<http://www.yutiansut.com:3000/topic/5f1bf1ec5778f910c1ba7318>

# coding:utf-8

#

# The MIT License (MIT)

#

# Copyright (c) 2018-2020 azai/Rgveda/GolemQuant

#

# Permission is hereby granted, free of charge, to any person obtaining a copy

# of this software and associated documentation files (the "Software"), to deal

# in the Software without restriction, including without limitation the rights

# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell

# copies of the Software, and to permit persons to whom the Software is

# furnished to do so, subject to the following conditions:

#

# The above copyright notice and this permission notice shall be included in

# all

# copies or substantial portions of the Software.

#

# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR

# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,

# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE

# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER

# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,

# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE

# SOFTWARE.

#

import datetime

import time

import numpy as np

import pandas as pd

import pymongo

try:

import QUANTAXIS as QA

from QUANTAXIS.QAUtil import (QASETTING,

DATABASE,

QA\_util\_log\_info,

QA\_util\_to\_json\_from\_pandas,)

from QUANTAXIS.QAUtil.QAParameter import ORDER\_DIRECTION

from QUANTAXIS.QAData.QADataStruct import (QA\_DataStruct\_Index\_min,

QA\_DataStruct\_Index\_day,

QA\_DataStruct\_Stock\_day,

QA\_DataStruct\_Stock\_min)

from QUANTAXIS.QAUtil.QADate\_Adv import (

QA\_util\_timestamp\_to\_str,

QA\_util\_datetime\_to\_Unix\_timestamp,

QA\_util\_print\_timestamp

)

except:

print('PLEASE run "pip install QUANTAXIS" to call these modules')

pass

try:

from GolemQ.GQUtil.parameter import (

AKA,

INDICATOR\_FIELD as FLD,

TREND\_STATUS as ST,

)

except:

class AKA():

"""

趋势状态常量，专有名称指标，定义成常量可以避免直接打字符串造成的拼写错误。

"""

# 蜡烛线指标

CODE = 'code'

NAME = 'name'

OPEN = 'open'

HIGH = 'high'

LOW = 'low'

CLOSE = 'close'

VOLUME = 'volume'

VOL = 'vol'

DATETIME = 'datetime'

LAST\_CLOSE = 'last\_close'

PRICE = 'price'

SYSTEM\_NAME = 'myQuant'

def \_\_setattr\_\_(self, name, value):

raise Exception(u'Const Class can\'t allow to change property\' value.')

return super().\_\_setattr\_\_(name, value)

class ST():

"""

趋势状态常量，专有名称指标，定义成常量可以避免直接打字符串造成的拼写错误。

"""

# 状态

POSITION\_R5 = 'POS\_R5'

TRIGGER\_R5 = 'TRG\_R5'

CANDIDATE = 'CANDIDATE'

def \_\_setattr\_\_(self, name, value):

raise Exception(u'Const Class can\'t allow to change property\' value.')

return super().\_\_setattr\_\_(name, value)

class FLD():

DATETIME = 'datetime'

ML\_FLU\_TREND = 'ML\_FLU\_TREND'

FLU\_POSITIVE = 'FLU\_POSITIVE'

FLU\_NEGATIVE = 'FLU\_NEGATIVE'

def \_\_setattr\_\_(self, name, value):

raise Exception(u'Const Class can\'t allow to change property\' value.')

return super().\_\_setattr\_\_(name, value)

def GQSignal\_util\_save\_indices\_day(code,

indices,

market\_type=QA.MARKET\_TYPE.STOCK\_CN,

portfolio='myportfolio',

ui\_log=None,

ui\_progress=None):

"""

在数据库中保存所有计算出来的股票日线指标，用于汇总评估和筛选数据——日线

save stock\_indices, state

Keyword Arguments:

client {[type]} -- [description] (default: {DATABASE})

"""

def \_check\_index(coll\_indices):

coll\_indices.create\_index([("code",

pymongo.ASCENDING),

(FLD.DATETIME,

pymongo.ASCENDING),],

unique=True)

coll\_indices.create\_index([("date",

pymongo.ASCENDING),

(ST.TRIGGER\_R5,

pymongo.ASCENDING),],)

coll\_indices.create\_index([("date",

pymongo.ASCENDING),

(ST.POSITION\_R5,

pymongo.ASCENDING),],)

coll\_indices.create\_index([('date\_stamp',

pymongo.ASCENDING),

(ST.TRIGGER\_R5,

pymongo.ASCENDING),],)

coll\_indices.create\_index([('date\_stamp',

pymongo.ASCENDING),

(ST.POSITION\_R5,

pymongo.ASCENDING),],)

coll\_indices.create\_index([("date",

pymongo.ASCENDING),

(FLD.FLU\_POSITIVE,

pymongo.ASCENDING),],)

coll\_indices.create\_index([('date\_stamp',

pymongo.ASCENDING),

(FLD.FLU\_POSITIVE,

pymongo.ASCENDING),],)

coll\_indices.create\_index([("code",

pymongo.ASCENDING),

('date\_stamp',

pymongo.ASCENDING),],

unique=True)

coll\_indices.create\_index([("code",

pymongo.ASCENDING),

("date",

pymongo.ASCENDING),],

unique=True)

coll\_indices.create\_index([("code",

pymongo.ASCENDING),

(FLD.DATETIME,

pymongo.ASCENDING),

(ST.CANDIDATE,

pymongo.ASCENDING),],

unique=True)

coll\_indices.create\_index([("code",

pymongo.ASCENDING),

('date\_stamp',

pymongo.ASCENDING),

(ST.CANDIDATE,

pymongo.ASCENDING),],

unique=True)

coll\_indices.create\_index([("code",

pymongo.ASCENDING),

("date",

pymongo.ASCENDING),

(ST.CANDIDATE,

pymongo.ASCENDING),],

unique=True)

def \_formatter\_data(indices):

frame = indices.reset\_index(1, drop=False)

# UTC时间转换为北京时间

frame['date'] = pd.to\_datetime(frame.index,).tz\_localize('Asia/Shanghai')

frame['date'] = frame['date'].dt.strftime('%Y-%m-%d')

frame['datetime'] = pd.to\_datetime(frame.index,).tz\_localize('Asia/Shanghai')

frame['datetime'] = frame['datetime'].dt.strftime('%Y-%m-%d %H:%M:%S')

# GMT+0 String 转换为 UTC Timestamp

frame['date\_stamp'] = pd.to\_datetime(frame['date']).astype(np.int64) // 10 \*\* 9

frame['created\_at'] = int(time.mktime(datetime.datetime.now().utctimetuple()))

frame = frame.tail(len(frame) - 150)

return frame

client = QASETTING.client[AKA.SYSTEM\_NAME]

# 同时写入横表和纵表，减少查询困扰

#coll\_day = client.get\_collection(

# 'indices\_{}'.format(datetime.date.today()))

try:

if (market\_type == QA.MARKET\_TYPE.STOCK\_CN):

#coll\_indices = client.stock\_cn\_indices\_day

coll\_indices = client.get\_collection('stock\_cn\_indices\_{}'.format(portfolio))

elif (market\_type == QA.MARKET\_TYPE.INDEX\_CN):

#coll\_indices = client.index\_cn\_indices\_day

coll\_indices = client.get\_collection('index\_cn\_indices\_{}'.format(portfolio))

elif (market\_type == QA.MARKET\_TYPE.FUND\_CN):

#coll\_indices = client.fund\_cn\_indices\_day

coll\_indices = client.get\_collection('fund\_cn\_indices\_{}'.format(portfolio))

elif (market\_type == QA.MARKET\_TYPE.FUTURE\_CN):

#coll\_indices = client.future\_cn\_indices\_day

coll\_indices = client.get\_collection('future\_cn\_indices\_{}'.format(portfolio))

elif (market\_type == QA.MARKET\_TYPE.CRYPTOCURRENCY):

#coll\_indices = client.cryptocurrency\_indices\_day

coll\_indices = client.get\_collection('cryptocurrency\_indices\_{}'.format(portfolio))

else:

QA\_util\_log\_info('WTF IS THIS! {} \n '.format(market\_type), ui\_log=ui\_log)

return False

except Exception as e:

QA\_util\_log\_info(e)

QA\_util\_log\_info('WTF IS THIS! \n ', ui\_log=ui\_log)

return False

\_check\_index(coll\_indices)

data = \_formatter\_data(indices)

err = []

# 查询是否新 tick

query\_id = {

"code": code,

'date\_stamp': {

'$in': data['date\_stamp'].tolist()

}

}

refcount = coll\_indices.count\_documents(query\_id)

if refcount > 0:

if (len(data) > 1):

# 删掉重复数据

coll\_indices.delete\_many(query\_id)

data = QA\_util\_to\_json\_from\_pandas(data)

coll\_indices.insert\_many(data)

else:

# 持续更新模式，更新单条记录

data.drop('created\_at', axis=1, inplace=True)

data = QA\_util\_to\_json\_from\_pandas(data)

coll\_indices.replace\_one(query\_id, data[0])

else:

# 新 tick，插入记录

data = QA\_util\_to\_json\_from\_pandas(data)

coll\_indices.insert\_many(data)

return True

def GQSignal\_util\_save\_indices\_min(code,

indices,

frequence,

market\_type=QA.MARKET\_TYPE.STOCK\_CN,

portfolio='myportfolio',

ui\_log=None,

ui\_progress=None):

"""

在数据库中保存所有计算出来的指标信息，用于汇总评估和筛选数据——分钟线

save stock\_indices, state

Keyword Arguments:

client {[type]} -- [description] (default: {DATABASE})

"""

def \_check\_index(coll\_indices):

coll\_indices.create\_index([("code",

pymongo.ASCENDING),

("type",

pymongo.ASCENDING),

(FLD.DATETIME,

pymongo.ASCENDING),],

unique=True)

coll\_indices.create\_index([("code",

pymongo.ASCENDING),

("type",

pymongo.ASCENDING),

("time\_stamp",

pymongo.ASCENDING),],

unique=True)

coll\_indices.create\_index([(FLD.DATETIME,

pymongo.ASCENDING),

("type",

pymongo.ASCENDING),

(ST.TRIGGER\_R5,

pymongo.ASCENDING),],)

coll\_indices.create\_index([(FLD.DATETIME,

pymongo.ASCENDING),

("type",

pymongo.ASCENDING),

(ST.POSITION\_R5,

pymongo.ASCENDING),],)

coll\_indices.create\_index([("type",

pymongo.ASCENDING),

("time\_stamp",

pymongo.ASCENDING),

(ST.TRIGGER\_R5,

pymongo.ASCENDING),],)

coll\_indices.create\_index([("type",

pymongo.ASCENDING),

("time\_stamp",

pymongo.ASCENDING),

(ST.POSITION\_R5,

pymongo.ASCENDING),],)

coll\_indices.create\_index([(FLD.DATETIME,

pymongo.ASCENDING),

("type",

pymongo.ASCENDING),

(FLD.FLU\_POSITIVE,

pymongo.ASCENDING),],)

coll\_indices.create\_index([("type",

pymongo.ASCENDING),

("time\_stamp",

pymongo.ASCENDING),

(FLD.FLU\_POSITIVE,

pymongo.ASCENDING),],)

coll\_indices.create\_index([("code",

pymongo.ASCENDING),

("type",

pymongo.ASCENDING),

(FLD.DATETIME,

pymongo.ASCENDING),

(ST.CANDIDATE,

pymongo.ASCENDING),],

unique=True)

coll\_indices.create\_index([("code",

pymongo.ASCENDING),

("type",

pymongo.ASCENDING),

("time\_stamp",

pymongo.ASCENDING),

(ST.CANDIDATE,

pymongo.ASCENDING),],

unique=True)

def \_formatter\_data(indices, frequence):

frame = indices.reset\_index(1, drop=False)

# UTC时间转换为北京时间

frame['date'] = pd.to\_datetime(frame.index,).tz\_localize('Asia/Shanghai')

frame['date'] = frame['date'].dt.strftime('%Y-%m-%d')

frame['datetime'] = pd.to\_datetime(frame.index,).tz\_localize('Asia/Shanghai')

frame['datetime'] = frame['datetime'].dt.strftime('%Y-%m-%d %H:%M:%S')

# GMT+0 String 转换为 UTC Timestamp

frame['time\_stamp'] = pd.to\_datetime(frame['datetime']).astype(np.int64) // 10 \*\* 9

frame['type'] = frequence

frame['created\_at'] = int(time.mktime(datetime.datetime.now().utctimetuple()))

frame = frame.tail(len(frame) - 150)

return frame

client = QASETTING.client[AKA.SYSTEM\_NAME]

# 同时写入横表和纵表，减少查询困扰

#coll\_day = client.get\_collection(

# 'indices\_{}'.format(datetime.date.today()))

try:

if (market\_type == QA.MARKET\_TYPE.STOCK\_CN):

#coll\_indices = client.stock\_cn\_indices\_min

coll\_indices = client.get\_collection('stock\_cn\_indices\_{}'.format(portfolio))

elif (market\_type == QA.MARKET\_TYPE.INDEX\_CN):

#coll\_indices = client.index\_cn\_indices\_min

coll\_indices = client.get\_collection('index\_cn\_indices\_{}'.format(portfolio))

elif (market\_type == QA.MARKET\_TYPE.FUND\_CN):

#coll\_indices = client.future\_cn\_indices\_min

coll\_indices = client.get\_collection('fund\_cn\_indices\_{}'.format(portfolio))

elif (market\_type == QA.MARKET\_TYPE.FUTURE\_CN):

#coll\_indices = client.future\_cn\_indices\_min

coll\_indices = client.get\_collection('future\_cn\_indices\_{}'.format(portfolio))

elif (market\_type == QA.MARKET\_TYPE.CRYPTOCURRENCY):

#coll\_indices = client.cryptocurrency\_indices\_min

coll\_indices = client.get\_collection('cryptocurrency\_indices\_{}'.format(portfolio))

else:

QA\_util\_log\_info('WTF IS THIS! \n ', ui\_log=ui\_log)

return False

except Exception as e:

QA\_util\_log\_info(e)

QA\_util\_log\_info('WTF IS THIS! \n ', ui\_log=ui\_log)

return False

\_check\_index(coll\_indices)

data = \_formatter\_data(indices, frequence)

err = []

# 查询是否新 tick

query\_id = {

"code": code,

'type': frequence,

"time\_stamp": {

'$in': data['time\_stamp'].tolist()

}

}

refcount = coll\_indices.count\_documents(query\_id)

if refcount > 0:

if (len(data) > 1):

# 删掉重复数据

coll\_indices.delete\_many(query\_id)

data = QA\_util\_to\_json\_from\_pandas(data)

coll\_indices.insert\_many(data)

else:

# 持续更新模式，更新单条记录

data.drop('created\_at', axis=1, inplace=True)

data = QA\_util\_to\_json\_from\_pandas(data)

coll\_indices.replace\_one(query\_id, data[0])

else:

# 新 tick，插入记录

data = QA\_util\_to\_json\_from\_pandas(data)

coll\_indices.insert\_many(data)

return True

def GQSignal\_fetch\_position\_singal\_day(start,

end,

frequence='day',

market\_type=QA.MARKET\_TYPE.STOCK\_CN,

portfolio='myportfolio',

format='numpy',

ui\_log=None,

ui\_progress=None):

"""

'获取股票指标日线'

Keyword Arguments:

client {[type]} -- [description] (default: {DATABASE})

"""

start = str(start)[0:10]

end = str(end)[0:10]

#code= [code] if isinstance(code,str) else code

client = QASETTING.client[AKA.SYSTEM\_NAME]

# 同时写入横表和纵表，减少查询困扰

#coll\_day = client.get\_collection(

# 'indices\_{}'.format(datetime.date.today()))

try:

if (market\_type == QA.MARKET\_TYPE.STOCK\_CN):

#coll\_indices = client.stock\_cn\_indices\_min

coll\_indices = client.get\_collection('stock\_cn\_indices\_{}'.format(portfolio))

elif (market\_type == QA.MARKET\_TYPE.INDEX\_CN):

#coll\_indices = client.index\_cn\_indices\_min

coll\_indices = client.get\_collection('index\_cn\_indices\_{}'.format(portfolio))

elif (market\_type == QA.MARKET\_TYPE.FUND\_CN):

#coll\_indices = client.future\_cn\_indices\_min

coll\_indices = client.get\_collection('fund\_cn\_indices\_{}'.format(portfolio))

elif (market\_type == QA.MARKET\_TYPE.FUTURE\_CN):

#coll\_indices = client.future\_cn\_indices\_min

coll\_indices = client.get\_collection('future\_cn\_indices\_{}'.format(portfolio))

elif (market\_type == QA.MARKET\_TYPE.CRYPTOCURRENCY):

#coll\_indices = client.cryptocurrency\_indices\_min

coll\_indices = client.get\_collection('cryptocurrency\_indices\_{}'.format(portfolio))

else:

QA\_util\_log\_info('WTF IS THIS! \n ', ui\_log=ui\_log)

return False

except Exception as e:

QA\_util\_log\_info(e)

QA\_util\_log\_info('WTF IS THIS! \n ', ui\_log=ui\_log)

return False

if QA\_util\_date\_valid(end):

cursor = coll\_indices.find({

ST.TRIGGER\_R5: {

'$gt': 0

},

"date\_stamp":

{

"$lte": QA\_util\_date\_stamp(end),

"$gte": QA\_util\_date\_stamp(start)

}

},

{"\_id": 0},

batch\_size=10000)

#res=[QA\_util\_dict\_remove\_key(data, '\_id') for data in cursor]

res = pd.DataFrame([item for item in cursor])

try:

res = res.assign(date=pd.to\_datetime(res.date)).drop\_duplicates((['date',

'code'])).set\_index(['date',

'code'],

drop=False)

codelist = QA.QA\_fetch\_stock\_name(res[AKA.CODE].tolist())

res['name'] = res.apply(lambda x:codelist.at[x.get(AKA.CODE), 'name'], axis=1)

except:

res = None

if format in ['P', 'p', 'pandas', 'pd']:

return res

elif format in ['json', 'dict']:

return QA\_util\_to\_json\_from\_pandas(res)

# 多种数据格式

elif format in ['n', 'N', 'numpy']:

return numpy.asarray(res)

elif format in ['list', 'l', 'L']:

return numpy.asarray(res).tolist()

else:

print("QA Error GQSignal\_fetch\_position\_singal\_day format parameter %s is none of \"P, p, pandas, pd , json, dict , n, N, numpy, list, l, L, !\" " % format)

return None

else:

QA\_util\_log\_info('QA Error GQSignal\_fetch\_position\_singal\_day data parameter start=%s end=%s is not right' % (start,

end))

def GQSignal\_fetch\_singal\_day(code,

start,

end,

frequence='day',

market\_type=QA.MARKET\_TYPE.STOCK\_CN,

portfolio='myportfolio',

format='numpy',

ui\_log=None,

ui\_progress=None):

"""

获取股票日线指标/策略信号数据

Keyword Arguments:

client {[type]} -- [description] (default: {DATABASE})

"""

start = str(start)[0:10]

end = str(end)[0:10]

#code= [code] if isinstance(code,str) else code

client = QASETTING.client[AKA.SYSTEM\_NAME]

# 同时写入横表和纵表，减少查询困扰

#coll\_day = client.get\_collection(

# 'indices\_{}'.format(datetime.date.today()))

try:

if (market\_type == QA.MARKET\_TYPE.STOCK\_CN):

#coll\_indices = client.stock\_cn\_indices\_min

coll\_indices = client.get\_collection('stock\_cn\_indices\_{}'.format(portfolio))

elif (market\_type == QA.MARKET\_TYPE.INDEX\_CN):

#coll\_indices = client.index\_cn\_indices\_min

coll\_indices = client.get\_collection('index\_cn\_indices\_{}'.format(portfolio))

elif (market\_type == QA.MARKET\_TYPE.FUND\_CN):

#coll\_indices = client.future\_cn\_indices\_min

coll\_indices = client.get\_collection('fund\_cn\_indices\_{}'.format(portfolio))

elif (market\_type == QA.MARKET\_TYPE.FUTURE\_CN):

#coll\_indices = client.future\_cn\_indices\_min

coll\_indices = client.get\_collection('future\_cn\_indices\_{}'.format(portfolio))

elif (market\_type == QA.MARKET\_TYPE.CRYPTOCURRENCY):

#coll\_indices = client.cryptocurrency\_indices\_min

coll\_indices = client.get\_collection('cryptocurrency\_indices\_{}'.format(portfolio))

else:

QA\_util\_log\_info('WTF IS THIS! \n ', ui\_log=ui\_log)

return False

except Exception as e:

QA\_util\_log\_info(e)

QA\_util\_log\_info('WTF IS THIS! \n ', ui\_log=ui\_log)

return False

# code checking

code = QA\_util\_code\_tolist(code)

if QA\_util\_date\_valid(end):

cursor = coll\_indices.find({

'code': {

'$in': code

},

"date\_stamp":

{

"$lte": QA\_util\_date\_stamp(end),

"$gte": QA\_util\_date\_stamp(start)

}

},

{"\_id": 0},

batch\_size=10000)

#res=[QA\_util\_dict\_remove\_key(data, '\_id') for data in cursor]

res = pd.DataFrame([item for item in cursor])

try:

res = res.assign(date=pd.to\_datetime(res.date)).drop\_duplicates((['date',

'code'])).set\_index(['date',

'code'], drop=False)

res.sort\_index(inplace=True)

except:

res = None

if format in ['P', 'p', 'pandas', 'pd']:

return res

elif format in ['json', 'dict']:

return QA\_util\_to\_json\_from\_pandas(res)

# 多种数据格式

elif format in ['n', 'N', 'numpy']:

return numpy.asarray(res)

elif format in ['list', 'l', 'L']:

return numpy.asarray(res).tolist()

else:

print("QA Error GQSignal\_fetch\_singal\_day format parameter %s is none of \"P, p, pandas, pd , json, dict , n, N, numpy, list, l, L, !\" " % format)

return None

else:

QA\_util\_log\_info('QA Error GQSignal\_fetch\_singal\_day data parameter start=%s end=%s is not right' % (start,

end))

``` 标记代码块