

任务一：数据获取

实验内容： 获取股票、指数、无风险利率及Fama-French三因子数据，保存至CSV文件。

选择的股票（覆盖不同行业）：

- **600519.SH** — 贵州茅台（消费/白酒）
- **601012.SH** — 隆基绿能（能源/光伏）
- **300750.SZ** — 宁德时代（科技/新能源电池）
- **600276.SH** — 恒瑞医药（医药）

数据范围： 2023年1月1日 ~ 2025年12月31日

```
In [ ]: import warnings
warnings.filterwarnings('ignore')

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams['font.sans-serif'] = ['SimHei']
matplotlib.rcParams['axes.unicode_minus'] = False

import tushare as ts
from scipy import stats
import statsmodels.api as sm
from statsmodels.stats.diagnostic import het_breuschpagan
from statsmodels.stats.stattools import durbin_watson
import time
import os

print("所有库导入成功！")
```

```
In [ ]: # ===== 任务1：数据获取 =====
# 设置Tushare Token
TOKEN = 'd4ade4dc2337bb63733c3ec90deb49bfe013ac6b81452bdba73641e7'
ts.set_token(TOKEN)
pro = ts.pro_api()

# 参数设置
START_DATE = '20230101'
END_DATE = '20251231'

# 选择的股票（覆盖不同行业）
STOCKS = {
    '600519.SH': '贵州茅台(消费)',
    '601012.SH': '隆基绿能(能源)',
    '300750.SZ': '宁德时代(科技)',
    '600276.SH': '恒瑞医药(医药)'
}
```

```

# 指数
INDICES = {
    '000001.SH': '上证综指',
    '000300.SH': '沪深300'
}

print("参数设置完成")
print(f"时间范围: {START_DATE} ~ {END_DATE}")
print(f"选定股票: {list(STOCKS.values())}")
print(f"选定指数: {list(INDICES.values())}")

```

```

In [ ]: # 1.1 获取个股日线数据
stock_data_list = []
for code, name in STOCKS.items():
    print(f"正在获取 {name} ({code}) 的日线数据...")
    df = pro.daily(ts_code=code, start_date=START_DATE, end_date=END_DATE)
    df['stock_name'] = name
    stock_data_list.append(df)
    time.sleep(0.3)

stock_daily = pd.concat(stock_data_list, ignore_index=True)
stock_daily['trade_date'] = pd.to_datetime(stock_daily['trade_date'])
stock_daily = stock_daily.sort_values(['ts_code', 'trade_date']).reset_index(drop=True)

print(f"\n个股日线数据获取完成, 共 {len(stock_daily)} 条记录")
print(stock_daily[['ts_code', 'stock_name', 'trade_date', 'open', 'close', 'vol']])

```

```

In [ ]: # 1.2 获取指数日线数据
index_data_list = []
for code, name in INDICES.items():
    print(f"正在获取 {name} ({code}) 的日线数据...")
    df = pro.index_daily(ts_code=code, start_date=START_DATE, end_date=END_DATE)
    df['index_name'] = name
    index_data_list.append(df)
    time.sleep(0.3)

index_daily = pd.concat(index_data_list, ignore_index=True)
index_daily['trade_date'] = pd.to_datetime(index_daily['trade_date'])
index_daily = index_daily.sort_values(['ts_code', 'trade_date']).reset_index(drop=True)

print(f"\n指数日线数据获取完成, 共 {len(index_daily)} 条记录")
print(index_daily[['ts_code', 'index_name', 'trade_date', 'open', 'close', 'vol']])

```

```

In [ ]: # 1.3 获取中国国债收益率作为无风险利率
# 使用Tushare的国债收益率接口 (1年期国债到期收益率)
# 注意: 该接口限制每分钟2次调用, 需等待35秒
print("正在获取国债收益率数据 (无风险利率) ...")

rf_list = []
for year in range(2023, 2026):
    s = f'{year}0101'
    e = f'{year}1231'
    print(f"  获取{year}年数据...")
    df_rf = pro.yc_cb(ts_code='1001.CB', curve_type='0', start_date=s, end_date=e)

```

```

if df_rf is not None and len(df_rf) > 0:
    rf_list.append(df_rf)
    print(f"    成功, 共 {len(df_rf)} 条记录")
else:
    print(f"    {year}年返回数据为空")
if year < 2025:
    print("    等待35秒 (API频率限制: 2次/分钟) ...")
    time.sleep(35)

rf_data = pd.concat(rf_list, ignore_index=True)
rf_data['trade_date'] = pd.to_datetime(rf_data['trade_date'])
# 取1年期收益率作为无风险利率
rf_1y = rf_data[rf_data['curve_term'] == 1][['trade_date', 'yield']].copy()
rf_1y = rf_1y.rename(columns={'yield': 'rf_annual_pct'})
rf_1y = rf_1y.sort_values('trade_date').drop_duplicates('trade_date', keep='first')
# 年化利率(%)转日利率
rf_1y['rf_daily'] = rf_1y['rf_annual_pct'] / 100 / 252
print(f"\n国债收益率数据获取完成, 共 {len(rf_1y)} 条记录")
print(rf_1y.head())

```

```

In [ ]: # 1.4 构建Fama-French三因子数据
# 由于Tushare fama_factor接口权限不足, 使用指数收益率差构建近似三因子 (中国A股市场常用方法)
# SMB (规模因子) ≈ 中证500 (中小盘) 收益率 - 沪深300 (大盘) 收益率
# HML (价值因子) ≈ 上证50 (价值蓝筹) 收益率 - 创业板指 (成长) 收益率
print("正在通过指数数据构建Fama-French三因子...")

factor_indices = {
    '000905.SH': '中证500', # 中小盘代理
    '000016.SH': '上证50', # 价值/大盘代理
    '399006.SZ': '创业板指', # 成长代理
}

factor_close = {}
for code, name in factor_indices.items():
    print(f"    获取 {name} ({code})...")
    df-fi = pro.index_daily(ts_code=code, start_date=START_DATE, end_date=END_DATE)
    df-fi['trade_date'] = pd.to_datetime(df-fi['trade_date'])
    df-fi = df-fi[['trade_date', 'close']].sort_values('trade_date').set_index('trade_date')
    factor_close[code] = df-fi['close']
    time.sleep(0.3)

# 合并为DataFrame并计算日对数收益率
factor_price = pd.DataFrame(factor_close).sort_index().dropna()
factor_ret = np.log(factor_price / factor_price.shift(1)).dropna()

# 沪深300收益率 (从Cell 5获取的index_daily中提取)
hs300_daily = index_daily[index_daily['ts_code'] == '000300.SH'][['trade_date', 'close']]
hs300_daily = hs300_daily.sort_values('trade_date').set_index('trade_date')
hs300_ret = np.log(hs300_daily['close'] / hs300_daily['close'].shift(1)).dropna()

# 构建三因子
ff_data = pd.DataFrame(index=factor_ret.index)
ff_data['mkt_rf'] = hs300_ret.reindex(ff_data.index) # 市场收益率
ff_data['smb'] = factor_ret['000905.SH'] - hs300_ret.reindex(factor_ret.index) # 规模因子
ff_data['hml'] = factor_ret['000016.SH'] - factor_ret['399006.SZ'] # 价值因子
ff_data = ff_data.dropna().reset_index().rename(columns={'index': 'trade_date'})

```

```

print(f"\n三因子数据构建完成，共 {len(ff_data)} 条记录")
print(ff_data.head(10))
print(f"\n因子描述性统计：")
print(ff_data[['mkt_rf', 'smb', 'hml']].describe())

```

```

In [ ]: # 1.5 验证三因子数据质量
print("=" * 60)
print("三因子数据质量检查")
print("=" * 60)
print(f"数据记录数: {len(ff_data)}")
print(f"日期范围: {ff_data['trade_date'].min()} ~ {ff_data['trade_date'].max()}")
print(f"\n各因子缺失值:\n{ff_data[['mkt_rf', 'smb', 'hml']].isnull().sum()}")
print(f"\n各因子基本统计:")
print(f"  MKT_RF  均值={ff_data['mkt_rf'].mean():.6f}, 标准差={ff_data['mkt_rf'].std()}")
print(f"  SMB      均值={ff_data['smb'].mean():.6f}, 标准差={ff_data['smb'].std():.6f}")
print(f"  HML      均值={ff_data['hml'].mean():.6f}, 标准差={ff_data['hml'].std():.6f}")
print("\n构建说明:")
print("  SMB = 中证500收益率 - 沪深300收益率 (规模因子: 小盘 - 大盘)")
print("  HML = 上证50收益率 - 创业板指收益率 (价值因子: 价值 - 成长)")

```

```

In [ ]: # 1.6 保存所有数据到CSV文件
stock_daily.to_csv('stock_daily.csv', index=False, encoding='utf-8-sig')
index_daily.to_csv('index_daily.csv', index=False, encoding='utf-8-sig')
rf_1y.to_csv('risk_free_rate.csv', index=False, encoding='utf-8-sig')
ff_data.to_csv('fama_french_factors.csv', index=False, encoding='utf-8-sig')

print("=" * 60)
print("任务1完成! 所有数据已保存至csv文件:")
print("  - stock_daily.csv      (个股日线数据)")
print("  - index_daily.csv      (指数日线数据)")
print("  - risk_free_rate.csv   (无风险利率数据)")
print("  - fama_french_factors.csv (三因子数据)")
print("=" * 60)

```