**pure系列选股因子回测框架**

**目录**

[一、 框架组成部分 1](#_Toc21478)

[（一）原始父类——灵活框架 1](#_Toc16428)

[1. pure\_moon回测框架 1](#_Toc14831)

[2. pure\_fall因子计算框架 2](#_Toc16148)

[3. pure\_winter剔除barra因子框架 4](#_Toc3529)

[（二） 继承子类——便捷框架 5](#_Toc9965)

[1. pure\_moonnight封装月度回测框架 5](#_Toc11430)

[2. pure\_moonlight常用辅助因子提取器 5](#_Toc17722)

[3. pure\_fallmount因子加工框架 6](#_Toc26290)

[4. pure\_snowtrain封装剔除barra因子框架 6](#_Toc18277)

[二、 框架使用流程 6](#_Toc492)

[三、 使用示例（以复现cpv价量相关性因子研报为例） 7](#_Toc30882)

1. **框架组成部分**

（一）**原始父类——灵活框架**

1. pure\_moon回测框架
2. pure\_moon为月度回测框架，使用时分为三步：
3. 声明一个pure\_moon对象。
4. 向对象中添加因子数据。

因子数据包含3种：

①self.set\_factor\_file设置因子文件路径。因子文件应为feather文件，为宽数据形式，’date’和股票代码为列名，时间在date列下。

②self.set\_factor\_df\_date\_as\_index通过传入dataframe来设置因子。其中dataframe的列名应为股票代码，索引应为时间。

③self.set\_factor\_df\_wide通过传入dataframe来设置因子。其中dataframe的列名应为’date’和股票代码，时间应在date列下。

1. 通过self.run()运行回测。

run的参数包括以下内容：

①groups\_num，要划分的组数，通常为5或者10。

②neutralize，是否要对输入的因子进行市值中性化。默认为False，使用时要注意在构建因子时是否已经进行了市值中性化。

③by10，默认采用的判断每月交易状态的方法是，比较每月正常是否多于非正常的天数；如果使用by10方法，则判断依据变为每月整天天数是否达到10天。默认为False，此方法非常慢，使用时要慎重。

④y2，使用matplotlib画图时，是否要对多空组合使用双y轴。默认为False。

⑤plt，回测后，是否使用matplotlib画出五分组及多空对冲净值走势图。默认为True。

⑥plotly，回测后，是否使用plotly画出五分组及多空对冲净值走势图。默认为False。

⑦filename，画图保存时的文件名，请勿加后缀。默认为“分组净值图”，如果通过self.set\_factor\_file方法来设置因子文件路径，则默认为因子文件的名字。

⑧time\_start，回测起始时间，已废弃，请勿使用。

⑨time\_end，回测终止时间，已废弃，请勿使用。

⑩print\_comments，输出多空对冲的结果评价。默认为True。

1. 直接调用pure\_moon对象，会返回宽数据形式的因子值。如：x=pure\_moon()，x()即返回因子值。注意：run之后的因子值，其索引时间都是下一期的。（与之相应的，在barra因子框架中，会先对输入的因子表的索引时间向前调整一期。用作barra以外的其他用途时，请小心。）
2. pure\_moon框架使用示例：

shen=pure\_moon()

shen.set\_factor\_file('因子数据/换手率因子turn20.feather')

shen.run(5,neutralize=True,by10=False)

1. pure\_fall因子计算框架
2. pure\_fall为用分钟数据计算月度因子的框架，不能直接使用，针对不同的课题、不同的因子构造方法，灵活定制自己的方案，具体而言，包含以下四个步骤：
3. 声明一个pure\_fall的子类，如class cpv\_corr(pure\_fall):。
4. 在子类中定义将分钟数据转化为日度数据的方法函数。
5. 在子类中定义将日度数据转化为月度数据的方法函数。
6. 调用self.run()计算因子。

run的参数包含以下内容：

①daily\_func，将分钟数据生成日因子的函数。

②monthly\_func，将日因子生成月因子的函数。

③neutralize，是否要对月因子做市值中性化。默认为False。

④hs300，是否限制股票池为沪深300成分股。默认为False。

⑤zz500，是否限制股票池为中证500成分股。默认为False。

请注意：hs300和zz500不能同时为True。（同时为True则表示二者成分股的交集部分）

1. 调用子类的对象，将返回月度因子。如：x=cpv\_corr()，x()则返回月度因子值，是一个dataframe的形式。（注意，不是pure\_fallmount对象）
2. 重写加法，将几个pure\_fall或pure\_fall的子类相加，如x=cpv\_corr()，y=cpv\_corr()，x+(y,)则会将几个因子做横截面标准化之后相加得到z（是一个dataframe）。返回一个新的月度因子值为z的pure\_fall对象。注意，加号后面的内容务必写成元组形式，如果只有两个对象相加，则x+(y,)，如果三个对象相加，则x+(y,z,)……依此类推……
3. 重写减法，将一个主因子，剔除一些相关因子，采取回归取残差的办法。形如x-(y,)或x-(y,z,)，返回一个月度因子值为残差矩阵的pure\_fall对象。
4. 子类的初始化参数：子类的初始化参数务必包含daily\_path和monthly\_path这两个参数。因为父类中的daily\_path和monthly\_path的路径不完整，因此务必在子类中补全路径，以便于保存因子数据。
5. 使用示例：

class cpv\_corr(pure\_fall):

def \_\_init\_\_(self,daily\_path='',monthly\_path=''):

'''继承分钟因子的共同特性'''

super(cpv\_corr,self).\_\_init\_\_()

self.daily\_factors\_path=self.daily\_factors\_path+daily\_path

self.monthly\_factors\_path=self.monthly\_factors\_path+monthly\_path

def func\_cpv\_corr(self,df):

'''计算每一组（一天即为一组）

在每一组内将分钟数据化为日频因子数据'''

df=df[['close','amount']]

return df.corr().iloc[0,1]

def func\_daily\_to\_monthly\_mean(self,df):

'''将日度因子转化为月度因子

此处即为每月月底算前20个交易日的相关系数的均值'''

df=df.rolling(20).mean()

df=df.resample('M').last()

return df

def func\_daily\_to\_monthly\_std(self,df):

'''将日度因子转化为月度因子

此处即为每月月底算前20个交易日的相关系数的便准查'''

df=df.rolling(20).apply(np.std)

df=df.resample('M').last()

return df

def trend\_in\_group(self,l):

'''对天数次序，做无截距项的回归，取回归系数'''

y=l.to\_numpy()

x=np.arange(1,21)

result=sm.OLS(y,x).fit()

return result.params[0]

def func\_daily\_to\_monthly\_trend(self,df):

'''保留每一天的趋势系数'''

tqdm.tqdm.pandas(desc='稍等')

df=df.rolling(20).progress\_apply(self.trend\_in\_group)

df=df.resample('M').last()

return df

cpv\_corr\_mean=cpv\_corr(daily\_path='价量相关性.feather',monthly\_path='价量相关性均值.feather')

cpv\_corr\_mean.run(cpv\_corr\_mean.func\_cpv\_corr,cpv\_corr\_mean.func\_daily\_to\_monthly\_mean,neutralize=True)

cpv\_corr\_std=cpv\_corr(daily\_path='价量相关性.feather',monthly\_path='价量相关性标准差.feather')

cpv\_corr\_std.run(cpv\_corr\_std.func\_cpv\_corr,cpv\_corr\_std.func\_daily\_to\_monthly\_std,neutralize=True)

cpv\_corr\_twins=cpv\_corr\_mean+cpv\_corr\_std

cpv\_corr\_here\_be=cpv\_corr\_twins[(cpv\_corr\_twins.index>=pd.Timestamp('2014-01-01'))&(cpv\_corr\_twins.index<=pd.Timestamp('2020-01-31'))]

cpv\_corr\_unde\_be=cpv\_corr()

cpv\_corr\_unde\_be.monthly\_factors=cpv\_corr\_here\_be

cpv\_corr\_de\_be=cpv\_corr\_unde\_be-(to\_deret20,)

1. pure\_winter剔除barra因子框架
2. pure\_winter是用于对因子剔除barra因子的框架。通过对10个barra因子值和28个申万一级行业哑变量回归取残差的方法，来获得纯净因子值。具体而言，包含以下三个步骤：
3. 声明一个pure\_winter对象。
4. 输入因子数据。输入因子数据有两种方式：

①通过self.set\_factors\_df传入长数据形式的因子数据。数据格式应为3列，第一列为时间，第二列为股票代码，第三列为因子值。

②通过self.set\_factors\_df\_wide传入宽数据形式的因子数据。数据格式应为，列名是股票代码，索引是时间。

1. 通过self.run()执行纯净化操作。
2. 通过调用对象，获取纯净因子值。如：x=pure\_winter()，x()即返回纯净化之后的因子。如果x(1)则返回一个月度因子值为纯净化因子的pure\_fallmount对象。
3. 使用示例：

snow=pure\_winter()

snow.set\_factors\_df(fac)

snow.run()

1. **继承子类——便捷框架**
2. pure\_moonnight封装月度回测框架
3. pure\_moonnight是对pure\_moon月度回测框架的封装。使用时只需一个步骤：
4. 生命一个pure\_moonnight对象，并在声明时指定因子数据，回测分组数目等。因子变量可以为一个因子数据文件路径，也可以为一个宽数据dataframe，此dataframe应为索引是时间，列名是股票代码形式。
5. 调用时，直接调用对象。如：x=pure\_moonnight(fac,5)。将返回因子值dataframe（宽数据形式），与其父类pure\_moon一样，返回的因子数据的时间列是延后了一期的。
6. 其余初始化参数与pure\_moon类的run函数的参数含义相同。
7. 使用示例：

shen\_be=pure\_moonnight(cpv\_corr\_de\_be,5,filename='价量相关性因子5分组')

1. pure\_moonlight常用辅助因子提取器
2. pure\_moonlight是继承自pure\_moon类的子类，但功能并非为回测。其用于自动计算月度因子，并对现有的月度因子进行更新存储。具体而言读入和更新数据只需要一个步骤：
3. 声明一个pure\_moonlight对象。（更新函数在初始化函数中，因子会在声明时自动更新）
4. 调用时，在对象的括号中，加入要调用的因子的简称，即可调用。如：x=pure\_moonlight()，x(‘ret20’)。目前支持的因子包括月底对数市值（cap\_as\_factor）20日传统反转因子（ret20），20日传统换手率因子（turn20），20日传统波动率因子（vol20）。

（3）使用示例：

bo=pure\_moonlight()

to\_decap=pure\_fallmount(bo('cap\_as\_factor'))

to\_deret20=pure\_fallmount(bo('ret20'))

to\_deturn20=pure\_fallmount(bo('turn20'))

to\_devol20=pure\_fallmount(bo('vol20'))

1. pure\_fallmount因子加工框架
2. pure\_fallmount是继承自父类pure\_fall的子类，专门用于对因子做横截面标准化之后相加、对因子剔除若干相关因子。使用时只需一个步骤：

1）对每个因子分别生成一个pure\_fallmount对象，初始化时，直接输入月度因子数据，注意，月度因子数据应为，列名是股票代码，索引是时间的形式。然后进行想加或相减的操作，将返回一个pure\_fallmount对象。详细内容参照父类文档。如：x=pure\_fallmount(fac1)，y=pure\_fallmount(fac2)，x+(y,)，x-(y,)

1. 调用时，直接对对象调用，将返回月度因子数据，为一个dataframe表，其格式为，列名是股票代码，索引是时间。
2. 使用示例：

bo=pure\_moonlight()

to\_decap=pure\_fallmount(bo('cap\_as\_factor'))

to\_deret20=pure\_fallmount(bo('ret20'))

to\_deturn20=pure\_fallmount(bo('turn20'))

to\_devol20=pure\_fallmount(bo('vol20'))

1. pure\_snowtrain封装剔除barra因子框架
2. pure\_snowtrain是对父类pure\_winter的封装，起到对因子剔除barra因子，得到纯净因子的效果。使用时只需要一个步骤：

1）声明一个pure\_snowtrain对象，并在初始化时输入原始因子值。

1. 调用时，直接对对象调用，将返回纯净化之后的因子值，格式为宽数据形式，代码为列名，索引为时间。如：x=pure\_snowtrain(fac)，x()。当调用方式为x(1)时，将返回一个月度因子值为纯净因子的pure\_fallmount对象。
2. 使用示例：

snow=pure\_snowtrain(final\_cpv())

1. **框架使用流程**
2. 创建一个pure\_fall的子类，写入自己用来生成月度因子的方法函数，得到原始因子fac1、fac2、fac3等。
3. 调用pure\_moon/pure\_moonnight回测框架，测试原始因子的效果。
4. 调用pure\_moonlight，获取常用的辅助因子，对得到的原始因子fac1、fac2、fac3进行加工，如合成、市值中性化、剔除相关因子等。
5. 调用pure\_moon/pure\_moonnight回测框架，测试加工后因子的效果。
6. 反复执行3、4步骤，不断加工和测试。
7. 调用pure\_winter/pure\_snowtrain对最终得到的因子，剔除barra因子，得到纯净因子snow\_fac。
8. 测试纯净因子snow\_fac的效果。
9. 测试参数敏感性。（待开发）
10. **使用示例（以复现cpv价量相关性因子研报为例）**

示例代码如下：

import tqdm

import numpy as np

import pandas as pd

import statsmodels.api as sm

from functools import partial

from pure.pure\_world import \*

class cpv\_corr(pure\_fall):

def \_\_init\_\_(self,daily\_path='',monthly\_path=''):

'''继承分钟因子的共同特性'''

super(cpv\_corr,self).\_\_init\_\_()

self.daily\_factors\_path=self.daily\_factors\_path+daily\_path

self.monthly\_factors\_path=self.monthly\_factors\_path+monthly\_path

def func\_cpv\_corr(self,df):

'''计算每一组（一天即为一组）

在每一组内将分钟数据化为日频因子数据'''

df=df[['close','amount']]

return df.corr().iloc[0,1]

def func\_daily\_to\_monthly\_mean(self,df):

'''将日度因子转化为月度因子

此处即为每月月底算前20个交易日的相关系数的均值'''

df=df.rolling(20).mean()

df=df.resample('M').last()

return df

def func\_daily\_to\_monthly\_std(self,df):

'''将日度因子转化为月度因子

此处即为每月月底算前20个交易日的相关系数的便准查'''

df=df.rolling(20).apply(np.std)

df=df.resample('M').last()

return df

def trend\_in\_group(self,l):

'''对天数次序，做无截距项的回归，取回归系数'''

y=l.to\_numpy()

x=np.arange(1,21)

result=sm.OLS(y,x).fit()

return result.params[0]

def func\_daily\_to\_monthly\_trend(self,df):

'''保留每一天的趋势系数'''

tqdm.tqdm.pandas(desc='稍等')

df=df.rolling(20).progress\_apply(self.trend\_in\_group)

df=df.resample('M').last()

return df

def main(hs300=False,zz500=False):

#市值、ret20、turn20、vol20

bo=pure\_moonlight()

to\_decap=pure\_fallmount(bo('cap\_as\_factor'))

to\_deret20=pure\_fallmount(bo('ret20'))

to\_deturn20=pure\_fallmount(bo('turn20'))

to\_devol20=pure\_fallmount(bo('vol20'))

#合成均值和标准差因子

cpv\_corr\_mean=cpv\_corr(daily\_path='价量相关性.feather',monthly\_path='价量相关性均值.feather')

cpv\_corr\_mean.run(cpv\_corr\_mean.func\_cpv\_corr,cpv\_corr\_mean.func\_daily\_to\_monthly\_mean,neutralize=True,hs300=hs300,zz500=zz500)

cpv\_corr\_std=cpv\_corr(daily\_path='价量相关性.feather',monthly\_path='价量相关性标准差.feather')

cpv\_corr\_std.run(cpv\_corr\_std.func\_cpv\_corr,cpv\_corr\_std.func\_daily\_to\_monthly\_std,neutralize=True,hs300=hs300,zz500=zz500)

cpv\_corr\_twins=(cpv\_corr\_mean+(cpv\_corr\_std,))()

cpv\_corr\_here\_be=cpv\_corr\_twins[(cpv\_corr\_twins.index>=pd.Timestamp('2014-01-01'))&(cpv\_corr\_twins.index<=pd.Timestamp('2020-01-31'))]

#剔除ret20并回测

cpv\_corr\_unde\_be=pure\_fallmount(cpv\_corr\_here\_be)

cpv\_corr\_de\_be=cpv\_corr\_unde\_be-(to\_deret20,)

cpv\_corr\_de\_be=cpv\_corr\_de\_be().iloc[:-1,:]

shen\_be=pure\_moonnight(cpv\_corr\_de\_be,5,plt=False,filename='价量相关性因子5分组')

#计算趋势因子

cpv\_corr\_trend=cpv\_corr(daily\_path='价量相关性.feather',monthly\_path='价量相关性趋.feather')

cpv\_corr\_trend.run(cpv\_corr\_trend.func\_cpv\_corr,cpv\_corr\_trend.func\_daily\_to\_monthly\_trend,neutralize=False,hs300=hs300,zz500=zz500)

# cpv\_corr\_twins.reset\_index().to\_feather('D:/因子数据/月频\_价量相关性.feather')

cpv\_corr\_here=cpv\_corr\_trend()[(cpv\_corr\_trend().index>=pd.Timestamp('2014-01-01'))&(cpv\_corr\_trend().index<=pd.Timestamp('2020-01-31'))]

#剔除市值、ret20、turn20、vol20并回测

cpv\_corr\_unde=pure\_fallmount(cpv\_corr\_here)

cpv\_corr\_de=cpv\_corr\_unde-(to\_decap,to\_deret20,to\_deturn20,to\_devol20,)

cpv\_corr\_de=cpv\_corr\_de().iloc[:-1,:]

shen=pure\_moonnight(cpv\_corr\_de,5,plt=False,filename='价量相关性因子5分组')

#合成均值、标准差因子和趋势因子并回测

deret20=pure\_fallmount(cpv\_corr\_de\_be)

trend=pure\_fallmount(cpv\_corr\_de)

final\_cpv=deret20+(trend,)

shen\_final=pure\_moonnight(final\_cpv(),5,plt=False,filename='价量相关性因子5分组')

#去除barra因子并回测

snow=pure\_snowtrain(final\_cpv())

snow\_shen=pure\_moonnight(snow(),5,plt=False,filename='纯净因子5分组')

main(False,False)

main(True,False)

main(False,True)