Using_PyMySQL

June 1, 2016

```
In [1]: # Just to know last time this was run:
        import time
        print time.ctime()

Wed Jun  1 17:22:35 2016
```

1 J Using PyMySQL to access MySQL databases

This package contains a pure-Python MySQL client library. In this sense, it does not need to have access to mysql header or library, which is the case for the mysqldb package. The goal of PyMySQL is to be a drop-in replacement for MySQLdb and work on CPython, PyPy, IronPython and Jython.

It is installed with "pip install pymysql"

We first import the usual libraries

```
In [2]: %matplotlib inline
        import numpy as np
        import matplotlib.pyplot as plt

/Users/christophemorisset/anaconda/lib/python2.7/site-packages/matplotlib/font_mana
  warnings.warn('Matplotlib is building the font cache using fc-list. This may take
```

This is the import of the library used to connect to MySQl database

```
In [3]: import pymysql
```

First you need to connect to a database. In our example, we will use the 3MdB database, which needs a password. https://sites.google.com/site/mexicanmillionmodels/

1.0.1 Connect to the database

```
In [5]: user_password = '***' # ask me for the password :-)

In [6]: # We create a connector to the database
        connector = pymysql.connect(host='132.248.1.102', port=3306, user='OVN_user
```

1.0.2 Use a cursor to send query and receive results

```
In [7]: # The cursor is used to send and receive the quesries to the databse
        cur = connector.cursor()
```

```
In [8]: # Send the query to be executed. It returns the number of lines of the resu
        cur.execute('select * from `lines` limit 15')
```

```
Out[8]: 15
```

```
In [9]: # get a description of the columns of the query results
        cur.description
```

```
Out[9]: ((u'Nl', 8, None, 20, 20, 0, False),
         (u'label', 253, None, 15, 15, 0, True),
         (u'id', 253, None, 20, 20, 0, True),
         (u'lambda', 5, None, 22, 22, 31, True),
         (u'name', 253, None, 40, 40, 0, False),
         (u'used', 3, None, 2, 2, 0, True))
```

```
In [10]: # fech all the resulting data into a variable
         lines = cur.fetchall()
```

```
In [11]: # close the cursor once used
         cur.close()
```

```
In [12]: # the result is in a form of tuple of tuples
         print lines
```

```
((1, 'BAC___3646A', 'Bac ', 3646.0, 'BalmHead', 1), (2, 'COUT__3646A', 'cout', 364
```

```
In [13]: # Each element of the first level tuple is a tuple corresponding to a row
         print len(lines)
         print lines[0]
```

```
15
(1, 'BAC___3646A', 'Bac ', 3646.0, 'BalmHead', 1)
```

1.0.3 Using a cursor that returns a dictionary

```
In [14]: cur_dic = connector.cursor(pymysql.cursors.DictCursor)
```

```
In [15]: cur_dic.execute('select * from `lines` limit 15')
```

```
Out[15]: 15
```

```
In [16]: lines_dic = cur_dic.fetchall()
```

```
In [17]: print lines_dic
```

```
[{u'used': 1, u'Nl': 1, u'name': 'BalmHead', u'label': 'BAC___3646A', u'id': 'Bac '
```

```
In [18]: # Each element of the table is a dictionary corresponding to a row od the
         print lines_dic[0]
```

```
{u'used': 1, u'Nl': 1, u'name': 'BalmHead', u'label': 'BAC___3646A', u'id': 'Bac ',
```

```
In [19]: # One can easily create a new dictionary than hold the data in columns, be
         new_dic = {k:np.array([d[k] for d in lines_dic]) for k in lines_dic[0].key
```

```
In [20]: # The names of the columns are the names use in the database
         new_dic['lambda']
```

```
Out[20]: array([  3.64600000e+03,   3.64600000e+03,   3.64600000e+03,
                  4.86100000e+03,   4.86100000e+03,   6.56300000e+03,
                  4.34000000e+03,   4.10200000e+03,   3.97000000e+03,
                  3.83500000e+03,   1.21600000e+03,   4.05100000e+00,
                  2.62500000e+00,   7.45800000e+00,   5.87600000e+03])
```

```
In [21]: # One can also transform the results into a numpy recarray.
         # First step: create a table from the dictionary
         lines_tab = [e.values() for e in lines_dic]
         lines_tab
```

```
Out[21]: [[1, 1, 'BalmHead', 'BAC___3646A', 'Bac ', 3646.0],
          [1, 2, 'OutwardBalmPeak', 'COUT__3646A', 'cout', 3646.0],
          [1, 3, 'ReflectedBalmPeak', 'CREF__3646A', 'cref', 3646.0],
          [1, 4, 'H I 4861', 'H__1__4861A', 'H  1', 4861.0],
          [1, 5, 'H I 4861', 'TOTL__4861A', 'TOTL', 4861.0],
          [1, 6, 'H I 6563', 'H__1__6563A', 'H  1', 6563.0],
          [1, 7, 'H I 4340', 'H__1__4340A', 'H  1', 4340.0],
          [1, 8, 'H I 4102', 'H__1__4102A', 'H  1', 4102.0],
          [1, 9, 'H I 3970', 'H__1__3970A', 'H  1', 3970.0],
          [1, 10, 'H I 3835', 'H__1__3835A', 'H  1', 3835.0],
          [1, 11, 'H I 1216', 'H__1__1216A', 'H  1', 1216.0],
          [1, 12, 'H I 4.051m', 'H__1_4051M', 'H  1', 4.051],
          [1, 13, 'H I 2.625m', 'H__1_2625M', 'H  1', 2.625],
          [1, 14, 'H I 7.458m', 'H__1_7458M', 'H  1', 7.458],
          [1, 15, 'He I 5876', 'HE_1__5876A', 'He 1', 5876.0]]
```

```
In [22]: # Second step: transform the table into a numpy recarray, using the names
         res = np.rec.fromrecords(lines_tab, names = lines_dic[0].keys())
```

```
In [23]: res
```

```
Out[23]: rec.array([(1, 1, 'BalmHead', 'BAC___3646A', 'Bac ', 3646.0),
          (1, 2, 'OutwardBalmPeak', 'COUT__3646A', 'cout', 3646.0),
```

```
(1, 3, 'ReflectedBalmPeak', 'CREF__3646A', 'cref', 3646.0),
(1, 4, 'H I 4861', 'H__1__4861A', 'H  1', 4861.0),
(1, 5, 'H I 4861', 'TOTL__4861A', 'TOTL', 4861.0),
(1, 6, 'H I 6563', 'H__1__6563A', 'H  1', 6563.0),
(1, 7, 'H I 4340', 'H__1__4340A', 'H  1', 4340.0),
(1, 8, 'H I 4102', 'H__1__4102A', 'H  1', 4102.0),
(1, 9, 'H I 3970', 'H__1__3970A', 'H  1', 3970.0),
(1, 10, 'H I 3835', 'H__1__3835A', 'H  1', 3835.0),
(1, 11, 'H I 1216', 'H__1__1216A', 'H  1', 1216.0),
(1, 12, 'H I 4.051m', 'H__1_4051M', 'H  1', 4.051),
(1, 13, 'H I 2.625m', 'H__1_2625M', 'H  1', 2.625),
(1, 14, 'H I 7.458m', 'H__1_7458M', 'H  1', 7.458),
(1, 15, 'He I 5876', 'HE_1__5876A', 'He 1', 5876.0)],
        dtype=[(u'used', '<i8'), (u'Nl', '<i8'), (u'name', 'S17'), (u'la
```

```
In [24]: res['lambda']
```

```
Out[24]: array([  3.64600000e+03,   3.64600000e+03,   3.64600000e+03,
                  4.86100000e+03,   4.86100000e+03,   6.56300000e+03,
                  4.34000000e+03,   4.10200000e+03,   3.97000000e+03,
                  3.83500000e+03,   1.21600000e+03,   4.05100000e+00,
                  2.62500000e+00,   7.45800000e+00,   5.87600000e+03])
```

1.0.4 Example of plotting the result of a query

```
In [25]: # Send the query
         N = cur_dic.execute('select O__3__5007A, N__2__6584A, H__1__6563A, oxygen
```

```
In [26]: print N
```

```
7854
```

```
In [27]: # obtain the results as a dictionnary
         res = cur_dic.fetchall()
```

```
In [28]: # transform the disctionary into a recarray
         data = np.rec.fromrecords([e.values() for e in res], names = res[0].keys()
```

```
In [29]: # check the data
         data
```

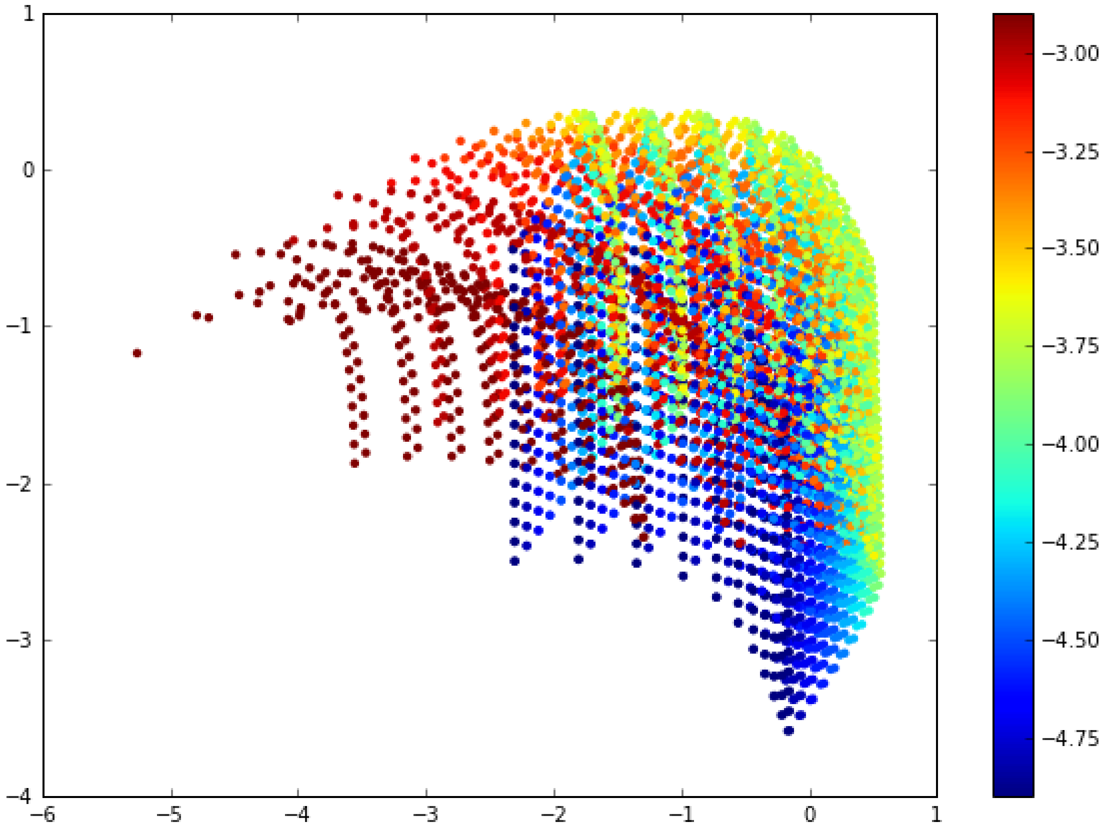
```
Out[29]: rec.array([(1.13306243836e+58, 8.465943086e+58, -3.1, 3.15741653467e+58),
 (3.42011987292e+59, 3.82678097448e+59, -4.7, 1.96658128904e+58),
 (1.9919317079e+55, 2.95364632532e+58, -2.9, 8.79993595982e+57), ...,
 (1.75269190656e+60, 5.79356475056e+59, -3.7, 5.08981089096e+58),
 (1.37202884837e+60, 5.15976659165e+59, -4.1, 3.20261785304e+57),
 (1.52244147812e+60, 5.27404255136e+59, -4.0, 3.89222406128e+58)],
         dtype=[(u'O__3__5007A', '<f8'), (u'H__1__6563A', '<f8'), (u'oxyg
```

In [30]: data['O__3__5007A']

Out[30]: array([1.13306244e+58,   3.42011987e+59,   1.99193171e+55, ...,
                 1.75269191e+60,   1.37202885e+60,   1.52244148e+60])

In [31]:
```
# Plot the results, using a column as color code
fig, ax = plt.subplots(figsize=(10,7))
scat = ax.scatter(np.log10(data['O__3__5007A'] / data['H__1__6563A']), np.
            c=data['oxygen'], edgecolor='none')
fig.colorbar(scat)
```

Out[31]: <matplotlib.colorbar.Colorbar at 0x1108c2650>



In [32]:
```
# Disconnect cursor and connector
cur_dic.close()
connector.close()
```

1.0.5 Easier way using pyCloudy library

In [33]:
```
# Import pyCloudy
import pyCloudy as pc
# pyCloudy version must be > 0.8.43
print pc.__version__
```

```
0.8.59b2
```

```
In [35]: pc.config.db_connector = 'PyMySQL'
         # Define the parameters of the connection in a dictionnary
         OVN_dic= {'host' : '132.248.1.102',
                   'user_name' : 'OVN_user',
                   'user_passwd' : '***',
                   'base_name' : '3MdB'}
         # Instantiate an object that will deal with the database connections and c
         db = pc.MdB(OVN_dic)
```

```
In [36]: res, N = db.exec_dB('select ref, count(*) from tab group by ref')
         print res
         print N
```

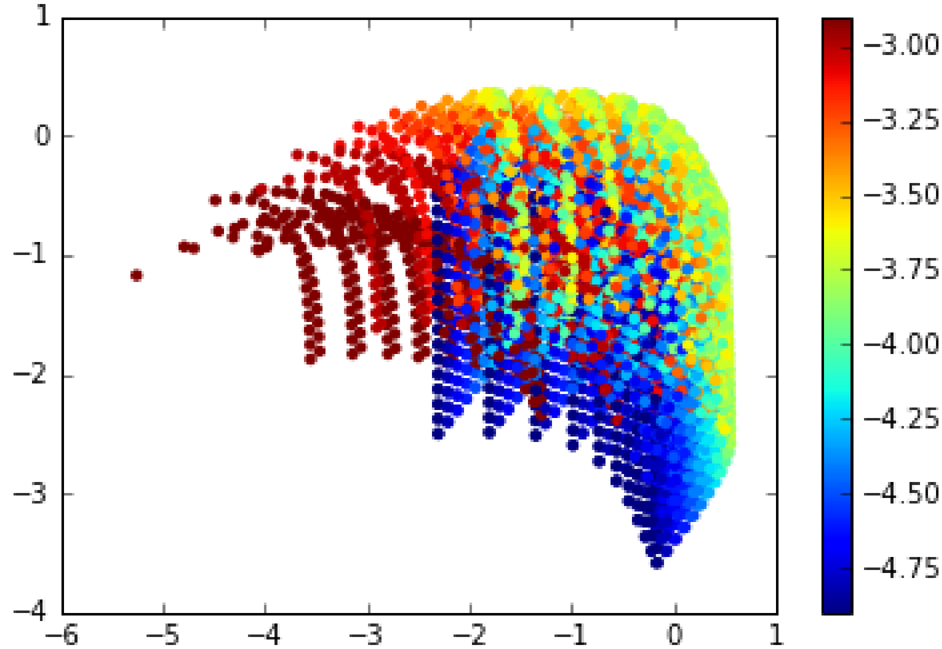
```
[{u'count(*)': 31500, u'ref': 'BOND'}, {u'count(*)': 85800, u'ref': 'CALIFA'}, {u'c
7
```

```
In [37]: # Obtain the result of a select command directly as a recarray
         data, N = db.select_dB(select_='O__3__5007A, N__2__6584A, H__1__6563A, oxy
                                limit_=None, format_='rec')
```

```
In [38]: # Check the data
         data
```

```
Out[38]: rec.array([(1.13306243836e+58, 8.465943086e+58, -3.1, 3.15741653467e+58),
          (3.42011987292e+59, 3.82678097448e+59, -4.7, 1.96658128904e+58),
          (1.9919317079e+55, 2.95364632532e+58, -2.9, 8.79993595982e+57), ...,
          (1.75269190656e+60, 5.79356475056e+59, -3.7, 5.08981089096e+58),
          (1.37202884837e+60, 5.15976659165e+59, -4.1, 3.20261785304e+57),
          (1.52244147812e+60, 5.27404255136e+59, -4.0, 3.89222406128e+58)],
                   dtype=[(u'O__3__5007A', '<f8'), (u'H__1__6563A', '<f8'), (u'oxyg
```

```
In [39]: # Make the same plot
         fig, ax = plt.subplots()
         scat = ax.scatter(np.log10(data['O__3__5007A'] / data['H__1__6563A']), np.
                      c=data['oxygen'], edgecolor='none')
         cb = fig.colorbar(scat)
```




```
In [40]: # The same, using pandas to store the results.
         # needs to have pandas installed : pip install --no-deps pandas
         data, N = db.select_dB(select_='O__3__5007A, N__2__6584A, H__1__6563A, oxy
                                limit_=None, format_='pandas')

In [41]: print type(data)
         data

<class 'pandas.core.frame.DataFrame'>
```

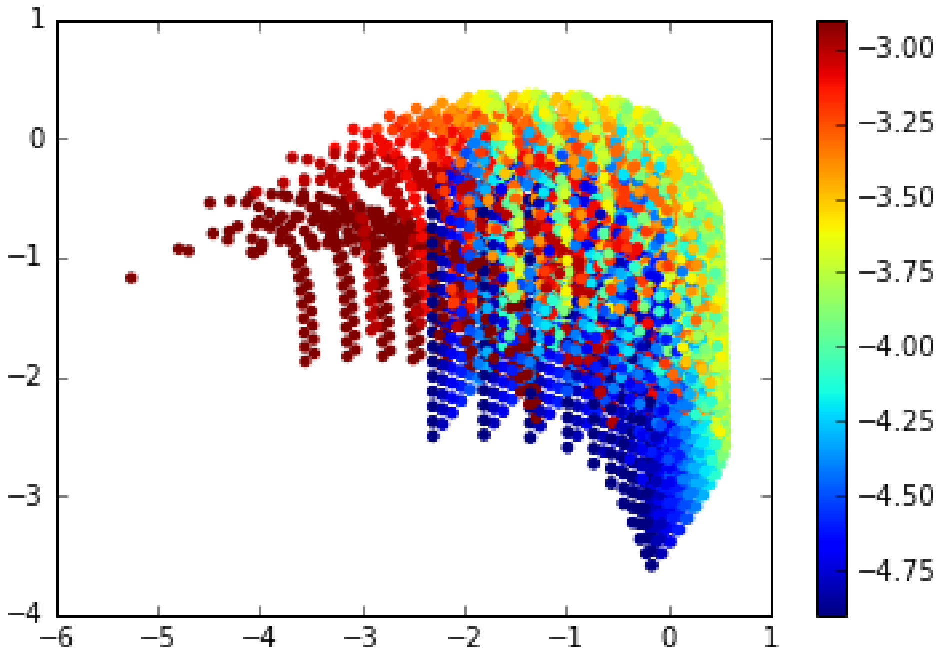
Out[41]:

	O__3__5007A	N__2__6584A	H__1__6563A	oxygen
0	1.133062e+58	3.157417e+58	8.465943e+58	-3.1
1	3.420120e+59	1.966581e+58	3.826781e+59	-4.7
2	1.991932e+55	8.799936e+57	2.953646e+58	-2.9
3	1.094455e+59	1.517612e+57	7.586338e+58	-3.9
4	1.300403e+56	9.594305e+56	1.593603e+58	-3.0
5	4.547320e+59	6.234337e+56	2.780188e+59	-4.3
6	3.482729e+59	4.703550e+56	3.828982e+59	-4.7
7	4.364881e+58	1.791885e+58	7.633109e+58	-4.5
8	7.768910e+58	6.520140e+57	1.388706e+59	-3.2
9	6.521827e+59	3.655320e+58	2.850901e+59	-4.0
10	5.827025e+58	2.315942e+57	7.668195e+58	-4.4
11	9.819971e+58	3.263721e+58	1.369624e+59	-3.3
12	1.216272e+58	1.322807e+58	4.584762e+58	-4.7
13	4.319418e+59	2.171798e+56	4.962521e+59	-4.8

14	8.732360e+59	6.007494e+58	7.277213e+59	-3.2
15	1.948541e+59	9.667164e+57	1.312013e+59	-3.5
16	1.218185e+59	3.149880e+57	5.708760e+59	-3.0
17	1.943526e+58	3.791206e+57	4.586533e+58	-4.5
18	4.011769e+55	3.750527e+57	4.869870e+57	-3.3
19	1.677233e+59	2.198075e+57	1.232347e+59	-4.2
20	7.564738e+59	3.233375e+57	3.091887e+59	-3.6
21	8.823437e+57	2.065070e+56	4.700258e+58	-4.9
22	3.226162e+57	4.962798e+58	8.706764e+58	-3.1
23	2.365680e+55	1.141878e+57	4.890240e+57	-4.9
24	8.174137e+59	1.582741e+57	3.000036e+59	-3.7
25	1.150354e+60	1.139883e+59	4.314785e+59	-3.7
26	2.211890e+56	5.419716e+57	5.111543e+58	-2.9
27	7.803436e+59	1.548125e+58	3.012500e+59	-3.7
28	5.340369e+58	1.015961e+58	1.260066e+59	-4.8
29	1.435415e+56	3.100620e+58	1.592877e+59	-2.9
...	...	...	...	...
7824	1.972405e+60	8.609873e+57	5.901718e+59	-3.6
7825	4.310580e+57	1.923966e+56	2.697913e+58	-4.7
7826	1.008846e+59	2.983662e+57	7.566729e+58	-4.0
7827	4.018091e+59	9.599803e+58	7.998521e+59	-3.1
7828	9.895354e+57	1.213889e+59	6.651922e+59	-3.0
7829	8.789083e+59	6.046069e+56	3.819032e+59	-4.2
7830	2.937330e+57	2.126139e+57	1.492873e+59	-2.9
7831	9.264124e+58	2.688380e+57	7.855083e+58	-3.5
7832	2.407456e+58	2.687464e+56	4.575668e+58	-4.4
7833	8.588467e+58	7.640683e+58	7.745671e+58	-3.7
7834	1.322239e+55	9.198548e+57	2.967463e+58	-2.9
7835	2.374835e+57	1.561329e+58	1.500132e+58	-3.5
7836	2.630657e+59	5.677059e+57	2.792541e+59	-4.6
7837	8.598621e+59	9.858896e+58	4.227971e+59	-3.9
7838	1.459214e+59	4.216686e+57	1.234976e+59	-4.3
7839	2.409456e+55	6.488206e+56	4.905731e+57	-4.9
7840	2.478336e+59	2.729407e+57	1.253125e+59	-3.8
7841	3.159983e+58	1.275284e+58	8.214903e+58	-3.2
7842	9.530954e+57	1.277263e+58	2.599781e+58	-4.2
7843	9.218411e+58	3.574314e+57	7.856340e+58	-3.5
7844	3.804036e+59	2.093539e+58	2.776236e+59	-4.4
7845	5.177660e+56	2.514500e+58	1.553671e+58	-3.3
7846	1.956973e+58	2.850425e+57	4.590933e+58	-4.5
7847	8.346006e+55	1.451319e+57	4.784272e+57	-3.4
7848	6.879302e+59	3.689720e+57	3.181142e+59	-3.5
7849	3.386232e+59	8.655237e+58	3.477404e+59	-3.3
7850	5.123895e+56	1.386879e+58	8.461529e+57	-3.6
7851	1.752692e+60	5.089811e+58	5.793565e+59	-3.7
7852	1.372029e+60	3.202618e+57	5.159767e+59	-4.1
7853	1.522441e+60	3.892224e+58	5.274043e+59	-4.0

```
[7854 rows x 4 columns]
```

In [42]:
```
# Make the same plot
fig, ax = plt.subplots()
scat = ax.scatter(np.log10(data['O__3__5007A'] / data['H__1__6563A']), np.
            c=data['oxygen'], edgecolor='none')
cb = fig.colorbar(scat)
```



In [43]:
```
db.close_dB()
```

1.0.6 Using pyCloudy to save the result in a file

In [44]:
```
from pyCloudy.db.MdB import MdB_subproc
```

In [45]:
```
db = MdB_subproc(OVN_dic)
```

In [46]:
```
# The queries used here are send by calling the linux mysql and sending th
# Notice the outfile.
db.select_dB(select_='O__3__5007A, N__2__6584A, H__1__6563A, oxygen', from
                    limit_=None, outfile='query_res.dat')
db.close_dB()
```

In [47]:
```
# The outfile contains the result of the query
!head query_res.dat
```

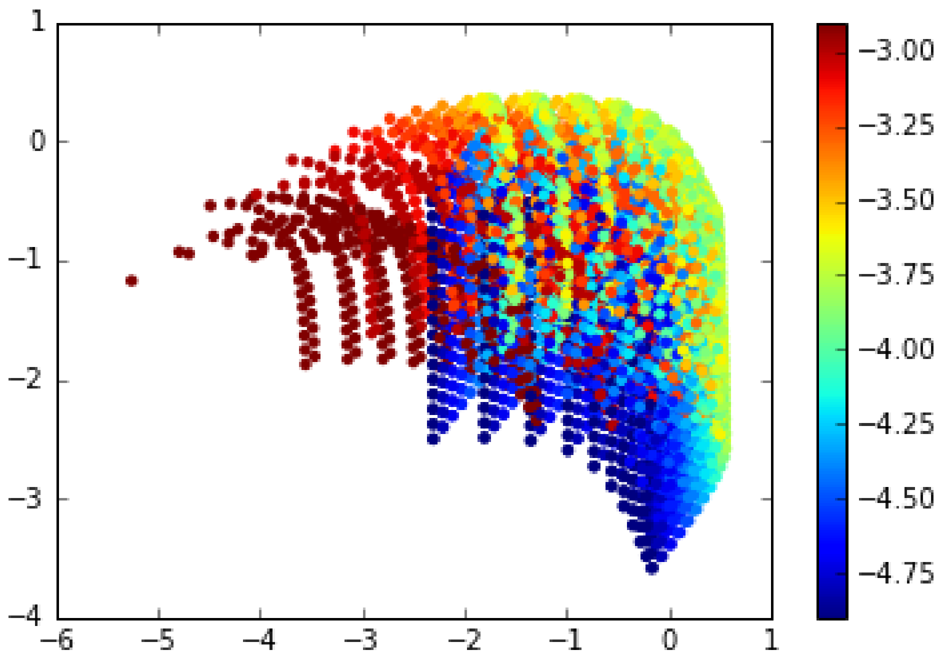
O__3__5007A	N__2__6584A	H__1__6563A	oxygen
1.13306243836e58	3.15741653467e58	8.465943086e58	-3.1
3.42011987292e59	1.96658128904e58	3.82678097448e59	-4.7
1.9919317079e55	8.79993595982e57	2.95364632532e58	-2.9
1.09445528168e59	1.51761218935e57	7.58633813601e58	-3.9
1.3004028118e56	9.59430498831e56	1.59360285671e58	-3
4.54731969943e59	6.23433665474e56	2.780187567e59	-4.3
3.48272851916e59	4.70354986736e56	3.82898210273e59	-4.7
4.36488135054e58	1.79188530745e58	7.63310885003e58	-4.5
7.76890989905e58	6.52013985859e57	1.38870636951e59	-3.2

```
In [48]: data = np.genfromtxt('query_res.dat', names=True, dtype=None)

In [49]: data

Out[49]: array([(1.13306243836e+58, 3.15741653467e+58, 8.465943086e+58, -3.1),
                (3.42011987292e+59, 1.96658128904e+58, 3.82678097448e+59, -4.7),
                (1.9919317079e+55, 8.79993595982e+57, 2.95364632532e+58, -2.9), ...
                (1.75269190656e+60, 5.08981089096e+58, 5.79356475056e+59, -3.7),
                (1.37202884837e+60, 3.20261785304e+57, 5.15976659165e+59, -4.1),
                (1.52244147812e+60, 3.89222406128e+58, 5.27404255136e+59, -4.0)],
               dtype=[('O__3__5007A', '<f8'), ('N__2__6584A', '<f8'), ('H__1__6563A

In [50]: # Make the same plot
         fig, ax = plt.subplots()
         scat = ax.scatter(np.log10(data['O__3__5007A'] / data['H__1__6563A']), np.
                     c=data['oxygen'], edgecolor='none')
         cb = fig.colorbar(scat)
```



1.0.7 Using pandas library

```
In [51]: import pandas as pd
         import pymysql
         import matplotlib.pyplot as plt

         co = pymysql.connect(host='132.248.1.102', db='3MdB', user='OVN_user', pas
         res = pd.read_sql("select log10(N__2__6584A/H__1__6563A) as n2, log10(O__3
         co.close()

In [52]: plt.scatter(res['n2'], res['o3'], c=res['O'], edgecolor='None')

Out[52]: <matplotlib.collections.PathCollection at 0x11dae7ad0>
```