Practice Project - Dataframe based: Case Study

Case study: air quality data of European monitoring stations (AirBase)

AirBase (The European Air quality dataBase): hourly measurements of all air quality monitoring stations from Europe.

AirBase is the European air quality database maintained by the European Environment Agency (EEA). It contains air quality monitoring data and information submitted by participating countries throughout Europe. The air quality database consists of a multi-annual time series of air quality measurement data and statistics for a number of air pollutants.

Table of Contents

In [ ]:
```
from IPython.display import HTML
HTML('<iframe src=http://www.eea.europa.eu/data-and-maps/data/airbase-the-european-air-quality-database-8#tab-data-by-country width=900 height=350></iframe>')
```

Some of the data files that are available from AirBase were included in the data folder: the hourly **concentrations of nitrogen dioxide (NO2)** for 4 different measurement stations:

- FR04037 (PARIS 13eme): urban background site at Square de Choisy
- FR04012 (Paris, Place Victor Basch): urban traffic site at Rue d'Alesia
- BETR802: urban traffic site in Antwerp, Belgium
- BETN029: rural background site in Houtem, Belgium

See http://www.eea.europa.eu/themes/air/interactive/no2

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

pd.options.display.max_rows = 8
```

Processing a single file

We will start with processing one of the downloaded files (`BETR8010000800100hour.1-1-1990.31-12-2012`). Looking at the data, you will see it does not look like a nice csv file:

In [ ]:
```
with open("data/BETR8010000800100hour.1-1-1990.31-12-2012") as f:
    print(f.readline())
```

```
1990-01-01      -999.000        0       -999.000        0       -999.000        0       -999.000        0       -999.000        0       -999.000        0       -999.000        0       -999.000        0
-999.000        0       -999.000        0       -999.000        0       -999.000        0       -999.000        0       -999.000        0       -999.000        0       -999.000        0       -999.000
0       -999.000        0       -999.000        0       -999.000        0       -999.000        0       -999.000        0       -999.000        0       -999.000        0       -999.000        0
```

So we will need to do some manual processing.

Just reading the tab-delimited data:

In [ ]:
```
data = pd.read_csv("data/BETR8010000800100hour.1-1-1990.31-12-2012", sep='\t')#, header=None)
```

In [ ]:
```
data.head()
```

Out[ ]:

	1990-01-01	-999.000	0	-999.000.1	0.1	-999.000.2	0.2	-999.000.3	0.3	-999.000.4	...	-999.000.19	0.19	-999.000.20	0.20	-999.000.21	0.21	-999.000.22	0.22	-999.000.23	0.23
0	1990-01-02	-999.0	0	-999.0	0	-999.0	0	-999.0	0	-999.0	...	57.0	1	58.0	1	54.0	1	49.0	1	48.0	1
1	1990-01-03	51.0	1	50.0	1	47.0	1	48.0	1	51.0	...	84.0	1	75.0	1	-999.0	0	-999.0	0	-999.0	0
2	1990-01-04	-999.0	0	-999.0	0	-999.0	0	-999.0	0	-999.0	...	69.0	1	65.0	1	64.0	1	60.0	1	59.0	1
3	1990-01-05	51.0	1	51.0	1	48.0	1	50.0	1	51.0	...	-999.0	0	-999.0	0	-999.0	0	-999.0	0	-999.0	0
4	1990-01-06	-999.0	0	-999.0	0	-999.0	0	-999.0	0	-999.0	...	-999.0	0	-999.0	0	-999.0	0	-999.0	0	-999.0	0

5 rows × 49 columns

The above data is clearly not ready to be used! Each row contains the 24 measurements for each hour of the day, and also contains a flag (0/1) indicating the quality of the data. Furthermore, there is no header row with column names.

EXERCISE:

Clean up this dataframe by using more options of $read_csv$ (see its [docstring](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html))

- specify the correct delimiter
- specify that the values of -999 and -9999 should be regarded as NaN
- specify are own column names (for how the column names are made up, see See http://stackoverflow.com/questions/6356041/python-intertwining-two-lists)

In [ ]:
```
# Column names: list consisting of 'date' and then intertwined the hour of the day and 'flag'
hours = ["{:02d}".format(i) for i in range(24)]
column_names = ['date'] + [item for pair in zip(hours, ['flag']*24) for item in pair]
```

In [ ]:
```
data = pd.read_csv("data/BETR8010000800100hour.1-1-1990.31-12-2012",sep='\t', header=None, names=column_names, na_values=[-999, -9999])
```

In [ ]:
```
data.head()
```

Out[ ]:

	1990-01-01	-999.000	0	-999.000.1	0.1	-999.000.2	0.2	-999.000.3	0.3	-999.000.4	...	-999.000.19	0.19	-999.000.20	0.20	-999.000.21	0.21	-999.000.22	0.22	-999.000.23	0.23
0	1990-01-02	-999.0	0	-999.0	0	-999.0	0	-999.0	0	-999.0	...	57.0	1	58.0	1	54.0	1	49.0	1	48.0	1
1	1990-01-03	51.0	1	50.0	1	47.0	1	48.0	1	51.0	...	84.0	1	75.0	1	-999.0	0	-999.0	0	-999.0	0
2	1990-01-04	-999.0	0	-999.0	0	-999.0	0	-999.0	0	-999.0	...	69.0	1	65.0	1	64.0	1	60.0	1	59.0	1
3	1990-01-05	51.0	1	51.0	1	48.0	1	50.0	1	51.0	...	-999.0	0	-999.0	0	-999.0	0	-999.0	0	-999.0	0
4	1990-01-06	-999.0	0	-999.0	0	-999.0	0	-999.0	0	-999.0	...	-999.0	0	-999.0	0	-999.0	0	-999.0	0	-999.0	0

5 rows × 49 columns

For the sake of this tutorial, we will disregard the 'flag' columns (indicating the quality of the data).

E

In [ ]:
```
flag_columns = [col for col in data.columns if 'flag' in col]
# we can now use this list to drop these columns
```

In [ ]:
```
data = data.drop(flag_columns, axis=1)
```

In [ ]:
```
data.head()
```

D

```
al
'fl
co
('
'fl
...
```

Out[ ]:

	1990-01-01	-999.000	0	-999.000.1	0.1	-999.000.2	0.2	-999.000.3	0.3	-999.
0	1990-01-02	-999.0	0	-999.0	0	-999.0	0	-999.0	0	
1	1990-01-03	51.0	1	50.0	1	47.0	1	48.0	1	
2	1990-01-04	-999.0	0	-999.0	0	-999.0	0	-999.0	0	
3	1990-01-05	51.0	1	51.0	1	48.0	1	50.0	1	
4	1990-01-06	-999.0	0	-999.0	0	-999.0	0	-999.0	0	

5 rows × 49 columns