trial for dominique

September 20, 2018

0.1 trials for Dominique

see if there is a correlation between the different residuals of different species

In [123]:
```
import numpy as np
import pandas as pd
```

In [124]:
```
datatote = pd.read_csv('parameterfitsFungi/nonlinfit7.txt',header=None)
datatote
```

Out[124]:

	0	1	2	3	4	5
0	1	E	-14250.087525	-11156.544121	0.311284	0.032444
1	1	H	1.018096	0.739876	0.003008	0.000050
2	1	Q	-9789.440306	-6685.082717	0.157009	0.035848
3	1	F	0.761479	0.713766	0.014239	0.000071
4	1	Y	-1684.372161	-908.851026	0.040757	0.038420
5	1	C	0.895032	0.821428	0.008780	0.000042
6	1	N	-4140.826663	-2706.734715	0.095442	0.033332
7	1	K	0.784327	0.687553	0.011547	0.000165
8	1	D	0.948727	0.697394	0.004132	0.000099
9	2	E	0.673710	0.626251	0.018714	0.000086
10	2	H	0.989170	0.976107	0.007740	0.000066
11	2	Q	0.816803	0.767509	0.012101	0.000088
12	2	F	-3687.720978	-1582.506833	0.028825	0.041911
13	2	Y	-2587.001144	-2235.826418	0.012774	0.037669
14	2	C	-3619.955879	-1657.441558	0.029112	0.041324
15	2	N	0.838011	0.821985	0.012522	0.000583
16	2	K	0.758210	0.605169	0.010346	0.000474
17	2	D	0.749415	0.679433	0.013879	0.000553
18	3	E	0.821380	0.108451	0.000632	0.000610
19	3	H	-11589.413849	-8094.006009	0.168378	0.048104
20	3	Q	0.580975	0.006971	0.002275	0.000062
21	3	F	-11706.062605	-8365.589404	0.172043	0.047108
22	3	Y	-12928.632809	-9525.001548	0.227368	0.044484
23	3	C	-14402.087881	-10883.053026	0.214905	0.050325
24	3	N	-12590.830327	-8809.616172	0.142494	0.042703
25	3	K	0.415288	-0.087842	0.004799	0.000332
26	3	D	-10854.019306	-7339.232054	0.114130	0.052524

```
27       4   E   -1738.537583  -1140.026394  0.009772  0.039113
28       4   H       0.858932      0.638749  0.006090  0.000048
29       4   Q   -3129.020864  -1479.074455  0.052632  0.038017
...    ...  ...           ...           ...       ...       ...
4124   459   N  -70721.500960 -70722.002563  0.000382  0.032644
4125   459   K       0.756902      0.762496  0.016520  0.000566
4126   459   D       0.675254      0.562190  0.015127  0.000697
4127   460   E   -9906.536888  -6936.652662  0.116883  0.046973
4128   460   H       0.741454      0.393203  0.005021  0.000364
4129   460   Q  -11477.377887  -8335.061045  0.133333  0.066079
4130   460   F       0.775717      0.602947  0.009124  0.000302
4131   460   Y   -1994.476624  -1888.340028  0.017021  0.037689
4132   460   C       1.131751      0.539917  0.000579  0.000535
4133   460   N       0.554127      0.405708  0.017973  0.002086
4134   460   K   -2996.703804  -2464.957004  0.012048  0.036406
4135   460   D       0.927990      0.393748  0.001283  0.000457
4136   461   E       0.656781      0.378977  0.008698  0.000745
4137   461   H   -4417.310371  -2372.719802  0.056604  0.038486
4138   461   Q       0.902042      0.824751  0.008444  0.000183
4139   461   F  -10939.077095  -7740.886631  0.136752  0.048038
4140   461   Y   -8649.288609  -5497.238085  0.122867  0.038049
4141   461   C   -7628.836326  -4664.165194  0.062992  0.051539
4142   461   N   -6368.377297  -4153.529721  0.072937  0.045530
4143   461   K       0.810762      0.430491  0.003677  0.000189
4144   461   D   -5206.456032  -3741.356291  0.056818  0.035351
4145   462   E       0.814121      0.541051  0.005712  0.000355
4146   462   H   -3266.399022  -1690.633077  0.043682  0.037880
4147   462   Q       0.715664      0.628771  0.014604  0.000148
4148   462   F  -10000.003514  -6615.122663  0.149590  0.047584
4149   462   Y   -8795.720937  -5880.565076  0.142390  0.044668
4150   462   C   -7801.326841  -4803.957699  0.066154  0.040919
4151   462   N   -6658.170162  -4547.657753  0.071813  0.045060
4152   462   K       0.903896      0.545003  0.003122  0.000158
4153   462   D   -3693.676304  -1974.508790  0.041667  0.038813

[4154 rows x 6 columns]
```

0.1.1 we remove uninsteresting data

In [125]:
```
data = datatot.drop([1,4,5], axis=1)
```

In [126]:
```
data
```

Out[126]:
```
        0             2             3
0       1 -14250.087525 -11156.544121
1       1      1.018096      0.739876
2       1  -9789.440306  -6685.082717
3       1      0.761479      0.713766
```

4	1	-1684.372161	-908.851026
5	1	0.895032	0.821428
6	1	-4140.826663	-2706.734715
7	1	0.784327	0.687553
8	1	0.948727	0.697394
9	2	0.673710	0.626251
10	2	0.989170	0.976107
11	2	0.816803	0.767509
12	2	-3687.720978	-1582.506833
13	2	-2587.001144	-2235.826418
14	2	-3619.955879	-1657.441558
15	2	0.838011	0.821985
16	2	0.758210	0.605169
17	2	0.749415	0.679433
18	3	0.821380	0.108451
19	3	-11589.413849	-8094.006009
20	3	0.580975	0.006971
21	3	-11706.062605	-8365.589404
22	3	-12928.632809	-9525.001548
23	3	-14402.087881	-10883.053026
24	3	-12590.830327	-8809.616172
25	3	0.415288	-0.087842
26	3	-10854.019306	-7339.232054
27	4	-1738.537583	-1140.026394
28	4	0.858932	0.638749
29	4	-3129.020864	-1479.074455
...	...	...	...
4124	459	-70721.500960	-70722.002563
4125	459	0.756902	0.762496
4126	459	0.675254	0.562190
4127	460	-9906.536888	-6936.652662
4128	460	0.741454	0.393203
4129	460	-11477.377887	-8335.061045
4130	460	0.775717	0.602947
4131	460	-1994.476624	-1888.340028
4132	460	1.131751	0.539917
4133	460	0.554127	0.405708
4134	460	-2996.703804	-2464.957004
4135	460	0.927990	0.393748
4136	461	0.656781	0.378977
4137	461	-4417.310371	-2372.719802
4138	461	0.902042	0.824751
4139	461	-10939.077095	-7740.886631
4140	461	-8649.288609	-5497.238085
4141	461	-7628.836326	-4664.165194
4142	461	-6368.377297	-4153.529721
4143	461	0.810762	0.430491
4144	461	-5206.456032	-3741.356291

```
4145  462      0.814121      0.541051
4146  462  -3266.399022  -1690.633077
4147  462      0.715664      0.628771
4148  462 -10000.003514  -6615.122663
4149  462  -8795.720937  -5880.565076
4150  462  -7801.326841  -4803.957699
4151  462  -6658.170162  -4547.657753
4152  462      0.903896      0.545003
4153  462  -3693.676304  -1974.508790

[4154 rows x 3 columns]
```

In [127]:
```
len(data)/data[0][len(data)-1]
```

Out[127]:
```
8
```

In [128]:
```
arr = np.zeros((data[0][len(data)-1],9))
data = np.array(data)
```

0.1.2 we put everything in a corresponding array organized by species and averaging the two regressors.

In [129]:
```
temp = 0
i = 0
for dat in data:
    if dat[0] == temp:
        i+=1
    else:
        temp=dat[0]
        i=0
    arr[int(dat[0])-1,i]=dat[1:].mean()
```

In [130]:
```
arr[8]
```

Out[130]:
```
array([ 2.82900000e-01, -1.79900889e+04,  3.45084500e-01, -5.98478642e+03,
       -3.50439912e+04, -1.72447122e+04, -3.62870317e+04,  7.37060000e-01,
       -3.18622675e+04])
```

we can see that the values vary a lot, from 9 to -10,000 with a lot of very low numbers. this creates quite high variances. However, High number for one AA often means the same for another.

In [131]:
```
arr.var(1)
```

Out[131]:
```
array([1.88320167e+07, 1.46298823e+06, 2.57848672e+07, 1.23875859e+06,
       3.38089684e+07, 5.81636619e+06, 9.85721808e+05, 1.15497369e+08,
       2.11076435e+08, 1.48011996e+06, 4.29659097e+06, 3.06602532e+06,
       1.13877470e+07, 1.56978859e+07, 1.58120395e+07, 3.36395088e+06,
       1.67513007e+07, 3.04943265e+06, 6.56091388e+06, 2.96761685e+06,
```

```
1.33168360e+06, 3.33227600e+06, 5.87558032e+06, 4.59123591e+06,
3.03355034e+06, 2.45890719e+06, 2.55713215e+06, 2.86463372e+06,
1.63368004e+06, 3.85745337e+06, 2.34380078e+06, 1.94293020e+06,
5.92445632e+06, 3.97194954e+06, 1.23150844e+07, 4.37447250e+06,
1.09796225e+07, 5.89096574e+06, 1.39090088e+07, 3.97052243e+06,
1.00380710e+07, 1.79882205e+07, 5.41169803e+06, 6.55981651e+06,
4.88902117e+06, 4.94541570e+06, 5.48681438e+06, 3.85717731e+06,
2.86394650e+06, 3.84317066e+06, 2.83628470e+06, 3.74389095e+06,
3.26953862e+05, 5.78246176e+06, 1.49316012e+07, 1.30144580e+07,
4.37046332e+07, 2.46755944e+07, 4.48815885e+06, 2.98772807e+07,
6.87974192e+06, 3.86565664e+07, 2.36927258e+07, 1.95623164e+07,
1.99481461e+07, 6.32308803e+06, 4.78252731e+08, 6.22502400e+06,
3.64617707e+06, 5.22190817e+06, 1.17093701e+07, 5.27698951e+06,
3.38054897e+06, 1.76455182e+07, 7.51754389e+06, 1.59776475e+07,
2.21454442e+06, 7.18669846e+06, 3.51090303e+06, 3.17123798e+06,
1.20604726e+07, 3.69905139e+06, 5.04433414e+06, 1.26668732e+06,
1.06905913e+06, 1.53474280e+07, 1.69079859e+07, 2.07075905e+07,
2.42201599e+07, 1.34099967e+07, 1.26650020e+07, 2.37231941e+07,
1.43004093e+07, 1.49120454e+07, 1.83995894e+07, 2.14378793e+07,
2.01599243e+07, 1.01046063e+07, 1.28098163e+07, 6.38373755e+06,
1.97839810e+07, 2.32459961e+07, 2.29265291e+07, 5.53194271e+06,
1.63919467e+06, 1.62717734e+06, 2.35072835e+06, 1.83852730e+06,
1.58536422e+06, 4.21419641e+06, 3.07130736e+07, 8.68655203e+06,
3.37935250e+06, 6.55805306e+06, 5.90843689e+06, 9.17023805e+06,
2.91149217e+07, 1.14810482e+07, 3.99649447e+07, 1.70703681e+07,
1.63093448e+07, 4.63117522e+07, 6.53035783e+06, 2.77565551e+07,
1.16463941e+07, 6.31852637e+07, 6.90915583e+05, 3.18942283e+06,
1.22394205e+06, 6.39633024e+06, 9.90753087e+05, 5.06260549e-02,
6.03045849e-01, 2.61134652e+06, 1.74098171e+07, 1.16715220e+06,
1.46259650e+06, 6.67530784e+06, 5.67530042e+07, 1.61486981e+07,
1.60781642e+07, 1.95150996e+06, 4.40356468e+06, 9.59475968e+05,
4.26347682e+06, 4.16116123e+06, 6.53982775e+06, 6.58502212e+06,
5.15489567e+06, 5.24978724e+06, 2.10989060e+06, 5.27896028e+06,
2.32195793e+07, 7.23585676e+06, 4.78554234e+06, 3.19049531e+06,
6.01784591e+06, 6.19861757e+06, 4.68382382e+06, 2.99302999e+06,
5.04097425e+06, 4.08687606e+06, 2.24222816e+06, 4.58821310e+06,
4.94997979e+06, 6.95500986e+06, 3.56843375e+06, 2.80198597e+07,
6.86389391e+05, 9.17994391e+06, 1.21316992e+07, 2.37831377e+06,
8.95224692e+06, 9.23015861e+06, 4.13769908e+06, 2.29496299e+07,
7.87008825e+05, 2.73050118e+07, 1.85128758e+07, 6.36773204e+07,
1.35541657e+08, 1.44615748e+06, 5.55521107e+06, 1.02479985e+07,
1.58164750e+06, 4.00406855e+06, 4.31400365e+06, 2.88568664e+07,
2.60832179e+06, 3.22386503e+07, 1.52216684e+06, 3.48264500e+07,
9.68331583e+06, 1.82697615e+07, 1.54121893e+07, 6.77521100e+06,
8.25528638e+05, 1.08542815e+06, 1.52890700e+06, 7.78605148e+06,
1.33796491e+07, 1.97745032e+07, 9.18654343e+06, 1.96972466e+07,
1.11054751e+06, 1.38791053e+06, 2.05460727e+06, 8.67058856e+06,
4.84675118e+06, 3.13369186e+06, 4.16571060e+06, 9.94191912e+06,
```

2.01582541e+07, 1.52861652e+06, 1.04830068e+07, 1.33461266e+07,
1.95876975e+07, 1.83932232e+07, 2.56055688e+07, 2.67689960e+06,
1.05542167e+07, 8.20731198e+06, 1.13753191e+07, 5.64520923e+06,
1.26836441e+07, 7.68381979e+06, 6.34826399e+06, 6.62274641e+06,
5.85627481e+06, 5.14294820e+06, 4.97408874e+06, 8.47531462e+06,
1.69730729e+06, 1.17912395e+07, 2.99436970e+07, 3.64545266e+06,
3.51211819e+06, 1.44956046e+06, 6.69134135e+06, 8.28803800e+07,
4.14704909e+07, 2.05873585e+06, 4.72873840e+05, 5.66083084e+06,
1.17229216e+07, 1.03304139e+07, 1.22865948e+07, 2.83782261e+07,
5.73942758e+06, 1.17907880e+07, 1.80165735e+07, 3.57793555e+07,
3.01143016e+06, 5.78403869e+06, 4.98246436e+07, 4.20907550e+06,
1.50520799e+07, 5.29772231e+06, 7.81432761e+06, 9.90867594e+06,
1.24057228e+08, 3.43579933e+07, 1.66956835e+08, 7.62859255e+06,
8.43518843e+06, 1.80551265e+06, 3.30203405e+07, 1.65763081e+07,
2.68283846e+07, 1.72915325e+01, 4.12590574e+07, 1.24848910e+05,
2.51873890e+05, 9.77528832e+06, 8.13973144e+06, 2.64148194e+06,
1.60038996e+06, 4.61519779e+06, 2.99009070e+06, 2.41859317e+06,
2.10296963e+06, 3.58443889e+06, 3.02089820e+06, 4.82079126e+06,
4.51896794e+06, 4.02154173e+06, 3.46358527e+06, 3.07673421e+06,
2.80870096e+06, 7.53179410e+06, 4.42138338e+06, 9.08593367e+06,
3.00691183e+06, 8.34010112e+06, 7.44651706e+06, 1.93017175e+06,
7.79572110e+06, 1.04239342e+07, 4.75729187e+06, 5.90087987e+06,
1.39370897e+07, 3.45942074e+05, 7.90278670e+05, 1.10088455e+06,
5.60566049e+06, 7.19917831e+06, 3.79348414e+07, 2.22384380e+07,
1.40133676e+08, 1.21543800e+06, 8.25017708e+06, 7.75083985e+06,
3.76622051e+06, 5.23813155e+06, 3.85506830e+06, 4.56368349e+06,
5.23379822e+06, 5.43756400e+06, 6.64785660e+06, 3.95081556e+06,
2.94733229e+06, 4.95199416e+06, 4.69818107e+06, 4.86564973e+06,
4.64040329e+06, 3.94305960e+06, 4.61178414e+06, 4.07642036e+06,
5.26336553e+06, 3.52848173e+06, 6.32809768e+06, 5.01382507e+06,
5.03483287e+06, 5.10822495e+06, 4.02557358e+07, 2.00945209e+07,
3.36501980e+06, 4.34892369e+06, 3.93617828e+06, 2.96558121e+06,
1.40604209e+07, 2.12266298e+07, 4.36724776e+06, 6.54385218e+06,
5.55194958e+06, 3.87626707e+06, 3.29377475e+06, 1.51826439e+06,
9.37999925e+05, 3.36073430e+07, 9.07656549e+05, 1.89510601e+07,
1.94816494e+08, 4.75831280e+07, 1.06583672e+08, 1.24617752e+07,
2.44581184e+07, 4.75834899e+08, 6.82491663e+06, 6.06789436e+06,
8.12656041e+06, 1.06683344e+07, 5.72758183e+06, 1.49120622e+07,
4.21211354e+06, 9.76848926e+06, 8.87133956e+06, 2.03235643e+07,
2.36784670e+06, 1.27205110e+07, 4.14394006e+06, 1.56190824e+07,
1.73631546e+07, 7.08569844e+05, 3.51081851e+06, 3.52644062e+06,
4.21911748e+05, 7.36412785e+05, 6.83779729e+05, 9.34570508e+06,
2.97913232e+06, 3.79177370e+06, 1.24840065e+07, 1.04519897e+08,
9.48024335e+05, 2.34031965e+06, 9.92277753e+05, 6.08757611e+07,
3.57570321e+07, 1.62612437e+07, 4.01948923e+07, 4.05616728e+07,
4.14063150e+07, 7.92694106e+07, 1.50006633e+07, 1.45089031e+07,
5.80395868e+06, 1.18598491e+07, 3.24824187e+06, 2.99964244e+06,
6.62106846e+05, 1.00557573e+06, 2.14978966e+06, 1.27117776e+06,

```
4.23182053e+05, 7.60094000e+05, 3.96055906e+07, 2.20187261e+07,
3.51621868e+07, 1.85421409e+07, 1.80177020e+07, 2.05574460e+07,
2.17109974e+07, 1.21648256e+07, 2.03800012e+07, 2.05764445e+07,
7.90366115e+06, 6.03541385e+07, 1.41614217e+06, 3.06125797e+06,
2.24503067e+07, 3.22979463e+07, 5.77298334e+06, 7.27681976e+06,
5.72277348e+06, 4.90141674e+06, 1.92305063e+07, 5.67383354e+06,
3.77912724e+06, 3.92947762e+06, 2.46157269e+06, 3.07665615e+06,
2.48992135e+06, 3.31623450e+06, 3.63110727e+06, 2.77053443e+07,
5.76350053e+06, 2.15446710e+06, 2.87738003e+06, 1.11224560e+06,
2.07609476e+07, 7.92735489e+06, 9.89938257e+06, 1.33606368e+07,
2.12840266e+07, 1.54804038e+06, 3.77961794e+06, 1.84245907e+07,
2.12522405e+07, 2.35397018e+07, 1.22732600e+05, 4.83443472e+06,
6.37423034e+06, 4.32173712e+07, 5.16333749e+08, 6.89368180e+06,
2.22570555e+06, 2.04592591e+07, 4.82631052e+08, 1.35023802e+07,
1.03059546e+07, 9.79626493e+06])
```

```
In [132]: arr.var()
```

```
Out[132]: 22724635.591483254
```

variance accross measures/ mean variance accross species

```
In [133]: arr.mean(0).var()
```

```
Out[133]: 1388583.6554276315
```

0.1.3 we can see that the mean is still pretty high.

however, mixing AA between different species, shows that a random association is finding much worse variance than the one we have (on average a difference of 2,832,176)

```
In [134]: arr.mean(0)
```

```
Out[134]: array([-2219.06645714, -2919.84412463, -2015.18948926, -4288.21485415,
                 -4477.73292545, -4078.47555626, -4687.57869146, -1166.91552409,
                 -2855.27048881])
```

```
In [135]: arr.shape
```

```
Out[135]: (462, 9)
```

```
In [149]: randarr = np.zeros(arr.shape)
          for i in range(arr.shape[1]):
              ind = np.arange(len(arr))
              np.random.shuffle(ind)
              randarr[:,i] = arr[ind,i]
```

```
In [137]: randarr
```

```
Out[137]: array([[ 4.15145500e-01, -6.20570878e+03,  6.97707000e-01, ...,
                 -1.49103579e+03,  1.06608050e+00,  8.41851000e-01],
                [-5.15551749e+03, -3.06515524e+03,  6.26267500e-01, ...,
                 -1.03056625e+04,  4.93779000e-01, -3.08304566e+03],
                [ 3.32430000e-01, -1.86764586e+03,  3.92859000e-01, ...,
                 -2.51911140e+03,  7.77000000e-01, -3.12708306e+03],
                ...,
                [-1.56705559e+04, -3.47117630e+03,  5.81732000e-01, ...,
                 -4.28548748e+03, -8.39942050e+03, -1.02534611e+04],
                [-1.24855159e+03, -6.34379273e+03,  7.27223000e-01, ...,
                 -3.62731461e+03,  5.13653500e-01, -5.19541578e+03],
                [ 5.54347000e-01, -2.13428459e+03, -2.30404766e+03, ...,
                 -9.43247771e+03,  5.71346000e-01, -2.95652433e+03]])

In [138]: randarr.mean(0)

Out[138]: array([-2219.06645714, -2919.84412463, -2015.18948926, -4288.21485415,
                 -4477.73292545, -4078.47555626, -4687.57869146, -1166.91552409,
                 -2855.27048881])

In [147]: randarr.mean(0).var()-arr.mean(0).var()

Out[147]: 2.3283064365386963e-10

In [153]: randarr = np.zeros(arr.shape)
          X = 2000
          a= 0
          for _ in range(X):
              for i in range(arr.shape[1]):
                  ind = np.arange(len(arr))
                  np.random.shuffle(ind)
                  randarr[:,i] = arr[ind,i]
              a += randarr.var(1).mean()-arr.var(1).mean()
          print a/X

2832176.8380952715


In [150]: randarr.var(1).mean()-arr.var(1).mean()

Out[150]: 2653785.4159249514

In [70]: arr.var(0)

Out[70]: array([41333202.63856429, 10727555.04310829, 19414596.46349642,
                12230304.06525019, 22099342.25172593, 19543347.84000982,
                28742207.88900693, 21790682.06159877, 16143229.1717399 ])
```

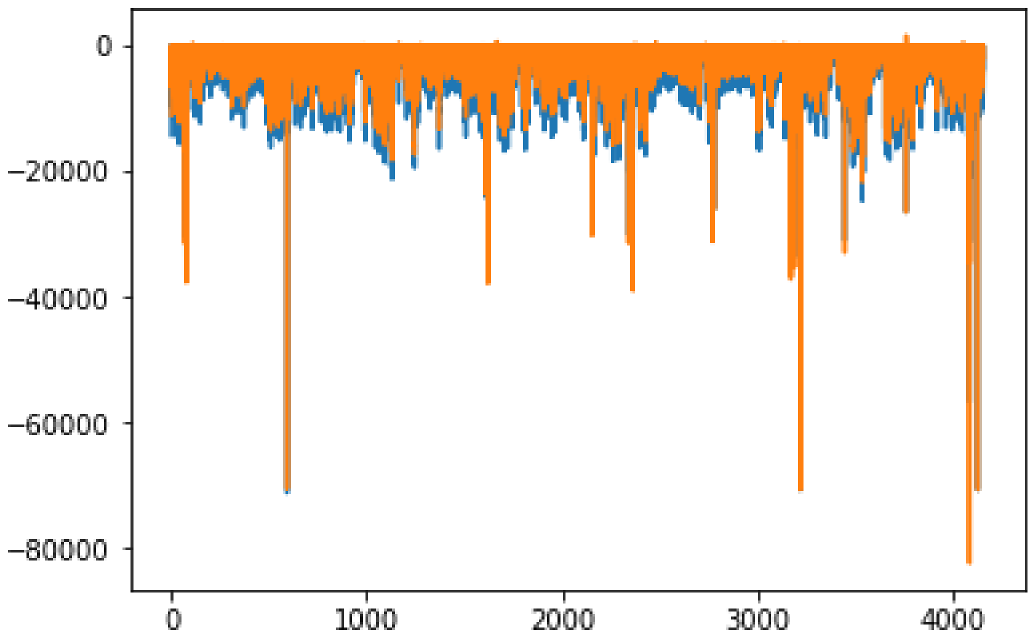
0.2 we can explore how the data looks like

we see here that even if there is some correlation, the variation is too high to be able to cluster it

```
In [31]: from matplotlib.pyplot import *
```

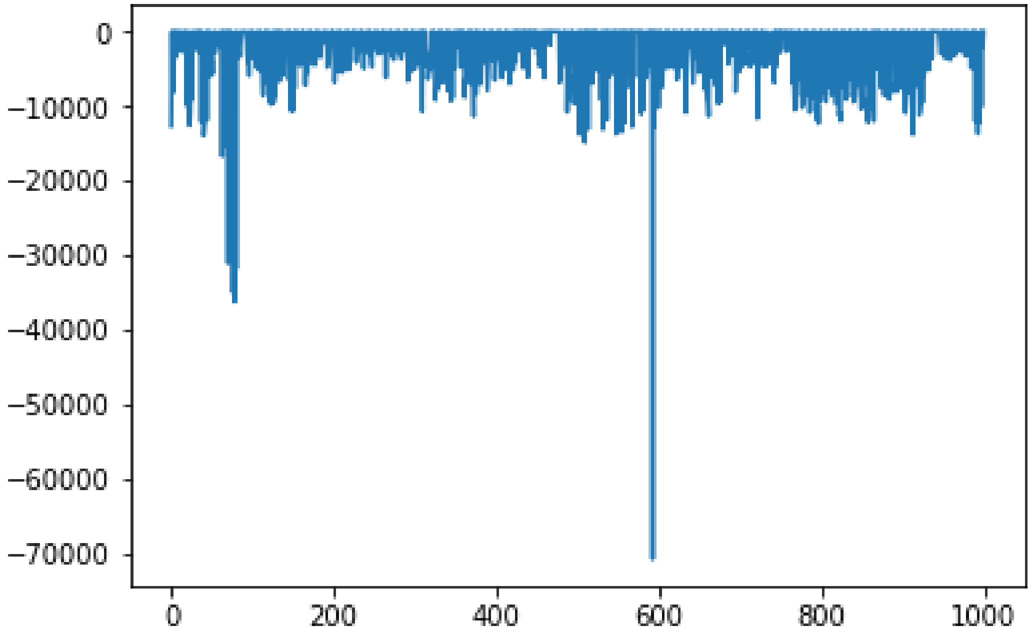
```
In [72]: plot(data[:,1:])
```

```
Out[72]: [<matplotlib.lines.Line2D at 0x10edf3fd0>,
          <matplotlib.lines.Line2D at 0x10ee11110>]
```

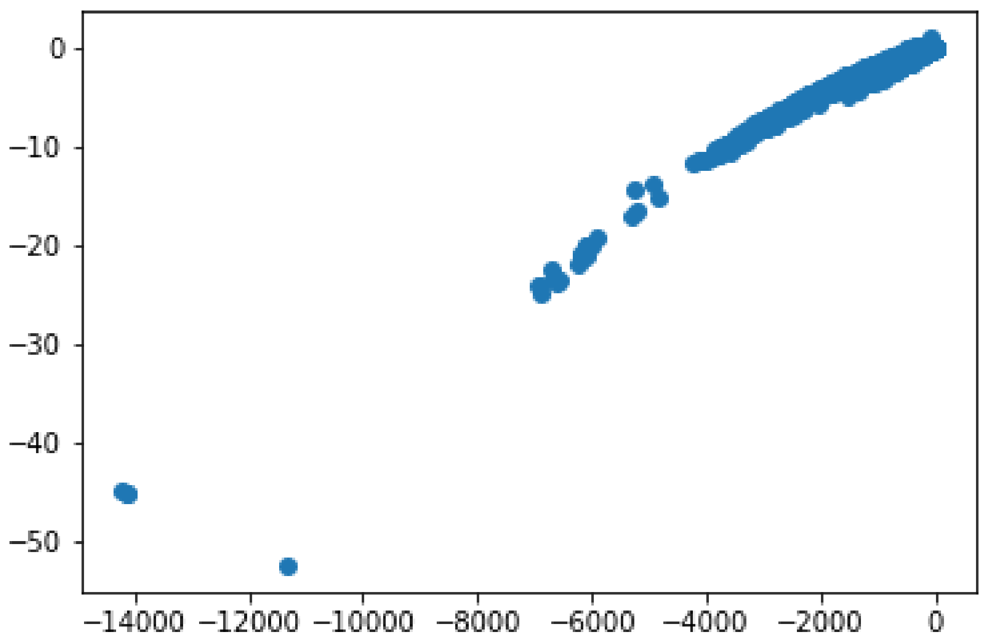



```
In [73]: plot(data[:1000,1:].mean(1))
```

```
Out[73]: [<matplotlib.lines.Line2D at 0x103c5a650>]
```




```
In [75]: scatter(data[:,1]/data[:,1].max(),data[:,2]/data[:,2].max())

Out[75]: <matplotlib.collections.PathCollection at 0x10a510690>
```



0.3 Given the repartition of the data (outliers) it will be almost impossible to cluster

we should better use direct exploration

```
In [36]: from bokeh.plotting import *
         from bokeh.io import save, show
         from bokeh.models import *
```

```
In [76]: data[:,1].max()
```

```
Out[76]: 5.0115110000000005
```

0.3.1 We can see that it is confirmed here, even if it does not cluster, we can see some correlation

this view allows us to zoom in the data

```
In [77]: X = -1
         species = [1,2,3,4,5,6,7,8,9,10]
         col = ['#f39c12', "#1abc9c", "#3498db", "#2ecc71", "#9b59b6", '#34495e', '#492000', '#
         '#043927', '#8F9779', '#f1c40f', '#e67e22', '#e74c3c', '#7f8c8d']
         source = ColumnDataSource(data=dict(x = data[:X,1]/data[:,1].min(),
                                             y = data[:X,2]/data[:,2].min(),
                                             names = np.array(datatot[0])[:X],
                                             colors= [col[0] if i-30 not in species else\
                                                     col[i-30] for i in np.array(datatot[0])[:X
         output_notebook()
         hover = HoverTool(tooltips=[("species: ", "@names")])
         p = figure(title="Plot of the regressors",
                    tools=[hover, WheelZoomTool(), PanTool(), SaveTool(), ResetTool()],
                    plot_width=800, plot_height=600)
         p.circle(x='x', y='y', source=source, size=10,color='colors')
         show(p)
```

0.3.2 we can further test it by running classifier to see if they are able to cluster the data according to our labels

the results are very low. . . .

```
In [154]: from sklearn.neighbors import KNeighborsClassifier
          from sklearn.gaussian_process import GaussianProcessClassifier
          from sklearn.svm import SVC
          svc = SVC(C=1.0, kernel='rbf', degree=40, gamma='auto', coef0=0.0, shrinking=True, p
                 tol=0.001, cache_size=200, class_weight=None, verbose=False,
                 max_iter=-1, random_state=None).fit(data[:,1:], data[:,0].astype(int))
          neigh = KNeighborsClassifier(n_neighbors=462).fit(data[:,1:], data[:,0].astype(int))
```

```
In [155]: neigh.score(data[:,1:], data[:,0].astype(int))
```

```
Out[155]: 0.01636976408281175
```

```
In [156]: svc.score(data[:,1:], data[:,0].astype(int))
```

```
Out[156]: 0.5642753972075109
```

0.4 Same for bacterias

```
In [120]: datatot = pd.read_csv('parameterFitsBacteria/nonlinfit7.txt',header=None)
          datatot
```

Out[120]:

	0	1	2	3	4	5
0	1	E	-14250.087525	-11156.544121	0.311284	0.032444
1	1	H	1.018096	0.739876	0.003008	0.000050
2	1	Q	-9789.440306	-6685.082717	0.157009	0.035848
3	1	F	0.761479	0.713766	0.014239	0.000071
4	1	Y	-1684.372161	-908.851026	0.040757	0.038420
5	1	C	0.895032	0.821428	0.008780	0.000042
6	1	N	-4140.826663	-2706.734715	0.095442	0.033332
7	1	K	0.784327	0.687553	0.011547	0.000165
8	1	D	0.948727	0.697394	0.004132	0.000099
9	2	E	0.673710	0.626251	0.018714	0.000086
10	2	H	0.989170	0.976107	0.007740	0.000066
11	2	Q	0.816803	0.767509	0.012101	0.000088
12	2	F	-3687.720978	-1582.506833	0.028825	0.041911
13	2	Y	-2587.001144	-2235.826418	0.012774	0.037669
14	2	C	-3619.955879	-1657.441558	0.029112	0.041324
15	2	N	0.838011	0.821985	0.012522	0.000583
16	2	K	0.758210	0.605169	0.010346	0.000474
17	2	D	0.749415	0.679433	0.013879	0.000553
18	3	E	0.821380	0.108451	0.000632	0.000610
19	3	H	-11589.413849	-8094.006009	0.168378	0.048104
20	3	Q	0.580975	0.006971	0.002275	0.000062
21	3	F	-11706.062605	-8365.589404	0.172043	0.047108
22	3	Y	-12928.632809	-9525.001548	0.227368	0.044484
23	3	C	-14402.087881	-10883.053026	0.214905	0.050325
24	3	N	-12590.830327	-8809.616172	0.142494	0.042703
25	3	K	0.415288	-0.087842	0.004799	0.000332
26	3	D	-10854.019306	-7339.232054	0.114130	0.052524
27	4	E	-1738.537583	-1140.026394	0.009772	0.039113
28	4	H	0.858932	0.638749	0.006090	0.000048
29	4	Q	-3129.020864	-1479.074455	0.052632	0.038017
...	...	...	...	...	...	...
4124	459	N	-70721.500960	-70722.002563	0.000382	0.032644
4125	459	K	0.756902	0.762496	0.016520	0.000566
4126	459	D	0.675254	0.562190	0.015127	0.000697
4127	460	E	-9906.536888	-6936.652662	0.116883	0.046973
4128	460	H	0.741454	0.393203	0.005021	0.000364
4129	460	Q	-11477.377887	-8335.061045	0.133333	0.066079
4130	460	F	0.775717	0.602947	0.009124	0.000302
4131	460	Y	-1994.476624	-1888.340028	0.017021	0.037689
4132	460	C	1.131751	0.539917	0.000579	0.000535
4133	460	N	0.554127	0.405708	0.017973	0.002086
4134	460	K	-2996.703804	-2464.957004	0.012048	0.036406
4135	460	D	0.927990	0.393748	0.001283	0.000457

```
4136  461   E       0.656781      0.378977  0.008698  0.000745
4137  461   H   -4417.310371  -2372.719802  0.056604  0.038486
4138  461   Q       0.902042      0.824751  0.008444  0.000183
4139  461   F  -10939.077095  -7740.886631  0.136752  0.048038
4140  461   Y   -8649.288609  -5497.238085  0.122867  0.038049
4141  461   C   -7628.836326  -4664.165194  0.062992  0.051539
4142  461   N   -6368.377297  -4153.529721  0.072937  0.045530
4143  461   K       0.810762      0.430491  0.003677  0.000189
4144  461   D   -5206.456032  -3741.356291  0.056818  0.035351
4145  462   E       0.814121      0.541051  0.005712  0.000355
4146  462   H   -3266.399022  -1690.633077  0.043682  0.037880
4147  462   Q       0.715664      0.628771  0.014604  0.000148
4148  462   F  -10000.003514  -6615.122663  0.149590  0.047584
4149  462   Y   -8795.720937  -5880.565076  0.142390  0.044668
4150  462   C   -7801.326841  -4803.957699  0.066154  0.040919
4151  462   N   -6658.170162  -4547.657753  0.071813  0.045060
4152  462   K       0.903896      0.545003  0.003122  0.000158
4153  462   D   -3693.676304  -1974.508790  0.041667  0.038813

[4154 rows x 6 columns]
```

```
In [103]: data = datatot.drop([1,4,5], axis=1)

In [104]: data

Out[104]:         0             2             3
          0       1 -14250.087525 -11156.544121
          1       1      1.018096      0.739876
          2       1  -9789.440306  -6685.082717
          3       1      0.761479      0.713766
          4       1  -1684.372161   -908.851026
          5       1      0.895032      0.821428
          6       1  -4140.826663  -2706.734715
          7       1      0.784327      0.687553
          8       1      0.948727      0.697394
          9       2      0.673710      0.626251
          10      2      0.989170      0.976107
          11      2      0.816803      0.767509
          12      2  -3687.720978  -1582.506833
          13      2  -2587.001144  -2235.826418
          14      2  -3619.955879  -1657.441558
          15      2      0.838011      0.821985
          16      2      0.758210      0.605169
          17      2      0.749415      0.679433
          18      3      0.821380      0.108451
          19      3 -11589.413849  -8094.006009
          20      3      0.580975      0.006971
          21      3 -11706.062605  -8365.589404
```

```
22      3 -12928.632809  -9525.001548
23      3 -14402.087881 -10883.053026
24      3 -12590.830327  -8809.616172
25      3      0.415288     -0.087842
26      3 -10854.019306  -7339.232054
27      4  -1738.537583  -1140.026394
28      4      0.858932      0.638749
29      4  -3129.020864  -1479.074455
...   ...           ...           ...
4124  459 -70721.500960 -70722.002563
4125  459      0.756902      0.762496
4126  459      0.675254      0.562190
4127  460  -9906.536888  -6936.652662
4128  460      0.741454      0.393203
4129  460 -11477.377887  -8335.061045
4130  460      0.775717      0.602947
4131  460  -1994.476624  -1888.340028
4132  460      1.131751      0.539917
4133  460      0.554127      0.405708
4134  460  -2996.703804  -2464.957004
4135  460      0.927990      0.393748
4136  461      0.656781      0.378977
4137  461  -4417.310371  -2372.719802
4138  461      0.902042      0.824751
4139  461 -10939.077095  -7740.886631
4140  461  -8649.288609  -5497.238085
4141  461  -7628.836326  -4664.165194
4142  461  -6368.377297  -4153.529721
4143  461      0.810762      0.430491
4144  461  -5206.456032  -3741.356291
4145  462      0.814121      0.541051
4146  462  -3266.399022  -1690.633077
4147  462      0.715664      0.628771
4148  462 -10000.003514  -6615.122663
4149  462  -8795.720937  -5880.565076
4150  462  -7801.326841  -4803.957699
4151  462  -6658.170162  -4547.657753
4152  462      0.903896      0.545003
4153  462  -3693.676304  -1974.508790

[4154 rows x 3 columns]
```

In [105]:
```python
len(data)/data[0][len(data)-1]
```

Out[105]: 8

In [106]:
```python
arr = np.zeros((data[0][len(data)-1],9))
data = np.array(data)
```

```
In [107]: temp = 0
          i = 0
          for dat in data:
              if dat[0] == temp:
                  i+=1
              else:
                  temp=dat[0]
                  i=0
              arr[int(dat[0])-1,i]=dat[1:].mean()
```

```
In [108]: arr[8]
```

```
Out[108]: array([ 2.82900000e-01, -1.79900889e+04,  3.45084500e-01, -5.98478642e+03,
                 -3.50439912e+04, -1.72447122e+04, -3.62870317e+04,  7.37060000e-01,
                 -3.18622675e+04])
```

At first I thought that as the distance inter cluster is higher than the overhall distance, there is correlation. However by shuffling, one can see that even a randomize set produce the exact same variance.

mean variance across measures/ total variance in dataset

```
In [109]: arr.var(1)
```

```
Out[109]: array([1.88320167e+07, 1.46298823e+06, 2.57848672e+07, 1.23875859e+06,
                 3.38089684e+07, 5.81636619e+06, 9.85721808e+05, 1.15497369e+08,
                 2.11076435e+08, 1.48011996e+06, 4.29659097e+06, 3.06602532e+06,
                 1.13877470e+07, 1.56978859e+07, 1.58120395e+07, 3.36395088e+06,
                 1.67513007e+07, 3.04943265e+06, 6.56091388e+06, 2.96761685e+06,
                 1.33168360e+06, 3.33227600e+06, 5.87558032e+06, 4.59123591e+06,
                 3.03355034e+06, 2.45890719e+06, 2.55713215e+06, 2.86463372e+06,
                 1.63368004e+06, 3.85745337e+06, 2.34380078e+06, 1.94293020e+06,
                 5.92445632e+06, 3.97194954e+06, 1.23150844e+07, 4.37447250e+06,
                 1.09796225e+07, 5.89096574e+06, 1.39090088e+07, 3.97052243e+06,
                 1.00380710e+07, 1.79882205e+07, 5.41169803e+06, 6.55981651e+06,
                 4.88902117e+06, 4.94541570e+06, 5.48681438e+06, 3.85717731e+06,
                 2.86394650e+06, 3.84317066e+06, 2.83628470e+06, 3.74389095e+06,
                 3.26953862e+05, 5.78246176e+06, 1.49316012e+07, 1.30144580e+07,
                 4.37046332e+07, 2.46755944e+07, 4.48815885e+06, 2.98772807e+07,
                 6.87974192e+06, 3.86565664e+07, 2.36927258e+07, 1.95623164e+07,
                 1.99481461e+07, 6.32308803e+06, 4.78252731e+08, 6.22502400e+06,
                 3.64617707e+06, 5.22190817e+06, 1.17093701e+07, 5.27698951e+06,
                 3.38054897e+06, 1.76455182e+07, 7.51754389e+06, 1.59776475e+07,
                 2.21454442e+06, 7.18669846e+06, 3.51090303e+06, 3.17123798e+06,
                 1.20604726e+07, 3.69905139e+06, 5.04433414e+06, 1.26668732e+06,
                 1.06905913e+06, 1.53474280e+07, 1.69079859e+07, 2.07075905e+07,
                 2.42201599e+07, 1.34099967e+07, 1.26650020e+07, 2.37231941e+07,
                 1.43004093e+07, 1.49120454e+07, 1.83995894e+07, 2.14378793e+07,
                 2.01599243e+07, 1.01046063e+07, 1.28098163e+07, 6.38373755e+06,
                 1.97839810e+07, 2.32459961e+07, 2.29265291e+07, 5.53194271e+06,
```

```
1.63919467e+06, 1.62717734e+06, 2.35072835e+06, 1.83852730e+06,
1.58536422e+06, 4.21419641e+06, 3.07130736e+07, 8.68655203e+06,
3.37935250e+06, 6.55805306e+06, 5.90843689e+06, 9.17023805e+06,
2.91149217e+07, 1.14810482e+07, 3.99649447e+07, 1.70703681e+07,
1.63093448e+07, 4.63117522e+07, 6.53035783e+06, 2.77565551e+07,
1.16463941e+07, 6.31852637e+07, 6.90915583e+05, 3.18942283e+06,
1.22394205e+06, 6.39633024e+06, 9.90753087e+05, 5.06260549e-02,
6.03045849e-01, 2.61134652e+06, 1.74098171e+07, 1.16715220e+06,
1.46259650e+06, 6.67530784e+06, 5.67530042e+07, 1.61486981e+07,
1.60781642e+07, 1.95150996e+06, 4.40356468e+06, 9.59475968e+05,
4.26347682e+06, 4.16116123e+06, 6.53982775e+06, 6.58502212e+06,
5.15489567e+06, 5.24978724e+06, 2.10989060e+06, 5.27896028e+06,
2.32195793e+07, 7.23585676e+06, 4.78554234e+06, 3.19049531e+06,
6.01784591e+06, 6.19861757e+06, 4.68382382e+06, 2.99302999e+06,
5.04097425e+06, 4.08687606e+06, 2.24222816e+06, 4.58821310e+06,
4.94997979e+06, 6.95500986e+06, 3.56843375e+06, 2.80198597e+07,
6.86389391e+05, 9.17994391e+06, 1.21316992e+07, 2.37831377e+06,
8.95224692e+06, 9.23015861e+06, 4.13769908e+06, 2.29496299e+07,
7.87008825e+05, 2.73050118e+07, 1.85128758e+07, 6.36773204e+07,
1.35541657e+08, 1.44615748e+06, 5.55521107e+06, 1.02479985e+07,
1.58164750e+06, 4.00406855e+06, 4.31400365e+06, 2.88568664e+07,
2.60832179e+06, 3.22386503e+07, 1.52216684e+06, 3.48264500e+07,
9.68331583e+06, 1.82697615e+07, 1.54121893e+07, 6.77521100e+06,
8.25528638e+05, 1.08542815e+06, 1.52890700e+06, 7.78605148e+06,
1.33796491e+07, 1.97745032e+07, 9.18654343e+06, 1.96972466e+07,
1.11054751e+06, 1.38791053e+06, 2.05460727e+06, 8.67058856e+06,
4.84675118e+06, 3.13369186e+06, 4.16571060e+06, 9.94191912e+06,
2.01582541e+07, 1.52861652e+06, 1.04830068e+07, 1.33461266e+07,
1.95876975e+07, 1.83932232e+07, 2.56055688e+07, 2.67689960e+06,
1.05542167e+07, 8.20731198e+06, 1.13753191e+07, 5.64520923e+06,
1.26836441e+07, 7.68381979e+06, 6.34826399e+06, 6.62274641e+06,
5.85627481e+06, 5.14294820e+06, 4.97408874e+06, 8.47531462e+06,
1.69730729e+06, 1.17912395e+07, 2.99436970e+07, 3.64545266e+06,
3.51211819e+06, 1.44956046e+06, 6.69134135e+06, 8.28803800e+07,
4.14704909e+07, 2.05873585e+06, 4.72873840e+05, 5.66083084e+06,
1.17229216e+07, 1.03304139e+07, 1.22865948e+07, 2.83782261e+07,
5.73942758e+06, 1.17907880e+07, 1.80165735e+07, 3.57793555e+07,
3.01143016e+06, 5.78403869e+06, 4.98246436e+07, 4.20907550e+06,
1.50520799e+07, 5.29772231e+06, 7.81432761e+06, 9.90867594e+06,
1.24057228e+08, 3.43579933e+07, 1.66956835e+08, 7.62859255e+06,
8.43518843e+06, 1.80551265e+06, 3.30203405e+07, 1.65763081e+07,
2.68283846e+07, 1.72915325e+01, 4.12590574e+07, 1.24848910e+05,
2.51873890e+05, 9.77528832e+06, 8.13973144e+06, 2.64148194e+06,
1.60038996e+06, 4.61519779e+06, 2.99009070e+06, 2.41859317e+06,
2.10296963e+06, 3.58443889e+06, 3.02089820e+06, 4.82079126e+06,
4.51896794e+06, 4.02154173e+06, 3.46358527e+06, 3.07673421e+06,
2.80870096e+06, 7.53179410e+06, 4.42138338e+06, 9.08593367e+06,
3.00691183e+06, 8.34010112e+06, 7.44651706e+06, 1.93017175e+06,
```

```
7.79572110e+06, 1.04239342e+07, 4.75729187e+06, 5.90087987e+06,
1.39370897e+07, 3.45942074e+05, 7.90278670e+05, 1.10088455e+06,
5.60566049e+06, 7.19917831e+06, 3.79348414e+07, 2.22384380e+07,
1.40133676e+08, 1.21543800e+06, 8.25017708e+06, 7.75083985e+06,
3.76622051e+06, 5.23813155e+06, 3.85506830e+06, 4.56368349e+06,
5.23379822e+06, 5.43756400e+06, 6.64785660e+06, 3.95081556e+06,
2.94733229e+06, 4.95199416e+06, 4.69818107e+06, 4.86564973e+06,
4.64040329e+06, 3.94305960e+06, 4.61178414e+06, 4.07642036e+06,
5.26336553e+06, 3.52848173e+06, 6.32809768e+06, 5.01382507e+06,
5.03483287e+06, 5.10822495e+06, 4.02557358e+07, 2.00945209e+07,
3.36501980e+06, 4.34892369e+06, 3.93617828e+06, 2.96558121e+06,
1.40604209e+07, 2.12266298e+07, 4.36724776e+06, 6.54385218e+06,
5.55194958e+06, 3.87626707e+06, 3.29377475e+06, 1.51826439e+06,
9.37999925e+05, 3.36073430e+07, 9.07656549e+05, 1.89510601e+07,
1.94816494e+08, 4.75831280e+07, 1.06583672e+08, 1.24617752e+07,
2.44581184e+07, 4.75834899e+08, 6.82491663e+06, 6.06789436e+06,
8.12656041e+06, 1.06683344e+07, 5.72758183e+06, 1.49120622e+07,
4.21211354e+06, 9.76848926e+06, 8.87133956e+06, 2.03235643e+07,
2.36784670e+06, 1.27205110e+07, 4.14394006e+06, 1.56190824e+07,
1.73631546e+07, 7.08569844e+05, 3.51081851e+06, 3.52644062e+06,
4.21911748e+05, 7.36412785e+05, 6.83779729e+05, 9.34570508e+06,
2.97913232e+06, 3.79177370e+06, 1.24840065e+07, 1.04519897e+08,
9.48024335e+05, 2.34031965e+06, 9.92277753e+05, 6.08757611e+07,
3.57570321e+07, 1.62612437e+07, 4.01948923e+07, 4.05616728e+07,
4.14063150e+07, 7.92694106e+07, 1.50006633e+07, 1.45089031e+07,
5.80395868e+06, 1.18598491e+07, 3.24824187e+06, 2.99964244e+06,
6.62106846e+05, 1.00557573e+06, 2.14978966e+06, 1.27117776e+06,
4.23182053e+05, 7.60094000e+05, 3.96055906e+07, 2.20187261e+07,
3.51621868e+07, 1.85421409e+07, 1.80177020e+07, 2.05574460e+07,
2.17109974e+07, 1.21648256e+07, 2.03800012e+07, 2.05764445e+07,
7.90366115e+06, 6.03541385e+07, 1.41614217e+06, 3.06125797e+06,
2.24503067e+07, 3.22979463e+07, 5.77298334e+06, 7.27681976e+06,
5.72277348e+06, 4.90141674e+06, 1.92305063e+07, 5.67383354e+06,
3.77912724e+06, 3.92947762e+06, 2.46157269e+06, 3.07665615e+06,
2.48992135e+06, 3.31623450e+06, 3.63110727e+06, 2.77053443e+07,
5.76350053e+06, 2.15446710e+06, 2.87738003e+06, 1.11224560e+06,
2.07609476e+07, 7.92735489e+06, 9.89938257e+06, 1.33606368e+07,
2.12840266e+07, 1.54804038e+06, 3.77961794e+06, 1.84245907e+07,
2.12522405e+07, 2.35397018e+07, 1.22732600e+05, 4.83443472e+06,
6.37423034e+06, 4.32173712e+07, 5.16333749e+08, 6.89368180e+06,
2.22570555e+06, 2.04592591e+07, 4.82631052e+08, 1.35023802e+07,
1.03059546e+07, 9.79626493e+06])
```

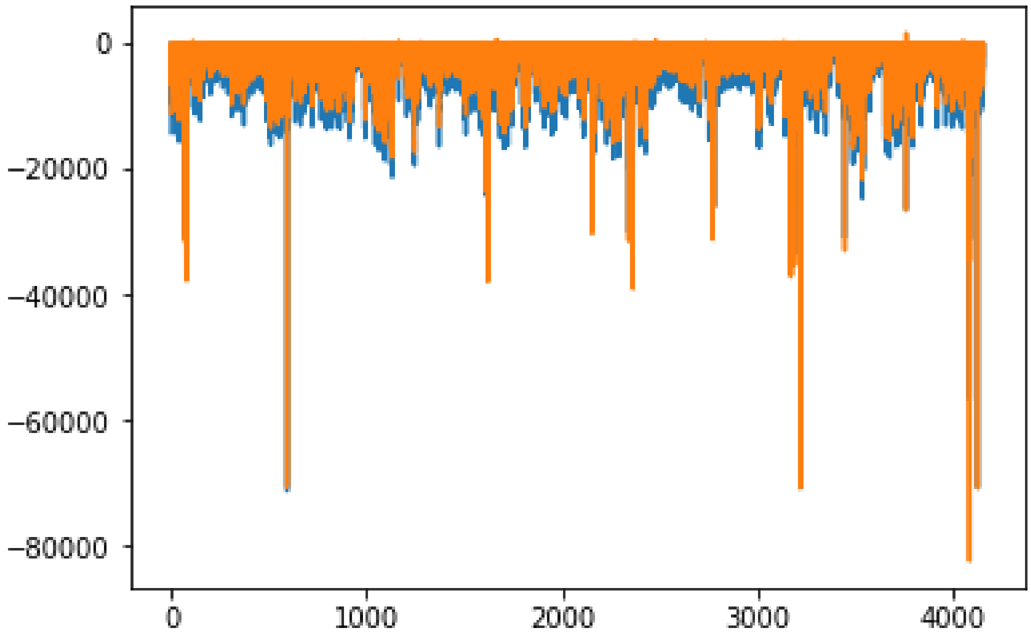
```
In [110]: arr.var()
```

```
Out[110]: 22724635.591483254
```

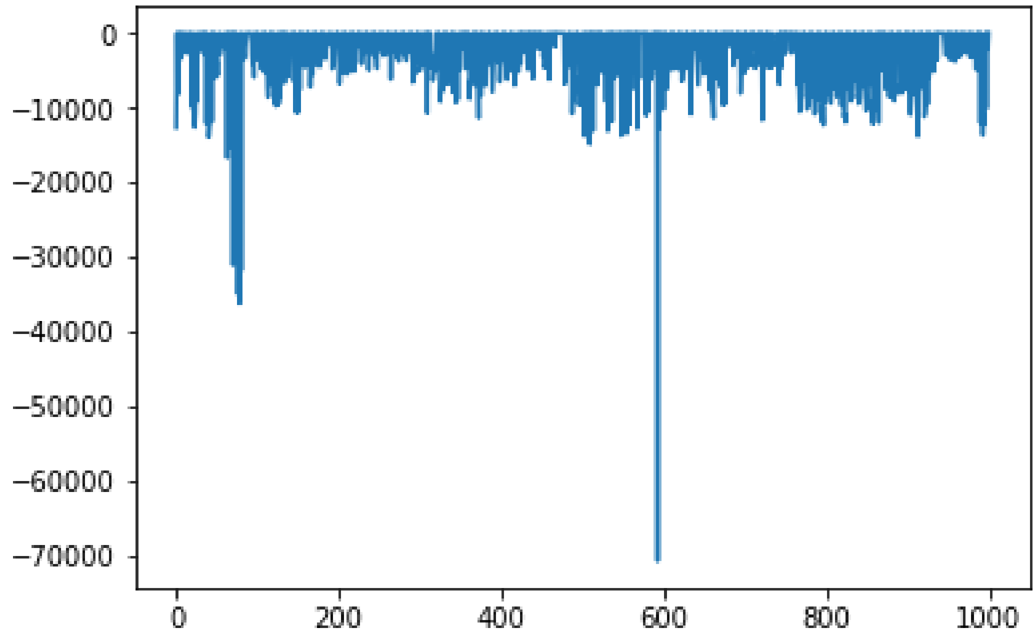
variance accross measures/ mean variance accross species

```
In [111]: arr.mean(0).var()

Out[111]: 1388583.6554276315

In [112]: arr.mean(0)

Out[112]: array([-2219.06645714, -2919.84412463, -2015.18948926, -4288.21485415,
                 -4477.73292545, -4078.47555626, -4687.57869146, -1166.91552409,
                 -2855.27048881])

In [113]: arr.shape

Out[113]: (462, 9)

In [114]: randarr = np.zeros(arr.shape)
          for i in range(arr.shape[1]):
              ind = np.arange(len(arr))
              np.random.shuffle(ind)
              randarr[:,i] = arr[ind,i]

In [115]: randarr

Out[115]: array([[ 4.80310500e-01, -8.20665403e+03,  8.27506000e-01, ...,
                   5.09000000e-01,  8.14207000e-01,  6.83017500e-01],
                 [ 1.81295400e+00, -5.50839054e+03, -7.39452358e+03, ...,
                   3.26664000e-01,  5.96259000e-01,  6.67008500e-01],
                 [ 5.86694000e-01, -2.05983162e+03,  7.60942000e-01, ...,
                  -7.16414784e+03, -1.12670978e+03,  1.31595050e+00],
                 ...,
                 [-8.42159477e+03,  7.28880000e-01, -6.37472343e+03, ...,
                   7.98908000e-01,  5.62661000e-01, -3.39828738e+03],
                 [ 6.15312500e-01,  8.96209500e-01,  6.26267500e-01, ...,
                  -3.29407793e+03,  8.95917000e-01,  8.86115000e-01],
                 [-3.56788777e+03, -2.26486971e+03,  6.17829500e-01, ...,
                   5.02751500e-01,  1.02734500e+00, -2.88560159e+03]])

In [116]: randarr.mean(0).var()-arr.mean(0).var()

Out[116]: 4.656612873077393e-10

In [117]: randarr.var(1).mean()-arr.var(1).mean()

Out[117]: 2882412.743031189

In [97]: arr.var(0)

Out[97]: array([41333202.63856429, 10727555.04310829, 19414596.46349642,
                12230304.06525019, 22099342.25172593, 19543347.84000982,
                28742207.88900693, 21790682.06159877, 16143229.1717399 ])
```

```
In [31]: from matplotlib.pyplot import *

In [118]: plot(data[:,1:])

Out[118]: [<matplotlib.lines.Line2D at 0x109a69810>,
           <matplotlib.lines.Line2D at 0x109a69910>]
```

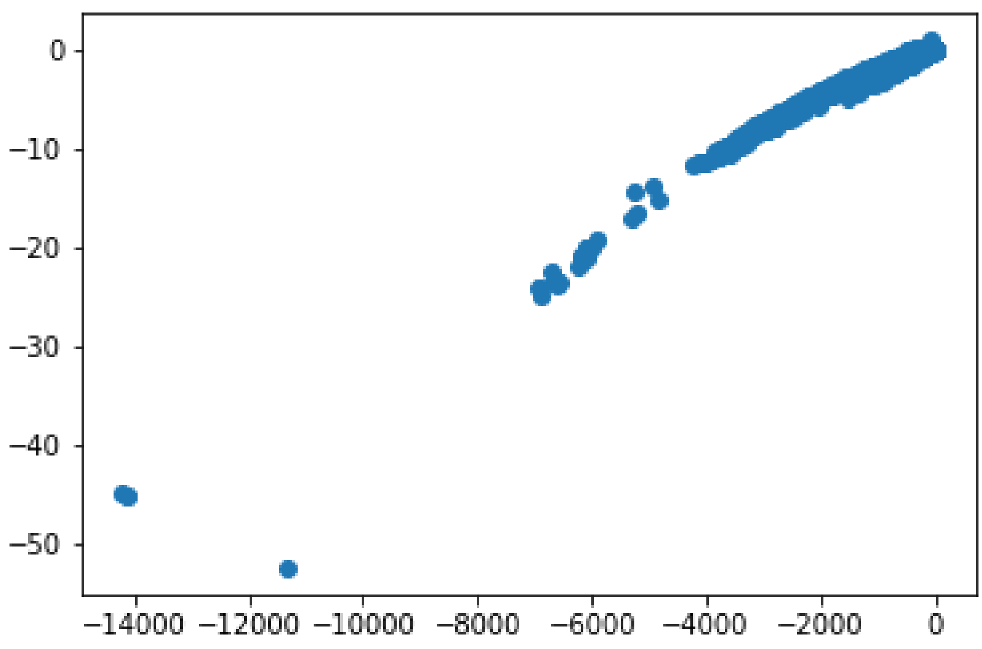



```
In [119]: plot(data[:1000,1:].mean(1))

Out[119]: [<matplotlib.lines.Line2D at 0x109c419d0>]
```




```
In [75]: scatter(data[:,1]/data[:,1].max(),data[:,2]/data[:,2].max())
```

```
Out[75]: <matplotlib.collections.PathCollection at 0x10a510690>
```



Given the repartition of the data (outliers) it will be almost impossible to cluster we should better use direct exploration

```
In [36]: from bokeh.plotting import *
         from bokeh.io import save, show
         from bokeh.models import *
```

```
In [76]: data[:,1].max()
```

```
Out[76]: 5.0115110000000005
```

We can see that it is confirmed here, even if it does not cluster, it is more a relation of variance and

```
In [77]: X = -1
         species = [1,2,3,4,5,6,7,8,9,10]
         col = ['#f39c12', "#1abc9c", "#3498db", "#2ecc71", "#9b59b6", '#34495e', '#492000', '
         '#043927', '#8F9779', '#f1c40f', '#e67e22', '#e74c3c', '#7f8c8d']
         source = ColumnDataSource(data=dict(x = data[:X,1]/data[:,1].min(),
                                             y = data[:X,2]/data[:,2].min(),
                                             names = np.array(datatot[0])[:X],
                                             colors= [col[0] if i-30 not in species else\
                                                     col[i-30] for i in np.array(datatot[0])[:
         output_notebook()
         hover = HoverTool(tooltips=[("species: ", "@names")])
         p = figure(title="Plot of the regressors",
                    tools=[hover, WheelZoomTool(), PanTool(), SaveTool(), ResetTool()],
                    plot_width=800, plot_height=600)
         p.circle(x='x', y='y', source=source, size=10,color='colors')
         show(p)
```

```
In [47]: from sklearn.neighbors import KNeighborsClassifier
         from sklearn.gaussian_process import GaussianProcessClassifier
         from sklearn.svm import SVC
         svc = SVC(C=1.0, kernel='rbf', degree=40, gamma='auto', coef0=0.0, shrinking=True, pro
                tol=0.001, cache_size=200, class_weight=None, verbose=False,
                max_iter=-1, random_state=None).fit(data[:,1:], data[:,0].astype(int))
         neigh = KNeighborsClassifier(n_neighbors=462).fit(data[:,1:], data[:,0].astype(int))
```

```
In [48]: neigh.score(data[:,1:], data[:,0].astype(int))
```

```
Out[48]: 0.010582010582010581
```

```
In [49]: svc.score(data[:,1:], data[:,0].astype(int))
```

```
Out[49]: 0.023569023569023569023
```

0.5 we are now looking at the full dataset

In [6]:
```
datatot = pd.DataFrame({'': []})
for i in range(5,15):
    datatot = pd.concat([datatot, pd.read_csv('parameterFitsBacteria/nonlinfit'+str(i)
datatot
```

Out[6]:

	0	1	2	3	4	5	
0	1.0	E	-62792.363178	-41171.105942	0.387295	0.053507	NaN
1	1.0	H	0.884914	0.625735	0.025491	0.000321	NaN
2	1.0	Q	-34078.001362	-14052.539028	0.223853	0.072042	NaN
3	1.0	F	-7417.175624	-7323.722600	0.052349	0.073091	NaN
4	1.0	Y	-11366.590431	-8018.684126	0.055767	0.073580	NaN
5	1.0	C	0.968817	0.864984	0.028707	0.000434	NaN
6	1.0	N	-14509.094395	-5446.502499	0.083904	0.075813	NaN
7	1.0	K	1.851802	1.645036	0.003847	0.001664	NaN
8	1.0	D	0.907276	0.647303	0.024459	0.000370	NaN
9	2.0	E	1.094369	0.975947	0.021372	0.000522	NaN
10	2.0	H	-19868.512605	-19868.590216	0.028694	0.077360	NaN
11	2.0	Q	0.879460	0.826876	0.036053	0.000334	NaN
12	2.0	F	-17245.573218	-10324.527574	0.076132	0.071973	NaN
13	2.0	Y	-3340.967483	-825.185441	0.041597	0.078089	NaN
14	2.0	C	-13539.463661	-2431.330570	0.082519	0.080565	NaN
15	2.0	N	-4876.630557	-4453.878513	0.024482	0.078637	NaN
16	2.0	K	0.953721	0.839989	0.028875	0.000569	NaN
17	2.0	D	0.955444	0.927154	0.032383	0.000105	NaN
18	3.0	E	0.648022	-0.061252	0.012024	0.000051	NaN
19	3.0	H	-48419.842753	-29259.971830	0.287958	0.059913	NaN
20	3.0	Q	0.718656	-0.090369	0.008074	0.000410	NaN
21	3.0	F	-49899.734542	-29418.743313	0.234177	0.109418	NaN
22	3.0	Y	-61797.377808	-40902.629785	0.354978	0.057879	NaN
23	3.0	C	-56755.199293	-35080.077306	0.358516	0.056577	NaN
24	3.0	N	-43746.892027	-22584.086268	0.251196	0.069826	NaN
25	3.0	K	0.801362	-0.063157	0.006064	0.000021	NaN
26	3.0	D	-41138.321246	-23718.158629	0.161458	0.118123	NaN
27	4.0	E	-102.507859	6077.565657	0.048571	0.085363	NaN
28	4.0	H	0.856398	0.638767	0.028769	0.000050	NaN
29	4.0	Q	-9391.532128	-5847.378522	0.063551	0.074700	NaN
...	...	...	...	...	...	...	..
4128	459.0	N	0.839945	0.699710	0.000050	0.000131	NaN
4129	459.0	K	0.436508	0.442044	0.004320	0.000214	NaN
4130	459.0	D	0.667253	0.528199	0.000174	0.000182	NaN
4131	460.0	E	0.406190	0.634501	0.021962	0.000381	NaN
4132	460.0	H	0.985126	0.653652	0.000001	0.000645	NaN
4133	460.0	Q	0.442645	0.726891	0.021257	0.000436	NaN
4134	460.0	F	0.816020	0.694422	0.000074	0.000418	NaN
4135	460.0	Y	0.930689	0.838514	0.000041	0.000352	NaN
4136	460.0	C	1.021081	0.611306	0.000000	0.000733	NaN
4137	460.0	N	0.677209	0.490812	0.000091	0.000128	NaN

4138	460.0	K	0.869084	0.978546	0.000364	0.000636	NaN
4139	460.0	D	0.781626	0.442542	0.000005	0.000741	NaN
4140	461.0	E	0.757875	0.587377	0.000063	0.000267	NaN
4141	461.0	H	0.571183	0.688012	0.003767	0.000529	NaN
4142	461.0	Q	0.815468	0.722794	0.000099	0.000167	NaN
4143	461.0	F	0.536594	0.908253	0.016678	0.000051	NaN
4144	461.0	Y	0.417824	0.662918	0.021889	0.000208	NaN
4145	461.0	C	0.577827	0.928626	0.011756	0.001116	NaN
4146	461.0	N	0.480479	0.637362	0.009399	0.000121	NaN
4147	461.0	K	0.564361	0.378703	0.000202	0.000135	NaN
4148	461.0	D	0.681498	0.810177	0.001866	0.000187	NaN
4149	462.0	E	0.658475	0.512497	0.000175	0.000150	NaN
4150	462.0	H	0.557306	0.676898	0.004250	0.000283	NaN
4151	462.0	Q	0.702023	0.639770	0.000324	0.000139	NaN
4152	462.0	F	1.490475	2.449030	0.000149	0.003616	NaN
4153	462.0	Y	0.503267	0.801784	0.015582	0.000323	NaN
4154	462.0	C	0.480082	0.733640	0.015162	0.000319	NaN
4155	462.0	N	0.560325	0.737822	0.005977	0.000271	NaN
4156	462.0	K	0.696857	0.477855	0.000052	0.000184	NaN
4157	462.0	D	0.755843	0.842702	0.000732	0.000813	NaN

[41550 rows x 7 columns]