回测框架

版本：V6.0

- Codes 文件夹：Single_Asset.py 和 Portfolio.py，分别是单资产和组合回测器的程序
- Data 文件夹：data.xlsx，回测所需价格和权重数据；以及 data-示例.xlsx，作为输入数据的格式示例
- 输出文件夹：输出的两个 Excel 文件以及 4 张净值图
- main.py：回测主程序
- 使用说明文件：Word 版本，PDF 版本

依赖包

- pandas, numpy, scipy, matplotlib

使用说明

- data.xlsx 文件有两个工作表，分别为【数据】和【权重】。前者用于存放各资产的收盘价或净值序列，后者用于存放建仓和调仓日各资产的权重（其他日期的可以直接不记录）。
- 注意事项包括但不限于（1）出现在【权重】工作表的资产必须同时出现在【数据】工作表中；（2）对两个工作表序列的日期，除从早到晚排列、没有重复以外无特殊要求，具体处理逻辑在下文代码函数定义部分详解。
- 虽然我们不可能穷举出所有可能使得程序出 bug 的情形，但只要使用者的输入符合常识、参照 data-示例.xlsx，就能尽量保证程序不出错。
- main.py 中可以更改年化时使用的期数、无风险利率、希望截取的起止日期、输入输出路径、高低风险资产名称列表及相应费率等参数，然后直接运行，就可以得到单一资产和资产组合回测结果的 Excel 表格。

代码函数/方法定义（为方便，下面函数和方法统称函数）

Single_Asset.py：使用示例请参考程序最后 if __name__ == '__main__'之后的部分

- **初始化函数**

  __init__(self, ann: int, rf=0.0, data=None)

  初始化一个单资产回测器对象。ann 为年化时使用的期数，与收盘价/净值数据频率对应，如日频取 250，周频取 52，月频取 12；rf 为年化无风险利率，单位不带百分号，i.e.如果无风险利率为年化 3%，输入 rf=0.03，默认 0；data 为价格序列的 DataFrame，格式对应 data.xlsx 中的【数据】工作表，可以不输入，设置这个参数是为了方便和其他程序对接，从而不用先导出到本地再读取。

- **读取价格信息函数**

load_sheet_from_file(self, input_path: str, sheet_name='数据')

读取 data.xlsx 文件中的收盘价/净值序列。input_path 为 data.xlsx 文件的路径；sheet_name 为对应的工作表名，默认数据。

- **数据日期切片函数**

  slice(self, start_date=None, end_date=None)

  截取希望保留下来的时间段，start_date 和 end_date 分别对应起止日期，若要输入，则格式为字符串，形式【YYYY-MM-DD】，默认 None。可以只输入一个或者均不输入，只输入 start_date 则会留下收盘价/净值序列中日期不早于 start_date 的，只输入 end_date 留下日期不晚于 end_date 的，均不输入则无影响。

- **对给定序列进行回测函数**

  backtest_series(self, nav_series: pd.Series, annualize: bool)

  对给定的序列进行回测，返回一个存储着回测结果的 DataFrame。nav_series 为需要进行回测的收盘价/净值序列，输入什么就测什么，annualize 为是否要。该函数无需用户显式调用，而是将具体回测过程打包，作为 backtest 函数中对全期和不同年份的回测的接口而已。

  计算了如下指标：区间收益率用最末收盘价除以最初收盘价减去 1、年化收益率等于区间收益率（若 annualize 为 False）或用初始化时给定的年化期数除以收益率序列长度作为年数计算区间收益率对应的年化复利（若 annualize 为 True）、年化波动率等于收益率序列样本标准差乘以初始化时给定的年化期数的平方根、最大回撤、夏普比率、卡玛比率、最大回撤开始日期、最大回撤形成日期。设置两种计算年化收益率的方法是因为在 backtest 函数中分年度回测时已经将 nav_series 切片好，此时不需要用复利方式来折算；同样，因为分年度的原因，最大回撤恢复日期可能需要用到切片好的 nav_series 之后的数据，因此不在这里计算最大回撤恢复日期。

- **回测函数**
  backtest(self, asset_name: str)

  单资产回测的核心程序，asset_name 为希望进行回测的资产名称。实际上就是将对应资产的完整序列以及分年度序列依次传入 backtest_series，再将各自结果拼接成一个大的 DataFrame。其中分年度时，每个年度都以上一年度最后一个价格作为当年的开盘价，除非数据中没有更早的年度。在 backtest_series 中无法计算的最大回撤恢复日期也在这里计算了。

  回测得到的 DataFrame 会作为值存储在 self.backtest_results 字典中，键为 asset_name。因此，若希望对多个资产进行回测，并不需要生成多个 Single_Asset 对象，只需将其统一放入 data.xlsx 是数据工作表中，多次运行 backtest。

- **最大回撤计算函数**
  mdd(self, nav_series: pd.Series)

给定一个收盘价/净值序列 nav_series，计算其对应的最大回撤、最大回撤开始日期、最大回撤形成日期并返回，逻辑比较直接。同样，无需用户显式调用，只是将具体回测过程打包，在 backtest 函数中被调用。

- **结果输出函数**

  output(self, output_path: str, asset_name_list: list)

  将 self.backtest_results 中存储的回测结果导出到本地的一个 Excel 中，Excel 文件路径为 output_path，需要输出的资产名称整理成列表形式的 asset_name_list。输出文件中，一个工作表对应一个资产，工作表名为字典的键，也即资产名。

Portfolio.py：使用示例请参考程序最后 if __name__ == '__main__'之后的部分

- **初始化函数**

  __init__(self, ann: int, rf=0.0, data=None, weight=None)

  初始化一个单资产回测器对象。ann 为年化时使用的期数，与收盘价/净值数据频率对应，如日频取 250，周频取 52，月频取 12；rf 为年化无风险利率，单位不带百分号，i.e.如果无风险利率为年化 3%，输入 rf=0.03，默认 0；data 和 weight 分别为价格序列和建仓调仓日权重序列的 DataFrame，格式分别对应 data.xlsx 中的【数据】和【权重】工作表，可以不输入，设置这两个参数是为了方便和其他程序对接，从而不用先导出到本地再读取。

- **读取价格和权重信息函数**

  load_sheets_from_file(self, input_path: str, data_sheet_name='数据', weight_sheet_name='权重')

  读取 data.xlsx 文件中的收盘价/净值序列和权重序列。input_path 为 data.xlsx 文件的路径；data_sheet_name 为收盘价/净值序列对应的工作表名，默认数据；weight_sheet_name 为权重序列对应的工作表名，默认权重。

- **输入交易费用函数**
  load_fee_rates(self, high_risk_name_list=None, high_risk_fee_rate=None, low_risk_name_list=None, low_risk_fee_rate=None)

  为组合中的各资产指定交易费用，分为高风险和低风险两类资产。high_risk_name_list 和 low_risk_name_list 分别为高风险和低风险各自包含的资产名称的列表，high_risk_fee_rate 和 low_risk_fee_rate 则为高风险和低风险资产适用的费率，单位不带万分号，i.e.如果费率为万三，输入 0.0003.函数中会检测是否有同时出现在高风险和低风险中的资产，以及是否 self.weight 即组合中所有资产都被指定了。

- **数据日期处理函数**

  slice(self, start_date=None, end_date=None)

截取希望保留下来的时间段，start_date 和 end_date 分别对应起止日期，若要输入，则格式为字符串，形式【YYYY-MM-DD】，默认 None。可以只输入一个或者均不输入，只输入 start_date 则会留下收盘价/净值序列中日期不早于 start_date 的，只输入 end_date 留下日期不晚于 end_date 的，均不输入则无影响。

此外，由于权重序列和收盘价/净值序列的日期不一定严格对应，函数除了切片还做了一些后续处理；但为了和 Single_Asset 保持统一，函数名仍然为 slice。具体来说，日期处理逻辑如下：

（1） 首先将 self.data 的收盘价/净值序列根据 start_date 和 end_date 切片，保留日期介于二者之间的部分。然后在根据处理完后的 self.data 的起止日期去截取 self.weight 的权重序列，因为所有早于或晚于收盘价/净值序列的权重信息实质上都是无意义的。

（2） 遍历（1）中处理完成后的 self.weight 索引的日期，依次处理，下称被遍历的日期为日期 A。如果日期 A 同样存在于 self.data 中，其对应的权重信息就可以直接被使用。如果日期 A 不存在于 self.data 中（其中一种可能的情形是日度价格序列只有收盘价而月度调仓使用了月末自然日），则寻找 self.data 中日期 A 之前的最后一个日期（下称日期 B）。若日期 B 在 self.weight 中已经指定了权重信息，则只能将 self.weight 中对应日期 A 的权重信息舍弃；但若日期 B 在 self.weight 中尚未指定权重，就将 self.weight 中对应日期 A 的权重信息修正为对应日期 B。这样，我们就可以考虑价格序列只有日度收盘价而调仓使用月末自然日的情况。

（3） 最终，对（1）和（2）中处理好的 self.data 和 self.weight，再将 self.data 中早于 self.weight 起始日的部分舍弃掉。这样，二者开始于同一个建仓日。

- **交易费用函数**
  calculate_fee(self, sb: pd.Series, sa: pd.Series, f: pd.Series, pa: pd.Series)

  计算某次调仓产生的交易费用，返回 float 类型的计算结果。sb 为调仓前各资产持有数量向量，sa 为调仓后持有数量向量，f 为各资产交易费率，pa 为调仓时各资产的价格向量。按定义，交易费用就是 sum(|sb-sa|*f*pa)。该函数无需用户显式调用，而是为后续生成组合净值曲线的 generate_nav 函数中具体计算过程提供一个接口。

- **生成组合净值函数**
  generate_nav(self)

  根据 slice 函数处理好的收盘价/净值序列和权重序列，计算组合考虑费用之后的净值曲线，以及各种详细参数，如持股数量、资产权重、换手率等。计算结果储存在 self.backtest_results 的字典中。详细计算过程可参考程序代码和注释，此处列出几个核心逻辑：
  （1）建仓日产生了交易费用，但将组合净值初始化为 1.
  （2）若某天不是调仓日，则持有股数不变，交易费用为 0，组合净值变化等于各资产持有数量乘以资产价格变化再求和。
  （3）若某天是调仓日，考虑到 self.data 是收盘价序列，还需要先将当天的资产

价格变化考虑进去，计算调仓前一瞬间的资产组合净值（也就是，假设这天不是调仓日时的净值），再按照收盘价调仓，调仓后净值等于调仓前减去交易费用。
（4）调仓计算需要解一个方程。假设调仓前后各资产持有数量向量分别为 sb、sa，调仓日收盘价向量为 pa，调仓目标权重为 wa，各资产交易费率向量为 f，调仓前后组合净值分别为 NAVb、NAVa，注意到 sa 和 NAVa 都是未知的。首先，按照个数=总价/价格的定义，sa=wa*NAVa/pa；其次，调仓前后净值减少幅度等于交易费用，因此 sum(|sb-sa|*f*pa)=NAVb-NAVa。将 sa 表达式代入后一个等式，就得到一个关于 NAVa 的一元方程，因此可以求解出 NAVa，再代回 sa 表达式得到 sa。

- **回测函数**
  backtest(self)

  实际上，就是在生成了组合净值曲线之后用 Single_Asset 对该净值曲线做单资产回测，并在其生成的回测结果表格（参见 Single_Asset.py 的 backtest 函数）技术上多加了记录换手率信息的列。对于某资产，只有在调仓日才产生换手率，当天换手率等于调仓前后资产占组合净值权重差的绝对值，一段时间的换手率则是将其内所有日期的换手率简单相加。各资产各调仓日换手率的计算是在 generate_nav 函数中完成的，但按资产分年度和全期汇总是在 backtest 函数中进行的。

- **结果输出函数**

  output(self, output_path: str)

  将 self.backtest_results 中存储的回测结果导出到本地的一个 Excel 中，Excel 文件路径为 output_path。

更新

（更新内容都添加到此处）