scrape_agricultural

December 11, 2019

https://github.com/QuantCS109/TrumpTweets/blob/master/notebooks_data_collection/scrape_agricultural.ip

[1]:
```
import sys
sys.path.append('..') #to add top-level to path

import pandas as pd
import numpy as np
from datetime import datetime
from pandas.tseries.offsets import BDay
from dateutil.relativedelta import relativedelta
from tqdm import tqdm
import re
import os
import pickle
import logging

from modules.opts import FuturesCurve

import warnings
warnings.filterwarnings("ignore")
```

1 Settlement file scraping

We scrape over 1300 CME options Settlement files provided to us by the class. We focus on agricultural products, Corn, Wheat and Soybeans, for which we have semi-daily data. A typical file looks like the one below:

We scrape the following data from the files. For every day, for every agricultural asset: Obtain the futures prices available in the settlement files. Obtain the 5 nearest expirations for both calls and puts. Look for every strike that has open interest. Save this strike, settlement, and open interest.

You can find this information in pickle files, saved as dictionaries, in the intermediate_data folder. The are called corn_dict,pickle, wheat_dict,pickle, soybeans_dict.pickle

[3]:
```
'''
C Corn Futures
DEC18     369'2     372'4     368'0     370'2     370'2     +'6      125068
 ↪ 369'4      140542      768674
```

```
MAR19     381'4     384'6     380'2     382'2     382'4     +'6       45155  ␣
 ↪ 381'6       64247      394455
MAY19     389'0     392'0     387'6     390'0     390'0     +'6       14581  ␣
 ↪ 389'2       17476      133449
JLY19     394'6     397'6     393'4     395'6     395'6     +1'0      16551  ␣
 ↪ 394'6       18990      126174
SEP19     396'2     399'4     395'4A    397'0     397'4     +1'2       2519  ␣
 ↪ 396'2        2450       68756
DEC19     401'4     404'4     400'4     402'4     402'4     +1'0      12583  ␣
 ↪ 401'4       10812      137744
MAR20     410'2     413'4     409'6A    411'2     411'2     +1'0        519  ␣
 ↪ 410'2         329        8443
MAY20     415'0     418'4B    415'0     417'6A    416'6     +1'0          7  ␣
 ↪ 415'6          36        1219
JLY20     421'4     423'4     420'0     421'4A    421'4     +'6         116  ␣
 ↪ 420'6         167        3425
SEP20      ----      ----      ----      ----     415'2     +'6              ␣
 ↪ 414'4           6         470
DEC20     416'0     417'4     415'0     416'2A    416'0     +'2         172  ␣
 ↪ 415'6          70        4704
JLY21     430'0     430'6     430'0     430'0     430'6     +'2          28  ␣
 ↪ 430'4           8          71
DEC21     420'0     420'4     420'0     420'4     420'2     +'4           4  ␣
 ↪ 419'6                     186

CDF NOV18 Short-Dated New Crop Corn Option CALL
2950       ----      ----      ----      ----     107'5     +1'0             ␣
 ↪ 106'5
3000       ----      ----      ----      ----     102'5     +1'0             ␣
 ↪ 101'5
3050       ----      ----      ----      ----      97'5     +1'0             ␣
 ↪  96'5
3100       ----      ----      ----      ----      92'5     +1'0             ␣
 ↪  91'5
3150       ----      ----      ----      ----      87'5     +1'0             ␣
 ↪  86'5
3200       ----      ----      ----      ----      82'5     +1'0             ␣
 ↪  81'5
3250       ----      ----      ----      ----      77'5     +1'0             ␣
 ↪  76'5
3300       ----      ----      ----      ----      72'5     +1'0             ␣
 ↪  71'5
3350       ----      ----      ----      ----      67'5     +1'0             ␣
 ↪  66'5
```

```
3400        ----      ----      ----      ----      62'5    +1'0                ␣
 ↪  61'5
3450        ----      ----      ----      ----      57'5    +1'0                ␣
 ↪  56'5
3500        ----      ----      ----      ----      52'5    +1'0                ␣
 ↪  51'5
3550        ----      ----      ----      ----      47'5    +1'0                ␣
 ↪  46'5
3600        ----      ----      ----      ----      42'5    +1'0                ␣
 ↪  41'5
3650        ----      ----      ----      ----      37'5    +1'0                ␣
 ↪  36'5
3700        ----      ----      ----      ----      32'5    +1'0                ␣
 ↪  31'5
3750        ----      ----      ----      ----      27'5    +1'0                ␣
 ↪  26'5
3800        ----      ----      ----      ----      22'5    +1'0                ␣
 ↪  21'5
3850        15'1      19'0B     15'1      ----      17'5    +1'0          10    ␣
 ↪  16'5                         10
''';
```

List of all Settlements archives

[2]:
```
lst =os.listdir('../data/input_data/settlements')
lst[0:10]
```

[2]:
```
['Corn_19_1_2018_ags_settlements.txt',
 'Wheat_2_6_2017_ags_settlements.txt',
 '13_11_2017_ags_settlements.txt',
 'Soybean Oil_26_9_2017_ags_settlements.txt',
 '21_8_2018_eonly_settlements.txt',
 'Soybean Oil_24_6_2019_ags_settlements.txt',
 'Wheat_17_4_2019_ags_settlements.txt',
 '7_11_2016_ags_settlements.txt',
 '16_7_2018_ags_settlements.txt',
 'Soybeans_13_12_2016_ags_settlements.txt']
```

Getting first entry in file name to identify agricultural contacts

[3]:
```
archs = pd.DataFrame([l.split('_') for l in lst])
list(set(archs[0]))[0:10]
```

[3]:
```
['2', 'Wheat', '12', '13', '14', '20', '10', '15', '27', '8']
```

Identifying agricultural contracts and separating by asset, sorting by date

[4]:
```
archs_corn = archs[ (archs[0]=='Corn') | (archs[0]=='corn') ]
archs_wheat = archs[ (archs[0]=='Wheat') | (archs[0]=='wheat') ]
archs_soybeans = archs[ (archs[0]=='Soybeans') | (archs[0]=='soybeans') ]

archs_corn[[1,2,3]] = archs_corn[[1,2,3]].astype('int')
archs_wheat[[1,2,3]] = archs_wheat[[1,2,3]].astype('int')
archs_soybeans[[1,2,3]] = archs_soybeans[[1,2,3]].astype('int')

arches_corn = archs_corn.sort_values(by=[3,2,1])
arches_wheat = archs_wheat.sort_values(by=[3,2,1])
arches_soybeans = archs_soybeans.sort_values(by=[3,2,1])

print("# of contracts corn: ",arches_corn.shape[0])
print("# of contracts wheat: ",arches_wheat.shape[0])
print("# of contracts soybeans: ",arches_soybeans.shape[0])

#first Corn contract
lst[arches_corn.index[0]]
```

```
# of contracts corn:  457
# of contracts wheat:  424
# of contracts soybeans:  440
```

[4]:
```
'corn_21_9_2016_settlements.txt'
```

Creating a dict with keys the months in the Settlements file format, and values as contract expirations in datetime format. Agricultural products expire on the 14th of every month, or the previous business day if a weekend/holiday

[5]:
```
mon = ['JAN','FEB',"MAR",'APR','MAY','JUN','JLY','AUG','SEP','OCT','NOV','DEC']
dummy_dict = {i : mon[i-1] for i in range(1,13)}
exp_dict = {}
for year in [16,17,18,19,20]:
    for month in range(1,13):
        exp_dict[mon[month - 1]+str(year)] = datetime(2000 + year, month, 15) -␣
 ↪BDay(1)

exp_dict['JAN16']
```

[5]:
```
Timestamp('2016-01-14 00:00:00')
```

[6]:
```
def scrape_settlements(asset):
    """
    This function scrapes all agricultural settlement files and obtains a␣
 ↪dictionary
    :param asset: Instrument to scrape, can be 'Corn', 'Wheat', or 'Soybeans'
```

```
    :return asset_full_dict: a dictionary with keys as datetime dates, those of␣
↪the related
        settlement file date. The values of the dictionary are two␣
↪dictionaries, one with key
        'Call' and one with key 'Put'. The values for each of these is 5␣
↪dictionaries, with keys
        0,1,..,4, with settlement information for a particular options contract␣
↪with a particular
        expiration date
    """

    # Futures curve for futures prices under 2 months in case settlement file␣
↪doesn't
    # have futures contract for an options expiry
    ft = FuturesCurve()
    asset_full_dict = {}

    # Indices for a specific asset's settlement files
    if asset == 'Corn':
        arches = arches_corn
    elif asset == 'Wheat':
        arches = arches_wheat
    elif asset == 'Soybeans':
        arches = arches_soybeans

    # List with dates from each settlement file name in datetime
    dates_list = []
    for ind in arches.index:
        dates_list.append(datetime(int(arches[3].loc[ind]),
                                   int(arches[2].loc[ind]), int(arches[1].
↪loc[ind])))

    # Loop through all files
    for arch_ind, arch in enumerate(tqdm(arches.index)):

        # If there's any unidentified error, for a particular date (could be␣
↪corrupt file or other)
        # ignore that date
        try:

            date_dict = {}
            doc_date = dates_list[arch_ind]
            doc = []
            with open('../data/input_data/settlements/' + lst[arch]) as f:
                for row in f:
                    doc.append(row)
```

```
            #Identifying options contracts by code
            if asset == 'Corn':
                cont_name = 'PY'
            elif asset == 'Wheat':
                cont_name = 'WZ'
            elif asset == "Soybeans":
                cont_name = 'CZO'

            # Finding indices inside the document for each option expiration␣
↪for lead contracts
            py = [d.startswith(cont_name) for d in doc]
            py_index = pd.DataFrame(doc)[py].index

            # Putting document in a DataFrame
            text = [re.sub("\s+", ",", d.strip()).split(',') for d in doc]
            df = pd.DataFrame(text)
            df = df[[0,1,2,3,4,8,9,10]]
            df['expiration'] = np.zeros(df.shape[0])
            df['future'] = np.zeros(df.shape[0])
            df['date'] = dates_list[arch_ind]

            # Converting column 8 (yesterday's settlement prices) into a float␣
↪with same format as strike column,
            # or empty string if non numerical data
            for ind in df.index:
                num = str(df[[8]].iloc[ind]).split()[1].split('\'')
                try:
                    if num[0] == '':
                        num[0] = 0
                    num = 10*float(num[0]) + float(num[1])
                    df[8].loc[ind] = num
                except:
                    pass

            # Identifying futures prices for the day
            futs = df[[0,8]][1:6]
            futs = futs.set_index(0)

            # Identifying if call or put
            py_index_call = py_index[df.iloc[py_index][4] == 'CALL']
            py_index_put = py_index[df.iloc[py_index][4] == 'PUT']

            # Looping through both calls and puts for a specific options code
            for py_i, py_ind in enumerate([py_index_call, py_index_put]):
                vol_dict = {}
```

```
                # Focus only on the first 5 expirations date for each contract
                for i in range(5):
                    last_ind = py_ind[i]
                    exp = str(df.iloc[py_ind[i]][1])

                    # Only look at options with at least 7 business days to␣
↪expiration to avoid
                    # implied volatility calculation issues
                    if exp_dict[exp] >= doc_date + BDay(7):

                        # Check if we have a futures price for a particular␣
↪options expiration
                        if exp in futs.index:
                            while df[0][last_ind + 1].isnumeric():
                                last_ind += 1
                            df.expiration[py_index[i]:last_ind] = exp_dict[exp]
                            df.future = float(futs.loc[exp])

                            #Dropping all NAs as they can have stale (wrong)␣
↪settlement prices
                            vol_dict[i] = df[py_ind[i]:last_ind][1:].dropna()
                            vol_dict[i] =␣
↪vol_dict[i][[0,8,9,10,'expiration','future','date']]
                            vol_dict[i] = vol_dict[i].rename(columns={0:
↪'strike',
                                                                      8:
↪'settle',
                                                                      9:
↪'volume',
                                                                      10:'oi'})
                            vol_dict[i].strike = vol_dict[i].strike.
↪astype('float')
                            vol_dict[i].settle = vol_dict[i].settle.
↪astype('float')
                            vol_dict[i].volume = vol_dict[i].volume.
↪astype('int')
                            vol_dict[i].oi = vol_dict[i].oi.astype('int')

                        # If we don't have the futures price for a particular␣
↪option, get it from
                        # FuturesCurve by interpolation if under two months
                        elif exp_dict[exp] <= pd.to_datetime(doc_date +␣
↪relativedelta(months = 2)- BDay(2)):
                            if asset == 'Corn':
                                fut_code = 'C'
                            elif asset == 'Wheat':
```

```
                                fut_code = 'W'
                            else:
                                fut_code = 'S'

                            # If there's an key error retreiving an␣
↪interpolated futures price,
                            # ignore options contract
                            try:
                                # Get price from FuturesCurve
                                fut_px = ft.get(fut_code, doc_date - BDay(1),␣
↪exp_dict[exp])
                                while df[0][last_ind + 1].isnumeric():
                                    last_ind += 1
                                df.expiration[py_index[i]:last_ind] =␣
↪exp_dict[exp]
                                # 10 * adjustment to match strike format
                                df.future = 10*fut_px
                                vol_dict[i] = df[py_ind[i]:last_ind][1:].
↪dropna()
                                #vol_dict[i] = df[py_ind[i]:last_ind][1:].
↪fillna(0)
                                vol_dict[i] = vol_dict[i][[0,8,9,␣
↪10,'expiration','future','date']]
                                vol_dict[i] = vol_dict[i].rename(columns={0:
↪'strike',
                                                                         8:
↪'settle',
                                                                         9:
↪'volume',
                                                                         10:
↪'oi'})
                                vol_dict[i].strike = vol_dict[i].strike.
↪astype('float')
                                vol_dict[i].settle = vol_dict[i].settle.
↪astype('float')
                                vol_dict[i].volume = vol_dict[i].volume.
↪astype('int')
                                vol_dict[i].oi = vol_dict[i].oi.astype('int')
                            except KeyError:
                                vol_dict[i] = ''
                        else:
                            vol_dict[i] = ''
                    # In any other case, return empty string for an options␣
↪contract
                    else:
                        vol_dict[i] = ''
```

```
                if py_i == 0:
                    date_dict['Call'] = vol_dict
                else:
                    date_dict['Put'] = vol_dict
            asset_full_dict[doc_date] = date_dict
        except Exception as e:
            logging.error(e,exc_info=True)
            print('Error with archive : {} in arches.index', arch)

    return asset_full_dict
```

[7]:
```
corn_dict = scrape_settlements('Corn')
wheat_dict = scrape_settlements('Corn')
soybeans_dict = scrape_settlements('Corn')
```

```
100%|      | 457/457 [17:51<00:00,  2.34s/it]
```

[8]:
```
corn_dict[list(corn_dict.keys())[1]]['Call'][3]
```

[8]:

	strike	settle	volume	oi	expiration	future	date
1057	3400.0	241.0	7	8767	2017-03-14 00:00:00	3500.0	2016-09-22
1058	3500.0	190.0	235	18542	2017-03-14 00:00:00	3500.0	2016-09-22
1059	3600.0	147.0	80	5823	2017-03-14 00:00:00	3500.0	2016-09-22
1060	3700.0	114.0	122	8834	2017-03-14 00:00:00	3500.0	2016-09-22
1061	3800.0	86.0	550	8678	2017-03-14 00:00:00	3500.0	2016-09-22
1062	3900.0	66.0	36	8963	2017-03-14 00:00:00	3500.0	2016-09-22
1063	4000.0	52.0	89	12390	2017-03-14 00:00:00	3500.0	2016-09-22
1064	4100.0	40.0	5	4069	2017-03-14 00:00:00	3500.0	2016-09-22
1065	4200.0	32.0	312	5164	2017-03-14 00:00:00	3500.0	2016-09-22

[13]:
```
with open('../data/intermediate_data/corn_dict.pickle', 'wb') as handle:
    pickle.dump(corn_dict, handle)
with open('../data/intermediate_data/wheat_dict.pickle', 'wb') as handle:
    pickle.dump(wheat_dict, handle)
with open('../data/intermediate_data/soybeans_dict.pickle', 'wb') as handle:
    pickle.dump(soybeans_dict, handle)
```