**Introduction**

In this lesson, we will explore how to use Seaborn to graph multiple statistical distributions, including box plots and violin plots.

Seaborn is optimized to work with large datasets — from its ability to natively interact with Pandas DataFrames, to automatically calculating and plotting aggregates. One of the most powerful aspects of Seaborn is its ability to visualize and compare distributions. Distributions provide us with more information about our data — how spread out it is, its range, etc.

Calculating and graphing distributions is integral to analyzing massive amounts of data. We'll look at how Seaborn allows us to move beyond the traditional distribution graphs to plots that enable us to communicate important statistical information.

Instructions

To your right, you'll find a Jupyter notebook with some example Seaborn charts. We won't be using Jupyter notebooks in this lesson, but they're a great way of combining text, code, and visualization. You can find more about them on the [Jupyter website](https://jupyter.org/).

Take a moment to look at the graphs to the right. What do you notice about each of these charts? What kind of statistical information could they be trying to convey?

When you're ready, continue to the first exercise.

\*\* see the Plotting Distributions with Seaborn PDF in this directory \*\*

**Bar Charts Hide Information**

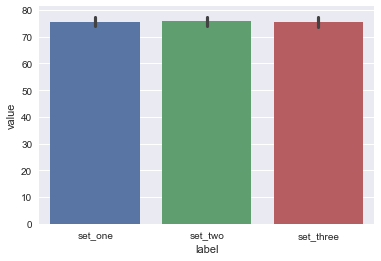
Before we dive into these new charts, we need to understand why we'd want to use them. To best illustrate this idea, we need to revisit bar charts.

We previously learned that Seaborn can quickly aggregate data to plot bar charts using the mean.

Here is a bar chart that uses three different randomly generated sets of data:

sns.barplot(data=df, x="label", y="value")

plt.show()



These three datasets look identical! As far as we can tell, they each have the same mean and similar confidence intervals.

We can get a lot of information from these bar charts, but we can't get everything. For example, what are the minimum and maximum values of these datasets? How spread out is this data?

While we may not see this information in our bar chart, these differences might be significant and worth understanding better.

Instructions

**1.**

You work as a scientist and are measuring the amounts of plastic in different bodies of water. You're interested in comparing data collected from different locations.

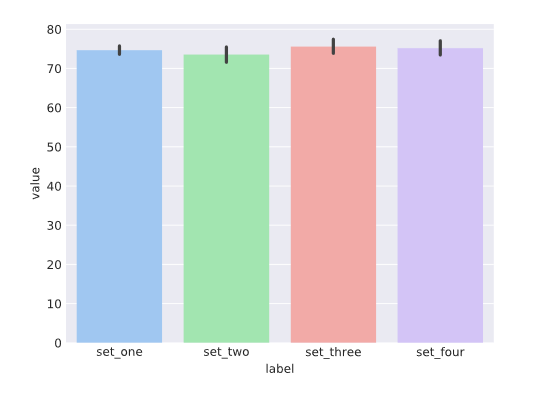
We've imported four different datasets using NumPy and have combined them into one DataFrame, df.

Use sns.barplot() to graph the datasets in one plot, with "label" as the x data and "value" as the y data.

**2.**

Use plt.show() to display the bar plots. How similar are they? How different? Are we able to make an adequate comparison of these values?

import codecademylib3\_seaborn  
import pandas as pd  
import numpy as np  
from matplotlib import pyplot as plt  
import seaborn as sns  
  
# Take in the data from the CSVs as NumPy arrays:  
set\_one = np.genfromtxt("dataset1.csv", delimiter=",")  
set\_two = np.genfromtxt("dataset2.csv", delimiter=",")  
set\_three = np.genfromtxt("dataset3.csv", delimiter=",")  
set\_four = np.genfromtxt("dataset4.csv", delimiter=",")  
  
# Creating a Pandas DataFrame:  
n=500  
df = pd.DataFrame({  
 "label": ["set\_one"] \* n + ["set\_two"] \* n + ["set\_three"] \* n + ["set\_four"] \* n,  
 "value": np.concatenate([set\_one, set\_two, set\_three, set\_four])  
})  
  
# Setting styles:  
sns.set\_style("darkgrid")  
sns.set\_palette("pastel")  
  
# Add your code below:  
sns.barplot(data=df, x="label", y="value")  
plt.show()



**KDE Plots, Part I**

Bar plots can tell us what the mean of our dataset is, but they don't give us any hints as to the distribution of the dataset values. For all we know, the data could be clustered around the mean or spread out evenly across the entire range.

To find out more about each of these datasets, we'll need to examine their distributions. A common way of doing so is by plotting the data as a histogram, but histograms have their drawback as well.

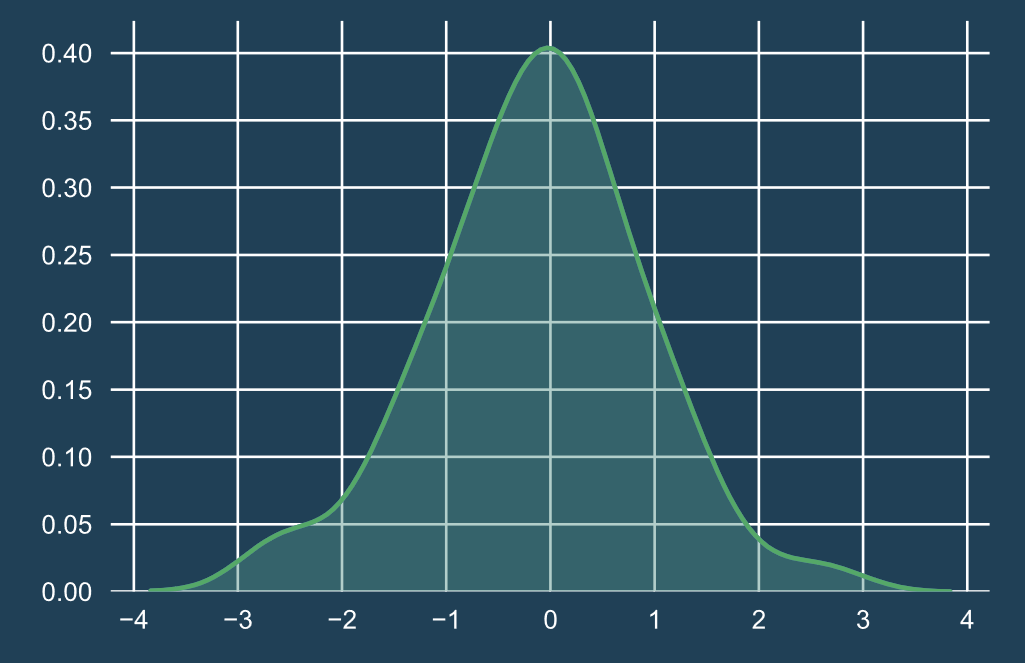
Seaborn offers another option for graphing distributions: *KDE Plots*.

KDE stands for Kernel Density Estimator. A KDE plot gives us the sense of a *univariate* as a curve. A univariate dataset only has one variable and is also referred to as being *one-dimensional*, as opposed to *bivariate* or two-dimensional datasets which have two variables.

KDE plots are preferable to histograms because depending on how you group the data into bins and the width of the bins, you can draw wildly different conclusions about the shape of the data. Using a KDE plot can mitigate these issues, because they smooth the datasets, allow us to generalize over the shape of our data, and aren't beholden to specific data points.

Instructions

Examine the KDE plot to the right. Notice how it shows the distribution of a dataset and has a smoother shape than a histogram.



# KDE Plots, Part II

To plot a KDE in Seaborn, we use the method sns.kdeplot().

A KDE plot takes the following arguments:

* data - the univariate dataset being visualized, like a Pandas DataFrame, Python list, or NumPy array
* shade - a boolean that determines whether or not the space underneath the curve is shaded

Let's examine the KDE plots of our three datasets:

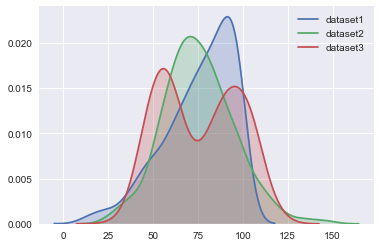
sns.kdeplot(dataset1, shade=True)

sns.kdeplot(dataset2, shade=True)

sns.kdeplot(dataset3, shade=True)

plt.legend()

plt.show()



Notice that when using a KDE we need to plot each of the original datasets separately, rather than using a combined dataframe like we did with the bar plot.

It looks like there are some big differences between the three datasets:

* Dataset 1 is skewed left
* Dataset 2 is normally distributed
* Dataset 3 is bimodal (it has two peaks)

So although all three datasets have roughly the same mean, the shapes of the KDE plots demonstrate the differences in how the values are distributed.

Instructions

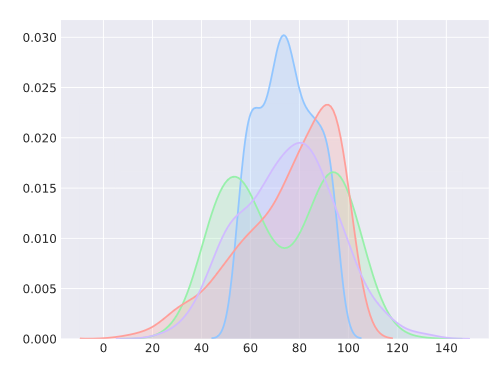
**1.**

Use sns.kdeplot() to graph the four datasets and set shade to True.

**2.**

Use plt.show() to display the KDE plots.

import codecademylib3\_seaborn  
import pandas as pd  
import numpy as np  
from matplotlib import pyplot as plt  
import seaborn as sns  
  
# Take in the data from the CSVs as NumPy arrays:  
set\_one = np.genfromtxt("dataset1.csv", delimiter=",")  
set\_two = np.genfromtxt("dataset2.csv", delimiter=",")  
set\_three = np.genfromtxt("dataset3.csv", delimiter=",")  
set\_four = np.genfromtxt("dataset4.csv", delimiter=",")  
  
# Creating a Pandas DataFrame:  
n=500  
df = pd.DataFrame({  
 "label": ["set\_one"] \* n + ["set\_two"] \* n + ["set\_three"] \* n + ["set\_four"] \* n,  
 "value": np.concatenate([set\_one, set\_two, set\_three, set\_four])  
})  
  
# Setting styles:  
sns.set\_style("darkgrid")  
sns.set\_palette("pastel")  
  
# Add your code below:  
sns.kdeplot(set\_one, shade=True)  
sns.kdeplot(set\_two, shade=True)  
sns.kdeplot(set\_three, shade=True)  
sns.kdeplot(set\_four, shade=True)  
  
plt.show()



**Box Plots, Part I**

While a KDE plot can tell us about the shape of the data, it's cumbersome to compare multiple KDE plots at once. They also can't tell us other statistical information, like the values of outliers.

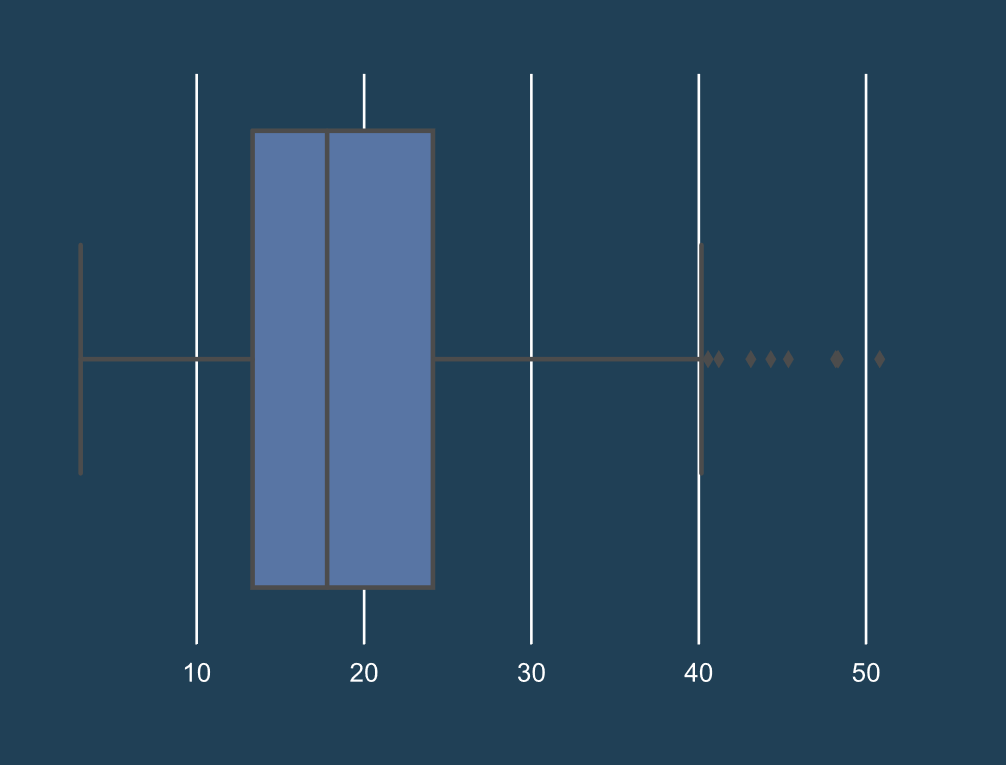
The *box plot* (also known as a box-and-whisker plot) can tell us about how our dataset is distributed, like a KDE plot. But it shows us the range of our dataset, gives us an idea about where a significant portion of our data lies, and whether or not any outliers are present.

Let's examine how we interpret a box plot:

* The **box** represents the interquartile range
* The **line in the middle** of the box is the median
* The **end lines** are the first and third quartiles
* The **diamonds** show outliers

Instructions

Examine the box plot to the right. Notice how it also is able to show distributions, along with information about outliers.



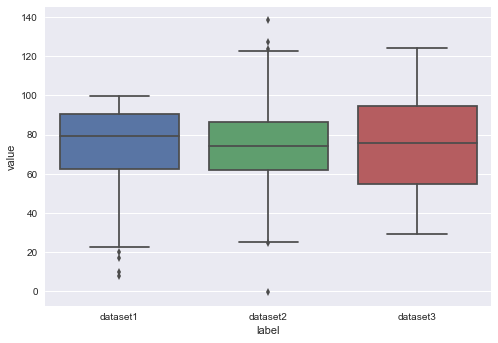
**Box Plots, Part II**

One advantage of the box plot over the KDE plot is that in Seaborn, it is easy to plot multiples and compare distributions.

Let's look again at our three datasets, and how they look plotted as box plots:

sns.boxplot(data=df, x='label', y='value')

plt.show()



The box plot does a good job of showing certain differences, the different between Dataset 1 and Dataset 2; however, it does not show that Dataset 3 is bimodal.

To plot a box plot in Seaborn, we use the method sns.boxplot().

A box plot takes the following arguments:

* data - the dataset we're plotting, like a DataFrame, list, or an array
* x - a one-dimensional set of values, like a Series, list, or array
* y - a second set of one-dimensional data

If you use a Pandas Series for the x and y values, the Series will also generate the axis labels. For example, if you use the value Series as your y value data, Seaborn will automatically apply that name as the y-axis label.

Instructions

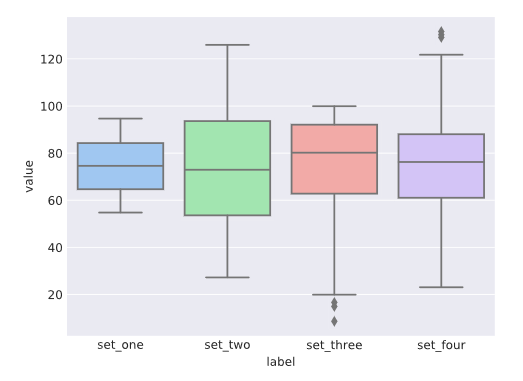
**1.**

Use sns.boxplot() to plot the four datasets as box plots.

**2.**

Use plt.show() to display the box plots. What else can we learn about these datasets?

import codecademylib3\_seaborn  
import pandas as pd  
import numpy as np  
from matplotlib import pyplot as plt  
import seaborn as sns  
  
# Take in the data from the CSVs as NumPy arrays:  
set\_one = np.genfromtxt("dataset1.csv", delimiter=",")  
set\_two = np.genfromtxt("dataset2.csv", delimiter=",")  
set\_three = np.genfromtxt("dataset3.csv", delimiter=",")  
set\_four = np.genfromtxt("dataset4.csv", delimiter=",")  
  
# Creating a Pandas DataFrame:  
n=500  
df = pd.DataFrame({  
 "label": ["set\_one"] \* n + ["set\_two"] \* n + ["set\_three"] \* n + ["set\_four"] \* n,  
 "value": np.concatenate([set\_one, set\_two, set\_three, set\_four])  
})  
  
# Setting styles:  
sns.set\_style("darkgrid")  
sns.set\_palette("pastel")  
  
# Add your code below:  
sns.boxplot(data=df, x='label', y='value')  
plt.show()



**Violin Plots, Part I**

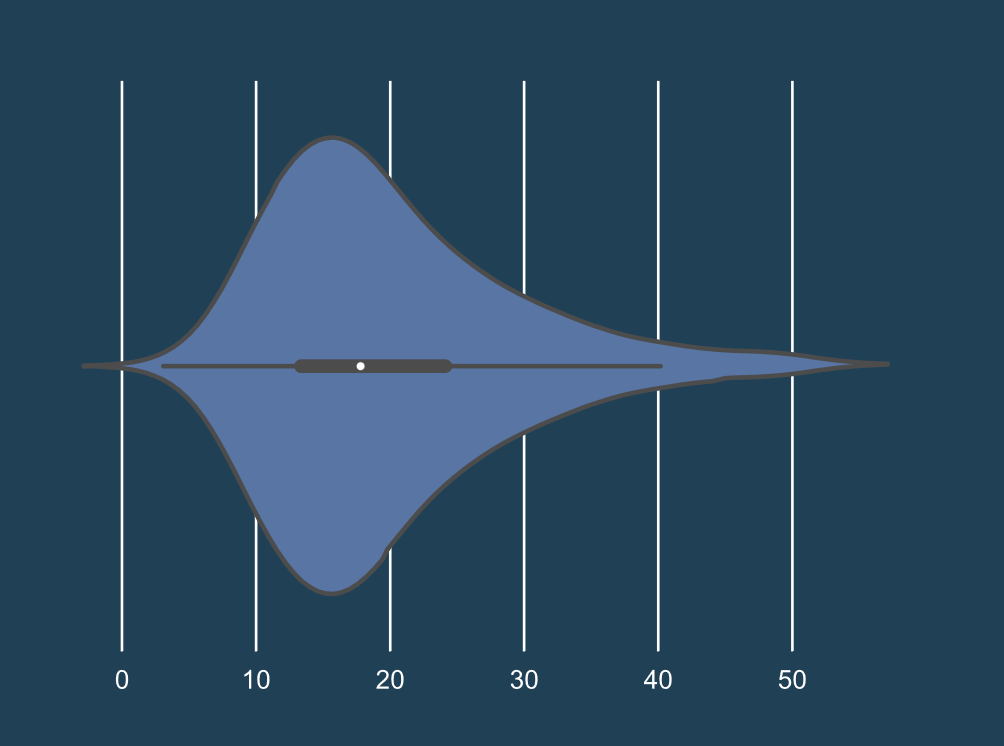
As we saw in the previous exercises, while it's possible to plot multiple histograms, it is not a great option for comparing distributions. Seaborn gives us another option for comparing distributions - a *violin plot*. Violin plots provide more information than box plots because instead of mapping each individual data point, we get an estimation of the dataset thanks to the KDE.

Violin plots are less familiar and trickier to read, so let's break down the different parts:

* There are two **KDE plots** that are symmetrical along the center line.
* A **white dot** represents the median.
* The **thick black line** in the center of each violin represents the interquartile range.
* The **lines that extend from the center** are the confidence intervals - just as we saw on the bar plots, a violin plot also displays the 95% confidence interval.

Instructions

Examine the violin plot to the right. Notice how it also is able to show distributions, like the KDE plot, as well as information about the median and interquartile range, like the box plot.



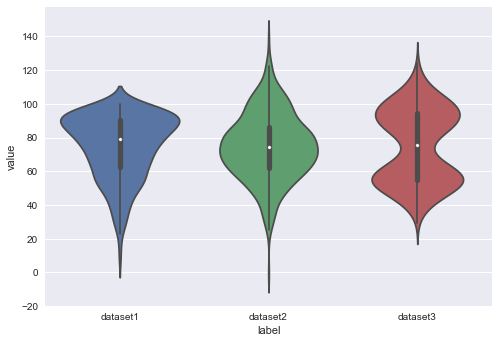
**Violin Plots, Part II**

Violin Plots are a powerful graphing tool that allows you to compare multiple distributions at once.

Let's look at how our original three data sets look like as violin plots:

sns.violinplot(data=df, x="label", y="value")

plt.show()



As we can see, violin plots allow us to graph and compare multiple distributions. It also retains the shape of the distributions, so we can easily tell that Dataset 1 is skewed left and that Dataset 3 is bimodal.

To plot a violin plot in Seaborn, use the method sns.violinplot().

There are several options for passing in relevant data to the x and y parameters:

* data - the dataset that we're plotting, such as a list, DataFrame, or array
* x, y, and hue - a one-dimensional set of data, such as a Series, list, or array
* any of the parameters to the function sns.boxplot()

Instructions

**1.**

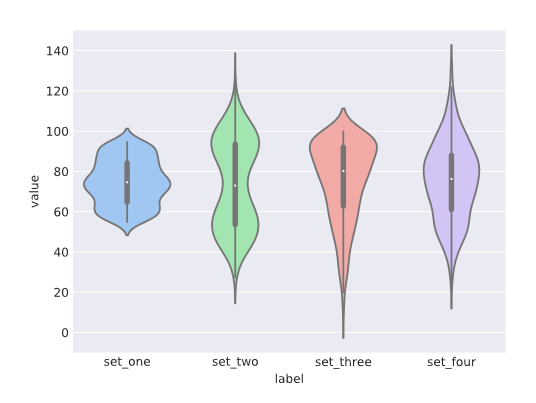
Using sns.violinplot(), plot the four datasets as violin plots.

**2.**

Use plt.show() to display the violin plots.

Notice how much easier it is to compare the shape of the datasets than when they were layered on top of each other!

import codecademylib3\_seaborn  
import pandas as pd  
import numpy as np  
from matplotlib import pyplot as plt  
import seaborn as sns  
  
# Take in the data from the CSVs as NumPy arrays:  
set\_one = np.genfromtxt("dataset1.csv", delimiter=",")  
set\_two = np.genfromtxt("dataset2.csv", delimiter=",")  
set\_three = np.genfromtxt("dataset3.csv", delimiter=",")  
set\_four = np.genfromtxt("dataset4.csv", delimiter=",")  
  
# Creating a Pandas DataFrame:  
n=500  
df = pd.DataFrame({  
 "label": ["set\_one"] \* n + ["set\_two"] \* n + ["set\_three"] \* n + ["set\_four"] \* n,  
 "value": np.concatenate([set\_one, set\_two, set\_three, set\_four])  
})  
  
# Setting styles:  
sns.set\_style("darkgrid")  
sns.set\_palette("pastel")  
  
# Add your code below:  
sns.violinplot(data=df, x="label", y="value")  
plt.show()



# Review

In this lesson, we examined how Seaborn has several plots that can visualize distributions. While bar plots can display basic aggregates, KDE plots, dist plots, box plots, and violin plots can show us distributions and other information.

* **KDE plot** - Kernel density estimator; shows a smoothed version of dataset. Use sns.kdeplot().
* **Box plot** - A classic statistical model that shows the median, interquartile range, and outliers. Use sns.boxplot().
* **Violin plot** - A combination of a KDE and a box plot. Good for showing multiple distributions at a time. Use sns.violinplot().

Instructions

Take the time to play around with the code that you have previously written.