### **Introduction**

When creating a data visualization, your goal is to communicate the insights found in the data. While visualizing communicates important information, styling will influence how your audience understands what you're trying to convey.

After you have formatted and visualized your data, the third and last step of data visualization is styling. Styling is the process of customizing the overall look of your visualization, or figure. Making intentional decisions about the details of the visualization will increase their impact and set your work apart.

In this article, we'll look at how to do the following techniques in Seaborn:

* customize the overall look of your figure, using background colors, grids, spines, and ticks
* scale plots for different contexts, such as presentations and reports

### **Customizing the Overall Look of Your Figure**

Seaborn enables you to change the presentation of your figures by changing the style of elements like the background color, grids, and spines. When deciding how to style your figures, you should take into consideration your audience and the context. Is your visualization part of a report and needs to convey specific information? Or is it part of a presentation? Or is your visualization meant as its own stand-alone, with no narrator in front of it, and no other visualizations to compare it to?

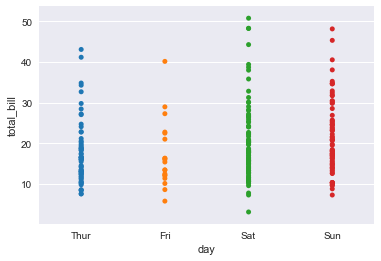
In this section, we'll explore three main aspects of customizing figures in Seaborn - background color, grids, and spines - and how these elements can change the look and meaning of your visualizations.

#### **Built-in Themes**

Seaborn has five built-in themes to style its plots: darkgrid, whitegrid, dark, white, and ticks. Seaborn defaults to using the darkgrid theme for its plots, but you can change this styling to better suit your presentation needs.

To use any of the preset themes pass the name of it to sns.set\_style().

sns.set\_style("darkgrid")  
sns.stripplot(x="day", y="total\_bill", data=tips)



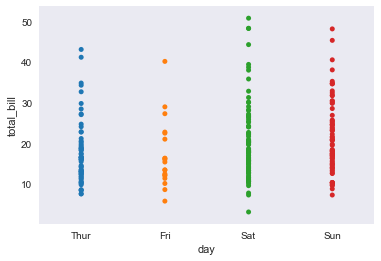
We'll explore the rest of the themes in the examples below.

### **Background Color**

When thinking about the look of your visualization, one thing to consider is the background color of your plot. The higher the contrast between the color palette of your plot and your figure background, the more legible your data visualization will be. Fun fact: dark blue on white is actually more legible than black on white!

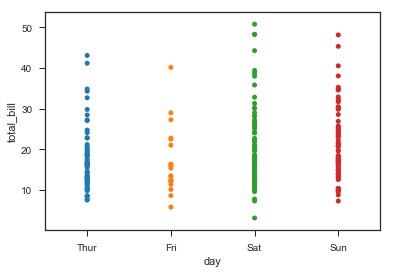
The dark background themes provide a nice change from the Matplotlib styling norms, but doesn't have as much contrast:

sns.set\_style("dark")  
sns.stripplot(x="day", y="total\_bill", data=tips)



The white and tick themes will allow the colors of your dataset to show more visibly and provides higher contrast so your plots are more legible:

sns.set\_style("ticks")  
sns.stripplot(x="day", y="total\_bill", data=tips)



#### 

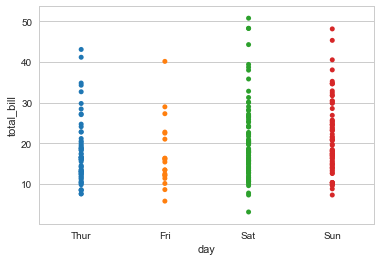
#### 

#### **Grids**

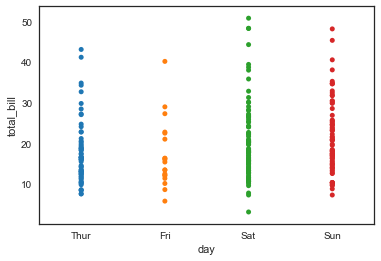
In addition to being able to define the background color of your figure, you can also choose whether or not to include a grid. Remember that the default theme includes a grid.

It's a good choice to use a grid when you want your audience to be able to draw their own conclusions about data. A grid allows the audience to read your chart and get specific information about certain values. Research papers and reports are a good example of when you would want to include a grid.

sns.set\_style("whitegrid")  
sns.stripplot(x="day", y="total\_bill", data=tips)



There are also instances where it would make more sense to not use a grid. If you're delivering a presentation, simplifying your charts in order to draw attention to the important visual details may mean taking out the grid. If you're interested in making more specific design choices, then leaving out the grids might be part of that aesthetic decision.

sns.set\_style("white")  
sns.stripplot(x="day", y="total\_bill", data=tips)

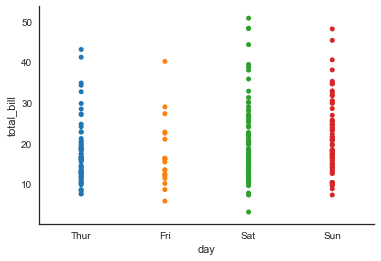
In this case, a blank background would allow your plot to shine.

#### **Despine**

In addition to changing the color background, you can also define the usage of *spines*. Spines are the borders of the figure that contain the visualization. By default, an image has four spines.

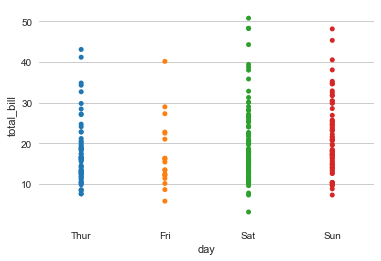
You may want to remove some or all of the spines for various reasons. A figure with the left and bottom spines resembles traditional graphs. You can automatically take away the top and right spines using the sns.despine()function. Note: this function must be called after you have called your plot.

sns.set\_style("white")  
sns.stripplot(x="day", y="total\_bill", data=tips)  
sns.despine()



Not including any spines at all may be an aesthetic decision. You can also specify how many spines you want to include by calling despine() and passing in the spines you want to get rid of, such as: left, bottom, top, right.

sns.set\_style("whitegrid")  
sns.stripplot(x="day", y="total\_bill", data=tips)  
sns.despine(left=True, bottom=True)



### **Scaling Figure Styles for Different Mediums**

Matplotlib allows you to generate powerful plots, but styling those plots for different presentation purposes is difficult. Seaborn makes it easy to produce the same plots in a variety of different visual formats so you can customize the presentation of your data for the appropriate context, whether it be a research paper or a conference poster.

You can set the visual format, or context, using sns.set\_context()

Within the usage of sns.set\_context(), there are three levels of complexity:

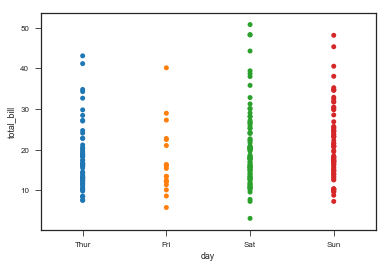
1. Pass in one parameter that adjusts the scale of the plot
2. Pass in two parameters - one for the scale and the other for the font size
3. Pass in three parameters - including the previous two, as well as the rc with the style parameter that you want to override

#### **Scaling Plots**

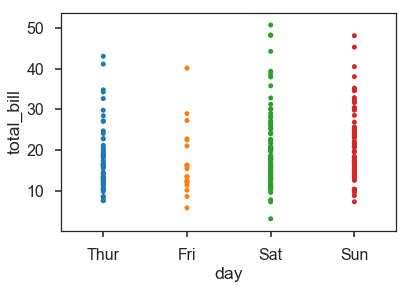
Seaborn has four presets which set the size of the plot and allow you to customize your figure depending on how it will be presented.

In order of relative size they are: paper, notebook, talk, and poster. The notebook style is the default.

sns.set\_style("ticks")  
  
# Smallest context:  
sns.set\_context("paper")  
sns.stripplot(x="day", y="total\_bill", data=tips)



sns.set\_style("ticks")  
  
# Largest Context:  
sns.set\_context("poster")  
sns.stripplot(x="day", y="total\_bill", data=tips)

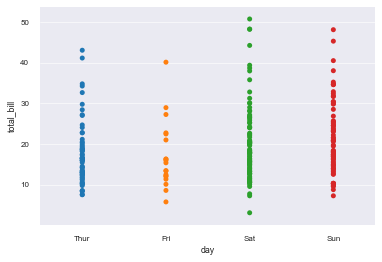


#### **Scaling Fonts and Line Widths**

You are also able to change the size of the text using the font\_scale parameter for sns.set\_context()

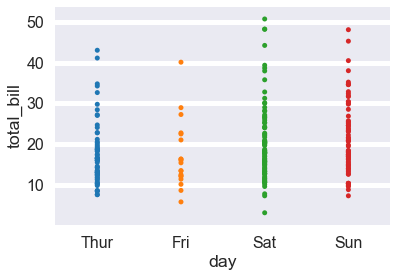
You may want to also change the line width so it matches. We do this with the rcparameter, which we'll explain in detail below.

# Set font scale and reduce grid line width to match  
sns.set\_context("poster", font\_scale = .5, rc={"grid.linewidth": 0.6})  
sns.stripplot(x="day", y="total\_bill", data=tips)



While you're able to change these parameters, you should keep in mind that it's not always useful to make certain changes. Notice in this example that we've changed the line width, but because of it's relative size to the plot, it distracts from the actual plotted data.

# Set font scale and increase grid line width to match  
sns.set\_context("poster", font\_scale = 1, rc={"grid.linewidth": 5})  
sns.stripplot(x="day", y="total\_bill", data=tips)



#### **The RC Parameter**

As we mentioned above, if you want to override any of these standards, you can use sns.set\_context and pass in the parameter rc to target and reset the value of an individual parameter in a dictionary. rc stands for the phrase 'run command' - essentially, configurations which will execute when you run your code.

sns.set\_style("ticks")  
sns.set\_context("poster")  
sns.stripplot(x="day", y="total\_bill", data=tips)  
sns.plotting\_context()

Returns:

{'axes.labelsize': 17.6,  
 'axes.titlesize': 19.200000000000003,  
 'font.size': 19.200000000000003,  
 'grid.linewidth': 1.6,  
 'legend.fontsize': 16.0,  
 'lines.linewidth': 2.8000000000000003,  
 'lines.markeredgewidth': 0.0,  
 'lines.markersize': 11.200000000000001,  
 'patch.linewidth': 0.48,  
 'xtick.labelsize': 16.0,  
 'xtick.major.pad': 11.200000000000001,  
 'xtick.major.width': 1.6,  
 'xtick.minor.width': 0.8,  
 'ytick.labelsize': 16.0,  
 'ytick.major.pad': 11.200000000000001,  
 'ytick.major.width': 1.6,  
 'ytick.minor.width': 0.8}

### **Conclusion**

As you can see, Seaborn offers a lot of opportunities to customize your plots and have them show a distinct style. The color of your background, background style such as lines and ticks, and the size of your font all play a role in improving legibility and aesthetics.