### 回测框架

#### 版本：V1.0

* 框架内含有7份文件，分别为代码文件夹、输出文件夹、数据文件夹、main.py以及不同版本使用说明
* 代码文件夹：Single\_asset.py和Portfolio.py
* 输出文件夹：输出的两个excel文件以及4张净值图
* 数据文件夹：输入的数据excel文件
* main.py：回测框架运行文件
* 使用说明文件：md，word，pdf版本

#### 依赖包

* chinese\_calendar
* 安装方法：pip install chinesecalendar

#### 使用说明

* 数据excel文件有两个sheet，分别为数据sheet和权重sheet（**数据要求：**数据sheet中的开始日期要晚于权重sheet中的开始日期，以确保数据sheet在第一天能够有权重数据进行建仓）
* main.py中可以更改对应参数（**使用者**应按照策略中的年化天数、开始结束日期、交易费率、无风险收益率、输入输出地址以及基准标的来对main.py中的参数进行修改）
* 直接运行main.py，可得到单一资产、资产组合的分年度和整体表现的excel表格。

#### 代码函数定义

##### main.py

* **main函数**
* main()
* 配置参数，根据实际情况进行调整
* ann：年化时考虑的天数
* start：策略开始日期
* end : 策略结束日期
* fee\_rate：交易费率
* rf : 无风险利率
* input\_path：输入文件夹地址
* file：输入数据文件名
* output\_path：输出文件夹地址
* asset\_file：资产组合输出文件名
* single\_file：单一资产输出文件名
* single：单一资产名字

##### Single\_asset.py

* **表格读取函数**
* sheet\_read()
* 读取数据表格，返回DataFrame
* **数据预处理函数**
* data\_process()
* 读取data文件，更改日期列列名，计算日收益率并将数据增加至DataFrame中
* 同时创建years列，代表每个交易日对应的年份，用于按年份groupby
* 返回DataFrame
* **标的总收益函数**
* wholeR(data)
* 计算标的整个策略期的总收益，返回Float
* data：数据表格
* **日收益率函数**  
  dailyR(data)
* 计算标的每日的收益率，返回Series
* data：读取到的数据表格，类型为DataFrame
* **年化收益率函数**  
  annualR(data, gb, period)
* 计算标的年化收益率，若为分年度，返回Series；若为整体，返回Float
* data：数据表格
* gb：将data按照年份进行groupby
* period：'by year' or 'all period'，分别表示分年度和整体
* **年化波动率函数**
* annualV(data, gb, period)
* 计算标的年化波动率，若为分年度，返回Series；若为整体，返回Float
* data：数据表格
* gb：将data按照年份进行groupby
* period：'by year' or 'all period'，分别表示分年度和整体
* **最大回撤函数**  
  max\_dd(returns)
* 计算标的最大回撤，返回最大回撤、回撤开始日期以及回撤结束日期
* returns：标的数据的日收益率
* **夏普比率函数**  
  sharpe(data, gb, period)
* 计算标的夏普比率，若为分年度，返回DataFrame；若为整体，返回Float
* data：数据表格
* gb：将data按照年份进行groupby
* period：'by year' or 'all period'，分别表示分年度和整体
* **carmar比率函数**  
  carmar(data, gb, period)
* 计算标的carmar比率，若为分年度，返回DataFrame；若为整体，返回Float
* data：数据表格
* gb：将data按照年份进行groupby
* period：'by year' or 'all period'，分别表示分年度和整体
* **输出函数**  
  output(data, df\_year, df\_all)
* 输出单一资产分年度和整体表现的excel表格
* df\_year：回测后得到的单一资产分年度表现DataFrame
* df\_all：回测后得到的单一资产整体表现DataFrame
* **回测函数**  
  single\_backtest()
* Single\_asset.py主函数，分别计算分年度和整体表现，并输出excel表格

##### **Portfolio.py**

* **表格读取函数**  
  sheet\_read()
* 分别读取数据表格和权重表格，返回两个DataFrame
* **数据预处理函数**  
  data\_clean(data, net)
* 生成日收益率、净值和年份的列，返回新的DataFrame
* data：数据表格
* net：净值Series
* **交易日处理函数**  
  time\_process(weight)
* 考虑到调仓日可能存在非交易日、节假日以及非周一到周五的情况
* 该函数能够将权重表格中的日期调整为交易日日期
* 返回Series
* weight：权重表格
* **标的总收益函数**
* wholeR(data)
* 计算标的整个策略期的总收益，返回Float
* data：数据表格
* **日收益率函数**  
  dailyR(data)
* 计算标的每日的收益率，返回Series
* data：数据表格
* **年化收益率函数**  
  annualR(data, gb, period)
* 计算标的年化收益率，若为分年度，返回Series；若为整体，返回Float
* data：数据表格
* gb：将data按照年份进行groupby
* period：'by year' or 'all period'，分别表示分年度和整体
* **年化波动率函数**  
  annualV(data, gb, period)
* 计算标的年化波动率，若为分年度，返回Series；若为整体，返回Float
* data：数据表格
* gb：将data按照年份进行groupby
* period：'by year' or 'all period'，分别表示分年度和整体
* **最大回撤函数**  
  max\_dd(returns)
* 计算标的最大回撤，返回最大回撤、回撤开始日期以及回撤结束日期
* returns：标的数据的日收益率
* **夏普比率函数**  
  sharpe(data, gb, period)
* 计算标的夏普比率，若为分年度，返回DataFrame；若为整体，返回Float
* data：读取到的数据表格，类型为DataFrame
* gb：将data按照年份进行groupby
* period：'by year' or 'all period'，分别表示分年度和整体
* **carmar比率函数**  
  carmar(data, gb, period)
* 计算标的carmar比率，若为分年度，返回DataFrame；若为整体，返回Float
* data：读取到的数据表格，类型为DataFrame
* gb：将data按照年份进行groupby
* period：'by year' or 'all period'，分别表示分年度和整体
* **换手率函数**  
  turnR(weight, period)
* 计算资产组合的换手率，若为分年度，返回DataFrame；若为整体，返回Float
* weight：权重表格
* period：'by year' or 'all period'，分别表示分年度和整体
* **交易费用函数**  
  turnR(volume)
* 计算资产组合的交易费用，返回Float
* volume：成交金额
* **净值函数**  
  net\_worth(data, weight, workday)
* 计算资产组合的净值（扣除了交易费用），返回Series
* data：数据表格
* weight：权重表格
* workday：权重日期调整后对应的交易日日期Series
* **输出函数**  
  output(data, net, df\_year, df\_all)
* 输出资产组合分年度和整体表现的excel表格
* 输出资产组合中每个资产的净值走势图以及资产组合的走势图
* data：数据表格
* net：净值Series
* df\_year：回测后得到的资产组合分年度表现DataFrame
* df\_all：回测后得到的资产组合整体表现DataFrame
* **回测函数**  
  portfolio\_backtest()
* Portfolio.py主函数，分别计算分年度和整体表现，并输出excel表格
* 输出资产组合中每个资产的净值走势图以及资产组合的走势图

### 更新

（更新内容都添加到此处）

#### 版本：V2.0

**Portfolio.py**

* **交易日处理函数**
* 修正时间处理错误
* 增加函数处理可拓展性，例如：前几次调仓日早于策略开始日期，以及后几次调仓日晚于策略结束日期的情况，函数都能进行处理
* **净值函数**
* 添加组合内资产每日权重的输出
* **换手率函数**
* 修正整体换手率计算时错误

#### 版本：V3.0

**main.py**

* **添加相对路径**
* 将代码中绝对路径修改为相对路径，便于不同使用者使用

**Single\_asset.py**

* **年化收益率函数**
* 修正单一资产整体表现中的年化收益率计算

**Portfolio.py**

* **输出函数**
* 增加输出每日净值sheet
* 增加输出资产组合和标的净值对比走势图
* **净值函数**
* 修正净值计算公式
* 测试异常权重值

#### 版本：V5.0

**main.py**

* **剔除cash变量**

**Portfolio.py**

* **新增：截取数据处理函数**
* data\_process(data, weight)
* 根据权重sheet中的资产，来选取数据sheet中的资产
* 根据预设的策略开始和结束日期，来选取对应区间的数据
* **表格读取函数**
* 修改了读取方式，将日期列赋为索引
* 剔除整个框架中，涉及日期列的代码
* **净值函数**
* 重新修改了净值计算方式
* 剔除cash变量，直接用净值计算每次调仓的交易费用

**Single\_asset.py**

* **表格读取函数**
* 修改了读取方式，将日期列赋为索引
* 剔除整个框架中，涉及日期列的代码