E164: Introduction to Biological System Design

Introduction to Biological Data Analysis

Ayush Pandey

If you have not setup your computer to run this notebook, make sure to follow steps here: https://docs.google.com/document/d/1js7XQbjorU5LCIoWfrzkUCXjjQkVcITkT0N8osiy4Go/edit?usp=sharing

The purpose of this interactive Python notebook is to introduce basic data analysis tools available in Python with the help of biological data examples.

If you have installed numpy, scipy, matplotlib, and pandas already, then you are all set to run this notebook.

Data Analysis with Python

Introduction to Pandas

You can think of Pandas as the Python data-management library. It can be used to manage large quantities of data in a structured way so you don't get lost and provides functions to retreive/store/edit your data so that it is easily interfaced with numerical, scientific, and plotting libraries.

More information on Pandas here: https://pandas.pydata.org/pandas-docs/stable/

Data is often stored as .csv files. Here's how to import a CSV file with Pandas and structure the data as desired.

In [4]:

```
import pandas as pd

url = 'https://raw.githubusercontent.com/biocircuits/bioscrape/master/infere
# To import a CSV file, you can use the `read_csv` function.
# Either provide a URL as its input or a string of address to the CSV file
df = pd.read_csv(url)
# You get a "dataframe" object out that contains all of the data in the CSV
```

In an IPYNB (like this one), the Pandas dataframes are nicely formatted:

In [5]:

```
df
```

Out[5]:

	Time	t	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A1
0	0:09:19	9.32	0.086	0.087	0.088	0.095	0.091	0.089	0.090	0.085	0.090	0.090	0.088	0.09
1	0:19:19	19.32	0.086	0.090	0.089	0.090	0.091	0.092	0.091	0.086	0.092	0.092	0.088	0.09
2	0:29:19	29.32	0.086	0.088	0.090	0.089	0.089	0.092	0.093	0.087	0.089	0.093	0.089	0.09
3	0:39:19	39.32	0.088	0.090	0.091	0.090	0.092	0.093	0.096	0.088	0.091	0.093	0.090	0.09
4	0:49:19	49.32	0.089	0.092	0.093	0.092	0.094	0.096	0.099	0.091	0.093	0.094	0.094	0.09
5	0:59:19	59.32	0.091	0.094	0.095	0.096	0.098	0.100	0.103	0.094	0.096	0.098	0.096	0.10
6	1:09:19	69.32	0.094	0.097	0.098	0.099	0.102	0.106	0.109	0.097	0.099	0.103	0.100	0.10
7	1:19:19	79.32	0.097	0.101	0.102	0.105	0.108	0.109	0.116	0.103	0.105	0.111	0.104	0.11
8	1:29:19	89.32	0.101	0.106	0.107	0.110	0.117	0.119	0.124	0.108	0.110	0.114	0.111	0.11
9	1:39:19	99.32	0.106	0.111	0.113	0.117	0.123	0.122	0.133	0.116	0.117	0.122	0.117	0.12
10	1:49:19	109.32	0.111	0.117	0.119	0.124	0.130	0.129	0.142	0.120	0.124	0.128	0.124	0.13
11	1:59:19	119.32	0.126	0.125	0.127	0.133	0.140	0.139	0.154	0.129	0.132	0.136	0.132	0.14
12	2:09:19	129.32	0.128	0.133	0.139	0.142	0.151	0.152	0.167	0.138	0.143	0.149	0.142	0.15
13	2:19:19	139.32	0.133	0.143	0.146	0.153	0.165	0.163	0.183	0.151	0.150	0.157	0.154	0.17
14	2:29:19	149.32	0.147	0.153	0.158	0.164	0.177	0.176	0.201	0.163	0.160	0.168	0.166	0.18
15	2:39:19	159.32	0.160	0.165	0.171	0.179	0.191	0.188	0.216	0.176	0.184	0.181	0.178	0.19
16	2:49:19	169.32	0.171	0.177	0.185	0.196	0.210	0.200	0.236	0.191	0.188	0.195	0.193	0.20
17	2:59:19	179.32	0.184	0.191	0.196	0.209	0.222	0.218	0.257	0.202	0.200	0.215	0.204	0.22
18	3:09:19	189.32	0.186	0.207	0.214	0.227	0.240	0.232	0.280	0.217	0.216	0.227	0.221	0.24
19	3:19:19	199.32	0.202	0.226	0.227	0.248	0.262	0.252	0.303	0.232	0.239	0.245	0.239	0.25
20	3:29:19	209.32	0.228	0.248	0.248	0.273	0.283	0.273	0.327	0.253	0.259	0.271	0.259	0.27
21	3:39:19	219.32	0.236	0.270	0.277	0.294	0.303	0.300	0.345	0.275	0.280	0.290	0.278	0.30
22	3:49:19	229.32	0.266	0.289	0.299	0.314	0.323	0.317	0.361	0.297	0.297	0.310	0.301	0.32
23	3:59:19	239.32	0.286	0.308	0.313	0.333	0.336	0.333	0.373	0.319	0.319	0.325	0.319	0.33
24	4:09:19	249.32	0.303	0.325	0.328	0.350	0.356	0.350	0.388	0.336	0.334	0.339	0.331	0.35
25	4:19:19	259.32	0.319	0.340	0.354	0.364	0.364	0.365	0.397	0.348	0.347	0.353	0.346	0.36
26	4:29:19	269.32	0.335	0.354	0.366	0.375	0.377	0.372	0.410	0.361	0.359	0.366	0.358	0.38
27	4:39:19	279.32	0.343	0.368	0.373	0.385	0.389	0.381	0.419	0.368	0.368	0.371	0.370	0.39
28	4:49:19	289.32	0.356	0.381	0.388	0.394	0.398	0.395	0.423	0.379	0.379	0.381	0.377	0.40
29	4:59:19	299.32	0.370	0.392	0.403	0.405	0.414	0.401	0.439	0.393	0.388	0.396	0.389	0.41
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
43	7:19:19	439.32	0.498	0.506	0.543	0.533	0.514	0.506	0.549	0.504	0.508	0.508	0.508	0.54
44	7:29:19	449.32	0.500	0.508	0.547	0.537	0.517	0.511	0.553	0.507	0.511	0.510	0.510	0.55

	Time	t	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A1
45	7:39:19	459.32	0.508	0.512	0.561	0.541	0.521	0.515	0.557	0.512	0.514	0.514	0.514	0.55
46	7:49:19	469.32	0.499	0.515	0.562	0.545	0.523	0.518	0.561	0.514	0.518	0.517	0.517	0.55
47	7:59:19	479.32	0.504	0.517	0.558	0.549	0.528	0.523	0.564	0.519	0.523	0.522	0.521	0.56
48	8:09:19	489.32	0.505	0.521	0.561	0.552	0.532	0.525	0.568	0.525	0.526	0.525	0.525	0.56
49	8:19:19	499.32	0.510	0.525	0.565	0.556	0.536	0.528	0.571	0.526	0.529	0.527	0.529	0.56
50	8:29:19	509.32	0.512	0.527	0.569	0.561	0.538	0.531	0.576	0.529	0.531	0.530	0.531	0.57
51	8:39:19	519.32	0.515	0.531	0.573	0.564	0.542	0.535	0.578	0.534	0.535	0.534	0.534	0.57
52	8:49:19	529.32	0.517	0.534	0.576	0.567	0.543	0.537	0.581	0.535	0.538	0.536	0.537	0.57
53	8:59:19	539.32	0.519	0.537	0.579	0.570	0.545	0.540	0.584	0.539	0.541	0.538	0.540	0.58
54	9:09:19	549.32	0.520	0.539	0.582	0.573	0.549	0.543	0.587	0.541	0.543	0.542	0.542	0.58
55	9:19:19	559.32	0.522	0.541	0.586	0.576	0.551	0.545	0.590	0.543	0.546	0.543	0.545	0.58
56	9:29:19	569.32	0.525	0.543	0.589	0.580	0.553	0.547	0.593	0.545	0.548	0.546	0.547	0.58
57	9:39:19	579.32	0.528	0.546	0.592	0.582	0.555	0.550	0.595	0.549	0.552	0.549	0.549	0.58
58	9:49:19	589.32	0.530	0.548	0.595	0.585	0.558	0.552	0.598	0.553	0.554	0.551	0.552	0.59
59	9:59:19	599.32	0.532	0.549	0.597	0.587	0.559	0.554	0.600	0.553	0.555	0.553	0.554	0.59
60	10:09:19	609.32	0.533	0.551	0.600	0.590	0.561	0.555	0.602	0.558	0.556	0.555	0.556	0.59
61	10:19:19	619.32	0.534	0.554	0.602	0.591	0.563	0.558	0.604	0.559	0.560	0.558	0.558	0.59
62	10:29:19	629.32	0.536	0.556	0.604	0.593	0.565	0.560	0.606	0.561	0.562	0.559	0.561	0.59
63	10:39:19	639.32	0.538	0.556	0.606	0.595	0.565	0.560	0.608	0.564	0.563	0.561	0.561	0.60
64	10:49:19	649.32	0.538	0.558	0.608	0.598	0.569	0.562	0.610	0.565	0.565	0.563	0.563	0.60
65	10:59:19	659.32	0.541	0.559	0.609	0.600	0.568	0.563	0.611	0.567	0.567	0.564	0.565	0.60
66	11:09:19	669.32	0.541	0.561	0.610	0.601	0.570	0.564	0.612	0.571	0.567	0.565	0.566	0.60
67	11:19:19	679.32	0.542	0.562	0.612	0.602	0.572	0.567	0.613	0.570	0.568	0.566	0.567	0.60
68	11:29:19	689.32	0.543	0.562	0.614	0.604	0.573	0.566	0.615	0.572	0.570	0.566	0.568	0.60
69	11:39:19	699.32	0.543	0.563	0.615	0.605	0.574	0.568	0.616	0.572	0.570	0.568	0.569	0.60
70	11:49:19	709.32	0.545	0.564	0.616	0.606	0.574	0.569	0.617	0.575	0.572	0.568	0.570	0.60
71	11:59:19	719.32	0.544	0.565	0.617	0.607	0.575	0.570	0.618	0.576	0.573	0.570	0.570	0.60
72	12:09:19	729.32	0.544	0.566	0.618	0.608	0.576	0.571	0.619	0.577	0.574	0.570	0.571	0.61

73 rows × 14 columns

Accessing the dataframe in arrays:

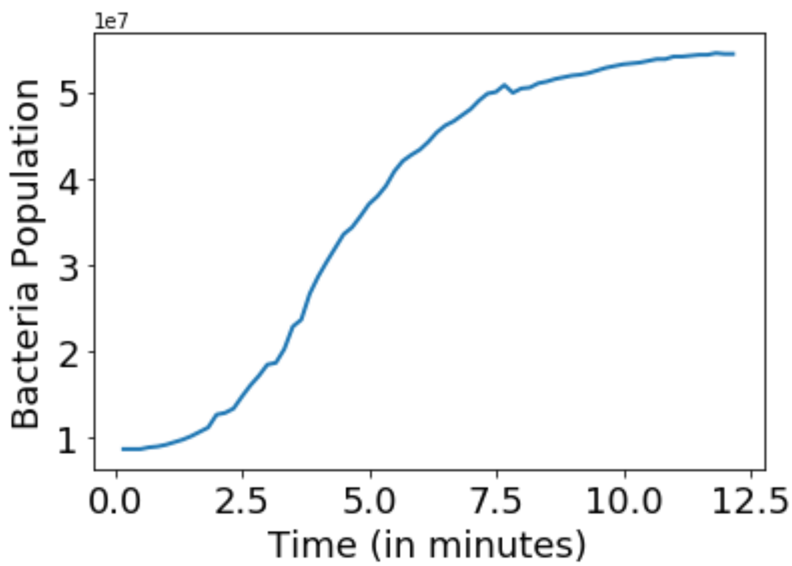
In [6]:
```
import numpy as np
time_array = np.array(df['t'])
```

In [8]:
```
# Here's the time in minutes from the dataset:
time_array
```

Out[8]:
```
array([  9.32,  19.32,  29.32,  39.32,  49.32,  59.32,  69.32,  79.32,
        89.32,  99.32, 109.32, 119.32, 129.32, 139.32, 149.32, 159.32,
       169.32, 179.32, 189.32, 199.32, 209.32, 219.32, 229.32, 239.32,
       249.32, 259.32, 269.32, 279.32, 289.32, 299.32, 309.32, 319.32,
       329.32, 339.32, 349.32, 359.32, 369.32, 379.32, 389.32, 399.32,
       409.32, 419.32, 429.32, 439.32, 449.32, 459.32, 469.32, 479.32,
       489.32, 499.32, 509.32, 519.32, 529.32, 539.32, 549.32, 559.32,
       569.32, 579.32, 589.32, 599.32, 609.32, 619.32, 629.32, 639.32,
       649.32, 659.32, 669.32, 679.32, 689.32, 699.32, 709.32, 719.32,
       729.32])
```

Plotting dataframe using matplotlib:

In [18]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
# Plot the data for the container A1 against time:
# Convert time to hours and "A1" density data to cell counts.
ax.plot(df['t']/60, df['A1']*1e8, lw = 2)
ax.set_xlabel('Time (in hours)', fontsize = 18)
ax.set_ylabel('Bacteria Population', fontsize = 18)
ax.tick_params(labelsize = 18)
```



Now plot all data together:

In [43]:

```
fig, ax = plt.subplots()
for column_title in df:
    # the first two columns are not data, they are time columns, so we igno
    if column_title == 't' or column_title == "Time":
        # Skip this and continue to iterate
        continue
    ax.plot(df['t']/60, df[column_title]*1e8, alpha = 0.7, lw = 2, label = c
ax.set_xlabel('Time (in hours)', fontsize = 18)
ax.set_ylabel('Bacteria Population', fontsize = 18)
ax.tick_params(labelsize = 14)
# bbox_to_anchor command  is used to set the position of the legend box
ax.legend(fontsize = 12, bbox_to_anchor = (1.05,1.1), loc = "best");
```