Factor Evaluation Framework 使用指南

概述

用于因子评估的Python框架，主要包含两个核心类：

- `FactorEvaluation`：主要的因子评估类
- `DataService`：数据服务类，用于获取K线数据和因子信号

核心功能

1. 因子评估指标

- **平稳性检验**: ADF检验判断因子是否平稳
- **相关性分析**: IC (Pearson) 和 Rank_IC (Spearman) 相关系数
- **信息比率**: IR值计算
- **分组收益分析**: 将因子分组后分析各组收益表现
- **因子分布分析**: 因子的统计分布特征
- **因子-收益率散点图**: 可视化因子值与未来收益率的关系（无趋势线，点大小适中）
- **滚动IC分析**: 分析因子预测能力的时间稳定性，显示IC和Rank IC的累积和趋势

2. 数据模式

- **单币种模式**: 分析单个数据集
- **多币种模式**: 同时分析多个币种和时间周期

快速开始

基本导入

```
from factor_evaluation_server import FactorEvaluation, DataService
import pandas as pd
import numpy as np
```

方式一：使用自己的数据（单币种模式）

```python
# 1. 准备数据 - DataFrame必须包含'close'列
df = pd.read_csv('your_data.csv')
df['open_time'] = pd.to_datetime(df['open_time'])
df = df.set_index('open_time')


# 2. 定义因子函数
def my_factor(df, window=20):
    """
    自定义因子函数
    参数: df - 包含价格数据的DataFrame
    返回: pandas.Series - 因子值
    """
    return df['close'].rolling(window).mean() / df['close'] - 1


# 3. 创建评估器并设置因子
evaluator = FactorEvaluation(
    df=df,                          # 数据
    future_return_periods=10,       # 未来收益计算周期
    factor_func=my_factor,          # 因子函数
    factor_name='ma_ratio',         # 因子名称
    window=20                       # 因子函数参数
)
```

```
# 4. 运行完整评估
results = evaluator.run_full_evaluation(
    n_groups_ir=10,                 # IR计算分组数
    n_groups_analysis=20,           # 分组收益分组数
    run_stationarity_test=True,     # 运行ADF检验
    scatter_s=1.0,                  # 散点图点大小，默认1.0
    initial_window=20000,           # 滚动IC初始窗口大小，默认20000
    rolling_step=2000               # 滚动IC滚动步长，默认2000
)
```

方式二：使用多币种数据

```
# 1. 创建多币种评估器
evaluator = FactorEvaluation(
    time_periods=['1h', '15m'],     # 时间周期列表
    future_return_periods=10,
    factor_func=my_factor,
    factor_name='ma_ratio',
    window=20
)


# 2. 运行评估
results = evaluator.run_full_evaluation()
```

因子函数编写规范

函数要求

1. **第一个参数必须是DataFrame**: 包含价格数据
2. **返回值必须是pandas.Series或numpy.array**: 与输入DataFrame长度一致

示例因子函数

```
def rsi_factor(df, window=14):
    """RSI因子"""
    delta = df['close'].diff()
    gain = (delta.where(delta > 0, 0)).rolling(window=window).mean()
    loss = (-delta.where(delta < 0, 0)).rolling(window=window).mean()
    rs = gain / loss
    rsi = 100 - (100 / (1 + rs))
    return rsi


def bollinger_position(df, window=20, std_mult=2):
    """布林带位置因子"""
    ma = df['close'].rolling(window).mean()
    std = df['close'].rolling(window).std()
    upper = ma + std_mult * std
    lower = ma - std_mult * std
    position = (df['close'] - lower) / (upper - lower)
    return position
```

```python
def volume_price_trend(df, window=10):
    """量价趋势因子"""
    price_change = df['close'].pct_change()
    volume_weighted = price_change * df['volume']
    return volume_weighted.rolling(window).mean()
```

数据服务使用

DataService类用法

```python
# 1. 创建数据服务
ds = DataService()

# 2. 获取K线数据
kline_data = ds['ETHUSDT_1h_2020_2025']  # 表名格式：币种_周期_时间范围

# 3. 获取因子信号(因子库内因子)
factor_signal = ds['ETHUSDT_1h_2020_2025', 'factor_name']  # (表名, 方法名)
```

新增图表功能

1. 因子-收益率散点图

图标说明

- 可视化因子值与未来收益率的关系
- 便于观察数据分布
- 显示Pearson相关系数

图表说明

滚动IC分析包含两个子图：

1. **上图**：滚动IC和Rank IC的累积和
   - 蓝线：IC累积和
   - 红线：Rank IC累积和
   - 显示因子预测能力的长期趋势
2. **下图**：单期滚动IC和Rank IC
   - 蓝线：各窗口的IC值
   - 红线：各窗口的Rank IC值
   - 显示因子预测能力的短期波动

结果解读

关键指标含义

1. **IC (Information Coefficient)**
   - 范围: [-1, 1]
   - 含义: 因子与未来收益的线性相关性
2. **Rank_IC**
   - 范围: [-1, 1]
   - 含义: 因子与未来收益的秩相关性
3. **IR (Information Ratio)**
   - 含义: IC的稳定性指标
   - 计算: mean(IC) / std(IC)
4. **分组收益分析**
   - 将因子值分组，观察各组平均收益
   - 好的因子应该呈现单调性

平稳性检验

- **ADF检验**: p_value < 0.05 表示因子平稳
- 平稳因子更适合用于策略构建

注意事项

1. **数据质量**: 确保输入数据没有重复索引和异常值
2. **未来数据**: 禁止使用未来数据,必要时使用walk-forward的方法进行检验
3. **未来收益周期**: 根据策略频率选择合适的future_return_periods
4. **分组数量**: 一般建议10-20组，数据量小时可减少分组
5. **多币种模式**: 会自动加载预定义的币种数据，确保数据源可用
6. **滚动IC分析**: 需要足够的数据量，至少要大于initial_window参数

常见问题

Q: 因子函数报错怎么办?

A: 检查函数是否正确处理了缺失值，确保返回值类型正确

Q: 如何提高因子表现?

A: 尝试不同的参数、组合多个因子、或者对因子进行变换（如排序、标准化）

Q: 多币种模式加载数据失败?

A: 检查网络连接和数据服务是否可用，确认币种和时间周期格式正确

Q: 滚动IC分析提示数据量不足?

A: 确保数据长度大于initial_window参数，或者减小initial_window的值

Q: 散点图点太小或太大看不清?

A: 调整s参数（点大小）和alpha参数（透明度），如s=2.0或s=0.5