1a_lezione

March 2, 2019

1 Prima lezione

```
In [1]: # RUN THIS CELL: it loads some style files
        from IPython.core.display import HTML
        with open( './style/custom.css', 'r' ) as f: html_style = f.read()
        HTML( html_style )
```

```
Out[1]: <IPython.core.display.HTML object>
```

1.1 Una selezione di linguaggi utili per l'analisi statistica di dati

Considereremo e confronteremo i sequenti 3 linguaggi. Ma ci concentreremo principalmente su `Python`.

`Python` È un linguaggio con un ampio spettro di applicazioni. Va per la maggiore nell'intelligenza artificiale e nelle data sciences, ma ha anche applicazioni web e UI. Ogni abito usa librerie specifiche.

`R` Pensato principalmente per la statistica (ed usato principlamente per questo). Per le più elementari funzioni statistiche non è necessario caricare nessumna libreria.

`Julia` Molto recente (versione 0.0 nel 2009, versione 1.0 nel 2018). Tra qualche anno potrebbe superare per diffusione le altre due alternative (almeno nell'ambito calcolo scientifico).

1.2 Utilizzo di librerie in `Python`

Esistono 3 modi di utilizzare una libreria in `Python`. Per esempio per poter usare la libreria `pandas` (per trattare tabelle di dati) possiamo scrivere

```
In [2]: import pandas
```

Però questo obbliga a usare il prefisso `pandas.` ogni volta che si usa una funzione della libreria. Per esempio per usare la funzione `read_csv` (funzione che legge una tabella si *coma separated values*) dovremmo scrivere `pandas.read_csv`. Questo rischia di rendere poco leggibile il codice.

Un'alternativa è importare singole funzioni della libreria. Per esempio se si vuole usare la solo la funzione `read_csv` potremmo scrivere

```
In [3]: from pandas import read_csv
```

In questo modo possiamo usare `read_csv` senza prefisso. Però questo è scomodo quando dobbiamo importare molte funzioni e quando non siamo ben sicuri di cosa vogliamo/dobbiamo usare.

Il problema si risolve usando una *wildcary*. Il seguente comando importa (senza prefisso) *tutte* le funzioni della libreria.

```
In [4]: from pandas import *
```

Ma questa soluzione è deprecata perché se usiamo due librerie che definiscono la stessa funzione non siamo sicuri quale stiamo usando.

Un compromesso è usare il comando

```
In [5]: import pandas as pd
```

In questo modo il prefisso da usare per le funzioni della libreria è `pd.` (più breve). Tipicamente ogni libreria ha un abbreviazione standard

1.3 Esempio lettura tabella csv

I dati della tabella `Cork_Airport.csv` che si trovav nella directory `BioTeIndu19/dati`. Questo notebook è nella directory `BioTeIndu19/lezioni` quindi il percorso relativo per arrivare alla tablella è `../dati/Cork_Airport.csv`.

Le prime 24 righe della tabella sono descrizione dei dati quindi dobbiamo saltarle.

```
In [32]: df = pd.read_csv("../dati/Cork_Airport.csv", skiprows=24 )
         #df
```

Vogliamo tenere solo le colonne:
`date` data
`maxtp` temperatura massima
`mintp` temperatura minima
`wdsp` velocità del vento

```
In [33]: df = df[ ['date', 'maxtp', 'mintp', 'wdsp'] ]
         df.tail(2) # stampiamo le ultime 2 righe del dataframe
```

```
Out[33]:              date  maxtp  mintp  wdsp
         20848  30-jan-2019    3.4   -2.1   5.0
         20849  31-jan-2019    6.4    1.3  10.5
```

La velocità del vento è in nodi ma noi la vogliamo in km/h.

Dobbiamo moltiplicare per 1.8552 tutti gli elementi della colonna `wdsp`.

Per convenienza, prima creiamo una copia del dataframe `df` che chiameremo `df2`.

Il metodo `map` applica una funzione a tutti gli elementi di un dataframe. Dobbiamo restringerci alla colonna 'wdsp'.

```
In [8]: df1 = df.copy()
        df1['wdsp_kmh'] = df1['wdsp'].map(lambda x: x*1.8552)
        df1.tail(2)
```

```
Out[8]:              date  maxtp  mintp  wdsp  wdsp_kmh
        20848  30-jan-2019    3.4   -2.1   5.0    9.2760
        20849  31-jan-2019    6.4    1.3  10.5   19.4796
```

Un altro modo è approfittare del fatto che pandas interpreta alune semplici operazioni (come addizine e sottrazione) in modo *vettoriale* cioè applicate ad ogni elemento di un vettore (array, serie, ecc.)

```
In [9]: df2 = df.copy()
        df2[ 'wdsp_kmh' ] = df2[ 'wdsp' ] * 1.8552
        df2.tail(2)
```

```
Out[9]:              date  maxtp  mintp  wdsp  wdsp_kmh
        20848  30-jan-2019    3.4   -2.1   5.0    9.2760
        20849  31-jan-2019    6.4    1.3  10.5   19.4796
```

```
In [10]: df3 = pd.DataFrame()
         df3[ 'date' ] = df[ 'date' ]
         df3[ 'max-min' ] = df[ ['maxtp', 'mintp']
                              ].apply(lambda x: round(x['maxtp']-x['mintp'], 1),
                                      axis=1,
                                     )
         df3[ 'max-min' ] = df3[ 'max-min' ] * (9/5) + 32
         df3
```

```
Out[10]:             date  max-min
         0      01-jan-1962    41.54
         1      02-jan-1962    43.70
         2      03-jan-1962    42.98
         3      04-jan-1962    42.80
         4      05-jan-1962    47.66
         5      06-jan-1962    34.88
         6      07-jan-1962    41.36
         7      08-jan-1962    46.04
         8      09-jan-1962    39.92
         9      10-jan-1962    48.20
         10     11-jan-1962    40.64
         11     12-jan-1962    42.62
         12     13-jan-1962    41.18
         13     14-jan-1962    42.08
         14     15-jan-1962    43.70
         15     16-jan-1962    44.06
         16     17-jan-1962    43.16
         17     18-jan-1962    43.52
         18     19-jan-1962    41.72
         19     20-jan-1962    38.30
         20     21-jan-1962    45.86
         21     22-jan-1962    42.98
         22     23-jan-1962    47.48
```

```
23     24-jan-1962    37.40
24     25-jan-1962    42.26
25     26-jan-1962    38.48
26     27-jan-1962    37.40
27     28-jan-1962    37.58
28     29-jan-1962    36.68
29     30-jan-1962    37.40
...            ...      ...
20820  02-jan-2019    34.16
20821  03-jan-2019    34.16
20822  04-jan-2019    34.70
20823  05-jan-2019    35.78
20824  06-jan-2019    38.12
20825  07-jan-2019    35.42
20826  08-jan-2019    37.22
20827  09-jan-2019    36.50
20828  10-jan-2019    36.32
20829  11-jan-2019    36.50
20830  12-jan-2019    36.14
20831  13-jan-2019    37.94
20832  14-jan-2019    39.56
20833  15-jan-2019    40.64
20834  16-jan-2019    43.16
20835  17-jan-2019    39.92
20836  18-jan-2019    42.26
20837  19-jan-2019    39.56
20838  20-jan-2019    41.54
20839  21-jan-2019    44.06
20840  22-jan-2019    41.18
20841  23-jan-2019    41.36
20842  24-jan-2019    35.60
20843  25-jan-2019    41.18
20844  26-jan-2019    42.08
20845  27-jan-2019    38.12
20846  28-jan-2019    38.84
20847  29-jan-2019    42.80
20848  30-jan-2019    41.90
20849  31-jan-2019    41.18

[20850 rows x 2 columns]
```

```
In [15]: df3.to_csv('../dati/escursione_termica.csv')
```

```
In [12]: df3[ 'max-min' ].max()
```

```
Out[12]: 59.0
```

```
In [23]: mask = df3[ 'max-min' ] >= 58
         df3[mask]
```

Out[23]:

	date	max-min
623	16-sep-1963	59.00
9670	23-jun-1988	58.64
11854	16-jun-1994	58.10

In [ ]:
```
!jupyter nbconvert 1a_lezione.ipynb --to html
```