Day 9: Data Analysis and Visualization

When we plot we need more libraries. You learned about matplot on Day 5, today we are going to build on what you learned and plot directly from pandas and I’m going to introduce seaborn.

Goals for the day:

- Practice making a pivot table
- More understanding of groupby
- Introduction to seaborn
- Introduction to graph automation

Functions Learned:

- Create a heatmap: sns.heatmap()
- Create a pairplot: sns.pairplot()
- Create a swarmplot: sns.swarmplot()
- Create a violinplot: sns.violinplot

I will not be able to finish the contents of this notebook but there are many useful tips here so I highly recommend that you complete on your own.

```
[1]: # this line prints your matplot based plots in this notebook, in some computers without this line your graph will pop out as a window
%matplotlib inline

#call all the matplot libraries I may need
import matplotlib.pyplot as plt
from matplotlib import cm

#the usual
import pandas as pd
import numpy as np

#seaborn
import seaborn as sns; sns.set(color_codes=True)

#some libraries that allow is to run from stats
```
0. Set Directory

```python
#this is the specific directory where the data we want to use is stored
datadirectory = '../data/'

#this is the directory where we want to store the data we finish analyzing
data_out_directory='../output/'
```

Seaborn

For all the plot examples I am going to do below I am going to use seaborn. You can look at the seaborn gallery and look at all the plotting options it offers.

14. Combination skills, Heatplots

I really want to know what the strongest pokemon type is for each generation and stats (Attack, Def. etc).

```python
# call in my data
pokemon=pd.read_csv(datadirectory+'Pokemon.csv')
pokemon.head()
```

```
<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Type 1</th>
<th>Type 2</th>
<th>Total</th>
<th>HP</th>
<th>Attack</th>
<th>Defense</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Bulbasaur</td>
<td>Grass</td>
<td>Poison</td>
<td>318</td>
<td>45</td>
<td>49</td>
<td>49</td>
</tr>
<tr>
<td>1</td>
<td>Ivysaur</td>
<td>Grass</td>
<td>Poison</td>
<td>405</td>
<td>60</td>
<td>62</td>
<td>63</td>
</tr>
<tr>
<td>2</td>
<td>Venusaur</td>
<td>Grass</td>
<td>Poison</td>
<td>525</td>
<td>80</td>
<td>82</td>
<td>83</td>
</tr>
<tr>
<td>3</td>
<td>VenusaurMega</td>
<td>Venusaur</td>
<td>Grass</td>
<td>625</td>
<td>80</td>
<td>100</td>
<td>123</td>
</tr>
<tr>
<td>4</td>
<td>Charmander</td>
<td>Fire</td>
<td>NaN</td>
<td>309</td>
<td>39</td>
<td>52</td>
<td>43</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sp. Atk</th>
<th>Sp. Def</th>
<th>Speed</th>
<th>Generation</th>
<th>Legendary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>65</td>
<td>65</td>
<td>45</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>80</td>
<td>80</td>
<td>60</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>100</td>
<td>80</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>122</td>
<td>120</td>
<td>80</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>60</td>
<td>50</td>
<td>65</td>
<td>1</td>
</tr>
</tbody>
</table>
```

```python
### make a pivot table
data_subset=pokemon.pivot_table(values='Attack',index=['Type1'],columns=['Generation'],aggfunc='min')
data_subset
```

```
<table>
<thead>
<tr>
<th>Generation</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bug</td>
<td>20.0</td>
<td>10.0</td>
<td>30.0</td>
<td>25.0</td>
<td>40.0</td>
<td>22.0</td>
</tr>
</tbody>
</table>
```
Dark    NaN  60.0  55.0  90.0  50.0  54.0
Dragon  64.0  NaN  70.0  70.0  87.0  50.0
Electric 30.0  40.0  40.0  45.0  55.0  38.0
Fairy  45.0   20.0   NaN  50.0   NaN  38.0
Fighting 80.0  35.0   40.0  70.0  80.0  82.0
Fire    41.0  40.0   60.0  58.0  30.0  45.0
Flying  NaN  NaN  NaN  NaN 100.0  30.0
Ghost  35.0   60.0  40.0  50.0  30.0  66.0
Grass   40.0  30.0  40.0  30.0  27.0  61.0
Ground 50.0   60.0  40.0  72.0  66.0  NaN
Ice     50.0   30.0  40.0  60.0  50.0  69.0
Normal  5.0   10.0  20.0   5.0  50.0  36.0
Poison  45.0  90.0  43.0  50.0  50.0  60.0
Psychic 20.0  33.0  23.0  25.0  25.0  48.0
Rock   40.0  64.0  41.0  42.0  75.0  50.0
Steel   NaN  80.0  55.0  24.0  55.0  50.0
Water   10.0  20.0  15.0  20.0  40.0  53.0

**Coding Challenge**

Make a function that makes a pivot table from the pokemon data. This function will allow me to use any aggregation function I want on any pokemon stats I want. The argument for this function should be `stats_wanted` and `aggfun_wanted`. The index is ‘Type 1’ and the columns ‘Generation’.  

```python
### your code here

```  

```python
def get_summary(stats_wanted, aggfun_wanted):
    data_subset = pokemon.pivot_table(values=stats_wanted, index=['Type/uni2423'], columns=['Generation'], aggfunc=aggfun_wanted)
    return data_subset

def get_summary('Speed', 'max')
```

---

**[5]:**  

```python
get_summary('Speed', 'max')
```  

**[5]:**  

<table>
<thead>
<tr>
<th>Generation</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug</td>
<td>145.0</td>
<td>95.0</td>
<td>160.0</td>
<td>95.0</td>
<td>145.0</td>
<td>89.0</td>
</tr>
<tr>
<td>Dark</td>
<td>NaN</td>
<td>115.0</td>
<td>115.0</td>
<td>125.0</td>
<td>106.0</td>
<td>99.0</td>
</tr>
<tr>
<td>Dragon</td>
<td>80.0</td>
<td>NaN</td>
<td>120.0</td>
<td>102.0</td>
<td>97.0</td>
<td>95.0</td>
</tr>
<tr>
<td>Electric</td>
<td>140.0</td>
<td>115.0</td>
<td>135.0</td>
<td>95.0</td>
<td>116.0</td>
<td>109.0</td>
</tr>
<tr>
<td>Fairy</td>
<td>60.0</td>
<td>45.0</td>
<td>NaN</td>
<td>80.0</td>
<td>NaN</td>
<td>99.0</td>
</tr>
<tr>
<td>Fighting</td>
<td>95.0</td>
<td>70.0</td>
<td>100.0</td>
<td>112.0</td>
<td>105.0</td>
<td>118.0</td>
</tr>
<tr>
<td>Fire</td>
<td>105.0</td>
<td>100.0</td>
<td>100.0</td>
<td>108.0</td>
<td>101.0</td>
<td>126.0</td>
</tr>
<tr>
<td>Flying</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>121.0</td>
<td>123.0</td>
</tr>
<tr>
<td>Ghost</td>
<td>130.0</td>
<td>85.0</td>
<td>75.0</td>
<td>105.0</td>
<td>80.0</td>
<td>99.0</td>
</tr>
<tr>
<td>Grass</td>
<td>80.0</td>
<td>110.0</td>
<td>145.0</td>
<td>127.0</td>
<td>116.0</td>
<td>68.0</td>
</tr>
</tbody>
</table>
### make a function that will aggregate my data based on whatever aggfun I want

```python
def get_summary(stat_wanted, aggfun_wanted):
    data_subset = pokemon.pivot_table(values=stat_wanted, index=['Type'],
                                      columns=['Generation'], aggfunc=[aggfun_wanted])

    ### drop a level in the columns so the table looks nicer
    data_subset.columns = data_subset.columns.droplevel(0)
    return data_subset

get_summary('Attack', 'max')
```

### Generation 1 2 3 4 5 6

<table>
<thead>
<tr>
<th>Type</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug</td>
<td>155.0</td>
<td>185.0</td>
<td>90.0</td>
<td>94.0</td>
<td>135.0</td>
<td>52.0</td>
</tr>
<tr>
<td>Dark</td>
<td>NaN</td>
<td>95.0</td>
<td>150.0</td>
<td>125.0</td>
<td>125.0</td>
<td>131.0</td>
</tr>
<tr>
<td>Dragon</td>
<td>134.0</td>
<td>NaN</td>
<td>180.0</td>
<td>170.0</td>
<td>170.0</td>
<td>100.0</td>
</tr>
<tr>
<td>Electric</td>
<td>90.0</td>
<td>95.0</td>
<td>75.0</td>
<td>123.0</td>
<td>115.0</td>
<td>58.0</td>
</tr>
<tr>
<td>Fairy</td>
<td>70.0</td>
<td>120.0</td>
<td>NaN</td>
<td>50.0</td>
<td>NaN</td>
<td>131.0</td>
</tr>
<tr>
<td>Fighting</td>
<td>130.0</td>
<td>95.0</td>
<td>120.0</td>
<td>145.0</td>
<td>140.0</td>
<td>124.0</td>
</tr>
<tr>
<td>Fire</td>
<td>130.0</td>
<td>130.0</td>
<td>160.0</td>
<td>104.0</td>
<td>140.0</td>
<td>110.0</td>
</tr>
<tr>
<td>Flying</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>115.0</td>
<td>70.0</td>
</tr>
<tr>
<td>Ghost</td>
<td>65.0</td>
<td>60.0</td>
<td>165.0</td>
<td>120.0</td>
<td>55.0</td>
<td>110.0</td>
</tr>
<tr>
<td>Grass</td>
<td>105.0</td>
<td>82.0</td>
<td>130.0</td>
<td>132.0</td>
<td>98.0</td>
<td>107.0</td>
</tr>
<tr>
<td>Ground</td>
<td>130.0</td>
<td>120.0</td>
<td>180.0</td>
<td>140.0</td>
<td>145.0</td>
<td>NaN</td>
</tr>
<tr>
<td>Ice</td>
<td>85.0</td>
<td>100.0</td>
<td>120.0</td>
<td>130.0</td>
<td>110.0</td>
<td>117.0</td>
</tr>
<tr>
<td>Normal</td>
<td>125.0</td>
<td>130.0</td>
<td>160.0</td>
<td>160.0</td>
<td>128.0</td>
<td>80.0</td>
</tr>
<tr>
<td>Poison</td>
<td>105.0</td>
<td>90.0</td>
<td>100.0</td>
<td>106.0</td>
<td>95.0</td>
<td>75.0</td>
</tr>
<tr>
<td>Psychic</td>
<td>190.0</td>
<td>100.0</td>
<td>180.0</td>
<td>165.0</td>
<td>100.0</td>
<td>160.0</td>
</tr>
<tr>
<td>Rock</td>
<td>135.0</td>
<td>164.0</td>
<td>125.0</td>
<td>165.0</td>
<td>140.0</td>
<td>160.0</td>
</tr>
<tr>
<td>Steel</td>
<td>NaN</td>
<td>125.0</td>
<td>145.0</td>
<td>120.0</td>
<td>100.0</td>
<td>150.0</td>
</tr>
<tr>
<td>Water</td>
<td>155.0</td>
<td>105.0</td>
<td>150.0</td>
<td>120.0</td>
<td>108.0</td>
<td>95.0</td>
</tr>
</tbody>
</table>
data_subset = pokemon.pivot_table(values=stat_wanted, index=['Type→1'], columns=['Generation'], aggfunc=[aggfun_wanted])

### drop a level in the columns so the table looks nicer
# data_subset.columns = data_subset.columns.droplevel(1)

## I can select more than one poke stat to plot, but from what I know with pivot you can only do one aggfunction

return data_subset

get_summary(['Attack', 'Speed'], ('max', 'min'))

[7]:
    max  
  min
Attack
  max  min
Generation  1  2  3  4  5  6  1  2  3  4
Type 1
Bug  155.0 185.0 90.0 94.0 135.0 52.0 20.0 10.0 30.0 25.0
Dark NaN 95.0 150.0 125.0 125.0 131.0 NaN 60.0 55.0 90.0
Dragon 134.0 NaN 180.0 170.0 170.0 100.0 64.0 NaN 70.0 70.0
Electric 90.0 95.0 75.0 123.0 115.0 58.0 30.0 40.0 40.0 45.0
Fairy 70.0 120.0 NaN 50.0 NaN 131.0 45.0 20.0 NaN 50.0
Fighting 130.0 95.0 120.0 145.0 140.0 124.0 80.0 35.0 40.0 70.0
Fire 130.0 130.0 160.0 104.0 140.0 110.0 41.0 40.0 60.0 58.0
Flying NaN NaN NaN NaN 115.0 70.0 NaN NaN NaN NaN
Ghost 65.0 60.0 165.0 120.0 55.0 110.0 35.0 60.0 40.0 50.0
Grass 85.0 100.0 120.0 130.0 110.0 117.0 50.0 30.0 40.0 60.0
Ground 125.0 130.0 160.0 160.0 128.0 80.0 5.0 10.0 20.0 5.0
Poison 105.0 90.0 100.0 106.0 95.0 75.0 45.0 90.0 43.0 50.0
Psychic 190.0 100.0 180.0 165.0 100.0 160.0 20.0 33.0 23.0 25.0
Rock 135.0 164.0 125.0 165.0 140.0 160.0 40.0 64.0 41.0 42.0
Steel NaN 125.0 145.0 120.0 100.0 150.0 NaN 80.0 55.0 24.0
Water 155.0 105.0 150.0 120.0 108.0 95.0 10.0 20.0 15.0 20.0

...  
...  
...  Speed  
...  max  min
Generation  ...  3  4  5  6  1  2  3  4  5
Type 1  
Bug  ...  160.0 95.0 145.0 89.0 25.0 5.0 15.0 25.0 20.0
Dark  ...  115.0 125.0 106.0 99.0 NaN 65.0 20.0 71.0 38.0
Dragon  ...  120.0 102.0 97.0 95.0 50.0 NaN 50.0 42.0 48.0
Electric  ...  135.0 95.0 116.0 109.0 45.0 35.0 65.0 45.0 40.0

5
<table>
<thead>
<tr>
<th>Type</th>
<th>Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fairy</td>
<td>NaN</td>
</tr>
<tr>
<td>Fighting</td>
<td>100.0</td>
</tr>
<tr>
<td>Fire</td>
<td>100.0</td>
</tr>
<tr>
<td>Flying</td>
<td>NaN</td>
</tr>
<tr>
<td>Ghost</td>
<td>75.0</td>
</tr>
<tr>
<td>Grass</td>
<td>145.0</td>
</tr>
<tr>
<td>Ground</td>
<td>100.0</td>
</tr>
<tr>
<td>Ice</td>
<td>100.0</td>
</tr>
<tr>
<td>Normal</td>
<td>125.0</td>
</tr>
<tr>
<td>Poison</td>
<td>65.0</td>
</tr>
<tr>
<td>Psychic</td>
<td>180.0</td>
</tr>
<tr>
<td>Rock</td>
<td>75.0</td>
</tr>
<tr>
<td>Steel</td>
<td>110.0</td>
</tr>
<tr>
<td>Water</td>
<td>105.0</td>
</tr>
</tbody>
</table>

### test if heatmap would look ok
```python
sns.heatmap(get_summary(['Attack'], 'max'))
```

[8]: <AxesSubplot:xlabel='None-None-Generation', ylabel='Type 1'>
more pivot

[9]: #make a function that will aggregate my data based on whatever aggfun I want

```python
get_summary(['Attack'], 'min')
```

[9]:

<table>
<thead>
<tr>
<th>min Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generation</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>Type 1</td>
</tr>
<tr>
<td>Bug</td>
</tr>
<tr>
<td>Dark</td>
</tr>
<tr>
<td>Dragon</td>
</tr>
<tr>
<td>Electric</td>
</tr>
<tr>
<td>Fairy</td>
</tr>
<tr>
<td>Fighting</td>
</tr>
<tr>
<td>Fire</td>
</tr>
<tr>
<td>Flying</td>
</tr>
<tr>
<td>Ghost</td>
</tr>
<tr>
<td>Grass</td>
</tr>
</tbody>
</table>
Ground  50.0  60.0  40.0  72.0  66.0  NaN
Ice     50.0  30.0  40.0  60.0  50.0  69.0
Normal  5.0  10.0  20.0  5.0  50.0  36.0
Poison  45.0  90.0  43.0  50.0  50.0  60.0
Psychic 20.0  33.0  23.0  25.0  25.0  48.0
Rock    40.0  64.0  41.0  42.0  75.0  50.0
Steel   NaN  80.0  55.0  24.0  55.0  50.0
Water   10.0  20.0  15.0  20.0  40.0  53.0

[10]:  

```python
# make a function that will aggregate my data based on whatever aggfun I want

get_summary(['Attack','Speed'],['min','max'])
```

[10]:

<table>
<thead>
<tr>
<th>Type 1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug</td>
<td>155.0</td>
<td>185.0</td>
<td>90.0</td>
<td>94.0</td>
<td>135.0</td>
<td>52.0</td>
<td>20.0</td>
<td>10.0</td>
<td>30.0</td>
<td>25.0</td>
</tr>
<tr>
<td>Dark</td>
<td>NaN</td>
<td>95.0</td>
<td>150.0</td>
<td>125.0</td>
<td>125.0</td>
<td>131.0</td>
<td>NaN</td>
<td>60.0</td>
<td>55.0</td>
<td>90.0</td>
</tr>
<tr>
<td>Dragon</td>
<td>134.0</td>
<td>NaN</td>
<td>180.0</td>
<td>170.0</td>
<td>170.0</td>
<td>100.0</td>
<td>64.0</td>
<td>NaN</td>
<td>70.0</td>
<td>70.0</td>
</tr>
<tr>
<td>Electric</td>
<td>90.0</td>
<td>95.0</td>
<td>75.0</td>
<td>123.0</td>
<td>115.0</td>
<td>58.0</td>
<td>30.0</td>
<td>40.0</td>
<td>40.0</td>
<td>45.0</td>
</tr>
<tr>
<td>Fairy</td>
<td>70.0</td>
<td>120.0</td>
<td>NaN</td>
<td>50.0</td>
<td>NaN</td>
<td>131.0</td>
<td>45.0</td>
<td>20.0</td>
<td>NaN</td>
<td>50.0</td>
</tr>
<tr>
<td>Fighting</td>
<td>130.0</td>
<td>95.0</td>
<td>120.0</td>
<td>145.0</td>
<td>140.0</td>
<td>124.0</td>
<td>80.0</td>
<td>35.0</td>
<td>40.0</td>
<td>70.0</td>
</tr>
<tr>
<td>Fire</td>
<td>130.0</td>
<td>130.0</td>
<td>160.0</td>
<td>104.0</td>
<td>140.0</td>
<td>110.0</td>
<td>41.0</td>
<td>40.0</td>
<td>60.0</td>
<td>50.0</td>
</tr>
<tr>
<td>Flying</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>115.0</td>
<td>70.0</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>Ghost</td>
<td>65.0</td>
<td>60.0</td>
<td>165.0</td>
<td>120.0</td>
<td>55.0</td>
<td>110.0</td>
<td>35.0</td>
<td>60.0</td>
<td>40.0</td>
<td>50.0</td>
</tr>
<tr>
<td>Grass</td>
<td>105.0</td>
<td>82.0</td>
<td>130.0</td>
<td>132.0</td>
<td>98.0</td>
<td>107.0</td>
<td>40.0</td>
<td>30.0</td>
<td>40.0</td>
<td>30.0</td>
</tr>
<tr>
<td>Ground</td>
<td>130.0</td>
<td>120.0</td>
<td>180.0</td>
<td>140.0</td>
<td>145.0</td>
<td>NaN</td>
<td>50.0</td>
<td>60.0</td>
<td>40.0</td>
<td>72.0</td>
</tr>
<tr>
<td>Ice</td>
<td>85.0</td>
<td>100.0</td>
<td>120.0</td>
<td>130.0</td>
<td>110.0</td>
<td>117.0</td>
<td>50.0</td>
<td>30.0</td>
<td>40.0</td>
<td>60.0</td>
</tr>
<tr>
<td>Normal</td>
<td>125.0</td>
<td>130.0</td>
<td>160.0</td>
<td>160.0</td>
<td>128.0</td>
<td>80.0</td>
<td>5.0</td>
<td>10.0</td>
<td>20.0</td>
<td>5.0</td>
</tr>
<tr>
<td>Poison</td>
<td>105.0</td>
<td>90.0</td>
<td>100.0</td>
<td>106.0</td>
<td>95.0</td>
<td>75.0</td>
<td>45.0</td>
<td>90.0</td>
<td>43.0</td>
<td>50.0</td>
</tr>
<tr>
<td>Psychic</td>
<td>190.0</td>
<td>100.0</td>
<td>180.0</td>
<td>165.0</td>
<td>100.0</td>
<td>160.0</td>
<td>20.0</td>
<td>33.0</td>
<td>23.0</td>
<td>25.0</td>
</tr>
<tr>
<td>Rock</td>
<td>135.0</td>
<td>164.0</td>
<td>125.0</td>
<td>165.0</td>
<td>140.0</td>
<td>160.0</td>
<td>40.0</td>
<td>64.0</td>
<td>41.0</td>
<td>42.0</td>
</tr>
<tr>
<td>Steel</td>
<td>NaN</td>
<td>125.0</td>
<td>145.0</td>
<td>120.0</td>
<td>100.0</td>
<td>150.0</td>
<td>NaN</td>
<td>80.0</td>
<td>55.0</td>
<td>24.0</td>
</tr>
<tr>
<td>Water</td>
<td>155.0</td>
<td>105.0</td>
<td>150.0</td>
<td>120.0</td>
<td>108.0</td>
<td>95.0</td>
<td>10.0</td>
<td>20.0</td>
<td>15.0</td>
<td>20.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type 1</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug</td>
<td>160.0</td>
<td>95.0</td>
<td>145.0</td>
<td>89.0</td>
<td>25.0</td>
<td>5.0</td>
<td>15.0</td>
<td>25.0</td>
<td>20.0</td>
</tr>
<tr>
<td>Dark</td>
<td>115.0</td>
<td>125.0</td>
<td>106.0</td>
<td>99.0</td>
<td>NaN</td>
<td>65.0</td>
<td>20.0</td>
<td>71.0</td>
<td>38.0</td>
</tr>
<tr>
<td>Dragon</td>
<td>120.0</td>
<td>102.0</td>
<td>97.0</td>
<td>95.0</td>
<td>50.0</td>
<td>NaN</td>
<td>50.0</td>
<td>42.0</td>
<td>48.0</td>
</tr>
</tbody>
</table>
### Generation 6

<table>
<thead>
<tr>
<th>Type</th>
<th>Bug</th>
<th>Dark</th>
<th>Dragon</th>
<th>Electric</th>
<th>Fairy</th>
<th>Fighting</th>
<th>Fire</th>
<th>Flying</th>
<th>Ghost</th>
<th>Grass</th>
<th>Ground</th>
<th>Ice</th>
<th>Normal</th>
<th>Poison</th>
<th>Psychic</th>
<th>Rock</th>
<th>Steel</th>
<th>Water</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>29.0</td>
<td>45.0</td>
<td>40.0</td>
<td>70.0</td>
<td>23.0</td>
<td>43.0</td>
<td>60.0</td>
<td>55.0</td>
<td>38.0</td>
<td>38.0</td>
<td>38.0</td>
<td>NaN</td>
<td>28.0</td>
<td>30.0</td>
<td>68.0</td>
<td>46.0</td>
<td>28.0</td>
<td>44.0</td>
</tr>
</tbody>
</table>

[18 rows x 24 columns]

more on groupby–Skip in class–

We can accomplish the same results with groupby but in my opinion it is less intuitive than pivot table. I wrote the code below to introduce you to group by, it is by no means comprehensive but
it will give you a place to start.

```python
## we can achieve something similar with groupby
## in my experience I use pivot tables when I want one agg func and groupby when I want multiple agg funcs
## you can achieve the same level of data analysis with pivot and groupby, you just have to find what works for you

def get_summary2(stat_wanted, aggfun_wanted):
    #step 1, aggregate by type 1 and generation
    data_subset = pokemon.groupby(['Type 1', 'Generation']).agg(aggfun_wanted)
    #step 2 select the column with the stat we want
    data_subset = data_subset[stat_wanted].agg(aggfun_wanted)
    # clean up the index
    data_subset = data_subset[stat_wanted].agg(aggfun_wanted)
    return data_subset

get_summary2('Attack', ['mean'])
```

```
  Type 1 Generation  mean
  ---  --------  ----
   Bug   1  76.428571
         2   85.416667
         3   55.833333
         4   62.600000
         5   77.611111
         ...
   Water  2   68.111111
           3   80.666667
           4   72.461538
           5   73.277778
           6   68.000000

[98 rows x 1 columns]
```

```python
## we can achieve something similar with groupby

def get_summary2(stat_wanted, aggfun_wanted):
    #note that with groupby you can include more columns in stats wanted and more
    #aggfunc arguments
    data_subset = pokemon.groupby(['Type 1', 'Generation'])[stat_wanted].agg(aggfun_wanted)
    return data_subset

def get_summary2(['Attack', 'Speed'], ['max', 'min'])
```

### Attack Speed

<table>
<thead>
<tr>
<th>Type 1</th>
<th>Generation</th>
<th>max</th>
<th>min</th>
<th>max</th>
<th>min</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug</td>
<td>1</td>
<td>155</td>
<td>20</td>
<td>145</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>185</td>
<td>10</td>
<td>95</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>90</td>
<td>30</td>
<td>160</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>94</td>
<td>25</td>
<td>95</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>135</td>
<td>40</td>
<td>145</td>
<td>20</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Water</td>
<td>2</td>
<td>105</td>
<td>20</td>
<td>85</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>150</td>
<td>15</td>
<td>105</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>120</td>
<td>20</td>
<td>115</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>108</td>
<td>40</td>
<td>108</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>95</td>
<td>53</td>
<td>122</td>
<td>44</td>
</tr>
</tbody>
</table>

[98 rows x 4 columns]

more info on unstack

```python
## we can achieve something similar with groupby
## and groupby allows us to have more aggfunctions that pivot table

```python
def get_summary2(stat_wanted, aggfun_wanted):
    ##note that with groupby you can include more columns in stats wanted and more
    aggfun arguments
    ##however the resulting dataframe has a multi index that makes plotting more
difficult
    ## we can go back to one index by doing unstack
    data_subset = pokemon.groupby(['Type 1', 'Generation'])[stat_wanted].
    agg(aggfun_wanted).unstack()
    return data_subset

get_summary2(['Attack', 'Speed'], ['max', 'min'])
```

```markdown
## we can achieve something similar with groupby
## and groupby allows us to have more aggfunctions that pivot table

```python
def get_summary2(stat_wanted, aggfun_wanted):
    ##note that with groupby you can include more columns in stats wanted and more
    aggfun arguments
    ##however the resulting dataframe has a multi index that makes plotting more
difficult
    ## we can go back to one index by doing unstack
    data_subset = pokemon.groupby(['Type 1', 'Generation'])[stat_wanted].
    agg(aggfun_wanted).unstack()
    return data_subset

get_summary2(['Attack', 'Speed'], ['max', 'min'])
```

### more info on unstack
<table>
<thead>
<tr>
<th></th>
<th>Speed</th>
<th>max</th>
<th>min</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Type 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bug</td>
<td>160.0</td>
<td>95.0</td>
<td>45.0</td>
</tr>
<tr>
<td>Dark</td>
<td>115.0</td>
<td>125.0</td>
<td>106.0</td>
</tr>
<tr>
<td>Dragon</td>
<td>120.0</td>
<td>102.0</td>
<td>97.0</td>
</tr>
<tr>
<td>Electric</td>
<td>135.0</td>
<td>95.0</td>
<td>116.0</td>
</tr>
<tr>
<td>Fairy</td>
<td>NaN</td>
<td>80.0</td>
<td>45.0</td>
</tr>
<tr>
<td>Fighting</td>
<td>100.0</td>
<td>121.0</td>
<td>118.0</td>
</tr>
<tr>
<td>Fire</td>
<td>100.0</td>
<td>108.0</td>
<td>101.0</td>
</tr>
<tr>
<td>Flying</td>
<td>NaN</td>
<td>121.0</td>
<td>123.0</td>
</tr>
<tr>
<td>Ghost</td>
<td>75.0</td>
<td>80.0</td>
<td>80.0</td>
</tr>
<tr>
<td>Grass</td>
<td>145.0</td>
<td>127.0</td>
<td>116.0</td>
</tr>
<tr>
<td>Ground</td>
<td>100.0</td>
<td>105.0</td>
<td>101.0</td>
</tr>
<tr>
<td>Ice</td>
<td>100.0</td>
<td>110.0</td>
<td>105.0</td>
</tr>
<tr>
<td>Normal</td>
<td>125.0</td>
<td>135.0</td>
<td>128.0</td>
</tr>
<tr>
<td>Poison</td>
<td>65.0</td>
<td>95.0</td>
<td>75.0</td>
</tr>
<tr>
<td>Psychic</td>
<td>180.0</td>
<td>115.0</td>
<td>114.0</td>
</tr>
<tr>
<td>Rock</td>
<td>75.0</td>
<td>58.0</td>
<td>110.0</td>
</tr>
<tr>
<td>Steel</td>
<td>110.0</td>
<td>90.0</td>
<td>108.0</td>
</tr>
<tr>
<td>Water</td>
<td>105.0</td>
<td>115.0</td>
<td>108.0</td>
</tr>
</tbody>
</table>

**Generation 6**

**Type 1**

- **Bug**: 29.0
- **Dark**: 45.0
- **Dragon**: 40.0
- **Electric**: 70.0
- **Fairy**: 23.0
- **Fighting**: 43.0
- **Fire**: 60.0
- **Flying**: 55.0
- **Ghost**: 38.0
- **Grass**: 38.0
<table>
<thead>
<tr>
<th>Type</th>
<th>Attack</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ground</td>
<td>NaN</td>
<td></td>
</tr>
<tr>
<td>Ice</td>
<td>28.0</td>
<td></td>
</tr>
<tr>
<td>Normal</td>
<td>57.0</td>
<td></td>
</tr>
<tr>
<td>Poison</td>
<td>30.0</td>
<td></td>
</tr>
<tr>
<td>Psychic</td>
<td>68.0</td>
<td></td>
</tr>
<tr>
<td>Rock</td>
<td>46.0</td>
<td></td>
</tr>
<tr>
<td>Steel</td>
<td>28.0</td>
<td></td>
</tr>
<tr>
<td>Water</td>
<td>44.0</td>
<td></td>
</tr>
</tbody>
</table>

[18 rows x 24 columns]

14.2 Automate graph making

I really want to create a way to visualize the different stats and figure out what the strongest/weakest pokemon types and pokemon are.

STEP1. I want to make many heatplots to view diff stats so I want to write my code below in such
a way that will allow me to do that.

```python
for place_in_list, i in enumerate(pokemon.columns):
    print (place_in_list, i)
```

<table>
<thead>
<tr>
<th></th>
<th>Number</th>
<th>Name</th>
<th>Type 1</th>
<th>Type 2</th>
<th>Total</th>
<th>HP</th>
<th>Attack</th>
<th>Defense</th>
<th>Sp. Atk</th>
<th>Sp. Def</th>
<th>Speed</th>
<th>Generation</th>
<th>Legendary</th>
</tr>
</thead>
</table>

```python
## let's make a list that will hold the variables I want
## when I change the contents of my list the rest of my code will react accordingly
stats_all=['HP', 'Attack', 'Speed', 'Total']

## call a figure object that always has 1 row but the columns change based on the contents of my list
fig, ax=plt.subplots(nrows=1, ncols=len(stats_all), sharey=True)

# change the figure size by the length of my list so my figure is always pretty
fig.set_size_inches(len(stats_all)*4, 6)

for count, stat in enumerate(stats_all):
    hold=get_summary(stat, 'max')
    sns.heatmap(hold, ax=ax[count], cmap="Reds")
    ax[count].set_ylabel('')
    ax[count].set_title(f'{stat}')
```
### 17: def get_heat(stats_all, aggfunction_wanted):

```python
## let's make a list that will hold the variables I want
## when I change the contents of my list the rest of my code will react accordingly

## call a figure object that always has 1 row but the columns change based on the contents of my list
fig, ax = plt.subplots(nrows=1, ncols=len(stats_all), sharey=True)

# change the figure size by the length of my list so my figure is always pretty
fig.set_size_inches(len(stats_all)*4, 6)

for count, stat in enumerate(stats_all):
    # print (count, stat)
    hold = get_summary(stat, aggfunction_wanted)
    sns.heatmap(hold, ax=ax[count], cmap="Reds")
    ax[count].set_ylabel('')
    ax[count].set_title(f'{stat}')
```

```python
plt.show()
```

### 18: # cool now we can explore the different pokemon stats based on diff math stats
get_heat(stats_all=['HP', 'Attack', 'Speed', 'Total', 'Sp. Atk'], aggfunction_wanted='max')
## make a function that quickly lets me view pokemons based on Type 1 and Generation

```python
def get_pokemon(type_1, generation, stat_wanted):
    out = pokemon.loc[((pokemon['Type 1'] == type_1) & (pokemon['Generation'] == generation))].copy()
    out.sort_values(by=stat_wanted, ascending=False, inplace=True)
    return out

get_pokemon('Psychic', 1, 'Attack')
```

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Type 1</th>
<th>Type 2</th>
<th>Total</th>
<th>HP</th>
<th>Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>163</td>
<td>MewtwoMega</td>
<td>Psychic</td>
<td>Fighting</td>
<td>780</td>
<td>106</td>
<td>190</td>
</tr>
<tr>
<td>164</td>
<td>MewtwoMega</td>
<td>Mewtwo X</td>
<td>Psychic</td>
<td>780</td>
<td>106</td>
<td>150</td>
</tr>
<tr>
<td>162</td>
<td>Mewtwo</td>
<td>Psychic</td>
<td>NaN</td>
<td>680</td>
<td>106</td>
<td>110</td>
</tr>
<tr>
<td>165</td>
<td>Mew</td>
<td>Psychic</td>
<td>NaN</td>
<td>600</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>105</td>
<td>Hypno</td>
<td>Psychic</td>
<td>NaN</td>
<td>483</td>
<td>85</td>
<td>73</td>
</tr>
<tr>
<td>70</td>
<td>Alakazam</td>
<td>Psychic</td>
<td>NaN</td>
<td>500</td>
<td>55</td>
<td>50</td>
</tr>
<tr>
<td>71</td>
<td>AlakazamMega</td>
<td>Alakazam</td>
<td>Psychic</td>
<td>590</td>
<td>55</td>
<td>50</td>
</tr>
<tr>
<td>104</td>
<td>Drowzee</td>
<td>Psychic</td>
<td>NaN</td>
<td>328</td>
<td>60</td>
<td>48</td>
</tr>
<tr>
<td>131</td>
<td>Mr. Mime</td>
<td>Psychic</td>
<td>Fairy</td>
<td>460</td>
<td>40</td>
<td>45</td>
</tr>
<tr>
<td>69</td>
<td>Kadabra</td>
<td>Psychic</td>
<td>NaN</td>
<td>400</td>
<td>40</td>
<td>35</td>
</tr>
<tr>
<td>68</td>
<td>Abra</td>
<td>Psychic</td>
<td>NaN</td>
<td>310</td>
<td>25</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Type 1</th>
<th>Type 2</th>
<th>Total</th>
<th>HP</th>
<th>Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>163</td>
<td>MewtwoMega</td>
<td>Psychic</td>
<td>Fighting</td>
<td>780</td>
<td>106</td>
<td>190</td>
</tr>
<tr>
<td>164</td>
<td>MewtwoMega</td>
<td>Mewtwo X</td>
<td>Psychic</td>
<td>780</td>
<td>106</td>
<td>150</td>
</tr>
<tr>
<td>162</td>
<td>Mewtwo</td>
<td>Psychic</td>
<td>NaN</td>
<td>680</td>
<td>106</td>
<td>110</td>
</tr>
<tr>
<td>165</td>
<td>Mew</td>
<td>Psychic</td>
<td>NaN</td>
<td>600</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>105</td>
<td>Hypno</td>
<td>Psychic</td>
<td>NaN</td>
<td>483</td>
<td>85</td>
<td>73</td>
</tr>
<tr>
<td>70</td>
<td>Alakazam</td>
<td>Psychic</td>
<td>NaN</td>
<td>500</td>
<td>55</td>
<td>50</td>
</tr>
<tr>
<td>71</td>
<td>AlakazamMega</td>
<td>Alakazam</td>
<td>Psychic</td>
<td>590</td>
<td>55</td>
<td>50</td>
</tr>
<tr>
<td>104</td>
<td>Drowzee</td>
<td>Psychic</td>
<td>NaN</td>
<td>328</td>
<td>60</td>
<td>48</td>
</tr>
<tr>
<td>131</td>
<td>Mr. Mime</td>
<td>Psychic</td>
<td>Fairy</td>
<td>460</td>
<td>40</td>
<td>45</td>
</tr>
<tr>
<td>69</td>
<td>Kadabra</td>
<td>Psychic</td>
<td>NaN</td>
<td>400</td>
<td>40</td>
<td>35</td>
</tr>
<tr>
<td>68</td>
<td>Abra</td>
<td>Psychic</td>
<td>NaN</td>
<td>310</td>
<td>25</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number</th>
<th>Sp. Atk</th>
<th>Sp. Def</th>
<th>Speed</th>
<th>Generation</th>
<th>Legendary</th>
</tr>
</thead>
<tbody>
<tr>
<td>163</td>
<td>154</td>
<td>130</td>
<td>100</td>
<td>1</td>
<td>True</td>
</tr>
<tr>
<td>164</td>
<td>194</td>
<td>120</td>
<td>140</td>
<td>1</td>
<td>True</td>
</tr>
<tr>
<td>162</td>
<td>154</td>
<td>90</td>
<td>130</td>
<td>1</td>
<td>True</td>
</tr>
<tr>
<td>165</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>1</td>
<td>False</td>
</tr>
<tr>
<td>105</td>
<td>73</td>
<td>67</td>
<td>115</td>
<td>1</td>
<td>False</td>
</tr>
<tr>
<td>70</td>
<td>135</td>
<td>95</td>
<td>120</td>
<td>1</td>
<td>False</td>
</tr>
<tr>
<td>71</td>
<td>175</td>
<td>95</td>
<td>150</td>
<td>1</td>
<td>False</td>
</tr>
<tr>
<td>104</td>
<td>43</td>
<td>42</td>
<td>90</td>
<td>1</td>
<td>False</td>
</tr>
</tbody>
</table>
Conclusion:

Pandas will help you clean, order and analyze your data. Then you can use libraries like seaborn so make figures for your manuscripts.

Google and stackoverflow are your friends, use them! Also YOUTUBE is a gold mine. This is on of my favorite channels but explore and find your own!

**RUN THE REST OF CODE AND SHOW GRAPHS**

15. Extra: Combination skills, Correlations

Scatter plots

I want to know what stats values are highly correlated. So I am going to use `sns.regplot`

```python
[20]: # not specifying how many subplots I want will allow me to add subplots with
    # the for loop
    fig = plt.figure()

    fig.subplots_adjust(hspace=0.2, wspace=0.2)

    # this list can change but it must always change by the nrow value you specify
    # in the for loop
    fig.set_size_inches(len(stats_all*4),10)

    for count,stat in enumerate(stats_all):
        ax = fig.add_subplot(2, 3, count+1)
        ax=sns.regplot(x='Total', y=stat, data=pokemon)

    plt.show()
    plt.close()
```
Cool figure, let’s add some labels so we can read it better

```python
[21]: fig = plt.figure()

z = len(stats_all)/2  # this variable is classified as a floating point number, we need to change it to an integer.

fig.subplots_adjust(hspace=0.3, wspace=0.2)

fig.set_size_inches(len(stats_all*4),10)

for count,stat in enumerate(stats_all):
    ax = fig.add_subplot(2, int(z), count+1)
    ax=sns.regplot(x='Total', y=stat, data=pokemon)

## get stats for a label
temp=pokemon[['Total',stat]].dropna()
results=stats.linregress(temp.iloc[:,0],temp.iloc[:,1])
# print (results)

## this code makes the labels in the box
props=dict(boxstyle='round',facecolor='white',alpha=.9)
textstr='m={:.3f}
b={:.3f}
$r^2$={:.3f}
p={:.3}'.format(results[0],results[1],results[2]**2,results[3])
ax.text(.75,.05,textstr,transform=ax.transAxes,va='bottom',fontsize=11,bbox=props) # change the formatting of the box

# this code here allows me to make a line manually
```
Let’s make the code into a function so we can easily change what we plot

```python
def correlates_with_wanted(stats_wanted):
    fig = plt.figure()
    fig.subplots_adjust(hspace=0.3, wspace=0.2)

    stats_all=['HP','Attack','Defense','Speed','Sp. Atk','Sp. Def','Total']
    stats_all.remove(stats_wanted)
    fig.set_size_inches(len(stats_all)+4,10)

    for count,stat in enumerate(stats_all):
        # remember that add_subplots is (nrow,ncol,number) and number starts at 1
        # while enumerate at 0
        ax = fig.add_subplot(2, int(z), count+1)
        ax=sns.regplot(x=stats_wanted, y=stat, data=pokemon)

        # get stats for a label
```
temp=pokemon[[stats_wanted,stat]].dropna()
results=stats.linregress(temp.iloc[:,0],temp.iloc[:,1])
# print(results)

props=dict(boxstyle='round',facecolor='white',alpha=.9)
textstr='m={:.3f}
b={:.3f}
$r^2$={:.3f}
p={:.3}
'.format(results[0],results[1],results[2]**2,results[3])
stats_out=ax.text(.75,.05,textstr,transform=ax.transAxes,va='bottom',fontsize=11,bbox=props)
# change the formatting of the box

# this code here would allow to make a line manually
x1=np.array([temp[stats_wanted].min(),temp[stats_wanted].max()])
y1=results[0]*x1+results[1]
ax.plot(x1,y1)

# add line equation as a title
m=results[0]
b=results[1]
ax.set_title('y={:.3f}x+{:.3f}'.format(m,b))

plt.show()
plt.close()
correlates_with_wanted('Defense')

### make a table of correlations
stats_all=['HP','Attack','Defense','Speed','Sp. Atk','Sp. Def','Total']
stats_df=pd.DataFrame(index=stats_all,columns=stats_all)
for i in stats_all:
    for k in stats_all:
        stats_results=stats.linregress(pokemon[i],pokemon[k])
        # print(stats_results)
        stats_df[i][k]=stats_results[2].round(2)

stats_df

<table>
<thead>
<tr>
<th></th>
<th>HP</th>
<th>Attack</th>
<th>Defense</th>
<th>Speed</th>
<th>Sp. Atk</th>
<th>Sp. Def</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>HP</td>
<td>1.0</td>
<td>0.42</td>
<td>0.24</td>
<td>0.18</td>
<td>0.36</td>
<td>0.38</td>
<td>0.62</td>
</tr>
<tr>
<td>Attack</td>
<td>0.42</td>
<td>1.0</td>
<td>0.44</td>
<td>0.38</td>
<td>0.4</td>
<td>0.26</td>
<td>0.74</td>
</tr>
</tbody>
</table>
### a faster trick

corr_all=pokemon.corr()
corr_all

Density plots

Once you have code that works all you need to do is change the plot kind and it will work

def density_with_wanted(stats_wanted):
    fig = plt.figure()
    fig.subplots_adjust(hspace=0.3, wspace=0.2)
    stats_all.remove(stats_wanted)
fig.set_size_inches(len(stats_all)*3,10)

for count,stat in enumerate(stats_all):
    # remember that add_subplots is (nrow,ncol,number) and number starts at 1
    while enumerate at 0
        ax = fig.add_subplot(2, int(z), count+1)
        ##change the kind of figure you want
        sns.kdeplot(x=pokemon[stats_wanted], y=pokemon[stat], cmap='Reds',
        shade=True, ax=ax)

plt.show()
plt.close()

density_with_wanted('Speed')

we can make the function a little more complicated by adding in another for loop. This will make it so that each subplots has multiple plots inside.

[59]: sns.set(style="dark")

def density_with_wanted(stats_wanted):
    fig = plt.figure()
    fig.subplots_adjust(hspace=0.3, wspace=0.2)

    stats_all=['HP','Attack','Defense','Speed','Sp. Atk']
stats_all.remove(stats_wanted)
fig.set_size_inches(len(stats_all*3),10)
for count,stat in enumerate(stats_all):
    ax = fig.add_subplot(2, int(z), count+1)
    for gen in [1,2,4,5]:
        pokemon_subset=pokemon.loc[pokemon['Generation']==gen]
        sns.kdeplot(x=pokemon_subset[stats_wanted], y=pokemon_subset[stat],
                    shade=True, ax=ax, thresh=0.05, n_levels=3)
plt.show()
plt.close()
density_with_wanted('Speed')
16. Extra: Data exploration

Once you have cleaned up your datatable it is really easy to plot. You just look for the kind of plot you want and specify the data you want. You can play around with the variables until you get something meaningful.

Pairplots

Like the plots we manually made above, Pairplot allows you to see relationships between variables.

```python
[60]: sns.pairplot(pokemon[['Type_1', 'Type_2', 'Attack', 'Defense', 'Speed', 'Total', 'Legendary']], hue='Legendary')
```

```python
[60]: <seaborn.axisgrid.PairGrid at 0x7f8f9dda95b0>
```

![Pairplot of Pokemon dataset showing relationships between variables.](image)
```python
[61]: sns.pairplot(pokemon[['Type_1', 'Type_2', 'Attack', 'Defense', 'Speed', 'Total', 'Generation']], hue='Generation')

[61]: <seaborn.axisgrid.PairGrid at 0x7f8f9ee74580>

[62]: pokemon['generation_legend'] = pokemon['Generation'].astype(str) + '_' + pokemon['Legendary'].astype(str)
pokemon.head()

[62]:   Number  Name       Type 1   Type 2  Total  HP  Attack  Defense
    0      1 Bulbasaur  Grass  Poison    318  45   49    49
    1      2      Ivysaur  Grass  Poison    405  60   62    63
    2      3    Venusaur  Grass  Poison    525  80   82    83
    3      3 VenusaurMega Venusaur  Grass  Poison    625  80  100   123
    4      4   Charmander   Fire     NaN    309  39    52    43
```
Swarmplots

I wanted to explore the Legendary distribution more so I looked in seaborn to see what kinds of plots are used for categorical data and swarmplots came up.
### this calls the style of the plot
sns.set(style="whitegrid", palette="muted")

### this calls the actual swarmplot
sns.swarmplot(x="Legendary", y="Attack", hue="Generation", data=pokemon)

### you will get a warning that some of the plots can't be placed, this is ok because it's just an example.

/Users/jsenger/opt/anaconda3/lib/python3.8/site-packages/seaborn/categorical.py:1296: UserWarning: 28.2% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
warnings.warn(msg, UserWarning)

/Users/jsenger/opt/anaconda3/lib/python3.8/site-packages/seaborn/categorical.py:1296: UserWarning: 16.3% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
warnings.warn(msg, UserWarning)

/Users/jsenger/opt/anaconda3/lib/python3.8/site-packages/seaborn/categorical.py:1296: UserWarning: 10.0% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
warnings.warn(msg, UserWarning)
warnings.warn(msg, UserWarning)
/Users/jsenger/opt/anaconda3/lib/python3.8/site-packages/seaborn/categorical.py:1296: UserWarning: 15.8% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)

[66]: <AxesSubplot:xlabel='Generation', ylabel='Attack'>

![Diagram of a swarm plot showing the relationship between Generation and Attack, with Legendary status indicated by different marker colors.](swarm.png)

[66]: fig, ax=plt.subplots()
    fig.set_size_inches(5,10)

    sns.swarmplot(x="Legendary", y="Speed", hue="Generation", data=pokemon,ax=ax)
    ax.set_title('Legendary')

    plt.savefig(data_out_directory+'swarm.pdf')

warnings.warn(msg, UserWarning)
/Users/jsenger/opt/anaconda3/lib/python3.8/site-packages/seaborn/categorical.py:1296: UserWarning: 24.4% of the points cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
Coding challenge

Here is the website on how to make violin plots with seaborn.

```python
### using the full pokemon data
### make a violinplot where the x is Attack, y is Type 1, and hue is legendary
## remember to change the figure size so the figure is readable

Answer

```python
#@title
fig, ax=plt.subplots()
fig.set_size_inches(5,15)
sns.violinplot(x="Attack", y="Type 1", data=pokemon, hue='Legendary',ax=ax)
ax.set_title('My violin plots')
```

```python
Text(0.5, 1.0, 'My violin plots')
```
17. Extra fun

THIS IS HOW YOU MAKE PLOTS

1. call figure and create your canvas . . . fig, ax= plt.subplots()
2. follow instructions on websites for the actual plot you want . . . some plot code
3. change some stuff . . . ax.set_title('My plot title')
4. save the plot if you want . . . plt.savefig('My_plot')

cluster with heatmap

more code here

### let's select the legendary and work with those

```
pokemon_legend=pokemon.loc[pokemon['Legendary']==True,]
pokemon_legend
```

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Type 1</th>
<th>Type 2</th>
<th>Total</th>
<th>HP</th>
<th>Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>156</td>
<td>Articuno</td>
<td>Ice</td>
<td>Flying</td>
<td>580</td>
<td>90</td>
<td>85</td>
</tr>
<tr>
<td>157</td>
<td>Zapdos</td>
<td>Electric</td>
<td>Flying</td>
<td>580</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td>158</td>
<td>Moltres</td>
<td>Fire</td>
<td>Flying</td>
<td>580</td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td>162</td>
<td>Mewtwo</td>
<td>Psychic</td>
<td>NaN</td>
<td>680</td>
<td>106</td>
<td>110</td>
</tr>
<tr>
<td>163</td>
<td>MewtwoMega</td>
<td>Psychic</td>
<td>Fighting</td>
<td>780</td>
<td>106</td>
<td>190</td>
</tr>
<tr>
<td>795</td>
<td>Diancie</td>
<td>Rock</td>
<td>Fairy</td>
<td>600</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>796</td>
<td>DiancieMega</td>
<td>Rock</td>
<td>Fairy</td>
<td>700</td>
<td>50</td>
<td>160</td>
</tr>
<tr>
<td>797</td>
<td>HoopaHoopa</td>
<td>Confined</td>
<td>Psychic</td>
<td>Ghost</td>
<td>600</td>
<td>80</td>
</tr>
<tr>
<td>798</td>
<td>HoopaHoopa</td>
<td>Unbound</td>
<td>Psychic</td>
<td>Dark</td>
<td>680</td>
<td>80</td>
</tr>
<tr>
<td>799</td>
<td>Volcanion</td>
<td>Fire</td>
<td>Water</td>
<td>600</td>
<td>80</td>
<td>110</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number</th>
<th>Defense</th>
<th>Sp. Atk</th>
<th>Sp. Def</th>
<th>Speed</th>
<th>Generation</th>
<th>Legendary</th>
<th>generation_legend</th>
</tr>
</thead>
<tbody>
<tr>
<td>156</td>
<td>100</td>
<td>95</td>
<td>125</td>
<td>85</td>
<td>1</td>
<td>True</td>
<td>1_True</td>
</tr>
<tr>
<td>157</td>
<td>85</td>
<td>125</td>
<td>90</td>
<td>100</td>
<td>1</td>
<td>True</td>
<td>1_True</td>
</tr>
<tr>
<td>158</td>
<td>90</td>
<td>125</td>
<td>85</td>
<td>90</td>
<td>1</td>
<td>True</td>
<td>1_True</td>
</tr>
<tr>
<td>162</td>
<td>90</td>
<td>154</td>
<td>90</td>
<td>130</td>
<td>1</td>
<td>True</td>
<td>1_True</td>
</tr>
<tr>
<td>795</td>
<td>150</td>
<td>100</td>
<td>150</td>
<td>50</td>
<td>6</td>
<td>True</td>
<td>6_True</td>
</tr>
<tr>
<td>796</td>
<td>110</td>
<td>160</td>
<td>110</td>
<td>110</td>
<td>6</td>
<td>True</td>
<td>6_True</td>
</tr>
<tr>
<td>797</td>
<td>60</td>
<td>150</td>
<td>130</td>
<td>70</td>
<td>6</td>
<td>True</td>
<td>6_True</td>
</tr>
<tr>
<td>798</td>
<td>60</td>
<td>170</td>
<td>130</td>
<td>80</td>
<td>6</td>
<td>True</td>
<td>6_True</td>
</tr>
<tr>
<td>799</td>
<td>120</td>
<td>130</td>
<td>90</td>
<td>70</td>
<td>6</td>
<td>True</td>
<td>6_True</td>
</tr>
</tbody>
</table>
[65 rows x 14 columns]


[70]: <seaborn.matrix.ClusterGrid at 0x7f8fa1f85340>

Venn Diagrams

I just googled python venn diagrams and searched through some of the pages that came out. more info HERE

[ ]: pip install matplotlib_venn
from matplotlib_venn import venn3

# Make the diagram, code from the website I listed
## I'm just going to add some stuff the website did not show

call figure and create your canvas
fig, ax = plt.subplots()

## follow instructions on website
venn3(subsets = (20, 10, 12, 10, 9, 4, 3), set_labels = ('Group A', 'Group B', 'Group C'), alpha = 0.5, ax=ax)

## change some stuff
ax.set_title('My Venn')

# save the plot if you want.
# plt.savefig('My_Venn')