Plotting

April 8, 2021

1 Plotting Data

Note that you don't necessarily need to use or install Jupyter Notebook to use the codes below.

[2]:
```
import pandas as pd
import numpy as np
from math import log2, e
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
```

1.0.1 Importing CSV files into Dataframes

Using Pandas we import our CSV timing data into dataframes(the data here are for illustration purposes and have been manually created/altered)

[3]:
```
df_almost = pd.read_csv("almost_sorted.csv",header=None, names=["Size",␣
 ↪"Elapsed_Time", "Num_Comp"])
df_random = pd.read_csv("random.csv",header=None, names=["Size",␣
 ↪"Elapsed_Time", "Num_Comp"])
print(df_almost)
```

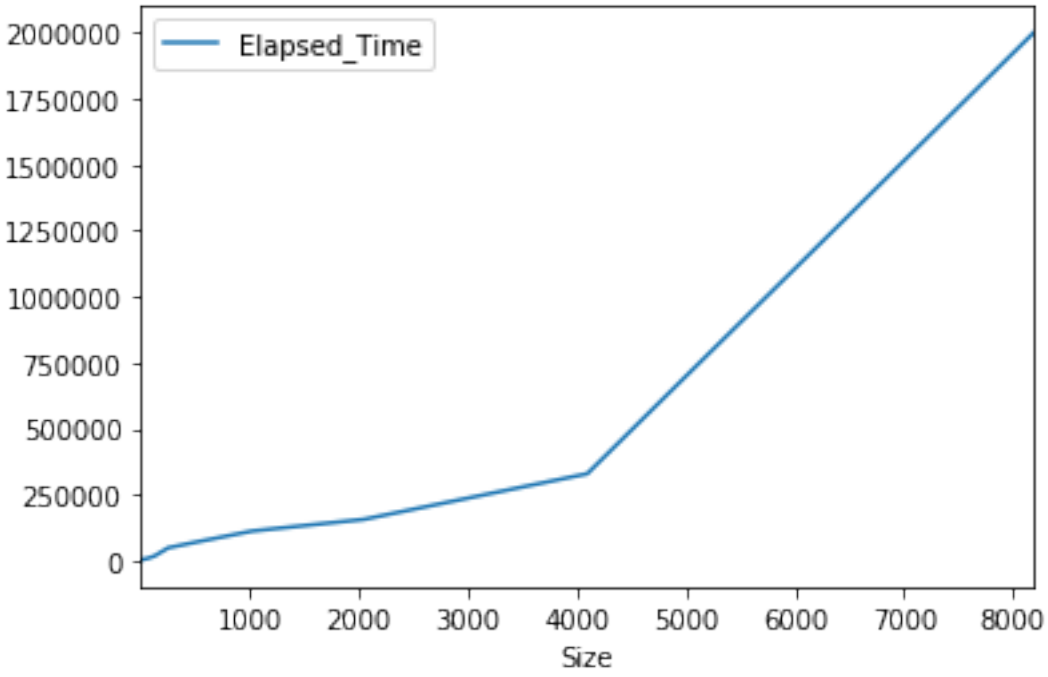
```
    Size  Elapsed_Time  Num_Comp
0      2           389         1
1      4           457         4
2      8          1290        12
3     16          2459        32
4     32          6052        82
5     64          8219       195
6    128         17154       456
7    256         48704      1000
8    512         70342      2000
9   1024        112039      5216
10  2048        156301     11476
11  4096        329085     25048
12  8192       2001364     54226
```

[3]:
```
display(df_almost)
display(df_random)
```

	Size	Elapsed_Time	Num_Comp
0	2	389	1
1	4	457	4
2	8	1290	12
3	16	2459	32
4	32	6052	82
5	64	8219	195
6	128	17154	456
7	256	48704	1000
8	512	70342	2000
9	1024	112039	5216
10	2048	156301	11476
11	4096	329085	25048
12	8192	2001364	54226

	Size	Elapsed_Time	Num_Comp
0	2	241	1
1	4	628	4
2	8	1335	12
3	16	4407	32
4	32	7101	81
5	64	9706	197
6	128	21087	462
7	256	50092	1058
8	512	85224	2387
9	1024	169928	5311
10	2048	290327	11700
11	4096	580078	25547
12	8192	1365788	55390

[4]:
```
df_almost.plot(x="Size", y="Elapsed_Time"); # Use y = "Num_Comp" for comparison␣
↪plot
```

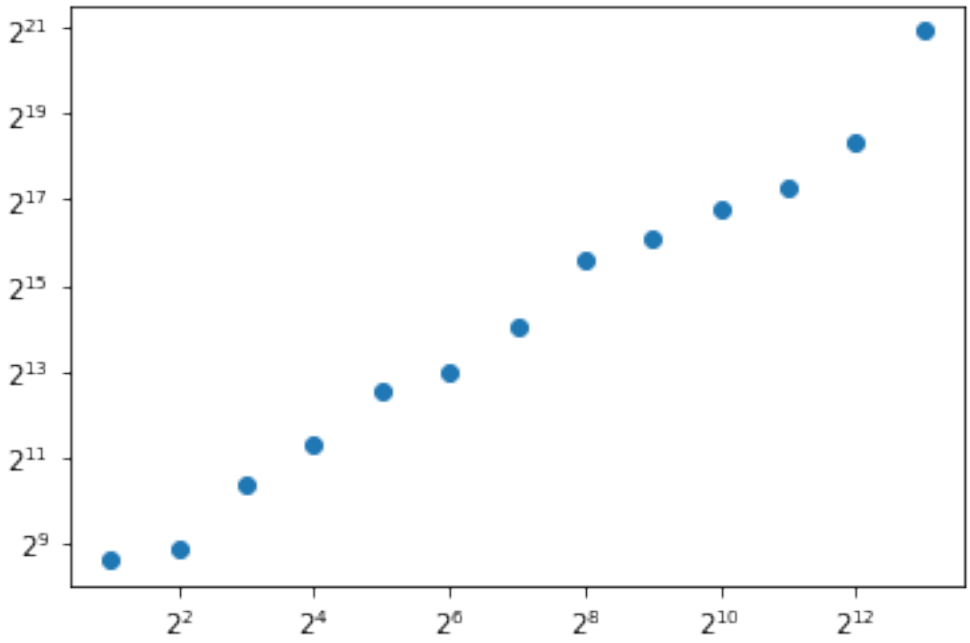


1.1 Log-log Plot

In a log-log plot x- and y-coordinates are in a logarithmic scale on the x-axis and the y-axis.

[5]:
```
x = df_almost['Size']
y = df_almost['Elapsed_Time']

p = plt.loglog(x, y, '.', basex = 2, basey = 2, markersize = 12)
```



1.2 Finding the best fit for the data using Numpy library

[6]:
```
logx, logy = np.log(x), np.log(y) # This is the log with natural base (e)
```

np.polyfit is a least squares polynomial fit function which accepts the data set and a polynomial function of any degree (specified by the user in the third argument), and returns an array of coefficients that minimizes the squared error.

[7]:
```
m, b = np.polyfit(logx, logy, 1) # m is the slope and b is the intercept
```

[8]:
```
m, b
```

[8]:
```
(0.9631385830955153, 5.117953038492062)
```

The numpy.poly1d() function helps to define a polynomial function.

[9]:
```
fit = np.poly1d((m, b)) # log y = m * log x + b
```

[10]:
```
expected_logy = fit(logx)
```

[11]:
```
expected_logy
```

[11]:
```
array([ 5.78554983,  6.45314663,  7.12074342,  7.78834021,  8.45593701,
         9.1235338 ,  9.79113059, 10.45872739, 11.12632418, 11.79392097,
        12.46151777, 13.12911456, 13.79671135])
```

R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.https://en.wikipedia.org/wiki/Coefficient_of_determination

We can use r2_score(y_true, y_pred) from the library sklearn to compute it.

[12]:
```
r2 = r2_score(logy, expected_logy)
r2
```

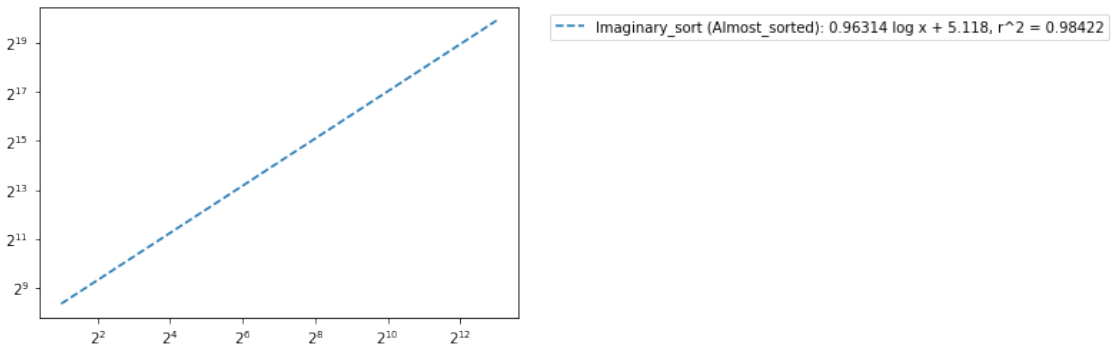
[12]:
```
0.9842176099401816
```

Plotting the log log of regression line:

[13]:
```
sort_name = "Imaginary_sort"
perm_name = "Almost_sorted"

fit_p = plt.loglog(x[::len(x)-1], (e ** expected_logy)[::len(y)-1], '--', basex␣
↪= 2, basey = 2,
label = f'{sort_name} ({perm_name}): {m:0.5} log x + {b:.5}, r^2 = {r2:.5}',
markersize = 6, color = p[-1].get_color())
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left') #to put the legend box␣
↪outside
```

[13]:
```
<matplotlib.legend.Legend at 0x7fc074815c10>
```



Plotting the best fit and the data alongside each other for the almost sorted permutation (both in blue). The orange data points belong to the random permutation (note that plotting two permutations of a particular sort is not required for your project, this is only for learning purposes).

[14]:
```
p = plt.loglog(x, y, '.', basex = 2, basey = 2, markersize = 12)

fit_p = plt.loglog(x[::len(x)-1], (e ** expected_logy)[::len(y)-1], '--', basex␣
↪= 2, basey = 2,
label = f'{sort_name} ({perm_name}): {m:0.5} log x + {b:.5}, error = {r2:.5}',
markersize = 6, color = p[-1].get_color())
```

```
x2 = df_random['Size']
y2 = df_random['Elapsed_Time']
p_random = plt.loglog(x2, y2, '.', basex = 2, basey = 2, markersize = 12,␣
 ↪label= 'Imaginery_sort (Randomomized permutation)')

plt.title("Merge_sort and its best fit line")
plt.xlabel('Input size (n, # of elements)')
plt.ylabel('Elapsed Time')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left') #to put the legend box␣
 ↪outside
```

[14]: <matplotlib.legend.Legend at 0x7fc0748d4210>