- This also allows you to make figures with differently-styled axes:

Seaborn Workshop

```
with sns.axes_style("darkgrid"):
plt.subplot(211)
sinplot()
plt.subplot(212)
sinplot(-1)
```



Seaborn Workshop

Overriding elements of the seaborn styles

- If you want to customize the seaborn styles, you can pass a dictionary of parameters to the `rc` argument of `axes_style()` and `set_style()`.

Seaborn Workshop

- Note that you can only override the parameters that are part of the style definition through this method. (However, the higher-level `set()` function takes a dictionary of any matplotlib parameters).
- If you want to see what parameters are included, you can just call the function with no arguments, which will return the current settings:

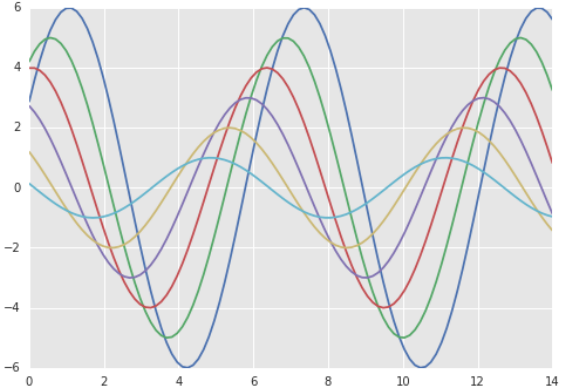
Seaborn Workshop

```
sns.axes_style()
{'axes.axisbelow': True,
'axes.edgecolor': '.8',
'axes.facecolor': 'white',
'axes.grid': True,
'axes.labelcolor': '.15',
'axes.linewidth': 1.0,
'figure.facecolor': 'white',
'font.family': [u'sans-serif'],
'font.sans-serif': [u'Arial',
...
...
```

Seaborn Workshop

You can then set different versions of these parameters:

```
sns.set_style("darkgrid", {"axes.facecolor": ".9"})
sinplot()
```



Seaborn Workshop

- Scaling plot elements with `plotting_context()` and `set_context()`
- A separate set of parameters control the scale of plot elements, which should let you use the same code to make plots that are suited for use in settings where larger or smaller plots are appropriate.

Seaborn Workshop

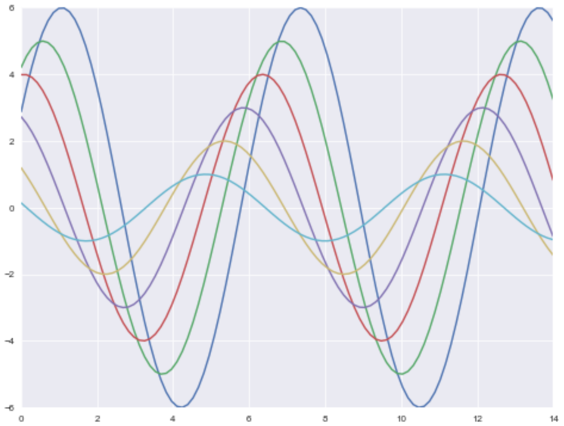
First lets reset the default parameters by calling `set()`:

```
sns.set()
```

- The four preset contexts, in order of relative size, are paper, notebook, talk, and poster.
- The notebook style is the default, and was used in the plots above.

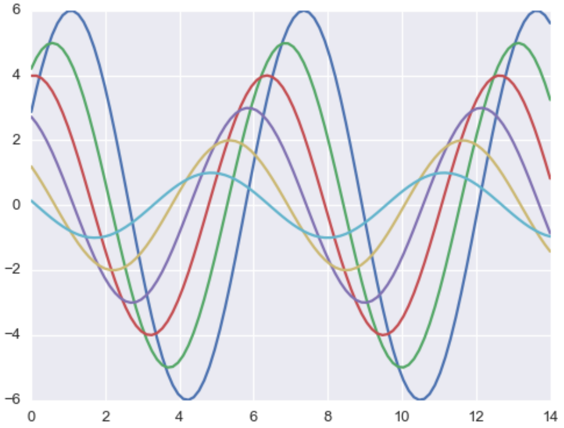
Seaborn Workshop

```
sns.set_context("paper")
plt.figure(figsize=(8, 6))
sinplot()
```



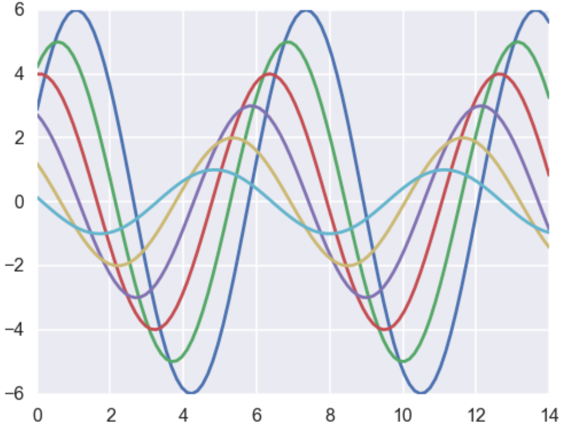
Seaborn Workshop

```
sns.set_context("talk")
plt.figure(figsize=(8, 6))
sinplot()
```



Seaborn Workshop

```
sns.set_context("poster")
plt.figure(figsize=(8, 6))
sinplot()
```



Seaborn Workshop

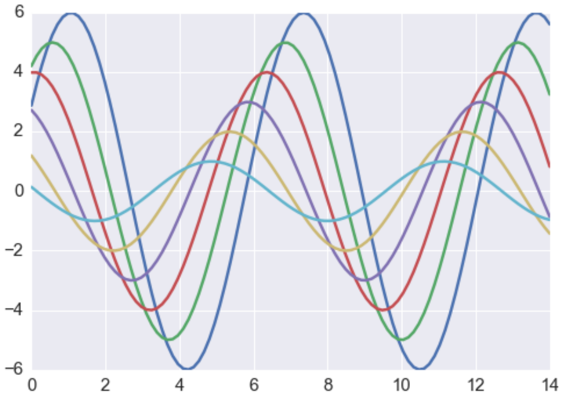
- Most of what you now know about the style functions should transfer to the context functions.
- You can call `set_context()` with one of these names to set the parameters, and you can override the parameters by providing a dictionary of parameter values.

Seaborn Workshop

- You can also independently scale the size of the font elements when changing the context.
- This option is also available through the top-level `set()` function.

Seaborn Workshop

```
sns.set_context("notebook", font_scale=1.5, rc={"lines
sinplot()
```



Similarly (although it might be less useful), you can temporarily control the scale of figures nested under a with statement.