Work Sample

Collect futures data according to its index from JoinQuant API.

1. User can choose different kinds of futures and time range to collect different data
2. Collect and print missing dates in each CSV file (except weekends)
3. Fill in missing data according to missing dates in each CSV file to improve efficiency

Code:

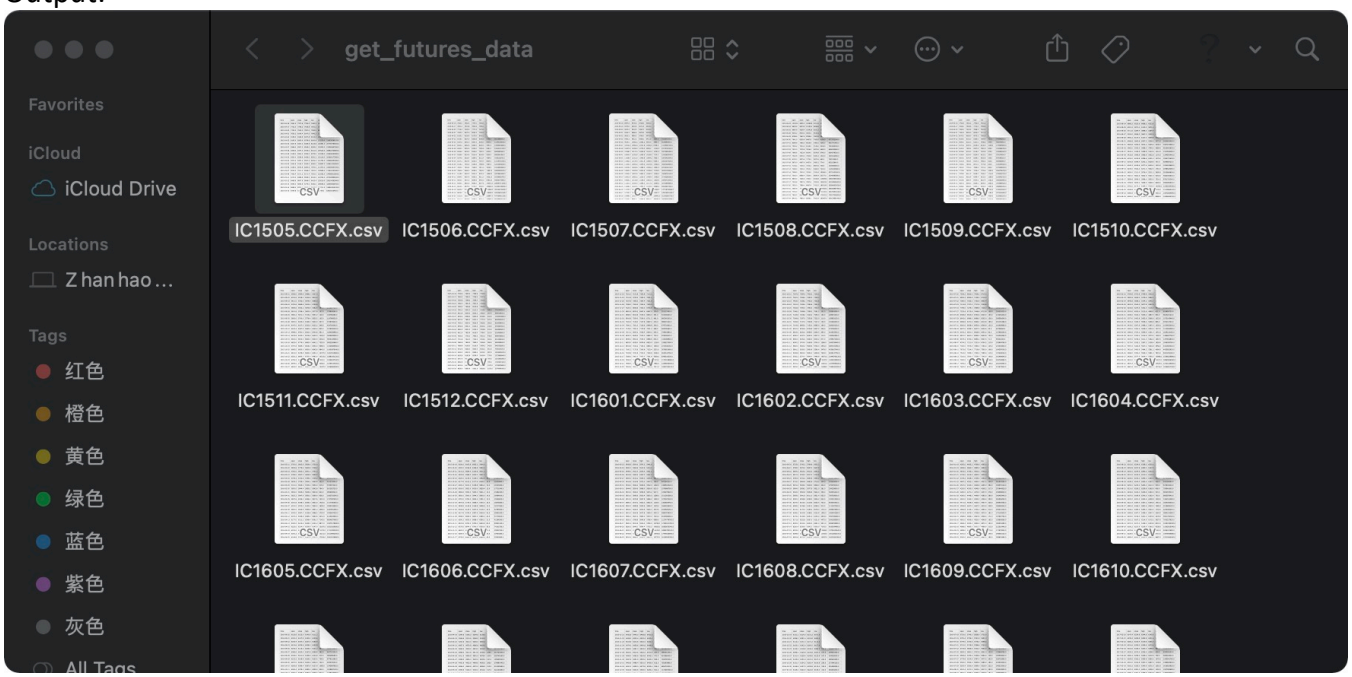
get_info_per_index_futures.py - /Users/tylerwang/Desktop/StudyQuant/量化编辑 12_27 王展浩/get_info_per_index_futures.py (3....

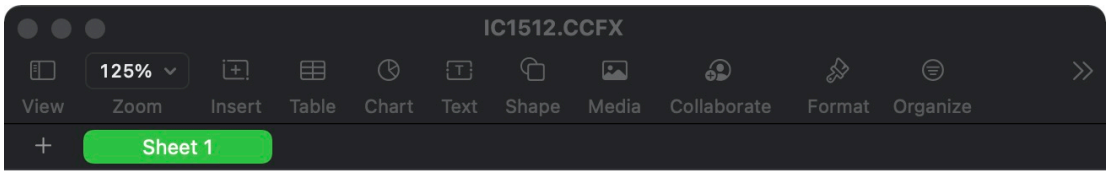
```python
import pandas as pd
from datetime import datetime, timedelta
import jqdatasdk as jq
import os


class FuturesInfo:
    data_exist = []
    empty = []

    def __init__(self, f_start, f_end, f_kind):
        self.f_start = f_start
        self.f_end = f_end
        self.f_kind = f_kind
        self.path = the_path

    def get_large_data(self):
        global data_exist
        global empty
        f = open("futures_list.txt", "w+")
        lis = f.read()
        data_exist = lis.split(" ")
        # log in the  JQ account
        jq.auth('18224433211', 'Haohao19971003.')
        all_futures = jq.get_all_securities(['futures'])
        # name the index of the DataFrame
        all_futures.index.name = 'id'
        # convert index column to list
        futures_idx = all_futures.index.values.tolist()
        substring = self.f_kind
        empty = []
        for idx in futures_idx:
            if substring in idx:
                empty.append(idx)
        count = 0
        for item in empty:
            future_info_yearly = jq.get_price(security=item, start_date=datetime.strptime(self.f_start, '%Y-%m-%d'),
                                              end_date=datetime.strptime(self.f_end, '%Y-%m-%d'), frequency='1m')
            str = item + '.csv'
            future_info_yearly.index.name = 'time'
            future_info_yearly.dropna(inplace=True)
            future_info_yearly.to_csv(os.path.join(self.path, str), mode='a')
            if item not in data_exist:
                data_exist.append(item)

            count += 1
            print(count)
        f = open("futures_list.txt", "w")
        f.write(" ".join(empty))
        f.close()
        for item in data_exist:
            if item == "":
                continue
            str = item + '.csv'
            df = pd.read_csv(self.path + str)
            df.drop_duplicates(subset=['time'], inplace=True)
            df.sort_values(by='time', inplace=True)
            df = df[df.time.str.contains('time') == False]
            df.to_csv(os.path.join(self.path, str), index=False, index_label='time')
        print(data_exist)


    def get_missing_data(self):
        global data_exist
        global path
        f = open("futures_list.txt", "r+")
        lis = f.read()
        data_exist = lis.split(" ")
        futures_dict = {}
        substring = self.f_kind
        for item in data_exist:
            if item == "":
                continue
            if substring in item:
                str = item + '.csv'
```

Ln: 1 Col: 0

```
                future_info = pd.read_csv(self.path + str)
                # groupby datetime and get the size of futures of that day
                date_with_data = future_info.groupby(['time']).size()
                all_date = pd.date_range(start=datetime.strptime(self.f_start, '%Y-%m-%d'),
                                         end=datetime.strptime(self.f_end, '%Y-%m-%d'))
                date_with_data.index = pd.DatetimeIndex(date_with_data.index)
                # set new index to data series, if its NaN then 0
                date_with_data = date_with_data.reindex(all_date, fill_value=0)
                # get the date of missing data
                date_with_data = date_with_data[date_with_data.index.dayofweek < 5]
                miss_date_list = []
                for index, value in date_with_data.items():
                    if value == 0:
                        miss_date_list.append(index.strftime('%Y-%m-%d'))
                        futures_dict[item] = miss_date_list
        for key, value in futures_dict.items():
            print(key, end=":")
            print(value)
        jq.auth('18224433211', 'Haohao19971003.')
        count = 0
        for item in futures_dict:
            date_list = []
            for time in futures_dict[item]:
                date_list.append(datetime.strptime(time, '%Y-%m-%d'))
            max_date = max(date_list)
            min_date = min(date_list)
            future_info = jq.get_price(security=item,
                                       start_date=min_date,
                                       end_date=max_date, frequency='1m')
            str = item + '.csv'
            future_info.to_csv(os.path.join(self.path, str), mode='a')
            count += 1
            print(count)
        for item in data_exist:
            if item == "":
                continue
            str = item + '.csv'
            df = pd.read_csv(self.path + str)
            df.drop_duplicates(subset=['time'], inplace=True)
            df.sort_values(by='time', inplace=True)
            df.dropna(inplace = True)
            df.to_csv(os.path.join(self.path, str), index=False, index_label='time')



# Name of Exchange         Code
# Shanghai Futures         XSGE
# Dalian Commodity         XDCE
# Zhengzhou Commodity      XZCE
# China Financial Futures  CCFX
# All futures data          ''
# this part is used for initial downloading
the_path = '/Users/tylerwang/Desktop/get_futures_data/'
# get_large = FuturesInfo('2015-01-01','2015-02-01','CCFX')
# get_large.get_large_data()

# # # this part is used for appending new data
get_miss = FuturesInfo('2003-01-01', '2015-02-01', 'CCFX')
get_miss.get_missing_data()
```

Output:


get_futures_data
Favorites
iCloud
iCloud Drive
Locations
Z han hao ...
Tags
红色
橙色
黄色
绿色
蓝色
紫色
灰色
IC1505.CCFX.csv
IC1506.CCFX.csv
IC1507.CCFX.csv
IC1508.CCFX.csv
IC1509.CCFX.csv
IC1510.CCFX.csv
IC1511.CCFX.csv
IC1512.CCFX.csv
IC1601.CCFX.csv
IC1602.CCFX.csv
IC1603.CCFX.csv
IC1604.CCFX.csv
IC1605.CCFX.csv
IC1606.CCFX.csv
IC1607.CCFX.csv
IC1608.CCFX.csv
IC1609.CCFX.csv
IC1610.CCFX.csv



IC1512.CCFX

time	open	close	high	low	volume	money
2015-04-16	7799.6	7625.2	7799.6	7525.0	484.0	739628800.0
2015-04-17	7625.2	7562.0	7706.2	7527.4	460.0	698317240.0
2015-04-20	7559.8	7388.2	7638.4	7302.4	425.0	634177400.0
2015-04-21	7462.0	7941.8	7947.6	7414.4	632.0	959647520.0
2015-04-22	7959.2	7926.8	8019.8	7900.2	751.0	1193756480.0
2015-04-23	7942.0	8043.0	8113.0	7918.0	486.0	778144360.0
2015-04-24	8015.8	8039.0	8053.0	7862.4	357.0	568516600.0
2015-04-27	8039.0	8068.2	8320.8	8026.6	633.0	1032973680.0
2015-04-28	8074.0	7894.2	8141.4	7777.2	537.0	848397520.0
2015-04-29	7900.0	8024.0	8025.8	7825.6	392.0	623508600.0
2015-04-30	8056.8	8041.0	8123.0	8020.4	233.0	376494760.0
2015-05-04	8060.0	8031.2	8132.0	7919.8	231.0	369382080.0
2015-05-05	8012.0	7830.6	8016.0	7780.0	210.0	331660960.0
2015-05-06	7830.6	7705.0	7950.0	7601.0	448.0	698972680.0
2015-05-07	7710.8	7612.0	7720.0	7610.0	393.0	601429240.0
2015-05-08	7660.8	7859.0	7859.0	7644.6	368.0	569350840.0
2015-05-11	7899.8	8109.0	8116.6	7854.2	343.0	546275280.0
2015-05-12	8132.6	8226.0	8230.0	8042.2	461.0	750814720.0
2015-05-13	8182.2	8203.0	8265.4	8100.0	987.0	1618820240.0
2015-05-14	8192.6	8162.8	8278.0	8100.4	779.0	1273860840.0
2015-05-15	8147.0	8216.6	8318.4	8101.4	764.0	1251830280.0
2015-05-18	8282.8	8378.2	8428.8	8222.2	795.0	1330901320.0
2015-05-19	8432.6	8800.4	8848.0	8350.0	1037.0	1793879880.0
2015-05-20	8838.0	8990.8	9181.6	8826.6	1150.0	2079454000.0
2015-05-21	9070.4	9435.0	9458.8	9042.6	1699.0	3151132360.0
2015-05-22	9500.2	9449.4	9583.4	9250.0	1494.0	2808296200.0
2015-05-25	9249.0	9646.6	9675.6	9151.0	910.0	1727307840.0
2015-05-26	9649.0	10200.6	10200.6	9649.0	1227.0	2437037920.0
2015-05-27	10245.0	10404.0	10503.0	10085.2	1878.0	3855006280.0
2015-05-28	10450.0	9840.0	10669.8	9750.0	2904.0	5992115120.0
2015-05-29	9858.2	10035.0	10200.0	9700.0	2898.0	5817874960.0
2015-06-01	10099.8	10906.0	10950.0	10063.2	1764.0	3700893000.0
2015-06-02	10922.0	11387.6	11548.0	10906.6	3169.0	7081306160.0