APS Data Processing

SRJ Likith

1 Steps

1 Read relevant files

First, the files containing the data and information that we need are read into `pandas` `DataFrames` using the `read_excel` and `read_csv` commands as appropriate.

```python
import pandas as pd
legend_df=pd.read_excel('SAXSframevstest_final.xlsx')
meta_df=pd.read_csv('aclarke_jul21_exp_tracking.csv',low_memory=False)
```

2 Cleaning the data

The first cleaning step, and these are aimed mainly at the `metadata` file, is to strip column names of leading and trailing whitespace characters. Next, any columns that consist of all `null`, `None`, or `NaN` values are removed. Lastly, the `Date` column is converted from strings to `datetime` objects. This is all achieved by the `clean_df` function and the helper function `dt_formatter`.

```python
from datetime import datetime as dt

def dt_formatter(dt_str):
    return dt.strptime(dt_str, '%a %b %d %H:%M:%S %Y)

def clean_df(in_df):
    in_df.columns=in_df.columns.str.strip()
    for c in in_df.columns:
        if in_df[c].isnull().all():
            del in_df[c]
    in_df['Date']=in_df['Date'].apply(dt_formatter)
    return in_df

meta_df=clean_df(meta_df)
```

3 Adding calculated column(s)

Based on the `saxs fnum` column, the `wax_fnum` column is added, which is `saxs fnum`/10 rounded to the nearest integer.

```python
def wax_fnum(row):
    return round(int(row['saxs fnum'])/10)

meta_df['wax fnum']=meta_df.apply(wax_fnum,axis=1)
```

4 Processing

In order to achieve all the processing steps, we loop through (`for` loop) the `legend_df` `DataFrame`. The `idx` variable contains the index/row number while the `row` variable takes the values of the rows themselves.

```python
for idx, row in lengend_df.iterrows():
```

Note that the most following lines are all contained within this `for` loop and so, are indented one level (4 spaces/1 tab).

For each row, the first thing to do is to read the `SAXS Frame start` and `SAXS Frame End`, and then apply the `buffer` value, i.e., the limits now become `SAXS Frame start-buffer_frames` and `SAXS Frame end+buffer_frames`. Note that the `buffer_frames=100` line needs to be run just once, and so, should be placed before the start of the `for` loop.

```python
buffer_frames=100 #this line needs to be placed before the for loop starts
  fnum_lims=[int(i) for i in [row['SAXS Frame Start'],row['SAXS Frame End']]]
  fnum_lims=[el+((-1)**(idex+1)*buffer_frames) for idex,el in enumerate(fnum_lims)]
```

Next, the sample name is read from the `SampleName` column and stripped of leading and trailing whitespaces. The `fname` variable is meant for naming the files and directories, and since this shouldn't have any `/` characters, these are replaced by _ characters.

```python
  sample=row['SampleName'].strip()
  fname=sample.replace('/','_')
```

Then, a small subset/subsection of the `meta_df` is chosen that consists of the relevant columns, and only those rows with `saxs fnum` that lie between the limits mentioned earlier. Also, the `MTS load (mm)` is renamed to `Force (N)`. Again, like the previous block of code, these lines are indented one level since they are placed within the `for` loop.

```python
  curr_meta=meta_df[['Date', 'MTS load (mm)', 'Furnace T1 (C)',
                    'saxs fnum', 'wax fnum', 'MTS crosshead (mm)']][
                    (meta_df['saxs fnum']>=fnum_lims[0]) &
                    (meta_df['saxs fnum']<=fnum_lims[1])]
  curr_meta.rename({'MTS load (mm)':'Force (N)'}, axis=1, inplace=True)
```

Next, for each row in `legend_df`, a folder is created (if it doesn't already exist), and the `curr_meta` `DataFrame` is written to that directory. Note that the `file_ops` function is defined *before* the loop. The variable `p` helps point to the directory where we want all the sections and their directories deposited. This folder is created if it doesn't already exist.

```python
from pathlib import Path
p=Path('./sections_dir_final')
p.mkdir(parents=True, exist_ok=True)
def file_ops(sample_name,dest_dir=p)
    target=dest_dir/sample_name
    if not target.exists():
        target.mkdir(parents=True,exist_ok=True)

#the following lines are contained within the for loop, and hence, indented
  file_ops(fname)
  curr_meta.to_csv(p/fname/(fname+'_metadata.csv'),sep=',', index=False)
```

Finally, a small `readme` file is deposited in each of the sub-directories, containing the following information about `buffer_frames`

```python
readme_str='buffer_frames=100 #number of frames added to the beginning and end of each section to act as a buffer for context i.e., each section, instead of consisting of SAXS Frame Start to SAXS Frame End, will consist of (SAXS Frame Start - buffer) to (SAXS Frame End + buffer)' #defined before the loop

#within the loop
  with open(p/fname/'readme.txt','w') as f:
      f.writelines(readme_str)
```

2 Appendix

```python
#!/usr/bin/python3

from pathlib import Path
import pandas as pd
from glob import glob
from datetime import datetime as dt

def dt_formatter(dt_str): #function to format the 'Date' column in meta_df ie dataframe containing the metadata file
    return dt.strptime(dt_str, '%a %b %d %H:%M:%S %Y')

def clean_df(in_df): #cleaning the meta_df. strips column names of leading and trailing whitespaces. deletes any columns where all the values are null, NaN, etc.
    in_df.columns=in_df.columns.str.strip()
    for c in in_df.columns:
        if in_df[c].isnull().all():
            del in_df[c]
    in_df['Date']=in_df['Date'].apply(dt_formatter)
    return in_df

def wax_fnum(row):
    return round(int(row['saxs fnum'])/10)

p=Path('./sections_dir_final')
p.mkdir(parents=True, exist_ok=True)

def file_ops(sample_name,dest_dir=p):
    target=dest_dir/sample_name
    if not target.exists():
        target.mkdir(parents=True, exist_ok=True)

pd.set_option('display.max_columns',None)
legend_df=pd.read_excel('SAXSframesvstest_final.xlsx') #contains SAXS frame start, end numbers and SampleNames
buffer_frames=100 #number of frames added to the beginning and end of each section to act as a buffer for context i.e., each section, instead of consisting of SAXS Frame Start to SAXS Frame End, will consist of (SAXS Frame Start - buffer) to (SAXS Frame End + buffer)
readme_str='buffer_frames=100 #number of frames added to the beginning and end of each section to act as a buffer for context i.e., each section, instead of consisting of SAXS Frame Start to SAXS Frame End, will consist of (SAXS Frame Start - buffer) to (SAXS Frame End + buffer)'

meta_df=pd.read_csv('aclarke_jul21_exp_tracking.csv',low_memory=False)
meta_df=clean_df(meta_df)
meta_df['wax fnum']=meta_df.apply(wax_fnum,axis=1)

for idx, row in legend_df.iterrows():
    fnum_lims=[int(i) for i in [row['SAXS Frame Start'],row['SAXS Frame End']]]
    fnum_lims=[el+((-1)**(idex+1)*buffer_frames) for idex,el in enumerate(fnum_lims)]#subtracting buffer_frames from frame start and adding it to frame end
    sample=row['SampleName'].strip()
    fname=sample.replace('/','_')
    print(sample)
    curr_meta=meta_df[['Date', 'MTS load (mm)', 'Furnace T1 (C)',
                       'saxs fnum', 'wax fnum', 'MTS crosshead (mm)']][(meta_df['saxs fnum']>=fnum_lims[0]) & (meta_df['saxs fnum']<=fnum_lims[1])]
    curr_meta.rename({'MTS load (mm)':'Force (N)'}, axis=1, inplace=True)
    file_ops(fname)
    curr_meta.to_csv(p/fname/(fname+'_metadata.csv'),sep=',',index=False)
    with open(p/fname/'readme.txt','w') as f:
        f.writelines(readme_str)
```

Listing 1: processor_final.py