Project2

2020068495 정천일, 2021016171 김선재

import timeit

import requests

import pandas as pd

import datetime

import os

import itertools

def get\_sim\_df\_order (fn): # csv파일 불러오기 fn은 파일명 위치까지? 아니 확장명은 빼고 용도까지 빼고 "2023-10-27-bithumb-BTC' 까지 -book.csv는 제외

    print ('loading... %s' % fn)

    book\_fn=fn+'-book.csv'

    df = pd.read\_csv(book\_fn, delimiter='|', header=None, names=['price', 'quantity', 'type', 'timestamp']).apply(pd.to\_numeric,errors='ignore') # 숫자로 변환후 무시

    #print df.to\_string();print '------'

    group\_order = df.groupby(['timestamp']) # timestamp를 기준으로 하나의 그룹만듬.

    print("complete")

    return group\_order

def write\_csv(fn, df):

    book\_fn=fn+'-book.csv' #저장할 때 편하게 하려고

    new\_fn = fn+'-feature.csv'

    should\_write\_header = os.path.exists(new\_fn)

    if should\_write\_header == False:

        df.to\_csv(new\_fn, index=False, header=True, mode = 'a')

    else:

        df.to\_csv(new\_fn, index=False, header=False, mode = 'a')

def faster\_calc\_indicators(fn):

    start\_time = timeit.default\_timer()

    # FROM CSV FILES (DAILY)

    #group\_o = get\_sim\_df\_order(raw\_book\_csv(raw\_fn, ('%s-%s-%s' % (\_tag, exchange, currency))))

    group\_order=get\_sim\_df\_order(fn)  #=group\_order를 나타낸다

    #group\_t = get\_sim\_df\_trade(raw\_trade\_csv(raw\_fn, ('%s-%s-%s' % (\_tag, exchange, currency)))) #fix for book-1 regression

    delay = timeit.default\_timer() - start\_time

    print ('df loading delay: %.2fs' % delay)

    level\_1 = 2

    level\_2 = 5

    print ('param levels' ,level\_1, level\_2)  #, exchange, currency)

    #(ratio, level, interval seconds )

    #book\_imbalance\_params = [(0.2,level\_1,1),(0.2,level\_2,1)]  # 간편하게 매개변수 조절하기 위함

    seq = 0

    print ('total groups:', len(group\_order.size().index))

    #main part

    #employee\_df={}

    employee\_df=pd.DataFrame({'book-imbalnace-0.2-5-1':pd.Series([]),'mid\_price\_top':pd.Series([]),'mid\_price\_wt':pd.Series([]),'mid\_price\_wt\_bias':pd.Series([]),'mid\_price\_mkt':pd.Series([]),

                  'mid\_price\_del':pd.Series([]),'mid\_price\_dom':pd.Series([]),'timestamp':pd.Series([])})

    new\_employee={'book-imbalnace-0.2-5-1':pd.Series([]),'mid\_price\_top':pd.Series([]),'mid\_price\_wt':pd.Series([]),'mid\_price\_wt\_bias':pd.Series([]),'mid\_price\_mkt':pd.Series([]),

                  'mid\_price\_del':pd.Series([]),'mid\_price\_dom':pd.Series([]),'timestamp':pd.Series([])}

    #for gr\_order in itertools.izip (group\_order):

    keys=group\_order.groups.keys()

    for i in keys:

        gr\_order=group\_order.get\_group(i)

        if gr\_order is None :

            print ('Warning: group is empty')

            continue

         #timestamp = (gr\_order[1].iloc[0])['timestamp']

        gr\_bid\_level = gr\_order[(gr\_order.type == 0)]

        gr\_ask\_level = gr\_order[(gr\_order.type == 1)]

        ratio=0.2 ; level=5 ; interval=1

        mid\_price\_top=(gr\_bid\_level.iloc[0].price + gr\_ask\_level.iloc[0].price) \* 0.5

        mid\_price\_wt=((gr\_bid\_level.head(level))['price'].mean() + (gr\_ask\_level.head(level))['price'].mean()) \* 0.5## 주의사항: csv가 이미 오름차순으로 정렬되어 있어야 한다.

        print(mid\_price\_wt)

        mid\_price\_mkt= round((gr\_bid\_level.iloc[0].price\*gr\_ask\_level.iloc[0].quantity + gr\_ask\_level.iloc[0].price\*gr\_bid\_level.iloc[0].quantity)

                             /(gr\_bid\_level.iloc[0].price+gr\_ask\_level.iloc[0].quantity),1)

        mid\_price\_wt\_bias=((gr\_bid\_level.head(level))['price'].mean() - (gr\_ask\_level.head(level))['price'].mean()) \* 0.5 # price\_wt가 bid에 가까운지 ask에 가까운지 (음수면 ask, 양수면 bid로 편향)

        #mid\_price\_vwap = (group\_t['total'].sum())/(group\_t['units\_traded'].sum())

        mid\_price\_del=((gr\_bid\_level.head(level))['price'][1:].mean() + (gr\_ask\_level.head(level))['price'][1:].mean()) \* 0.5

        mid\_price\_dom=((gr\_bid\_level.head(5))['quantity'].sum() + (gr\_ask\_level.head(5))['quantity'].sum()) \* 0.5   # Deepths Of Market

        #gr\_bid\_level.head(5)['quantity'].sum() = bid of dom

        #gr\_ask\_level.head(5)['quantity'].sum() = ask of dom

        #(gr\_bid\_level.iloc[0].price + gr\_ask\_level.iloc[0].price) \* 0.5 = mid price in given hint.py

        quant\_v\_bid = gr\_bid\_level.quantity\*\*ratio

        price\_v\_bid = gr\_bid\_level.price \* quant\_v\_bid

        quant\_v\_ask = gr\_ask\_level.quantity\*\*ratio

        price\_v\_ask = gr\_ask\_level.price \* quant\_v\_ask

        askQty = quant\_v\_ask.values.sum()

        bidPx = price\_v\_bid.values.sum()

        bidQty = quant\_v\_bid.values.sum()

        askPx = price\_v\_ask.values.sum()

        bid\_ask\_spread = interval

        book\_price = (((askQty\*bidPx)/bidQty) + ((bidQty\*askPx)/askQty)) / (bidQty+askQty)

        book\_imbalance=(book\_price-mid\_price\_top)/bid\_ask\_spread

        new\_employee = pd.DataFrame({'book-imbalnace-0.2-5-1': [book\_imbalance],'mid\_price\_top': [mid\_price\_top],'mid\_price\_wt': [mid\_price\_wt]

                           ,'mid\_price\_wt\_bias':[mid\_price\_wt\_bias],'mid\_price\_mkt': [mid\_price\_mkt], 'mid\_price\_del': [mid\_price\_del], 'mid\_price\_dom':[mid\_price\_dom],'timestamp': [i]})

        employee\_df = employee\_df.append(new\_employee, ignore\_index=True)

        seq += 1

        if seq%500==0 : # 진행률 확인하려고

            print(seq)

    write\_csv(fn, employee\_df)

faster\_calc\_indicators('./2023-10-27-bithumb-BTC') # 우리의 경우는 './2023-10-27-bithumb-BTC-book.csv'