


 [learn-co-curriculum](#) / [dsc-seaborn-lab](#) Public [View license](#) 0 stars  112 forks Star Watch ▾[Code](#) [Issues](#) [Pull requests](#) 1 [Actions](#) [Projects](#) [Security](#) [Insights](#) solution ▾

...

This branch is [3 commits ahead](#), [5 commits behind](#) master. [Contribute](#) ▾

LoreDirick Auto-create curriculum branch from master + solution ...

on Oct 25, 2019

 4[View code](#) README.md

Seaborn - Lab

Introduction

In this lab, we'll get some practice working with a second, more advanced visualization library, *Seaborn*!

Objectives

You will be able to:

- Construct plots with Seaborn using its pre-built functionality

Getting Started

In this lab, we'll explore several different kinds of visualizations we can create with Seaborn. Seaborn is built on top of Matplotlib, so you'll find that it will feel quite familiar.

Let's get started by importing some things and creating a toy dataset to work with for our first visualization.

In the cell below:

- Import `numpy` and set the standard alias of `np`
- Import `seaborn` and set the standard alias of `sns`
- Set `%matplotlib inline` so that our visualizations appear in the notebook, and not as separate files

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Great! Now, run the cell below to create a sample dataset.

```
data = np.random.normal(size=(20, 10)) + np.arange(10) / 2
```

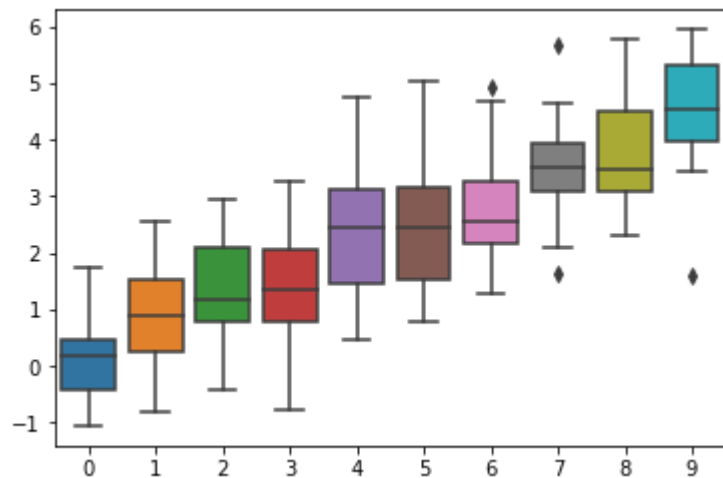
Basic Visualiations with Seaborn

We'll start off by creating a boxplot with the dataset we just created so that we can get a feel for the common workflow of Seaborn.

In the cell below:

- Create a `boxplot` and pass in the parameter `data=data` . Store the object returned in the variable `boxplot`

```
boxplot = sns.boxplot(data=data)
```



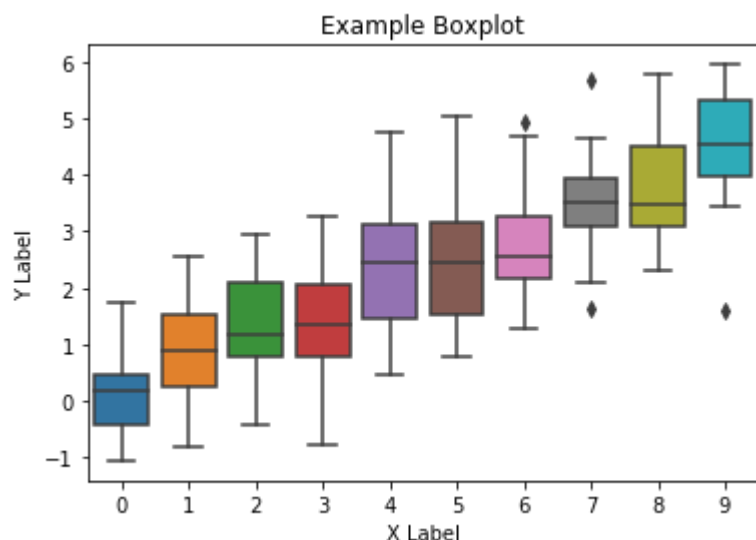
That's a nice looking visualization, for only a single line of code! However, it's missing axis labels and a title. Let's fix that.

In the cell below:

- Copy and paste the code from the cell above to recreate our boxplot
- Call the `boxplot` object's `set()` method and pass in the following parameters:
 - `xlabel= 'X Label'`
 - `ylabel= 'Y Label'`
 - `title = 'Example Boxplot'`

```
boxplot = sns.boxplot(data=data)
boxplot.set(xlabel = "X Label", ylabel='Y Label', title='Example Boxplot')
```

```
[Text(0, 0.5, 'Y Label'),
Text(0.5, 0, 'X Label'),
Text(0.5, 1.0, 'Example Boxplot')]
```



That wasn't too bad! Note that we can also use **Method Chaining** to set all the label and title information by combining the two lines in the cell above!

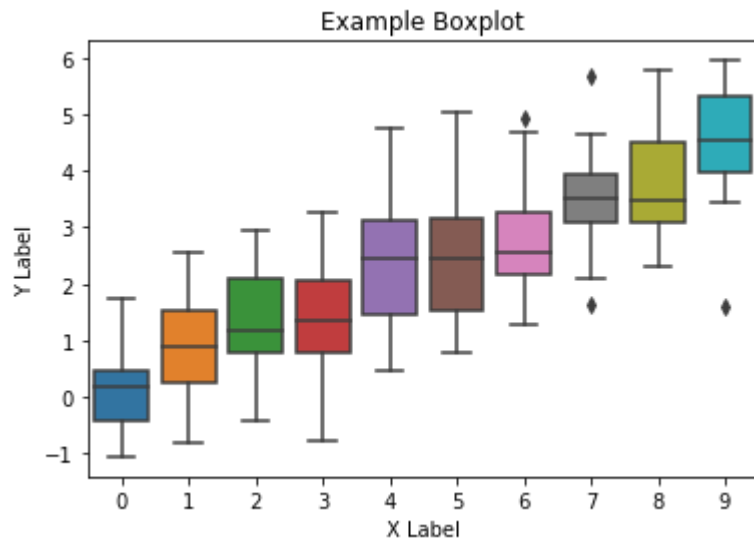
In the cell below:

- Recreate the labeled boxplot by calling `.set()` and passing in the appropriate parameter values immediately after calling `sns.boxplot(data=data)` to create the visualization.

NOTE: For this visualization, you do not need to store the object in a variable. Just call the methods.

```
sns.boxplot(data=data).set(xlabel = "X Label", ylabel='Y Label', title='Example Boxp
```

```
[Text(0, 0.5, 'Y Label'),
Text(0.5, 0, 'X Label'),
Text(0.5, 1.0, 'Example Boxplot')]
```



Great! As you can see, Seaborn is a pretty easy library to work with. It also has very detailed and easy-to-follow documentation, complete with a ton of examples and tutorials. If you're ever unsure of how to build something, don't be afraid to look at the [Seaborn Documentation](#), or Google!

Changing Style and Context

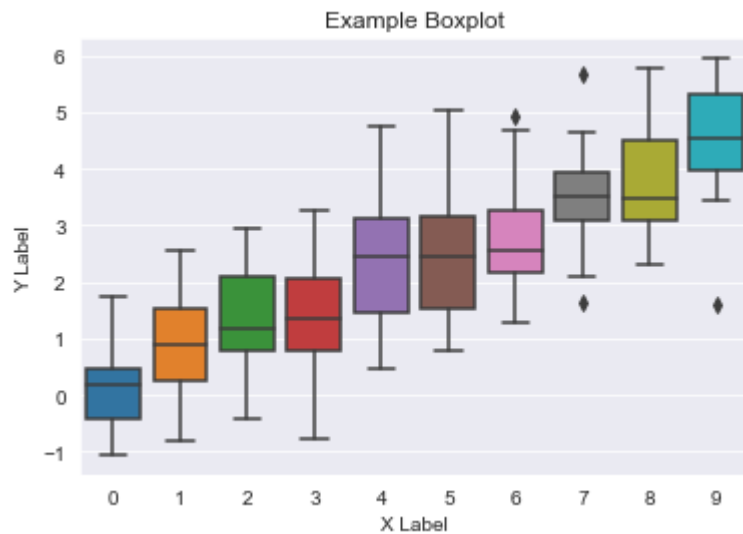
One of the main reasons Data Scientists love Seaborn is because the visualizations it creates are just plain prettier than those made by matplotlib. Seaborn makes it very simple to style our visualizations--all we need to do is use the `set_style()` method!

In the cell below:

- Call Seaborn's `set_style()` method and pass in the string `'darkgrid'`.
- Recreate the labeled boxplot that we made in the cell above.

```
sns.set_style('darkgrid')
sns.boxplot(data=data).set(xlabel = "X Label", ylabel='Y Label', title='Example Boxp
```

```
[Text(0, 0.5, 'Y Label'),
 Text(0.5, 0, 'X Label'),
 Text(0.5, 1.0, 'Example Boxplot')]
```



That's much easier to read! There are several different styles that we can choose from. To see examples of the different styles we can use, check out the [documentation](#) for controlling figure aesthetics.

Before we move on, let's make one more change. While the plot looks much better now, the size of the text for ticks and axis labels is so small that it would be hard for people to read it unless they're right in front of the monitor--that's a problem, if the visualizations are going to be used in something like a tech talk or presentation!

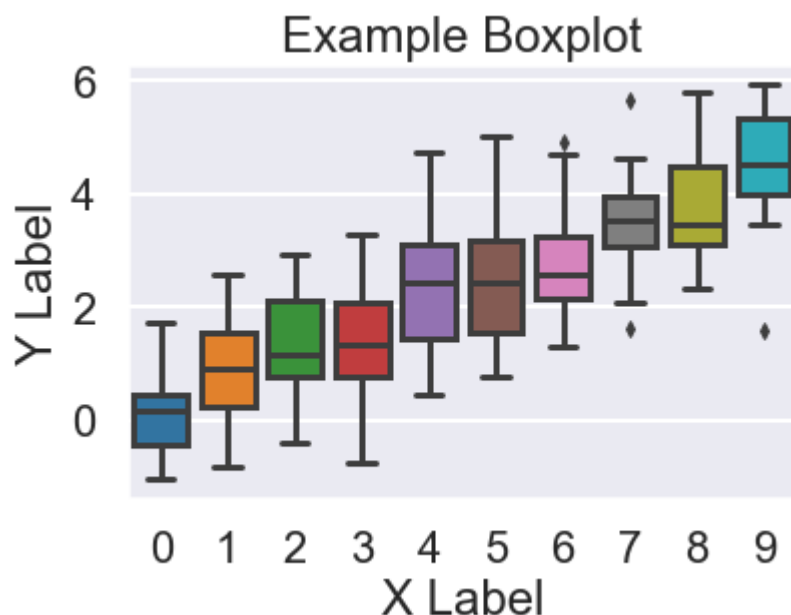
For this reason, we can also set the context, using the--you guessed it-- `set_context()` method!

In the cell below:

- Call Seaborn's `set_context()` method and pass in the string `'poster'`.
- Recreate the labeled boxplot that we made in the cell above.

```
sns.set_context('poster')
sns.boxplot(data=data).set(xlabel = "X Label", ylabel='Y Label', title='Example Boxp
```

```
[Text(0, 0.5, 'Y Label'),
 Text(0.5, 0, 'X Label'),
 Text(0.5, 1.0, 'Example Boxplot')]
```



Much better! That's much more readable. From smallest to largest, the different context settings we can use are 'paper' , 'notebook' , 'talk' , and 'poster' .

A Quick Note on Contexts and Styles

When you call `set_context` or `set_style` , you're setting a global parameter that will apply to all future plots you create during this session. Any visualizations you have already created will not change--however, they will change if you rerun the cell that created them!

Let's change our context back to 'notebook' so that the next visualizations we create don't look too big.

In the cell below, change the context back to 'notebook' .

```
sns.set_context('notebook')
```

More Advanced Visualizations

One awesome feature of Seaborn is the ability to quickly and easily create advanced visualizations such as **Regression Plots**. To end this lab, we'll see a few examples, and explore how they are created.

Regression Lines with Confidence Intervals

There are also several different types of regression plots Seaborn makes available for this purpose. For this example, we're going to create an advanced regression plot that also visualizes the confidence interval for our regression line. We'll even have the visualization **condition on** a 3rd variable, to show how the regression lines differ for each group, depending on the value of the 3rd variable.

For this visualization, we'll need a more advanced dataset than the example we created and used above. Luckily, Seaborn comes with some preloaded datasets. We can see the names of all the datasets by calling Seaborn's `get_dataset_names()` method.

Do this now in the cell below.

```
sns.get_dataset_names()
```

```
C:\Users\medio\AppData\Local\Continuum\anaconda3\lib\site-  
packages\seaborn\utils.py:376: UserWarning: No parser was explicitly specified,  
so I'm using the best available HTML parser for this system ("lxml"). This  
usually isn't a problem, but if you run this code on another system, or in a  
different virtual environment, it may use a different parser and behave  
differently.
```

The code that caused this warning is on line 376 of the file
C:\Users\medio\AppData\Local\Continuum\anaconda3\lib\site-
packages\seaborn\utils.py. To get rid of this warning, pass the additional
argument `'features="lxml"'` to the BeautifulSoup constructor.

```
gh_list = BeautifulSoup(http)
```

```
['anscombe',  
 'attention',  
 'brain_networks',  
 'car_crashes',  
 'diamonds',  
 'dots',  
 'exercise',  
 'flights',  
 'fmri',  
 'gammas',  
 'iris',  
 'mpg',  
 'planets',
```



```
'tips',  
'titanic']
```

Great! For the remainder of this notebook, we'll use the 'tips' dataset. We can get this dataset by calling Seaborn's `load_dataset()` method and passing in the string 'tips'. Seaborn is even considerate enough to return the dataset as a pandas DataFrame!

In the cell below, get the tips dataset and store it in the variable `tips`. Then, display the head of the DataFrame so we can see what we're working with.

```
tips = sns.load_dataset('tips')  
tips.head()
```

```
<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }
```

```
.dataframe tbody tr th {  
    vertical-align: top;  
}
```

```
.dataframe thead th {  
    text-align: right;  
}
```

```
</style>
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

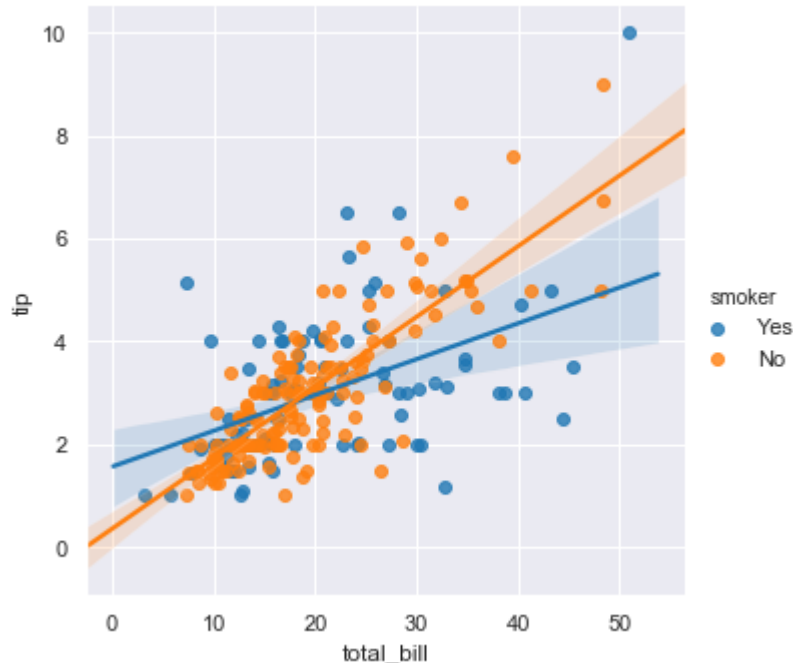
Now that we have our dataset, we can create our regression plot. There are several kinds of regression plots we can use. For this example, we'll use the `lmp1ot` function.

In the cell below:

- Call Seaborn's `lmp1ot` function and pass in the following arguments:
 - `x='total_bill'`
 - `y='tip'`

- `hue='smoker'`
- `data= tips`

```
sns.lmplot(x="total_bill", y="tip", hue="smoker", data=tips);
```

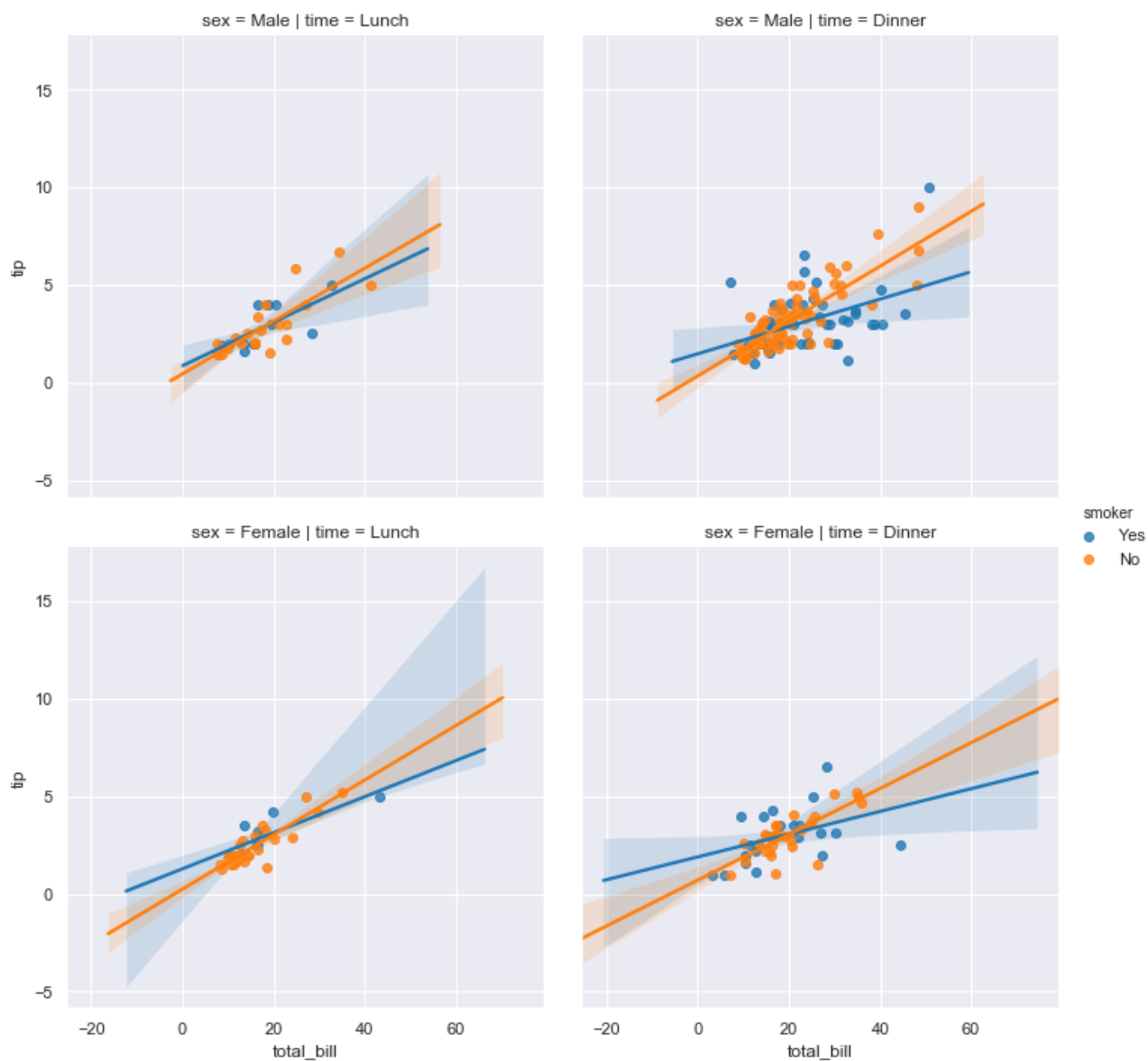


Very cool! That visualization contains *a lot* of information, and it does it in a way that is easy to interpret and understand. Best of all, it didn't take much work on our part--all we had to do was tell the function the name of the column to use for the x axis, the name of the column to use for the y axis, and the name of the variable to condition on, as denoted by the two different colors.

If we want to get even more ambitious, we can create multiple subplots by using the `row=` and `column=` parameters, as well!

Run the cell below to see an example, and see if you can figure out how the code works.

```
sns.lmplot(x="total_bill", y="tip", hue="smoker",  
           col="time", row="sex", data=tips)
```



Summary

In this lab, we explored the *Seaborn* library, and explored the sorts of data visualizations we can create with it!

Releases

No releases published

Packages

No packages published

Contributors 3



mike-kane Mike Kane



LoreDirick Lore Dirick



fpolchow Forest Polchow

Languages

● **Jupyter Notebook** 100.0%