Sort Profiling

September 1, 2018

```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns

        % matplotlib inline

        sns.set()
```

```
In [2]: data_rand = pd.read_csv('test/results_random.csv')
        data_asc  = pd.read_csv('test/results_ascending.csv')
        data_desc = pd.read_csv('test/results_descending.csv')
```

```
In [3]: data_rand.head()
```

Out[3]:

	iterations	bubble sort	selection sort	insertion sort	count sort	merge sort	quick sort
0	5000	0.081125	0.031936	0.019234	0.000073	0.001063	0.000661
1	10000	0.332168	0.126577	0.079063	0.000111	0.002171	0.001445
2	15000	0.786153	0.285987	0.174660	0.000156	0.003406	0.002185
3	20000	1.432733	0.508159	0.307466	0.000222	0.004559	0.002902
4	25000	2.270886	0.792993	0.486891	0.000274	0.005886	0.004023

Data: Random Element Arrays.

```
In [4]: data_desc.head()
```

Out[4]:

	iterations	bubble sort	selection sort	insertion sort	count sort
0	5000	0.091378	0.033983	0.039397	0.000051
1	10000	0.351127	0.135031	0.155261	0.000118
2	15000	0.784085	0.315804	0.374631	0.000179
3	20000	1.386115	0.538413	0.621597	0.000207
4	25000	2.185639	0.856136	0.982883	0.000266

```
    merge sort   quick sort
0     0.000751     0.056687
1     0.001568     0.230190
2     0.002575     0.509938
3     0.003207     0.904469
4     0.004139     1.421281
```

Data: Decending Order Arrays.

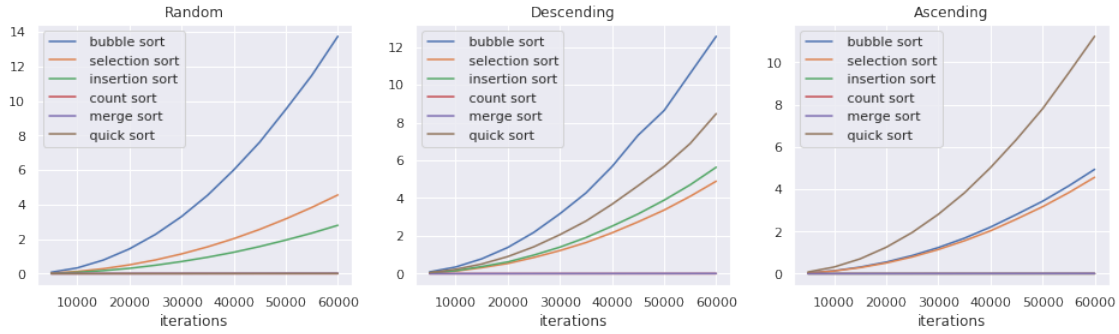
```
In [5]: data_asc.head()
```

```
Out[5]:    iterations   bubble sort   selection sort   insertion sort   count sort  \
        0        5000      0.039028         0.031803         0.000036     0.000053
        1       10000      0.136703         0.126663         0.000042     0.000113
        2       15000      0.309125         0.285911         0.000060     0.000158
        3       20000      0.547984         0.505178         0.000080     0.000216
        4       25000      0.857445         0.791985         0.000099     0.000251

            merge sort   quick sort
        0     0.000753     0.078848
        1     0.001543     0.311870
        2     0.002390     0.699363
        3     0.003186     1.247366
        4     0.003928     1.950618
```

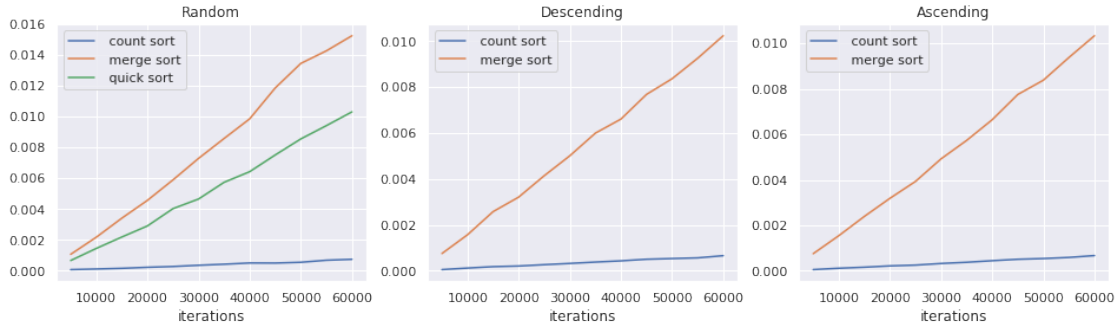
Data: Ascending Order Arrays.

```
In [6]: fig, axes = plt.subplots(1, 3, figsize=(16, 4))
        data_rand.plot(x='iterations', ax=axes[0], title='Random')
        data_desc.plot(x='iterations', ax=axes[1], title='Descending')
        data_asc.plot (x='iterations', ax=axes[2], title='Ascending')
        plt.show()
```



A comparision of iterations vs run-times for all sorts in the above three cases.

```
In [7]: fig, axes = plt.subplots(1, 3, figsize=(16, 4))
        data_rand.plot(x='iterations', y=[4,5,6], ax=axes[0], title='Random')
        data_desc.plot(x='iterations', y=[4,5], ax=axes[1], title='Descending')
        data_asc.plot (x='iterations', y=[4,5], ax=axes[2], title='Ascending')
        plt.show()
```



A magnified view of the faster sorting algoriths in each case.