Market Data downloading

# HKEX\_NO\_ADJ: hkex\_no\_adjust

Download data:

these codes are in *code\_download\_hkex\_noadj* folder

1. Step1: find max date in MarketDataUpdate

*code: findingDayRangeInMarketData.py*

1. Step2: download data from bloomberg

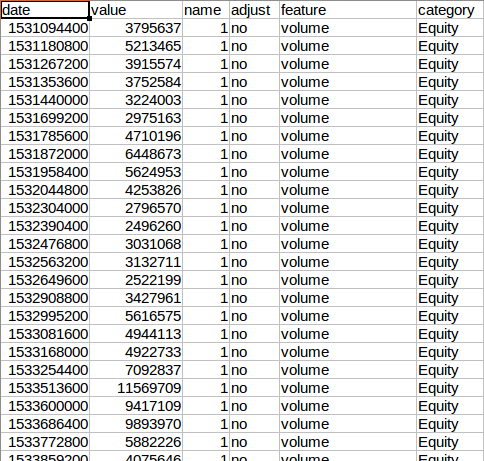
*code: downloadNoAdj.py*

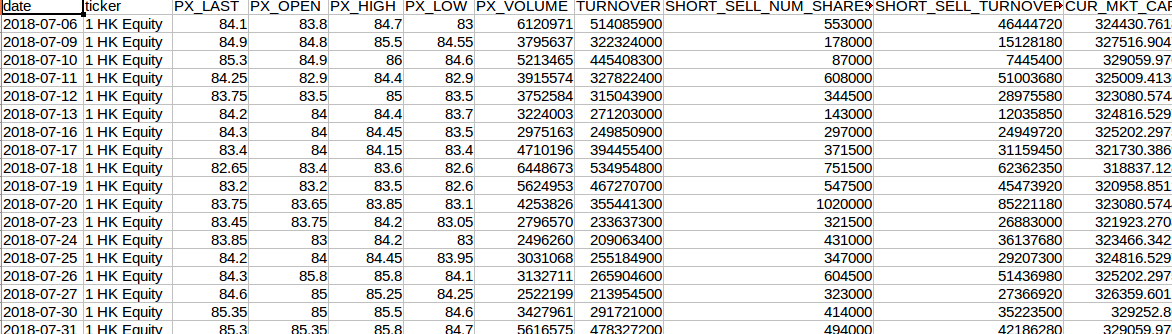
only do part 1 can download data.

1. Step 3: change format of data

*code: changeFormatHkex.py*

this is because the data uploaded should be the format like that.



But data downloaded from bloomberg is like

1. Step 4: upload csv file to *MongoDBcode:*

*code:uploadHkexNOadjustToMongoDB.py*

# **BEFORE unloading, make sure MarketDataUpdate is empty.**

I highly recommend that we download HKEX\_NOADJ data through above 4 scripts, although you can only use *updatehkex-C.py*, this script. This is because we can only use Carol’s computer to download data and it takes us much time to change format of data so that other people cannot use Carol’s computer.

1. Step 5: update data without overlap in MarketData

*code: merge\_hkex\_without\_overlap.py*

this step is to make sure we do not update dupliacted data

Checking data:

* All scripts about checking are in *checking\_hkex\_noadj.py*

1. check whether correctly downloaded

we randomly retrieve a value in Bloomberg and check whether the data in MongoDB is the same

*Function: def check\_downloaded()*

1. check whether download all the tickers

*Function: def checkDownloaded()*

1. check whether dtype is the same as the one in MongoDB

*Function: def check\_dtype()*

1. find max day in MarketData and min day in MarketDataUpdate

this step is to check max day in MarketData and min day in MarketDataUpdate. We should make sure there is no missing data.

*Function: def findDayRange()*

1. checking whether dataset has duplicated data

*Function: def checking\_duplicated():*

1. if the dataset has duplicated data, drop duplicated data

*Function: def dropDuplicated()*

# *#Todo: range test, e.g dividend yield should < 1,*

* Three standard deviation principles checking data we downloaded

The script below only check close price of no adjusted price

*Code: roling\_std.py*

Ticker list

1. updated list

Actually the ticker list we use (universe 1025.csv) is not the finally updated ticker list. Therefore, when updating data, it may raise some errors.

Final ticker list (2019. 8.9 ) contains ticker in MarketDataUpdate. It can be listed as follows:

*File: MarketDataUpdate\_list(2019.8.9).csv*

1. NO new data list

When downloading data, some ticker have no new data and you would find that you download an empty file.

*code: update\_stock\_empty\_no\_new-data.py*

1. **Abnormal start ticker list:**

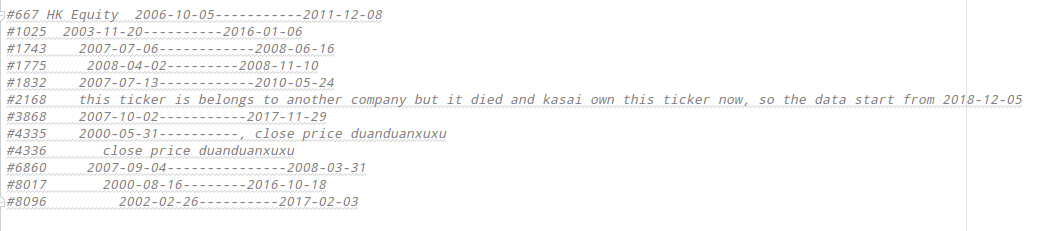
**The list contains tickers that only have data after 2018 in Bloomberg. But in MongoDB MarketData they have previous time data. One reason is that ticker is replaced by a new company after previous one delisted, such as 2168 HK Equity, kaisa company**

|  |  |
| --- | --- |
| **New data info** | |
| **8096 HK Equity** | 2019.3.14—now |
| **4335 HK Equity** | no new data |
| **4336 HK Equity** | no new data |
| **6860 HK Equity** | 2018.7.11—now |
| **667 HK Equity** | 2019.6.11—now |
| **8017 HK Equity** | 2018.09.27 –now |
| **2168 HK Equity** | 2018.12.5 –now |
| **3868 HK Equity** | 2019.5.27 –now |
| **1832 HK Equity** | 2019.5.15 –now |
| **1775 HK Equity** | 2018.7.12 –now |
| **1743 HK Equity** | 2019.1.3 –now |
| **1025 HK Equity** | 2019.2.27 –now |

(now means until today，2019.8.9, they have data)

Old data info:

*Code: abnormal\_start\_list\_info.py*



1. 358ticker\_list

File: 358ticker\_list.csv

the ticker list contains tickers have relatively large market capital

There are 4 companies delisted in the list

|  |  |
| --- | --- |
| **Delisted ticker** | |
| **805 HK Equity** | Up to 2018.1.30 |
| **13 HK Equity** | No data |
| **1880 HK Equity** | Up to 2017.1.27 |
| **1893 HK Equity** | Up to 2018.4.23 |

# HKEX\_ADJ

Download data

We do not download adjust price but calculate it.

: these data is in HKEX\_AF\_Daily, MarketDataUpdate, {feature: 'price\_af\_star'}

can be found in HKEX\_AF\_Daily, MarketDataUpdate

until now, we only use feature: price\_af.

Calculating adjust price

*Code: calculate\_adj\_price.py*

**Checking data:**

1. Calculate error between adjprice we calculate and adj price download from Bloomberg

*Code: finding\_the\_date\_fix\_enpowerment.py*

This script is to find old adjustment factor and list wrong\_cal\_error is about this.

After checking ticker in *358ticker\_list.csv,* the result seems OK.

*File: check\_list\_error.csv*

1. Checking price equal

If a stock does not have adjustment factor, it seems its adjust price is equal to unadjust price. This script is to check whether these two prices are equal.

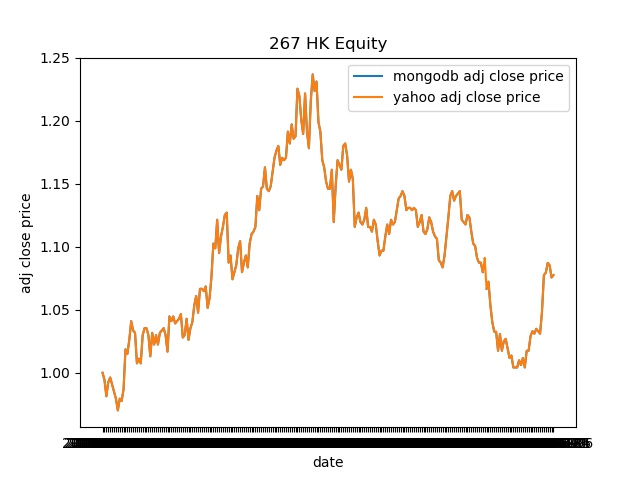
*Code: checking\_whether\_noadjustment\_price\_equal.py*

1. Checking the trend, compared with Yahoo Finance

*Code: adj\_price\_plot\_trend.py*

If adj price is correctly calculated, the trend should be the same as data from Yahoo.

Result can be seen as follows:

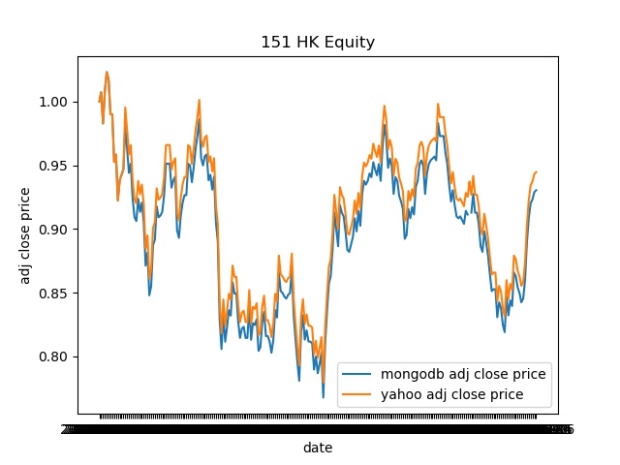
*\*

We can do this by randomly choose 30 samples from ticker\_list. Then, manually download data from Yahoo Finance.

The way to download data from Yahoo can see this file.

*File:Download data from Yahoo Finance.docx*

during checking, we found that there is something wrong.



When checking 151 HK Equity, the trend between the price we calculated and the one in yahoo are not the same, so we should check data in bloomberg.

1. Checking the trend, compared with bloomberg

*code: adj\_compared\_with\_bbg.py*

since there is monthly limited in bloomberg, we can only download the data need to be double checked.

If the trend between ours and bloomberg’s are the same, we finish checking.

# HKEX\_AF

1. Step 1: Download hkex\_adjustment factor

*Code: update\_hkex\_adjustment\_factors.py*

Also, I recommend download first and then upload.

1. Step 2: data cleaning and upload to MongoDB

*Code: af\_data\_cleaning\_and\_upload.py*

**HKEX\_AF\_Daily**

* + - 1. Step 1: delete old data

Make sure only delete {feature: ‘price\_af’}, keep ‘price\_af star’ data

*Code: delete\_price\_af.py*

# FX\_Vol

Download Data

1. Step 1: find min date in MarketData and max date in MarketDataUpdate

*Code: findDayRangeFXVOL.py*

1. Step 2: download data in Bloomberg

*Code: update\_fx\_vol\_c.py*

Since FX\_Vol Data would be downloaded in a csv file and the file is not big, we can directly upload

# **BEFORE unloading, make sure FX\_Vol in MarketDataUpdate is empty.**

# FX

We download FX data in Factset

Factset usage

*File: update\_fx\_about\_macro.doc*

Download Data

1. Step 1：downolad csv file in Factset

*Code: macro.py*

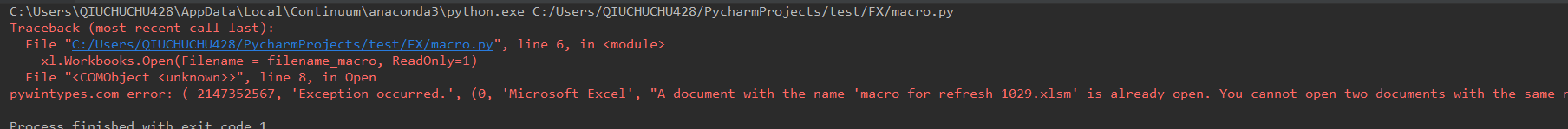
* Make sure excel has open Factset
* Make sure the path is modified in macro

Now the path is



* During updating data, keep *macro\_for\_refresh\_1029.csv file* close

Otherwise, it would raise below error



* Make sure *Template\_FX.xlsx* is open

1. Step 2：change format

*Code: changeFormat\_FX.py*

1. Step 3: upload to MongoDB

*Code: uploadCsvToMongoDB.py*

# Equity

1. Step 1: find min date in MarketData and max date in MarketDataUpdate

*Code: check\_Equity.py----def checkdayRange()*

1. Step 2: Download Equity data in Bloomberg

*Code: update\_Equity.py*

# Checking data:

All scripts about checking are in *checking\_Equity.py*

* Checking whether correctly downloaded

randomly retrieve a value in Bloomberg and check whether the data in MongoDB is the same

*Function: def check\_downloaded()*

* check whether dtype is the same as the one in MongoDB

*Function: def check\_dtype()*

* check if all the tickers have the same startDate to update from

*Function: checkdayRange()*

* check if the downloaded data and updated data have same value on the startDate

*Function: def check\_same()*

* check whether there is duplicatd data, if there is, drop duplicated data

*Function: def check\_duplicated()*

# Equity\_Vol

* 1. Step1: find min date in MarketData and max date in MarketDataUpdate

*Code: findMinDateEquiy\_Vol*

* 1. Step2: Download Equity data in Bloomberg

*Code: update\_equity\_vol.py*