跨境 ETF 套利策略设计

摘要

本文基于对 ETF 流动性和收益风格的深入分析，综合考虑分类、选股、择时等因素，逐步构建了基于 LCAPM 和修正 LCAPM 模型的“相关系数—协整”法选股策略，以及 GARCH 与布林带通道配对交易套利择时止损策略，并针对不同市场环境提出了相应的 ETF 套利策略。通过可靠的模型求解结果和详细解释，为投资者提供了在有限决策时间内多维度综合考虑进行套利决策的支持。

针对问题一，首先对交易数据预处理，基于 **3σ原则**剔除离群值与异常值，通过基于**流动性风险溢价**的 **LCAPM 模型**来建立一二级市场间套利模型，并计算一二级市场间的预期收益率，选出最适合套利的 10 只 ETF 基金。

针对问题二，在问题一基础上额外考虑二级市场间 T0 交易 ETF 波动性风险对预期收益的影响，通过添加**波动性风险因子**修正 LCAPM 模型；并通过**“相关系数-协整”**法选股模型的构建，选出最适合套利的 10 只跨境 ETF 基金。最后基于**布林带通道**的套利模型，给出在实际操作中择时的策略。

针对问题三，通过 **Pearson 系数**在最适合套利的跨境 ETF 与四只股指期货的交易组合进行筛选，在建立 **OLS 回归方程**后，继续通过后续 **ADF 检验**与**残差平稳性**的分析选出最适合套利的资产对。最后通过 **GARCH 的配对交易套利**模型的构建，**动态调整**套利区间参数 K 与a，从而达到**收益最大化**的目的。

针对问题四， 通过前三问的建模与求解，最终选定**香港证券 ETF(513090)**作为**恒生科技 ETF(513130)**的配对资产，通过模型求解后，套利胜率达 75%，年化收益率可达 209.0069%，最大回撤为 11.71%，套利总利润 187784.36。选定**IF 沪深 300 指数**与**新经济 ETF (159822)**构建配对套利资产组合，套利胜率达 75%，年化收益率达 106.89%，最大回撤为 8.38%，套利净利润 24182.28。具有较好的表现。

关键词：LCAPM 模型、流动性溢价、波动性风险因子、协整法检验、布林带通道、GARCH 模型

一、问题重述

1.1 问题背景

交易型开放式指数基金（ETF）是一种追踪“标的指数”变化的基金。它可以在二级市场按市场价格买卖 ETF 份额，交易“一篮子股票”，同时也可以通过实物方式随时向基金管理公司申购或赎回基金份额。在一级市场，ETF 以参考单位基金净值（IOPV）进行报价，可称之为价值。而在二级市场中，ETF 的价格则由市场供需决定。当一级市场和二级市场的价格与价值出现过度偏离时，就可以进行套利。套利的基本形式主要有两种：

折价套利： 当 IOPV 大于 ETF 价格时，可以在二级市场买入 ETF，然后在一级市场赎回 ETF，得到一篮子股票。最后将这一篮子股票在二级市场上卖出，减去成本即为利润。

溢价套利： 当 IOPV 小于 ETF 价格时，可以在二级市场上买入一篮子股票，然后在一级市场申购 ETF。最后在二级市场上卖出 ETF，减去成本即为利润。

这两种套利策略帮助投资者在一级市场和二级市场之间实现收益。

1.2 问题要求

附件表单提供了 886 条 ETF 基金交易数据，其中包括 343 条深圳交易所上市 ETF 基金交易数据和 540 条上海交易所上市 ETF 基金交易数据。依据附件数据相关信息与各交易所官网提供资料建立模型，解决下列四个问题：

问题 1：对附件给出的基金进行分析，并基于此建立一级市场与二级市场之间的套利模型，比较筛选出最适合进行套利的前十只基金。

问题 2：跨境 ETF 可实现 T+0 交易，即实现当日的买卖。修改问题一的策略，构建跨境 ETF 套利的模型，再比较筛选出最适合进行套利的前十只基金。

问题 3：股指期货中有上证 50、沪深 3000、中证 500、中证 1000 四个指数的期货。通过分析，建构跨境 ETF 与股指期货的跨市场套利模型，并比较筛选出最适合跨市场套利的跨境 ETF 和股指期货组合。

问题 4：基于问题二建立的跨境 ETF 套利模型对恒生科技 ETF（513130）进行实测，再基于问题三的跨市场套利模型对跨境 ETF 和股指期货组合进行实测，并分析各评价指标，制作实测报告。

二、模型假设

本文提出以下合理假设：

假设一：交易市场是弱有效的，由于信息差、交易者情绪等原因，存套利机会。

假设二：发生的买卖都能迅速完成，不存在“卖空“等情况。

假设三：不考虑中间因节假日等休盘的数据间隔，将数据视为连续的序列。

假设四：假设所有交易的交易费率为期货 0.005%，股票为 0.01%，价格滑点为 0.01。

三、符号说明

表 1 符号说明表

符号	含义
ω_t^i	ETFi 在 t 时刻投资组合中的权重
f_t^i	ETFI 在 t 时刻的对应的回报指数
$E(r_t^i)$	预期收益率
v_t^i	波动率
c_t^i	非流动性指标
c_t^M	每日市场波动
r_t^M	每日市场收益

*其余符号在文中说明

四、问题分析

4.1 问题一分析

针对问题一，基于所提供的 ETF 基金数据，从而确定最适合套利的十只基金。为实现这一目标，我们将依据历史数据设计最佳套利指标，使用从 2023 年 8 月 1 日至 2023 年 10 月 31 日的 ETF 基金信息，计算超过八百只 ETF 基金的多

项指标，并进行排名。我们的研究重点在于构建一级市场和二级市场之间的套利模型，以挑选出最具潜力的 ETF 基金。套利的适合程度将通过各基金的收益资产水平来衡量，该指标将以预期收益率为依据。在数据分析前，我们将先对所提供数据进行预处理，以剔除其中的异常值。随后，我们将进行数据可视化和回归分析，以便更好地理解基金表现和市场趋势。

本题的核心目标在于构建一个适用于 ETF 基金的套利模型和策略。为实现这一目标，我们选择了基于 ETF 流动性构建流动性风险溢价的 LCAPM（Liquidity-augmented Capital Asset Pricing Model）模型。在该模型中，我们将追踪误差作为被解释变量，并通过取 ETF 价格与其净值的对数差来度量。同时，我们将非流动性指标 ILLIQ 作为解释变量，用以衡量 ETF 的日度流动性。通过此模型，我们将从流动性的角度评估不同资产收益的能力，为套利提供依据。

具体而言，我们将分析所提供的八百多只基金的预期收益率，并以此为依据，选取其中预期收益最高的前十只基金进行投资套利。这一策略将充分利用 LCAPM 模型中的流动性风险溢价，以指导投资决策，从而实现更为精准和可靠的套利操作。

4.2 问题二分析

在第二题的研究中，我们将依托第一题的基础，专注于建立适用于跨境 ETF 的套利模型。跨境 ETF 与境内 ETF 相似之处在于它们都涉及一级市场与二级市场之间的套利机会。然而，跨境 ETF 相较于境内 ETF 具有特殊之处，即涉及二级市场与二级市场之间的套利机会。为了充分利用这一特点，我们将扩展第一题的套利模型，以包括这种二级市场间的套利机会。

具体而言，在套利模型中，我们将保留第一题中的一级市场与二级市场之间的套利部分，并且在此基础上剔除原模型中的 β^{3i}与 β^{4i}，即市场流动性与个体资产收益之间的关系以及个体资产流动性与市场收益之间的关系。通过修正后的 LCAPM 模型，我们将计算出收益的期望值$E_2(r_t^i)$。接着，我们将将这一期望收益值与第一题得到的$E_1(r_t^i)$相加，得到综合考虑一级市场与二级市场、以及二级市场与二级市场套利机会的期望收益率。

在这一模型的基础上，我们将选择期望收益率最大的前十只 ETF 基金进行深入研究。在这十只基金中，我们将再次筛选出期望收益率最大的一只 ETF 基金。同时，我们将进行对 ETF 收益率的 Pearson 相关性分析，在剩余的 87 只跨境 ETF 中找出与选定 ETF 基金相关性最高的一只。最后，我们将建立基于布林带通道套利模型的套利策略，以实现更为精准和高效的跨境 ETF 套利操作。

4.3 问题三分析

在问题三的研究中，我们的主要任务是基于跨境 ETF 与国际市场的联动性，以及国际市场对 A 股的影响，建立跨境 ETF 与股指期货的跨市场套利模型。为达到这一目标，我们将采用遍历的方法，将四只股指期货与跨境 ETF 两两组合，进行详尽的相关性分析。我们的目标是找出具有满足套利要求、并拥有长期均衡关系的配对组合。

首先，我们将运用协整配对的方法，在遍历的过程中找出最适合套利的跨境 ETF 与四只股指期货的交易组合。这一步骤将确保我们选择的配对组合具有稳定的协整关系。在确定了最优的配对组合后，我们将基于 GARCH 模型进行套利策略的分析。我们将运用该模型对跨境 ETF 和股指期货的价格波动进行预测，并根据预测结果制定最优的套利策略。这个过程将涉及到对套利机会的实时监测和调整，以确保套利利润的最大化。

4.4 问题四分析

结合二三问建立的跨境 ETF 套利模型，对恒生科技 ETF(513130)，以及任务三选出的跨境 ETF 和股指期货组合进行实测，将测得的策略收益、最大回撤及收益率曲线等评价指标制作成实测报告。

五、问题一的模型建立与求解

5.1 基于流动性风险溢价的 LCAPM 模型的构建

5.1.1 数据预处理

本题附件表单给出了 2023 年 8 月 1 日至 223 年 10 月 31 日每天 9:30-11:30 和 13:00-15:00 间每分钟 ETF 基金的收盘价、开盘价、最高价、最低价、成交额和成交量等信息，分析后发现每单交易均价存在异常数据。根据经济学理论，

价格围绕价值波动，但是少量交易均价数据偏离价值太远，且成交手数多为 1 手，合理认为这些数据不具备分析价值。于故使用正态分布的 3σ原则对数据进行去极值处理，当每天的 ETF 数据点，即总交易额除以成交手所得数据超出了均值加减 3 个标准差的范围时将其数据归零。

5.1.2 构建基于流动性风险溢价的 LCAPM 模型

LCAPM 考虑了与流动性风险和市场风险相关的 3 个风险溢价因子，并认为流动性调整的资产收益和与考虑了市场流动性后的市场收益呈线性关系。因为 ETF 是像股票一样在交易所交易的，因此 ETF 的市场流动性也能影响单个 ETF 的预期收益。

（1）预期收益率$E\left(r_t^i\right)$

$$E\left(r_t^i - r^f\right) = E\left(c_t^i\right) + \left(\beta^{1i} + \beta^{2i} - \beta^{3i} - \beta^{4i}\right)E(r_t^M - c_t^M - r^f) \tag{5.1}$$

其中 β^{1i}是传统 CAPM 模型的部分，代表了资产收益和市场收益之间的关系。β^{2i}表明了市场流动性与个体资产流动性之间的关系；β^{3i}是市场流动性和个体资产收益之间的关系；β^{4i}是个体资产流动性和市场收益之间的关系。r^f为无风险收益率，取十年期国债$r^f = \frac{2.6658\%}{365} = 0.0073036\%$。

$$\beta^{1i} = \frac{cov\left(r_t^i, r_t^M\right)}{var(r_t^M - c_t^M)} \tag{5.2}$$

$$\beta^{2i} = \frac{cov\left(c_t^i, c_t^M\right)}{var(r_t^M - c_t^M)} \tag{5.3}$$

$$\beta^{3i} = \frac{cov\left(r_t^i, c_t^M\right)}{var(r_t^M - c_t^M)} \tag{5.4}$$

$$\beta^{4i} = \frac{cov\left(c_t^i, r_t^M\right)}{var(r_t^M - c_t^M)} \tag{5.5}$$

该模型很好地将流动性的风险分解为多个部分，将资产收益与流动性相结合，符合本题题意，本文即通过此模型来验证流动性风险与资产收益。

（2）非流动性指标 ILLIQ

$$ILLIQ = c_t^i = \frac{1}{D_{i,t}} \sum_{d=1}^{D_{i,t}} \frac{\left|R_{i,t,d}\right|}{VOLD_{i,t,d}} = \frac{1}{8} \sum_{h=1}^{8} \frac{|P_{max} - P_{min}|}{SUM_h} \tag{5.6}$$

在(5.5)中，ILLIQ 表示非流动性指标，非流动性指标可以验证交易金额对价格变动的影响程度。$D_{i,t}$为标的 i 在 t 月有效交易天数，令其为从每日的 9:30-11:30 和 13:00-15:00 开始计算的 8 个 0.5h；$|R_{i,t,d}|$为标的资产 i 在 t 月第 d 个交易日的回报率，令其为$|P_{max} - P_{min}|$，0.5h 内的最高成交额减去最低成交额的绝对值；$VOLD_{i,t,d}$为标的 i 在 t 月第 d 个交易日的交易额，令其为SUM_h，该 0.5h 内的成交总额。若 ILLIQ 指标越高，ETF 的流动性越差，排名较前的 ETF 流动性更强，ILLIQ 更小。

每日 ETF 的流动性为：

$$c_t^M = \sum \omega_t^i c_t^i = \frac{1}{800+} \sum_{i=1}^{800+} c_t^i \tag{5.7}$$

其中ω_t^i为 ETFi 在 t 时刻投资组合中的权重。

（3）每日市场收益r_t^M

$$r_t^M = \sum_{i=t}^{N_t} \omega_t^i f_t^i = \sum_{i=1}^{800+} f_t^i \tag{5.8}$$

$$f_t^i = \frac{\left|P_{arg} - IOPV_{开}\right|}{IOPV_{开}} \tag{5.9}$$

每日市场收益的计算方法为构成该投资组合的每只 ETF 的对应指数的回报的加权平均值。每只 ETF 追踪的指数并不在市场上交易，使用标的指数收益来计算市场收益可以避免因交易而产生的潜在计量误差。ω_t^i为 ETFi 在 t 时刻投资组合中的权重；f_t^i为 ETFI 在 t 时刻的对应的指数回报，即折溢率；P_{arg}为 SUM 除以成交手数目；$IOPV_{开}$为开盘时该 ETF 的价值。

5.2 基于流动性风险溢价的 LCAPM 模型的求解

当每只 ETF 基金权重相等时，得出每只 ETF 的预期收益率$E\left(r_t^i\right)$及其排序，并从中选出 10 只预期收益率最高的 ETF 基金，所求结果如下：

表 2 每只 ETF 基金代码的预期收益率及排序表

排序	ETF 基金代码	预期收益率
1	516730.SH	0.677613255
2	159001.SZ	0.610039028
3	517390.SH	0.489304376
4	511090.SH	0.476889995
5	511020.SH	0.476365103
6	512330.SH	0.474578669
7	515530.SH	0.468306246
8	159649.SZ	0.459259327
9	560950.SH	0.446720498
10	513390.SH	0.444503863

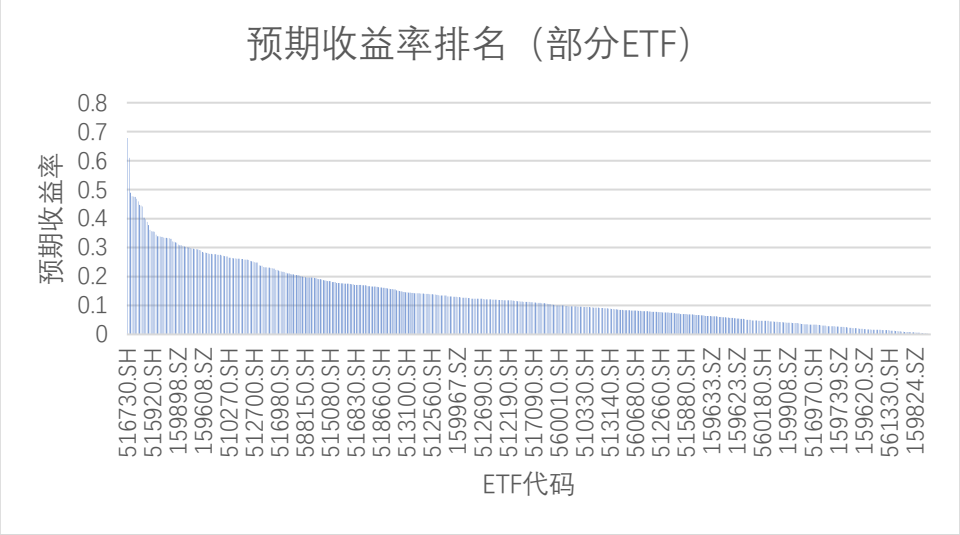

图 1 ETF 基金预期收益率图

六、问题二的模型建立与求解

6.1 基于 LCAPM 修正模型与 GARCH 的跨境 ETF 套利模型的构建

6.1.1 数据预处理

在第一题数据预处理的基础上，我们从上海交易所和深圳交易所的官网数据披露中查找跨境 ETF 的基金代码，并在题目所给附件中对照筛选出相关的跨境 ETF 基金，最终找到跨境 ETF 基金共 88 只，下面对这 88 只跨境 ETF 基金进行讨论。

6.1.2 基于“相关系数—协整”法选股与 GARCH 的跨境 ETF 套利模型的建立

（1）二级市场间的预期收益率$E_2(r_t^i)$

$$E(r_t^i) = E(v_t^i) + r^f + (\beta^{1i} + \beta^{2i})E(r_t^M - r^f) \tag{6.1}$$

其中β^{1i}是传统 CAPM 模型的部分，代表了资产收益和市场收益之间的关系。β^{2i}表明了市场流动性与个体资产流动性之间的关系；β^{3i}是市场流动性和个体资产收益之间的关系；β^{4i}是个体资产流动性和市场收益之间的关系，β^{3i}和β^{4i}在二级市场的范围中与前两个指标重复，故在本修正模型中剔除；r^f为无风险收益率，取值同模型一；v_t^i为波动率; f_t^i为收益率；P_{arg}是$\frac{\sum 成交额}{\sum 交易手量}$，具体所在区间由角标决定，t 为一天，h 为 30min。

$$\beta^{1i} = \frac{cov(r_t^i, r_t^M)}{var(r_t^M - c_t^M)} \tag{6.2}$$

$$\beta^{2i} = \frac{cov(c_t^i, c_t^M)}{var(r_t^M - c_t^M)} \tag{6.3}$$

该模型很好地将流动性的风险分解为多个部分，将二级市场间资产收益与b波动相结，即通过此模型来验证二级市场间 b 波动性性风险与资产收益。

（2）波动率v_t^i

$$v_t^i = \sqrt{\frac{\sum_{n=1}^{8}(x - f_t^i)^2}{8-1}} \tag{6.4}$$

$$x = \frac{\left|P_{\arg h} - P_{\arg h-1}\right|}{P_{\arg h-1}} \tag{6.5}$$

$$v_t^M = \frac{1}{N}\sum_{i=1}^{88} v_t^i \tag{6.6}$$

（3）每日市场收益r_t^M

$$r_t^M = \sum_{i=t}^{N_t} \omega_t^i f_t^i = \frac{1}{N}\sum_{i=1}^{88} f_t^i \tag{6.7}$$

$$f_t^i = \frac{\left|P_{\arg t} - P_{\arg t-1}\right|}{P_{\arg t-1}} \tag{6.8}$$

每日市场收益的计算方法为构成该投资组合的每只 ETF 的对应指数的回报的加权平均值。ω_t^i为 ETFi 在 t 时刻投资组合中的权重；f_t^i为 ETFI 在 t 时刻的对

应的指数回报，即折溢率；P_{arg}是$\frac{\sum 成交额}{\sum 交易手量}$，具体所在区间由角标决定，t 为一天，h 为 30min。

（4）总预期收益率$E(r_t^i)$

总预期收益率为第一题所得一二级市场间预期收益率$E_1(r_t^i)$加上本题求得的二级市场间预期收益率$E_2(r_t^i)$，将所求得的和进行排位，取最大的前十只作为最适合套利的跨境 ETF 基金。

（5）建设跨境 ETF 套利模型

在上述选出的十只跨境 ETF 中，再选出一只最适合套利的跨境 ETF，接着在剩下的 87 只跨境 ETF 中，使用遍历的方法，对两只 ETF 的价格与收益率进行 Pearson 相关性分析，在另外 87 只跨境 ETF 中找出相关性最高的一只，再建立基于交易套利模型的套利策略。88 只跨境 ETF（部分）相关系数热图如下所示：

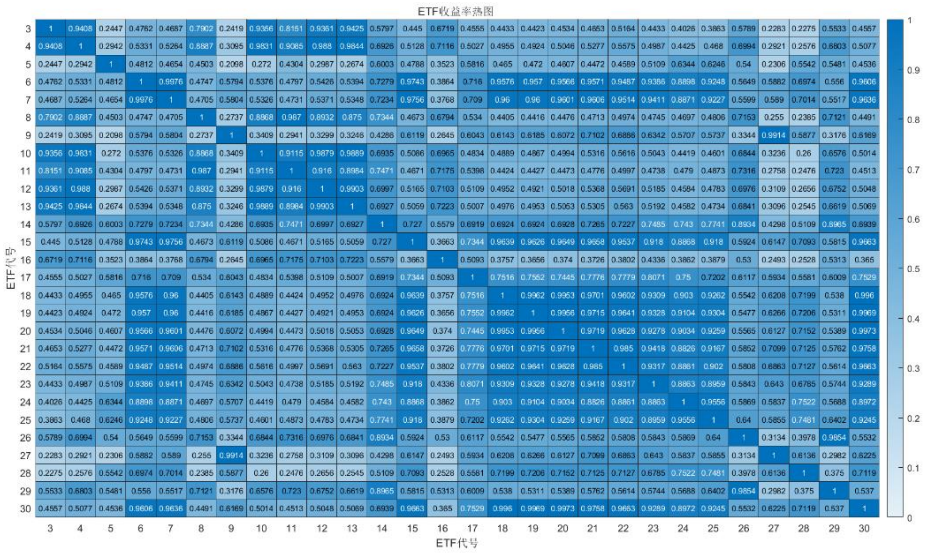

图 2 跨境 ETF（部分）相关系数热图

挑选出相关系数较高的两只代表性 ETF 的收益率与折溢率图如下：

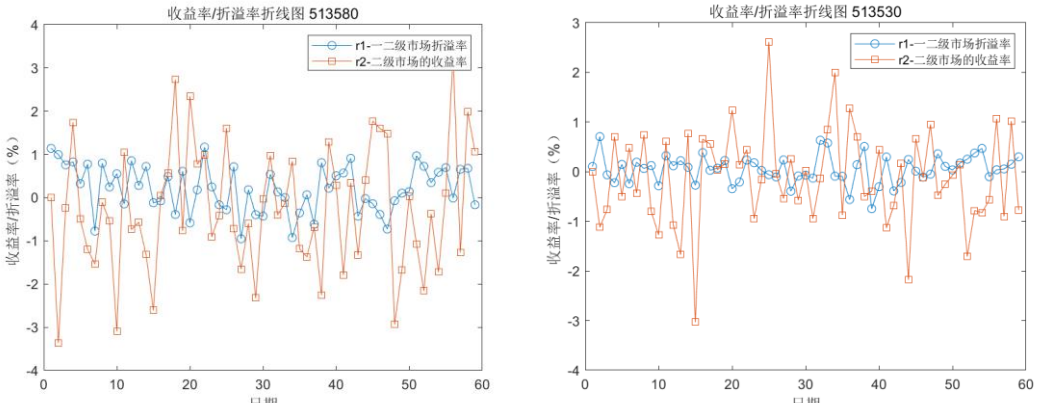

图 3 两只代表性 ETF 的收益率与折溢率图

① 一阶单整检验

由于收益率$r_t = \frac{P_t - P_{t-1}}{P_t}$， 在进行对两只选定的跨境 ETF 的协整检验前，我们需要确保它们的对数价格时间序列是一阶单整的。为了实现这一目标，我们首先使用最小二乘法构建回归方程，观察 $\ln P_t$与其一阶差分序列是否通过了 ADF 检验。其中，P_t表示每日交易价格的平均值。

ADF 检验的模型为：

$$\Delta y_t = \alpha + \beta_t + \gamma y_{t-1} + \sigma_1 \Delta y_{t-1} + \cdots + \sigma_p \Delta y_{t-p} + \beta x_t + \varepsilon_t \quad (6.9)$$

原假设 H_0：$\gamma = 0$，即序列存在单位根，表示序列非平稳。

备择假设H_1：$\gamma < 0$，即序列不存在单位根，表明序列平稳。

$$\text{统计量} DF = \frac{\gamma}{SE(\gamma)} \quad (6.10)$$

在给定的显著水平下，将计算出来的统计量 DF 值与临界值相比，若小于临界值，则该序列为平稳的时间序列；反之，时间序列不平稳。

② E-G 协整检验

若 $\ln P_t$序列是非平稳的，则考虑它的一阶差分序列，若其平稳，则说明 ETF 与股指期货的对数收盘价是一阶单整，可以进行下一步协整检验。

本文研究对象为两只跨境 ETF 的股价序列，故 k = 2。假定资产 X 与 Y 的时间序列分别为$\{ lnP_t^X \}$、$\{ lnP_t^Y \}$，且均为一阶单整序列。采用 OLS 法回归，以资产 X 价格对数$\{ lnP_t^X \}$为因变量，资产 Y 价格对数$\{ lnP_t^Y \}$为自变量，则回归方程如下：

$$lnP_t^Y = \beta lnP_t^X + \varepsilon_t + \alpha \quad (6.11)$$

得残差序列表达式为：

$$\varepsilon_t = lnP_t^Y - \beta lnP_t^X - \alpha \quad (6.12)$$

对残差序列$\{\varepsilon_t\}$进行 ADF 检验，检验是否为平稳时间序列。若残差序列$\{\varepsilon_t\}$通过 ADF 检验，是平稳时间序列，则说明资产 X 与 Y 的价格时间序列具有协整关系。

当该只 ETF 通过协整检验后，可以构成价差序列，同时交易头寸是由协整回归系数决定的，进而能够进行后续统计套利过程。

根据 ETF 与股指期货的对数价格序列构造的 OLS 回归方程，构造价差序列：

$$Spread_t = lnP_t^Y - lnP_t^X = \alpha + \varepsilon_t \tag{6.13}$$

残差序列 $\{\varepsilon_t\}$ 是均值接近为 0 的平稳的时间序列，因此对价差序列去中心化处理，得到：

$$Mspread_t = Spread_t - mean(Spread_t) = \varepsilon_t \tag{6.14}$$

③ 布林带通道配对交易套利模型

布林带通道（Bollinger Bands）是约翰·布林带（John Bollinger）根据标准差原理设计出的一个技术分析指标。这个指标具有判断资产是否处于超买和超卖中以及反映未来资产价格走势的作用。布林带通道是一种由三条线组成的技术分析指标，包括上轨线、中轨线和下轨线。中轨线是移动均线，而上下轨线则在中轨线的基础上加减移动标准差得出。第 t 期中轨线计算公式为：

$$u_t = \frac{1}{\alpha}\sum_{i=t-(\alpha-1)}^{\tau} P_i \tag{6.15}$$

其中P_t表示资产第 t 期的价格，α表示移动均线的天数。第 t 期价格的移动标准差σ_t的计算公式为：

$$\sigma_t = \sqrt{\frac{i}{\alpha}\sum_{i=t-(\alpha-1)}^{t}(P_i - u_t)^2} \tag{6.16}$$

第 t 期上轨线的计算公式为：

$$up_t = u_t + K_1\sigma_t \tag{6.17}$$

第 t 期下轨的计算公式为：

$$down_t = u_t + K_1\sigma_t \tag{6.18}$$

其中，参数α和K_1影响了布林带通道的形状，移动均线天数α较小时，布林带走势陡峭，较贴合资产价格走势；而较大的α则使布林带走势平缓。标准差倍数K_1的增大使通道宽度增加，反之则变窄。基于布林带残差波动择时策略，通过设置标准差倍数λ作为建仓阈值，当价差突破布林带的上下轨时，表明存在套利机会，可进行套利交易。参数α和K_1的值可通过试错调整来优化策略效果。

6.2 基于 LCAPM 修正模型与 GARCH 的跨境 ETF 套利模型的求解

88 只跨境 ETF 中，求出第一部分一级市场和二级市场间预期收益率$E_1(r_t^i)$并进行排位，结果如图所示：

表 3 每只跨境 ETF 基金代码的预期收益率$E_1(r_t^i)$及排序表

排序	跨境 ETF 基金代码	预期收益率$E_1(r_t^i)$
1	513390.SH	0.353946
2	513990.SH	0.286799
3	513000.SH	0.261984
4	159822.SZ	0.236866
5	513230.SH	0.194995
6	159696.SZ	0.193617
7	513590.SH	0.190520
8	513030.SH	0.179949
9	513110.SH	0.156961
10	513560.SH	0.145024

88 只跨境 ETF 中，求出第二部分二级市场之间的预期收益率$E_2(r_t^i)$并进行排位，结果如图所示：

表 4 每只跨境 ETF 基金代码的预期收益率$E_2(r_t^i)$及排序表

排序	跨境 ETF 基金代码	预期收益率$E_2(r_t^i)$
1	513950.SH	0.018223
2	159742.SZ	0.017980
3	159747.SZ	0.017644
4	513040.SH	0.017613
5	159607.SZ	0.016973
6	513380.SH	0.016836
7	513150.SH	0.016836
8	159688.SZ	0.016558
9	513590.SH	0.016426
10	513260.SH	0.016335

当每只跨境 ETF 基金权重相等时，将第一部分一级市场和二级市场间预期收益率$E_1(r_t^i)$与第二部分二级市场之间的预期收益率$E_2(r_t^i)$进行求和，得出每只跨境 ETF 的总预期收益率$E(r_t^i)$及其排序，并从中选出 10 只预期收益率最高的跨境 ETF 基金，所求最终结果如下：

表 5 每只跨境 ETF 基金代码的总预期收益率$E(r_t^i)$及排序表

排序	跨境ETF基金代码	总预期收益率$E(r_t^i)$
1	513390.SH	0.359122
2	513990.SH	0.298588

3	513000.SH	0.268723
4	159822.SZ	0.246377
5	513230.SH	0.210071
6	513590.SH	0.206946
7	159696.SZ	0.196988
8	513030.SH	0.183834
9	513110.SH	0.161028
10	513560.SH	0.160383

七、问题三的模型建立与求解

7.1 基于“相关系数—协整”法选股与 GARCH 的配对交易套利模型的构建

7.1.1“相关系数—协整”法选股

该部分检验同模型二，检验对象改为股指期货与跨境 ETF 的组合。方法步骤同前，不再赘述。

7.1.2 GARCH 配对交易套利模型

在使用 GARCH 模型拟合序列时，序列必须是纯随机和异方差的。建模过程主要分为三步：

自相关分析： 首先，对残差序列$\{\varepsilon_t\}$进行自相关性分析。绘制自相关函数图（ACF 图），观察不同滞后阶数下的自相关系数，并确定合适的滞后阶数（n）。这有助于分析时间序列数据中的周期性和趋势。进行 F 检验，判断是否存在一阶自回归条件异方差，以确保充分挖掘原始序列包含的有效信息。根据分析结果建立 p 阶移动平均 AR 模型

ARCH-LM 效应检验： 进行 ARCH-LM 检验，确定残差是否存在 ARCH 效应（异方差性）。如果残差中存在 ARCH 效应，即残差的方差与滞后残差的平方存在关联，那么在回归中，滞后残差的平方项将具有显著的系数。这一步骤确保研究对象的序列中不存在 ARCH 效应。

最后进行基于 GARCH 模型的建模，确定公式中相关的系数α，β，α_0，μ_0，GARCH 模型公式如下：

$$r = \mu_0 + \alpha_t \tag{7.1}$$

$$\alpha_t = \varepsilon_t \sigma_t \tag{7.2}$$

$$\sigma_t^2 = \alpha_0 + \beta \mu_{t-1}^2 + \alpha \sigma_{t-1}^2 \tag{7.3}$$

对所求结果进行两次相关性检验，第一次通过 p 是否小于 0.05 检验上述系数是否显著；第二次通过对$\tilde{a}_t$和$\tilde{a}_t^2$的白噪声检验确认模型可以接受。

计算标准化残差：

$$\tilde{a}_t = \frac{a_t}{\sigma_t} \tag{7.4}$$

为了估计参数，可以假定初始的σ_t^2已知，递推计算后续的σ_t^2并计算条件似然函数，求条件似然函数的最大值点得到参数估计。

在统计套利过程中，影响套利收益的因素众多，其中关键在于交易头寸的开仓、平仓和止损信号。实现配对交易策略通常包括以下四个步骤，此处以交易比例为β的 X 和 Y 的组合为例：

(1)建仓操作（正向）：若价差在 t 时刻向上穿越 t 建仓线，即当$Mspread_t > \sigma_t$时，卖出 k 只 X 同时买入$k\beta$只 Y。在下一时刻，若价差反向向下回复到均值附近（即当 t+1 时刻$Mspread_{t+1} < \sigma_{t+1}$时），进行平仓获利操作，卖出 k 只 X 同时买入$k\beta$只 Y。

(2)平仓止损操作（正向）：若价差在 t 时刻向上穿越 t 建仓线，即当$Mspread_t > \sigma_t$时，卖出 k 只 X 同时买入$k\beta$只 Y。在下一时刻，若价差仍然继续向上穿越，即$Mspread_{t+1} > 2\sigma_{t+1}$时，偏离建仓平仓区间，立即进行平仓止损操作，卖出$k\beta$只 Y。

(3)建仓操作（反向）：若价差在 t 时刻向下穿越 t 建仓线，即当$Mspread_t < -\sigma_t$时，买入 k 只 X 同时卖出$k\beta$只 Y。在下一时刻，若价差反向向上回复到均值附近（即当 t+1 时刻$Mspread_{t+1} > -\sigma_{t+1}$时），进行平仓获利操作，卖出 k 只 X 同时买入$k\beta$只 Y。

(4)平仓止损操作（反向）：若价差在 t 时刻向下穿越 t 建仓线，即当$Mspread_t < -\sigma_t$时，卖出 k 只 X 同时买入$k\beta$只 Y。在下一时刻，若价差仍然继续向下穿越，即当$Mspread_{t+1} < -2\sigma_{t+1}$时，偏离建仓平仓区间，立即进行平仓止损操作，卖出$k\beta$只 Y。

则获利 M 可表示为：

$$M = kP_{t+1}^Y - k\beta P_t^X = kP_{t+1}^Y - k\beta P_{t+1}^X + k\beta P_{t+1}^X - k\beta P_t^X \tag{7.5}$$

即用$Mspread_{t+1}$减去前后价差。

表 6 具体交易规则表

序号	建仓	操作	平仓	操作	收益
1	上触发点	卖出资产组合	平仓线	买入资产组合	正收益
2			上止损线		负收益
3	下触发点	买入资产组合	平仓线	卖出资产组合	正收益
4			下止损线		负收益

7.2 基于“相关系数—协整”法选股与 GARCH 的配对交易套利模型的求解

首先将 88 只跨境 ETF 分别与四只股权期货做 Pearson 相关系数，选取对四只 ETF 相关系数均在 0.8 以上的 ETF 共 43 只，如下：

表 7 跨境 ETF 名称表

跨境 ETF 名称					
159605	159607	159688	159740	159741	159742
159747	159750	159822	159823	159850	510990
513010	513020	513040	513050	513070	513090
513130	513150	513160	513180	513220	513230
513260	513320	513330	513380	513550	513560
513580	513590	513600	513660	513690	513770
513860	513890	513900	513950	513960	513970
513980					

随后对这 43 只 ETF 作 ADF 检验，并于四只股权期货相比。结果显示均为一阶平稳序列，可以进行下一步检测。

然后，对 43 只 ETF 与股权期货分别构造 OLS 回归方程，剔除拟合系数 R2 在 0.80 以下的 ETF，剩下的都是拟合较好的 ETF。其中，有 6 只 ETF 通过检验，并且有 3 只 ETF 拟合系数均在 0.85 以上，标*注明，如下：

表 8 ETF 与股指期货关系表

代码	IC	IF	IH	IM
510990*	0.968013	0.985387	0.973962	0.922135
513600*	0.896114	0.924173	0.884353	0.864781
159822	0.890156	0.932285	0.822308	0.861094
159823*	0.870636	0.907696	0.852775	0.853788

159850	0.857334	0.89234	0.873019	0.843274
513900	0.879961	0.91241	0.806463	0.859388

最后，对残差进行 ADF 检验，结果显示如下：

表 9 残差的 ADF 检验表

代码	IC		IF		IH		IM	
说明	P	ADF	P	ADF	P	ADF	P	ADF
510990	0.0113	2.5676	0.1432	1.4212	0.0439	2.0053	0.0481	1.9643
513600	0.0018	3.2964	0.0040	2.9971	0.0183	2.3761	0.0033	3.0897
159822	0.0046	2.9146	0.0070	2.7604	0.0992	1.6162	0.0055	2.8369
159823	0.0025	3.2060	0.0056	2.8331	0.0400	2.0452	0.0020	3.2678
159850	0.0057	2.8243	0.0159	2.4298	0.0393	2.0537	0.0052	2.8536
513900	0.0022	3.2333	0.0056	2.8305	0.0637	1.8343	0.0020	3.2719

其中对于 IH 股指期货，残差检验均在 5%以上，故剔除。对于 510990，残差检验不通过，剔除。参照第二问的适宜套利的 ETF 排名后如下：

表 10 ETF 排名表

ETF 代码	513600	159822	159823	513900
绝对排名	62	4	55	40
相对排名	4	1	3	2

最后选择 159822 与 IF 作为套利的股指期货与跨境 ETF 组合。

表 11 资产对 159822—IF 的分析表

资产对 159822--IF	方程
OLS 回归方程	$ln\, p_t^Y = 0.7484\, ln\, p_t^X - 1.6356 + \varepsilon_t$
去中心化价差序列	$Mspread_t = ln\, p_t^Y - 0.7484\, ln\, p_t^X + 1.6356$

其中 0.7484 为该组合对下，交易阈值触发时，股指期货和 ETF 之间的交易比例。当建仓阈值触发时，每买入 1 单位 IF，卖出 0.7484 单位 159822。对组合的成交价格趋势图如下，发现确实相关性较大，走势波动基本一致，直观地验证了模型的有效性。

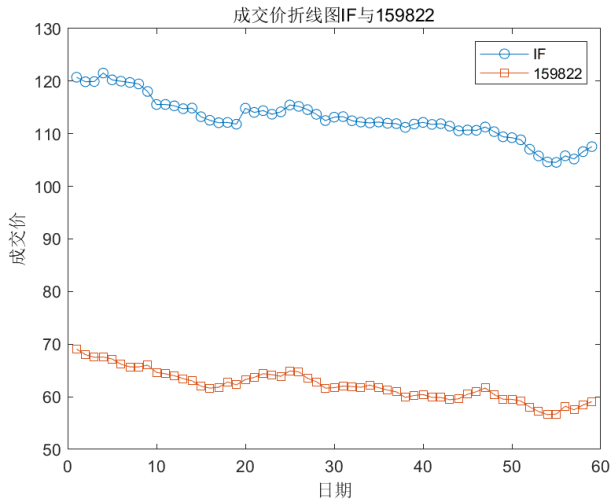

图 4 IF 与 159822 的成交价格趋势图

八、问题四的模型建立与求解

8.1 “相关系数—协整”法选股

题目中首先确定与恒生科技 ETF(513130)价格变化趋势相近的跨境 ETF,通过求解 Pearson 相关系数可得，在 87 只 ETF 中，与恒生科技正相关的有 74 只，负相关的有 8 只，剩余 6 只由于每日交易价格为负数，属于异常 ETF。将异常 ETF 与负相关的 ETF 提出后，按照排名选出如下几只 ETF：

表 12 ETF 代码与相关系数表

ETF 代码	159607	159605	513220	159822	513860
相关系数	0.9904	0.9900	0.9888	0.9887	0.9857
ETF 代码	513050	513770	513090	513030	513080
相关系数	0.9843	0.9795	0.9434	0.892	0.8686

可以画出成交价格的趋势图如下，取第一支（159607）与最后一支（513050）

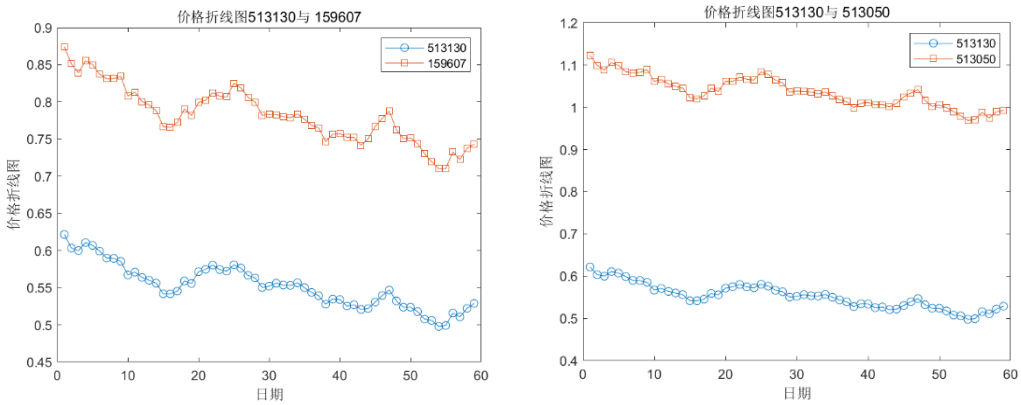

图 5 513130 与其他 ETF 成交价格的趋势图

通过折线图能验证选出的 ETF 与恒生科技有着较大的相关性，其基本走势相似，可以用于配对交易。接着，对选出的 ETF 做对进行平稳性检验，相关 ETF 各频率样本数据下对数化序列平稳性检验结果如下表所示：

表 13 ETF 的对数化序列平稳性检验表

ETF 代码	P_value	ADF	结论
159607	0.8684	0.7262	
159605	0.8754	0.7613	
513220	0.7726	0.3427	
159822	0.9250	1.0796	
513860	0.8764	0.7672	不平稳，继续判断是否同阶单整
513050	0.0734	-1.7663	
513770	0.9452	1.2580	
513090	0.0842	-1.6987	
513030	0.0582	-1.8768	
513080	0.1123	-1.5545	

由检验结果可得，对数化处理时间序列均未通过 ADF 检验，为非平稳序列。而 513130 也为非平稳序。 为进一步筛选，对上述不平稳 ETF 的对数化时间序列进行一阶差分后再次检验其平稳性。ETF 一阶差分对数化序列平稳性检验结果如下表所示：

表 14 ETF 一阶差分对数化序列平稳性检验表

ETF 代码	P_value 一阶	ADF 一阶	结论

159607	0.0010	-3.6769	平稳
159605	0.0010	-3.6532	
513220	0.0010	-4.0242	
159822	0.0010	-3.4016	
513860	0.0010	-3.5736	
513050	0.0010	-3.7535	
513770	0.0010	-3.6895	
513090	0.0010	-5.0471	
513030	0.0010	-4.7975	
513080	0.0010	-4.9656	

根据检验结果，上述 10 只 ETF 的价格时间序列取对数后再进行一阶差分得到的 ADF 检验值均能够拒绝原假设，为平稳序列。因此，满足相关系数要求可以进行后续协整检验。

根据 ADF 检验的结果，本文将以 ETF 的对数价格序列为基础，构建回归方程、预测残差并检验残差项的平稳性，若残差项平稳，则该组股票对通过协整检验；反之未通过协整检验，该组 ETF 不存在长期稳定关系，不参与后续配对。ETF 进行协整检验，得到的结果如下：

表 15 ETF 协整检验表

ETF 代码	P_value 残差	ADF 残差	结论
159607	0.0096	-2.6276	1%
159605	0.0059	-2.8165	1%
513220	0.0094	-2.6377	1%
159822	0.0112	-2.5707	5%
513860	0.0099	-2.6128	1%
513050	0.0063	-2.7935	1%
513770	0.0115	-2.5618	5%
513090	0.0010	-3.7941	1%
513030	0.0093	-2.6424	1%
513080	0.0116	-2.5591	5%

根据 ETF 对协整回归后残差序列检验结果，其中“159822-513130”、

“513770-513130”、“513080-513130”回归后残差序列均在5%置信区间上平稳，虽能够拒绝原假设，残差序列平稳，但其显著性不及其他 ETF 对，因此予以剔除。其余ETF对回归后残差序列均在1%置信区间上平稳，通过协整检验，可以用于后续的配对交易。

最后，综合Pearson系数以及残差平稳性检验，挑选出如下ETF对，OLS方程为：

表16 ETF的OLS回归方程表

ETF对	OLS回归方程
513130-513090	$ln\,p_t^Y = 1.2340\,ln\,p_t^X - 1.8416 + \varepsilon_t$

根据理论基础的分析，能够使用残差序列代替 ETF 对去中心化的价差序列，因此，该组ETF去中心化的价差序列为：

表17 ETF去中心化的价差序列表

ETF对	OLS回归方程
513130-513090	$Mspread_t = ln\,p_t^Y - 1.2340\,ln\,p_t^X + 1.8416$

其中1.2340为该ETF对下，交易阈值触发时，两只ETF之间的交易比例。当建仓阈值触发时，每买入1单位513130，卖出0.8512单位513090。

根据第二问基于布林带残差波动择时策略的交易信号，布林带的移动均线的天数在股市行情软件中默认设置为 20 日，因此本文中也采取 20 日。对于建仓阈值设定为20日移动均线±1倍标准差，止损阈值设定为20日移动均线±2倍标准差，平仓阈值为20日移动均线。基于布林带理论，平仓线、建仓线、止损线随着时间推移是变动的。在样本区间内将平仓线、建仓线、止损线、去中心化价差序列$Mspread_t$绘制在同一张图中，如图所示：

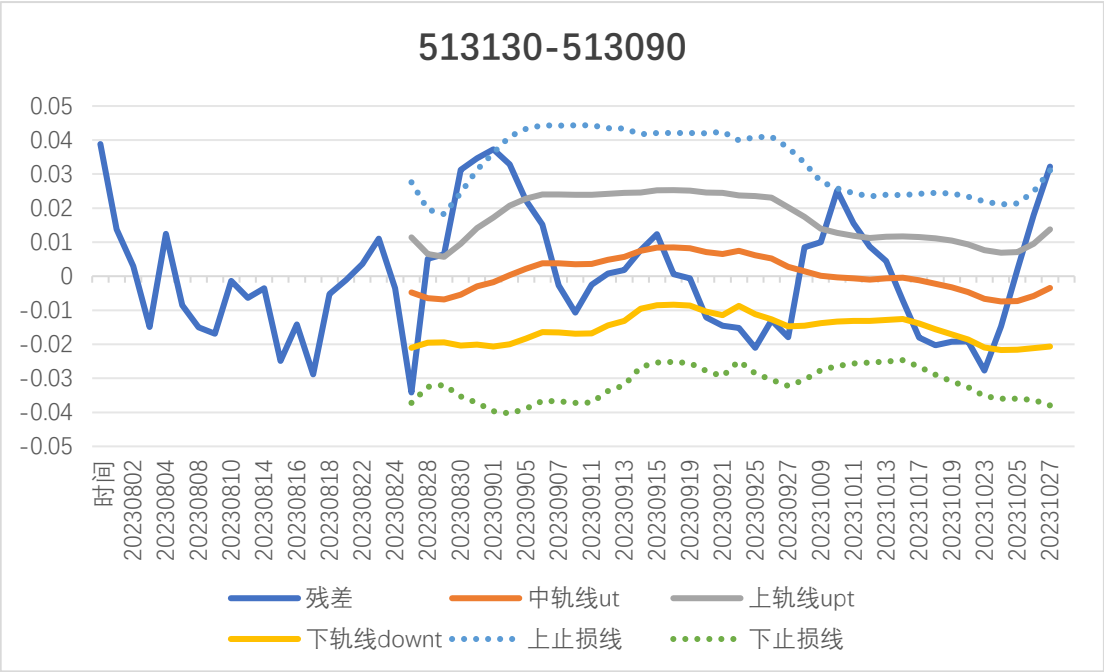

图 6 513130-513090 的去中心化价差序列

其中，布林带通道在图的前半部分空缺的原因是：中轨线需要当日以及前 20 天的数据生成，基于附件中数据，布林带通道在前 20 天，即 8.1—8.28 数据用于计算中轨线，从 8.29 开始到 10.30 每天都有对应的中轨线数值。

基于布林带残差波动择时策略在样本内的区间中有 4 次套利，2 次正向建仓，2 次反向建仓。策略交易采用一次性全额建仓，根据交易信号计算在样本内区间的利润、交易费用、套利盈亏以及总资产，将回测的结果如表所示：

表 18 基于布林带残差波动择时策略的交易信号回测结果表

日期	建仓类型	利润	手续费	盈亏情况	总资产
8.29-8.30	反向建仓	-10518.3	301.0456	亏损	1489181
9.20-9.28	正向建仓	155321.8	286.5073	盈利	1644216
10.10-10.16	反向建仓	18065.6	269.1086	盈利	1662012
10.17-10.25	正向建仓	24915.26	329.9044	盈利	1686598

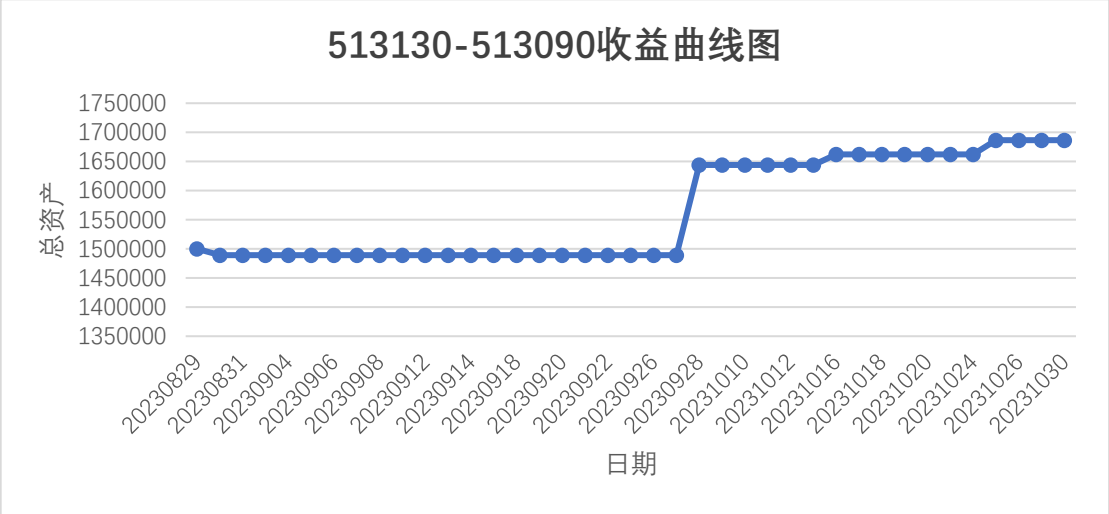

图 7 513130-513090 收益曲线图

将策略收益、风险和绩效综合评价指标如表所示：

表 19 评价指标与数值表

评价指标	数值
套利次数	4
套利胜率	75%
净利润	187784.36
年化收益率	209.0069%
最大回撤	11.7050%

同理可得 IF 与 159822 的去中心价差序列如下图：

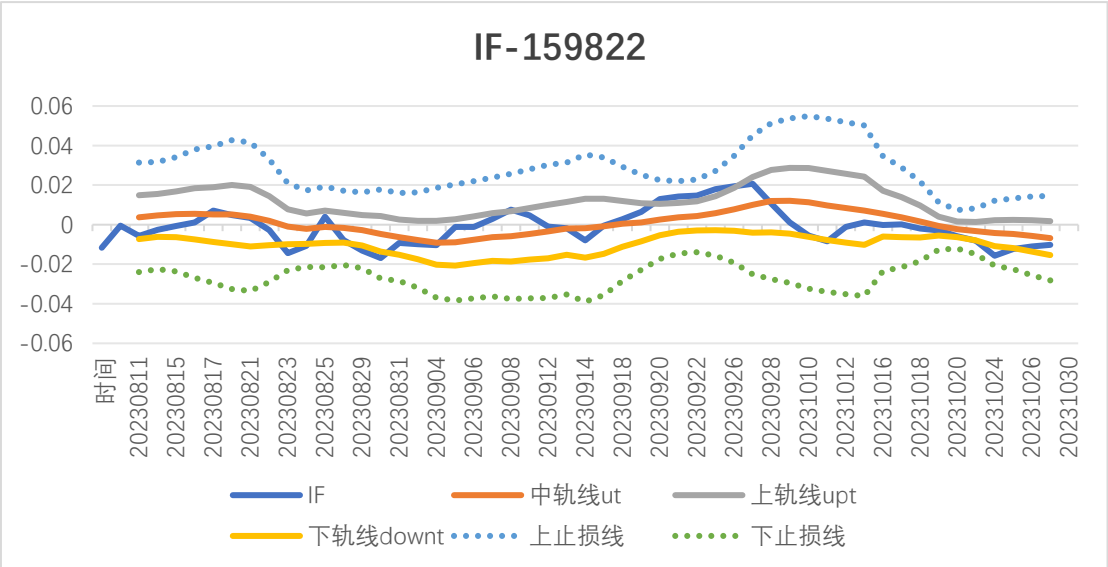

图 8 IF 与 159822 的去中心化价差序列

基于 GARCH 套利策略在样本内的区间中有 4 次套利，2 次正向建仓，2 次反向建仓。根据交易信号计算在样本内区间的利润、交易费用、套利盈亏以及总资产，将回测的结果如表所示：

表 20 基于 GARCH 套利策略的交易信号回测结果表

日期	建仓类型	利润	手续费	盈亏情况	总资产
8.24-8.28	正向建仓	24053.46	101.19	盈利	1523952.267
8.29-9.05	正向建仓	15230.17	153.15	盈利	1539029.282
9.08-9.14	反向建仓	4237.34	308.22	盈利	1542958.397
9.20-9.28	反向建仓	-18625.38	150.73	亏损	1524182.28

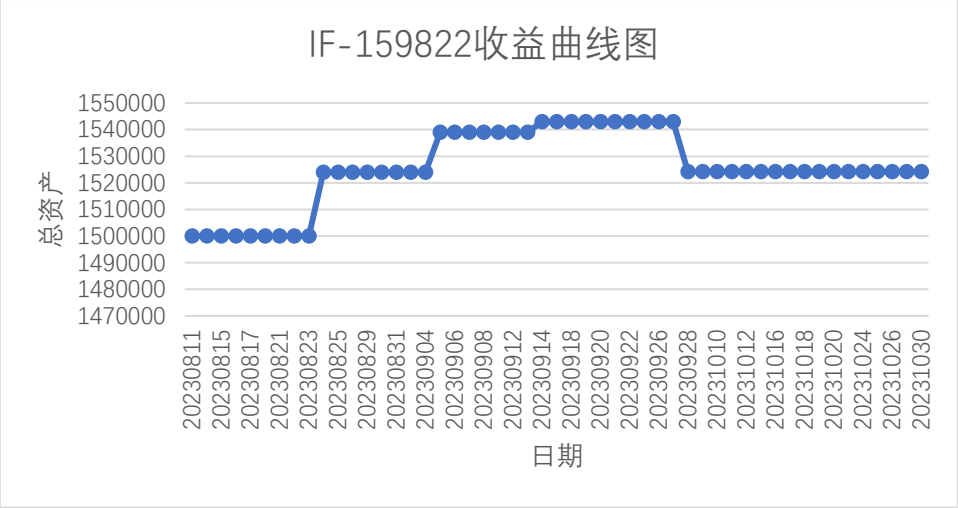

图 9 IF-159822 收益曲线图

将策略收益、风险和绩效综合评价指标如表所示：

表 21 评价指标与数值表 2

评价指标	数值
套利次数	4
套利胜率	75%
净利润	24182.28
年化收益率	106.8908%
最大回撤	8.3752%

九、优缺点分析

9.1 优点

1. 通过流动性风险与波动性风险指标能很好的刻画每只 ETF 在市场中的活跃程度以及规模，修正 LCAPM 模型能帮助投资人迅速选出符合预期的 ETF，进行下一步的套利操作。

2.通过协整相关法选取投资组合，能在可接受的风险内获得较大的收益，通过布林带通道与 GARCH 套利策略，面对不同性质的投资组合都能提供较好的方法套利，且建仓线平仓线随每日数据变化，能够很好的追踪 ETF 的实时情况，以便投资人做出相应调整。

3. 在 ETF 的配对方法上，选择先通过计算相关系数的大小进行初步筛选，再通过协整法确定。

9.2 缺点

1. 在 GARCH 模型中，我们假定正负扰动通过影响波动率对资产收益率产生相同的影响。然而，在实际的资产收益率中，正负扰动对波动率的影响是不同的。较大的负扰动往往比正扰动引起更大的波动。模型没有考虑这种非对称性。

2.未考虑事件套利的可能性，模型只对常规交易套利做出了分析与求解。

参考文献

[1] 周扬博.ETF 基金与股价波动率关系研究.2023.西南财经大学,MA thesis.

[2] 陆政. 流动性风险与 ETF 追踪误差和收益[D].苏州科技大学,2022.DOI:10.27748/d.cnki.gszkj.2022.000523.

[3] 闫舒悦. 基于 MS-GARCH 模型的高频股票配对交易策略研究[D].南京信息工程大学,2023.DOI:10.27248/d.cnki.gnjqc.2023.001487.

[4] 彭寒香. 科创 ETF 套利策略研究[D].广东财经大学,2022.DOI:10.27734/d.cnki.ggdsx.2022.000821.

附录 1：数据处理与计算代码（python）

1. 求成交金额和成交量之比

```
import os
import pandas as pd

# 指定文件夹路径
folder_path = "数据 O1"
# 遍历文件夹下的所有文件
for file_name in os.listdir(folder_path):
    if file_name.endswith(".xlsx"):
        file_path = os.path.join(folder_path, file_name)
        # 读取 Excel 文件
          df = pd.read_excel(file_path)
        df.columns = ["时间", "收盘价", "开盘价", "最高价", "最低价", "成交额（元）", "成交量（手）
"]
        # 计算倒数第二个数据除以倒数第一个数据的结果
          df["结果"] = df.iloc[:, -2] / df.iloc[:, -1]
        # 将结果放在该行的最后一个位置
          df = df[["时间", "收盘价", "开盘价", "最高价", "最低价", "成交额（元）", "成交量（手）", "
结果"]]

        # 保存更改后的 Excel 文件
          df.to_excel(file_path, index=False)
        print(file_name + " 处理完成")
```

2. 平均值方差排除错误数据

```
import os
import pandas as pd

# 遍历文件夹
folder = '数据 O2'
for filename in os.listdir(folder):
    # 只处理.xlsx 文件
     if filename.endswith('.xlsx'):
        # 读取文件
          filepath = os.path.join(folder, filename)
        df = pd.read_excel(filepath)
        df = df.apply(lambda x: x.apply(lambda y: None if y == "none" else y))
        # 计算最后一列的平均值和方差
          mean = df.iloc[:, -1].mean()
```

```
        std = df.iloc[:, -1].std()

        # 找出在平均值-3*方差和平均值+3*方差之间的数据
          lower_bound = mean - 3 * std
        upper_bound = mean + 3 * std
        print('平均值:', mean)
        print('方差:', std)

        # 遍历文件，将不在范围内的数据变为空值
          for i in range(df.shape[0]):
            row = df.iloc[i, :]
            if row.iloc[-1] < lower_bound or row.iloc[-1] > upper_bound:
                df.iloc[i, [-3, -2, -1]] = None

        df.to_excel(filename, index=False)
        print(filename, '处理完成')
```

3. 众数+求和（日为单位）

```
import os
import pandas as pd

# 指定数据文件夹路径
folder_path = "数据 03"

# 遍历数据文件夹中的所有 xlsx 文件
for file_name in os.listdir(folder_path):
    if file_name.endswith(".xlsx"):
        # 读取 xlsx 文件
          file_path = os.path.join(folder_path, file_name)
        data = pd.read_excel(file_path)

        # 定义时间列表
          time_list = [20230801, 20230802, 20230803, 20230804, 20230807,
20230808, 20230809, 20230810,
                     20230811, 20230814, 20230815, 20230816, 20230817,
20230818, 20230821, 20230822,
                     20230823, 20230824, 20230825, 20230828, 20230829,
20230830, 20230831, 20230901,
                     20230904, 20230905, 20230906, 20230907, 20230908,
20230911, 20230912, 20230913,
                     20230914, 20230915, 20230918, 20230919, 20230920,
20230921, 20230922, 20230925,
```

```
                20230926, 20230927, 20230928, 20231009, 20231010,
20231011, 20231012, 20231013,
                20231016, 20231017, 20231018, 20231019, 20231020,
20231023, 20231024, 20231025,
                20231026, 20231027, 20231030]

    # 定义一个空的列表来存储结果
     results = []
    # 对每个时间元素执行相同的操作
     for time in time_list:
        # 筛选出“时间”栏中所有为 time 的数据
            selected_data = data[data["时间"] == time]
        # 分别求出收盘价栏、开盘价栏、最高价栏和最低价栏的众数
            close_price_mode = selected_data["收盘价"].mode()[0]
        open_price_mode = selected_data["开盘价"].mode()[0]
        high_price_mode = selected_data["最高价"].mode()[0]
        low_price_mode = selected_data["最低价"].mode()[0]
        # 求和值
            sum_values1 = selected_data["成交额（元）"].sum()
        sum_values2 = selected_data["成交量（手）"].sum()
        # 保存结果
            result = pd.DataFrame({
             "时间": [time],
             "收盘价众数": [close_price_mode],
             "开盘价众数": [open_price_mode],
             "最高价众数": [high_price_mode],
             "最低价众数": [low_price_mode],
             "成交额（元）求和值": [sum_values1],
             "成交量（手）求和值": [sum_values2]
        })

        # 将结果添加到结果列表中
            results.append(result)
    # 将所有结果合并到一个 DataFrame 中
     final_data = pd.concat(results)
    # 保存结果到新的 xlsx 文件
     output_file_name = file_name.split(".")[0] + "_output.xlsx"
    output_file_path = os.path.join(folder_path, output_file_name)
    final_data.to_excel(output_file_path, index=False)
    print(f"处理完成：{file_name}")
```

4. 众数+求和（小时为单位）

```
import os
import pandas as pd

# 指定数据文件夹路径
folder_path = "数据 04"

# 遍历数据文件夹中的所有 xlsx 文件
for file_name in os.listdir(folder_path):
    if file_name.endswith(".xlsx"):
        # 读取 xlsx 文件
          file_path = os.path.join(folder_path, file_name)
        data = pd.read_excel(file_path)

        # 定义时间列表
          time_list = ['20230801 09','20230801 10','20230801 11','20230801
13','20230801 14','20230802 09','20230802 10','20230802 11','20230802
13','20230802 14','20230803 09','20230803 10','20230803 11','20230803
13','20230803 14','20230804 09','20230804 10','20230804 11','20230804
13','20230804 14','20230807 09','20230807 10','20230807 11','20230807
13','20230807 14','20230808 09','20230808 10','20230808 11','20230808
13','20230808 14','20230809 09','20230809 10','20230809 11','20230809
13','20230809 14','20230810 09','20230810 10','20230810 11','20230810
13','20230810 14','20230811 09','20230811 10','20230811 11','20230811
13','20230811 14','20230814 09','20230814 10','20230814 11','20230814
13','20230814 14','20230815 09','20230815 10','20230815 11','20230815
13','20230815 14','20230816 09','20230816 10','20230816 11','20230816
13','20230816 14','20230817 09','20230817 10','20230817 11','20230817
13','20230817 14','20230818 09','20230818 10','20230818 11','20230818
13','20230818 14','20230821 09','20230821 10','20230821 11','20230821
13','20230821 14','20230822 09','20230822 10','20230822 11','20230822
13','20230822 14','20230823 09','20230823 10','20230823 11','20230823
13','20230823 14','20230824 09','20230824 10','20230824 11','20230824
13','20230824 14','20230825 09','20230825 10','20230825 11','20230825
13','20230825 14','20230828 09','20230828 10','20230828 11','20230828
13','20230828 14','20230829 09','20230829 10','20230829 11','20230829
13','20230829 14','20230830 09','20230830 10','20230830 11','20230830
13','20230830 14','20230831 09','20230831 10','20230831 11','20230831
13','20230831 14','20230901 09','20230901 10','20230901 11','20230901
13','20230901 14','20230904 09','20230904 10','20230904 11','20230904
13','20230904 14','20230905 09','20230905 10','20230905 11','20230905
13','20230905 14','20230906 09','20230906 10','20230906 11','20230906
13','20230906 14','20230907 09','20230907 10','20230907 11','20230907
```

```
13','20230907 14','20230908 09','20230908 10','20230908 11','20230908
13','20230908 14','20230911 09','20230911 10','20230911 11','20230911
13','20230911 14','20230912 09','20230912 10','20230912 11','20230912
13','20230912 14','20230913 09','20230913 10','20230913 11','20230913
13','20230913 14','20230914 09','20230914 10','20230914 11','20230914
13','20230914 14','20230915 09','20230915 10','20230915 11','20230915
13','20230915 14','20230918 09','20230918 10','20230918 11','20230918
13','20230918 14','20230919 09','20230919 10','20230919 11','20230919
13','20230919 14','20230919 15','20230920 09','20230920 10','20230920
11','20230920 13','20230920 14','20230921 09','20230921 10','20230921
11','20230921 13','20230921 14','20230922 09','20230922 10','20230922
11','20230922 13','20230922 14','20230925 09','20230925 10','20230925
11','20230925 13','20230925 14','20230926 09','20230926 10','20230926
11','20230926 13','20230926 14','20230927 09','20230927 10','20230927
11','20230927 13','20230927 14','20230928 09','20230928 10','20230928
11','20230928 13','20230928 14','20231009 09','20231009 10','20231009
11','20231009 13','20231009 14','20231010 09','20231010 10','20231010
11','20231010 13','20231010 14','20231011 09','20231011 10','20231011
11','20231011 13','20231011 14','20231012 09','20231012 10','20231012
11','20231012 13','20231012 14','20231013 09','20231013 10','20231013
11','20231013 13','20231013 14','20231016 09','20231016 10','20231016
11','20231016 13','20231016 14','20231017 09','20231017 10','20231017
11','20231017 13','20231017 14','20231018 09','20231018 10','20231018
11','20231018 13','20231018 14','20231019 09','20231019 10','20231019
11','20231019 13','20231019 14','20231020 09','20231020 10','20231020
11','20231020 13','20231020 14','20231023 09','20231023 10','20231023
11','20231023 13','20231023 14','20231024 09','20231024 10','20231024
11','20231024 13','20231024 14','20231025 09','20231025 10','20231025
11','20231025 13','20231025 14','20231026 09','20231026 10','20231026
11','20231026 13','20231026 14','20231027 09','20231027 10','20231027
11','20231027 13','20231027 14','20231030 09','20231030 10','20231030
11','20231030 13','20231030 14']
    # 定义一个空的列表来存储结果
     results = []

    # 对每个时间元素执行相同的操作
     for time in time_list:
       # 筛选出“时间”栏中所有为 time 的数据
          selected_data = data[data["时间"] == time]
       # 分别求出收盘价栏、开盘价栏、最高价栏和最低价栏的众数
          close_price_mode = selected_data["收盘价"].mode()[0]
       open_price_mode = selected_data["开盘价"].mode()[0]
```

```
            high_price_mode = selected_data["最高价"].mode()[0]
            low_price_mode = selected_data["最低价"].mode()[0]
            # 求和值
               sum_values1 = selected_data["成交额（元）"].sum()
            sum_values2 = selected_data["成交量（手）"].sum()
            # 保存结果
               result = pd.DataFrame({
                "时间": [time],
                "收盘价众数": [close_price_mode],
                "开盘价众数": [open_price_mode],
                "最高价众数": [high_price_mode],
                "最低价众数": [low_price_mode],
                "成交额（元）求和值": [sum_values1],
                "成交量（手）求和值": [sum_values2]
            })

            # 将结果添加到结果列表中
               results.append(result)
        # 将所有结果合并到一个 DataFrame 中
          final_data = pd.concat(results)
        # 保存结果到新的 xlsx 文件
          output_file_name = file_name.split(".")[0] + "_output.xlsx"
        output_file_path = os.path.join(folder_path, output_file_name)
        final_data.to_excel(output_file_path, index=False)
        print(f"处理完成：{file_name}")
```

5. cti 小时为单位初步计算

```
import os
import pandas as pd

# 指定数据文件夹路径
folder_path = "数据 05"
# 遍历数据文件夹中的所有 xlsx 文件
for file_name in os.listdir(folder_path):
    if file_name.endswith(".xlsx"):
        # 读取 xlsx 文件
          file_path = os.path.join(folder_path, file_name)
        data = pd.read_excel(file_path)
        data.replace(0, None, inplace=True)

        # 定义时间列表
          time_list = ['20230801 09','20230801 10','20230801 11','20230801
13','20230801 14','20230802 09','20230802 10','20230802 11','20230802
```

13','20230802 14','20230803 09','20230803 10','20230803 11','20230803
13','20230803 14','20230804 09','20230804 10','20230804 11','20230804
13','20230804 14','20230807 09','20230807 10','20230807 11','20230807
13','20230807 14','20230808 09','20230808 10','20230808 11','20230808
13','20230808 14','20230809 09','20230809 10','20230809 11','20230809
13','20230809 14','20230810 09','20230810 10','20230810 11','20230810
13','20230810 14','20230811 09','20230811 10','20230811 11','20230811
13','20230811 14','20230814 09','20230814 10','20230814 11','20230814
13','20230814 14','20230815 09','20230815 10','20230815 11','20230815
13','20230815 14','20230816 09','20230816 10','20230816 11','20230816
13','20230816 14','20230817 09','20230817 10','20230817 11','20230817
13','20230817 14','20230818 09','20230818 10','20230818 11','20230818
13','20230818 14','20230821 09','20230821 10','20230821 11','20230821
13','20230821 14','20230822 09','20230822 10','20230822 11','20230822
13','20230822 14','20230823 09','20230823 10','20230823 11','20230823
13','20230823 14','20230824 09','20230824 10','20230824 11','20230824
13','20230824 14','20230825 09','20230825 10','20230825 11','20230825
13','20230825 14','20230828 09','20230828 10','20230828 11','20230828
13','20230828 14','20230829 09','20230829 10','20230829 11','20230829
13','20230829 14','20230830 09','20230830 10','20230830 11','20230830
13','20230830 14','20230831 09','20230831 10','20230831 11','20230831
13','20230831 14','20230901 09','20230901 10','20230901 11','20230901
13','20230901 14','20230904 09','20230904 10','20230904 11','20230904
13','20230904 14','20230905 09','20230905 10','20230905 11','20230905
13','20230905 14','20230906 09','20230906 10','20230906 11','20230906
13','20230906 14','20230907 09','20230907 10','20230907 11','20230907
13','20230907 14','20230908 09','20230908 10','20230908 11','20230908
13','20230908 14','20230911 09','20230911 10','20230911 11','20230911
13','20230911 14','20230912 09','20230912 10','20230912 11','20230912
13','20230912 14','20230913 09','20230913 10','20230913 11','20230913
13','20230913 14','20230914 09','20230914 10','20230914 11','20230914
13','20230914 14','20230915 09','20230915 10','20230915 11','20230915
13','20230915 14','20230918 09','20230918 10','20230918 11','20230918
13','20230918 14','20230919 09','20230919 10','20230919 11','20230919
13','20230919 14','20230919 15','20230920 09','20230920 10','20230920
11','20230920 13','20230920 14','20230921 09','20230921 10','20230921
11','20230921 13','20230921 14','20230922 09','20230922 10','20230922
11','20230922 13','20230922 14','20230925 09','20230925 10','20230925
11','20230925 13','20230925 14','20230926 09','20230926 10','20230926
11','20230926 13','20230926 14','20230927 09','20230927 10','20230927
11','20230927 13','20230927 14','20230928 09','20230928 10','20230928
11','20230928 13','20230928 14','20231009 09','20231009 10','20231009

```
11','20231009 13','20231009 14','20231010 09','20231010 10','20231010
11','20231010 13','20231010 14','20231011 09','20231011 10','20231011
11','20231011 13','20231011 14','20231012 09','20231012 10','20231012
11','20231012 13','20231012 14','20231013 09','20231013 10','20231013
11','20231013 13','20231013 14','20231016 09','20231016 10','20231016
11','20231016 13','20231016 14','20231017 09','20231017 10','20231017
11','20231017 13','20231017 14','20231018 09','20231018 10','20231018
11','20231018 13','20231018 14','20231019 09','20231019 10','20231019
11','20231019 13','20231019 14','20231020 09','20231020 10','20231020
11','20231020 13','20231020 14','20231023 09','20231023 10','20231023
11','20231023 13','20231023 14','20231024 09','20231024 10','20231024
11','20231024 13','20231024 14','20231025 09','20231025 10','20231025
11','20231025 13','20231025 14','20231026 09','20231026 10','20231026
11','20231026 13','20231026 14','20231027 09','20231027 10','20231027
11','20231027 13','20231027 14','20231030 09','20231030 10','20231030
11','20231030 13','20231030 14']
    # 定义一个空的列表来存储结果
     results = []
    # 对每个时间元素执行相同的操作
     for time in time_list:
        # 筛选出“时间”栏中所有为 time 的数据
          selected_data = data[data["时间"] == time]
        # 分别成交额最大最小和综合
          max_values = selected_data["成交额（元）"].max()
        min_values = selected_data["成交额（元）"].min()
        sum_values = selected_data["成交额（元）"].sum()
        cti_values = (max_values - min_values) / sum_values
        # 保存结果
          result = pd.DataFrame({
           "时间": [time],
           "成交额最大值": [max_values],
           "成交额最小值": [min_values],
           "成交额总和": [sum_values],
           "cti": [cti_values]
        })

        # 将结果添加到结果列表中
          results.append(result)
    # 将所有结果合并到一个 DataFrame 中
     final_data = pd.concat(results)

    # 保存结果到新的 xlsx 文件
```

```
        output_file_name = file_name.split(".")[0] + "_output.xlsx"
        output_file_path = os.path.join(folder_path, output_file_name)
        final_data.to_excel(output_file_path, index=False)
        print(f"处理完成：{file_name}")
```

6. cti 计算

```
import os
import pandas as pd

# 指定数据文件夹路径
folder_path = "数据 06"

# 遍历数据文件夹中的所有 xlsx 文件
for file_name in os.listdir(folder_path):
    if file_name.endswith(".xlsx"):
        # 读取 xlsx 文件
        file_path = os.path.join(folder_path, file_name)
        data = pd.read_excel(file_path)

        # 定义时间列表
        time_list = [20230801, 20230802, 20230803, 20230804, 20230807,
20230808, 20230809, 20230810,
                     20230811, 20230814, 20230815, 20230816, 20230817,
20230818, 20230821, 20230822,
                     20230823, 20230824, 20230825, 20230828, 20230829,
20230830, 20230831, 20230901,
                     20230904, 20230905, 20230906, 20230907, 20230908,
20230911, 20230912, 20230913,
                     20230914, 20230915, 20230918, 20230919, 20230920,
20230921, 20230922, 20230925,
                     20230926, 20230927, 20230928, 20231009, 20231010,
20231011, 20231012, 20231013,
                     20231016, 20231017, 20231018, 20231019, 20231020,
20231023, 20231024, 20231025,
                     20231026, 20231027, 20231030]

        # 定义一个空的列表来存储结果
        results = []

        # 对每个时间元素执行相同的操作
        for time in time_list:
            # 筛选出“时间”栏中所有为 time 的数据
            selected_data = data[data["时间"] == time]
```

```
        # 分别求出收盘价栏、开盘价栏、最高价栏和最低价栏的众数
            cti_day = selected_data["cti"].mean()

        # 保存结果
            result = pd.DataFrame({
             "时间": [time],
             "cti 平均值": [cti_day]
        })

        # 将结果添加到结果列表中
            results.append(result)

    # 将所有结果合并到一个 DataFrame 中
      final_data = pd.concat(results)

    # 保存结果到新的 xlsx 文件
      output_file_name = file_name.split(".")[0] + "_output_output.xlsx"
    output_file_path = os.path.join(folder_path, output_file_name)
    final_data.to_excel(output_file_path, index=False)

    print(f"处理完成：{file_name}")
```

7. 07.cti 汇总

```
import os
import pandas as pd

# 创建一个新的 DataFrame 对象，用于存储合并后的数据
merged_data = pd.DataFrame()

example = pd.read_excel('159001.SZ.xlsx')
merged_data['时间'] = example['时间']

# 指定数据文件夹路径
folder_path = "0751"

# 遍历数据文件夹中的所有 xlsx 文件
for file_name in os.listdir(folder_path):
    if file_name.endswith(".xlsx"):
        # 读取 xlsx 文件
          file_path = os.path.join(folder_path, file_name)
        data = pd.read_excel(file_path)
        merged_data[file_path] = data['cti 平均值']
```

```
        print(file_path, "读取成功")

# 将新创建的 DataFrame 对象保存到一个新的 Excel 文件中
merged_data.to_excel('merged_file.xlsx')
```

8. ctm 计算

```
import pandas as pd

# 读取 Excel 文件
data = pd.read_excel('cti_day_tran.xlsx')

# 对每一列求平均值
column_means = data.mean(axis=0)

# 将结果存储在一个新的 DataFrame 中
result = pd.DataFrame({'列名': column_means.index, '平均值': column_means.values})

# 将结果写入一个新的 Excel 文件，命名为 cmi.xlsx
result.to_excel('cmi.xlsx', index=False)
```

9. fit 折溢率汇总

```
import os
import pandas as pd

# 指定数据文件夹路径
folder_path = "数据 09"

# 遍历数据文件夹中的所有 xlsx 文件
for file_name in os.listdir(folder_path):
    if file_name.endswith(".xlsx"):
        # 读取 xlsx 文件
          file_path = os.path.join(folder_path, file_name)
        data = pd.read_excel(file_path)

        # 定义时间列表
          time_list = [20230801, 20230802, 20230803, 20230804, 20230807,
20230808, 20230809, 20230810,
                     20230811, 20230814, 20230815, 20230816, 20230817,
20230818, 20230821, 20230822,
                     20230823, 20230824, 20230825, 20230828, 20230829,
20230830, 20230831, 20230901,
                     20230904, 20230905, 20230906, 20230907, 20230908,
```

```
20230911, 20230912, 20230913,
                20230914, 20230915, 20230918, 20230919, 20230920,
20230921, 20230922, 20230925,
                20230926, 20230927, 20230928, 20231009, 20231010,
20231011, 20231012, 20231013,
                20231016, 20231017, 20231018, 20231019, 20231020,
20231023, 20231024, 20231025,
                20231026, 20231027, 20231030]

        # 定义一个空的列表来存储结果
         results = []

        # 对每个时间元素执行相同的操作
         for time in time_list:
            # 筛选出“时间”栏中所有为 time 的数据
              selected_data = data[data["时间"] == time]

            # 分别求出收盘价栏、开盘价栏、最高价栏和最低价栏的众数
              close_price_mode = selected_data["收盘价"].mode()[0]
            open_price_mode = selected_data["开盘价"].mode()[0]
            high_price_mode = selected_data["最高价"].mode()[0]
            low_price_mode = selected_data["最低价"].mode()[0]

            # 求和值
              sum_values1 = selected_data["成交额（元）"].sum()
            sum_values2 = selected_data["成交量（手）"].sum()

            p_arg = selected_data["结果"].mean()

            # 保存结果
              result = pd.DataFrame({
               "时间": [time],
               "收盘价众数": [close_price_mode],
               "开盘价众数": [open_price_mode],
               "最高价众数": [high_price_mode],
               "最低价众数": [low_price_mode],
               "成交额（元）求和值": [sum_values1],
               "成交量（手）求和值": [sum_values2],
               "单价": [p_arg]
            })

            # 将结果添加到结果列表中
```

```
            results.append(result)

        # 将所有结果合并到一个 DataFrame 中
        final_data = pd.concat(results)

        # 保存结果到新的 xlsx 文件
        output_file_name = file_name.split(".")[0] + "_output.xlsx"
        output_file_path = os.path.join(folder_path, output_file_name)
        final_data.to_excel(output_file_path, index=False)
        print(f"处理完成：{file_name}")
```

10. fit 折溢率计算

```
import os
import pandas as pd

# 指定文件夹路径
folder_path = "数据 0901"

# 遍历文件夹下的所有文件
for file_name in os.listdir(folder_path):
    if file_name.endswith(".xlsx"):
        file_path = os.path.join(folder_path, file_name)

        # 读取 Excel 文件
        df = pd.read_excel(file_path)
        df.columns = ["时间", "收盘价", "开盘价", "最高价", "最低价", "成交额（元）", "成交量（手）","单价"]
        # 计算倒数第二个数据除以倒数第一个数据的结果
        df["折溢率"] = abs(((df.iloc[:, -1] / 100) - df.iloc[:, 2]) / df.iloc[:, 2])

        # 将结果放在该行的最后一个位置
        df = df[["时间", "收盘价", "开盘价", "最高价", "最低价", "成交额（元）", "成交量（手）", "单价","折溢率"]]

        # 保存更改后的 Excel 文件
        df.to_excel(file_path, index=False)
        print(file_name + " 处理完成")
```

11. rtm 计算

```
import os
import pandas as pd

# 创建一个新的 DataFrame 对象，用于存储合并后的数据
```

```
merged_data = pd.DataFrame()
example = pd.read_excel('159001_output.xlsx')
merged_data['时间'] = example['时间']

# 指定数据文件夹路径
folder_path = "数据 10"
# 遍历数据文件夹中的所有 xlsx 文件
for file_name in os.listdir(folder_path):
    if file_name.endswith(".xlsx"):
        # 读取 xlsx 文件
          file_path = os.path.join(folder_path, file_name)
        data = pd.read_excel(file_path)
        merged_data[file_path] = data['折溢率']
        print(file_path, "读取成功")
# 将新创建的 DataFrame 对象保存到一个新的 Excel 文件中
merged_data.to_excel('merged_file.xlsx')
```

12. rtm 汇总

```
import pandas as pd
# 读取 Excel 文件
data = pd.read_excel('vri_day_tran.xlsx')
# 对每一列求平均值
column_means = data.mean(axis=0)
# 将结果存储在一个新的 DataFrame 中
result = pd.DataFrame({'列名': column_means.index, '平均值': column_means.values})
# 将结果写入一个新的 Excel 文件，命名为 cmi.xlsx
result.to_excel('vtm.xlsx', index=False)
```

13. 获取日期/时间列表

```
import pandas as pd
# 读取 Excel 文件
df = pd.read_excel('159001.SZ - 副本.xlsx')

# 获取“时间”列的字符串列表，并按照它们出现的先后顺序排序
sorted_time_list = df['时间'].unique()
# 输出结果
print(sorted_time_list)
```

14. 去除空白值

```
import pandas as pd
# 读取 Excel 文件
df = pd.read_excel('cti_day.xlsx', sheet_name='Sheet1')
```

```
# 将 0 值替换为 None
df = df.applymap(lambda x: None if x == 0 else x)
# 保存修改后的数据到 Excel 文件
df.to_excel('cti_day.xlsx', sheet_name='Sheet1', index=False)
```

15. 文件中跨境 etf 提取

```
import os

# 切换到包含文件的文件夹
os.chdir('数据 11')

# 定义需要保留的文件名列表
file_need = ['159501', '159502', '159509', '159513', '159519', '159605', '159607',
'159612', '159615', '159632', '159655', '159659', '159660', '159687', '159688',
'159696', '159699', '159740', '159741', '159742', '159747', '159750', '159822',
'159823', '159850', '159866', '159892', '510900', '513000', '513010', '513020',
'513030', '513040', '513050', '513060', '513070', '513080', '513090', '513100',
'513110', '513120', '513130', '513140', '513150', '513160', '513180', '513190',
'513193', '513200', '513220', '513230', '513260', '513280', '513290', '513300',
'513310', '513320', '513330', '513360', '513380', '513390', '513500', '513520',
'513530', '513550', '513560', '513580', '513590', '513600', '513650', '513660',
'513690', '513700', '513770', '513800', '513810', '513853', '513854', '513860',
'513873', '513880', '513890', '513900', '513950', '513960', '513970', '513980',
'513990']

# 获取当前工作目录下的所有文件名
files = os.listdir()

# 遍历文件名列表
for file in files:
    # 如果文件名以 file_need 中的任何元素开头，则不执行任何操作
    if any(file.startswith(need_file) for need_file in file_need):
        continue
    # 如果文件名不以 file_need 中的任何元素开头，则删除该文件
    else:
        os.remove(file)
```

16. 表格中跨境 etf 处理

```
import pandas as pd

file_need = ['159501', '159502', '159509', '159513', '159519', '159605', '159607',
'159612', '159615', '159632', '159655', '159659', '159660', '159687', '159688',
```

```
'159696', '159699', '159740', '159741', '159742', '159747', '159750', '159822',
'159823', '159850', '159866', '159892', '510900', '513000', '513010', '513020',
'513030', '513040', '513050', '513060', '513070', '513080', '513090', '513100',
'513110', '513120', '513130', '513140', '513150', '513160', '513180', '513190',
'513193', '513200', '513220', '513230', '513260', '513280', '513290', '513300',
'513310', '513320', '513330', '513360', '513380', '513390', '513500', '513520',
'513530', '513550', '513560', '513580', '513590', '513600', '513650', '513660',
'513690', '513700', '513770', '513800', '513810', '513853', '513854', '513860',
'513873', '513880', '513890', '513900', '513950', '513960', '513970', '513980',
'513990']

# 读取.xlsx 文件
df = pd.read_excel('cmi_ctm.xlsx')

df["列名"] = df["列名"].astype(str)

# 删除第一列名称开头不在列表 file_need 的行
df = df.drop(df[df.iloc[:, 0].str.startswith(tuple(file_need)).eq(False)].index)

# 将删除后的数据保存到新的.xlsx 文件中
df.to_excel('cmi_ctm.xlsx', index=False)
```

17. 股指期货提取

```
import os
import pandas as pd

# 指定文件夹路径
folder_path = "数据"

# 遍历文件夹下的所有文件
for file_name in os.listdir(folder_path):
    if file_name.endswith(".csv"):
        file_path = os.path.join(folder_path, file_name)

        # 读取 CSV 文件
        df = pd.read_csv(file_path, encoding="gbk")

        # 选择"合约代码"列以"IC"、"IF"、"IH"、"IM"开头的行
        df = df[df["合约代码"].str.startswith(("IC", "IF", "IH", "IM"))]

        # 保存更改后的 csv 文件
```

```
        df.to_csv(file_path, index=False)
        print(file_name + " 处理完成")
```

18. 股指期货平均&求和

```
import os
import pandas as pd

# 指定数据文件夹路径
folder_path = "数据 02"

# 遍历数据文件夹中的所有 xlsx 文件
for file_name in os.listdir(folder_path):
    if file_name.endswith(".csv"):
        # 读取 xlsx 文件
        file_path = os.path.join(folder_path, file_name)
        df = pd.read_csv(file_path,encoding="utf-8")

        # 定义一个空的列表来存储结果
        results = []

        ic_data = df[df['合约代码'].str.startswith('IC')]

        # 分别求平均
        jinkaipan = ic_data["今开盘"].mean()
        high = ic_data["最高价"].mean()
        low = ic_data["最低价"].mean()

        chengjiaolaing = ic_data["成交量"].sum()
        chengjiaojine = ic_data["成交金额"].sum()
        chicangliang = ic_data["持仓量"].sum()
        chicangbianhhua = ic_data["持仓变化"].sum()

        jinshoupan = ic_data["今收盘"].mean()
        jinjiesuan = ic_data["今结算"].mean()
        qianjiesuan = ic_data["前结算"].mean()

        zhangdie1 = ic_data["涨跌 1"].mean()
        zhangdie2 = ic_data["涨跌 2"].mean()

        # 保存结果
        result = pd.DataFrame({
            "型号": ["IC"],
```

```
    "今开盘平均": [jinkaipan],
    "最高价平均": [high],
    "最低价平均": [low],
    "成交量求和": [chengjiaolaing],
    "成交金额求和": [chengjiaojine],
    "持仓量求和": [chicangliang],
    "持仓变化求和": [chicangbianhhua],
    "今收盘平均": [jinshoupan],
    "今结算平均": [jinjiesuan],
    "前结算平均": [qianjiesuan],
    "涨跌 1 平均": [zhangdie1],
    "涨跌 2 平均": [zhangdie2]
})

# 将结果添加到结果列表中
  results.append(result)

ic_data = df[df['合约代码'].str.startswith('IF')]

# 分别求平均
  jinkaipan = ic_data["今开盘"].mean()
high = ic_data["最高价"].mean()
low = ic_data["最低价"].mean()

chengjiaolaing = ic_data["成交量"].sum()
chengjiaojine = ic_data["成交金额"].sum()
chicangliang = ic_data["持仓量"].sum()
chicangbianhhua = ic_data["持仓变化"].sum()

jinshoupan = ic_data["今收盘"].mean()
jinjiesuan = ic_data["今结算"].mean()
qianjiesuan = ic_data["前结算"].mean()

zhangdie1 = ic_data["涨跌 1"].mean()
zhangdie2 = ic_data["涨跌 2"].mean()

# 保存结果
  result = pd.DataFrame({
    "型号": ["IF"],
    "今开盘平均": [jinkaipan],
    "最高价平均": [high],
    "最低价平均": [low],
```

```
    "成交量求和": [chengjiaolaing],
    "成交金额求和": [chengjiaojine],
    "持仓量求和": [chicangliang],
    "持仓变化求和": [chicangbianhhua],
    "今收盘平均": [jinshoupan],
    "今结算平均": [jinjiesuan],
    "前结算平均": [qianjiesuan],
    "涨跌 1 平均": [zhangdie1],
    "涨跌 2 平均": [zhangdie2]
})

# 将结果添加到结果列表中
  results.append(result)

ic_data = df[df['合约代码'].str.startswith('IH')]

# 分别求平均
  jinkaipan = ic_data["今开盘"].mean()
high = ic_data["最高价"].mean()
low = ic_data["最低价"].mean()

chengjiaolaing = ic_data["成交量"].sum()
chengjiaojine = ic_data["成交金额"].sum()
chicangliang = ic_data["持仓量"].sum()
chicangbianhhua = ic_data["持仓变化"].sum()

jinshoupan = ic_data["今收盘"].mean()
jinjiesuan = ic_data["今结算"].mean()
qianjiesuan = ic_data["前结算"].mean()

zhangdie1 = ic_data["涨跌 1"].mean()
zhangdie2 = ic_data["涨跌 2"].mean()

# 保存结果
  result = pd.DataFrame({
    "型号": ["IH"],
    "今开盘平均": [jinkaipan],
    "最高价平均": [high],
    "最低价平均": [low],
    "成交量求和": [chengjiaolaing],
    "成交金额求和": [chengjiaojine],
    "持仓量求和": [chicangliang],
```

```
    "持仓变化求和": [chicangbianhhua],
    "今收盘平均": [jinshoupan],
    "今结算平均": [jinjiesuan],
    "前结算平均": [qianjiesuan],
    "涨跌 1 平均": [zhangdie1],
    "涨跌 2 平均": [zhangdie2]
})

# 将结果添加到结果列表中
  results.append(result)

ic_data = df[df['合约代码'].str.startswith('IM')]

# 分别求平均
  jinkaipan = ic_data["今开盘"].mean()
high = ic_data["最高价"].mean()
low = ic_data["最低价"].mean()

chengjiaolaing = ic_data["成交量"].sum()
chengjiaojine = ic_data["成交金额"].sum()
chicangliang = ic_data["持仓量"].sum()
chicangbianhhua = ic_data["持仓变化"].sum()

jinshoupan = ic_data["今收盘"].mean()
jinjiesuan = ic_data["今结算"].mean()
qianjiesuan = ic_data["前结算"].mean()

zhangdie1 = ic_data["涨跌 1"].mean()
zhangdie2 = ic_data["涨跌 2"].mean()

# 保存结果
  result = pd.DataFrame({
    "型号": ["IM"],
    "今开盘平均": [jinkaipan],
    "最高价平均": [high],
    "最低价平均": [low],
    "成交量求和": [chengjiaolaing],
    "成交金额求和": [chengjiaojine],
    "持仓量求和": [chicangliang],
    "持仓变化求和": [chicangbianhhua],
    "今收盘平均": [jinshoupan],
    "今结算平均": [jinjiesuan],
```

```
        "前结算平均": [qianjiesuan],
        "涨跌 1 平均": [zhangdie1],
        "涨跌 2 平均": [zhangdie2]
    })

    # 将结果添加到结果列表中
    results.append(result)


    # 将所有结果合并到一个 DataFrame 中
    final_data = pd.concat(results)

    # 保存结果到新的 xlsx 文件
    output_file_name = file_name.split(".")[0] + "_output.csv"
    output_file_path = os.path.join(folder_path, output_file_name)
    final_data.to_csv(output_file_path, index=False)

    print(f"处理完成：{file_name}")
```

19. 股指期货按股读取

```
import os
import pandas as pd

# 指定要搜索的文件夹路径
folder_path = '数据 04'

# 创建一个新的空 DataFrame，用于存储提取的数据
df = pd.DataFrame(columns=['型号','时间','今开盘平均','最高价平均','最低价平均','成交量求和','成交金额求和','持仓量求和','持仓变化求和','今收盘平均','今结算平均','前结算平均','涨跌 1 平均','涨跌 2 平均'])

# 遍历文件夹中的所有文件
for root, dirs, files in os.walk(folder_path):
    for file in files:
        # 检查文件是否为 CSV 文件
        if file.endswith('.csv'):
            # 获取文件所在的路径
            file_path = os.path.join(root, file)

            # 读取 CSV 文件
            data = pd.read_csv(file_path)
```

```
    # 提取“合约代码”为“IC”的行
    ic_data = data[data['型号'] == 'IM']

    # 将提取的数据添加到新的 DataFrame 中
    df = df.append({'型号': ic_data['型号'].mode()[0], '时间': file, '今开盘平均': ic_data['今开盘平均'].mode()[0],
                    '最高价平均': ic_data['最高价平均'].mode()[0], '最低价平均': ic_data['最低价平均'].mode()[0],'成交量求和': ic_data['成交量求和'].mode()[0],
                    '成交金额求和': ic_data['成交金额求和'].mode()[0], '持仓量求和': ic_data['持仓量求和'].mode()[0],'持仓变化求和': ic_data['持仓变化求和'].mode()[0],
                    '今收盘平均': ic_data['今收盘平均'].mode()[0], '今结算平均': ic_data['今结算平均'].mode()[0],'前结算平均': ic_data['前结算平均'].mode()[0],
                    '涨跌 1 平均': ic_data['涨跌 1 平均'].mode()[0],'涨跌 2 平均': ic_data['涨跌 2 平均'].mode()[0]}, ignore_index=True)

# 将提取的数据输出到 CSV 文件中
df.to_csv('IM.csv', index=False)
```

附录 2：数据处理与计算代码（matlab）

1.计算股指期货与跨境 ETF 的 Pearson 相关系数

```
clc
clear
%初始化存储 Pearson 相关系数的矩阵
correlationMatrix = zeros(83);
%循环遍历所有 Excel 表格
for i = 1:4
    for j = 1:82
        filename0=["IC.xlsx", 'IF.xlsx', 'IH.xlsx', 'IM.xlsx'];
        filename=["159501_output.xlsx", '159509_output.xlsx',
'159513_output.xlsx', '159519_output.xlsx', '159605_output.xlsx',
'159607_output.xlsx', '159612_output.xlsx', '159615_output.xlsx',
'159632_output.xlsx', '159655_output.xlsx', '159659_output.xlsx',
'159660_output.xlsx', '159687_output.xlsx', '159688_output.xlsx',
'159696_output.xlsx', '159699_output.xlsx', '159740_output.xlsx',
'159741_output.xlsx', '159742_output.xlsx', '159747_output.xlsx',
'159750_output.xlsx', '159822_output.xlsx', '159823_output.xlsx',
'159850_output.xlsx', '159866_output.xlsx', '159892_output.xlsx',
'510990_output.xlsx', '513000_output.xlsx', '513010_output.xlsx',
'513020_output.xlsx', '513030_output.xlsx', '513040_output.xlsx',
'513050_output.xlsx', '513060_output.xlsx', '513070_output.xlsx',
'513080_output.xlsx', '513090_output.xlsx', '513100_output.xlsx',
```

```
'513110_output.xlsx', '513120_output.xlsx', '513130_output.xlsx',
'513140_output.xlsx', '513150_output.xlsx', '513160_output.xlsx',
'513180_output.xlsx', '513190_output.xlsx', '513200_output.xlsx',
'513220_output.xlsx', '513230_output.xlsx', '513260_output.xlsx',
'513280_output.xlsx', '513290_output.xlsx', '513300_output.xlsx',
'513310_output.xlsx', '513320_output.xlsx', '513330_output.xlsx',
'513360_output.xlsx', '513380_output.xlsx', '513390_output.xlsx',
'513500_output.xlsx', '513520_output.xlsx', '513530_output.xlsx',
'513550_output.xlsx', '513560_output.xlsx', '513580_output.xlsx',
'513590_output.xlsx', '513600_output.xlsx', '513650_output.xlsx',
'513660_output.xlsx', '513690_output.xlsx', '513700_output.xlsx',
'513770_output.xlsx', '513800_output.xlsx', '513810_output.xlsx',
'513860_output.xlsx', '513880_output.xlsx', '513890_output.xlsx',
'513900_output.xlsx', '513950_output.xlsx', '513960_output.xlsx',
'513970_output.xlsx', '513980_output.xlsx', '513990_output.xlsx'
];
        % 读取数据
        data1 = xlsread(char(filename0(i)), 'Sheet2');
        data2 = xlsread(char(filename(j)), 'Sheet2');


        % 提取列数据
        column10_data1 = data1(:, 13);
        column10_data2 = data2(:, 8).*100;

        % 计算 Pearson 相关系数
        correlationCoefficient = corr(column10_data1, column10_data2);

        % 存储 Pearson 相关系数到矩阵中
       % correlationMatrix(i, j) = correlationCoefficient;
        correlationMatrix(j, i) = correlationCoefficient;
    end
end

% 显示 Pearson 相关系数矩阵
disp('Pearson 相关系数矩阵：');
disp(correlationMatrix);




% % 将相关系数矩阵转换成列向量，并记录对应的行和列索引
```

```
% [rows, cols] = size(correlationMatrix);
% correlationVector = correlationMatrix(triu(true(rows), 1));
%
% % 对相关系数进行降序排序，并获取前 10 个相关系数的索引
% [sortedCorrelation, sortedIndex] = sort(correlationVector, 'descend');
% top10Index = sortedIndex(1:20);
%
% % 将索引转换为行列坐标
% [rowIndex, colIndex] = ind2sub([rows, cols], top10Index);
%
% % 显示前 10 个相关系数和对应的行列索引
% disp('前 10 个相关系数：');
% disp(sortedCorrelation(1:20));
% disp('对应的行列索引：');
% disp([rowIndex, colIndex]);
%
% heatmap(correlationMatrix)
% xlabel('ETF 代号');
% ylabel('ETF 代号');
% title('ETF 收益率热图 ' );
```

2.绘制股指期货与跨境 ETF 成交价折线图

```
for i = 4:4     %手动更改 1——4
    for j = 1:82
        filename0=["IC.xlsx", 'IF.xlsx', 'IH.xlsx', 'IM.xlsx'];
        filename=["159501_output.xlsx", '159509_output.xlsx',
'159513_output.xlsx', '159519_output.xlsx', '159605_output.xlsx',
'159607_output.xlsx', '159612_output.xlsx', '159615_output.xlsx',
'159632_output.xlsx', '159655_output.xlsx', '159659_output.xlsx',
'159660_output.xlsx', '159687_output.xlsx', '159688_output.xlsx',
'159696_output.xlsx', '159699_output.xlsx', '159740_output.xlsx',
'159741_output.xlsx', '159742_output.xlsx', '159747_output.xlsx',
'159750_output.xlsx', '159822_output.xlsx', '159823_output.xlsx',
'159850_output.xlsx', '159866_output.xlsx', '159892_output.xlsx',
'510990_output.xlsx', '513000_output.xlsx', '513010_output.xlsx',
'513020_output.xlsx', '513030_output.xlsx', '513040_output.xlsx',
'513050_output.xlsx', '513060_output.xlsx', '513070_output.xlsx',
'513080_output.xlsx', '513090_output.xlsx', '513100_output.xlsx',
'513110_output.xlsx', '513120_output.xlsx', '513130_output.xlsx',
'513140_output.xlsx', '513150_output.xlsx', '513160_output.xlsx',
'513180_output.xlsx', '513190_output.xlsx', '513200_output.xlsx',
'513220_output.xlsx', '513230_output.xlsx', '513260_output.xlsx',
```

```
'513280_output.xlsx', '513290_output.xlsx', '513300_output.xlsx',
'513310_output.xlsx', '513320_output.xlsx', '513330_output.xlsx',
'513360_output.xlsx', '513380_output.xlsx', '513390_output.xlsx',
'513500_output.xlsx', '513520_output.xlsx', '513530_output.xlsx',
'513550_output.xlsx', '513560_output.xlsx', '513580_output.xlsx',
'513590_output.xlsx', '513600_output.xlsx', '513650_output.xlsx',
'513660_output.xlsx', '513690_output.xlsx', '513700_output.xlsx',
'513770_output.xlsx', '513800_output.xlsx', '513810_output.xlsx',
'513860_output.xlsx', '513880_output.xlsx', '513890_output.xlsx',
'513900_output.xlsx', '513950_output.xlsx', '513960_output.xlsx',
'513970_output.xlsx', '513980_output.xlsx', '513990_output.xlsx'
];
        name0=["IC", 'IF', 'IH', 'IM'];
        name=["159501", "159509", "159513", "159519", "159605", "159607",
"159612", "159615", "159632", "159655", "159659", "159660", "159687",
"159688", "159696", "159699", "159740", "159741", "159742", "159747",
"159750", "159822", "159823", "159850", "159866", "159892", "510990",
"513000", "513010", "513020", "513030", "513040", "513050", "513060",
"513070", "513080", "513090", "513100", "513110", "513120", "513130",
"513140", "513150", "513160", "513180", "513190", "513200", "513220",
"513230", "513260", "513280", "513290", "513300", "513310", "513320",
"513330", "513360", "513380", "513390", "513500", "513520", "513530",
"513550", "513560", "513580", "513590", "513600", "513650", "513660",
"513690", "513700", "513770", "513800", "513810", "513860", "513880",
"513890", "513900", "513950", "513960", "513970", "513980", "513990"];
    currentname0 = char(name0(i));
    currentname = char(name(j));

    data1 = xlsread(char(filename0(i)), 'Sheet2');
    data2 = xlsread(char(filename(j)), 'Sheet2');

    %获取列的数据
    P1 = data1(:, 13);
    P2 = data2(:, 8).*100;

    %绘制折线图
    figure;
    plot(P1, '-o', 'DisplayName', currentname0); % currentname0 的折线图
    hold on;
    plot(P2, '-s', 'DisplayName', currentname); % currentname 的折线图
    xlabel('日期');
    ylabel('成交价');
```

```
    legend('show');
    title(['成交价折线图 IM 与' currentname]);

    %保存折线图为图片文件（可以根据需要修改文件名和文件格式）
    imgFileName = ['成交价折线图 IM 与' currentname  '.png'];
    saveas(gcf, imgFileName);

    %关闭当前图形窗口，以便下一次循环时重新创建新的图形窗口
     close(gcf);
     end
end
```

3.计算 ADF，看是否通过检验

```
clear
clc

results = table();
lagValue = 2;

for i = 1:43    %手动更改 filename0 对应股指期货的 ADF 或 filename 对应 ETF 的 ADF
         filename0=["IC.xlsx", 'IF.xlsx', 'IH.xlsx', 'IM.xlsx'];
          filename=["159605_output.xlsx", "159607_output.xlsx",
"159688_output.xlsx", "159740_output.xlsx", "159741_output.xlsx",
"159742_output.xlsx", "159747_output.xlsx", "159750_output.xlsx",
"159822_output.xlsx", "159823_output.xlsx", "159850_output.xlsx",
"510990_output.xlsx", "513010_output.xlsx", "513020_output.xlsx",
"513040_output.xlsx", "513050_output.xlsx", "513070_output.xlsx",
"513090_output.xlsx", "513130_output.xlsx", "513150_output.xlsx",
"513160_output.xlsx", "513180_output.xlsx", "513220_output.xlsx",
"513230_output.xlsx", "513260_output.xlsx", "513320_output.xlsx",
"513330_output.xlsx", "513380_output.xlsx", "513550_output.xlsx",
"513560_output.xlsx", "513580_output.xlsx", "513590_output.xlsx",
"513600_output.xlsx", "513660_output.xlsx", "513690_output.xlsx",
"513770_output.xlsx", "513860_output.xlsx", "513890_output.xlsx",
"513900_output.xlsx", "513950_output.xlsx", "513960_output.xlsx",
"513970_output.xlsx", "513980_output.xlsx"];
    data1 = xlsread(char(filename(i)), 'Sheet2');

    %第八列：价格    第九列：折溢率    第十列：收益率
    P1 = data1(:,8).*100;
    P1=log(P1);
    diffData = diff(P1);
```

```
% 进行 ADF 检验
[h1, pValue1, stat1, cValue1] = adftest(diffData, 'model', 'AR', 'lags',
lagValue);

    % 存储结果到数组或表格中
    % 这里将结果存储为一个表格
    currentResult = table(pValue1, stat1, cValue1);

    % 将当前表格的结果追加到存储结果的表格中
    results = [results; currentResult];
end
```

4.计算组合的 OLS 回归方程

```
clc
clear
results = [];
for i = 4:4  %手动更改 1——4
    for j = 1:6
        filename0=["IC.xlsx", 'IF.xlsx', 'IH.xlsx', 'IM.xlsx'];
       filename=["510990_output.xlsx", "513600_output.xlsx",
"159822_output.xlsx", "159823_output.xlsx", "159850_output.xlsx",
"513900_output.xlsx"];

     %OLS 回归计算 mdl = fitlm(X, Y);
    data1 = xlsread(char(filename0(i)), 'Sheet2');
    data2 = xlsread(char(filename(j)), 'Sheet2');


    P1 = data1(:, 13);
    P2 = data2(:, 8).*100;

    P1=log(P1);
    P2 = log(P2);
    % 进行 OLS 回归
    mdl = fitlm(P2, P1);

   % 获取回归系数、P 值和 R^2 值
    coefficients = mdl.Coefficients.Estimate;
    coefficients=coefficients';
    pValues = mdl.Coefficients.pValue;
```

```
    rSquared = mdl.Rsquared.Ordinary;

    % 将当前表格的回归系数、P 值和 R^2 值追加到结果数组中
    results = [results; coefficients, pValues(2), rSquared];

    end
end
%将结果数组转换为表格
resultsTable = array2table(results, 'VariableNames',
{'Intercept','X_Coefficient', 'P_Value', 'R_Squared'});
% 显示或保存 resultsTable 表格，或者按照需要进行进一步处理
disp(resultsTable);

for i = 4:4
    for j = 1:6
        filename0=["IC.xlsx", 'IF.xlsx', 'IH.xlsx', 'IM.xlsx'];
       filename=["510990_output.xlsx", "513600_output.xlsx",
"159822_output.xlsx", "159823_output.xlsx", "159850_output.xlsx",
"513900_output.xlsx"];

    data1 = xlsread(char(filename0(i)), 'Sheet2');
    data2 = xlsread(char(filename(j)), 'Sheet2');
    P1 = data1(:, 13);
    P2 = data2(:, 8).*100;

    P1=log(P1);
    P2 = log(P2);
    % 构造回归方程: Y = Intercept + X_Coefficient * X
    Intercept = results(:, 1); % 截距
    X_Coefficient = results(:, 2); % X 系数
    predicted_Y=[];

    jieju=Intercept(j,: );
    xishu= X_Coefficient(j,:);
    % 计算 Y 的拟合值
    predicted_Y = jieju + xishu .* P2;
    residuals=[];
    % 计算残差（真实值减去拟合值）
    residuals = P1 - predicted_Y;
    data3 = xlsread("OLS.xlsx", 'Sheet4');
    data3(:,2*j-1)=predicted_Y;
    data3(:,2*j)=residuals;
```

```
    xlswrite("OLS.xlsx", data3, 'Sheet4');
    % 创建新的表格，将拟合值和残差放入表格中
    resultsT  = table(predicted_Y, residuals, 'VariableNames',
{'Predicted_Y', 'Residuals'});

    % 显示或保存 resultsTable 表格，或者按照需要进行进一步处理
    disp(resultsT);
        end
    end
```

5.计算组合的 OLS 回归方程的残差的 ADF

```
results = table();
lagValue = 2;
for i = 1:6

    data3 = xlsread("OLS.xlsx", 'IM'); % 手动更改 IM、IF、IC、
    residuals= data3(:,2*(i));
% 进行 ADF 检验
[h1, pValue1, stat1, cValue1] = adftest(residuals, 'model', 'AR', 'lags',
lagValue);

    % 存储结果到数组或表格中
    % 这里将结果存储为一个表格
    currentResult = table(pValue1, stat1);

    % 将当前表格的结果追加到存储结果的表格中
    results = [results; currentResult];
end

clc
```

6.计算跨境 ETF 间的 Pearson 相关系数

```
% 初始化存储 Pearson 相关系数的矩阵
%correlationMatrix = zeros(88);
% 循环遍历所有 Excel 表格
% for i = 88:88
%     for j = 15:47
%         filename=["159501_output.xlsx", '159502_output.xlsx',
'159509_output.xlsx', '159513_output.xlsx', '159519_output.xlsx',
'159605_output.xlsx', '159607_output.xlsx', '159612_output.xlsx',
'159615_output.xlsx', '159632_output.xlsx', '159655_output.xlsx',
```

```
'159659_output.xlsx', '159660_output.xlsx', '159687_output.xlsx',
'159688_output.xlsx', '159696_output.xlsx', '159699_output.xlsx',
'159740_output.xlsx', '159741_output.xlsx', '159742_output.xlsx',
'159747_output.xlsx', '159750_output.xlsx', '159822_output.xlsx',
'159823_output.xlsx', '159850_output.xlsx', '159866_output.xlsx',
'159892_output.xlsx', '510990_output.xlsx', '513000_output.xlsx',
'513010_output.xlsx', '513020_output.xlsx', '513030_output.xlsx',
'513040_output.xlsx', '513050_output.xlsx', '513060_output.xlsx',
'513070_output.xlsx', '513080_output.xlsx', '513090_output.xlsx',
'513100_output.xlsx', '513110_output.xlsx', '513120_output.xlsx',
'513130_output.xlsx', '513140_output.xlsx', '513150_output.xlsx',
'513160_output.xlsx', '513180_output.xlsx', '513190_output.xlsx',
'513193_output.xlsx', '513200_output.xlsx', '513220_output.xlsx',
'513230_output.xlsx', '513260_output.xlsx', '513280_output.xlsx',
'513290_output.xlsx', '513300_output.xlsx', '513310_output.xlsx',
'513320_output.xlsx', '513330_output.xlsx', '513360_output.xlsx',
'513380_output.xlsx', '513390_output.xlsx', '513500_output.xlsx',
'513520_output.xlsx', '513530_output.xlsx', '513550_output.xlsx',
'513560_output.xlsx', '513580_output.xlsx', '513590_output.xlsx',
'513600_output.xlsx', '513650_output.xlsx', '513660_output.xlsx',
'513690_output.xlsx', '513700_output.xlsx', '513770_output.xlsx',
'513800_output.xlsx', '513810_output.xlsx', '513853_output.xlsx',
'513854_output.xlsx', '513860_output.xlsx', '513873_output.xlsx',
'513880_output.xlsx', '513890_output.xlsx', '513900_output.xlsx',
'513950_output.xlsx', '513960_output.xlsx', '513970_output.xlsx',
'513980_output.xlsx', '513990_output.xlsx'
% ];
%         % 读取数据
%         data1 = xlsread(char(filename(i)), 'Sheet2');
%         data2 = xlsread(char(filename(j)), 'Sheet2');
%
%
%         % 提取第 9 列数据
%         column10_data1 = data1(:, 10);
%         column10_data2 = data2(:, 10);
%
%         % 计算 Pearson 相关系数
%         correlationCoefficient = corr(column10_data1, column10_data2);
%
%         % 存储 Pearson 相关系数到矩阵中
%         correlationMatrix(i, j) = correlationCoefficient;
%         correlationMatrix(j, i) = correlationCoefficient;
```

```
%     end
% end
%
% % 显示 Pearson 相关系数矩阵
% disp('Pearson 相关系数矩阵：');
% disp(correlationMatrix);



% 将相关系数矩阵转换成列向量，并记录对应的行和列索引
[rows, cols] = size(correlationMatrix);
correlationVector = correlationMatrix(triu(true(rows), 1));

% 对相关系数进行降序排序，并获取前 10 个相关系数的索引
[sortedCorrelation, sortedIndex] = sort(correlationVector, 'descend');
top10Index = sortedIndex(1:20);

% 将索引转换为行列坐标
[rowIndex, colIndex] = ind2sub([rows, cols], top10Index);

% 显示前 10 个相关系数和对应的行列索引
disp('前 10 个相关系数：');
disp(sortedCorrelation(1:20));
disp('对应的行列索引：');
disp([rowIndex, colIndex]);

heatmap(correlationMatrix)
xlabel('ETF 代号');
ylabel('ETF 代号');
title('ETF 收益率热图 ' );
```

7.计算跨境 ETF 间的 Pearson 相关系数

```
clc

%初始化存储 Pearson 相关系数的矩阵
correlationMatrix = zeros(88);
%循环遍历所有 Excel 表格
for i = 42:42
    for j = 81:88
```

```
        filename=["159501_output.xlsx", '159502_output.xlsx',
'159509_output.xlsx', '159513_output.xlsx', '159519_output.xlsx',
'159605_output.xlsx', '159607_output.xlsx', '159612_output.xlsx',
'159615_output.xlsx', '159632_output.xlsx', '159655_output.xlsx',
'159659_output.xlsx', '159660_output.xlsx', '159687_output.xlsx',
'159688_output.xlsx', '159696_output.xlsx', '159699_output.xlsx',
'159740_output.xlsx', '159741_output.xlsx', '159742_output.xlsx',
'159747_output.xlsx', '159750_output.xlsx', '159822_output.xlsx',
'159823_output.xlsx', '159850_output.xlsx', '159866_output.xlsx',
'159892_output.xlsx', '510990_output.xlsx', '513000_output.xlsx',
'513010_output.xlsx', '513020_output.xlsx', '513030_output.xlsx',
'513040_output.xlsx', '513050_output.xlsx', '513060_output.xlsx',
'513070_output.xlsx', '513080_output.xlsx', '513090_output.xlsx',
'513100_output.xlsx', '513110_output.xlsx', '513120_output.xlsx',
'513130_output.xlsx', '513140_output.xlsx', '513150_output.xlsx',
'513160_output.xlsx', '513180_output.xlsx', '513190_output.xlsx',
'513193_output.xlsx', '513200_output.xlsx', '513220_output.xlsx',
'513230_output.xlsx', '513260_output.xlsx', '513280_output.xlsx',
'513290_output.xlsx', '513300_output.xlsx', '513310_output.xlsx',
'513320_output.xlsx', '513330_output.xlsx', '513360_output.xlsx',
'513380_output.xlsx', '513390_output.xlsx', '513500_output.xlsx',
'513520_output.xlsx', '513530_output.xlsx', '513550_output.xlsx',
'513560_output.xlsx', '513580_output.xlsx', '513590_output.xlsx',
'513600_output.xlsx', '513650_output.xlsx', '513660_output.xlsx',
'513690_output.xlsx', '513700_output.xlsx', '513770_output.xlsx',
'513800_output.xlsx', '513810_output.xlsx', '513853_output.xlsx',
'513854_output.xlsx', '513860_output.xlsx', '513873_output.xlsx',
'513880_output.xlsx', '513890_output.xlsx', '513900_output.xlsx',
'513950_output.xlsx', '513960_output.xlsx', '513970_output.xlsx',
'513980_output.xlsx', '513990_output.xlsx'
];
        % 读取数据
        data1 = xlsread(char(filename(i)), 'Sheet2');
        data2 = xlsread(char(filename(j)), 'Sheet2');


        % 提取列数据
        column10_data1 = data1(:, 8);
        column10_data2 = data2(:, 8);

        % 计算 Pearson 相关系数
        correlationCoefficient = corr(column10_data1, column10_data2);
```

```
        % 存储 Pearson 相关系数到矩阵中
        correlationMatrix(i, j) = correlationCoefficient;
        correlationMatrix(j, i) = correlationCoefficient;
    end
end

% 显示 Pearson 相关系数矩阵
disp('Pearson 相关系数矩阵：');
disp(correlationMatrix);




% % 将相关系数矩阵转换成列向量，并记录对应的行和列索引
% [rows, cols] = size(correlationMatrix);
% correlationVector = correlationMatrix(triu(true(rows), 1));
%
% % 对相关系数进行降序排序，并获取前 10 个相关系数的索引
% [sortedCorrelation, sortedIndex] = sort(correlationVector, 'descend');
% top10Index = sortedIndex(1:20);
%
% % 将索引转换为行列坐标
% [rowIndex, colIndex] = ind2sub([rows, cols], top10Index);
%
% % 显示前 10 个相关系数和对应的行列索引
% disp('前 10 个相关系数：');
% disp(sortedCorrelation(1:20));
% disp('对应的行列索引：');
% disp([rowIndex, colIndex]);
%
% heatmap(correlationMatrix)
% xlabel('ETF 代号');
% ylabel('ETF 代号');
% title('ETF 收益率热图 ' );
```

8.绘制跨境 ETF 间折线图

```
for i = 1:1
     for j = 2:41
         filename=["513130_output.xlsx" "513580_output.xlsx"
"513260_output.xlsx" "513010_output.xlsx" "513380_output.xlsx"
"513180_output.xlsx" "159747_output.xlsx" "159742_output.xlsx"
"159740_output.xlsx" "159741_output.xlsx" "513890_output.xlsx"
```

```
"513230_output.xlsx" "159607_output.xlsx" "159750_output.xlsx"
"159605_output.xlsx" "513220_output.xlsx" "159822_output.xlsx"
"513070_output.xlsx" "513590_output.xlsx" "513020_output.xlsx"
"513320_output.xlsx" "513860_output.xlsx" "513050_output.xlsx"
"513960_output.xlsx" "513560_output.xlsx" "159688_output.xlsx"
"513160_output.xlsx" "513330_output.xlsx" "513770_output.xlsx"
"513040_output.xlsx" "513970_output.xlsx" "159823_output.xlsx"
"513900_output.xlsx" "513980_output.xlsx" "159850_output.xlsx"
"513150_output.xlsx" "513600_output.xlsx" "513550_output.xlsx"
"513660_output.xlsx" "513090_output.xlsx" "510990_output.xlsx"];
          name=["513130" "513580" "513260" "513010" "513380" "513180"
"159747" "159742" "159740" "159741" "513890" "513230" "159607" "159750"
"159605" "513220" "159822" "513070" "513590" "513020" "513320" "513860"
"513050" "513960" "513560" "159688" "513160" "513330" "513770" "513040"
"513970" "159823" "513900" "513980" "159850" "513150" "513600" "513550"
"513660" "513090" "510990"
];
    currentname = char(name(j));

    data1 = xlsread(char(filename(i)), 'Sheet2');
    data2 = xlsread(char(filename(j)), 'Sheet2');

    %获取第九列和第十列的数据
    P1 = data1(:,10);
    P2 = data2(:, 10);

    %绘制折线图
    figure;
    plot(P1, '-o', 'DisplayName', '513130'); % 第九列的折线图
    hold on;
    plot(P2, '-s', 'DisplayName', currentname); % 第十列的折线图
    xlabel('日期');
    ylabel('收益率折线图');
    legend('show');
    title(['收益率折线图_513130 与 ' currentname]);

    %保存折线图为图片文件（可以根据需要修改文件名和文件格式）
    imgFileName = ['收益率折线图_513130 与' char(name(j)) '.png'];
    saveas(gcf, imgFileName);

    %关闭当前图形窗口，以便下一次循环时重新创建新的图形窗口
     close(gcf);
```

```
    end
end
```

9.计算检验跨境 ETF 的 ADF

```
clear
clc

results = table();
lagValue = 2;

for i = 1:74

        filename=["513130_output.xlsx"  "513580_output.xlsx"
"513260_output.xlsx"  "513010_output.xlsx"  "513380_output.xlsx"
"513180_output.xlsx"  "159747_output.xlsx"  "159742_output.xlsx"
"159740_output.xlsx"  "159741_output.xlsx"  "513890_output.xlsx"
"513230_output.xlsx"  "159607_output.xlsx"  "159750_output.xlsx"
"159605_output.xlsx"  "513220_output.xlsx"  "159822_output.xlsx"
"513070_output.xlsx"  "513590_output.xlsx"  "513020_output.xlsx"
"513320_output.xlsx"  "513860_output.xlsx"  "513050_output.xlsx"
"513960_output.xlsx"  "513560_output.xlsx"  "159688_output.xlsx"
"513160_output.xlsx"  "513330_output.xlsx"  "513770_output.xlsx"
"513040_output.xlsx"  "513970_output.xlsx"  "159823_output.xlsx"
"513900_output.xlsx"  "513980_output.xlsx"  "159850_output.xlsx"
"513150_output.xlsx"  "513600_output.xlsx"  "513550_output.xlsx"
"513660_output.xlsx"  "513090_output.xlsx"  "510990_output.xlsx"
"513030_output.xlsx"  "513080_output.xlsx"  "513950_output.xlsx"
"513650_output.xlsx"  "513690_output.xlsx"  "159513_output.xlsx"
"159687_output.xlsx"  "513880_output.xlsx"  "513140_output.xlsx"
"513000_output.xlsx"  "159866_output.xlsx"  "513520_output.xlsx"
"513290_output.xlsx"  "159659_output.xlsx"  "159501_output.xlsx"
"159655_output.xlsx"  "159612_output.xlsx"  "513500_output.xlsx"
"513310_output.xlsx"  "159660_output.xlsx"  "513300_output.xlsx"
"513100_output.xlsx"  "513530_output.xlsx"  "159632_output.xlsx"
"513110_output.xlsx"  "513390_output.xlsx"  "513280_output.xlsx"
"513700_output.xlsx"  "513200_output.xlsx"  "159615_output.xlsx"
"513800_output.xlsx"  "159892_output.xlsx"  "513360_output.xlsx"
];
         name=["513130" "513580" "513260" "513010" "513380" "513180"
"159747" "159742" "159740" "159741" "513890" "513230" "159607" "159750"
"159605" "513220" "159822" "513070" "513590" "513020" "513320" "513860"
"513050" "513960" "513560" "159688" "513160" "513330" "513770" "513040"
```

```
"513970" "159823" "513900" "513980" "159850" "513150" "513600" "513550"
"513660" "513090" "510990"
];
    currentname = char(name(1));

    data1 = xlsread(char(filename(i)), 'Sheet2');


    %第八列：价格    第九列：折溢率    第十列：收益率
    P1 = data1(:,8);
    P1=log(P1);
    diffData = diff(P1);


% 进行 ADF 检验
[h1, pValue1, stat1, cValue1] = adftest(diffData, 'model', 'AR', 'lags',
lagValue);

    % 存储结果到数组或表格中
    % 这里将结果存储为一个表格
    currentResult = table(pValue1, stat1, cValue1);

    % 将当前表格的结果追加到存储结果的表格中
    results = [results; currentResult];
end
```

9.计算跨境 ETF 的 OLS 回归方程

```
clc
% clear
% results = [];
% for i = 1:1
%     for j = 2:8
%          filename=["513130_output.xlsx" "513090_output.xlsx"
"159605_output.xlsx"  "513050_output.xlsx"  "513030_output.xlsx"
"513220_output.xlsx"  "159607_output.xlsx"  "513860_output.xlsx"
% ];
%      %OLS 回归计算 mdl = fitlm(X, Y);
%     data1 = xlsread(char(filename(i)), 'Sheet2');
%     data2 = xlsread(char(filename(j)), 'Sheet2');
%
%     %第八列：价格    第九列：折溢率    第十列：收益率
%
```

```
%     P1 = data1(:,8).*100;
%     P1=log(P1);
%     P2 = data2(:,8).*100;
%     P2 = log(P2);
%     % 进行 OLS 回归
%     mdl = fitlm(P2, P1);
%
%    % 获取回归系数、P 值和 R^2 值
%     coefficients = mdl.Coefficients.Estimate;
%     coefficients=coefficients';
%     pValues = mdl.Coefficients.pValue;
%     rSquared = mdl.Rsquared.Ordinary;
%
%     % 将当前表格的回归系数、P 值和 R^2 值追加到结果数组中
%     results = [results; coefficients, pValues(2), rSquared];
%
%     end
% end
% %将结果数组转换为表格
% resultsTable = array2table(results, 'VariableNames',
{'Intercept','X_Coefficient', 'P_Value', 'R_Squared'});
% % 显示或保存 resultsTable 表格，或者按照需要进行进一步处理
% disp(resultsTable);

for i = 1:1
    for j = 2:8
         filename=["513130_output.xlsx" "513090_output.xlsx"
"159605_output.xlsx"  "513050_output.xlsx"  "513030_output.xlsx"
"513220_output.xlsx"  "159607_output.xlsx"  "513860_output.xlsx"
];

    data1 = xlsread(char(filename(i)), 'Sheet2');
    data2 = xlsread(char(filename(j)), 'Sheet2');

    %第八列：价格    第九列：折溢率    第十列：收益率

    P1 = data1(:,8).*100;
    P1=log(P1);
    P2 = data2(:,8).*100;
    P2 = log(P2);
% 构造回归方程: Y = Intercept + X_Coefficient * X
Intercept = results(:, 1); % 截距
```

```
X_Coefficient = results(:, 2); % X 系数
predicted_Y=[];

jieju=Intercept(j-1,: );
xishu= X_Coefficient(j-1,:);
% 计算 Y 的拟合值
predicted_Y = jieju + xishu .* P2;
residuals=[];
% 计算残差（真实值减去拟合值）
residuals = P1 - predicted_Y;
data3 = xlsread("OLS.xlsx", 'Sheet3');
data3(:,2*(j-1)-1)=predicted_Y;
data3(:,2*(j-1))=residuals;
xlswrite("OLS.xlsx", data3, 'Sheet3');
% 创建新的表格，将拟合值和残差放入表格中
resultsT  = table(predicted_Y, residuals, 'VariableNames', {'Predicted_Y',
'Residuals'});

% 显示或保存 resultsTable 表格，或者按照需要进行进一步处理
disp(resultsT);
    end
end
```

11.计算跨境 ETF 的 GARCH

```
results = table();
lagValue = 2;
for i = 1:73
    data3 = xlsread("OLS.xlsx", 'Sheet2');
    residuals= data3(:,2*(i));
% 进行 ADF 检验
[h1, pValue1, stat1, cValue1] = adftest(residuals, 'model', 'AR', 'lags',
lagValue);

    % 存储结果到数组或表格中
    % 这里将结果存储为一个表格
    currentResult = table(pValue1, stat1);

    % 将当前表格的结果追加到存储结果的表格中
    results = [results; currentResult];
end
```

```
clc
clear
% 使用 garch 函数估计 GARCH(1,1)模型
 for j = 1:1

    data3 = xlsread("OLS.xlsx", 'Sheet3');
    returns=data3(:,2*j);
    spec = garch(1,1);
    model = estimate(spec, returns);

    % 查看模型的参数
    disp(model);

    % 预测未来的波动率
    numPeriods = 1; % 预测的期数
    [forecastVolatility, forecastMean] = forecast(model, numPeriods);
    % 使用 GARCH 模型的 forecast 函数生成波动率预测
    numPeriods = 10; % 想要预测的期数
    [forecastVolatility, forecastMean] = forecast(model, numPeriods, 'Y0',
returns);

    % 输出预测的波动率
    disp(forecastVolatility);

    % 绘制实际波动率和预测波动率的图表
    plot(returns);
    hold on;
    plot([nan(size(returns)); forecastVolatility], 'r', 'LineWidth', 2);
    legend('实际波动率', '预测波动率');
    xlabel('时间');
    ylabel('波动率');
    title('GARCH(1,1)模型预测波动率');
 end
```