pandas-demo

April 4, 2018

0.0.1 Pandas library

numpy = n-dimensional arrays, like Matlab.

pandas dataframes = 2-dimensional labelled data containers.

Can be used to manipulate large tables of dissimilar data, like Excel.

Features: - Labelled indexes (columns, rows) - Multi-level indexing - Capable of holding any data type (integers, strings, floats, Python objects, . . .)

In [8]:
```
import pandas as pd # import pandas data manipulation library
import numpy as np # import numpy numerical library
import matplotlib.pyplot as plt # import matplotlib plotting library
import seaborn # import seaborn for advanced plotting
```

In [9]:
```
data = pd.read_csv('weatherdata.csv') # import weather data to data
data
```

Out[9]:

	Date	HH:MM	Dry Bulb Temp	Dew Point Temp	Relative Humidity	\
0	01/01/70	1:00	0.1	-0.6	94	
1	01/01/70	2:00	0.5	0.0	96	
2	01/01/70	3:00	0.8	0.5	98	
3	01/01/70	4:00	0.8	0.7	99	
4	01/01/70	5:00	0.8	0.8	100	
5	01/01/70	6:00	0.7	0.7	100	
6	01/01/70	7:00	1.7	1.7	100	
7	01/01/70	8:00	2.2	2.2	100	
8	01/01/70	9:00	2.8	2.2	96	
9	01/01/70	10:00	3.3	2.2	92	
10	01/01/70	11:00	4.4	2.8	89	
11	01/01/70	12:00	4.4	2.8	89	
12	01/01/70	13:00	6.1	3.3	82	
13	01/01/70	14:00	6.1	2.8	79	
14	01/01/70	15:00	5.6	2.8	82	
15	01/01/70	16:00	5.0	3.3	89	
16	01/01/70	17:00	5.0	4.4	96	
17	01/01/70	18:00	4.4	3.3	93	
18	01/01/70	19:00	4.4	3.3	93	
19	01/01/70	20:00	4.4	3.9	97	
20	01/01/70	21:00	3.3	2.2	92	

21	01/01/70	22:00	4.4	2.8	89
22	01/01/70	23:00	5.0	3.3	89
23	01/01/70	0:00	4.4	3.9	97
24	02/01/70	1:00	4.4	4.4	100
25	02/01/70	2:00	4.4	4.4	100
26	02/01/70	3:00	4.4	4.4	100
27	02/01/70	4:00	3.9	3.9	100
28	02/01/70	5:00	3.9	3.9	100
29	02/01/70	6:00	3.9	3.9	100
...	...	...	...	...	...
8730	30/12/81	19:00	0.2	-1.3	88
8731	30/12/81	20:00	2.0	-3.2	66
8732	30/12/81	21:00	1.8	-4.0	63
8733	30/12/81	22:00	1.8	-4.0	63
8734	30/12/81	23:00	2.1	-0.4	83
8735	30/12/81	0:00	2.8	0.2	83
8736	31/12/81	1:00	2.4	0.1	85
8737	31/12/81	2:00	1.9	-1.0	80
8738	31/12/81	3:00	2.0	-1.5	76
8739	31/12/81	4:00	2.6	-0.4	80
8740	31/12/81	5:00	3.7	0.7	81
8741	31/12/81	6:00	4.7	1.3	79
8742	31/12/81	7:00	4.8	1.9	81
8743	31/12/81	8:00	4.6	1.3	79
8744	31/12/81	9:00	4.4	1.3	80
8745	31/12/81	10:00	4.2	-0.3	72
8746	31/12/81	11:00	4.4	-1.3	66
8747	31/12/81	12:00	4.2	-1.9	63
8748	31/12/81	13:00	3.6	-0.9	72
8749	31/12/81	14:00	2.7	0.1	83
8750	31/12/81	15:00	2.2	1.1	92
8751	31/12/81	16:00	0.8	0.5	98
8752	31/12/81	17:00	1.0	1.0	100
8753	31/12/81	18:00	0.9	0.0	94
8754	31/12/81	19:00	0.7	-0.4	92
8755	31/12/81	20:00	0.3	-0.7	92
8756	31/12/81	21:00	0.0	-1.1	91
8757	31/12/81	22:00	-0.4	-1.4	92
8758	31/12/81	23:00	-0.4	-1.5	91
8759	31/12/81	0:00	-0.2	-1.2	92

	Atmospheric Pressure	Extraterrestrial Horizontal Radiation	\
0	101810	0	
1	102500	0	
2	102970	0	
3	103190	0	
4	103270	0	
5	103350	0	

6	103170	0
7	103140	0
8	103190	59
9	103210	224
10	103180	350
11	103150	423
12	103090	439
13	103090	398
14	103090	300
15	103090	154
16	103070	12
17	103050	0
18	103040	0
19	103040	0
20	102990	0
21	102980	0
22	102960	0
23	102960	0
24	102980	0
25	102990	0
26	102970	0
27	102950	0
28	102900	0
29	102880	0
...	...	...
8730	101330	0
8731	101290	0
8732	101230	0
8733	101190	0
8734	101090	0
8735	100990	0
8736	100850	0
8737	100750	0
8738	100650	0
8739	100480	0
8740	100350	0
8741	100210	0
8742	100080	0
8743	99940	0
8744	99870	59
8745	99770	224
8746	99670	349
8747	99570	422
8748	99500	438
8749	99430	395
8750	99370	298
8751	99330	151
8752	99270	11

8753	99370	0
8754	99300	0
8755	99410	0
8756	99530	0
8757	99800	0
8758	100310	0
8759	101020	0

	Global Horizontal Radiation	Direct Normal Radiation	\
0	0	0	
1	0	0	
2	0	0	
3	0	0	
4	0	0	
5	0	0	
6	0	0	
7	0	0	
8	8	21	
9	23	0	
10	94	78	
11	106	83	
12	214	502	
13	102	81	
14	38	0	
15	11	0	
16	0	0	
17	0	0	
18	0	0	
19	0	0	
20	0	0	
21	0	0	
22	0	0	
23	0	0	
24	0	0	
25	0	0	
26	0	0	
27	0	0	
28	0	0	
29	0	0	
...	...	...	
8730	0	0	
8731	0	0	
8732	0	0	
8733	0	0	
8734	0	0	
8735	0	0	
8736	0	0	
8737	0	0	

8738	0	0
8739	0	0
8740	0	0
8741	0	0
8742	0	0
8743	0	0
8744	8	21
8745	54	68
8746	72	0
8747	115	0
8748	135	0
8749	57	0
8750	42	0
8751	4	0
8752	0	0
8753	0	0
8754	0	0
8755	0	0
8756	0	0
8757	0	0
8758	0	0
8759	0	0

	Diffuse Horizontal Radiation	Wind Direction	Wind Speed
0	0	292	3.1
1	0	292	3.6
2	0	270	2.8
3	0	270	2.2
4	0	270	3.6
5	0	270	2.8
6	0	270	3.1
7	0	270	2.8
8	7	292	3.6
9	23	270	3.6
10	75	315	3.9
11	81	315	3.1
12	57	315	2.8
13	79	315	0.8
14	38	315	2.8
15	11	0	2.8
16	0	0	0.0
17	0	338	2.2
18	0	0	0.0
19	0	0	0.0
20	0	0	0.0
21	0	0	2.2
22	0	45	3.1
23	0	338	2.2

```
24                          0             292          3.1
25                          0             292          3.1
26                          0             270          2.2
27                          0             292          2.2
28                          0             292          2.8
29                          0             270          2.8
...                       ...             ...          ...
8730                        0              20          1.1
8731                        0              70          3.1
8732                        0              80          4.2
8733                        0              70          4.2
8734                        0              70          3.6
8735                        0             110          4.7
8736                        0              80          5.3
8737                        0              70          5.6
8738                        0              70          4.2
8739                        0              90          4.7
8740                        0             110          5.3
8741                        0             140          6.7
8742                        0             140          7.8
8743                        0             110          6.1
8744                        7             120          5.3
8745                       43             130          7.2
8746                       72             130          6.7
8747                      115             140          6.7
8748                      135             140          7.8
8749                       57             130          6.1
8750                       42              90          6.1
8751                        4              70          3.6
8752                        0             310          4.2
8753                        0             330          2.5
8754                        0             350          1.9
8755                        0               0          0.0
8756                        0               0          0.0
8757                        0               0          0.0
8758                        0               0          0.0
8759                        0             250          2.5

[8760 rows x 12 columns]
```

We now have a Pandas DataFrame object called `data`.
We can access columns using square brackets and the column name.

```
In [10]: T_air = data['Dry Bulb Temp'] # get the dry bulb temp column
         T_air

Out[10]: 0       0.1
         1       0.5
```

```
2       0.8
3       0.8
4       0.8
5       0.7
6       1.7
7       2.2
8       2.8
9       3.3
10      4.4
11      4.4
12      6.1
13      6.1
14      5.6
15      5.0
16      5.0
17      4.4
18      4.4
19      4.4
20      3.3
21      4.4
22      5.0
23      4.4
24      4.4
25      4.4
26      4.4
27      3.9
28      3.9
29      3.9
       ...
8730    0.2
8731    2.0
8732    1.8
8733    1.8
8734    2.1
8735    2.8
8736    2.4
8737    1.9
8738    2.0
8739    2.6
8740    3.7
8741    4.7
8742    4.8
8743    4.6
8744    4.4
8745    4.2
8746    4.4
8747    4.2
8748    3.6
```

```
8749    2.7
8750    2.2
8751    0.8
8752    1.0
8753    0.9
8754    0.7
8755    0.3
8756    0.0
8757   -0.4
8758   -0.4
8759   -0.2
Name: Dry Bulb Temp, dtype: float64
```

We can list the column names to assist with this.

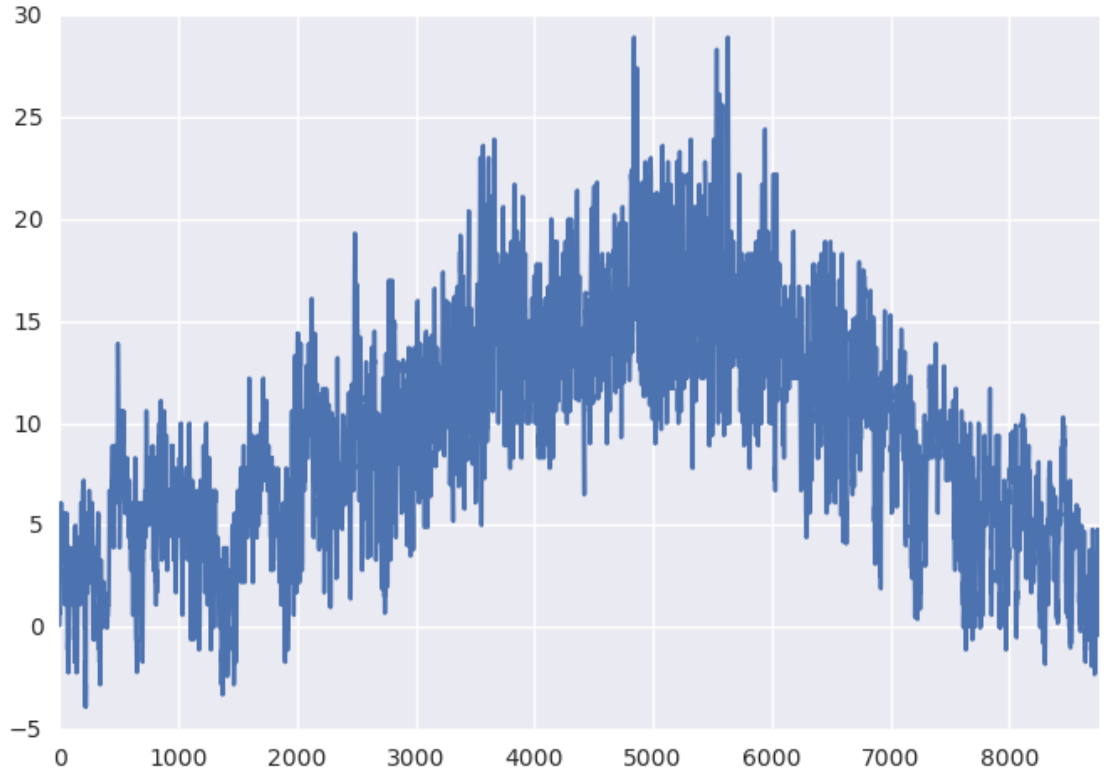
```
In [11]: data.columns
```

```
Out[11]: Index(['Date', 'HH:MM', 'Dry Bulb Temp', 'Dew Point Temp', 'Relative Humic
                'Atmospheric Pressure', 'Extraterrestrial Horizontal Radiation',
                'Global Horizontal Radiation', 'Direct Normal Radiation',
                'Diffuse Horizontal Radiation', 'Wind Direction', 'Wind Speed'],
               dtype='object')
```

We now have a Pandas Series object called T_air, which we can plot by adding `.plot()`

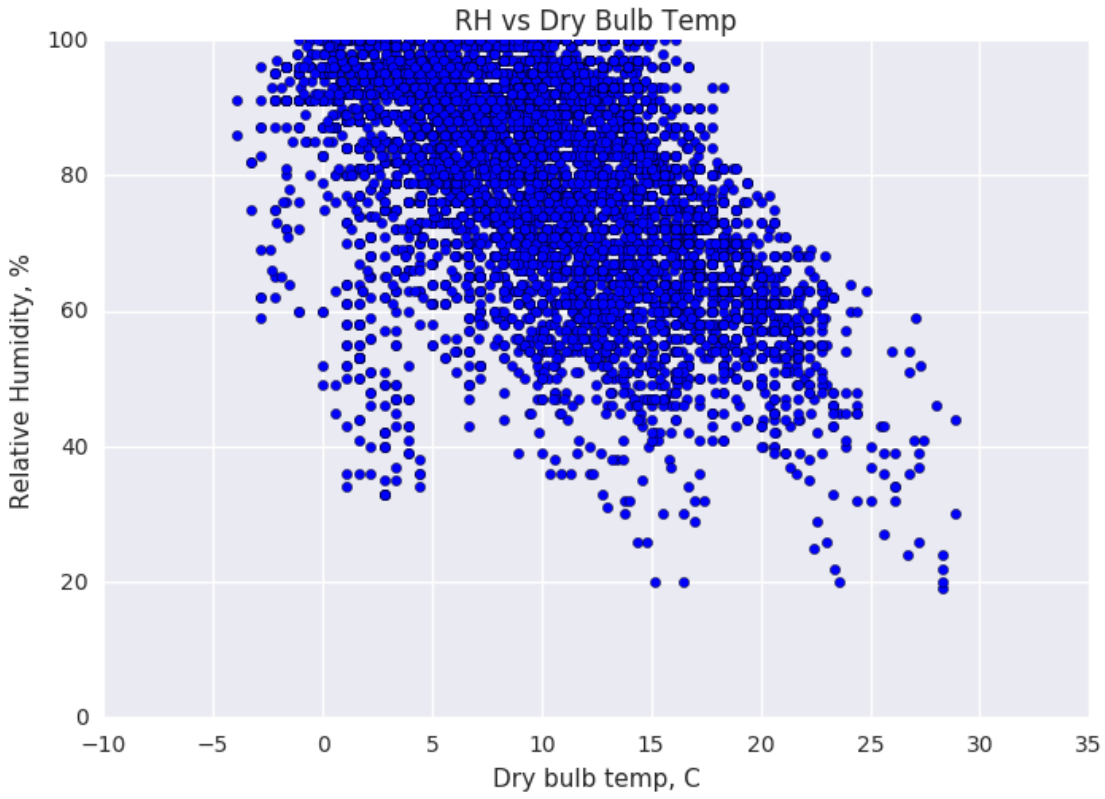
```
In [22]: T_air.plot()
```

```
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb2b507bba8>
```



We can make a scatter plot using MatPlotLib by getting another column and passing both to plt.scatter().

```
In [23]: RH = data['Relative Humidity'] # get the Relative Humidity column

         plt.scatter(T_air, RH) # make a scatter plot
         plt.xlabel('Dry bulb temp, C') # label the x axis
         plt.ylabel('Relative Humidity, %') # label the y axis
         plt.title('RH vs Dry Bulb Temp') # add a title
         plt.ylim((0,100)) # set the limits of the y axis
         plt.grid(True) # show gridlines
```



We can use conditional statements directly on dataframe objects to isolate parts of the data:

```
In [25]: T_air > 5

Out[25]: 0       False
         1       False
         2       False
         3       False
         4       False
```

```
5       False
6       False
7       False
8       False
9       False
10      False
11      False
12       True
13       True
14       True
15      False
16      False
17      False
18      False
19      False
20      False
21      False
22      False
23      False
24      False
25      False
26      False
27      False
28      False
29      False
        ...
8730    False
8731    False
8732    False
8733    False
8734    False
8735    False
8736    False
8737    False
8738    False
8739    False
8740    False
8741    False
8742    False
8743    False
8744    False
8745    False
8746    False
8747    False
8748    False
8749    False
8750    False
8751    False
```

```
8752    False
8753    False
8754    False
8755    False
8756    False
8757    False
8758    False
8759    False
Name: Dry Bulb Temp, dtype: bool
```

This output array of binary values can be easily counted using sum():

```
In [26]: hours_over_25 = sum(T_air > 25)
         print(hours_over_25)
```

```
28
```

We can combine conditionals using `&` for `AND` or `|` for `OR` (remember the brackets)

```
In [28]: comfortable_hours = sum((T_air > 18) & (T_air < 23))
         print(comfortable_hours)
```

```
554
```

Or we can use it as the index for the whole dataframe:

```
In [29]: data[T_air > 25]
```

```
Out[29]:          Date  HH:MM  Dry Bulb Temp  Dew Point Temp  Relative Humidity  \
         4836  21/07/80  13:00           26.0            16.1                 54
         4837  21/07/80  14:00           26.8            15.8                 51
         4838  21/07/80  15:00           28.0            15.3                 46
         4839  21/07/80  16:00           28.9            15.4                 44
         4840  21/07/80  17:00           27.1            18.3                 59
         4841  21/07/80  18:00           27.3            16.5                 52
         4861  22/07/80  14:00           26.8            16.6                 54
         4862  22/07/80  15:00           27.4            13.1                 41
         4863  22/07/80  16:00           27.0            12.8                 41
         4864  22/07/80  17:00           26.8            10.3                 36
         4865  22/07/80  18:00           25.5            12.0                 43
         5531  19/08/66  12:00           25.6             5.0                 27
         5532  19/08/66  13:00           27.2             6.1                 26
         5533  19/08/66  14:00           28.3             5.6                 24
         5534  19/08/66  15:00           28.3             4.4                 22
         5535  19/08/66  16:00           28.3             2.2                 19
         5536  19/08/66  17:00           28.3             3.3                 20
```

5537	19/08/66	18:00	26.7	4.4	24
5557	20/08/66	14:00	25.6	9.4	36
5558	20/08/66	15:00	26.1	8.9	34
5559	20/08/66	16:00	26.1	8.3	32
5560	20/08/66	17:00	26.1	8.9	34
5581	21/08/66	14:00	25.6	10.6	39
5626	23/08/66	11:00	26.1	11.1	39
5628	23/08/66	13:00	27.2	12.2	39
5629	23/08/66	14:00	28.9	9.4	30
5630	23/08/66	15:00	25.6	12.2	43
5631	23/08/66	16:00	27.2	11.1	37

	Atmospheric Pressure	Extraterrestrial Horizontal Radiation	\
4836	101400	1163	
4837	101330	1126	
4838	101230	1036	
4839	101120	899	
4840	101060	725	
4841	100960	525	
4861	101190	1124	
4862	101230	1034	
4863	101230	897	
4864	101230	723	
4865	101290	523	
5531	101810	1063	
5532	101760	1081	
5533	101740	1040	
5534	101720	944	
5535	101640	798	
5536	101620	614	
5537	101610	403	
5557	101890	1036	
5558	101810	939	
5559	101760	793	
5560	101690	608	
5581	101330	1031	
5626	101400	974	
5628	101310	1064	
5629	101230	1022	
5630	101230	924	
5631	101190	777	

	Global Horizontal Radiation	Direct Normal Radiation	\
4836	921	959	
4837	891	957	
4838	813	944	
4839	695	921	
4840	545	881	

4841	373	805
4861	769	670
4862	684	659
4863	571	556
4864	446	532
4865	341	661
5531	853	975
5532	873	983
5533	834	974
5534	749	959
5535	623	932
5536	466	886
5537	286	790
5557	824	966
5558	739	949
5559	612	919
5560	454	869
5581	758	771
5626	711	622
5628	711	769
5629	804	959
5630	723	854
5631	591	819

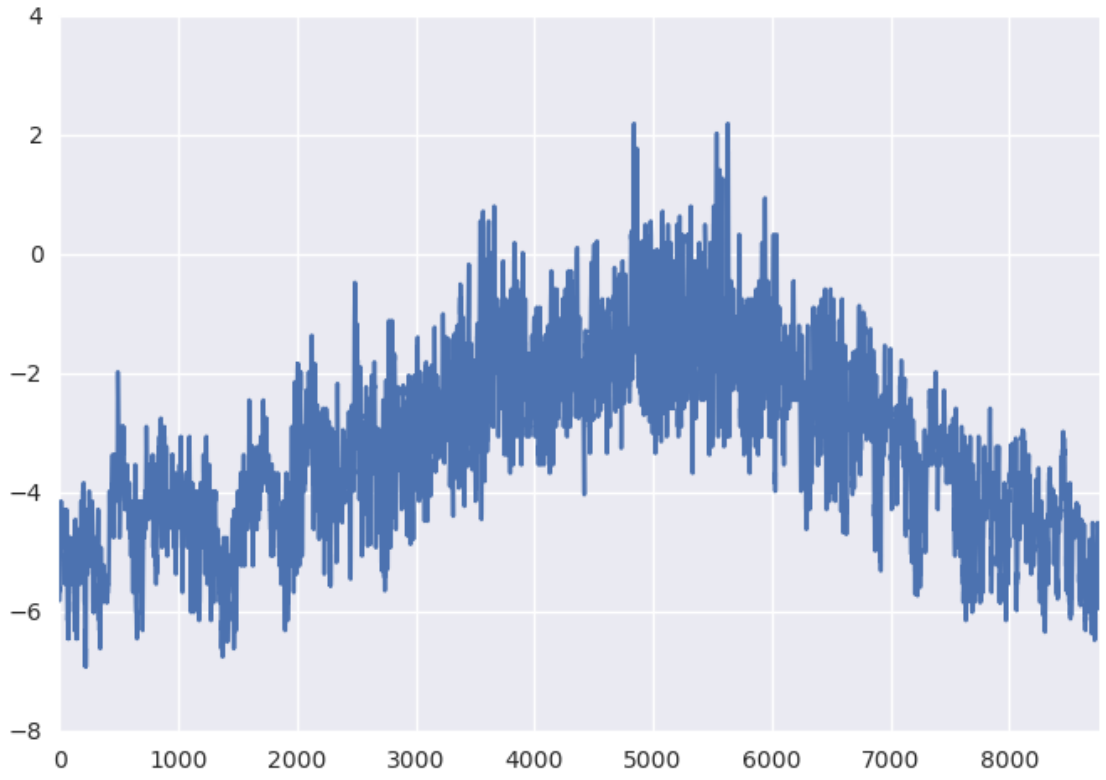
	Diffuse Horizontal Radiation	Wind Direction	Wind Speed
4836	76	20	1.9
4837	75	10	3.1
4838	72	0	2.5
4839	68	280	1.9
4840	61	30	1.1
4841	53	50	1.1
4861	198	90	1.9
4862	167	270	4.2
4863	193	260	2.5
4864	154	260	6.1
4865	80	280	5.6
5531	74	22	5.3
5532	74	0	5.3
5533	73	0	5.3
5534	70	0	6.4
5535	65	22	3.6
5536	58	338	3.6
5537	47	292	2.8
5557	73	90	2.8
5558	70	90	3.1
5559	65	68	2.8
5560	57	68	2.2
5581	161	112	3.1

```
5626                256              248             1.4
5628                 97               90             3.1
5629                 69              112             2.8
5630                131              158             3.6
5631                114              135             2.8
```

We can create some named variables and write equations directly involving dataframe objects:

```
In [30]: T_in = 21 # indoor temp in C
         Q_vent = (T_air - T_in)/3.6 # Q_vent in W
         Q_vent.plot() # plot Q_vent
```

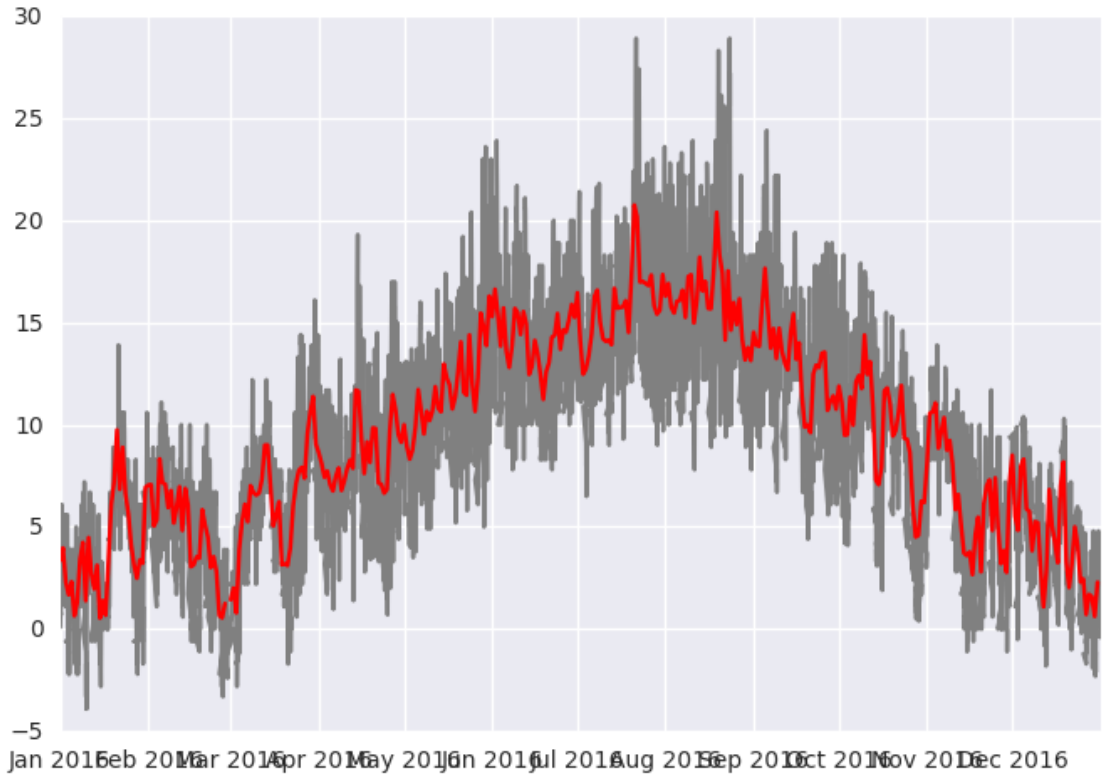
```
Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb2b504ad30>
```



We can use the data and time to make a Python datetime object to use as the index, then resample it at a different frequency:

```
In [34]: data['Datetime'] = pd.to_datetime(data['Date'] + ' ' + data['HH:MM'], form
         data = data.set_index('Datetime') # set as index
         data.index = data.index.map(lambda t: t.replace(year=2016)) # overwrite ye

         day_av = data.resample('D').mean() # resample daily by taking the average

         plt.plot(data['Dry Bulb Temp'],color='gray') # plot hourly data (grey line
         plt.plot(day_av['Dry Bulb Temp'],color='r') # plot daily averages (red lin
```

```
Out[34]: [<matplotlib.lines.Line2D at 0x7fb2b4cf1b00>]
```



We can create a pivot table by hour of day and month of year, applying the np.mean() function:

```
In [72]: cm = seaborn.light_palette("red", as_cmap=True)
         pd.pivot_table(data,index=data.index.month,columns=data.index.hour,
                        values='Dry Bulb Temp',aggfunc=np.mean).style.format("{:.3}

Out[72]: <pandas.formats.style.Styler at 0x7f2e85a358d0>
```

We can calculate the correlation between each column in the dataframe:

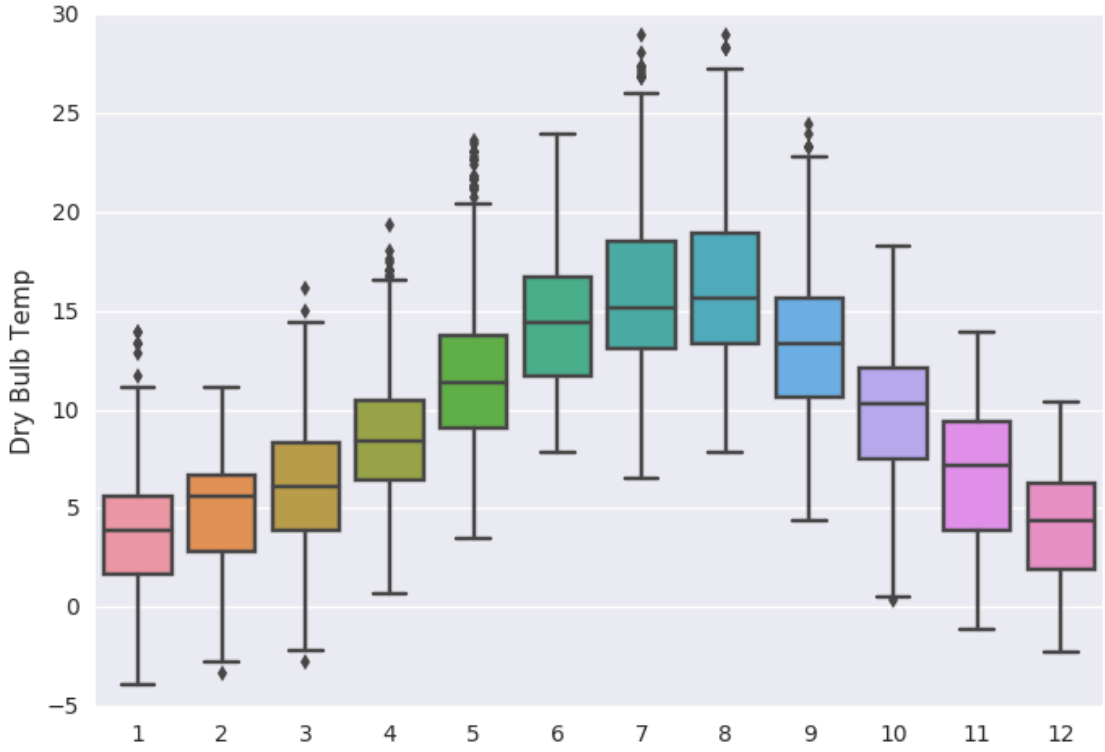
```
In [60]: data.corr(method='pearson').style.format("{:.2}").background_gradient(cmap

Out[60]: <pandas.formats.style.Styler at 0x7f2e85b9b780>
```

0.0.2 Seaborn library

Seaborn (imported at the beginning of this file) contains lots of advanced plotting functions.
See https://seaborn.pydata.org/ for more examples.
We can use it to create monthly box plots in a single line.

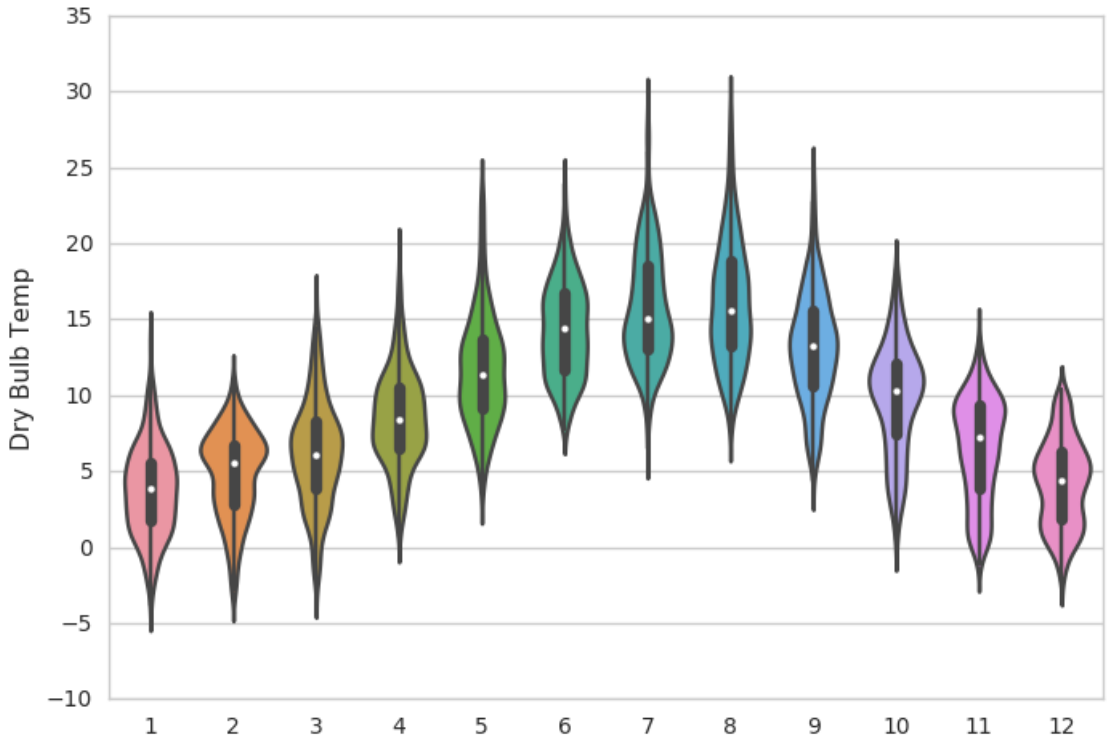
```
In [36]: T_air = data['Dry Bulb Temp'] # get the dry bulb temp
         seaborn.boxplot(T_air.index.month,T_air) # make box plots with months on t
```

```
Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb2b4c5de10>
```



Or use a violin plot to show the full monthly distribution:

```
In [32]: seaborn.violinplot(T_air.index.month,T_air)

Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x7f2e866686a0>
```



0.0.3 Bokeh library

Bokeh (imported at the beginning of this file) contains lots more advanced plotting functions.
See https://bokeh.pydata.org/en/latest/docs/gallery.html for more examples.
We can use it to create interactive plots using slider bars etc.

0.0.4 Moving from Matlab

- Python for Matlab users
- Cheatsheet of Matlab->Python commands (mostly NumPy)
- The equivilent of saving a .mat file is pickle().
- Call Matlab from Python via the API
- Compile Matlab into a Python library

0.0.5 Other sources of info

- Learning Python: From Zero to Hero Starting very simple and builds up to the more advanced.
- Python Data Science Handbook Focus on Numpy and Pandas for data analysis.
- 150 of the Best Machine Learning, NLP, and Python Tutorials General Python tutorials in the third section.

0.0.6 Doing things Pythonically

- The Zen of Python
- Video of examples of coding Pythonically
- Style guide

0.0.7 Other development environment options

- Anaconda is an easy way to install Python. Includes:
- The Conda package manager and a lot of useful packages.
- The Spyder editor
- A lo
- PyCharm is another nice editor to use with Python, including:
- Git integration.
- Viewing code structure.
- Shortcuts to move around, e.g. go to function definition.
- Seeing variable values using the PyCharm debugger: view the contents of a variable, including pandas dataframes.