# NDAWN_Requests_lab04

December 1, 2021

### 0.0.1 Extracting NDAWN Temp Data (Nov 30-Nov 1) : Lab04

- I calulates the average temp for each station based on the 30 day increment
- I extracted teh Min and Max temp for each weather station based on the 30 day increment
- Exported a New csv file based on these calulated values fro each station (a 29 (stations)x 6(columns) shaped file)

[1]:
```
import pandas as pd
import requests
from datetime import date
from io import StringIO

class ndawn_request:

    def __init__(self, startDate='YYYY-MM-DD', endDate='YYYY-MM-DD', ontology =␣
↪None, location = None, save = False):

        self.start = startDate

        self.end = endDate

        # List of ontology terms, and their URL codes to build request URL
        self.ontology = {
            'Air Temperature': ['variable=hdt', 'variable=hdt9'],
            'Relative Humidity': ['variable=hdrh', 'variable=hdrh9'],
            'Soil Temperature': ['variable=hdbst', 'variable=hdtst'],
            'Wind Speed': ['variable=hdws', 'variable=hdmxws',␣
↪'variable=hdws10', 'variable=hdmxws10'],
            'Wind Direction': ['variable=hdwd', 'variable=hdsdwd',␣
↪'&variable=hdwd10', 'variable=hdsdwd10'],
            'Solar Radiation': ['variable=hdsr'],
            'Rainfall': ['variable=hdr'],
            'Air Pressure': ['variable=hdbp'],
            'Dew Point': ['variable=hddp'],
            'Wind Chill': ['variable=hdwc']
        }
        # Concatenate the ontology keys into a list for exception printout later
        ontologiesErrorMessage = '\n'.join(list(self.ontology.keys()))
```

```
        # List of stations, and URL codes to build request URL
        self.stations = {
            'Ada': 78,
            'Becker': 118,
            'Campbell': 87,
            'Clarissa': 124,
            'Eldred': 2,
            'Fox': 93,
            'Greenbush': 70,
            'Hubbard': 119,
            'Humboldt': 4,
            'Kennedy': 82,
            'Little Falls': 120,
            'Mavie': 71,
            'Ottertail': 103,
            'Parkers Prairie': 116,
            'Perham': 114,
            'Perley': 3,
            'Pine Point': 115,
            'Rice': 121,
            'Roseau': 61,
            'Sabin': 60,
            'Staples': 122,
            'Stephen': 5,
            'Ulen': 91,
            'Wadena': 117,
            'Warren': 6,
            'Waukon': 92,
            'Westport': 123,
            'Williams': 95
        }
        # Concatenate station names into a list for exception printout later
        stationsErrorMessage = '\n'.join(list(self.stations.keys()))

        self.save = save

        # This checks the start and end dates supplied to make sure they are␣
↪valid
        # Start by converting dates into iso format
        startDateCheck = date.fromisoformat(startDate)
        endDateCheck = date.fromisoformat(endDate)
        # If start date is after end date, raise exception
        if startDateCheck > endDateCheck:
            raise Exception('End date cannot be before start date')

        # Create empty list to hold URL codes for ontology terms
```

```
        self.activeMeasures = []
        # If user supplies ontology terms
        if ontology is not None:
            for item in ontology:
                # If user-supplied term is not in the dictionary, raise␣
↪exception
                if item not in self.ontology.keys():
                    raise Exception('Ontology term [' + str(item) + '] not␣
↪recognized. Available ontology terms include: ' + '\n' +␣
↪ontologiesErrorMessage)
                # Otherwise, append URL codes for ontology terms into the list␣
↪of measurements to be requested
                else:
                    for code in self.ontology[item]:
                        self.activeMeasures.append(code)
        # If user does not supply ontology terms, add all URL codes in␣
↪dictionary to the list of measurements to be requested
        else:
            for key in self.ontology:
                for code in self.ontology[key]:
                    self.activeMeasures.append(code)

        # Create empty list to hold URL codes for stations
        self.activeStations = []
        # If user supplies station names
        if location is not None:
            for name in location:
                # If user-supplied name is not in the dictionary, raise␣
↪exception
                if name not in self.stations.keys():
                    raise Exception('Station [' + str(name) + '] not recognized.
↪ Available stations include: ' + '\n' + stationsErrorMessage)
                # Otherwise, append URL codes for stations into the list of␣
↪stations to be requested
                else:
                    self.activeStations.append('station=' + str(self.
↪stations[name]))
        # If user does not supply station names, add all station URL codes in␣
↪dictionary to the list of stations to be requested
        else:
            for key in self.stations:
                self.activeStations.append('station=' + str(self.stations[key]))

    def get_data(self):

        # Construct API call for the request
```

```
        baseURL = 'https://ndawn.ndsu.nodak.edu/table.csv?'
        stations = '&'.join(self.activeStations)
        measurements = '&'.join(self.activeMeasures)
        options = '&ttype=hourly&quick_pick=&begin_date=' + self.start +␣
↪'&end_date=' + self.end
        finalURL = str(baseURL + stations + '&' + measurements + options)

        # Request page
        page = requests.get(finalURL)
        # If status code not 200, raise exception
        if page.status_code != 200:
            raise Exception('URL request status not 200. Status code = ' + page.
↪status_code)

        print('Request successful')

        # Convert csv data to string
        content = str(page.content)
        # Remove large, unnecessary header
        trimContent = content[content.find('Station'):len(content)]
        # Replace newline/return with string literal newline
        formatContent = trimContent.replace('\\r\\n', '\n')
        # Convert content to file object
        contentFile = StringIO(formatContent)

        # Read content into pandas dataframe. Second header row contains units
        ndawnData = pd.read_csv(contentFile, header = [0, 1])

        # Concatenate headers to include units
        # Assign column list to object
        columnHeaders = list(ndawnData.columns)
        # List of new headers
        newHeaderList = []
        # Iterate through column names
        for number in range(0, len(columnHeaders)):
            # If no unit, keep header unchanged, pass into new list
            if 'Unnamed' in columnHeaders[number][1]:
                newHeaderList.append(columnHeaders[number][0])
            # If unit exists, concatenate header and unit, pass into new list
            else:
                newHeader = columnHeaders[number][0] + ' (' +␣
↪columnHeaders[number][1] + ') '
                newHeaderList.append(newHeader)
        # Assign new column names
        ndawnData.columns = newHeaderList

        # Create single column for datetime
```

```
        ndawnData['Date'] = pd.to_datetime(ndawnData[['Year', 'Month', 'Day']])

        # Save to csv if save option selected
        if self.save:
            ndawnData.to_csv('ndawnData.csv', index=False)

        return ndawnData
"""
# Example syntax:
exampleRequest = ndawn_request(startDate='2020-06-23', endDate='2020-06-28',␣
 ↪ontology=['Air Pressure', 'Relative Humidity', 'Soil Temperature', 'Wind␣
 ↪Direction', 'Wind Speed'], location=['Mavie', 'Ottertail', 'Perham',␣
 ↪'Perley'])
ndawnDF = exampleRequest.get_data()
"""
```

```
Request successful
```

[6]:
```
exampleRequest = ndawn_request(startDate='2021-10-20', endDate='2021-11-10',␣
 ↪ontology=['Air Temperature'], location=['Ada', 'Becker',␣
 ↪'Campbell','Clarissa',
'Eldred',
'Fox',
'Greenbush',
'Hubbard',
'Humboldt',
 'Kennedy',
'Little Falls',
'Mavie',
'Ottertail',
 'Parkers Prairie',
'Perham',
'Perley',
 'Pine Point',
'Rice',
'Roseau',
 'Sabin',
'Staples',
'Stephen',
'Ulen',
'Wadena',
'Warren',
'Waukon',
'Westport','Williams'])
ndawnDF = exampleRequest.get_data()
```

```
Request successful
```

[7]:
```
ndawnDF
```

[7]:
```
      Station Name  Latitude (deg)   Longitude (deg)   Elevation (ft)   \
0              Ada        47.321100        -96.513900            910.0
1              Ada        47.321100        -96.513900            910.0
2              Ada        47.321100        -96.513900            910.0
3              Ada        47.321100        -96.513900            910.0
4              Ada        47.321100        -96.513900            910.0
…              …                …                …                …
14108     Williams        48.858454        -94.980897           1093.0
14109     Williams        48.858454        -94.980897           1093.0
14110     Williams        48.858454        -94.980897           1093.0
14111     Williams        48.858454        -94.980897           1093.0
14112            '              NaN               NaN              NaN

         Year  Month   Day  Hour (CST)   Avg Air Temp (Degrees F)   \
0      2021.0   10.0  20.0        100.0                     46.166
1      2021.0   10.0  20.0        200.0                     46.141
2      2021.0   10.0  20.0        300.0                     45.019
3      2021.0   10.0  20.0        400.0                     44.901
4      2021.0   10.0  20.0        500.0                     43.957
…       …    …   …          …                         …
14108  2021.0   11.0   9.0       2100.0                     22.950
14109  2021.0   11.0   9.0       2200.0                     20.469
14110  2021.0   11.0   9.0       2300.0                     19.260
14111  2021.0   11.0   9.0       2400.0                     18.736
14112     NaN    NaN   NaN          NaN                        NaN

      Avg Air Temp Flag  Avg Air Temp at 9 m (Degrees F)   \
0                   NaN                              NaN
1                   NaN                              NaN
2                   NaN                              NaN
3                   NaN                              NaN
4                   NaN                              NaN
…                   …                                …
14108               NaN                              NaN
14109               NaN                              NaN
14110               NaN                              NaN
14111               NaN                              NaN
14112               NaN                              NaN

       Avg Air Temp at 9 m Flag       Date
0                           NaN 2021-10-20
1                           NaN 2021-10-20
2                           NaN 2021-10-20
3                           NaN 2021-10-20
4                           NaN 2021-10-20
```

```
…                          …         …
14108                        NaN 2021-11-09
14109                        NaN 2021-11-09
14110                        NaN 2021-11-09
14111                        NaN 2021-11-09
14112                        NaN        NaT

[14113 rows x 13 columns]
```

[8]:
```
type(ndawnDF)
```

[8]:
```
pandas.core.frame.DataFrame
```

[15]:
```
#assessing the amount of data per station...
count = 0
for value in ndawnDF['Station Name']:
    if value == 'Ada':
        count +=1
        #print(count)
    else:
        pass
print(count)
```

```
504
```

[28]:
```
#this is the data I used and estracted for Lab 4

exampleRequest = ndawn_request(startDate='2021-11-01', endDate='2021-11-30',␣
 ↪ontology=['Air Temperature'], location=['Ada', 'Becker',␣
 ↪'Campbell','Clarissa',
'Eldred',
'Fox',
'Greenbush',
'Hubbard',
'Humboldt',
 'Kennedy',
'Little Falls',
'Mavie',
'Ottertail',
 'Parkers Prairie',
'Perham',
'Perley',
 'Pine Point',
'Rice',
'Roseau',
 'Sabin',
'Staples',
```

```
'Stephen',
'Ulen',
'Wadena',
'Warren',
'Waukon',
'Westport','Williams'])
ndawnDF1 = exampleRequest.get_data()

#a brief inspection
ndawnDF1
```

```
Request successful
```

[28]:

| | Station Name | Latitude (deg) | Longitude (deg) | Elevation (ft) | \ |
|---|---|---|---|---|---|
| 0 | Ada | 47.321100 | -96.513900 | 910.0 | |
| 1 | Ada | 47.321100 | -96.513900 | 910.0 | |
| 2 | Ada | 47.321100 | -96.513900 | 910.0 | |
| 3 | Ada | 47.321100 | -96.513900 | 910.0 | |
| 4 | Ada | 47.321100 | -96.513900 | 910.0 | |
| … | … | … | … | … | |
| 19484 | Williams | 48.858454 | -94.980897 | 1093.0 | |
| 19485 | Williams | 48.858454 | -94.980897 | 1093.0 | |
| 19486 | Williams | 48.858454 | -94.980897 | 1093.0 | |
| 19487 | Williams | 48.858454 | -94.980897 | 1093.0 | |
| 19488 | ' | NaN | NaN | NaN | |

| | Year | Month | Day | Hour (CST) | Avg Air Temp (Degrees F) | \ |
|---|---|---|---|---|---|---|
| 0 | 2021.0 | 11.0 | 1.0 | 100.0 | 33.759 | |
| 1 | 2021.0 | 11.0 | 1.0 | 200.0 | 33.544 | |
| 2 | 2021.0 | 11.0 | 1.0 | 300.0 | 33.269 | |
| 3 | 2021.0 | 11.0 | 1.0 | 400.0 | 32.842 | |
| 4 | 2021.0 | 11.0 | 1.0 | 500.0 | 30.852 | |
| … | … | … | … | … | … | |
| 19484 | 2021.0 | 11.0 | 29.0 | 2100.0 | 25.880 | |
| 19485 | 2021.0 | 11.0 | 29.0 | 2200.0 | 27.507 | |
| 19486 | 2021.0 | 11.0 | 29.0 | 2300.0 | 28.744 | |
| 19487 | 2021.0 | 11.0 | 29.0 | 2400.0 | 28.429 | |
| 19488 | NaN | NaN | NaN | NaN | NaN | |

| | Avg Air Temp Flag | Avg Air Temp at 9 m (Degrees F) | \ |
|---|---|---|---|
| 0 | NaN | NaN | |
| 1 | NaN | NaN | |
| 2 | NaN | NaN | |
| 3 | NaN | NaN | |
| 4 | NaN | NaN | |
| … | … | … | |
| 19484 | NaN | NaN | |

```
19485                NaN                                    NaN
19486                NaN                                    NaN
19487                NaN                                    NaN
19488                NaN                                    NaN

       Avg Air Temp at 9 m Flag       Date
0                           NaN 2021-11-01
1                           NaN 2021-11-01
2                           NaN 2021-11-01
3                           NaN 2021-11-01
4                           NaN 2021-11-01
…                           …          …
19484                       NaN 2021-11-29
19485                       NaN 2021-11-29
19486                       NaN 2021-11-29
19487                       NaN 2021-11-29
19488                       NaN        NaT

[19489 rows x 13 columns]
```

[29]:
```
#still learning about the data i've acquired
ndawnDF1.shape
```

[29]:
```
(19489, 13)
```

[9]:
```
ndawnDF1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19489 entries, 0 to 19488
Data columns (total 13 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Station Name                      19489 non-null  object
 1   Latitude (deg)                    19488 non-null  float64
 2   Longitude (deg)                   19488 non-null  float64
 3   Elevation (ft)                    19488 non-null  float64
 4   Year                              19488 non-null  float64
 5   Month                             19488 non-null  float64
 6   Day                               19488 non-null  float64
 7   Hour (CST)                        19488 non-null  float64
 8   Avg Air Temp (Degrees F)          19488 non-null  float64
 9   Avg Air Temp Flag                 1 non-null      object
 10  Avg Air Temp at 9 m (Degrees F)   0 non-null      float64
 11  Avg Air Temp at 9 m Flag          0 non-null      float64
 12  Date                              19488 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(10), object(2)
memory usage: 1.9+ MB
```

[12]:
```
counta = 0
countb = 0
countc = 0
countc2= 0
counte = 0
countf = 0
countg =0
counth =0
for value in ndawnDF1['Station Name']:
    if value == 'Ada':
        counta +=1
        #print(count)
    if value == 'Becker':
        countb +=1
    if value == 'Campbell':
        countc +=1
    if value == 'Clarissa':
        countc2 +=1
    if value =='Eldred':
        counte +=1
    if value =='Fox':
        countf+=1
    if value =='Greenbush':
        countg+=1
    if value =='Hubbard':
        counth+=1
    else:
        pass
print(counta,countb, countc, countc2, counte, countf, counth, countg)

#i think i have 696 observations for each station...
```

```
696 696 696 696 696 696 696 696
```

[25]:
```
#make a new columns that average the hourly temp, identify the min and max temp␣
 ↪by station:
AveTemp30day= ndawnDF1.groupby('Station Name', as_index=False)['Avg Air Temp␣
 ↪(Degrees F) '].mean()
AveTemp30day

MinTemp30day =ndawnDF1.groupby('Station Name', as_index=False)['Avg Air Temp␣
 ↪(Degrees F) '].min()
MinTemp30day

MaxTemp30day =ndawnDF1.groupby('Station Name', as_index=False)['Avg Air Temp␣
 ↪(Degrees F) '].max()
MaxTemp30day
```

```
#source: https://stackoverflow.com/questions/30482071/
 ↪how-to-calculate-mean-values-grouped-on-another-column-in-pandas
```

[25]:

| | Station Name | Avg Air Temp (Degrees F) |
|---|---|---|
| 0 | ' | NaN |
| 1 | Ada | 59.018 |
| 2 | Becker | 66.596 |
| 3 | Campbell | 67.316 |
| 4 | Clarissa | 62.852 |
| 5 | Eldred | 58.226 |
| 6 | Fox | 58.874 |
| 7 | Greenbush | 57.632 |
| 8 | Hubbard | 56.822 |
| 9 | Humboldt | 55.904 |
| 10 | Kennedy | 58.478 |
| 11 | Little Falls | 62.348 |
| 12 | Mavie | 56.840 |
| 13 | Ottertail | 60.746 |
| 14 | Parkers Prairie | 61.916 |
| 15 | Perham | 60.008 |
| 16 | Perley | 59.450 |
| 17 | Pine Point | 56.624 |
| 18 | Rice | 63.338 |
| 19 | Roseau | 56.876 |
| 20 | Sabin | 63.518 |
| 21 | Staples | 58.478 |
| 22 | Stephen | 56.606 |
| 23 | Ulen | 58.802 |
| 24 | Wadena | 59.720 |
| 25 | Warren | 59.072 |
| 26 | Waukon | 57.596 |
| 27 | Westport | 63.320 |
| 28 | Williams | 58.010 |

[33]:
```
#merge my newly acquired columns to the preexisting dataframe (and reanme the␣
 ↪columns)
ndawnDF2 = pd.merge(ndawnDF1, AveTemp30day, on= ['Station Name'])
ndawnDF2 = ndawnDF2.rename(columns={'Avg Air Temp (Degrees F) _y':␣
 ↪'AveTemp30day'})

ndawnDF2 = pd.merge(ndawnDF2, MinTemp30day, on= ['Station Name'])
ndawnDF2 = ndawnDF2.rename(columns={'Avg Air Temp (Degrees F) ':␣
 ↪'MinTemp30day'})

ndawnDF2 = pd.merge(ndawnDF2, MaxTemp30day, on= ['Station Name'])
```

```
ndawnDF2 = ndawnDF2.rename(columns={'Avg Air Temp (Degrees F) ':␣
↪'MaxTemp30day'})

#visual inspection of my new columns
ndawnDF2.head()
```

[33]:

| | Station Name | Latitude (deg) | Longitude (deg) | Elevation (ft) | Year | Month | Day | Hour (CST) | Avg Air Temp (Degrees F) _x | Avg Air Temp Flag | Avg Air Temp at 9 m (Degrees F) | Avg Air Temp at 9 m Flag | Date | AveTemp30day | MinTemp30day | MaxTemp30day |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Ada | 47.3211 | -96.5139 | 910.0 | 2021.0 | 11.0 | 1.0 | 100.0 | 33.759 | NaN | NaN | NaN | 2021-11-01 | 30.729091 | 1.292 | 59.018 |
| 1 | Ada | 47.3211 | -96.5139 | 910.0 | 2021.0 | 11.0 | 1.0 | 200.0 | 33.544 | NaN | NaN | NaN | 2021-11-01 | 30.729091 | 1.292 | 59.018 |
| 2 | Ada | 47.3211 | -96.5139 | 910.0 | 2021.0 | 11.0 | 1.0 | 300.0 | 33.269 | NaN | NaN | NaN | 2021-11-01 | 30.729091 | 1.292 | 59.018 |
| 3 | Ada | 47.3211 | -96.5139 | 910.0 | 2021.0 | 11.0 | 1.0 | 400.0 | 32.842 | NaN | NaN | NaN | 2021-11-01 | 30.729091 | 1.292 | 59.018 |
| 4 | Ada | 47.3211 | -96.5139 | 910.0 | 2021.0 | 11.0 | 1.0 | 500.0 | 30.852 | NaN | NaN | NaN | 2021-11-01 | 30.729091 | 1.292 | 59.018 |

[35]:
```
#export newly extracted NDAWN data
ndawnDF2.to_csv('NDAWN_nov1_30_a.csv') #exports the original sized data with␣
↪19489 rows...(less than ideal)

#source: https://towardsdatascience.com/
↪how-to-export-pandas-dataframe-to-csv-2038e43d9c03
```

[75]:
```
# i only need observation per station ... 28 rows... Let's see if i can shrink␣
↪my dataframe to the essentials before loading it in ArcPro

#desired columns
names = ['Station Name', 'Latitude (deg) ', 'Longitude (deg) ']
sub1 = ndawnDF1[names]
```

```
sub1.head()

#drop duplcated rows from the original dataframe
sub1 = sub1.drop_duplicates(subset='Station Name', keep="first")
sub1

#merge preexisting dataframe to my newly acquired columns resulting in smaller␣
 ↪dataset  (and rename the columns)
ndawnDF3 = pd.merge(AveTemp30day, sub1, how="outer", on= ['Station Name'])
ndawnDF3 = ndawnDF3.rename(columns={'Avg Air Temp (Degrees F) _y':␣
 ↪'AveTemp30day'})

ndawnDF3 = pd.merge(MinTemp30day,ndawnDF3 , on= ['Station Name'])
ndawnDF3 = ndawnDF3.rename(columns={'Avg Air Temp (Degrees F) _x':␣
 ↪'MinTemp30day'})

ndawnDF3 = pd.merge( MaxTemp30day, ndawnDF3, on= ['Station Name'])
ndawnDF3 = ndawnDF3.rename(columns={'Avg Air Temp (Degrees F) ':␣
 ↪'MaxTemp30day'})

# #visual inspection of my new Dataframe
ndawnDF3.head()
```

[75]:

| | Station Name | MaxTemp30day | MinTemp30day | Avg Air Temp (Degrees F) _y | Latitude (deg) | Longitude (deg) |
|---|---|---|---|---|---|---|
| 0 | ' | NaN | NaN | NaN | NaN | NaN |
| 1 | Ada | 59.018 | 1.292 | 30.729091 | 47.321100 | -96.513900 |
| 2 | Becker | 66.596 | 9.464 | 34.248182 | 45.344300 | -93.850000 |
| 3 | Campbell | 67.316 | 3.614 | 31.862050 | 46.064932 | -96.370165 |
| 4 | Clarissa | 62.852 | 5.252 | 31.377217 | 46.111560 | -94.905800 |

[76]:
```
#inspecting the shape...
ndawnDF3.shape
```

[76]: (29, 6)

[77]:
```
for item in ndawnDF3:
    print(item)
```

```
Station Name
MaxTemp30day
```

```
MinTemp30day
Avg Air Temp (Degrees F) _y
Latitude (deg)
Longitude (deg)
```

[78]:
```
#export newly extracted NDAWN data
ndawnDF3.to_csv('Ave_NDAWN_temps.csv')
```