多因子分析(rank-IC/IC)

> 研究2016/06/30的海通证券《因子预测能力初探——多因子模型系列报告之一》

在多因子选股模型中，IC 通常被用于衡量因子对收益率的预测能力，选取较为广泛应用的因子，利用沪深300 成分股和中证500成分股分析其Rank-IC在2014 年之后的表现，并总结各自特点。

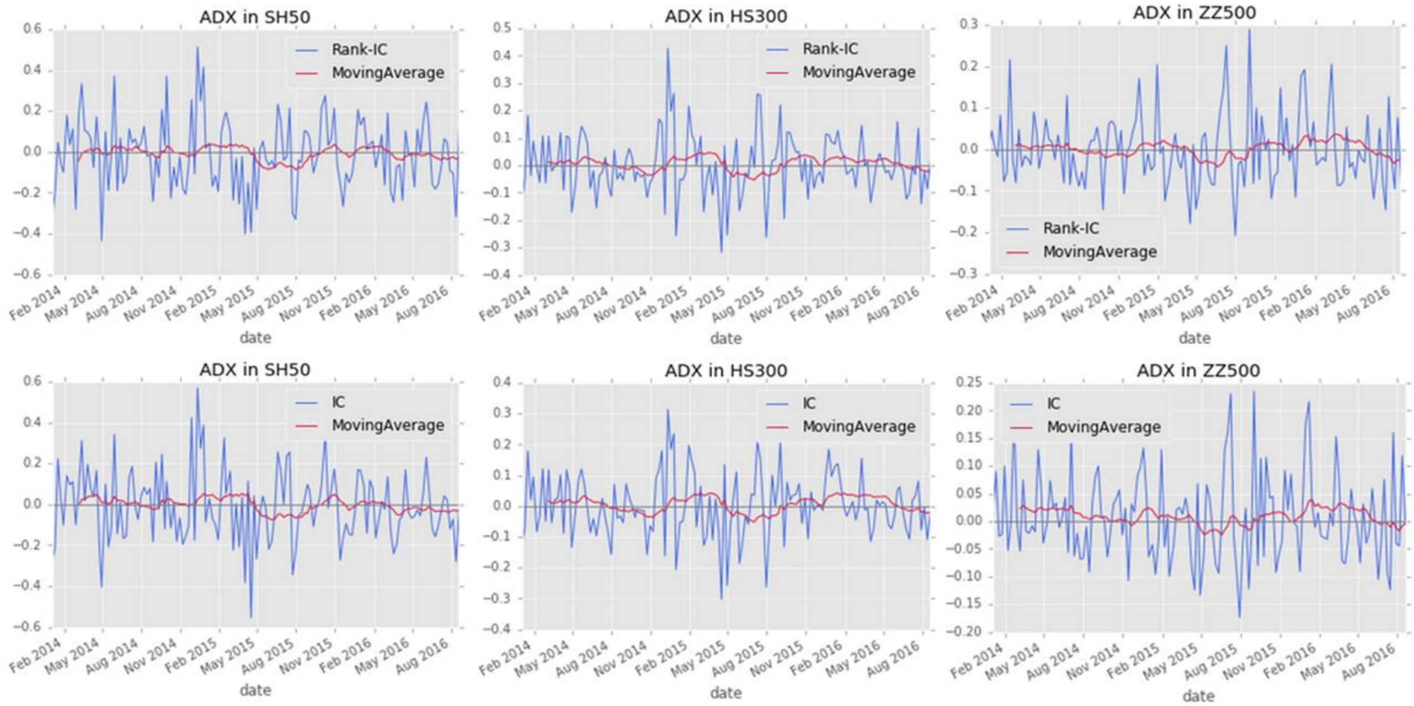
纳入考虑的因子主要有：

```
ADX 平均动向指数,趋势型因子
RSI 相对强弱指标,超买超卖型因子
VOL20 20日平均换手率,成交量型因子
MTM 动量指标,趋势型因子
ROA 资产回报率,盈利能力和收益质量类因子
ROE 权益回报率,盈利能力和收益质量类因子
PB 市净率,估值与市值类因子
PE 市盈率,估值与市值类因子
NetAssetGrowRate 净资产增长率,成长能力类因子
NetProfitGrowRate 净利润增长率,属于成长能力类因子
HBETA 历史贝塔,超买超卖型因子
marketValue 总市值
InventoryTRate 存货周转率
OperatingRevenueGrowRate 营业收入增长率,属于成长能力类因子
```

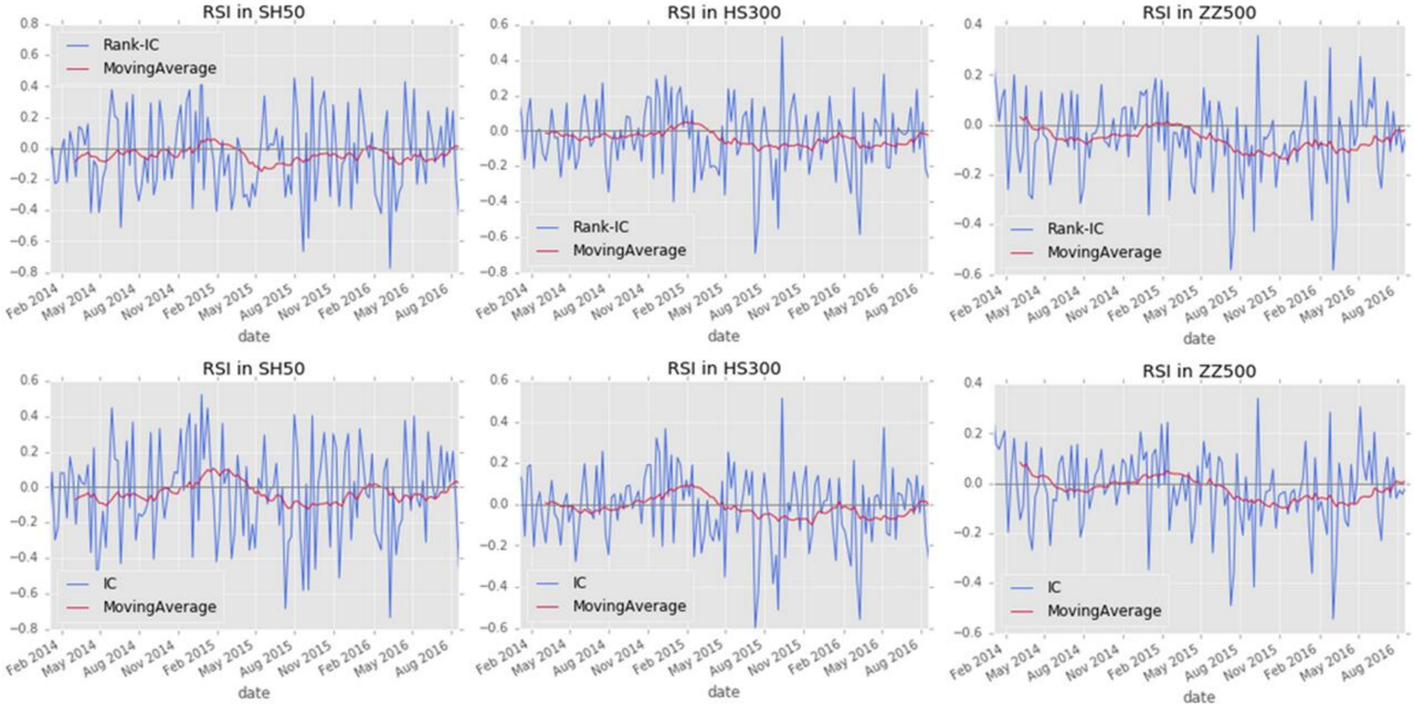
IC也就是信息系数(information coefficient)，计算的是T期的因子与T+1期return之间的相关性； rank-IC，计算的是T期的因子与T+1期return rank值之间的相关性。

分别对以上14个因子在上证50、沪深300、中证500上进行分析：

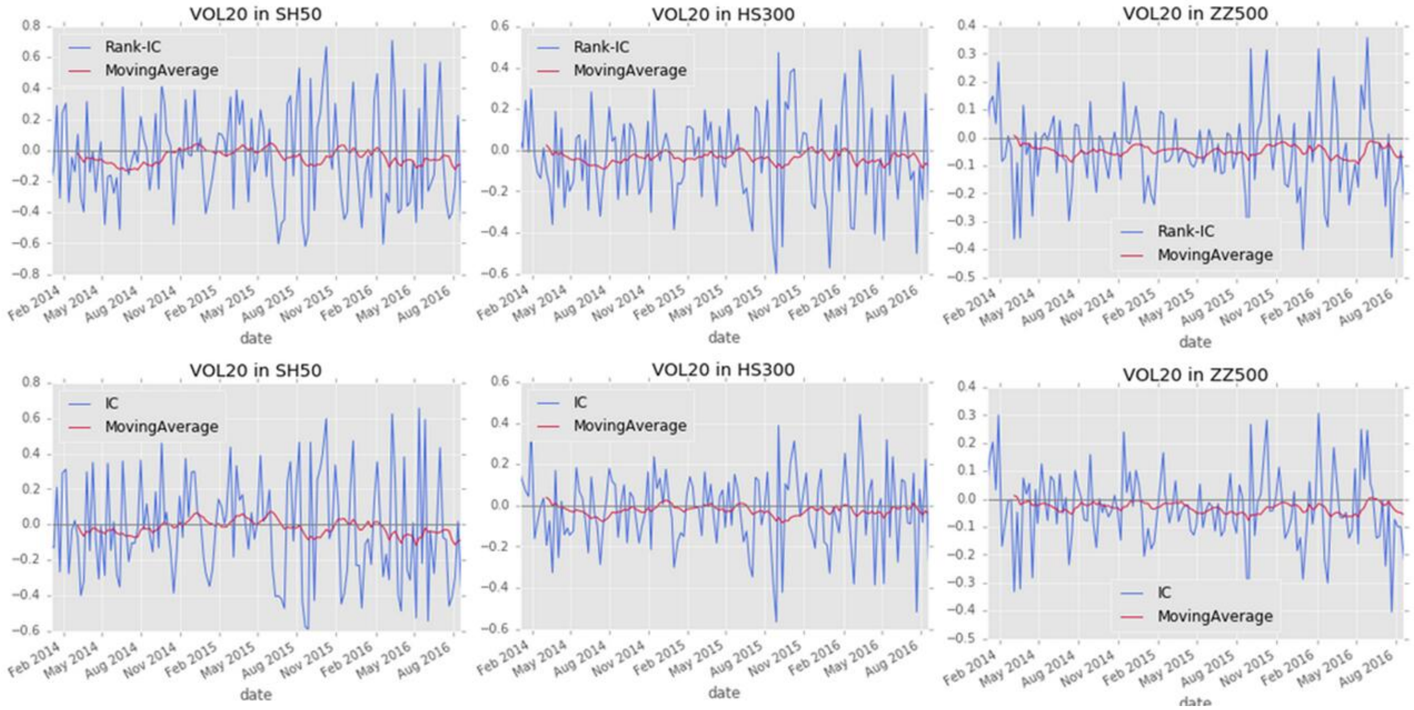
ADX



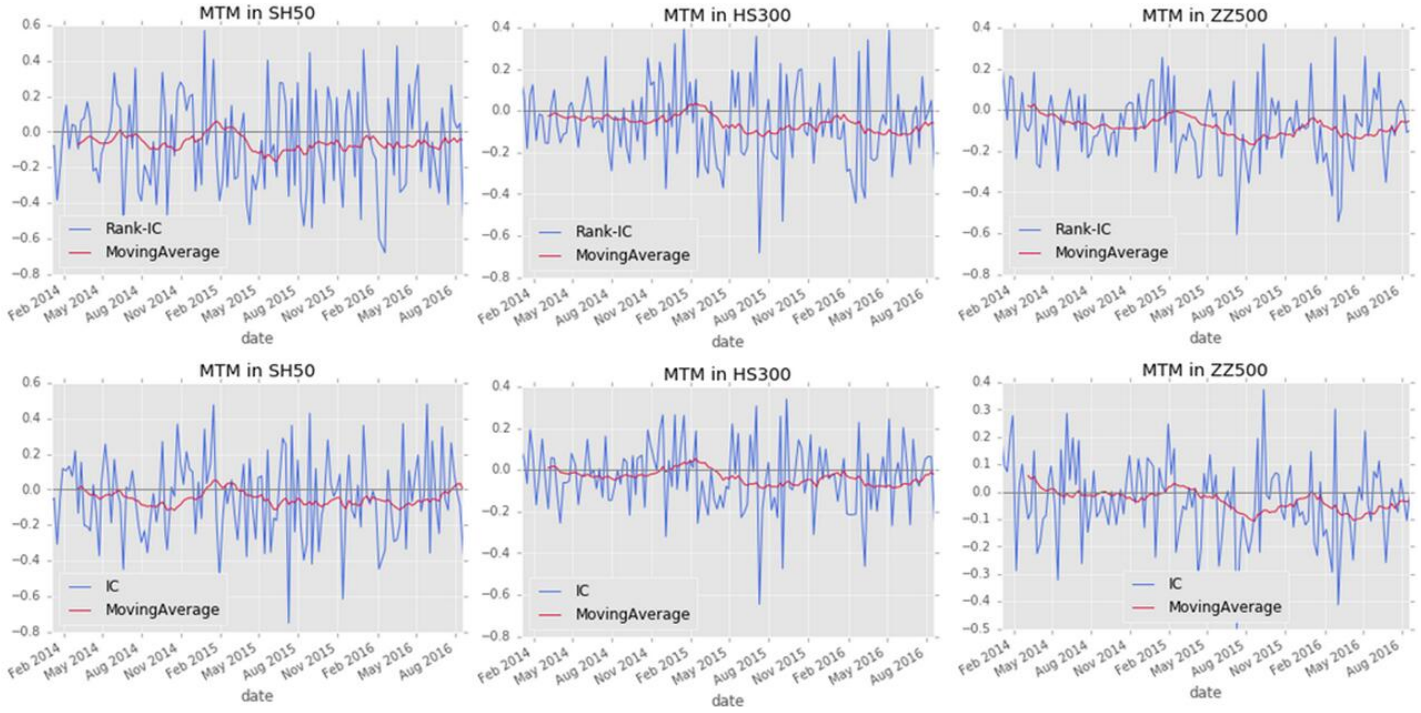
RSI


RSI in SH50
Rank-IC
MovingAverage
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
Feb 2014
May 2014
Aug 2014
Nov 2014
Feb 2015
May 2015
Aug 2015
Nov 2015
Feb 2016
May 2016
Aug 2016
date
RSI in HS300
Rank-IC
MovingAverage
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
date
RSI in ZZ500
Rank-IC
MovingAverage
0.4
0.2
0.0
−0.2
−0.4
−0.6
date
RSI in SH50
IC
MovingAverage
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
date
RSI in HS300
IC
MovingAverage
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
date
RSI in ZZ500
IC
MovingAverage
0.4
0.2
0.0
−0.2
−0.4
−0.6
date

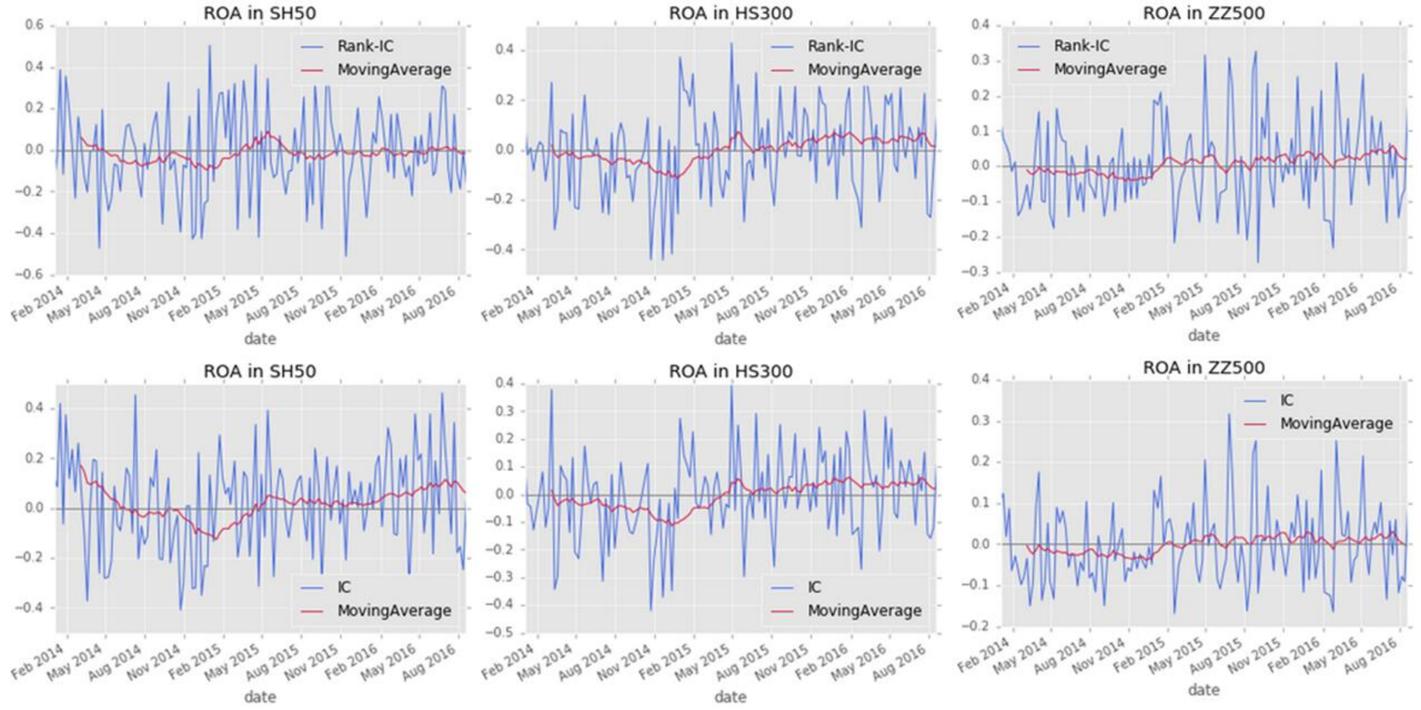
VOL20


VOL20 in SH50
Rank-IC
MovingAverage
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
Feb 2014
May 2014
Aug 2014
Nov 2014
Feb 2015
May 2015
Aug 2015
Nov 2015
Feb 2016
May 2016
Aug 2016
date
VOL20 in HS300
Rank-IC
MovingAverage
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
date
VOL20 in ZZ500
Rank-IC
MovingAverage
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
date
VOL20 in SH50
IC
MovingAverage
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
date
VOL20 in HS300
IC
MovingAverage
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
date
VOL20 in ZZ500
IC
MovingAverage
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
date

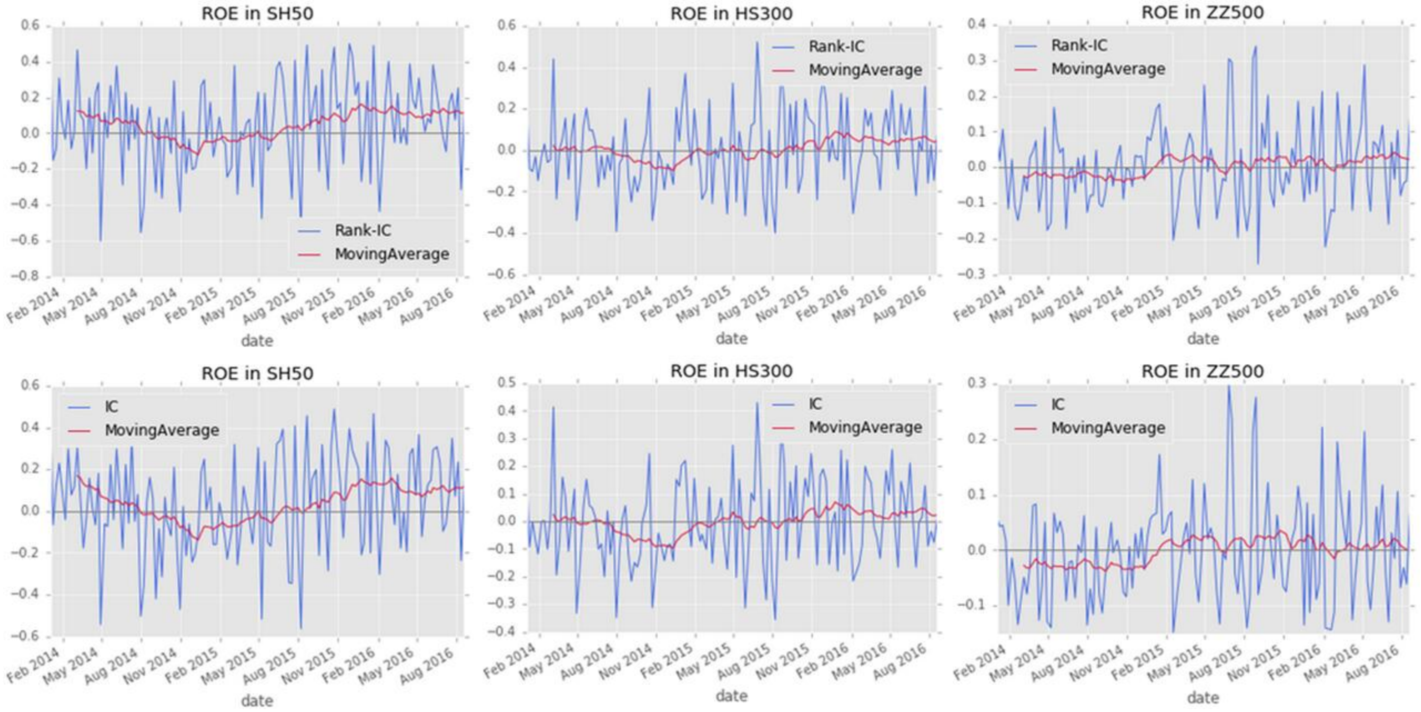
MTM


MTM in SH50
Rank-IC
MovingAverage
MTM in HS300
Rank-IC
MovingAverage
MTM in ZZ500
Rank-IC
MovingAverage
MTM in SH50
IC
MovingAverage
MTM in HS300
IC
MovingAverage
MTM in ZZ500
IC
MovingAverage
0.6
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
−0.8
Feb 2014
May 2014
Aug 2014
Nov 2014
Feb 2015
May 2015
Aug 2015
Nov 2015
Feb 2016
May 2016
Aug 2016
date

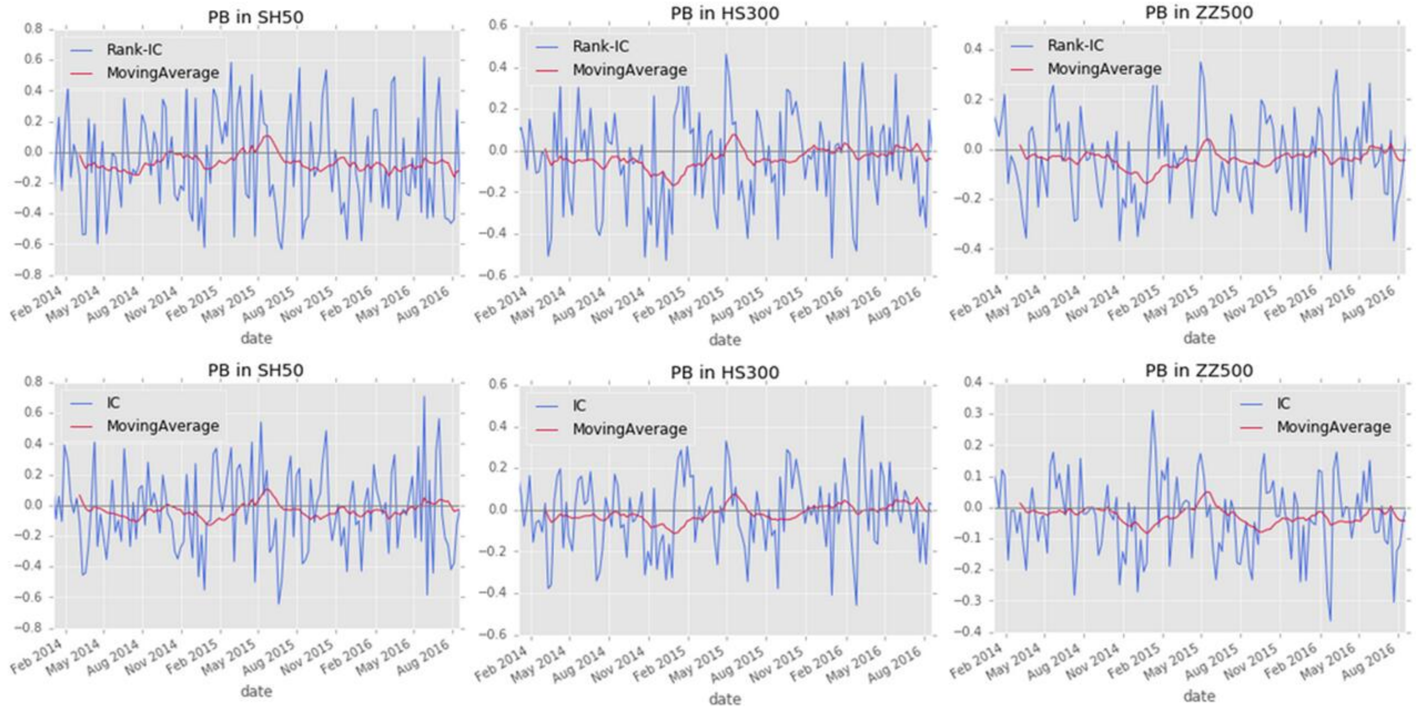
ROA


ROA in SH50
Rank-IC
MovingAverage
ROA in HS300
Rank-IC
MovingAverage
ROA in ZZ500
Rank-IC
MovingAverage
ROA in SH50
IC
MovingAverage
ROA in HS300
IC
MovingAverage
ROA in ZZ500
IC
MovingAverage
0.6
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.6
Feb 2014
May 2014
Aug 2014
Nov 2014
Feb 2015
May 2015
Aug 2015
Nov 2015
Feb 2016
May 2016
Aug 2016
date

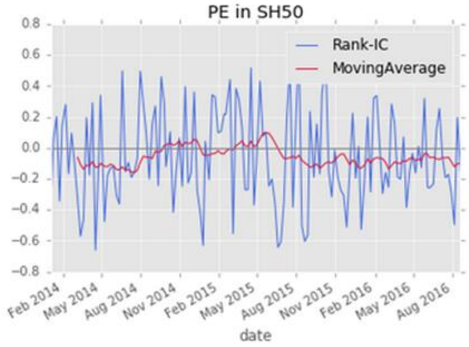
ROE

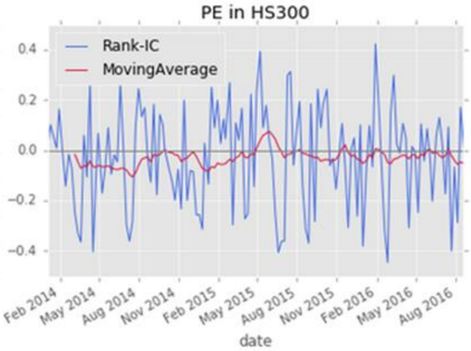

ROE in SH50
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
Rank-IC
MovingAverage
Feb 2014
May 2014
Aug 2014
Nov 2014
Feb 2015
May 2015
Aug 2015
Nov 2015
Feb 2016
May 2016
Aug 2016
date
ROE in HS300
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
Rank-IC
MovingAverage
date
ROE in ZZ500
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
Rank-IC
MovingAverage
date
ROE in SH50
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
IC
MovingAverage
date
ROE in HS300
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
IC
MovingAverage
date
ROE in ZZ500
0.3
0.2
0.1
0.0
−0.1
IC
MovingAverage
date

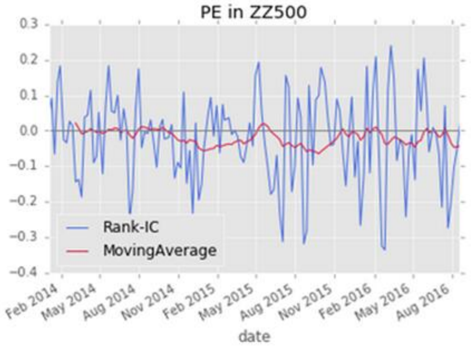
PB

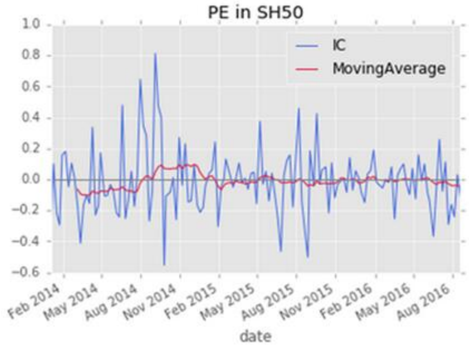

PB in SH50
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
Rank-IC
MovingAverage
Feb 2014
May 2014
Aug 2014
Nov 2014
Feb 2015
May 2015
Aug 2015
Nov 2015
Feb 2016
May 2016
Aug 2016
date
PB in HS300
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
Rank-IC
MovingAverage
date
PB in ZZ500
0.4
0.2
0.0
−0.2
−0.4
Rank-IC
MovingAverage
date
PB in SH50
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
IC
MovingAverage
date
PB in HS300
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
IC
MovingAverage
date
PB in ZZ500
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
IC
MovingAverage
date

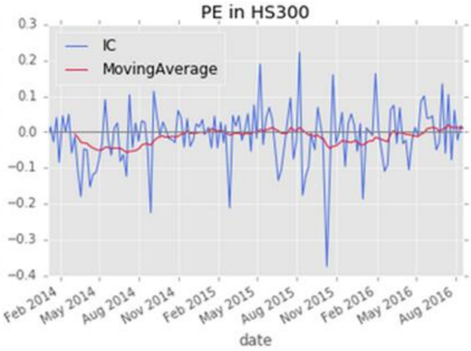
PE

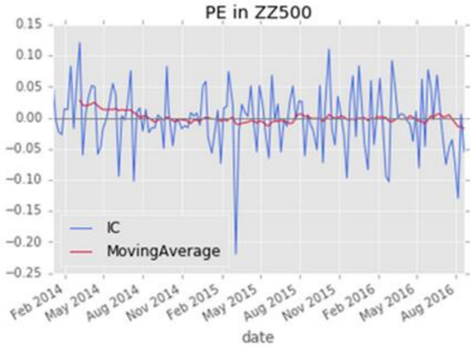




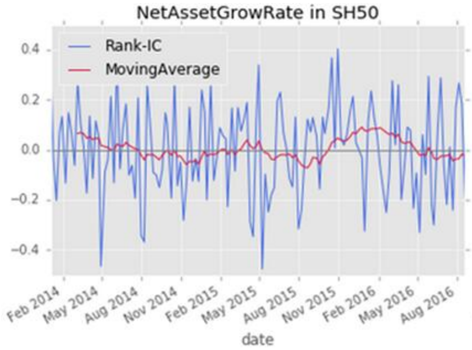


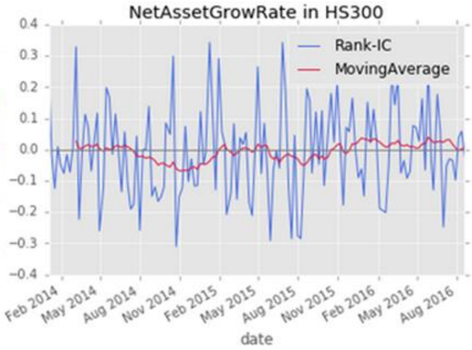


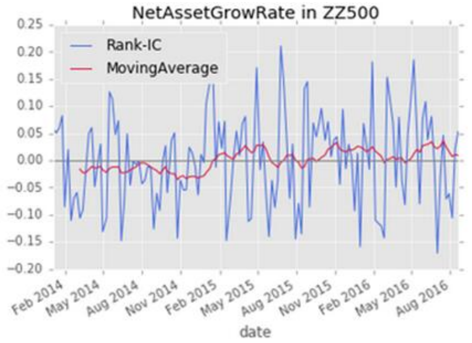


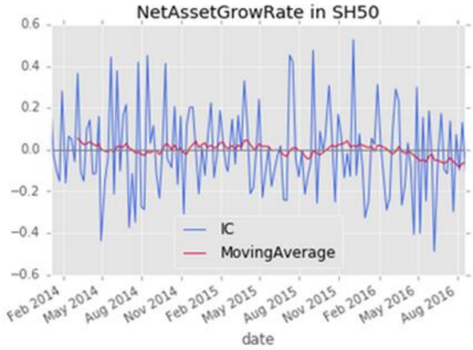


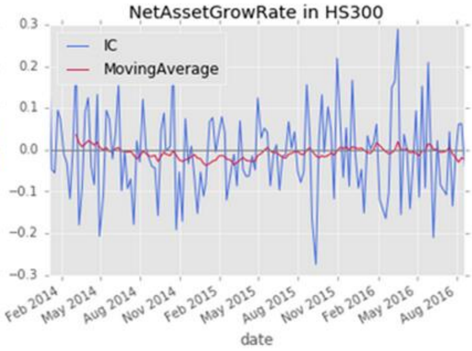
NetAssetGrowRate

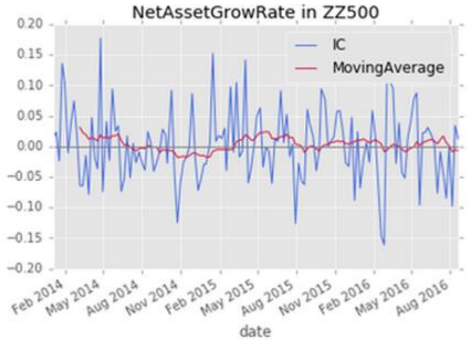




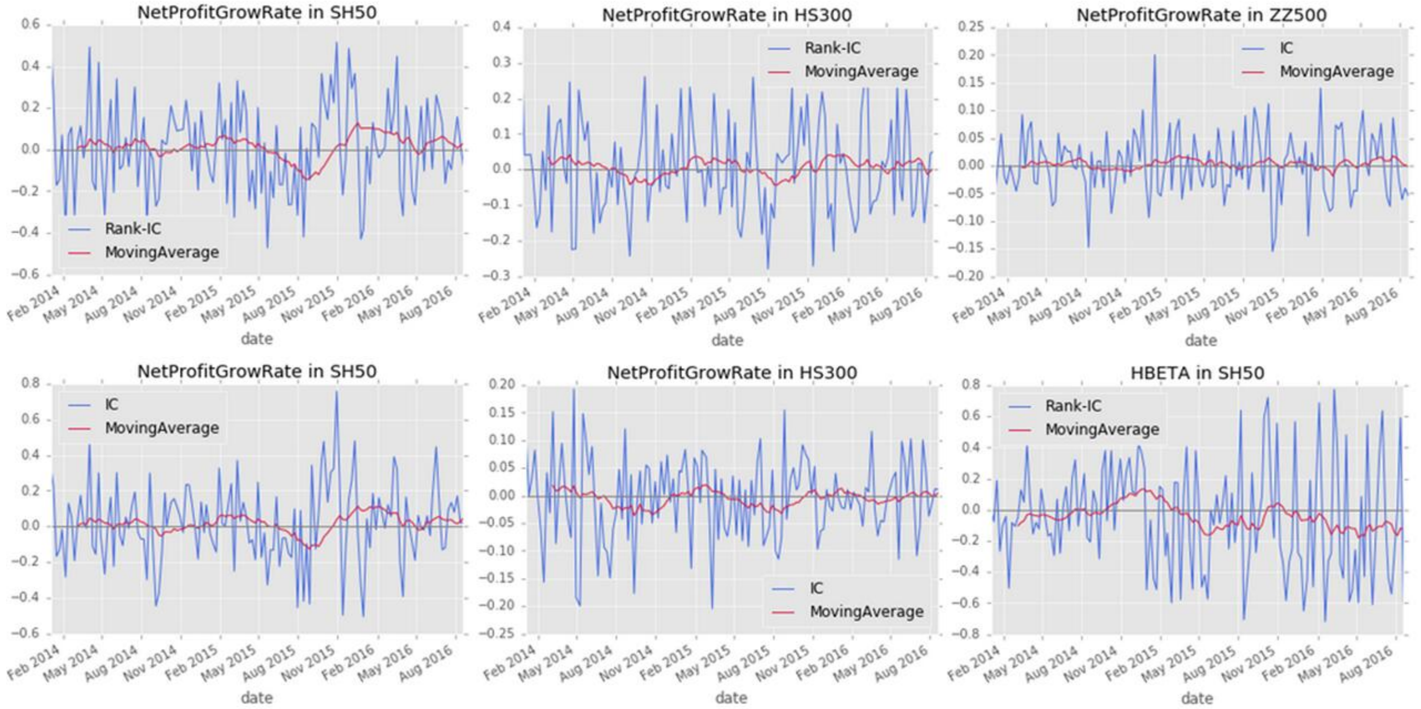




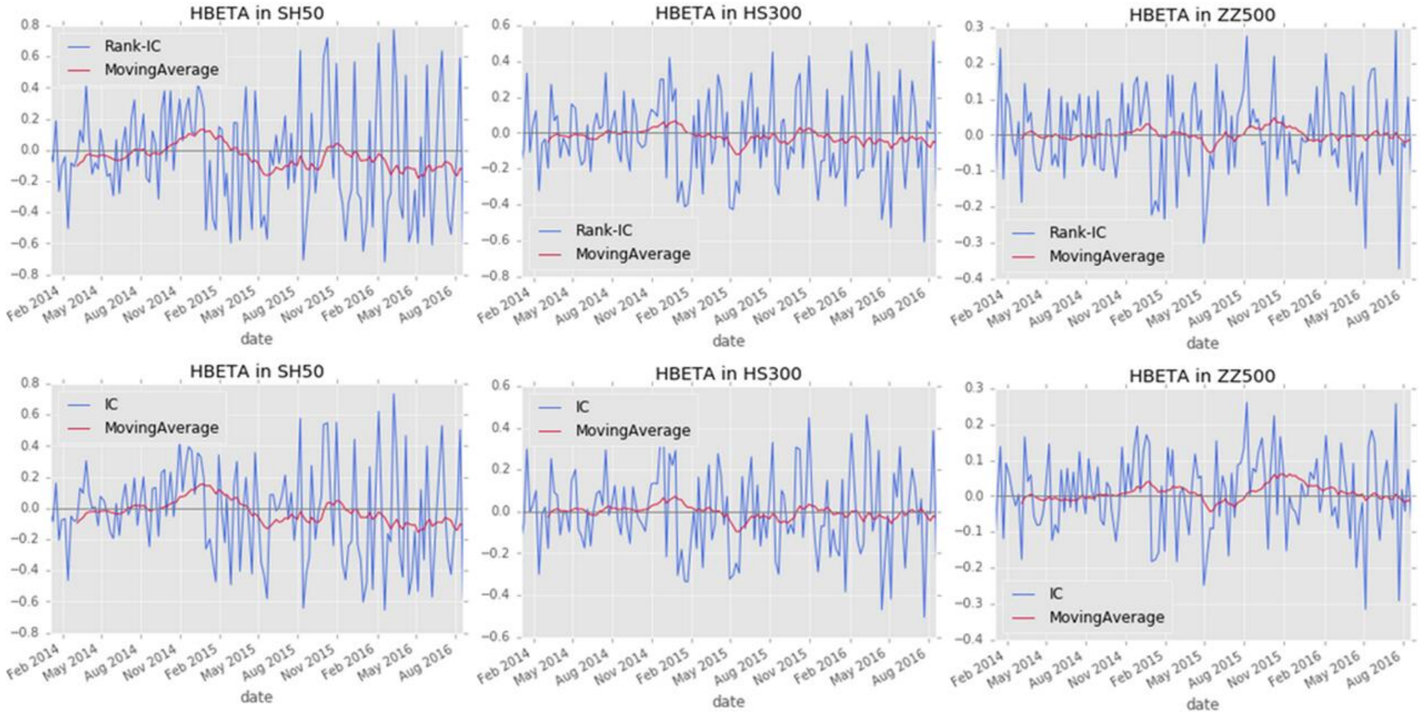




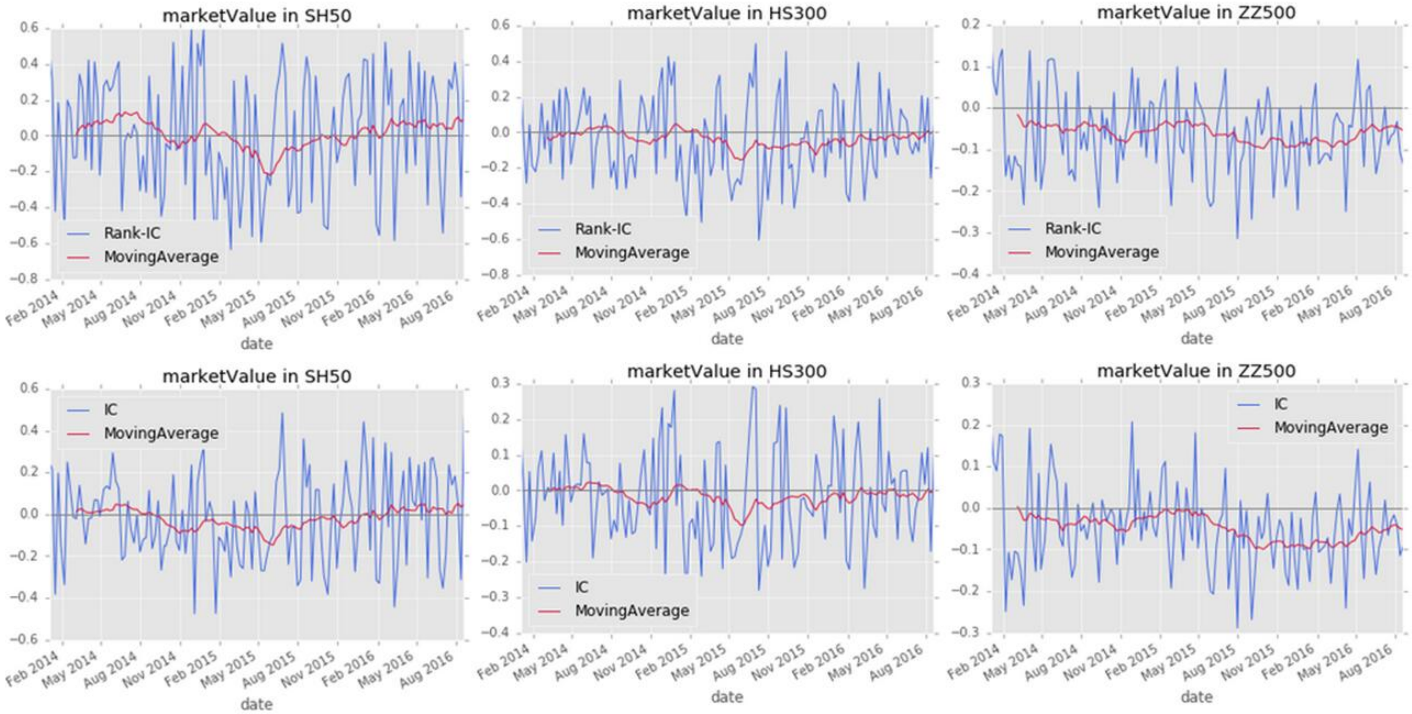
NetProfitGrowRate



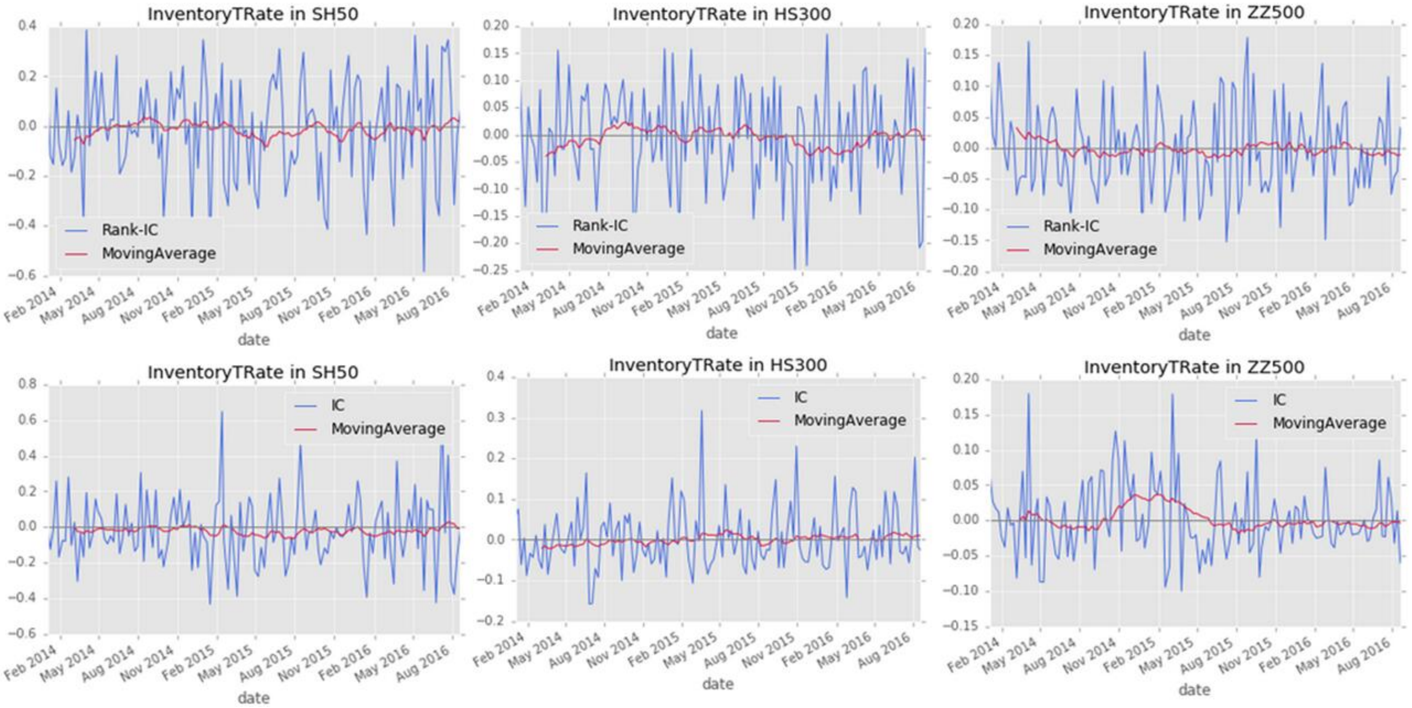
HBETA



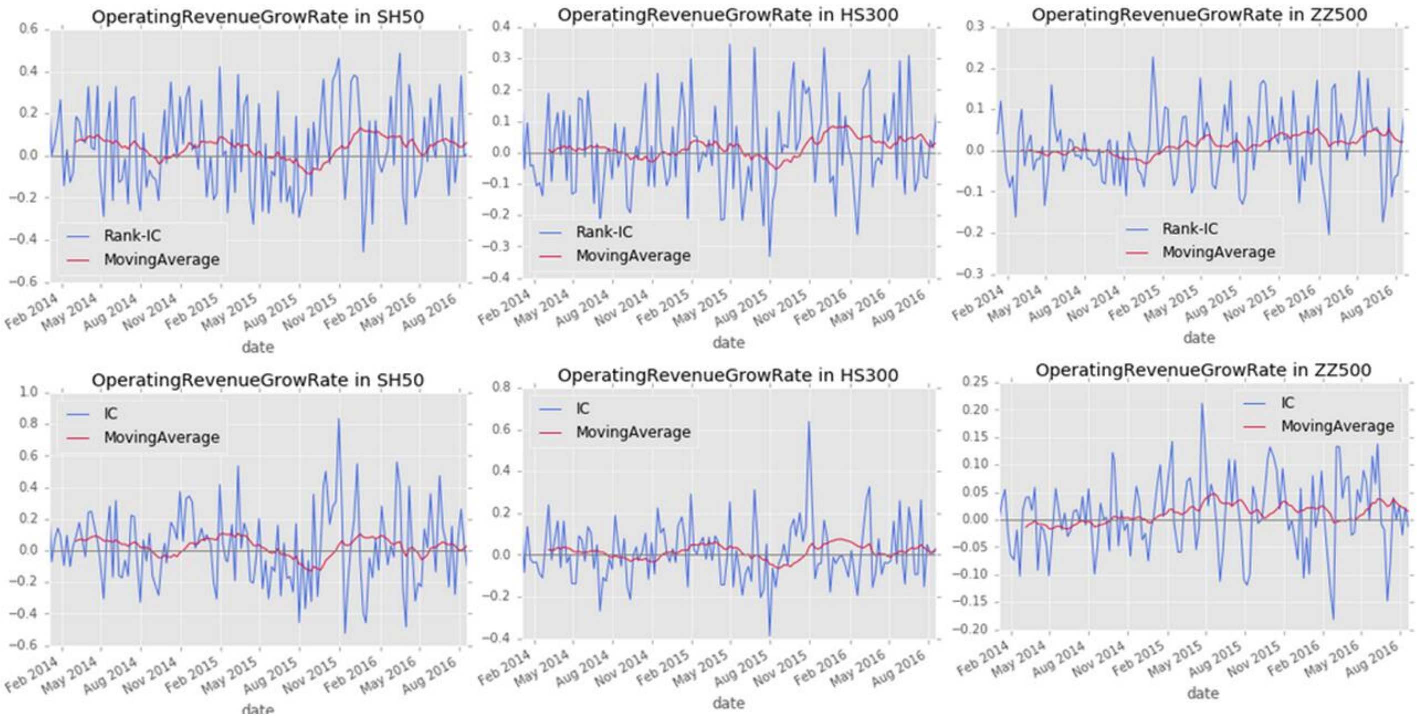
marketValue



InventoryTRate



OperatingRevenueGrowRate



总结

比较明显的负向相关的有：*RSI*、VOL20、*MTM*、*PE*、*marketValue*

比较明显的正向相关的有：*ROA*、ROE、NetAssetGrowRate、NetProfitGrowRate、*OperatingRevenueGrowRate*

```
# coding:-*- utf-8 -*-
import scipy.stats as st
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import datetime
import numpy as np
from pandas import Series, DataFrame
from quartz.api import *

DataAPI.settings.cache_enabled = False

IDXMAP = {
    'SH50': '000016',
    'HS300': '000300',
    'ZZ500': '000905',
}

IDXMAP_REVERSE = {
    '000016': 'SH50',
    '000300': 'HS300',
    '000905': 'ZZ500',
}

IDX = ['000016', '000300', '000905']

FACTORS_NAME = [
    "ADX",# 平均动向指数,趋势型因子
    "RSI",# 相对强弱指标,超买超卖型因子
    "VOL20",# 20日平均换手率,成交量型因子
    "MTM",# 动量指标,趋势型因子
    "ROA",# 资产回报率,盈利能力和收益质量类因子
    "ROE",# 权益回报率,盈利能力和收益质量类因子
    "PB",# 市净率,估值与市值类因子
    "PE",# 市盈率,估值与市值类因子
    "NetAssetGrowRate",# 净资产增长率,成长能力类因子
    "NetProfitGrowRate",# 净利润增长率,属于成长能力类因子
    "HBETA",# 历史贝塔,超买超卖型因子
    "marketValue",# 总市值
    "InventoryTRate", #存货周转率
    "OperatingRevenueGrowRate",# 营业收入增长率,属于成长能力类因子
]

class MutiFactorsSelect(object):
    def __init__(self, begin_day="20140101", end_day="20160831"):
        self.begin_day = begin_day
        self.end_day = end_day

    @property
    def day_list(self):
        '''
        获取当周最后交易日list
        '''
        begin_day = self.begin_day
        end_day = self.end_day
        cal_dates = DataAPI.TradeCalGet(exchangeCD=u"XSHG", beginDate=begin_day, endDate=end_day, field="calendarDate,isWeekEnd")
        trading_days = cal_dates[cal_dates["isWeekEnd"]==1]["calendarDate"].tolist()
        return [day.replace("-","") for day in trading_days]

    def factors_one_day_get(self, factor_name, tradeDate, ticker):
        '''
        多只股票某天单因子数据
        '''
        factor_api_field = "secID," + factor_name
        # 获取当日成分股
        cons_id_df = DataAPI.IdxConsGet(secID=u"",ticker=ticker,intoDate=tradeDate,isNew=u"",field=u"consID",pandas="1")
        cons_id_ls = cons_id_df["consID"].tolist()
        cons_id_str = ",".join(cons_id_ls)
        if factor_name in ["ADX","RSI","VOL20","MTM","ROA","ROE","PB","PE","NetAssetGrowRate","NetProfitGrowRate","HBETA","Invento
            return DataAPI.MktStockFactorsOneDayGet(tradeDate=tradeDate,secID=cons_id_str,ticker=u"",field=factor_api_field,pandas
        elif factor_name in ["marketValue"]:
            return DataAPI.MktEqudGet(tradeDate=tradeDate,secID=cons_id_str,ticker=u"",field=factor_api_field,pandas="1")
        else:
            return DataFrame()

    def rank_ic(self, factor_name, ticker="000300"):
        '''
        计算rank-ic并且画图
        @factor_name为因子名,如"RSI"
        @ticker为指数代码
        '''
        trading_days = self.day_list
        rank_ic_ls = []
        ic_ls = []

        for i in range(len(trading_days)-1):
            # 获取当日成分股
            cons_id_df = DataAPI.IdxConsGet(secID=u"",ticker=ticker,intoDate=trading_days[i],isNew=u"",field=u"consID",pandas="1")
            cons_id_ls = cons_id_df["consID"].tolist()
            cons_id_str = ",".join(cons_id_ls)
            #获取每周最后一个交易日的因子值
            factor_df = self.factors_one_day_get(factor_name, trading_days[i], ticker)
            # 获取相应股票未来一周的收益
            weekly_return = DataAPI.MktEquwAdjGet(secID=cons_id_str,beginDate=trading_days[i+1],endDate=trading_days[i+1],field=u"
            factor_return_df = factor_df.merge(weekly_return,on='secID', how="inner")
            factor_return_df[factor_return_df["return"]==0] = None
            factor_return_df.dropna(inplace=True)
            rank_ic, rank_ic_p_value = st.pearsonr(factor_return_df[factor_name].rank(),factor_return_df["return"].rank())

            rank_ic_temp_dict = {}
            rank_ic_temp_dict["date"] = trading_days[i]
            rank_ic_temp_dict["Rank-IC"] = rank_ic
            rank_ic_ls.append(rank_ic_temp_dict)

            ic, ic_p_value = st.pearsonr(factor_return_df[factor_name],factor_return_df["return"])

            ic_temp_dict = {}
            ic_temp_dict["date"] = trading_days[i]
            ic_temp_dict["IC"] = ic
            ic_ls.append(ic_temp_dict)

        rank_ic_result_df = pd.DataFrame(rank_ic_ls)
        rank_ic_result_df["MovingAverage"] = pd.rolling_mean(rank_ic_result_df["Rank-IC"], window=20,min_periods=10)

        ic_result_df = pd.DataFrame(ic_ls)
        ic_result_df["MovingAverage"] = pd.rolling_mean(ic_result_df["IC"], window=20,min_periods=10)
        # plot
        matplotlib.style.use('ggplot')
        rank_ic_result_df["date"] = rank_ic_result_df["date"].apply(str)
        rank_ic_result_df.index = pd.to_datetime(rank_ic_result_df["date"])
        rank_ic_result_df.plot(colors=['royalblue', 'crimson']).set_title(factor_name+" in "+IDXMAP_REVERSE[ticker])
        plt.axhline(0, color='Gray')

        ic_result_df["date"] = ic_result_df["date"].apply(str)
        ic_result_df.index = pd.to_datetime(ic_result_df["date"])
        ic_result_df.plot(colors=['royalblue', 'crimson']).set_title(factor_name+" in "+IDXMAP_REVERSE[ticker])
        plt.axhline(0, color='Gray')


if __name__ == '__main__':
    for i in range(len(FACTORS_NAME)):
        for j in range(len(IDX)):
            MutiFactorsSelect().rank_ic(FACTORS_NAME[i],IDX[j])
```