## Work Sample

Screen the value stocks using Benjamin Graham value investment method

1. Back test the value stocks in China stock market for last 10 years
2. Adjust positions 4 times per year to satisfy Benjamin Graham value investment method
3. Compare the strategy return with the benchmark (SSE Composite)

Code:

value_investing.py - /Users/tylerwang/Desktop/StudyQuant/量化编辑 12_27王展浩/value_investing.py (3.8.3)

```
from jqdata import finance
from jqdata import *
import pandas as pd
from dateutil.relativedelta import relativedelta
from datetime import datetime,timedelta
from dateutil.parser import parse
from jqfactor import winsorize_med,standardlize,neutralize

def initialize(context):
    # set trading commission
    set_order_cost(OrderCost(open_commission=0.0013,close_commission=0.0013),type='stock')
    # set SSE composite as benchmark
    set_benchmark('000001.XSHG')
    # run the script once a month, adjust positions before the market open
    run_monthly(monthly,1,time='9:00')
    set_option('use_real_price', True)
    # 3-year interest rate
    s = {'2002-02-21' : 2.52 , '2004-10-29' : 3.24, '2006-08-19' :  3.69 , '2007-03-18' : 3.96 , '2007-05-19' : 4.41 ,
    '2007-07-21' : 4.68 , '2007-08-22' : 4.95 , '2007-09-15' : 5.22 , '2007-12-21' : 5.4 , '2008-10-09' : 5.13 ,
    '2008-10-30' : 4.77 , '2008-11-27' : 3.6 , '2008-12-23' : 3.33, '2010-10-20' : 3.85 , '2010-12-26' : 4.15 ,
    '2011-02-09' : 4.5, '2011-04-06' : 4.75 , '2011-07-07' : 5.00, '2012-06-08' : 4.65 , '2012-07-06' : 4.25,
    '2014-11-22' : 4.00 , '2015-03-01' : 3.75, '2015-05-11' : 3.50, '2015-06-28' : 3.25,'2015-08-26' : 3.00,
    '2015-10-24' : 2.75}
    s = pd.Series(s)
    g.bond_yield = s[s.index<=context.current_dt.strftime('%Y-%m-%d')][-1]
    g.today = context.current_dt.strftime('%Y-%m-%d')
    # adjust positions 4 times per year
    g.month = 1
    g.period = 3
    # set the stock pool
    g.scu = get_index_stocks('000002.XSHG')

# Change the non-trading day data to previous trading day data
def shift_trading_day(date):
    tradingday = get_all_trade_days()
    date1 = datetime.date(date)
    for i in tradingday[::-1]:
        if i<=date1:
            return i


# Current reciprocal of PE ratio is greater than twice of current 3-year interest rate
def condition_a(context):
    df = get_fundamentals(query(valuation.code,valuation.pe_ratio).
    filter(valuation.code.in_(g.scu), 1/valuation.pe_ratio*100 > g.bond_yield*2),date=g.today)
    buylist = list(df['code'])
    return buylist


# Current PE ratio is lower than 40 percent of highest PE ratio in last 5 years
def condition_b(context):
    gfc = get_fundamentals_continuously(query(valuation.code,valuation.pe_ratio_lyr).
    filter(valuation.code.in_(g.scu)), end_date=g.today,count=250*5, panel=False)
    five_year_max = np.max(gfc['pe_ratio_lyr'])
    df = get_fundamentals(query(valuation.code,valuation.pe_ratio_lyr).
    filter(valuation.code.in_(g.scu), valuation.pe_ratio_lyr<five_year_max*0.4),date=g.today)
    buylist = list(df['code'])
    return buylist


# Current dividend yield is greater than 2/3 of 3-year interest rate
def condition_c(context):
    frq = finance.run_query(query(finance.STK_XR_XD.code,finance.STK_XR_XD.bonus_ratio_rmb,finance.STK_XR_XD.report_date).
    filter(finance.STK_XR_XD.code.in_(g.scu),finance.STK_XR_XD.bonus_ratio_rmb>g.bond_yield*(2/3)))
    pre_list = list(frq['code'])
    df = get_fundamentals(query(valuation.code).
    filter(valuation.code.in_(pre_list)),date=g.today)
    buylist = list(df['code'])
    return buylist


# Current market cap is lower than 2/3 of total tangible assets
def condition_d(context):
    df = get_fundamentals(query(valuation.code,valuation.market_cap,balance.total_current_assets,balance.fixed_assets,balance.total_liability).
    filter(valuation.code.in_(g.scu),valuation.market_cap < (2/3)*(balance.total_current_assets+balance.fixed_assets-balance.total_liability)),date=g.today)
```

Ln: 9 Col: 0

```
    buylist = list(df['code'])
    return buylist


# Current market cap is lower than 2/3 of (current assets - total liability)
def condition_e(context):
    df = get_fundamentals(query(valuation.code,valuation.market_cap,balance.total_current_assets,balance.total_liability).
    filter(valuation.code.in_(g.scu),valuation.market_cap < (2/3)*(balance.total_current_assets- balance.total_liability)),date=g.today)
    buylist = list(df['code'])
    return buylist


# Total liability is lower than tangible assets
def condition_f(context):
    df = get_fundamentals(query(valuation.code,balance.total_current_assets,balance.total_liability,balance.fixed_assets).
    filter(valuation.code.in_(g.scu),balance.total_liability < balance.fixed_assets+balance.total_current_assets-balance.total_liability),date=g.today)
    buylist = list(df['code'])
    return buylist


# (current assets / current liability) > 2
def condition_g(context):
    df = get_fundamentals(query(valuation.code,balance.total_current_assets, balance.total_current_liability).
    filter(valuation.code.in_(g.scu),balance.total_current_assets/balance.total_current_liability>2),date=g.today)
    buylist = list(df['code'])
    return buylist


# Total liability < current liability * 2
def condition_h(context):
    df = get_fundamentals(query(valuation.code,balance.total_current_assets, balance.total_liability).
    filter(valuation.code.in_(g.scu),balance.total_liability<(balance.total_current_assets - balance.total_liability)*2),date=g.today)
    buylist = list(df['code'])
    return buylist


# average yearly return in the past 10 years > 7%
def condition_i(context):
    stocks = get_index_stocks('000002.XSHG',date=None)
    now = get_price(stocks, start_date = shift_trading_day(context.current_dt-timedelta(days=1)), end_date = shift_trading_day(context.current_dt - timedelta(days=1)), frequency = 'daily', fields = ['close'],panel=False)
    ago = get_price(stocks, start_date = shift_trading_day(context.current_dt-timedelta(days=1)-relativedelta(years=10)), end_date = shift_trading_day(context.current_dt - timedelta(days=1)-relativedelta(years=10)), frequency = 'daily', fields = ['close'],pane
    now['bool'] = now['close']/ago['close']**(1/10)-1>0.07
    buylist = list(now.loc[now['bool'] == True, 'code'])
    return buylist


# Yearly return cannot below -5% twice in the last 10 years
def condition_j(context):
    stocks = get_index_stocks('000002.XSHG',date=None)
    pre_list=[]
    count = 0
    for j in range(10):
        now = get_price(stocks, start_date = shift_trading_day(context.current_dt-timedelta(days=1)-relativedelta(years=j)), end_date = shift_trading_day(context.current_dt- timedelta(days=1)-relativedelta(years=j)), frequency = 'daily', fields = ['close'],pan
        ago = get_price(stocks, start_date = shift_trading_day(context.current_dt-timedelta(days=1)-relativedelta(years=j+1)), end_date = shift_trading_day(context.current_dt - timedelta(days=1)-relativedelta(years=j+1)), frequency = 'daily', fields = ['close'
        now['bool'] = (now['close']-ago['close'])/ago['close']< -0.05
        pre_list.append(now)
    new_df = pd.concat(pre_list)
    fil_df = new_df.groupby(["code"])['bool'].sum().to_frame()
    index_label = fil_df[fil_df['bool']<=2].index.tolist()
    return index_label


# Winsorize, standardlize and neutralize the factors
def clean_factor(factors,date):

    factors = factors.fillna(factors.mean())

    factors = winsorize_med(factors, scale=3, inclusive=True, inf2nan=True, axis=0)

    factors = standardlize(factors, inf2nan=True, axis=0)
```

Ln: 74 Col: 129

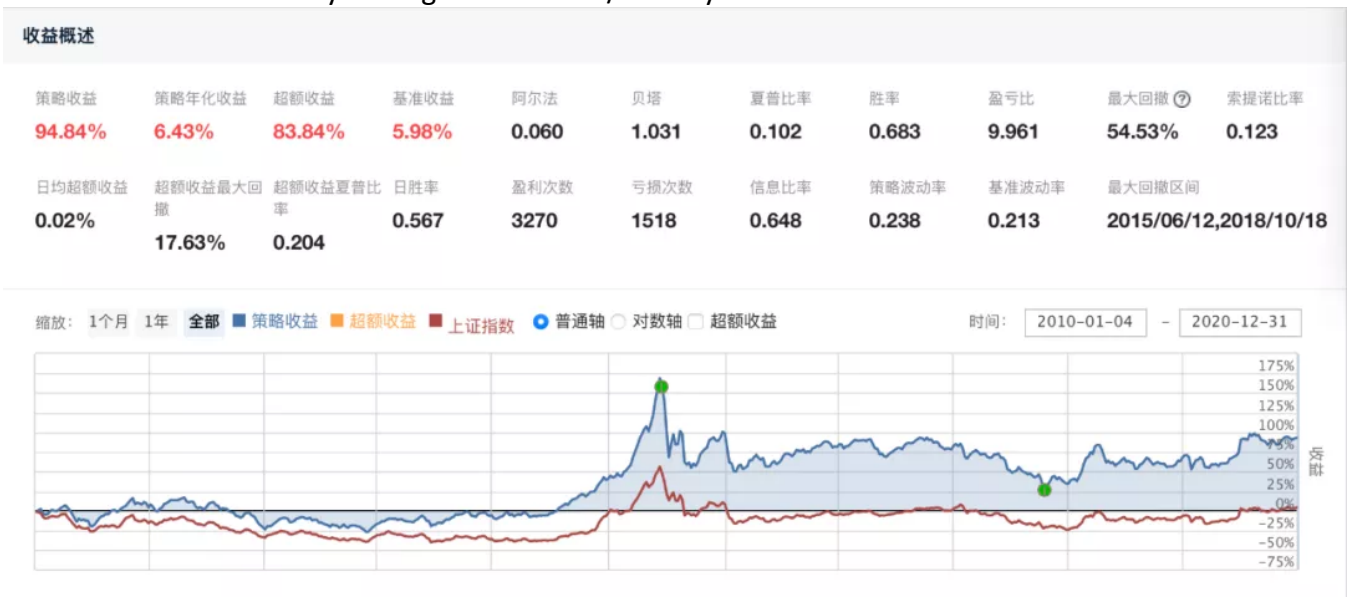
```
    factors = neutralize(factors,['sw_l1', 'pe_ratio'] ,date=str(date),axis=0)
    return factors


# Screen stocks and sort by PE ratio before Winsorize, standardlize and neutralize the factors
def preprocess_data(buylist):
    df = get_fundamentals(query(valuation.code,valuation.pe_ratio).
        filter(valuation.code.in_(buylist)),date=g.today).set_index('code')
    new_df = clean_factor(df,g.today)
    final_df = new_df.sort_values('pe_ratio',ascending = False)
    final_list = list(final_df.index.values)
    return final_list


# Trade a list of stocks
def trade(context, buylist):
    for stock in context.portfolio.positions:
        if stock not in buylist:
            order_target(stock,0)
    # Sell all positions which are not in the list and buy all positions which are in the list (split funds evenly in every chosen stock)
    position_per_stk = context.portfolio.total_value/len(buylist)
    for stock in buylist:
        order_target_value(stock,position_per_stk)
    return


def monthly(context):
    # Adjust the positions in Jan, April, July, October
    if g.month % g.period == 1:
        # buylistA = condition_a(context)
        # buylistB = condition_b(context)
        # buylistC = condition_c(context)
        # buylistD = condition_d(context)
        # buylistE = condition_e(context)
        # buylistF = condition_f(context)
        # buylistG = condition_g(context)
        # buylistH = condition_h(context)
        # buylistI = condition_i(context)
        buylistJ = condition_j(context)
        final_list = preprocess_data(buylistJ)
        trade(context,final_list)
    else:
        pass
    g.month = g.month + 1
```

Hardly a stock can satisfy all rules, so we back test each rule
Output:

A. Current reciprocal of PE ratio is greater than twice of current 3-year interest rate


B. Current PE ratio is lower than 40 percent of highest PE ratio in last 5 years

C. Current dividend yield is greater than 2/3 of 3-year interest rate




D. Current market cap is lower than 2/3 of total tangible assets

E. Current market cap is lower than 2/3 of (current assets - total liability)


F. Total liability is lower than tangible assets

G. (current assets / current liability) > 2


H. Total liability < current liability * 2

I. average yearly return in the past 10 years > 7%


J. Yearly return cannot below -5% twice in the last 10 years