Pandas_TimeSeries exercise

May 22, 2019

In [ ]:
```
import pandas as pd
import numpy as np
import datetime

%matplotlib inline
```

In [ ]:
```
# datetime - date, time, datetime

date = datetime.date(year=2013, month=6, day=7)
time = datetime.time(hour=12, minute=30, second=19, microsecond=463198)
dt = datetime.datetime(year=2013, month=6, day=7,
                       hour=12, minute=30, second=19, microsecond=463198)

print("date is ", date)
print("time is", time)
print("datetime is", dt)
```

In [ ]:
```
# timedelta -

td = datetime.timedelta(weeks=2, days=5, hours=10, minutes=20,
                        seconds=6.73, milliseconds=99, microseconds=8)
print(td)
```

In [ ]:
```
#

print('new date is', date + td)
print('new datetime is', dt + td)
```

In [ ]:
```
time + td
```

In [ ]:
```
# Pandas - Timestamp

pd.Timestamp(year=2012, month=12, day=21, hour=5, minute=10, second=8, microsecond=99)
```

In [ ]:
```
# Timestamp -

pd.Timestamp('2016/1/10')
```

In [ ]:
```
pd.Timestamp('2014-5/10')
```

```
In [ ]: pd.Timestamp('Jan 3, 2019 20:45.56')

In [ ]: pd.Timestamp('2016-01-05T05:34:43.123456789')

In [ ]: # Timestamp - 1970.01.01

        pd.Timestamp(500)

In [ ]: pd.Timestamp(5000, unit='D')

In [ ]: # Pandas - Timestamp, to_datetime

        pd.to_datetime('2015-5-13')

In [ ]: pd.to_datetime('2015-13-5', dayfirst=True)

In [ ]: pd.to_datetime(100, unit='D', origin='2013-1-1')

In [ ]: s = pd.Series([10, 100, 1000, 10000])
        pd.to_datetime(s, unit='D')

In [ ]: s = pd.Series(['12-5-2015', '14-1-2013', '20/12/2017', '40/23/2017'])
        pd.to_datetime(s, dayfirst=True, errors='coerce')

In [ ]: s = pd.Series(['12-5-2015', '14-1-2013', '20/12/2017', '40/23/2017'])
        pd.to_datetime(s, dayfirst=True, errors='raise')

In [ ]: s = pd.Series(['12-5-2015', '14-1-2013', '20/12/2017', '40/23/2017'])
        pd.to_datetime(s, dayfirst=True, errors='ignore')

In [ ]: pd.to_datetime(['Aug 3 1999 3:45:56', '10/31/2017'])

In [ ]: # pandas - Timedelta, to_timedelta

In [ ]: pd.Timedelta('12 days 5 hours 3 minutes 123456789 nanoseconds')

In [ ]: pd.Timedelta(days=5, minutes=7.34)

In [ ]: pd.Timedelta(100, unit='W')

In [ ]: pd.to_timedelta('5 dayz', errors='ignore')

In [ ]: pd.to_timedelta('67:15:45.454')

In [ ]: s = pd.Series([10, 100])
        pd.to_timedelta(s, unit='s')

In [ ]: time_strings = ['2 days 24 minutes 89.67 seconds', '00:45:23.6']
        pd.to_timedelta(time_strings)

In [ ]: # Timedelta
```

```
In [ ]: pd.Timedelta('12 days 5 hours 3 minutes') * 2

In [ ]: pd.Timestamp('1/1/2017') + pd.Timedelta('12 days 5 hours 3 minutes') * 2

In [ ]: td1 = pd.to_timedelta([10, 100], unit='s')
        td2 = pd.to_timedelta(['3 hours', '4 hours'])
        td1 + td2

In [ ]: pd.Timedelta('12 days') / pd.Timedelta('3 days')

In [ ]: ts = pd.Timestamp('2016-10-1 4:23:23.9')

In [ ]: ts.ceil('h')

In [ ]: ts.floor('h')

In [ ]: s = pd.Series(['12-5-2015', '14-1-2013', '20/12/2017', '40/23/2017'])
        pd.to_datetime(s, dayfirst=True, errors='coerce')

In [ ]: ts.year, ts.month, ts.day, ts.hour, ts.minute, ts.second

In [ ]: ts

In [ ]: ts.dayofweek, ts.dayofyear, ts.daysinmonth

In [ ]: pyts = ts.to_pydatetime()

In [ ]: pyts

In [ ]: td = pd.Timedelta(125.8723, unit='h')
        td

In [ ]: td.round('min')

In [ ]: td.components

In [ ]: td.total_seconds()
```

0.1 There's more...

```
In [ ]: date_string_list = ['Sep 30 1984'] * 10000

In [ ]: %timeit pd.to_datetime(date_string_list, format='%b %d %Y')

In [ ]: %timeit pd.to_datetime(date_string_list)
```

1 Slicing time series intelligently

```
In [ ]: crime = pd.read_hdf('crime.h5', 'crime')
        crime.dtypes

In [ ]: crime.head()

In [ ]: crime.info()

In [ ]: crime = crime.set_index('REPORTED_DATE')
        crime.head()

In [ ]: pd.options.display.max_rows = 4

In [ ]: crime.loc['2016-05-12 16:45:00']

In [ ]: crime.loc['2016-05-12']

In [ ]: crime.loc['2016-05-12'].shape

In [ ]: crime.loc['2016-05'].shape

In [ ]: crime.loc['2016'].shape

In [ ]: crime.loc['2016-05-12 03']

In [ ]: crime.loc['2016-05-12 03'].shape

In [ ]: crime.loc['Dec 2015'].sort_index()

In [ ]: crime.loc['2016 Sep, 15'].shape

In [ ]: crime.loc['2016 Sep, 15']

In [ ]: crime.loc['21st October 2014 05']

In [ ]: crime.loc['21st October 2014 05'].shape

In [ ]: crime.loc['2015-3-4':'2016-1-1'].sort_index()

In [ ]: crime.loc['2015-3-4 22':'2016-1-1 23:45:00'].sort_index()

In [ ]: crime.loc['2015-3-4':'2015-3-5']
```

1.1 How it works...

```
In [ ]: crime.info()

In [ ]: crime.memory_usage()

In [ ]: mem_cat=crime.memory_usage().sum()
        mem_cat

In [ ]: crime_obj = crime.astype({'OFFENSE_TYPE_ID':'object',
                                  'OFFENSE_CATEGORY_ID':'object',
                                  'NEIGHBORHOOD_ID':'object'})

        crime_obj.info()

In [ ]: mem_obj = crime_obj.memory_usage().sum()
        mem_obj

In [ ]: mb = 2 ** 20
        round(mem_cat / mb, 1), round(mem_obj / mb, 1)

In [ ]: crime.index[:2]

In [ ]: crime.index[-2:]

In [ ]: crime.head()
```

1.2 There's more...

```
In [ ]: %timeit crime.loc['2015-3-4':'2016-1-1']

In [ ]: crime_sort = crime.sort_index()

In [ ]: %timeit crime_sort.loc['2015-3-4':'2016-1-1']

In [ ]: pd.options.display.max_rows = 60
```

2 Using methods that only work with a DatetimeIndex

```
In [ ]: crime = pd.read_hdf('data/crime.h5', 'crime').set_index('REPORTED_DATE')
        print(type(crime.index))

In [ ]: crime.between_time('2:00', '5:00', include_end=False).head()

In [ ]: crime.at_time('5:47').head()

In [ ]: crime_sort = crime.sort_index()

In [ ]: pd.options.display.max_rows = 6

In [ ]: crime_sort.first(pd.offsets.MonthBegin(6))
```

```
In [ ]: crime_sort.first(pd.offsets.MonthEnd(6))
```

```
In [ ]: crime_sort.first(pd.offsets.MonthBegin(6, normalize=True))
```

```
In [ ]: crime_sort.loc[:'2012-06']
```

```
In [ ]: crime_sort.first('5D')
```

```
In [ ]: crime_sort.first('5B')
```

```
In [ ]: crime_sort.first('7W')
```

```
In [ ]: crime_sort.first('3QS')
```

2.1 How it works...

```
In [ ]: import datetime
        crime.between_time(datetime.time(2,0), datetime.time(5,0), include_end=False)
```

```
In [ ]: first_date = crime_sort.index[0]
        first_date
```

```
In [ ]: first_date + pd.offsets.MonthBegin(6)
```

```
In [ ]: first_date + pd.offsets.MonthEnd(6)
```

2.2 There's more...

```
In [ ]: dt = pd.Timestamp('2012-1-16 13:40')
        dt + pd.DateOffset(months=1)
```

```
In [ ]: do = pd.DateOffset(years=2, months=5, days=3, hours=8, seconds=10)
        pd.Timestamp('2012-1-22 03:22') + do
```

```
In [ ]: pd.options.display.max_rows=60
```

3 Counting the number of weekly crimes

```
In [ ]: crime_sort = pd.read_hdf('crime.h5', 'crime') \
                        .set_index('REPORTED_DATE') \
                        .sort_index()
```

```
In [ ]: crime_sort.resample('W')
```

```
In [ ]: weekly_crimes = crime_sort.resample('W').size()
        weekly_crimes.head()
```

```
In [ ]: len(crime_sort.loc[:'2012-1-8'])
```

```
In [ ]: len(crime_sort.loc['2012-1-9':'2012-1-15'])
```

```
In [ ]: crime_sort.resample('W-THU').size().head()
```

```
In [ ]: weekly_crimes_gby = crime_sort.groupby(pd.Grouper(freq='W')).size()
        weekly_crimes_gby.head()
```

```
In [ ]: weekly_crimes.equals(weekly_crimes_gby)
```

3.1 How it works...

```
In [ ]: r = crime_sort.resample('W')
        resample_methods = [attr for attr in dir(r) if attr[0].islower()]
        print(resample_methods)
```

3.2 There's more...

```
In [ ]: crime = pd.read_hdf('data/crime.h5', 'crime')
        weekly_crimes2 = crime.resample('W', on='REPORTED_DATE').size()
        weekly_crimes2.equals(weekly_crimes)
```

```
In [ ]: weekly_crimes_gby2 = crime.groupby(pd.Grouper(key='REPORTED_DATE', freq='W')).size()
        weekly_crimes_gby2.equals(weekly_crimes_gby)
```

```
In [ ]: weekly_crimes.plot(figsize=(16,4), title='All Denver Crimes')
```

4 Aggregating weekly crime and traffic separately

```
In [ ]: crime_sort = pd.read_hdf('data/crime.h5', 'crime') \
                      .set_index('REPORTED_DATE') \
                      .sort_index()
```

```
In [ ]: crime_quarterly = crime_sort.resample('Q')['IS_CRIME', 'IS_TRAFFIC'].sum()
        crime_quarterly.head()
```

```
In [ ]: crime_sort.resample('QS')['IS_CRIME', 'IS_TRAFFIC'].sum().head()
```

```
In [ ]: crime_sort.loc['2012-4-1':'2012-6-30', ['IS_CRIME', 'IS_TRAFFIC']].sum()
```

```
In [ ]: crime_quarterly_gby = crime_sort.groupby(pd.Grouper(freq='Q'))['IS_CRIME', 'IS_TRAFFIC
        crime_quarterly_gby.equals(crime_quarterly)
```

```
In [ ]: plot_kwargs = dict(figsize=(16,4),
                           color=['black', 'lightgrey'],
                           title='Denver Crimes and Traffic Accidents')
        crime_quarterly.plot(**plot_kwargs)
```

4.1 How it works...

```
In [ ]: crime_sort.resample('Q').sum().head()
```

```
In [ ]: crime_sort.resample('QS-MAR')['IS_CRIME', 'IS_TRAFFIC'].sum().head()
```

4.2 There's more...

```
In [ ]: crime_begin = crime_quarterly.iloc[0]
        crime_begin
```

```
In [ ]: crime_quarterly.div(crime_begin) \
                      .sub(1) \
                      .round(2) \
                      .plot(**plot_kwargs)
```

5 Measuring crime by weekday and year

```
In [ ]: crime = pd.read_hdf('data/crime.h5', 'crime')
        crime.head()
```

```
In [ ]: wd_counts = crime['REPORTED_DATE'].dt.weekday_name.value_counts()
        wd_counts
```

```
In [ ]: days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday',
                'Friday', 'Saturday', 'Sunday']
        title = 'Denver Crimes and Traffic Accidents per Weekday'
        wd_counts.reindex(days).plot(kind='barh', title=title)
```

```
In [ ]: title = 'Denver Crimes and Traffic Accidents per Year'
        crime['REPORTED_DATE'].dt.year.value_counts() \
                                      .sort_index() \
                                      .plot(kind='barh', title=title)
```

```
In [ ]: weekday = crime['REPORTED_DATE'].dt.weekday_name
        year = crime['REPORTED_DATE'].dt.year

        crime_wd_y = crime.groupby([year, weekday]).size()
        crime_wd_y.head(10)
```

```
In [ ]: crime_table = crime_wd_y.rename_axis(['Year', 'Weekday']).unstack('Weekday')
        crime_table
```

```
In [ ]: criteria = crime['REPORTED_DATE'].dt.year == 2017
        crime.loc[criteria, 'REPORTED_DATE'].dt.dayofyear.max()
```

```
In [ ]: round(272 / 365, 3)
```

```
In [ ]: crime_pct = crime['REPORTED_DATE'].dt.dayofyear.le(272) \
                                          .groupby(year) \
                                          .mean() \
                                          .round(3)
        crime_pct
```

```
In [ ]: crime_pct.loc[2012:2016].median()
```

```
In [ ]: crime_table.loc[2017] = crime_table.loc[2017].div(.748).astype('int')
        crime_table = crime_table.reindex(columns=days)
        crime_table
```

```
In [ ]: import seaborn as sns
        sns.heatmap(crime_table, cmap='Greys')
```

```
In [ ]: denver_pop = pd.read_csv('data/denver_pop.csv', index_col='Year')
        denver_pop
```

```
In [ ]: den_100k = denver_pop.div(100000).squeeze()
        crime_table2 = crime_table.div(den_100k, axis='index').astype('int')
        crime_table2
```

```
In [ ]: sns.heatmap(crime_table2, cmap='Greys')
```

5.1 How it works...

```
In [ ]: wd_counts.loc[days]
```

```
In [ ]: crime_table / den_100k
```

5.2 There's more...

```
In [ ]: ADJ_2017 = .748
```

```
        def count_crime(df, offense_cat):
            df = df[df['OFFENSE_CATEGORY_ID'] == offense_cat]
            weekday = df['REPORTED_DATE'].dt.weekday_name
            year = df['REPORTED_DATE'].dt.year

            ct = df.groupby([year, weekday]).size().unstack()
            ct.loc[2017] = ct.loc[2017].div(ADJ_2017).astype('int')

            pop = pd.read_csv('data/denver_pop.csv', index_col='Year')
            pop = pop.squeeze().div(100000)

            ct = ct.div(pop, axis=0).astype('int')
            ct = ct.reindex(columns=days)
            sns.heatmap(ct, cmap='Greys')
            return ct
```

```
In [ ]: count_crime(crime, 'auto-theft')
```

6 Grouping with anonymous functions with a DatetimeIndex

```
In [ ]: crime_sort = pd.read_hdf('data/crime.h5', 'crime') \
                        .set_index('REPORTED_DATE') \
                        .sort_index()
```

```
In [ ]: common_attrs = set(dir(crime_sort.index)) & set(dir(pd.Timestamp))
        print([attr for attr in common_attrs if attr[0] != '_'])
```

```
In [ ]: crime_sort.index.weekday_name.value_counts()
```

```
In [ ]: crime_sort.groupby(lambda x: x.weekday_name)['IS_CRIME', 'IS_TRAFFIC'].sum()
```

```
In [ ]: funcs = [lambda x: x.round('2h').hour, lambda x: x.year]
        cr_group = crime_sort.groupby(funcs)['IS_CRIME', 'IS_TRAFFIC'].sum()
        cr_final = cr_group.unstack()
        cr_final.style.highlight_max(color='lightgrey')
```

6.1 There's more...

```
In [ ]: cr_final.xs('IS_TRAFFIC', axis='columns', level=0).head()
```

```
In [ ]: cr_final.xs(2016, axis='columns', level=1).head()
```

7 Grouping by a DatetimeIndex and another column

```
In [ ]: employee = pd.read_csv('data/employee.csv',
                              parse_dates=['JOB_DATE', 'HIRE_DATE'],
                              index_col='HIRE_DATE')
        employee.head()

In [ ]: employee.groupby('GENDER')['BASE_SALARY'].mean().round(-2)

In [ ]: employee.resample('10AS')['BASE_SALARY'].mean().round(-2)

In [ ]: sal_avg = employee.groupby('GENDER').resample('10AS')['BASE_SALARY'].mean().round(-2)
        sal_avg

In [ ]: sal_avg.unstack('GENDER')

In [ ]: employee[employee['GENDER'] == 'Male'].index.min()

In [ ]: employee[employee['GENDER'] == 'Female'].index.min()

In [ ]: sal_avg2 = employee.groupby(['GENDER', pd.Grouper(freq='10AS')])['BASE_SALARY'].mean()
        sal_avg2

In [ ]: sal_final = sal_avg2.unstack('GENDER')
        sal_final
```

7.1 How it works...

```
In [ ]: 'resample' in dir(employee.groupby('GENDER'))

In [ ]: 'groupby' in dir(employee.resample('10AS'))
```

7.2 There's more...

```
In [ ]: years = sal_final.index.year
        years_right = years + 9
        sal_final.index = years.astype(str) + '-' + years_right.astype(str)
        sal_final

In [ ]: cuts = pd.cut(employee.index.year, bins=5, precision=0)
        cuts.categories.values

In [ ]: employee.groupby([cuts, 'GENDER'])['BASE_SALARY'].mean().unstack('GENDER').round(-2)
```

8 Finding the last time crime was 20% lower with merge_asof

```
In [ ]: crime_sort = pd.read_hdf('data/crime.h5', 'crime') \
                      .set_index('REPORTED_DATE') \
                      .sort_index()
```

```
In [ ]: crime_sort.index.max()

In [ ]: crime_sort = crime_sort[:'2017-8']
        crime_sort.index.max()

In [ ]: all_data = crime_sort.groupby([pd.Grouper(freq='M'), 'OFFENSE_CATEGORY_ID']).size()
        all_data.head()

In [ ]: all_data = all_data.sort_values().reset_index(name='Total')
        all_data.head()

In [ ]: goal = all_data[all_data['REPORTED_DATE'] == '2017-8-31'].reset_index(drop=True)
        goal['Total_Goal'] = goal['Total'].mul(.8).astype(int)
        goal.head()

In [ ]: pd.merge_asof(goal, all_data, left_on='Total_Goal', right_on='Total',
                      by='OFFENSE_CATEGORY_ID', suffixes=('_Current', '_Last'))

In [ ]: pd.merge_asof(goal.sample(10), all_data, left_on='Total_Goal', right_on='Total',
                      by='OFFENSE_CATEGORY_ID', suffixes=('_Current', '_Last'))

In [ ]:
```