Reshaping Data

About the data

In this notebook, we will using daily temperature data from the National Centers for Environmental Information (NCEI) API. We will use the Global Historical Climatology Network - Daily (GHCND) data set; see the documentation here.

This data was collected for New York City for October 2018, using the Boonton 1 station (GHCNDUSC00280907). It contains:

the daily minimum temperature (TMIN) the daily maximum temperature (TMAX) the daily temperature at time of observation (TOBS)

Note: The NCEI is part of the National Oceanic and Atmospheric Administration (NOAA) and, as you can see from the URL for the API, this resource was created when the NCEI was called the NCDC. Should the URL for this resource change in the future, you can search for the NCEI weather API to find the updated one.

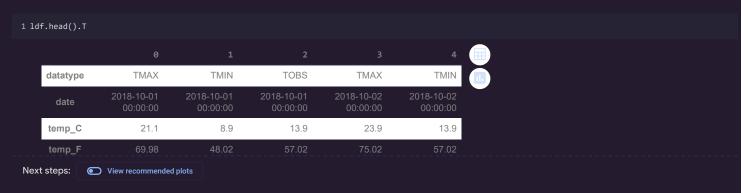
Setup

We need to import pandas and read in the long-format data to get started:

```
1 import pandas as p
3 ld = '/content/long_data.csv'
 4 wd = '/content/wide_data.csv
6 ldf = p.read_csv(
7 ld, usecols=['date', 'datatype', 'value']
      columns={
           'value' : 'temp_C'
12 ).assign(
      date = lambda x: p.to_datetime(x.date),
temp_F=lambda x: (x.temp_C * 9/5)+32
16 ldf.head()
     0
            TMAX 2018-10-01
                                    21.1
                                           69.98
             TOBS 2018-10-01
                                           57.02
      4
             TMIN 2018-10-02
                                           57.02
                                    13.9
```

Transposing

Transposing swaps the rows and the columns. We use the T attribute to do so:



Pivoting

Going from long to wide format.

pivot()

We can restructure our data by picking a column to go in the index (index), a column whose unique values will become column names (columns), and the values to place in those columns (values). The pivot() method can be used when we don't need to perform any aggregation in addition to our restructuring (when our index is unique); if this is not the case, we need the pivot_table() method which we will cover in future modules.

```
1 pvtd_df = ldf.pivot(
2    index='date',columns='datatype',values='temp_C'
3 )
4 pvtd_df.head()
```

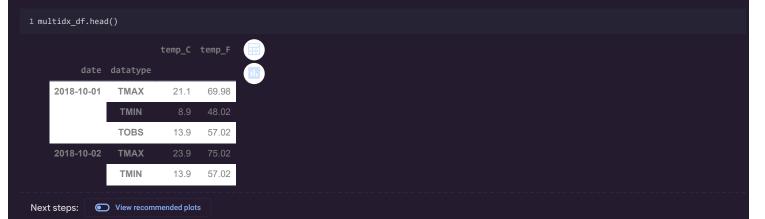
```
\square
     2018-10-01 21.1 8.9 13.9
     2018-10-02
     2018-10-03 25.0 15.6 16.1
     2018-10-05 23.3 11.7 18.9
 Note there is also the pd.pivot() function which yields equivalent results:
1 # dili man sha nagwowork ser :( \downarrow\downarrow\downarrow
3 # index=long_df.date, columns=long_df.datatype, values=long_df.temp_C
6 p.pivot(
     data=ldf, index='date', columns='datatype', values='temp_C'
8 ).head()
     2018-10-01 21.1 8.9 13.9
     2018-10-02
     2018-10-03 25.0 15.6 16.1
     2018-10-05 23.3 11.7
                             18.9
Now that the data is pivoted, we have wide-format data that we can grab summary statistics with:
1 pvtd_df.describe()
     count 31.000000 31.000000 31.000000
                5.714962 6.513252 6.596550
        std
       25%
                12 750000
                           2.500000
                                     5.550000
       50%
       75%
               21.950000 13.600000 16.100000
We can also provide multiple values to pivot on, which will result in a hierarchical index:
1 pvtd df = ldf.pivot(
     index='date',columns='datatype',values=['temp_C', 'temp_F']
4 pvtd_df.head()
                temp C
                                  temp F
     2018-10-01 21.1 8.9 13.9 69.98 48.02 57.02
     2018-10-03 25.0 15.6 16.1 77.00 60.08 60.98
     2018-10-05 23.3 11.7 18.9 73.94 53.06 66.02
With the hierarchical index, if we want to select TMIN in Fahrenheit, we will first need to select 'temp_F' and then 'TMIN':
1 pvtd_df['temp_F']['TMIN'].head()
    date
    2018-10-01
                   48.02
    2018-10-02
2018-10-03
                  57.02
60.08
    2018-10-04
2018-10-05
                  53.06
53.06
     Name: TMIN, dtype: float64
```

v unstack()

We have been working with a single index throughout this chapter; however, we can create an index from any number of columns with set_index(). This gives us a Multilndex where the outermost level corresponds to the first element in the list provided to set_index():

```
1 multidx_df = ldf.set_index(['date','datatype'])
2 multidx_df.index
     MultiIndex([('2018-10-01',
                                         'TMIN'),
'TOBS'),
                       '2018-10-01',
'2018-10-01',
                       '2018-10-02'
'2018-10-02'
                                         'TMAX'),
                                         'TMIN'
                        2018-10-02
                                         'TOBS'
                        2018-10-03
                                          'TMAX
                        2018-10-03
                       2018-10-03
2018-10-04
                                          'TOBS'
                       '2018-10-04'
'2018-10-04'
                                         'TMIN'
                                         'TOBS'
                       '2018-10-05'
'2018-10-05'
                                         'TMAX'
                                         'TMIN
                        2018-10-05
                                         'TOBS'
                                         'TMAX'
                        2018-10-06
                        2018-10-06
                       2018-10-06
2018-10-07
                                          'TOBS
                       '2018-10-07'
'2018-10-07'
                                         'TMIN'
                                         'TOBS
                        2018-10-08'
                                         'TMAX'
                                         'TMIN')
                        2018-10-08'
                                         'TOBS'
                        2018-10-08'
                        2018-10-09'
                        2018-10-09
                       '2018-10-09'
'2018-10-10'
                                          'TOBS'
                       '2018-10-10'
'2018-10-10'
                                         'TMIN'
                                         'TOBS
                        2018-10-11
                                         'TMAX'
                                         'TMIN
                        2018-10-11
                        2018-10-11
                                         'TMAX'
                        2018-10-12
                                         'TOBS')
                        2018-10-12
                        2018-10-13
                       2018-10-13
2018-10-13
                                         'TMIN'
                                         'TOBS
                       '2018-10-14'
'2018-10-14'
                                         'TMAX'
                                         'TMIN'
                        2018-10-14
                       2018-10-15
2018-10-15
                                          'TMAX
                       2018-10-15
2018-10-16
                                          'TOBS'
                       '2018-10-16'
'2018-10-16'
                                         'TMIN'
                                         'TOBS'
                        2018-10-17
                                         'TMAX'
                                         'TMIN'
                        2018-10-17'
                        2018-10-17
                                         'TMAX
                        2018-10-18
                        2018-10-18',
                       '2018-10-18',
'2018-10-19',
                                         'TOBS'),
                        2018-10-19',
                                         'TMIN'),
                                         'TOBS
                        2018-10-19',
                                         'TMAX'),
                      ('2018-10-20',
```

Notice there are now 2 index sections of the dataframe:



With the Multilndex, we can no longer use pivot(). We must now use unstack(), which by default moves the innermost index onto the columns:

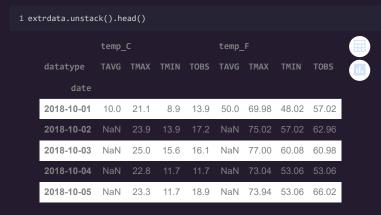
```
1 ustckd_df = multidx_df.unstack()
2 ustckd_df.head()
```



The unstack() method also provides the fill_value parameter, which let's us fill-in any NaN values that might arise from this restructuring of the data. Consider the case that we have data for the average temperature on October 1, 2018, but no other date:

```
1 extrdata = ldf.append(
[{'datatype': 'TAVG','date':'2018-10-01','temp_C': 10, 'temp_F':50}]
3 ).set_index(['date','datatype']).sort_index()
5 extrdata.head(8)
     extrdata = ldf.append(
<ipython-input-21-ff45ca88741b>:3: FutureWarning: Inferring datetime64[ns] from data cont
) set index(['date','datatype']).sort_index()
      2018-10-01
                                         10.0
                        TAVG
                                                  50.00
                          TMIN
                                          8.9
                                                  48.02
                         TOBS
      2018-10-02
                         TMAX
                                         23.9
                                                   75.02
                         TOBS
                                         17.2
                                                   62.96
```

If we use unstack() in this case, we will have NaN for the TAVG columns every day but October 1, 2018:

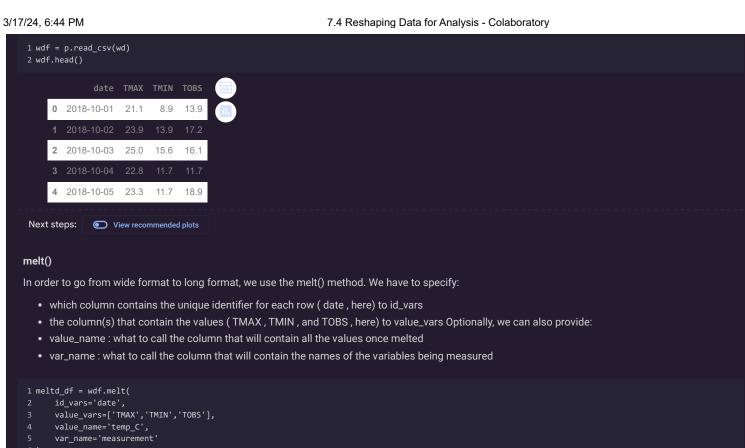


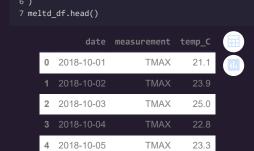
To address this, we can pass in an appropriate fill_value. However, we are restricted to passing in a value for this, not a strategy (like we saw with fillna()), so while -40 is definitely not be the best value, we can use it to illustrate how this works, since this is the temperature at which Fahrenheit and Celsius are equal:

1 extrdata.unstack(fill_value=-40).head() temp F 2018-10-01 13.9 50.0 69.98 48.02 57.02 10.0 21.1 8.9 2018-10-03 -40.0 25.0 15.6 16.1 -40.0 77.00 60.08 60.98 2018-10-04 2018-10-05 -40.0 73.94 11.7

Melting

Going from wide to long format.





Next steps: View recommended plots

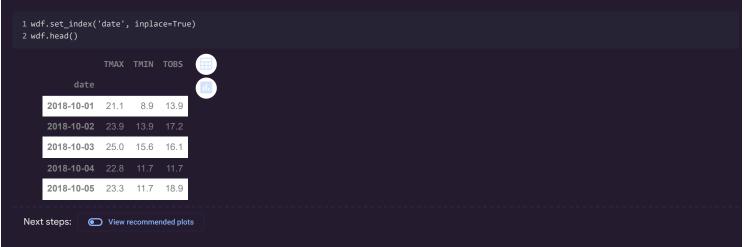
wdf, id_vars='date',

Just as we also had pd.pivot() there is a pd.melt():

```
value_vars=['TMAX','TMIN','TOBS'],
value_name='temp_C',
      var_name='measurement'
7 ).head()
     0 2018-10-01
                             TMAX
                                        21.1
     2 2018-10-03
                             TMAX
                                         25.0
     4 2018-10-05
                              TMAX
                                         23.3
```

stack()

Another option is stack() which will pivot the columns of the dataframe into the innermost level of a Multilndex . To illustrate this, let's set our index to be the date column:



By running stack() now, we will create a second level in our index which will contain the column names of our dataframe (TMAX , TMIN , TOBS). This will leave us with a Series containing the values:

```
1 stckd_seris = wdf.stack()
2 stckd_seris.head()
     date
      2018-10-01
                    TMAX
                              8.9
13.9
                    TMIN
                              23.9
13.9
     2018-10-02
                    TMAX
                    TMIN
     dtype: float64
We can use the to_frame() method on our Series object to turn it into a DataFrame . Since the series doesn't have a name at the moment, we
will pass in the name as an argument:
1 stckd_df = stckd_seris.to_frame('values')
2 stckd_df.head()
      2018-10-01 TMAX
                               21.1
                    TMIN
                    TOBS
                                13.9
      2018-10-02 TMAX
                    TMIN
                                13.9
Next steps:

    View recommended plots

Once again, we have a MultiIndex:
1 stckd_df.index
                    ('2018-10-10',
('2018-10-10',
                                      'TMAX'),
'TMIN'),
'TOBS'),
                       2018-10-10',
                      '2018-10-11'
'2018-10-11'
                                       'TMAX')
                                       'TMIN'
                      2018-10-11
2018-10-12
                                       'TOBS'
                                       'TMAX'
                       2018-10-12
                                       'TMIN'
                      2018-10-12
2018-10-13
                                       'TOBS
                                       'TMAX'
                                       'TMIN')
                       2018-10-13
                       2018-10-13
                       2018-10-14
                                       'TMAX')
                       2018-10-14
                                       'TMIN'
                       2018-10-14
                                       'TOBS'
                       2018-10-15
                       2018-10-15
                                       'TMIN'
                                       'TOBS
                       2018-10-15
                       2018-10-16
                                       'TMAX'
                      '2018-10-16'
'2018-10-16'
                                       'TMIN')
                      2018-10-17
2018-10-17
                                       'TMAX'
                                       'TMIN
                       2018-10-17
                                       'TOBS'
                                       'TMAX'
                       2018-10-18'
                       2018-10-18'
                                       'TMIN'
                                       'TOBS
                       2018-10-18'
                      '2018-10-19'
'2018-10-19'
                                       'TMIN'
                      '2018-10-20'
'2018-10-20'
                                       'TMAX'
                                       'TMIN'
                       2018-10-20'
                                       'TOBS'
                       2018-10-21
                                        'TMAX
                       2018-10-21
                                       'TMIN'
                       2018-10-21
                                       'TOBS'
                       2018-10-22
                                       'TMAX'
                       2018-10-22
                                       'TMIN')
                                       'TOBS'
                       2018-10-22
                      2018-10-23
2018-10-23
                                       'TMAX'
                                       'TMIN
                      2018-10-23
2018-10-24
                                       'TOBS'
                                       'TMAX
                       2018-10-24
                                       'TMIN'
                      '2018-10-24'
'2018-10-25'
                                        TOBS
                                       'TMAX'
                       2018-10-25
                                       'TMTN')
                       2018-10-25
                       2018-10-26'
                                       'TMAX')
                       2018-10-26',
                                      'TMIN')
                                       'TOBS'
                       2018-10-26
                       2018-10-27
                       2018-10-27
                                       'TMIN'
                       2018-10-27
                                        TOBS
                       2018-10-28'
                                       'TMAX'
                      '2018-10-28'
'2018-10-28'
                                       'TMIN'),
                                       'TMAX'),
                      2018-10-29
Unfortunately, we don't have a name for the datatype level:
1 stckd_df.index.names
     FrozenList(['date', None])
```

```
We can use rename() to address this though:
1 stckd_df.index.rename(['date','datatype'],inplace=True)
2 stckd_df.index.names
     FrozenList(['date', 'datatype'])
```