Plotting with seaborn using the matplotlib object-oriented interface

I strongly prefer using `matplotlib` in OOP style:

```
f, axarr = plt.subplots(2, sharex=True)
axarr[0].plot(...)
axarr[1].plot(...)
```

This makes it easier to keep track of multiple figures and subplots.

Question: How to use seaborn this way? Or, how to change this example to OOP style? How to tell `seaborn` plotting functions like `lmplot` which `Figure` or `Axes` it plots to?

python oop matplotlib seaborn

edited Jun 1 '14 at 16:40
mwaskom
8,122 2 19 33

asked May 31 '14 at 11:39
Frozen Flame
518 3 17

1 Answer

It depends a bit on which seaborn function you are using.

The plotting functions in seaborn are broadly divided into two classes

- "Axes-level" functions, including `regplot`, `boxplot`, `kdeplot`, and many others
- "Figure-level" functions, including `lmplot`, `factorplot`, `jointplot` and one or two others

The first group is identified by taking an explicit `ax` argument and returning an `Axes` object. As this suggests, you can use them in an "object oriented" style by passing your `Axes` to them:

```
f, (ax1, ax2) = plt.subplots(2)
sns.regplot(x, y, ax=ax1)
sns.kdeplot(x, ax=ax2)
```

Axes-level functions will only draw onto an `Axes` and won't otherwise mess with the figure, so they can coexist perfectly happily in an object-oriented matplotlib script.

The second group of functions (Figure-level) are distinguished by the fact that the resulting plot can potentially include several Axes which are always organized in a "meaningful" way. That means that the functions need to have total control over the figure, so it isn't possible to plot, say, an `lmplot` onto one that already exists. Calling the function always initializes a figure and sets it up for the specific plot it's drawing.

However, once you've called `lmplot`, it will return an object of the type `FacetGrid`. This object has some methods for operating on the resulting plot that know a bit about the structure of the plot. It also exposes the underlying figure and array of axes at the `FacetGrid.fig` and `FacetGrid.axes` arguments. The `jointplot` function is very similar, but it uses a `JointGrid` object. So you can still use these functions in an object-oriented context, but all of your customization has to come after you've called the function.

edited Apr 10 '15 at 14:51

answered May 31 '14 at 18:22
mwaskom
8,122 2 19 33