# 单因子测试(下)——回归测试法

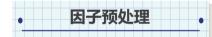
原创 量化小白H 量化小白上分记 2018-12-23

收录于话题

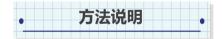
#多因子模型

12个 >

之前两篇分别总结了因子数据的预处理和单因子测试的分层测试法,本篇总结回归测试法,相较于分层测试法,回归测试法更简洁。



与分层测试法不同,回归法测试时,因子可以不进行中性化处理,只进行异常值处理和标准化(zscore)处理,将中性化的过程包含在测试过程中。



每一期,对全样本做一次回归,回归时将本期到下一期的股票收益率作为因变量,当期的因子暴露值作为自变量,同时考虑到市值中性和行业中性,加入行业虚拟变量和市值变量作为自变量,市值变量可以取对数消除量纲,之后进行回归,即

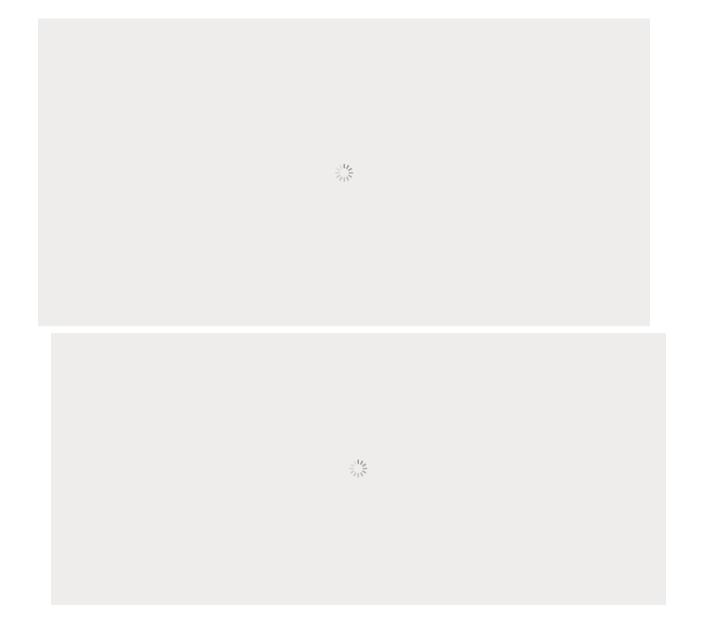
$$\begin{bmatrix} r_{t1} \\ ... \\ r_{tn} \end{bmatrix} = \begin{bmatrix} \beta_{t1} & I_{t1,1} & ... I_{t1,k} & m_{t1} \\ ... & ... & ... & ... \\ \beta_{tn} & I_{tn,1} & ... I_{tn,k} & m_{tn} \end{bmatrix} \begin{bmatrix} f_{t1} \\ ... \\ f_{tm} \end{bmatrix} + \begin{bmatrix} u_{t1} \\ ... \\ u_{tn} \end{bmatrix}$$
其中,r为股票收益率, $\beta$ 为因子暴露,

其中,r为股票收益率, $\beta$ 为因于泰路 I为行业虚拟变量,m为市值 f为因子收益

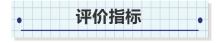
得到的回归结果中,因子暴露的系数即为因子收益率,通过多期回归后,就可以得到因子收益率序列及回归的t值序列,通过这两个序列可以构造指标分析因子的表现。

## 一些细节

1. 包含行业虚拟变量的回归中,如果再加有截距项,就会导致完全共线性,因此回归前必须进行处理,大概有这么几种方法:**不加截距项**(如上方程);**加截距项,但将其中一个行业变量的暴露值全部设为0或者删除**,消除共线性;**施加线性约束,转化为带约束的线性回归**,约束的选择和理由如下,这个方法我没试过,所以就直接贴出来吧。



2. 这里的回归一般用最常见的OLS,但考虑到数据的异方差性,有的研报里也会用加权最小二乘(WHS)或者稳健回归(RLM)。



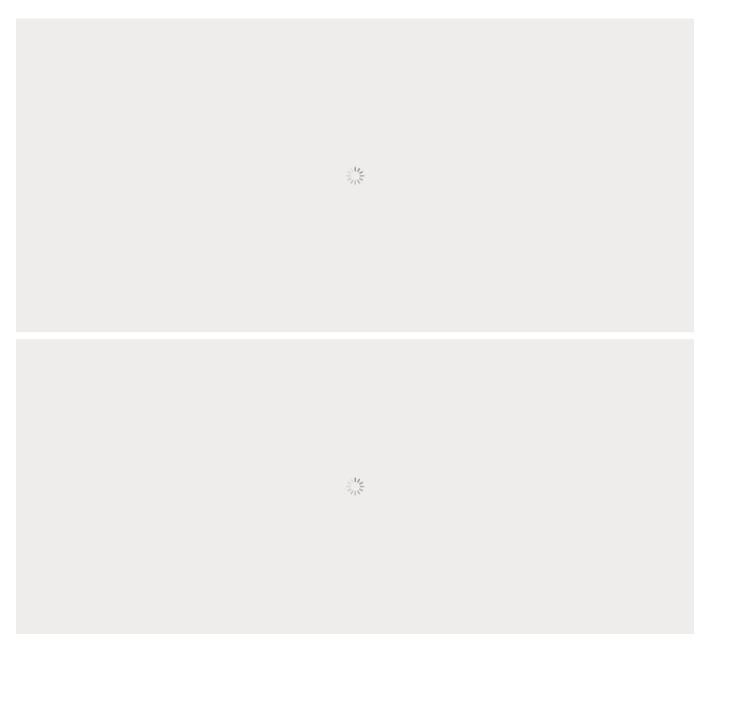
- 1. 因子收益率序列检验t值的均值
- 2. 因子收益率序列大于0的概率
- 3. t值序列绝对值的均值
- 4. t值序列绝对值大于等于2的概率
- 5. IC, ICIR
- 6. IC均值
- 7. IC绝对值大于0.02的概率

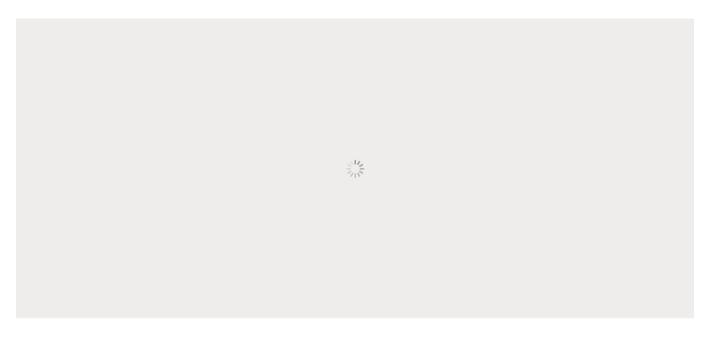
基本都是一些非常简单的指标,至于为什么取t值绝对值大于2,IC值大于0.02,也没有太好的原因,但也是符合常理的,t值绝对值越大,回归方程系数的显著性越高,IC表示相关系数,绝对值越大,表明因子暴露跟股票收益率的相关性更高。

• 因子测试实例 •

测试因子:pb\_lf,需倒数

测试区间:2010年1月-2018年5月





### PB因子收益序列显著大于0,因子收益状况良好



```
def MultiPeriodFactorTest(factor,DateStart,DateEnd,if_reciprocal,if_neutral_industry=True,
    # 获取交易日序列
    BargainDate = w.tdays(DateStart, DateEnd, "Period=M")
    BargainDate = pd.DataFrame(BargainDate.Data[0],columns = ['date'])
    # 循环调用单期因子测试函数,得到收益率序列,ICF列,t值序列
    result = pd.DataFrame(columns=["DateStart","DateEnd","factor_return","t_values","IC"])
    for i in range(1,BargainDate.shape[0]):
        datebuy = BargainDate.date[i-1]
        datesell = BargainDate.date[i]
        result1 = SingelePeriodFactorTest(factor,datebuy,datesell,if_neutral_industry, if_neutral result = result.append(result1)

result.factor_return = result.factor_return

# 计算均值
    t_mean = result.t_values.mean()
    return_mean = result.factor_return.mean()
    t_abs_mean = result.t_values.abs().mean()
```

```
if_t_0 = pd.DataFrame.mean(result.t_values>0)
IC mean = result.IC.mean()
IC std = result.IC.std()
if IC 0 = pd.DataFrame.mean(result.IC>0)
if abs IC 002 = pd.DataFrame.mean(result.IC.abs()>0.02)
ICIR = IC mean/IC std
final = {"因子收益序列t均值":t mean,
     "因子收益序列均值":return mean,
     "t>0比例":if t 0,
     "abs(t)均值":t abs mean,
     "IC均值":IC mean,
     "IC标准差":IC std,
     "IC>0比例":if IC 0,
     "abs(IC)>0.02比例":if abs IC 002,
     "ICIR":ICIR
```

```
return(final, result)
       单期因子测试函数
      def SingelePeriodFactorTest(factor,DateStart,DateEnd,if neutral industry, if neutral mktcage
        factor name = factor['value'][0][0]
        print(factor name + '单因子测试: ' + str(DateStart) + ' -- ' + str(DateEnd))
        data = getData(factor,DateStart)
        if if reciprocal:
           data.iloc[:,0] = 1/data.iloc[:,0]
        stock list = data.index.tolist()
        stock_list = list(map(lambda x:x.strip('.SZ'),stock_list))
        stock list = list(map(lambda x:x.strip('.SH'),stock list))
        stock list = tuple(stock list)
        price s = pd.read sql("select stockcode, closeprice as price s \
                      from database\
                      where stockcode in {} and bargaindate='{}'\
                      order by stockcode".format((stock list),DateStart),stkbase)
        price s=price s.set index('stockcode')
        price e = pd.read sql("select stockcode, closeprice as price e \
                      from database\
100
                      where stockcode in {} and bargaindate='{}'\
                      order by stockcode".format((stock list),DateEnd),stkbase)
104
        price e=price e.set index('stockcode')
        price=pd.merge(price s,price e,left index=True,right index=True)
        price['rt']=price['price e']/price['price s']-1
110
        factor_all= norm(data, if_neutral_industry, if_neutral_mktcap)
112
        factor all.index.name = 'stockcode'
        factor all = factor all.reset index()
114
        factor all['stockcode'] = factor all['stockcode'].apply(lambda x:x.strip('.SZ|.SH'))
115
        factor all = factor all.set index('stockcode')
116
        Alldata = pd.merge(factor all,price,left index=True,right index=True,how = 'left')
117
118
119
120
```

```
121
        y = Alldata[['rt']]
122
        X = factor all
123
        X['Intercept'] = 1
124
125
        model = sml.OLS(y,X)
126
        result=model.fit()
127
128
129
        IC = calc IC(factor all[factor name], y, 'Spearman')
130
131
132
        print(str(DateStart) + "---" +str(DateEnd)+","+factor name + '因子收益为:'+ str(round(result
133
        return t=pd.DataFrame([[DateStart,DateEnd,result.params[0],result.tvalues[0],IC]],
134
135
                     columns=["DateStart","DateEnd","factor return","t values","IC"])
136
        return(return t)
137
```

1. 20170410-光大证券-光大证券多因子系列报告之一:因子测试框架

```
收录于话题 #多因子模型·12个 >
く上一篇
                                            下一篇 >
单因子测试(中)——分层测试法
                           多因子尝试(一):因子加权方法在选股中
                                             的应用
```

喜欢此内容的人还喜欢

因未发奖金,一名程序员删代码泄愤,平台崩溃近三个小时





# 蒙古国30000头羊终于到了!全国人民流下了感动的口水( ̄--- ̄)

骑驴





暴涨40%!北京楼市,正处于启动前夜?

米宅北京

