unswDataImportWeather

Following attempting to import the array data, we now need the weather data for 2018. The weather data is in a completely different format to the Array data (conveniently easier to handle for pandas). Using the same techniques in our Array import we can attempt to bring across the weather data. However, some initial complications revealed some stark issues with the dataset: Dated files could sometimes contain 0KB of data and would break Pandas, and some dated files (dated only for a single day) sometimes contained up to 5 days of weather data.

We deal with both these file issues below, firstly with a try, except statement and simply ignoring any data that can't easily be extracted. The second issue of the multiple days in a single dated file is solved almost for us by pandas and the standardised datetime format in the file (when viewing in excel, it only has minutes:seconds, however when imported throuhg pandas correctly contains yyyy/mm/dd hh/mm/ss/milliseconds) which allows us to assign the data properly to it's date after extraction

In [1]:

```
import pandas as pd
import glob
import os

#file_name = r"C:\Users\Clairvoyant Cabbage\Documents\PythonProject\Thesis\UNSWData\201
8\2018-01-01.csv"
#file_name = r"C:\Users\Clairvoyant Cabbage\Documents\PythonProject\Thesis\84-Site_12-B
P-Solar.csv"

headers = "Timestamp,TZ,01Tpvtg_in (oC),02Tpvtg_out (oC),03Ttankg_in (oC),04Ttankg_out
 (oC),05Ttankg (oC),07Tpvtug_in (oC),08Tpvtug_out (oC),09Ttankug_in (oC),10Ttankug_out
 (oC),11Ttankug (oC),06Flowg,12Flowug,(IR02)T (oC),(SPN1)G_ht (W/m2),(SPN1)G_hd (W/m2),
(SR12)G_tilt (W/m2),(IR02)U/S (W/m2)"
headers = dict(enumerate(headers.split(',')))
headers = {k: headers[k] for k in (0, 15, 16)}
for item in headers:
    print(item, headers[item])
path = r"C:\Users\Clairvoyant Cabbage\Documents\PythonProject\Thesis\UNSWData\2018-Weat
her"
all_files = glob.iglob(os.path.join(path, "*csv"))


li = []

for f in all_files:
    try:
        df = pd.read_csv(f, header=0, usecols=headers).assign(filename = os.path.basena
me(f))
        li.append(df)
        break #remove to attempt all files
    except:
        print("failed: " + f)

df1 = pd.concat(li, axis=0)
df1 = df1.rename(columns = {'Timestamp':'timestamp', '(SPN1)G_ht (W/m2)':'GHI', '(SPN1)
G_hd (W/m2)':'DHI'})

#df1 = pd.concat((pd.read_csv(f, delimiter=";", header=None, skiprows=6, usecols=header
s).assign(filename = os.path.basename(f)) for f in all_files))
#df1 = pd.read_csv(file_name, delimiter=";", header=None, skiprows=6, usecols=headers)

'''df1 = df1.rename(columns = headers)
df1.index = df1['filename'].str.split('.', expand = True)[0] + " " + df1['TimeStamp']
df1 = df1.drop(columns = ['filename'])'''
print("at datetime")
df1.info()
print(df1)
#df1.index = pd.to_datetime(df1.timestamp, errors='coerce')

df1['timestamp'] = pd.to_datetime(df1['timestamp'].map(lambda x: '.'.join(str(x).split(
'.')[:-1])))
df1.index = df1['timestamp']
print(df1)
df1.info()
```

```
0 Timestamp
15 (SPN1)G_ht (W/m2)
16 (SPN1)G_hd (W/m2)
at datetime
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5754 entries, 0 to 5753
Data columns (total 4 columns):
timestamp    5754 non-null object
GHI          5754 non-null float64
DHI          5754 non-null float64
filename     5754 non-null object
dtypes: float64(2), object(2)
memory usage: 179.9+ KB
                   timestamp        GHI        DHI                filena
me
0     2018/01/17 00:00:15.000  36.318780  32.853374  000_20180118T000000.C
SV
1     2018/01/17 00:00:30.000  35.687920  32.845600  000_20180118T000000.C
SV
2     2018/01/17 00:00:45.000  36.332024  32.874088  000_20180118T000000.C
SV
3     2018/01/17 00:01:00.000  38.123100  34.693352  000_20180118T000000.C
SV
4     2018/01/17 00:01:15.001  38.755364  35.947684  000_20180118T000000.C
SV
...                       ...        ...        ...                   
...
5749  2018/01/17 23:59:00.002   2.138962   1.270248  000_20180118T000000.C
SV
5750  2018/01/17 23:59:15.000   1.756393   0.709045  000_20180118T000000.C
SV
5751  2018/01/17 23:59:30.000   2.074356   1.308228  000_20180118T000000.C
SV
5752  2018/01/17 23:59:45.000   1.919243   1.022835  000_20180118T000000.C
SV
5753  2018/01/18 00:00:00.007   1.687950   0.910660  000_20180118T000000.C
SV

[5754 rows x 4 columns]
                              timestamp        GHI        DHI  \
timestamp
2018-01-17 00:00:15 2018-01-17 00:00:15  36.318780  32.853374
2018-01-17 00:00:30 2018-01-17 00:00:30  35.687920  32.845600
2018-01-17 00:00:45 2018-01-17 00:00:45  36.332024  32.874088
2018-01-17 00:01:00 2018-01-17 00:01:00  38.123100  34.693352
2018-01-17 00:01:15 2018-01-17 00:01:15  38.755364  35.947684
...                                 ...        ...        ...
2018-01-17 23:59:00 2018-01-17 23:59:00   2.138962   1.270248
2018-01-17 23:59:15 2018-01-17 23:59:15   1.756393   0.709045
2018-01-17 23:59:30 2018-01-17 23:59:30   2.074356   1.308228
2018-01-17 23:59:45 2018-01-17 23:59:45   1.919243   1.022835
2018-01-18 00:00:00 2018-01-18 00:00:00   1.687950   0.910660

                                    filename
timestamp
2018-01-17 00:00:15  000_20180118T000000.CSV
2018-01-17 00:00:30  000_20180118T000000.CSV
2018-01-17 00:00:45  000_20180118T000000.CSV
2018-01-17 00:01:00  000_20180118T000000.CSV
2018-01-17 00:01:15  000_20180118T000000.CSV
...                                      ...
```

```
2018-01-17 23:59:00  000_20180118T000000.CSV
2018-01-17 23:59:15  000_20180118T000000.CSV
2018-01-17 23:59:30  000_20180118T000000.CSV
2018-01-17 23:59:45  000_20180118T000000.CSV
2018-01-18 00:00:00  000_20180118T000000.CSV

[5754 rows x 4 columns]
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 5754 entries, 2018-01-17 00:00:15 to 2018-01-18 00:00:00
Data columns (total 4 columns):
timestamp    5754 non-null datetime64[ns]
GHI          5754 non-null float64
DHI          5754 non-null float64
filename     5754 non-null object
dtypes: datetime64[ns](1), float64(2), object(1)
memory usage: 224.8+ KB
```

Result

As a result of the weather extraction, we end up with (maybe) the Timestamp, GHI, and DHI which are required further down the pipeline. Pandas has some issue converting the datetime object from the weather file to a proper datetime64 type due to containing millisecond data, so I used a string manipulation workaround to 'round' off the millisecond component. We can see that we properly have a datetime64[ns] aligning in dtype to the Array data we processed earlier. This datetime data however is at 15 second intervals, unlike the 5 minute intervals of the Array data which is an issue we will overcome later.