看了一下，指数和FFF三因子，它们的日期是对的上的，所以就不用向老师一样进行转换到%Y%m的步骤。而且自己做的还是周度数据。

Usecols[0,1,3]而不是usecols[‘0,1,3’]

indexcode=np.unique(index['code'].values)

indexcode

和indexcode=np.unique(index['code'])

Indexcode

两者效果是一样的

Usecols[1,2]错 应该是usecols=[1,2]

Passing 'suffixes' which cause duplicate columns {'return\_x', 'code\_x'} is not allowed.

在pycharm中可以直接连接，就算重复也没关系，但是在jupyter中要注意！！！

.loc和.iloc的区别

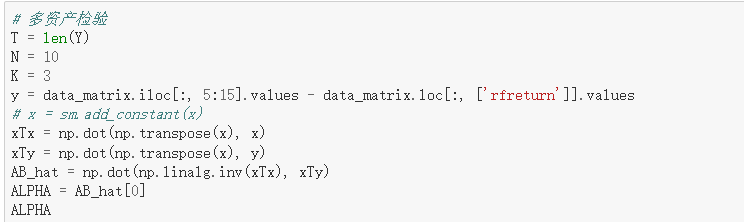
iloc使用顺序数字来索引数据，而不能使用字符型的标签来索引数据；注意：这里的顺序数字是指从0开始计数！

loc使用实际设置的索引来索引数据。但行列名为数字时，loc也可以索引数字，但这里的数字不一定从0开始编号，是对应具体行列名的数字！

并且需要注意的是，loc是前闭后闭，iloc是前闭后开。

x = data\_matrix.loc[0:2, 'date'].values取0，1，2行数据

x = data\_matrix.iloc[0:2, 1:5].values取0，1行；1，2，3，4列数据



这里需要使用X，不然少一列的。

对于numpy.array，假设matrix（N\*4）为numpy.array。那么，matrix[0]是取其第一行，输出会变为4\*1（）4rows\*1columns向量

sm.add\_constant(x)会直接给x加上一列全为1的列向量。同时也会有一个返回值，就是更改后的x

所以才会有这个X = sm.add\_constant(x)

老师给的代码中，在计算矩阵乘积时，直接先计算了最终结果的转置矩阵，然后通过AB\_hat[0]直接取第一行，也就是原来应该取的第一列，也完成了功能。

np.transpose（x）

np.linalg.inv(x)不会直接改变x

7\*4 ABHAT

注意`data\_matrix.loc[:,['rfreturn']]`和`data\_matrix.loc[:,'rfreturn']`是不同的

1. \*\*`data\_matrix.loc[:, ['rfreturn']].values`\*\*：

在这个语句中，我们使用一个包含单个元素的列表（\*\*`['rfreturn']`\*\*）来选取列。因此，返回的是一个二维 numpy 数组，数组的形状为 (n, 1)，其中 n 是 'rfreturn' 列的行数。这意味着在返回的数组中，每个元素都是一个包含单个值的子数组。

总之，两个语句的主要区别在于返回的 numpy 数组的形状。第一个语句返回一个二维数组，形状为 (n, 1)；第二个语句返回一个一维数组，形状为 (n,)。

Rows=5,6

data.dropna(inplace = True, subset=['stksize', 'stkep'])

data

##直接全部drop不是更好？

究竟需不需要全部drop？

uym = np.unique(data['yearmonth'].values)##有没有可能有一些股票没有完整的年月。这没什么关系，因为后面是根据每个时点作为一组，在其中进行排序划分

print(len(uym))

uym

np.sum(self.data['yearmonth'] == self.months[0])

计算self.data['yearmonth']中等于self.months[0]的元素的数量，并返回结果。

在jupyter里不要用read\_excel！！

将csv中的date都进来就是str类型

dm = pd.merge(left = dm,

right = self.data.loc[ind, ['code', 'monret']],

on='code',

how='left',##防止有些股票的数据在某些月份没有而被删除。在某些月份没有的情况下同样可以算平均值

sort=True)

这里和下面的def sort\_single\_ind(self):有很大的关联。如果这里使用了inner那么一些在month[0]以外的月份不存在的股票就会被删除，导致接下来按照month[0]获得的ssi会匹配不上数据量

for i in range(2,7):

filename='RESSET\_MRESSTK\_'+str(i)+'.csv'

trans=pd.read\_csv(filename,usecols=[0,2,3,4,5,6])

trans.columns=['code','date','close','tshare','monret','pe']

data=pd.concat([data,trans])

注意这里的range(2,7)应该从2开始

for i in range(5, 258, 12):

这里之所以用264-6=258，是因为range前闭后开，258不会被去到，也就不会用258及其后面的年月

Fama-macbeth不一定需要使用滚动窗口

reg\_order=['a','b']

['const'] + reg\_order]

for reg, reg\_name in zip(reg\_lst, reg\_names):元组打包

df = DF.loc[:, reg].copy() loc中reg的地方不能是作为索引的列

在copy时，索引也会一起copy过去。不过直接赋值也是一样的。

df是dataframe对象

T = df.index是获取df中的索引。

DF=pd.DataFrame({'stk':[8,9],'month':[5,6],'size':[1,2],'share':[3,4]})

DF=DF.set\_index('stk')

DF=DF.set\_index('month',append=True)

DF.index

输出结果：

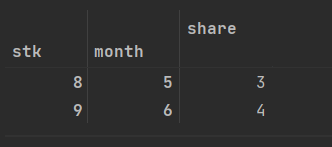
MultiIndex([(8, 5),

(9, 6)],

names=['stk', 'month'])

df.index.names[1] 这里的names是包含所有索引名称的list

对于df：



T = df.index.get\_level\_values(df.index.names[1])

获得的是:



[]里面是df.index.names[1]这个索引中的所有取值

对于m={k: pd.DataFrame(v) for k, v in res\_ts.items()}。m是一个字典，值的类型为dataframe。这种样式的字典可以使用pd.concat(m)来进行改造成一个dataframe，其中键就变成了索引。

~~当对带有索引的dataframe使用dropna时，存在na的那一个索引范围都会被删除~~

dropna的返回值为none

Dataframe中的不算在数据中的行标号也是一个索引，所以可以自行加入一个索引后进行temp = ret.set\_index('code',append=True).swaplevel()

Temp

Resetindex()可以将原来的索引都变为数据列

对于有索引的dataframe，比如A，b=A[‘’]选取的列中不能加索引的名称