Daily average temperature

January 28, 2019

1 Calculation of the daily average temperature

1.1 Problem description

We have an excel file with climate measurements, which are taken at (about) 5-minutes intervales. They include two columns: air temperature and relative humidity, along with a timestamp. The data spans for more than one month in total.

We would like to use python and pandas to calculate the average temperature for each day, and write it out to a csv file. It should have two columns: Date and Temperature

1.2 Read and check data

```
In [1]: import numpy as np
        import pandas as pd

In [2]: df_climate = pd.read_excel( 'data/climate.xlsx', sheet_name='greenhouse' )

In [3]: df_climate.head()
```

Out[3]:

	Num.	Date and Time	Temp. (řC)	Rel. Humidity (%)
0	4321	2018-04-18 00:00:21	18.53	52.37
1	4322	2018-04-18 00:05:19	18.49	52.44
2	4323	2018-04-18 00:10:19	17.97	55.13
3	4324	2018-04-18 00:15:19	17.58	56.38
4	4325	2018-04-18 00:20:18	17.48	53.81

```
In [4]: df_climate.tail()
```

Out[4]:

	Num.	Date and Time	Temp. (řC)	Rel. Humidity (%)
10164	14684	2018-05-23 23:35:32	19.27	54.42
10165	14685	2018-05-23 23:40:30	19.24	54.79
10166	14686	2018-05-23 23:45:27	19.22	55.16
10167	14687	2018-05-23 23:50:25	19.19	55.52
10168	14688	2018-05-23 23:55:22	19.19	55.52

1.2.1 Define a time index

```
In [5]: df_climate.index = pd.DatetimeIndex( df_climate[ 'Date and Time' ] )

In [6]: df_climate.head()
```

Out[6]:

Date and Time	Num.	Date and Time	Temp. (řC)	Rel. Humidity (%)
2018-04-18 00:00:21	4321	2018-04-18 00:00:21	18.53	52.37
2018-04-18 00:05:19	4322	2018-04-18 00:05:19	18.49	52.44
2018-04-18 00:10:19	4323	2018-04-18 00:10:19	17.97	55.13
2018-04-18 00:15:19	4324	2018-04-18 00:15:19	17.58	56.38
2018-04-18 00:20:18	4325	2018-04-18 00:20:18	17.48	53.81

1.3 Method 1

1.3.1 Know how many days are there in the data, iterate that many times, and select a *start* and *end* points with a 1 day time span between. Each loop iteration, calculate the mean and append it to an empty list.

1.3.2 "n times look for records after a starting point and before an ending point, get those rows and calculate the average"

How many days are in the data?

We get that information from the index:

```
In [7]: df_climate.index.date

Out[7]: array([datetime.date(2018, 4, 18), datetime.date(2018, 4, 18),
               datetime.date(2018, 4, 18), ..., datetime.date(2018, 5, 23),
               datetime.date(2018, 5, 23), datetime.date(2018, 5, 23)],
              dtype=object)

In [8]: df_climate.index.date.shape

Out[8]: (10169,)
```

10169 is the total number of lines, but there are many lines, many rows with the same day (at different hourtimes). Therefore we need only the dates that are ***not repeated***.

For not-repeated-elements, we can use `np.unique`. It returns a list (actually a numpy array, if you want it 'more correctly') with only the elements that are not repeated:

```
In [9]: some_list = [ 1, 2, 3, 3, 3, 4 ]
        np.unique( some_list )

Out[9]: array([1, 2, 3, 4])
```

For the case of the dates, we have 36 different days, a little more than one month:

```
In [10]: unique_dates = np.unique( df_climate.index.date )
```

```
In [11]: print( unique_dates.shape )

(36,)
```

Now, we want to define a point to start iterating. That very start will be the midnight of the first day in the list. We get then the first point in the index with `min()` and we set the hourtime to 00:00:00

```
In [12]: first_start = df_climate.index.min()
         print( first_start )

         first_start = first_start.replace(hour=0, minute=0, second=0)
         print( first_start )

2018-04-18 00:00:21
2018-04-18 00:00:00
```

Now we create a loop, iterate over ranges of 1 day length, select that part of the original data frame, calculate the mean and append it to an empty list:

```
In [13]: end = first_start # Trick to get started in the beginning, in the first loop

         partial_results = []

         how_many_days = unique_dates.shape[0] # It is the first position of a tuple

         for i in range( how_many_days ):

             start = end
             end = start + pd.Timedelta( '1D' )

             condition_start = df_climate.index > start
             condition_end = df_climate.index < end

             partial_results.append( df_climate[ condition_start & condition_end ]['Temp. (řC)

In [14]: # print( partial_results ) # Activate to see the contents of the list, 36 values
```

Or we make it with a new pandas data frame instead of an empty list, using this form:

```
In [15]: pd.DataFrame( data=[ [start.date(), 35] ], columns=['Timestamp','Temperature'] )

Out[15]:     Timestamp  Temperature
         0  2018-05-23           35
```

We will create a number of rows and then add them to an empty data frame. They need to have the same columns, and we will use this form:

`df = pd.concat( [ df, new_row ] )`

That reads "the data frame is what it was before, plus a new row concatenated at the end"

```
In [16]: end = first_start # Trick to get started in the beginning, in the first loop

         df_daily_avg = pd.DataFrame( columns=['Timestamp','Temperature'] )

         how_many_days = unique_dates.shape[0] # It is the first position of a tuple

         for i in range( how_many_days ):

             start = end
             end = start + pd.Timedelta( '1D' )

             condition_start = df_climate.index >= start
             condition_end = df_climate.index < end

             current_average = df_climate[ condition_start & condition_end ]['Temp. (řC)'].mean
             new_row = pd.DataFrame( data=[ [start.date(), current_average] ], columns=['Times

             df_daily_avg = pd.concat( [ df_daily_avg, new_row ] ) # Add the new row to the da
```

```
In [17]: df_daily_avg.head()
```

Out[17]:

	Timestamp	Temperature
0	2018-04-18	21.353715
0	2018-04-19	21.316585
0	2018-04-20	22.221908
0	2018-04-21	19.976771
0	2018-04-22	20.016563

Write to a csv file:

```
In [18]: df_daily_avg.to_csv( 'daily_avg_temp_1.csv' )
```

1.4 Method 2

1.4.1 Make a list of the dates themselves, and make the for loop iterate over them: "For each date, take the part of the table that has that date and calculate the average".

First we can test with an empty list, like before:

```
In [19]: all_dates = np.unique( df_climate.index.date )

         partial_results = []

         for current_date in all_dates:

             condition = df_climate.index.date == current_date

             partial_results.append( df_climate[ condition ]['Temp. (řC)'].mean() )
```

```
In [20]: # print( partial_results ) # Activate to see the contents of the list, 36 values
```

```
In [21]: all_dates = np.unique( df_climate.index.date )

         df_daily_avg = pd.DataFrame( columns=['Timestamp','Temperature'] )

         for current_date in all_dates:

             condition = df_climate.index.date == current_date

             current_average =  df_climate[ condition ]['Temp. (řC)'].mean()
             new_row = pd.DataFrame( data=[ [current_date, current_average] ], columns=['Timest

             df_daily_avg = pd.concat( [ df_daily_avg, new_row ] ) # Add the new row to the da

In [22]: df_daily_avg.head()
```

Out[22]:

	Timestamp	Temperature
0	2018-04-18	21.353715
0	2018-04-19	21.316585
0	2018-04-20	22.221908
0	2018-04-21	19.976771
0	2018-04-22	20.016563

Write to a csv file:

```
In [23]: df_daily_avg.to_csv( 'daily_avg_temp_2.csv' )
```

1.5 Method 3

1.5.1 Resample the dataframe, downsampling it to one-day periods. Use the *mean()* as aggregation function for the resampling.

```
In [24]: df_climate.head()
```

Out[24]:

	Num.	Date and Time	Temp. (řC)	Rel. Humidity (%)
Date and Time				
2018-04-18 00:00:21	4321	2018-04-18 00:00:21	18.53	52.37
2018-04-18 00:05:19	4322	2018-04-18 00:05:19	18.49	52.44
2018-04-18 00:10:19	4323	2018-04-18 00:10:19	17.97	55.13
2018-04-18 00:15:19	4324	2018-04-18 00:15:19	17.58	56.38
2018-04-18 00:20:18	4325	2018-04-18 00:20:18	17.48	53.81

Notice that we use double brackets to select the columns (one in this case), to get a pandas `dataframe`.

Otherwise we would get a `series` (which is also ok, but...)

```
In [25]: df_daily_avg = df_climate.resample( '1d' )[['Temp. (řC)']].mean()

In [26]: df_daily_avg.head()
```

```
Out[26]:                Temp. (řC)
         Date and Time
         2018-04-18      21.353715
         2018-04-19      21.316585
         2018-04-20      22.221908
         2018-04-21      19.976771
         2018-04-22      20.016562
```

Write to a csv file:

```
In [27]: df_daily_avg.to_csv( 'daily_avg_temp_3.csv' )
```
