Converting Radiation's ".dat" files to a NetCDF file

The files found in radiations FTP server (found at ftp://ftp.cmdl.noaa.gov in /g-rad/baseline/) are organized by test sites like ALT (Alert Observatory in Alert, Nunavut, Canada; a BSRN site) and BRW (Barrow Observatory in Barrow, Alaska, United States; a baseline GMD observatory). There are six total baseline observatories and four BSRN sites. Each testing location is a directory holding `.zip` files, one per month. Each `.zip` file, when unzipped, contains a single `.dat` file:

```
                       ALT_RAD
                                                           ALT_RAD2
                        DIRECT        D_GLOBAL          U_GLOBAL          Zenith
Year Mn Dy Hr Mi              DIFFUSE2          D_IR              U_IR
2004  9  1  0  1    1.04   79.40   78.67  303.58   61.06  310.95  85.142
2004  9  1  0  2    0.71   74.36   73.91  303.80   57.82  310.92  85.171
2004  9  1  0  3    0.67   71.80   71.64  304.25   56.84  310.98  85.199
2004  9  1  0  4    0.75   74.35   74.83  304.21   59.68  310.89  85.227
```

This is the first eight lines of the first file in the `ALT` directory (`alt_2004_09.dat`). The first issue is evident right away: the headers are not all on one line. In fact, some headers use two lines. This format is fairly legible to humans, but any program would have a hard time reading this. I decided the first thing to do would be to convert these files to CSV's (as most data libraries would certainly have readers for CSV's). There is an irregular amount of whitespace between columns, but I knew python wouldn't have had an issue with this.

My second issue is exemplified with these exerpts from the `BAO` directory:

```
                      BAO0_RAD
                                                           BAO8_RAD
                        DIRECT        D_GLOBAL          U_GLOBAL          Zenith
Year Mn Dy Hr Mi               DIFFUSE          D_IR              U_IR
1992  1  1  0  3 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00  93.734
1992  1  1  0  6 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00  94.245
1992  1  1  0  9 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00  94.758
1992  1  1  0 12 -999.00 -999.00 -999.00 -999.00 -999.00 -999.00  95.273
```

```
                    BAO_RAD

                     DIRECT        D_GLOBAL          Zenith
Year Mn Dy Hr Mi         DIFFUSE2            D_IR
2016  5  1  0  1    0.32   105.18   105.66  302.53   69.491
2016  5  1  0  2    0.37    97.06    96.79  306.56   69.681
2016  5  1  0  3    0.24    91.87    92.35  307.61   69.872
2016  5  1  0  4    0.00    93.20    94.02  306.54   70.062
```

The files shown are `bao_1992_01.dat` (shown first) and `bao_2016_05.dat` (shown second). The headers change. Specifically, `DIFFUSE` changes to `DIFFUSE2`. Additionally `BAO8_RAD U_GLOBAL` and `U_IR` are removed. The headers swap at the beginning of 2016.

How to convert ".dat" to ".csv"

In there initial state, these `.dat` files are impossible to work with. I chose to use `.csv` as an intermediate file type because nearly every language and framework has a csv reader/writer build in. If you're unfamiliar with `.csv` files, they are simply comma delimted files. Here's an example of the first eight lines of `alt_2004_09` expressed as a `.csv`.

```
Year,Mn,Dy,Hr,Mi,DIRECT,DIFFUSE2,D_GLOBAL,D_IR,U_GLOBAL,U_IR,Zenith
2004,9,1,0,1,1.04,79.40,78.67,303.58,61.06,310.95,85.142
2004,9,1,0,2,0.71,74.36,73.91,303.80,57.82,310.92,85.171
2004,9,1,0,3,0.67,71.80,71.64,304.25,56.84,310.98,85.199
2004,9,1,0,4,0.75,74.35,74.83,304.21,59.68,310.89,85.227
2004,9,1,0,5,0.83,75.57,75.27,304.50,59.84,310.77,85.255
2004,9,1,0,6,0.82,75.82,75.73,304.45,59.87,310.82,85.283
2004,9,1,0,7,0.53,78.45,78.66,304.41,61.67,310.57,85.311
```

I wrote a python module called `dat_parse.py`. In short, it keeps track of the left-to-right positions of each header. This is important because although they are on different lines, the left-to-right order should be maintained. The file `dat_parse.py` is just a module that is imported by `dat_convert.py`. The file `dat_convert.py` can be called directly with a commands like

```
python dat_convert.py YOUR_FILE.dat
python dat_convert.py *.dat
```

to convert from `.dat` to `.csv`. The first converts a single file, the second converts all the `.dat` files (in your working directory). I recomend changing the `out_name` definition. M file structure is almost certainly different from yours.

Converting CSV to NetCDF

A quick google search reveals that there are endless ways to do this, but I've decided to use `xarray` and `pandas` libraries. My workflow was to make a dataframe from each CSV and then use that dataframe to export out to a NetCDF file. I chose this method because of how flexible a dataframe is. For instance, a dataframe could contain all the data from a testing location (even with the changing headers) or even all the readiation data in total. Here is what I did:

In [7]:
```
import pandas as pd
import xarray as xr
from os.path import basename
import os
```

In [8]:
```
input = "alt_2004_09.csv"
out_name = "test.nc"
df1=pd.DataFrame()
df1 = pd.read_csv(input,
    sep=",",
    parse_dates = {"Date":[0,1,2,3,4]},
    date_parser=lambda x:pd.to_datetime(x,format="%Y %m %d %H %M"),
    index_col=['Date'])
df1.loc[:,"TestSite"]="ALT"
xds=xr.Dataset.from_dataframe(df1)
xds.to_netcdf(out_name)
del df1
```

Essentially the file is seperated by commas (due to it's `.csv` nature), the date (columns 0-4) is parsed (and used as an index), and the testing location is added into a new column. This should return a .nc file.

Adding Metadata

The `netCDF4` library is very useful for adding metadata. Here's an example where I add the global attribute `title`:

In [14]:
```
import netCDF4 as nc

foo = nc.Dataset('test.nc','r+')
foo.title = 'NOAA/ESRL/GMD/GRAD Radiation Archive - ALT'
```

To add metadata to a variable is pretty simple as well. Here I add to the `Zenith` variable:

In [15]:
```
foo.variables['Zenith'].long_name = "Solar zenith angle"
```

You can access all the variables info by calling `foo.variables` and you can access any single variable by its name by using `foo.variable['VARIABLE_NAME']`.

The action of actually adding metadata in this way is not difficult, and it's easy to run at scale (adding similar metadata to many different files). I'm just not sure what the metadata actually is.

Questions Going Forward

Ideally we could add metadata at scale; there are a couple thousand files in radiation's baseline folder alone. Can we determine what metadata is exactly the same for all of them? Maybe they only differ depending on the variables that are present, or on the testing site.

In my mind, all group's should have the same kinds of metadata. This was where my idea of a common excel spreadsheet came from. I don't think it would be too hard to parse and add into existing netCDF files.