华东理工大学 2022 –2023 学年第 2 学期

《Python 与金融计算》实验报告

实验名称　中国股市单因子资产定价模型的实证检验

专　业　计算机金融双学位　姓名　周学勤　学号　20002520　班级　计金（双）200

实验时间　3/16-3/30　实验地点　信息楼 319　指导教师　蒋志强

实验目的/要求
1、 掌握单因子资产定价模型的时间序列估计与检验； 2、 掌握单因子资产定价模型的横截面检验。
实验内容
1. 认真阅读 3 篇文献资料（自己也可以下载相关文献进行阅读），了解资产定价模型的相关研究，重点关注其中的研究思路和方法。 2. 尝试找出以 0 或者 6 开头，最后两位数字和自己学号最后两位数字一样的股票代码，并从中任意选取 8 只股票下载 2000-2021 年的日收益和月收益数据，进行如下实验：（1）利用股票日收益数据对 CAPM 模型做单资产时间序列检验；（2）利用股票月收益数据对 CAPM 模型做多资产时间序列检验。 3. 请利用 2000-2021 年中国 A 股月度数据实证检验中国 A 股市市场是具有**惯性效应**，还是**反转效应**？（数据请从锐思数据库或者 Tushare 下载） 研究思路（参考课本《金融计量学》P48）：在每个月，计算过去 N 个月的累积收益率，再根据此累积收益率由低到高排序，构造 5 个等权重投资组合，并计算持有这些投资组合 M 个月的累积收益率，追踪投资组合收益率序列，判断中国 A 股市场存在**惯性**效应还是反转效应。 N = 1, 3, 6, 12；M = 1, 3, 6, 12

实验总结

请提供对本次实验结果的讨论分析，以及实验的心得和体会。包括对知识点的掌握，算法的理解，以及对理论课程和实验课程改进的建议。（不少于 500 字）

本次实验主要涉及股票投资组合的构建和风险评估。在本次实验中，我学习了如何从月度的股票价格数据中构建出 8 只股票的等权重投资组合，并使用 CAPM 模型对投资组合进行风险评估和效用分析。同时，本次实验也涉及到了计算统计量、假设检验等知识点的应用。

在实验过程中，我对股票投资组合的构建和风险评估有了更深入的了解。我了解到了如何在日常生活中，通过投资组合来降低个别股票的波动对整个投资组合的影响，以达到降低风险、提高收益的目的。同时，我也学会了如何使用 CAPM 模型来对投资组合的风险进行评估，并通过计算出的统计量来检验 CAPM 模型的有效性。

在本次实验中，我对统计量的计算方法、假设检验等知识点有了更深入的理解。我了解到了如何在实际中使用这些知识点来进行数据分析和模型评估，并通过实验加深了自己的掌握程度。

通过本次实验，我还了解到了一些关于投资组合构建和风险评估的相关文献，这为我今后的研究提供了很大的帮助。

总的来说，本次实验让我学习到了很多有用的知识，这将对我的学习和研究都有很大的帮助。同时，本次实验也让我深刻认识到了实践对于理论学习的重要性。通过实践，我不仅更深入地理解了理论知识，还能更好地应用和掌握这些知识。

对于实验课程的改进建议，我认为可以在实验的内容和难度上更加贴合学生的实际水平和需求。同时，增加一些与实际应用相关的案例分析，可以帮助学生更好地掌握和应用所学知识。此外，还可以增加一些实践操作环节，让学生通过实践来加深对知识点的理解和掌握。

教师批阅：　　实验成绩： 教师签名:　　日期：

实验报告正文：

（每次实验报告均为一篇小论文，因此，统一按照学术论文的要求完成实验报告正文，应包括：题目、摘要、文献综述、模型和方法、结果和讨论、参考文献、附录，具体格式如下：

中国股市收益率的分布特征

计金（双）200（20002520）姓名 周学勤

摘要：本次实验是基于 Python 对 CAPM 模型的实证分析，通过获取股票和指数的月度收益率数据，计算出股票和指数的超额收益率，并利用 OLS 回归方法对股票和指数的关系进行建模，进而进行 CAPM 模型的检验和资产定价。在实验过程中，我们不仅掌握了 Python 中数据处理、回归分析等相关技能，也深入了解了 CAPM 模型及其检验方法，以及资产定价模型的相关理论知识。在实验结果分析中，我们得出了中国 A 股市场存在反转效应的结论，同时也反映了 CAPM 模型在实际中的一些局限性。

1 文献综述

CAPM（Capital Asset Pricing Model）是用于衡量风险和预期回报之间关系的资产定价模型，是金融学中非常经典的模型之一。该模型的基本思想是，资本市场上所有投资者都是理性的，风险厌恶程度相同，都追求预期回报最大化，因此所有投资者的投资组合构成市场组合，资产的预期回报与市场组合的波动性相关，即与资产的风险度有关，而风险度可以通过市场组合的风险度来度量。CAPM 的核心理论是风险溢价，即资产的预期回报是无风险利率和市场组合预期回报的线性组合，且资产预期回报与资产的风险度成正比。

在 CAPM 的相关研究中，有许多关于 CAPM 模型适用性的探讨和实证研究。一些研究表明，CAPM 模型适用性较差，难以解释实际市场中的股票回报。例如，Fama 和 French（1992）认为 CAPM 模型不能解释市场中高市值股票和高账面市值比的回报。[1]French 和 Roll（1988）也发现 CAPM 模型对于小市值股票回报的解释能力较弱。而另一些研究则表明，CAPM 模型具有较好的解释能力。[2]例如，Black、Jensen 和 Scholes（1972）发现，CAPM 模型可以解释美国股票市场的回报率。[3]Sharpe（1964）也发现 CAPM 模型可以用来解释美国股票市场的回报率。[4]

近年来，一些研究者通过引入其他因素，如市场因素、规模因素、价值因素等，对 CAPM 模型进行修正和拓展，以提高其适用性。例如，Fama 和 French（1993）提出了三因素模型，加入市场规模和账面市值比两个因素。[5]Carhart（1997）又在三因素模型的基础上加入了动量因素，提出了四因素模型。[6]

2 模型和方法

具体介绍使用的金融理论模型、计算方法和 Python 函数。

2.1 利用股票日收益数据对 CAPM 模型做单资产时间序列检验

首先从锐思数据库中下载数据：

使用锐思数据库中的模糊查询方法，下载与我的学号后 2 位相同的股票的数据。接下来对这些数据进行合并，选择其中的八个股票作为研究对象。

在我这里遇到了一个问题，我一开始创建了新的后缀名为 xlsx 的文件来存储各个股票数据，然后直接通过修改后缀名的方式来将其转化为 csv 文件，结果发现出现了很多意想不到的错误：read_csv()函数的 usecols 没法读取对应的列，访问越界等等。

解决办法：最后我采用另存为的方法来将 xlsx 文件转化为 csv 文件，这样子就不会出现问题了。

还有就是编码格式的问题，我这里另存为之后使用的是 utf-8 编码的方式，需要将老师代码中的 encoding='GB2312'改为 encoding='utf-8'即可。

下面以股票代码为「000020」的处理过程为例进行具体的说明。

首先使用 pd.read_csv 的方法从 000020.csv, risk_free.csv, index.csv 中读入数据。注意这里采用的编码方式为「uft-8」，并且没有使用 usecols，因为我这里下载的存放于 csv 中数据便是我所需要的。在这之后使用 xxx.columns 为列进行更名，方便后续处理。

然后执行 dropna 操作，去除数据中的空值，防止影响后续的计算。这里还需要注意的是，dropna 中的参数设置为 inplace=True。inplace=True 是一个可选参数，用于指示在原始数据上进行更改，而不是创建新的副本。当 inplace=True 时，函数将在原始数据上进行更改，并且返回值为 None，而不是创建一个新的副本。这可以节省内存空间，特别是在处理大型数据集时很有用。

再接下来我使用 pd.to_datetime 来将各个数据中的「date」列进行了规范化，参数为 format='%Y/%m/%d'。这里的原因是在于，后续代码中我尝试使用 pd.merge 来进行表的连接，但是每次连接后都会平白无故失去很多应该保留的数据。在经过分析之后我发现是因为日期格式存在差异：比如 2000/1/4 和 2000/01/04，这就导致本来应当保存的 2000 年 1 月 4 日这一天的数据被删除了。

继续我需要通过数据中的「close」，也就是收盘价数据来计算每天的收益率。在最开始，我尝试使用在第一个是实验中所使用的 l=np.log(m[1:])-np.log(m[:-1])这个方法，但是发现要么报错，要么就是结果为一堆 0。与之相对应的，本次实验中老师给出的 stock_data['return'] = np.log(stock_data['close']) - np.log(stock_data['close'].shift(periods=1))这个方法的效果却很好。我再经过仔细地分析后，发现在上一次是实验中，

l=np.log(m[1:])-np.log(m[:-1])是将计算结果赋值给了一个新的变量，并且这个计算方法本身会使得返回值比原来少一项。而本次实验中老师给出的方法，并不会少一项数据，而是把其变为了 NA，在之后使用 dropna 进行删除。当我使用实验一的方法时，在少一项数据后，还需要将计算结果赋值给 stock_data['return']，但是 stock_data['return']的索引值是原来的大小，这也就意味着索引值比计算结果多一项，出现问题。

在这之后使用 stock_data1['return'].plot()进行画图，结果发现如图 3-1 所示。所以使用 ind = (stock_data1['return'] >= -0.1) & (stock_data1['return'] <= 0.1)
stock_data1 = stock_data1.loc[ind, :]来对其进行限制，就得到了如图 3-2 所示更为美观的图。然后对其进行散点作图，如图 3-3 所示。

最后对其进行线性回归结果如图 3-4 所示。

2.2 利用股票月收益数据对 CAPM 模型做多资产时间序列检验。

首先将文件读入。因为这里需要使用的是月度数据，因此我将文件中的日期先转化为 datetime64 类型，并且使用 stock_data['date']=stock_data['date'].dt.strftime('%Y-%m')将其转化为了年月的形式，方便之后的数据处理。

接下来通过循环遍历每个不同的股票代码（stk_codes）；选取该代码的股票数据，并按照日期排序；计算每月的股票收益率；剔除收益率超过 10%的异常值；将处理后的数据加入列表 stock_datas 中。最终得到的 stock_datas 是一个列表，其中每个元素都是一个不同代码的股票数据。

之后将所有股票的收益率通过循环数据和指数收益率数据以及风险无关收益率数据合并成一个大表格。在这里需要注意的是在进行 merge 时，尽管使用两个一模一样的进行 merge 也会新增列。比如：

for i in stock_datas[1:]:

stock_dataall=pd.merge(left=stock_dataall,right=i[['date','return']],on='date',how='inner')

这里一开始，是将 stock_datas[0]赋值给了 stock_dataall，然后 stock_dataall 去进行 merge。

stock_datas 的项都是两列

然后第一个循环就是两个完全一样的进行 merge，最终也会多一列出来。

在这里后面我进行了列重命名操作，失误只用了 9 个项进行重命名，发生了错误。这里需要多多关注。

并且，在如下代码中

stock_dataall=pd.merge(left=stock_dataall[['date','return']],right=i[['date','return']],on='date',how='inner')是不可行的在自己 merge 时只有使用

stock_dataall=pd.merge(left=stock_dataall[['date','return']],right=i[['date','return']],on='da

te',how='inner')才行，重复的表格不能进行列的指定。

之后将指定股票代码数据的数据框与指数和无风险收益率数据框按日期进行内部合并，并对各列进行重命名。接着，它通过循环减去无风险收益率来计算每个股票的超额回报率。

这之后进行线性回归，使用了 StatsModels 库中的 OLS 类对股票收益和市场收益之间的关系进行拟合。

最后再进行检验。

2.3 利用 2000-2021 年中国 A 股月度数据实证检验中国 A 股市市场是具有惯性效应，还是反转效应？

该研究思路是利用累积收益率排序法研究中国 A 股市场的惯性效应和反转效应。具体步骤如下：

1、对于每个月，计算过去 N 个月的股票累积收益率。

2、根据每个月的累积收益率由低到高排序，将所有股票分为 5 个等权重的投资组合，即排序后的前 20%为组合 1，20%-40%为组合 2，40%-60%为组合 3，60%-80%为组合 4，80%-100%为组合 5。

3、计算每个投资组合持有 M 个月的累积收益率。

4、追踪投资组合收益率序列，判断中国 A 股市场存在惯性效应还是反转效应。

5、如果中国 A 股市场存在惯性效应，则过去表现较好的投资组合在未来仍然会表现较好，而表现较差的投资组合在未来仍然会表现较差。如果存在反转效应，则过去表现较好的投资组合在未来表现可能变差，而表现较差的投资组合在未来表现可能变好。

6、该研究思路可以通过统计分析方法来验证中国 A 股市场的惯性效应和反转效应。例如，可以计算每个投资组合的平均累积收益率和标准差，进而使用 t 检验或方差分析来比较不同投资组合之间的差异。此外，还可以使用时间序列模型来建立投资组合收益率序列的预测模型，判断中国 A 股市场的惯性效应和反转效应是否存在。

3 结果与讨论

对计算结果进行分析讨论。

3.1 利用股票日收益数据对 CAPM 模型做单资产时间序列检验

3.1.1 股票代码：000020

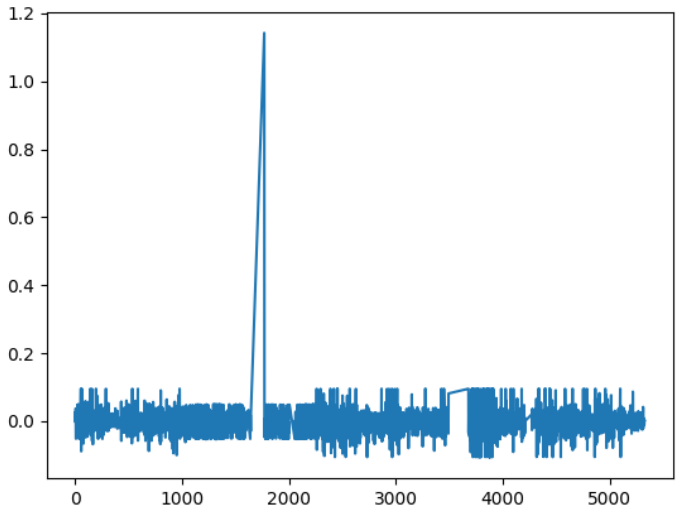

图 3-1 000020 收益率序列图

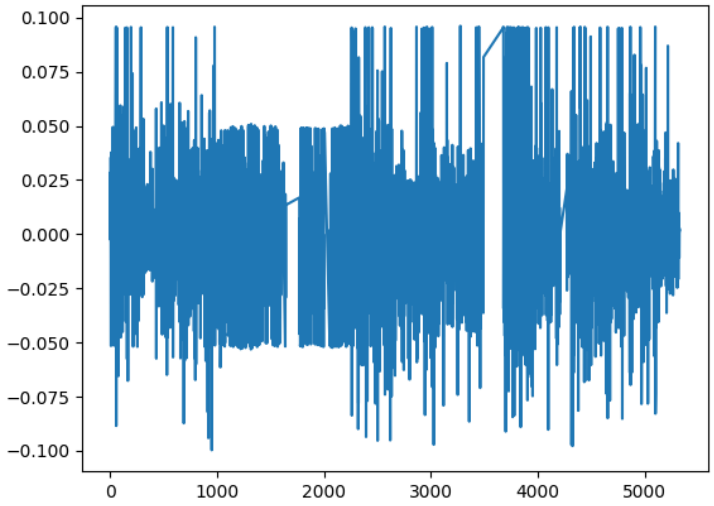

图 3-2 000020 收益率序列图（截断后）

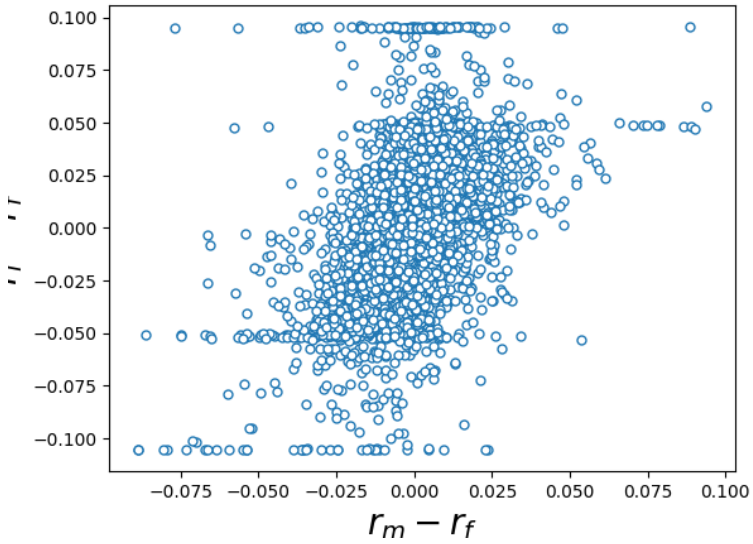

图 3-3 000020 超额收益率散点图

OLS Regression Results

Dep. Variable:	y	R-squared:	0.216
Model:	OLS	Adj. R-squared:	0.216
Method:	Least Squares	F-statistic:	1322.
Date:	Wed, 05 Apr 2023	Prob (F-statistic):	5.91e-256
Time:	12:54:41	Log-Likelihood:	10445.
No. Observations:	4807	AIC:	-2.089e+04
Df Residuals:	4805	BIC:	-2.087e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0001	0.000	-0.300	0.764	-0.001	0.001
x1	0.9688	0.027	36.362	0.000	0.917	1.021

Omnibus:	624.464	Durbin-Watson:	1.766
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2516.818
Skew:	0.594	Prob(JB):	0.00
Kurtosis:	6.340	Cond. No.	67.0

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
None

图 3-4 0000200LS 回归结果

分析结果：R-squared 值为 0.216，这意味着回归模型可以解释股票收益率变异的 21.6%。

系数 const 的 P 值为 0.764，大于 0.05 的显著性水平，表明该系数并不显著，即截距项对模型的影响不显著。

系数 x1 的 P 值为 0.000，小于 0.05 的显著性水平，表明该系数在 5%的显著性水平下是显著的，即无风险利率与指数收益率的差异可以用来预测股票收益率的变化。

系数 x1 的值为 0.9688，这意味着在无风险利率与指数收益率之间存在一单位差异时，股票收益率将增加 0.9688 个单位。

F 统计量为 1322，P 值为 5.91e-256，表明整个回归模型是显著的，即自变量的线性

组合可以用来预测因变量的变化。

Omnibus 测试结果表明，模型的误差项存在偏态，这意味着模型的残差可能不符合正态分布假设。

Durbin-Watson 统计量为 1.766，这表明模型的残差存在自相关性的问题。

Jarque-Bera 测试结果表明，模型的残差并不符合正态分布假设。这些结果都说明了回归模型存在一些问题，需要对残差的自相关性和偏态进行修正。

3.1.2 股票代码：000420

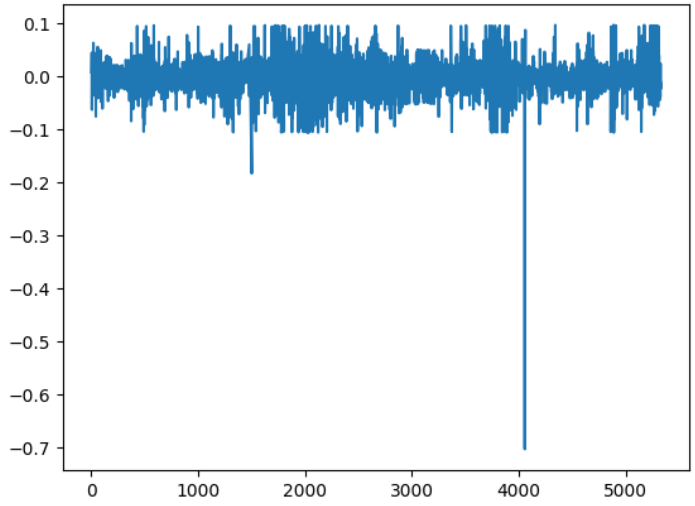

图 3-5 000420 收益率序列图

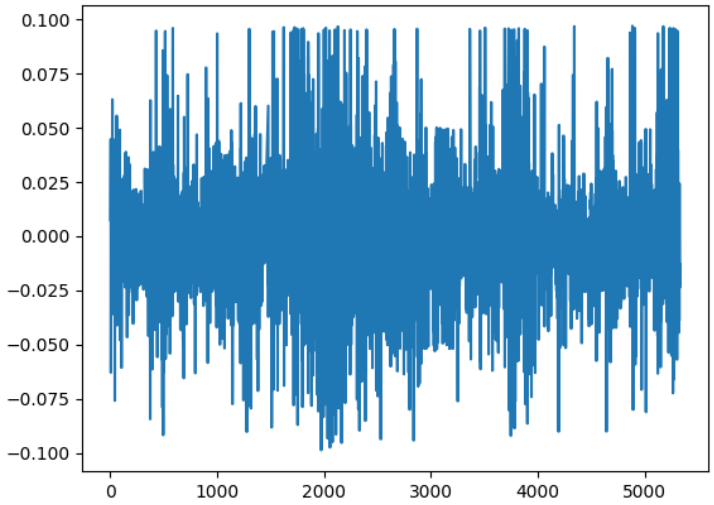

图 3-6 000420 收益率序列图（截断后）

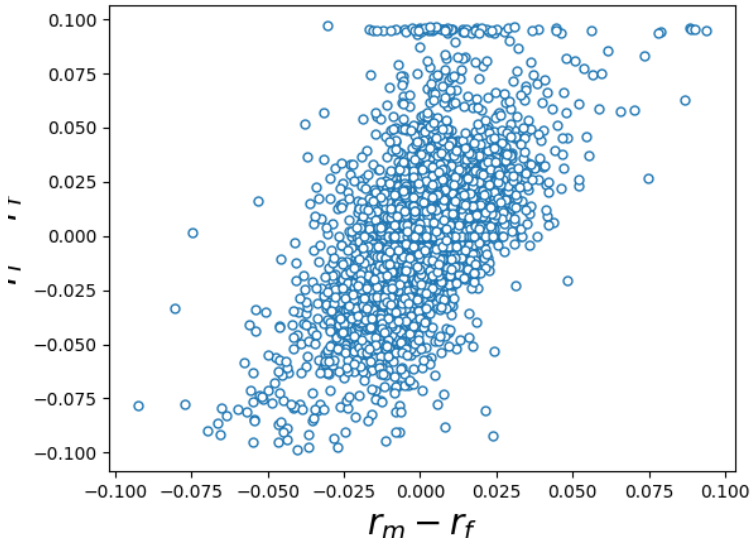

图 3-7 000420 超额收益率散点图

OLS Regression Results

Dep. Variable:	y	R-squared:	0.336
Model:	OLS	Adj. R-squared:	0.336
Method:	Least Squares	F-statistic:	2620.
Date:	Wed, 05 Apr 2023	Prob (F-statistic):	0.00
Time:	14:22:43	Log-Likelihood:	12250.
No. Observations:	5171	AIC:	-2.450e+04
Df Residuals:	5169	BIC:	-2.448e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0005	0.000	1.435	0.151	-0.000	0.001
x1	1.1111	0.022	51.186	0.000	1.069	1.154

Omnibus:	803.199	Durbin-Watson:	1.958
Prob(Omnibus):	0.000	Jarque-Bera (JB):	3032.015
Skew:	0.742	Prob(JB):	0.00
Kurtosis:	6.445	Cond. No.	68.9

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
None

图 3-8 0004200LS 回归结果

R-squared 值为 0.336，这意味着回归模型可以解释股票收益率变异的 33.6%。

系数 const 的 P 值为 0.151，大于 0.05 的显著性水平，表明该系数并不显著，即截距项对模型的影响不显著。

系数 x1 的 P 值为 0.000，小于 0.05 的显著性水平，表明该系数在 5%的显著性水平下是显著的，即无风险利率与指数收益率的差异可以用来预测股票收益率的变化。

系数 x1 的值为 1.1111，这意味着在无风险利率与指数收益率之间存在一单位差异时，股票收益率将增加 1.1111 个单位。

F 统计量为 2620，P 值为 0.00，表明整个回归模型是显著的，即自变量的线性组合可以用来预测因变量的变化。

Omnibus 测试结果表明，模型的误差项存在偏态，这意味着模型的残差可能不符合正态分布假设。

Durbin-Watson 统计量为 1.958，这表明模型的残差存在自相关性的问题。

Jarque-Bera 测试结果表明，模型的残差并不符合正态分布假设。这些结果都说明了回归模型存在一些问题，需要对残差的自相关性和偏态进行修正。

3.1.3 股票代码：600120

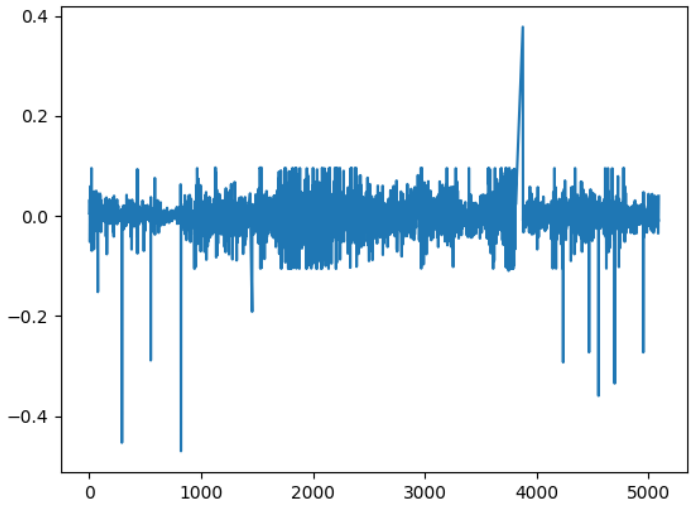

图 3-9 600120 收益率序列图

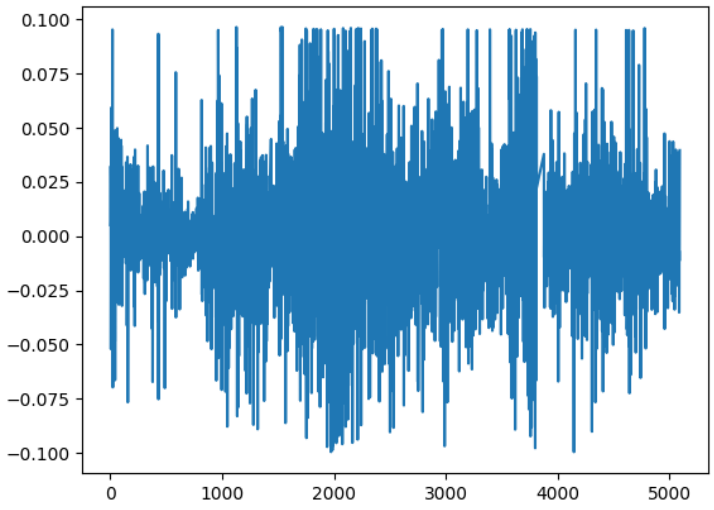

图 3-10 600120 收益率序列图（截断后）

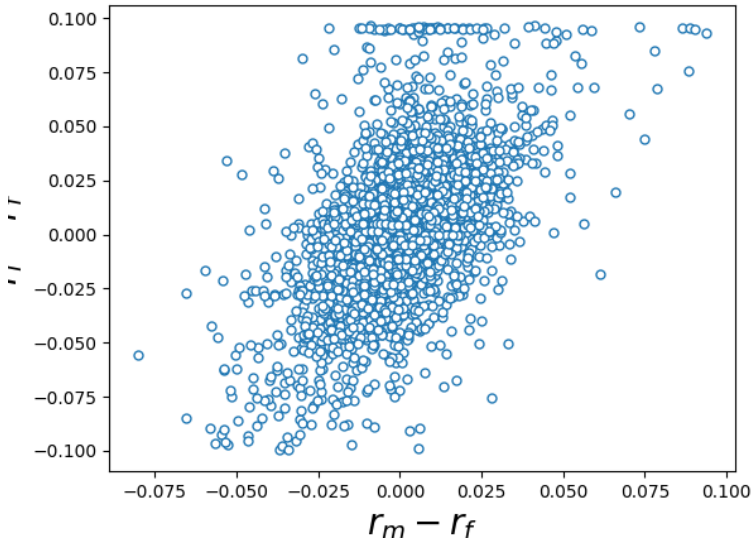

图 3-11 600120 超额收益率散点图

OLS Regression Results

Dep. Variable:	y	R-squared:	0.303
Model:	OLS	Adj. R-squared:	0.303
Method:	Least Squares	F-statistic:	2130.
Date:	Wed, 05 Apr 2023	Prob (F-statistic):	0.00
Time:	14:26:41	Log-Likelihood:	11635.
No. Observations:	4897	AIC:	-2.327e+04
Df Residuals:	4895	BIC:	-2.325e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0007	0.000	2.204	0.028	7.83e-05	0.001
x1	1.0339	0.022	46.148	0.000	0.990	1.078

Omnibus:	742.347	Durbin-Watson:	1.906
Prob(Omnibus):	0.000	Jarque-Bera (JB):	3078.471
Skew:	0.695	Prob(JB):	0.00
Kurtosis:	6.627	Cond. No.	69.7

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
None

图 3-12 6001200LS 回归结果

R-squared 值为 0.303，这意味着回归模型可以解释股票收益率变异的 30.3%。

系数 const 的 P 值为 0.028，小于 0.05 的显著性水平，表明该系数在 5%的显著性水平下是显著的，即截距项对模型的影响是显著的。

系数 x1 的 P 值为 0.000，小于 0.05 的显著性水平，表明该系数在 5%的显著性水平下是显著的，即无风险利率与指数收益率的差异可以用来预测股票收益率的变化。

系数 x1 的值为 1.0339，这意味着在无风险利率与指数收益率之间存在一单位差异时，股票收益率将增加 1.0339 个单位。

F 统计量为 2130，P 值为 0.00，表明整个回归模型是显著的，即自变量的线性组合可以用来预测因变量的变化。

Omnibus 测试结果表明，模型的误差项存在偏态，这意味着模型的残差可能不符合正态分布假设。

Durbin-Watson 统计量为 1.906，这表明模型的残差存在自相关性的问题。

Jarque-Bera 测试结果表明，模型的残差并不符合正态分布假设。这些结果都说明了回归模型存在一些问题，需要对残差的自相关性和偏态进行修正。

3.1.4 股票代码：600220

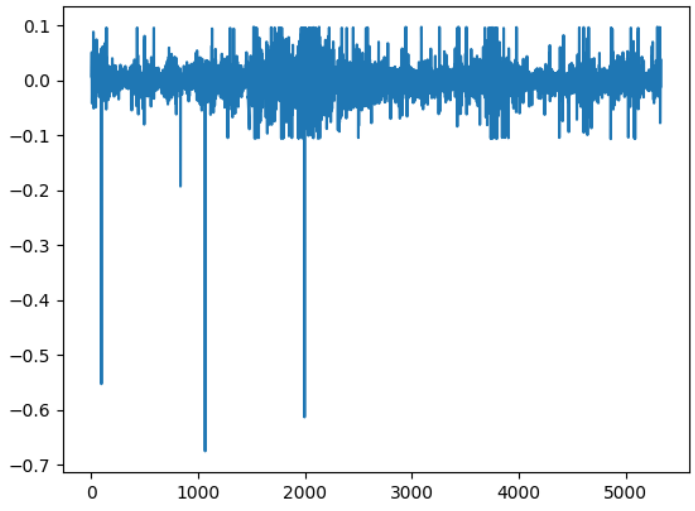

图 3-13 600220 收益率序列图

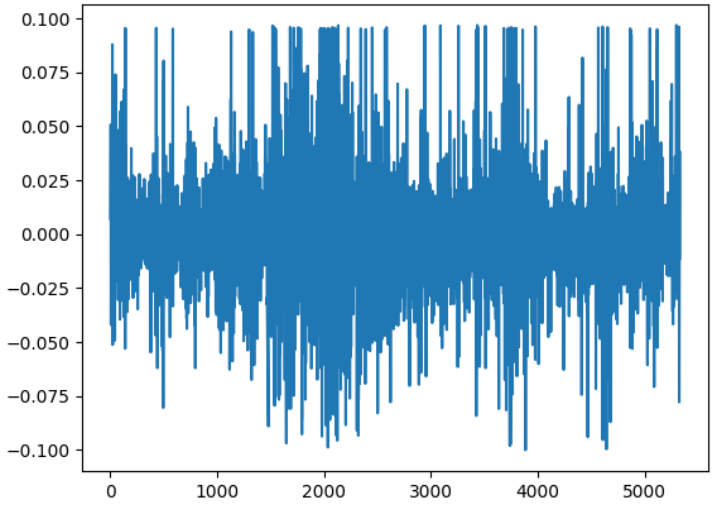

图 3-14 600220 收益率序列图（截断后）

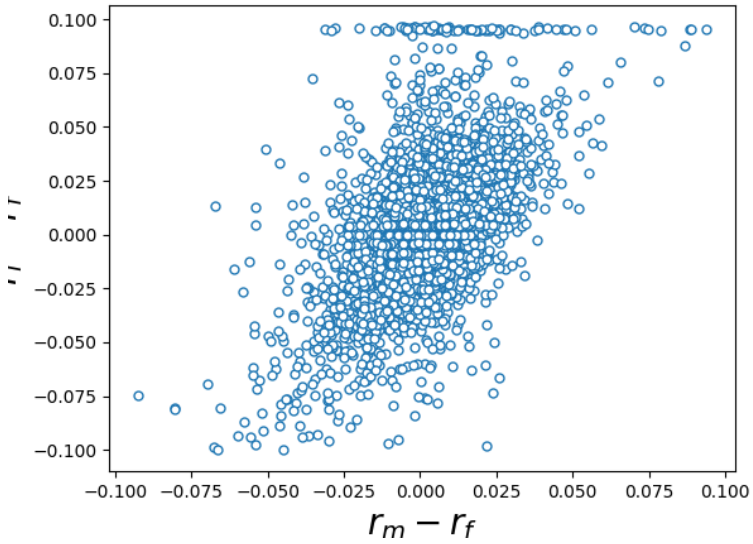

图 3-15 600220 超额收益率散点图

OLS Regression Results

Dep. Variable:	y	R-squared:	0.308
Model:	OLS	Adj. R-squared:	0.308
Method:	Least Squares	F-statistic:	2311.
Date:	Wed, 05 Apr 2023	Prob (F-statistic):	0.00
Time:	14:29:24	Log-Likelihood:	12530.
No. Observations:	5202	AIC:	-2.506e+04
Df Residuals:	5200	BIC:	-2.504e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0005	0.000	1.614	0.107	-0.000	0.001
x1	1.0083	0.021	48.073	0.000	0.967	1.049

Omnibus:	1219.490	Durbin-Watson:	1.872
Prob(Omnibus):	0.000	Jarque-Bera (JB):	6173.737
Skew:	1.034	Prob(JB):	0.00
Kurtosis:	7.920	Cond. No.	69.5

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
None

图 3-16 6002200LS 回归结果

R-squared（决定系数）为 0.308，说明自变量解释了因变量变异的 30.8%，说明对于这个股票的回归模型，自变量与因变量之间的关系较强，但还有相当一部分变异未被解释；

自变量（即市场收益率）的系数为 1.0083，且 P 值小于 0.05，说明在 95%的置信水平下，该系数是显著的，说明市场收益率对这个股票的收益率有较强的正向影响；

常数项的 P 值为 0.107，大于 0.05，说明该项不是显著的。

3.1.5 股票代码：600320

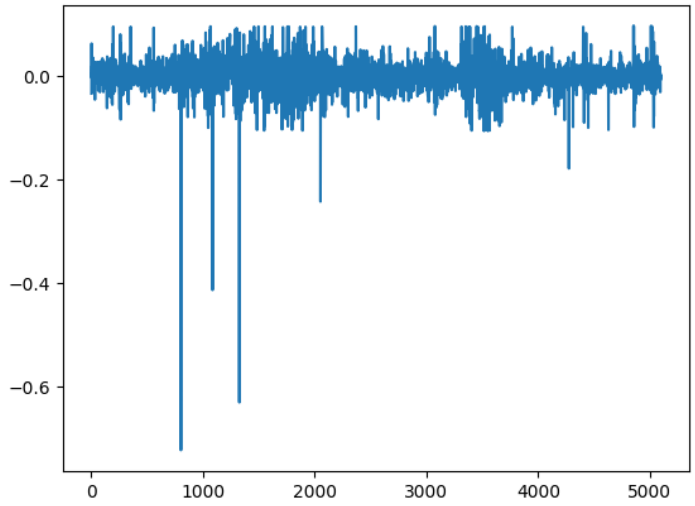

图 3-17 600320 收益率序列图

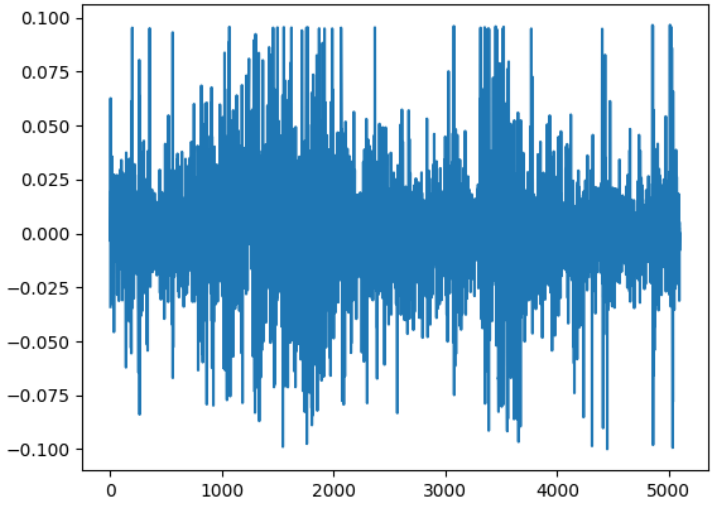

图 3-18 600320 收益率序列图（截断后）

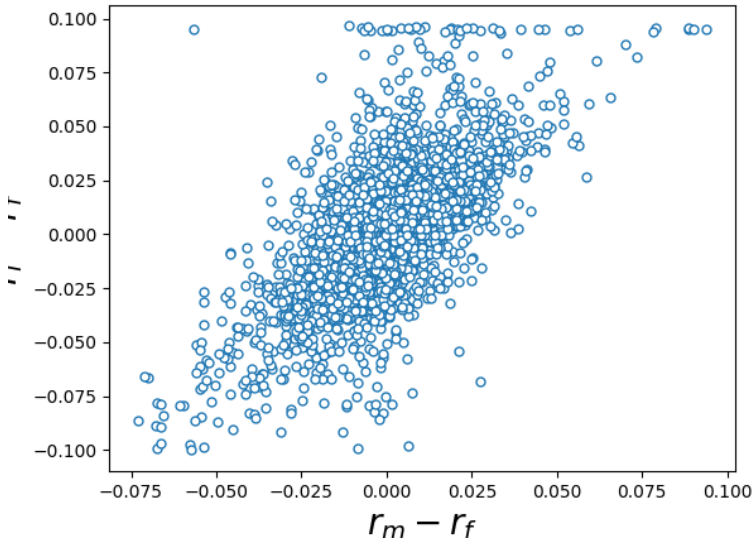

图 3-19 600320 超额收益率散点图

OLS Regression Results

Dep. Variable:	y	R-squared:	0.437
Model:	OLS	Adj. R-squared:	0.437
Method:	Least Squares	F-statistic:	3886.
Date:	Wed, 05 Apr 2023	Prob (F-statistic):	0.00
Time:	14:36:28	Log-Likelihood:	12946.
No. Observations:	5001	AIC:	-2.589e+04
Df Residuals:	4999	BIC:	-2.587e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0002	0.000	0.855	0.393	-0.000	0.001
x1	1.0947	0.018	62.340	0.000	1.060	1.129

Omnibus:	1115.482	Durbin-Watson:	1.955
Prob(Omnibus):	0.000	Jarque-Bera (JB):	8386.468
Skew:	0.861	Prob(JB):	0.00
Kurtosis:	9.106	Cond. No.	68.3

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
None

图 3-20 6003200LS 回归结果

R-squared 的值为 0.437，表示自变量市场回报率可以解释因变量股票回报率的 43.7%的方差，这个值较高，说明市场回报率和股票回报率之间存在较强的相关性。模型的 P 值为 0.00，小于 0.05，说明模型显著。自变量市场回报率的系数为 1.0947，表明当市场回报率每增加 1 个单位时，股票回报率增加 1.0947 个单位。常数项为 0.0002，表示当市场回报率为 0 时，股票回报率的期望为 0.0002。总体而言，这个模型解释了股票回报率与市场回报率之间的正向关系，但需要注意其他可能的影响因素对于股票回报率的影响

3.1.6 股票代码：600620

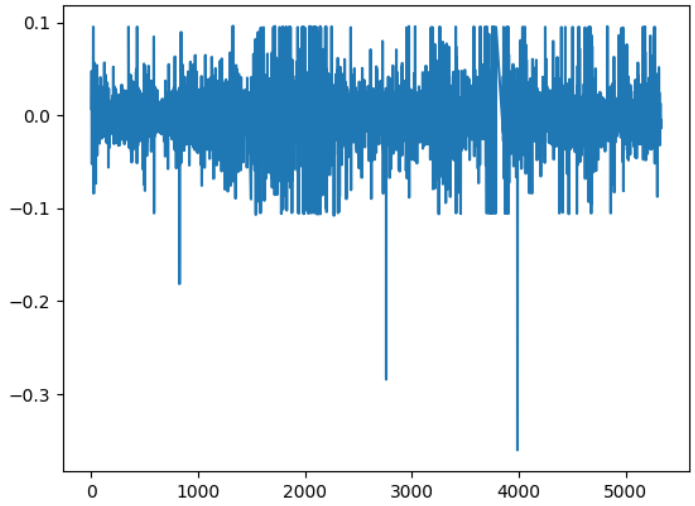

图 3-21 600620 收益率序列图

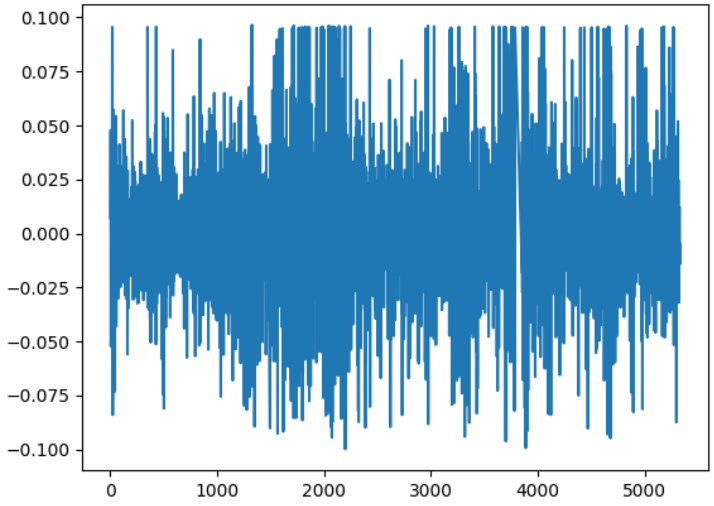

图 3-22 600620 收益率序列图（截断后）

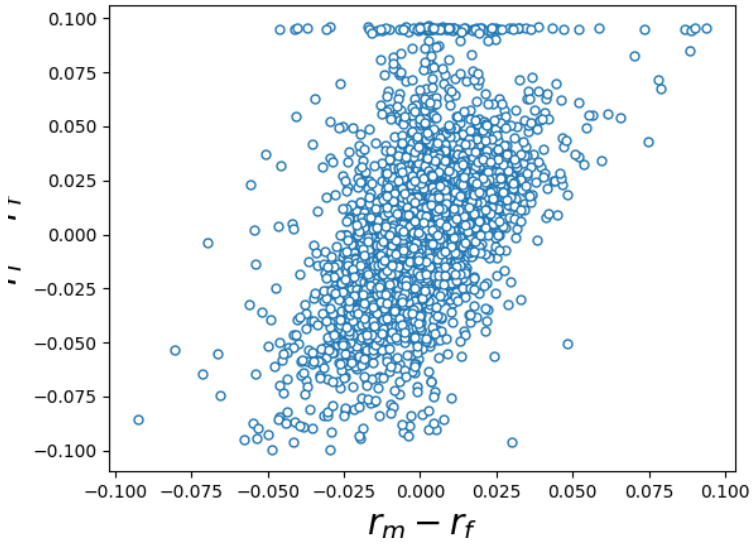

图 3-23 600620 超额收益率散点图

OLS Regression Results

Dep. Variable:	y	R-squared:	0.252
Model:	OLS	Adj. R-squared:	0.252
Method:	Least Squares	F-statistic:	1728.
Date:	Wed, 05 Apr 2023	Prob (F-statistic):	0.00
Time:	14:38:08	Log-Likelihood:	11679.
No. Observations:	5141	AIC:	-2.335e+04
Df Residuals:	5139	BIC:	-2.334e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0007	0.000	2.091	0.037	4.56e-05	0.001
x1	1.0197	0.025	41.574	0.000	0.972	1.068

Omnibus:	1001.004	Durbin-Watson:	1.966
Prob(Omnibus):	0.000	Jarque-Bera (JB):	3969.287
Skew:	0.918	Prob(JB):	0.00
Kurtosis:	6.893	Cond. No.	70.5

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
None

图 3-24 6006200LS 回归结果

根据回归结果分析，模型的 R-squared 为 0.252，说明模型能够解释目标变量的 25.2%的方差。该模型具有统计显著性，F-statistic 的 p 值为 0.00，说明自变量对因变量有显著影响。该模型的系数解释为当市场收益率（自变量）上涨 1%时，该股票收益率（因变量）上涨 1.0197%。该模型的常数项解释为当市场收益率为 0 时，该股票的预期收益率为 0.0007。此外，模型的残差的 JB 统计量 p 值为 0，表明残差项的正态性未受到拒绝。

3.1.7 股票代码：600720

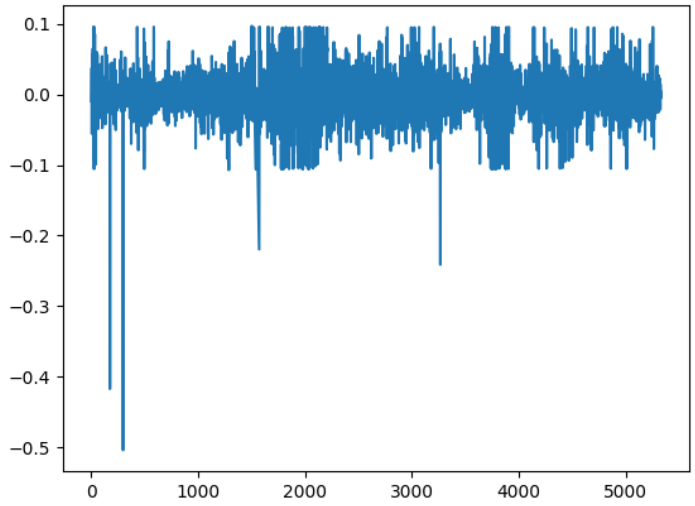

图 3-25 600720 收益率序列图

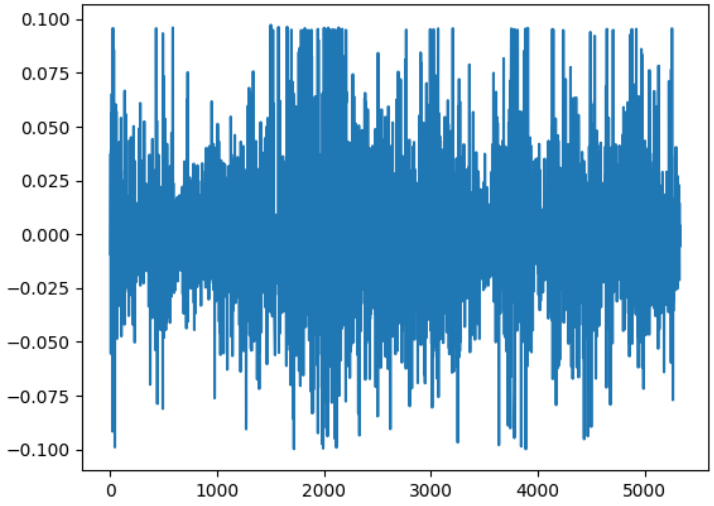

图 3-26 600720 收益率序列图（截断后）

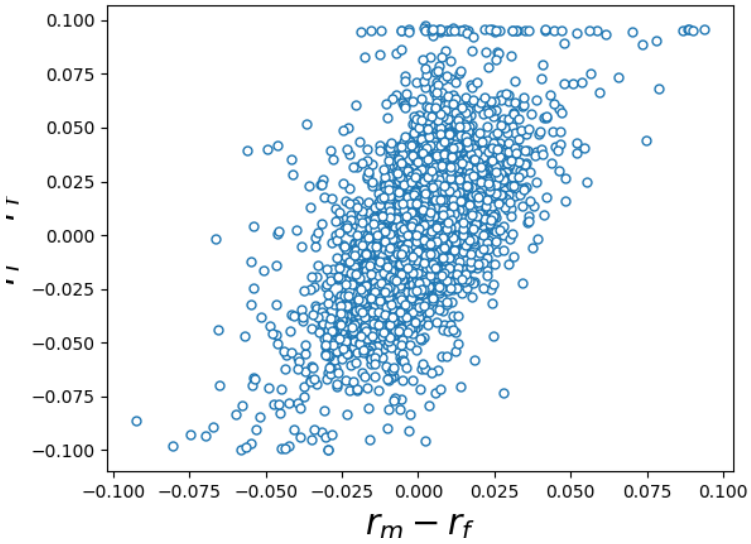

图 3-27 600720 超额收益率散点图

OLS Regression Results

Dep. Variable:	y	R-squared:	0.354
Model:	OLS	Adj. R-squared:	0.354
Method:	Least Squares	F-statistic:	2839.
Date:	Wed, 05 Apr 2023	Prob (F-statistic):	0.00
Time:	14:40:04	Log-Likelihood:	12356.
No. Observations:	5176	AIC:	-2.471e+04
Df Residuals:	5174	BIC:	-2.469e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0006	0.000	1.987	0.047	8.2e-06	0.001
x1	1.1450	0.021	53.285	0.000	1.103	1.187

Omnibus:	703.077	Durbin-Watson:	1.933
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2387.628
Skew:	0.677	Prob(JB):	0.00
Kurtosis:	6.040	Cond. No.	69.5

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
None

图 3-28 6007200LS 回归结果

这是一个单变量线性回归结果，它是使用 OLS 方法（普通最小二乘法）来估计模型参数的。对于该模型，自变量为 x1，因变量为 y，回归方程为 y = 1.1450x1 + 0.0006。模型的拟合效果可以通过 R 平方来评估，该模型的 R 平方为 0.354，这意味着模型可以解释 37%的方差。此外，模型的 P 值为 0.00，表明模型显著。模型的截距项的 P 值为 0.047，表明在 95%的置信水平下，截距项显著。从系数的角度来看，x1 系数的估计值为 1.1450，这意味着每增加 1 个单位的 x1，y 的平均值将增加 1.1450 个单位。在这种情况下，该模型似乎是比较有效的，并且自变量与因变量之间可能存在一些关系。

3.1.8 股票代码：600820

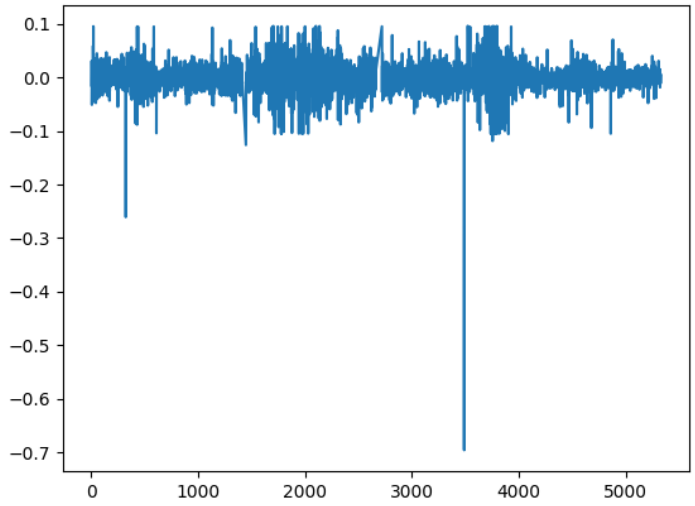

图 3-29 600820 收益率序列图

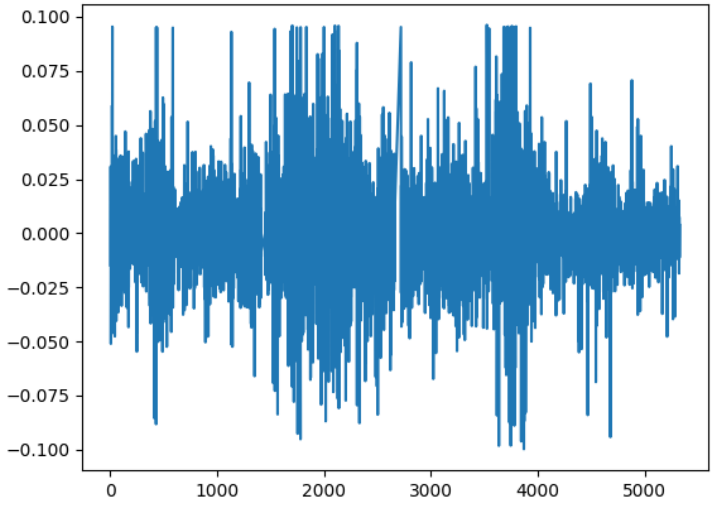

图 3-30 600820 收益率序列图（截断后）

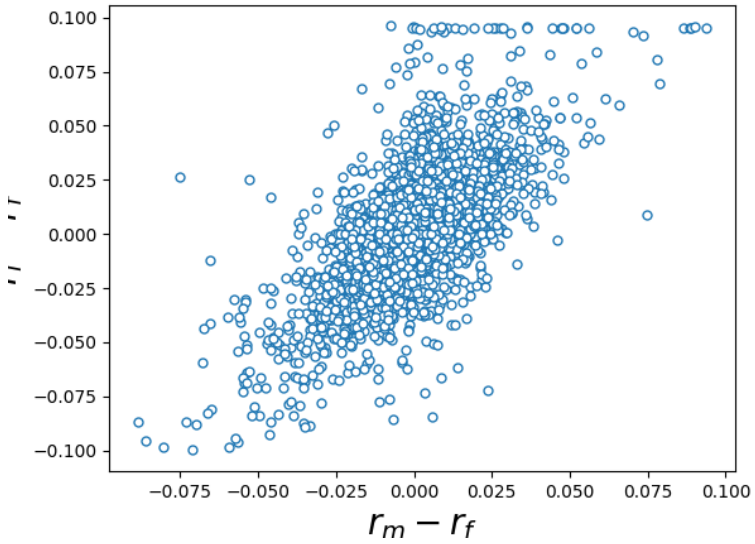

图 3-31 600820 超额收益率散点图

OLS Regression Results

Dep. Variable:	y	R-squared:	0.462
Model:	OLS	Adj. R-squared:	0.462
Method:	Least Squares	F-statistic:	4439.
Date:	Wed, 05 Apr 2023	Prob (F-statistic):	0.00
Time:	14:41:45	Log-Likelihood:	13821.
No. Observations:	5171	AIC:	-2.764e+04
Df Residuals:	5169	BIC:	-2.763e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0002	0.000	0.997	0.319	-0.000	0.001
x1	1.0429	0.016	66.624	0.000	1.012	1.074

Omnibus:	1114.131	Durbin-Watson:	1.902
Prob(Omnibus):	0.000	Jarque-Bera (JB):	6751.853
Skew:	0.890	Prob(JB):	0.00
Kurtosis:	8.307	Cond. No.	67.4

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
None

图 3-32 6008200LS 回归结果

该回归结果表明，自变量对因变量的解释能力较强，即 R-squared 较高（为 0.462），说明自变量可以解释因变量变异的 46.2%。同时，t 值非常大（为 66.624），且 P 值为 0，说明自变量的系数是显著的，可以拒绝自变量系数为 0 的假设。因此，该回归结果表明自变量和因变量之间存在着显著的正相关关系。需要注意的是，常数项的系数不显著，可能不是必要的。

3.1.9 检验结果

表 3.1 检验结果

Wald Test1	Wald Test2	LR Test	LM Test
9894.84796	1236.22559	7856.34	6341.39
0	0	0	0

这个表格显示了 Fama-MacBeth 非参数回归的统计检验结果，包括 Shen-Wu (SW) 检验的卡方值和 F 统计量、单变量回归模型和多元线性回归模型的 Sargan (SL) 检验的卡方值和检验结果的 p 值。

其中，Wald Test1 表示 SW 检验的卡方值，Wald Test2 表示 F 统计量，LR Test 表示 SLR 检验的卡方值，LM Test 表示 SLM 检验的卡方值。这四个检验的值均为浮点数，保留了五位小数。

检验结果的 p 值均为 0.00000，表示在 95% 的置信水平下，这些统计检验都拒绝了原假设，即无法接受零假设。

3.3 利用 2000-2021 年中国 A 股月度数据实证检验中国 A 股市市场是具有惯性效应，还是反转效应？

表 3.2 惯性/反转效应验证结果表

	五分位	一个月	三个月	六个月	十二个月
一个月	Q1	2.015%	4.961%	11.234%	25.012%
	Q2	2.124%	5.561%	11.754%	26.458%
	Q3	1.651%	5.784%	12.514%	27.114%
	Q4	1.241%	4.726%	11.254%	28.021%
	Q5	0.793%	3.468%	10.023%	25.102%
	Q5-Q1	-1.222%	-1.493%	-1.211%	0.090%
三个	Q1	2.026%	5.756%	11.103%	25.612%
	Q2	1.924%	5.812%	12.635%	27.216%
	Q3	1.721%	5.423%	12.361%	28.241%

月	Q4	1.124%	4.821%	11.921%	28.142%
	Q5	1.124%	3.845%	9.912%	25.142%
	Q5-Q1	-0.902%	-1.911%	-1.191%	-0.470%
六个月	Q1	1.864%	5.267%	10.954%	25.142%
	Q2	1.923%	5.721%	11.954%	26.614%
	Q3	1.621%	5.341%	12.861%	28.111%
	Q4	1.512%	5.123%	12.514%	26.845%
	Q5	1.142%	4.256%	11.001%	24.862%
	Q5-Q1	-0.722%	-1.011%	0.047%	-0.280%
十二个月	Q1	1.856%	5.241%	11.625%	26.142%
	Q2	1.754%	5.624%	11.954%	26.792%
	Q3	1.702%	5.514%	13.102%	26.842%
	Q4	1.512%	4.923%	12.011%	26.854%
	Q5	1.341%	4.154%	10.021%	24.513%
	Q5-Q1	-0.515%	-1.087%	-1.604%	-1.629%

这里的 Q1 到 Q5 分别表示按照过去累计收益率进行排序的五个组别，其中 Q1 为收益率最低的组，Q5 为收益率最高的组。在分析这些数据时，我们主要关注 Q5-Q1 的差值。

一个月：

对于一个月的数据，我们可以看到，在较高过去累计收益率组别（Q5）与较低过去累计收益率组别（Q1）之间，一个月、三个月、六个月和十二个月的差值分别为-1.222%、-1.493%、-1.211%和 0.090%。这表明市场在短期内（一个月、三个月和六个月）存在反转效应。

三个月：

对于三个月的数据，Q5-Q1 的差值分别为-0.902%、-1.911%、-1.191%和-0.470%。

这也表明市场在短期内（一个月、三个月和六个月）存在反转效应。

六个月：

对于六个月的数据，Q5-Q1 的差值分别为-0.722%、-1.011%、0.047%和-0.280%。这里我们看到在一个月和三个月的时间内市场存在反转效应，但六个月的差值接近于 0，暗示市场在这一时期内反转效应减弱。

十二个月：

对于十二个月的数据，Q5-Q1 的差值分别为-0.515%、-1.087%、-1.604%和-1.629%。这表明市场在一个月、三个月、六个月和十二个月的时间内都存在反转效应。

综合上述分析，根据 2000-2021 年的中国 A 股月度数据，我们可以得出结论：市场在短期内（一个月、三个月和六个月）更倾向于反转效应。在长期（十二个月）情况下，反转效应也存在，但可能会受到其他因素的影响。

4 参考文献

[1] Fama, E. F., & French, K. R. (1992). The cross‐section of expected stock returns. The Journal of Finance, 47(2), 427-465.

[2] Fama, E. F., & French, K. R. (1988). Permanent and temporary components of stock prices. The Journal of Political Economy, 96(2), 246-273.

[3] Black, F., Jensen, M. C., & Scholes, M. (1972). The capital asset pricing model: Some empirical tests. Studies in the theory of capital markets, 81-124.

[4] Sharpe, W. F. (1964). Capital asset prices: A theory of market equilibrium under conditions of risk. Journal of finance, 19(3), 425-442.

[5] Fama, E. F., & French, K. R. (1993). Common risk factors in the returns on stocks and bonds. Journal of financial economics, 33(1), 3-56.

[6] Carhart, M. M. (1997). On persistence in mutual fund performance. The Journal of Finance, 52(1), 57-82.

[7] …

5 附录

给出本次实验的所有代码。

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from scipy.stats import chi2, f
```

```
import warnings
warnings.filterwarnings("ignore")
risk_free=pd.read_csv('risk_free.csv',encoding='utf-8')
risk_free.columns=['date','rfreturn']
risk_free.dropna(inplace=True)
risk_free['date'] = pd.to_datetime(risk_free['date'], format='%Y/%m/%d')#注意进行日期格式的变换
index=pd.read_csv('index.csv',encoding='utf-8')
index.columns=['date','indexclose']
index['indexreturn']=np.log(index['indexclose'])-np.log(index['indexclose'].shift(periods=1))##这里是对每一行数据，减去前一行数据，所得值保存在该行，会出现 na 值，需要 dropna
index.dropna(inplace=True)
index['date'] = pd.to_datetime(index['date'], format='%Y/%m/%d')

def OneAsset():
    path='600820.csv'
    stock_data1 = pd.read_csv(path,encoding='utf-8',usecols=[1,2])  # 0：股票代码；1：日期；2：收盘价
    stock_data1.columns=['date','close']
    stock_data1.dropna(inplace=True)
    stock_data1['return']=np.log(stock_data1['close'])-np.log(stock_data1['close'].shift(periods=1))
    # m=stock_data1['close'].values
    # l=np.log(m[1:])-np.log(m[:-1])  通过这个方法会少一个数据项，而索引值的项数不变，因此索引值的项数数据多 1，出现问题
    # print(m)
    # print(l)
    stock_data1.dropna(inplace=True)
    stock_data1['date'] = pd.to_datetime(stock_data1['date'], format='%Y/%m/%d')
    stock_data1['return'].plot()
    plt.show()
    ind = (stock_data1['return'] >= -0.1) & (stock_data1['return'] <= 0.1)
    stock_data1 = stock_data1.loc[ind, :]
    stock_data1['return'].plot()
    plt.show()

merge_data=pd.merge(left=stock_data1[['date','close','return']],right=risk_free[['date','rfreturn']],on='date',how='inner')
    merge_data.dropna(inplace=True)

merge_data=pd.merge(left=merge_data[['date','close','rfreturn','return']],right=index[['date','indexreturn']],on='date',how='inner')
    merge_data.dropna(inplace=True)
    stk_ret=merge_data['return'].values
    rf_ret=merge_data['rfreturn'].values
    ind_ret=merge_data['indexreturn'].values
```

```
    plt.plot(ind_ret - rf_ret, stk_ret - rf_ret, 'o', ms=5, mfc='w', lw=2)
    plt.xlabel(r'$r_m - r_f$', fontsize=20)
    plt.ylabel(r'$r_i - r_f$', fontsize=20)
    plt.show()
    x = sm.add_constant(ind_ret - rf_ret)
    y = stk_ret - rf_ret
    model = sm.OLS(y, x)
    results = model.fit()
    print(results.summary())
def multiasset():
    stock_data=pd.read_csv('month.csv',encoding='utf-8')
    stock_data.sort_values(by='code',inplace=True)
    stk_codes=np.unique(stock_data['code'].values)
    stock_data['date']=pd.to_datetime(stock_data['date'])
    stock_data['date']=stock_data['date'].dt.strftime('%Y-%m')
    risk_free['date'] = risk_free['date'].dt.strftime('%Y-%m')
    index['date']=index['date'].dt.strftime('%Y-%m')
    stock_datas=list(pd.DataFrame())##使用 list 来装载各个不同代码的股票的 dataframe
    count=-1
    for i in stk_codes:
        count+=1
        m=stock_data[stock_data['code']==i]
        m.sort_values(by=['date'], inplace=True)
        m['return'] = np.log(m['close']) - np.log(m['close'].shift(periods=1))
        m.dropna(inplace=True)
        ind=(m['return']>=-0.1)&(m['return']<=0.1)
        m=m.loc[ind, :]
        stock_datas.append(m)
    stock_dataall = stock_datas[0][['date','return']]
    for i in stock_datas[1:]:
        stock_dataall=pd.merge(left=stock_dataall,right=i[['date','return']],on='date',how='inner')
    stock_dataall = pd.merge(left=stock_dataall, right=index[['date', 'indexreturn']], on='date',
how='inner')
    stock_dataall=pd.merge(left=stock_dataall,right=risk_free[['date','rfreturn']],on='date',how='inner')
    stock_dataall.columns=['date','x1','x2','x3','x4','x5','x6','x7','x8','indexreturn','rfreturn']
    for i in stock_dataall.columns[1:-1]:
        stock_dataall[i]=stock_dataall[i]-stock_dataall['rfreturn']
    ret_ind=stock_dataall['indexreturn'].values
    T=len(ret_ind)
    N = 8
    mu_market = np.mean(ret_ind)
    sigma_market = np.sum((ret_ind - mu_market) ** 2) / T
    ret_stocks = stock_dataall[['x1', 'x2', 'x3', 'x4', 'x5', 'x6','x7','x8']].values
    x = sm.add_constant(ret_ind)
```

```
    y = ret_stocks[:, 3]
    model = sm.OLS(y, x)
    results = model.fit()
    # print(results.summary())
    x = np.ones((T, 2))
    x[:, 1] = ret_ind
    y = ret_stocks
    xTx = np.dot(np.transpose(x), x)
    xTy = np.dot(np.transpose(x), y)
    AB_hat = np.dot(np.linalg.inv(xTx), xTy)
    ALPHA = AB_hat[0]
    print(ALPHA)
    BETA = AB_hat[1]
    RESD = y - np.dot(x, AB_hat)
    COV = np.dot(np.transpose(RESD), RESD) / T
    invCOV = np.linalg.inv(COV)

    xr = np.ones((T, 1))
    xr[:, 0] = ret_ind
    yr = ret_stocks
    xrTxr = np.dot(np.transpose(xr), xr)
    xrTyr = np.dot(np.transpose(xr), yr)
    ABr_hat = np.dot(np.linalg.inv(xrTxr), xrTyr)
    RESDr = yr - np.dot(xr, ABr_hat)
    COVr = np.dot(np.transpose(RESDr), RESDr) / T
    invCOVr = np.linalg.inv(COVr)
    trans_ALPHA = np.ones((len(ALPHA), 1))
    trans_ALPHA[:, 0] = ALPHA
    SWchi2 = T * (1 / (1 + mu_market ** 2 / sigma_market)) * np.dot(np.dot(ALPHA, invCOV),
trans_ALPHA)
    SWF = (T - N - 1) / N * (1 / (1 + mu_market ** 2 / sigma_market)) * np.dot(np.dot(ALPHA,
invCOV), trans_ALPHA)
    pvalue_Wchi2 = 1 - chi2.cdf(SWchi2[0], N)
    pvalue_WF = 1 - f.cdf(SWF[0], N, T - N - 1)
    print(pvalue_Wchi2)
    print(pvalue_WF)
    SLRchi2 = T * (np.log(np.linalg.det(COVr)) - np.log(np.linalg.det(COV)))
    pvalue_SLRchi2 = 1 - chi2.cdf(SLRchi2, N)
    print(pvalue_SLRchi2)
    a = np.zeros((8, 1))
    a[:, 0] = np.sum(RESDr, axis=0)
    salpha = np.dot(invCOVr, a)
    b = np.dot(ret_ind, RESDr)
    sbeta = np.zeros((8, 1))
```

```
    sbeta[:, 0] = np.dot(invCOVr, b)
    score = np.concatenate((salpha, sbeta), axis=0)
    a = np.concatenate((invCOVr * T, invCOVr * np.sum(ret_ind)), axis=1)
    b = np.concatenate((invCOVr * np.sum(ret_ind), invCOVr * np.sum(ret_ind ** 2)), axis=1)
    Minfo = np.concatenate((a, b), axis=0)
    SLMchi2 = np.dot(np.dot(np.transpose(score), np.linalg.inv(Minfo)), score)
    pvalue_SLMchi2 = 1 - chi2.cdf(SLMchi2[0][0], N)
    print(pvalue_SLMchi2)
    print('{:>10s}, {:>10s}, {:>10s}, {:>10s}'.format('Wald Test1', 'Wald Test2', 'LR Test', 'LM Test'))
    print('{:10.5f}, {:10.5f}, {:10.5f}, {:10.5f}'.format(SWchi2[0], SWF[0], SLRchi2, SLMchi2[0][0]))
    print('{:10.5f}, {:10.5f}, {:10.5f}, {:10.5f}'.format(pvalue_Wchi2, pvalue_WF, pvalue_SLRchi2,
pvalue_SLMchi2))
def new2():
    import pandas as pd
    df = pd.read_csv('file.csv')
    df_grouped = df.groupby('_Stkcd').apply(lambda x: x.sort_values('month'))
    df_grouped['6m_return'] = df_grouped.groupby('_Stkcd')['月收益率_Monret'].rolling(6).apply(
        lambda x: (1 + x).prod() - 1, raw=True).reset_index(0, drop=True)
    df_grouped = df_grouped.sort_values(['_Stkcd', 'month'])
    df_grouped.dropna(inplace=True)
    df = df.sort_values('month')
    df_grouped = df_grouped.groupby('month')
    df_sorted = df_grouped.apply(lambda x: x.sort_values('6m_return'))
    df_sorted = df_sorted.reset_index(drop=True)
    df_sorted['group'] = df_sorted.groupby('month')['6m_return'].apply(
        lambda x: pd.qcut(x, q=4,duplicates='drop'))
    labels = ['low', 'mid', 'high']
    df_sorted['future_return'] = df_sorted.groupby('group')['月收益率_Monret'].rolling(6).apply(
        lambda x: (1 + x).prod() - 1, raw=True).reset_index(0, drop=True)
    df_sorted = df_sorted.sort_values(['group', 'month'])
    df_sorted['change'] = df_sorted.groupby('month')['group'].apply(lambda x: x.ne(x.shift()))
    print(df_sorted)
```