Organizing and Visualizing Data in Python

Becky Lee

2020-05-09

Hello world! Welcome to my very first blog post. This beginner level guide will show you various ways of organizing and visualizing data using Python. Everything is at the beginner level! Let's get started.

For this tutorial style walkthrough, we'll be working with the pokemon dataset, renamed 'poke' here for convenience. The python packages 'pandas' and 'seaborn' will come in handy too!

```
import pandas as pd
import seaborn as sns
pokemon=pd.read_csv("pokemon.csv",index_col=0)
poke=pokemon
poke.head()
```

#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	1	False
2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	1	False
3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	1	False
3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	1	False
4	Charmander	Fire	NaN	309	39	52	43	60	50	65	1	False

Now that we are somewhat familiar with the dataset, we could start organizing and trying different functions.

One way is to select certain columns, like we do in R with the select() fucntion from dplyr. The format of the code is shown below, where the numbers represent their respective columns. Not that the Name column is '0' and not '1'.

```
poke.iloc[:,[0,4,5,6]].head()
```

#	Name	HP	Attack	Defense
1	Bulbasaur	45	49	49
2	Ivysaur	60	62	63
3	Venusaur	80	82	83
3	VenusaurMega Venusaur	80	100	123
4	Charmander	39	52	43

The exact same result could be done when using the column names instead of numbers.

```
poke1=poke[["Name","HP","Attack","Defense"]]
poke1.head()
```

#	Name	HP	Attack	Defense
1	Bulbasaur	45	49	49
2	Ivysaur	60	62	63
3	Venusaur	80	82	83
3	VenusaurMega Venusaur	80	100	123
4	Charmander	39	52	43

Various functions, such as mean(), median(), max(), or count() could be performed on data. The code below takes the mean HP of the dataset defined earlier as 'poke1'.

```
poke1["HP"].mean()
```

```
69.25875000000006
```

We could also use certain criteria to select only observations which meet that condition. By using count() and mean(), we know that only 378 pokemon out of the original 800 have an HP that is above average.

```
poke1[poke1["HP"] > 69.25875].count()
```

```
Name       378
HP         378
Attack     378
Defense    378
dtype: int64
```

Using the condition above, create another dataset, 'poke2' for example. This dataset only includes pokemon which have higher than average HP. Let's create a new column called 'Total', which will be the 'Attack' plus 'Defense' point number.

```
poke2 = poke[(poke["HP"] > 69.25875)]
poke2['Total']=poke2['Attack']+poke2['Defense']
poke2.head()
```

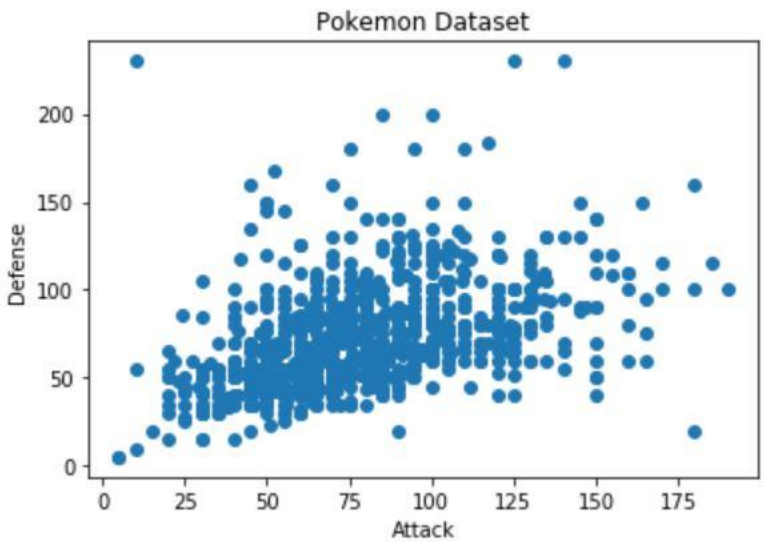
```
/opt/jupyterhub/pyvenv/lib/python3.6/site-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#index
ing-view-versus-copy
```

#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
3	Venusaur	Grass	Poison	165	80	82	83	100	100	80	1	False
3	VenusaurMega Venusaur	Grass	Poison	223	80	100	123	122	120	80	1	False
6	Charizard	Fire	Flying	162	78	84	78	109	85	100	1	False
6	CharizardMega Charizard X	Fire	Dragon	241	78	130	111	130	85	100	1	False
6	CharizardMega Charizard Y	Fire	Flying	182	78	104	78	159	115	100	1	False

Visualizing data is an important part of coding, especially for scientists. Let's go through a few different types of plots, starting with a general scatterplot.

```python
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax.scatter(poke['Attack'], poke['Defense'])
ax.set_title('Pokemon Dataset')
ax.set_xlabel('Attack')
ax.set_ylabel('Defense')
```

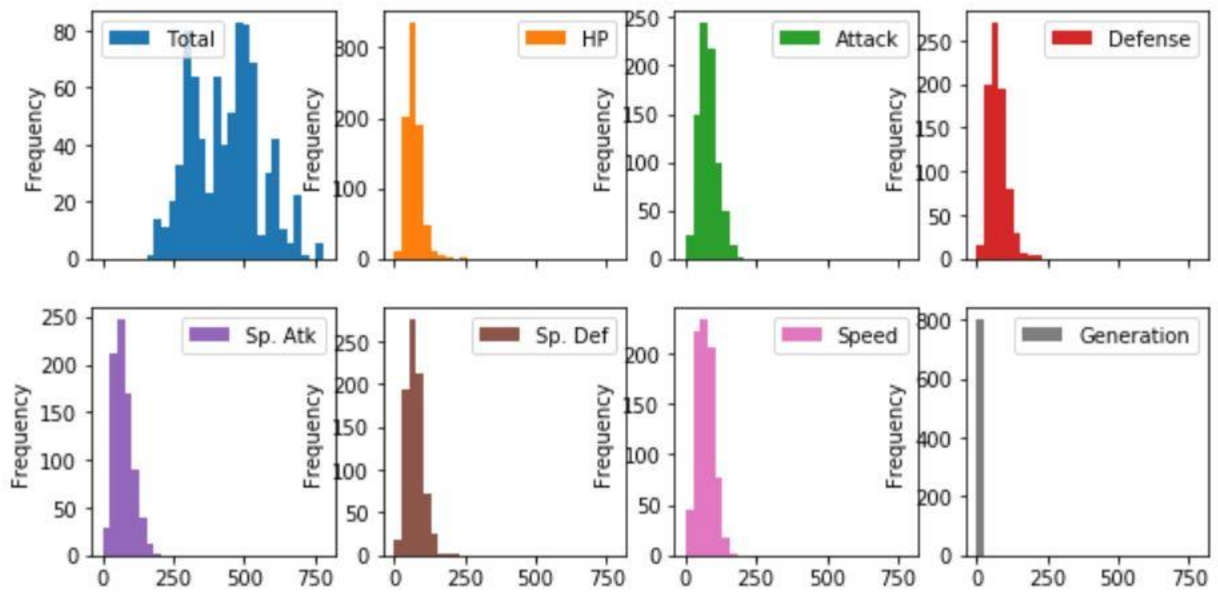
```
Text(0,0.5,'Defense')
```



We could also do faceting, and make separate histograms for each column variable. This includes every pokemon and gives simple distributions which are helpful for eyeballing trends.

```python
poke.plot.hist(subplots=True, layout=(3,3), figsize=(10, 10), bins=30)
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f8c69b29860>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8c69b446d8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8c69aebc50>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f8c69a9d208>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8c69ac4780>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8c69ac47b8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f8c69a1e2b0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8c69a46828>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f8c699f1da0>]], dtype=object)
```



To find correlations between every single variable, create a correlation table using corr().

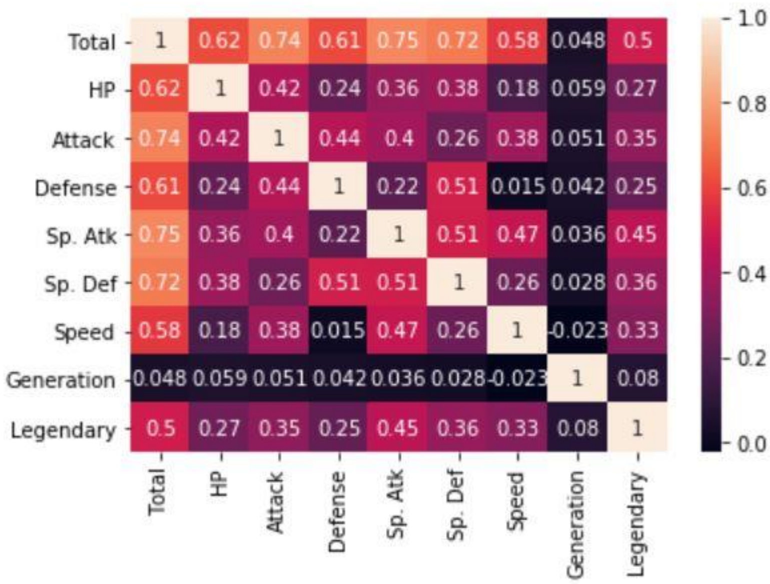
```
import numpy as np

corr = poke.corr()
im = ax.imshow(corr.values)
corr
```

	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
Total	1.000000	0.618748	0.736211	0.612787	0.747250	0.717609	0.575943	0.048384	0.501758
HP	0.618748	1.000000	0.422386	0.239622	0.362380	0.378718	0.175952	0.058683	0.273620
Attack	0.736211	0.422386	1.000000	0.438687	0.396362	0.263990	0.381240	0.051451	0.345408
Defense	0.612787	0.239622	0.438687	1.000000	0.223549	0.510747	0.015227	0.042419	0.246377
Sp. Atk	0.747250	0.362380	0.396362	0.223549	1.000000	0.506121	0.473018	0.036437	0.448907
Sp. Def	0.717609	0.378718	0.263990	0.510747	0.506121	1.000000	0.259133	0.028486	0.363937
Speed	0.575943	0.175952	0.381240	0.015227	0.473018	0.259133	1.000000	-0.023121	0.326715
Generation	0.048384	0.058683	0.051451	0.042419	0.036437	0.028486	-0.023121	1.000000	0.079794
Legendary	0.501758	0.273620	0.345408	0.246377	0.448907	0.363937	0.326715	0.079794	1.000000

A correlation heatmap puts those numbers above into a visual display with colors.

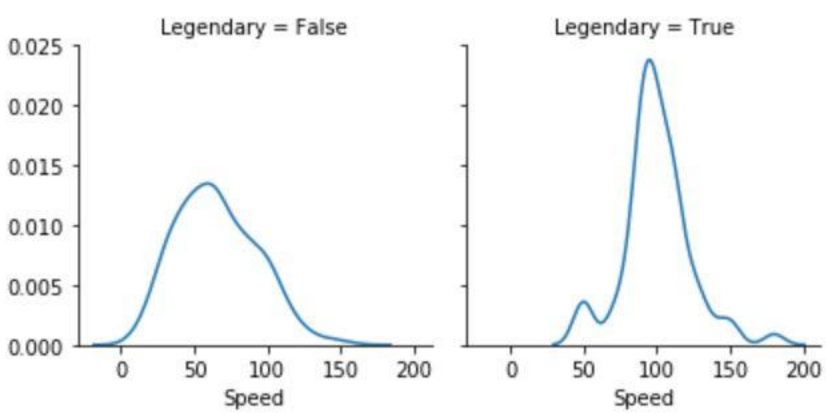
```python
import seaborn as sns

sns.heatmap(poke.corr(), annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8c6a036748>
```



To see if being Legendary significantly affects a pokemon's stats, we could facet once again. The example below compares the speed distributions for Legendary vs. non-Legendary pokemons.

```python
g = sns.FacetGrid(poke, col='Legendary')
g = g.map(sns.kdeplot, 'Speed')
```



That's the end of the tutorial, but there is plenty more to explore in Python! Good luck.