基于经济周期的大类资产配置策略构建

摘要

大类资产配置对于全球资管公司有着着重要意义，不仅如此，大类资产配置也能为投资者们提供能够有效控制风险的投资组合。本文主要研究利用宏观经济指标对经济状态进行划分，并根据宏观经济周期预测模型，建立了基于 Savitzky-Golay 滤波器、熵权法、Spearman 相关系数、改良时钟、ARIMA 时间序列分析、风险平价模型，通过 python 编程对各参数求解，最终依据变权重的风险平价模型提出相应大类资产配置策略。

针对问题一，我们采用经典美林时钟的思想，为了使模型更具有推广价值，我们基于原始数据，对经典美林时钟模型进行改良，提出金融周期-经济周期运行原理，有效解释了中国的周期运行规律，建立改良时钟模型。接着，用 Savitzky-Golay 滤波器平滑了新增债务曲线，对新增债务数据进行平滑化。最后将新增债务曲线和 GDP 平减指数作为经济-金融周期的同步指标，判定上行或下行趋势，将经济周期划分为四个不同周期，一般情况下经济周期便按照第一阶段到第四阶段的顺序进行。

针对问题二，我们使用了第一问所提供的指标建立了 ARIMA(p,q,d) 模型，对新增债务曲线和 GDP 平减指数进行预测。首先我们进行了模型的 ADF 检验，并计算差分的阶数，此后便根据赤池信息准则 AIC 的值，在最小的自由参数使用下寻找最佳拟合方程，以得出 ARIMA 中 p 和 q 的最优解。在绘制出预测曲线图后，我们使用第一问的划分机制，依据两曲线的走势图，对未来五年宏观经济环境进行了划分。

针对问题三，首先我们使用了熵权法，挑选了中证 1000 指数、标普高盛商品全收益指数、中债综合财富指数、货币基金代表四个大类资产指数。然后分阶段计算了所选大类资产指数的风险收益特征，包括了年化收益率、日标准差以及夏普率。最后，我们首先进行了正态性检验，发现数据不符合正态性，便使用了 Spearman 相关性分析，得到了债券与其他指标均呈强负相关性的特点。

针对问题四，首先基于问题二预测，我们挑选了中证 1000、标普商品、中债综合财富指数和货币基金代表四个大类资产指数。接着建立了经典风险平价模型和大类资产配置权重值的最优化模型，发现最终资产配置组合集中于波动小的资产，较大程度降低了投资收益。因此，我们根据风险平价模型原理，将风险平分到每个资产改良为按各资产收益率标准差成比例分配，求解最优化模型，发现略微提高对高风险资产的风险承受能力时，投资组合有相对高的收益：第 I 阶段年化收益率 11.2% 、日标准差 0.016、夏普率 0.74；第 II 阶段年化率 8.6% 、日标准差 0.007、夏普率 0.63；第 III 阶段年化率 3.1% 、日标准差 0.011、夏普率 0.48；第 IV 阶段年化率 6.3% 、日标准差 0.033、夏普率 0.66 。

综上所述，本文依据多种模型，全面分析了基于投资时钟、宏观经济周期预测模型及基于预测的配置策略模型如何进行大类资产配置，经结果分析检验，本文模型具有合理性和一定的现实意义。

关键字： 改良美林时钟　ARIMA 时间序列　Spearman 相关系数　风险平价模型

目录

一、 问题背景

当今世界正经历着百年未有之大变局，新冠肺炎的全球大流行更是在加剧这一变局的演进。经济全球化遭遇逆流，保护主义、单边主义盛行，国际贸易与投资也在大幅度萎缩，经济、科技、文化、政治等格局正在发生深刻的调整，世界正处在动荡的变革期。

也正是在这样的变革期间，人们对于大类资产的配置需求在逐步攀升，“把鸡蛋放在不同篮子”的大类资产配置模式，是我们在一定程度上抵御风险的重要手段。立足长远，结合经历的时代背景与所处的社会环境，我们更是可以通过一个成功的大类资产配置模式，来收获可观的投资收益；放眼全球，这一投资方式早已成为了资管业界的共识，其中一站式大类资产配置平台的搭建，也是全球资管公司在群雄逐鹿之中成功的关键。

然而，站在当下的角度去判断某一大类资产的未来表现是非常困难的，事实上也并没有哪一项资产可以一直作为最佳的投资选择，成为穿越不同时期的价值“长青树”。因此，我们需要结合当下的环境以及对未来宏观经济的判断，实现资产配置组合的动态调整。此外，由于我国政策制定框架及流动性具有特殊性，我们不能直接拿欧美国家已经成型完善的资产配置模型进行生搬硬套，而应该以此为鉴，找到一个更符合中国国情的大类资产配置方案与预测模型。

现有一系列相关的宏观经济指标数据，以及大类资产的指数行情数据，我们需要解决以下问题。

二、 问题重述

- **问题一**

  （1）寻找出两个以上高频有效的宏观经济指标。

  （2）将 2001 年-2021 年国内的宏观经济运行状况划分为不同的经济状态。

- **问题二**

  （1）使用经济或数学模型，模拟预测出中国未来五年的经济增长、通胀、利率等宏观经济环境。

  （2）根据问题一中规定的划分，判断中国未来五年所处的经济状态。

- **问题三**

  （1）挑选出四类资产（股票、大宗商品、债券、现金及其等价物）的四个代表指数。

  （2）根据问题一中规定的划分，预测以上大类资产指数在各种经济状态下的风险收益特征。

  （3）分析大类资产指数之间的相关性。

- **问题四**

  （1）预测出我国未来五年的经济状态。

  （2）挑选合适的大类资产指数以构建投资组合，并预测其风险收益特征。

三、 模型假设

1. 假设附件中的数据和我们在网上获取的数据是真实有效的。
2. 假设在数据预处理中，误差在合理的范围内对数据结果的影响可以忽略。
3. 假设未来 5 年内不会发生重大突发事件对中国经济造成严重影响。
4. 假设中国的经济运行是具有周期性的。
5. 假设大类资产指数交易时不考虑交易费用。

四、 符号说明

符号	含义
L_t	中国宏观杠杆率
D_t	累计债务率
G_D	GDP 平减指数
G_t	名义 GDP
G'_t	实际 GDP

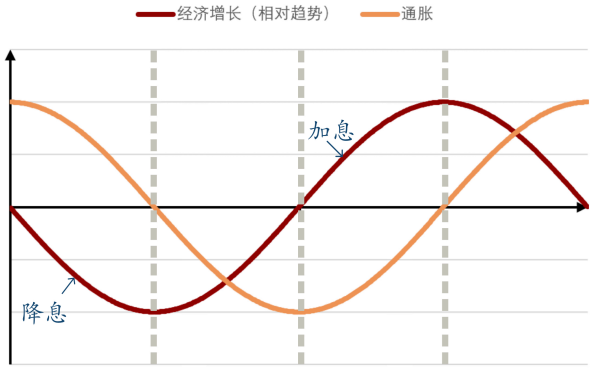
五、 问题一的分析与求解

5.1 问题一的分析

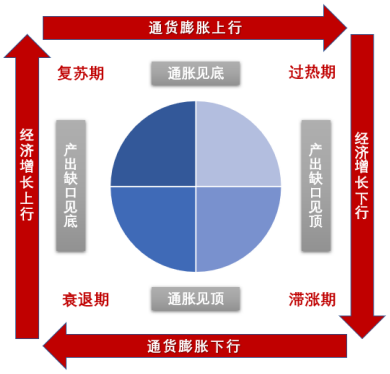
对问题一，我们采用经典的投资时钟思想寻找了新增债务规模和 GDP 平减指数作为高频有效的宏观经济指标，并主要根据这两个宏观经济指标将 2001 年-2021 年国内的宏观经济指标运行状况划分了不同的经济状态。本文中投资时钟是基于 2004 年由美林证券通过对超过 30 年的数据统计分析得出的美林时钟模型下进行改进，宏观经济具有周期性，美林时钟的基本假设为：经济增长的变动会明显领先于通胀的变动，而经济增长和通胀只有涨跌两个方向，因此用这两个指标可以将经济周期划分为四个阶段，即衰退期，其变现为经济增长和通胀都下行；复苏期，这个时期表现为经济增长上行和通胀下行；过热期，这个时期表现为经济增长和通胀都上行；滞涨期，这个时期表现为经济增长下行和通胀上行。

5.2 问题一的求解

5.2.1 美林时钟的验证

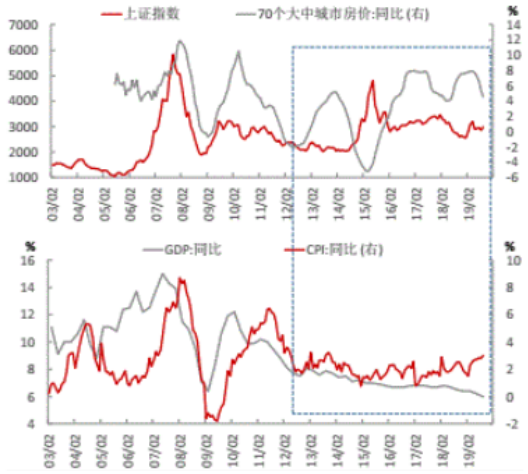


(a) 美林时钟

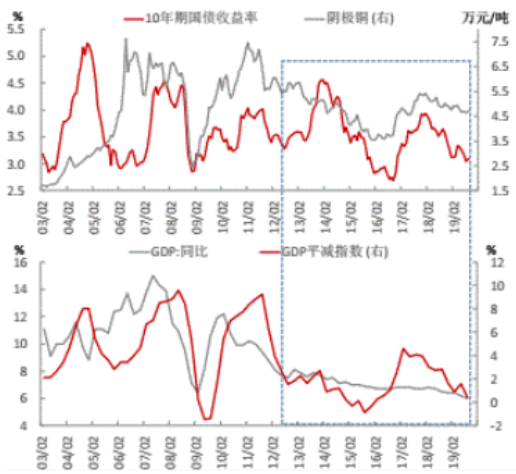


(b) 直观图

美林时钟完成于 2004 年，是对美国 1973 年至 2004 年经验的总结。但由相关研究表明，自 2008 年金融危机之后，美国等众发达国家的经济增长对于国家本身的通胀的影响越来越小，以产出缺口衡量的国家经济增长和以 CPI 衡量的通胀水平和 20 世纪 70 年代相比已经有了明显的差异。由相关资料表明，自 2013 年以后，以实际 GDP 增速来衡量的中国实际经济增长与通胀变动幅度都已经很小，，包括股票、债券、商品和房地产在内的资产价格波动幅度很大，这显然与美林时钟在初始假设下所得到的结论有较大差别。因此，在中国难以用以实际 GDP 增速来衡量的经济增长和以 CPI 衡量的通胀这两个指标来完全划分中国的经济周期，即美林时钟的前提假设在中国不成立。



(a) 经济增长与 **CPI** 波幅小，但资产价格波幅较大



(b) 经济增长波幅小，**GDP** 平减指数波动较大

5.2.2 经济指标的选取与改良投资时钟模型的构建

基于美林时钟的前提假设下，我们增加金融周期维度的影响和新的宏观指标：国内债务规模变化数，建立改良的美林时钟，进一步沿用投资时钟的思想将宏观经济运行状

况进行划分。在这里，我们对划分经济周期的划分方法作出调整，增加金融周期对经济周期划分的影响，既考虑经济周期也考虑金融周期。考虑国内债务规模变化数，其定义为中国宏观经济债务的增量，对于题给数据中国宏观杠杆率，中国宏观杠杆率是指累计债务量与上一年度名义 GDP 的比值，即

$$L_t = \frac{D_t}{G_t} \tag{1}$$

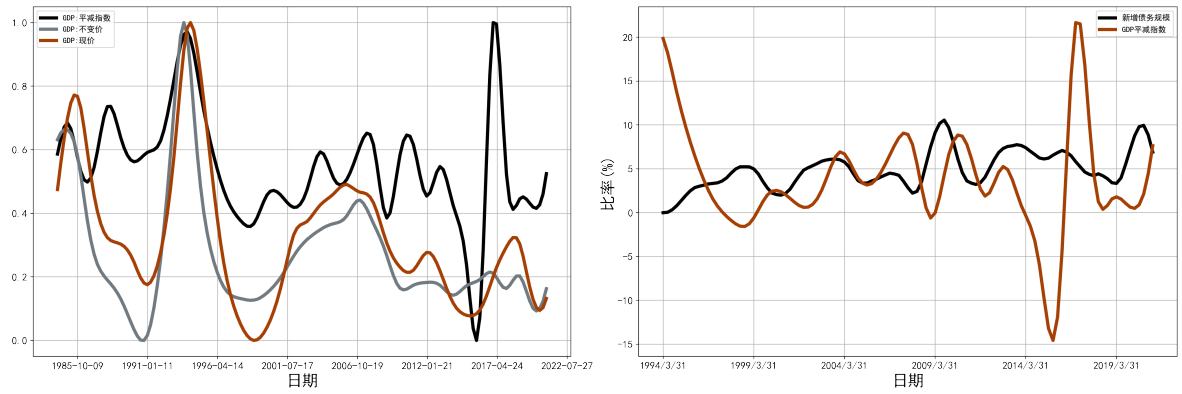
因此，国内债务规模的变化数即为债务的增量与名义 GDP 的比值，即

$$\Delta D_t = \frac{L_t G_t - L_{t-1} G_{t-1}}{G_t} \tag{2}$$

在这里国内债务规模变化数将作为连接金融周期和经济周期的核心变量，将国内债务规模变化数作为国内金融周期的的同步指标。另一方面，国内的名义 GDP 与 GDP 平减指数保持同步，因此我们用 GDP 平减指数作为经济周期的同步指标，其定义为名义 GDP 与实际 GDP 之商。

$$G_D = \frac{G_t}{G'_t} \tag{3}$$

为了计算 GDP 平减指数，我们补充了实际 GDP 数据，由此可以得到 GDP 平减指数。



(a) 名义 **GDP** 与 **GDP** 平减指数同步　　**(b)** 国内债务规模变化树领先 **GDP** 平减指数半年左右

我们将 GDP 平减指数和国内债务规模变化数放在一起看，可以发现国内债务规模变化数领先 GDP 平减指数半年左右，用国内债务规模变化数与 GDP 平减指数划分出四个阶段，国内债务规模变化数上行通胀下行的阶段为第一阶段，国内债务规模变化数上行通胀上行的阶段为第二阶段，国内债务规模变化数下行通胀上行的阶段为第三阶段，国内债务规模变化数下行通胀下行阶段为第四阶段。一般情况下，周期按照从第一阶段到第四阶段的顺序运行。

5.2.3 宏观经济运行状况的划分

由于新增债务规模变化指数的波动性过强，我们没有办法对其进行趋势的拟合及经济状况的划分，因此我们首先对新增债务指数进行 Savitzky-Golay 滤波处理，极大地消除了滤波的干扰项。如下便是我们基于改良投资时钟制订的经济划分依据。

表 **1**　经济周期划分规则

	新增债务规模	通胀
第一阶段	上行 ↑	下行 ↓
第二阶段	上行 ↑	上行 ↑
第三阶段	下行 ↓	上行 ↑
第四阶段	下行 ↓	下行 ↓

此后，我们便可以依据两指标斜率的变化，使用曲线波段极值分析法，对四个经济周期进行划分，以下为我们的波段划分图。

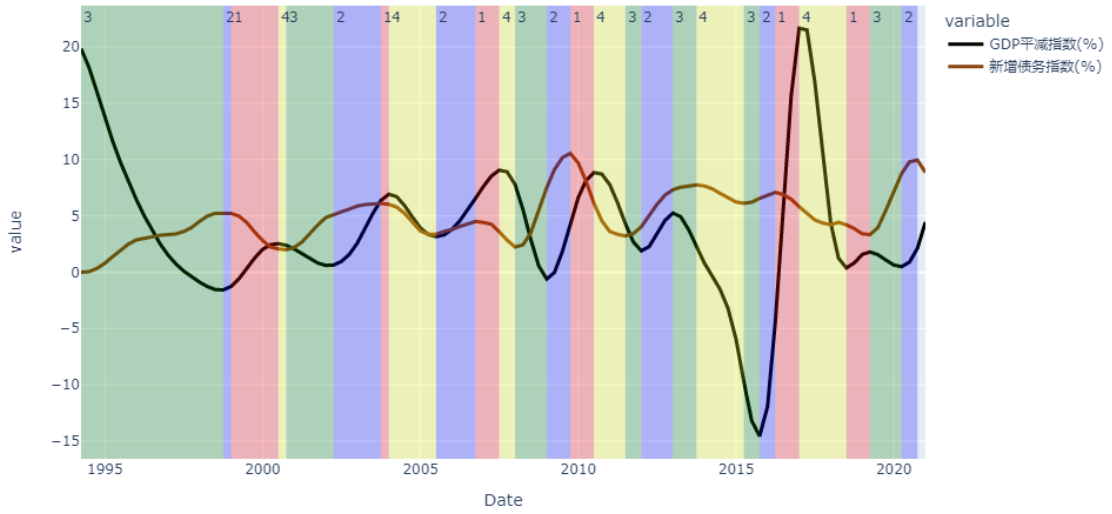


图 **4**　通过国内债务规模变化数和 **GDP** 平减指数把周期划分四个阶段

对于不同的经济周期，我们使用了不同的颜色，并同时在每一个分格的左上角进行了周期数据的标注。在大部分情况下，我们的时钟均是按着顺序进行轮动的，若遇到了突发事件或是宏观经济风格的快速切换，便会出现部分周期过短或是被跳过的现象。

六、 问题二的分析与求解

6.1 问题二的分析

对于第二问，我们首先需要对中国未来五年的宏观经济环境进行预测。沿袭第一问的观点，我们选择了 GDP 平减指数以及新增债务规模指数作为我们判断宏观经济情况

的指标。由于上述指标是以季度为单位所组成的一组时间序列数据，结合宏观经济模型自身的周期性，以及历史白噪声的移动平均现象，符合 ARIMA 模型的显著特征，因此我们以 ARIMA 模型来对宏观经济指标进行时间序列分析。

在进行 ARIMA 模型预测之前，我们需要对其进行 ADF 检验，以验证其数据本身的平稳性。此外, 对于 ARIMA 中的 (p,d,q) 参数，我们通过赤池信息量准则，选取 AIC 值最小且所添加自由参数最少的公式进行拟合。同时，我们采用了滤波后的数据集来作为 ARIMA 模型参照的历史数据，在一定程度上减弱了诸如疫情、经济危机等黑天鹅事件对模型训练的结果影响，从而建立改良后的 ARIMA 模型，来预测未来五年上述指标的走向变化。根据我们所预测的指标结果，我们便可以将其在第一问中的划分结果中进行归类。

6.2 问题二的求解

6.2.1 ADF 检验

由于 ARIMA 模型要求序列具有较强的平稳性，而对于不平稳的数据，我们无法通过时间序列模型来捕捉其规律。因此我们需要对数据进行 ADF 检验，即通过分析 t 值的方法来判断序列是否可以显著地拒绝不平稳序列的假设，即 P 值应满足 $P < 0.05$ 的条件。ADF 检验的方法步骤如下所示：

1. $AR(p)$ 模型的改写。

$$\rho = \sum_{i=1}^{p} \phi_i \tag{4}$$

$$y_t = \sum_{i=1}^{p} \phi_i y_{t-i} + \varepsilon_t = \rho y_{t-1} + \sum_{i=1}^{p-1} \zeta_i \Delta y_{t-i} + \varepsilon t = x_t' R + \varepsilon t \tag{5}$$

其中检验统计量的值 t 的计算公式如下所示，我们可以通过 t 值的结果并结合 ADF 检验的对照表，判断出我们模型的平稳性。

$$t = \frac{\hat{\rho} - 1}{\hat{\sigma}_\rho} \tag{6}$$

2. 样本数据集的构建

对于 $y_t t = x_t' R + \varepsilon t$, 我们有 $R = (\zeta_1, \ldots, \zeta_{p-1}, \rho)'$, 对于样本数据集 $Y = (y_{p+1}, \ldots, y_T)'$

我们构建了矩阵 $X = \begin{bmatrix} \Delta y_2 & \Delta y_3 & \cdots & \Delta y_p & y_p \\ \Delta y_3 & \Delta y_4 & \cdots & \Delta y_{p+1} & y_{p+1} \\ \vdots & \vdots & & \vdots & \vdots \\ \Delta y_{T-p+1} & \Delta y_{T-p+2} & \cdots & \Delta y_{T-1} & y_{T-1} \end{bmatrix}$

由此我们可以得到：$\hat{R} = (X'X)^{-1} X'Y$，且 $\hat{Y} = X\hat{R}$

3. ADF 的三种情况分类讨论

ADF 分为三种情况，分别是**普通的 AR(p1) 过程、带漂移项的 AR(p2) 过程以及趋势平稳的 AR(p3) 过程**。由于他们三者分别对应了不同类型的 p 阶自回归过程，因此在计算单位根 t 值时他们所对应的 $\hat{\sigma}_\rho$ 是不相同的，而 $\hat{\sigma}_\rho$ 的差异性则来源于其中 s_t 的计算方法，如下是他们三种自回归过程计算中 s_t 的值以及 $\hat{\sigma}_\rho$ 的公式。

$$\hat{\sigma}_\rho = \left(s_t e_i' \left(X'X\right) e_i\right)^{\frac{1}{2}} \tag{7}$$

$$\begin{cases} s_t = \dfrac{1}{T-2p}(Y-\hat{Y})'(Y-\hat{Y}) \quad , AR(p1) \\ s_t = \dfrac{1}{T-2p-1}(Y-\hat{Y})'(Y-\hat{Y}) \quad , AR(p2) \\ s_t = \dfrac{1}{T-2p-2}(Y-\hat{Y})'(Y-\hat{Y}) \quad , AR(p3) \end{cases} \tag{8}$$

4. 计算差分此后我们便计算差分的值，对于我们选择的数据 f(x)，我们根据如下公式计算出了差分的结果。

$$\begin{aligned} x_k &= x_0 + kh, (k = 0, 1, \ldots, n) \\ \Delta f\left(x_k\right) &= f\left(x_{k+1}\right) - f\left(x_k\right) \end{aligned} \tag{9}$$

差分是一种线性算子，在 ADF 检验不满足情况的时候，我们可以使用差分的方式来降低函数的阶数，从而保证我们的时间序列预测效果。在对这两个指标进行预测的时候，我们总共使用了 2 阶差分的数据进行计算，得到了如下的 ADF 检验表结果。

5. 结果呈现

根据上述公式，我们得到了如下的 ADF 检验结果：

表 **2**　新增债务规模和 **GDP** 平减指数的 **ADF** 检验结果

变量	差分阶数	t	P(保留三位小数)
新增债务规模	0	-2.386	0.146
	1	-5.312	0
	2	-4.745	0
GDP 平减指数: 同比 (%)	0	-3.403	0.011
	1	-3.104	0.026
	2	-5.328	0

根据对 t 值的分析以及 P 值得比较，我们得到所给数据可以显著性地拒绝序列不平稳的假设 (P<0.05)。在满足通过显著性检验得基础上，我们尽可能地选择较小得阶数，因此新增债务规模选择 1 阶差分，GDP 平减指数则是 0 阶。

6.2.2 ARIMA 模型的建立

在 ARIMA(p,d,q) 中，AR 是" 自回归"，p 为自回归项数，而 MA 为" 滑动平均"，q 为滑动平均项数，d 为差分阶数。在上一步 ADF 检验当中，我们已经完成了对于模型的 d 值计算，下面我们将结合移动平均以及自回归原理对模型进行构建。

表 3　ARIMA 模型的参数组成

$ARIMA(p,d,q)$	
$AR(p)$	p 阶自回归
$MA(q)$	q 阶滑动平均
d	使之成为平稳序列的差分阶数

1. 计算 AIC 的值

   AIC 是赤池信息量准则的简称，可以评估统计模型的复杂度以及衡量统计模型拟合的优良性。AIC 在一般情况下可以表示为

$$AIC = 2k - 2\ln(L) \tag{10}$$

   其中 k 为参数数量，L 为似然函数，此外，我们通过增加自由参数的数目，在避免模型出现过度拟合的情况之下提高了模型拟合的优良性。因此我们应优先考虑 AIC 值最小的函数，并尽可能地减少自由参数的使用。

   由此我们所计算出的 AIC 值的结果如下所示：

表 4　AIC 值的计算结果

变量	AIC	变量	AIC
新增债务规模	-59.782	GDP 平减指数: 同比 (%)	-42.518
	-55.42		-33.277
	-40.454		-33.708

2. 基于 AIC 的值计算 ARIMA 的 p、q 值

根据系统基于 AIC 信息准则所自动寻找的最优参数，我们得到了以下最终的模型选择结果结果：

(1) 在新增债务规模的预测中使用 ARIMA(3,1,5) 模型

(2) 在 GDP 平减指数的预测中使用 ARIMA(4,0,2) 模型

3. 绘制预测指数结果

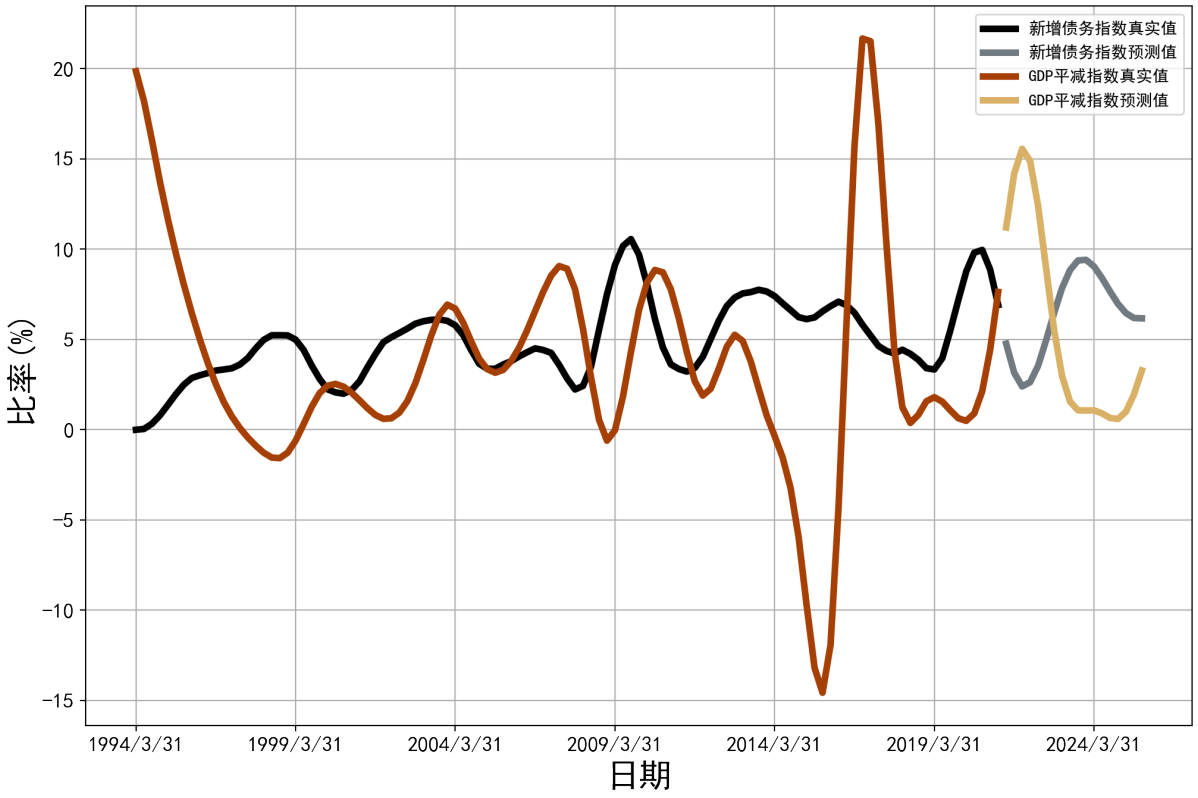


图 **5** **ARIMA** 时间序列预测结果图

6.2.3 经济状态的划分

根据上图，我们预测 GDP 平减指数在经过一段时间的上升后，又回到了之前的稳定波动阶段，通货膨胀的状态得到了良好的控制；新增债务指数没有发生明显的变化，并保持在之前的波动范围，中央政府在后疫情时代的宏观调控政策卓有成效。

根据第一问中我们对经济状态的划分原则，由于我们所使用的预测数据是经过滤波后的数据，我们无需再在划分周期时对所得结果进行滤波处理。对预测值进行求导，我们发现在 2021 年第二季度到 2024 年伊始，以及 2025 年的 1 到 3 季度，均呈现了新增债务指数与 GDP 平减指数相背离的情况。其中 22 年到 24 年的绿色部分为第三阶段，应以货币类资产的投资为主，其余红色部分则为第一阶段。

由于 GDP 平减指数和新增债务指数均在 2022 年实现了转折，因此我们在 2022 年跳过了第二阶段而直接来到了第三阶段。其余部分则是按照改良投资时钟的轮动规律，分别被划分至了黄色的第四阶段以及蓝色的第二阶段。

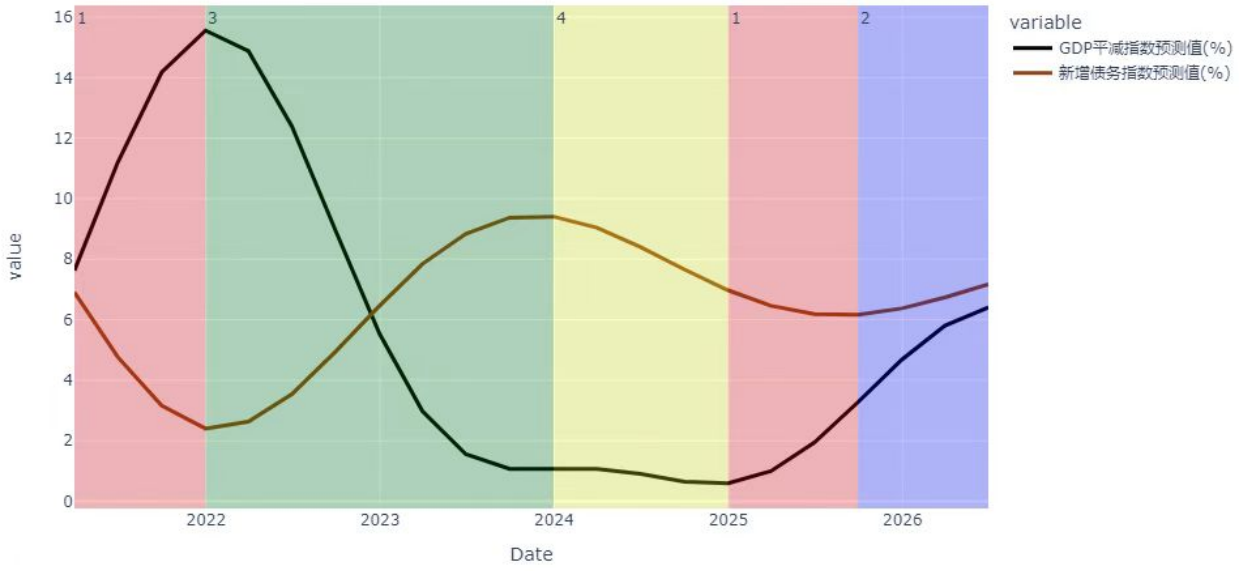


图 **6**　预测结果的宏观经济环境划分

七、 问题三分析与求解

7.1 问题三的分析

在问题三中，我们首先需要挑选出这四大类资产的代表指数。从数据本身出发，我们考虑哪组数据所含有的信息量更大，因此我们引入信息熵的概念，并使用熵权法来进行信息熵的计算。通过信息熵，我们可以得到指标的权重，并挑选信息权重最大的指数来做为代表指数。

其次，在风险收益特征方面，由于夏普率可以同时平衡期望收益和收益率标准差，并判断出在多承担风险的条件下所可以获得的超额收益，因此我们将夏普率的值作为衡量不同指数优劣的指标，将期望收益作为衡量资产收益的因素，将收益率标准差作为资产指数的风险特征。我们首先以交易日为单位，计算出以上三个风险收益特征的季度年化指标，并根据第一问中我们所划分的时钟经济状态，以时间跨度为权重，计算并分析出不同指数在不同经济状态下的风险收益特征。

最后，我们基于斯皮尔曼相关系数，对不同的大类资产指数进行一一比较，从而判断不同指数之间的相关性。

7.2 问题三的求解

7.2.1 基于熵权法的大类资产指数筛选

对于问题三所给的指数，我们发现在股票部分一共有四个指标，大宗商品部分一共有两个指标，债券部分有三个指标，而货币部分只有一个指标。因此我们首先可以直接在货币资产部分选择题目所提供的货币基金这一指标，而在股票、大宗商品以及债券这三类资产部分，我们引入信息熵的概念，使用如下步骤进行信息熵值和权重的计算：

1. 归一化矩阵的构建与数据的归一化

   首先我们使用 $min-max$ 标准化的方法，构建归一化矩阵并使得指标数据的值落入 $[0,1]$ 的区间中。由于我们所选择的指标均为正向指标，故无需考虑正向化的问题。首先我们根据数据对不同类的资产分开计算，以不同的指数为列，年份为行，构建一个 $i \times j$ 的矩阵 X，并在计算后使用矩阵 Y 对矩阵 X 进行替换。熵权法的计算公式如下所示，其中 y_{ij} 代表了数据标准化后得值，x_{ij} 代表数据标准化前的值，而 $min(X)$ 和 $max(X)$ 则分别代表数据整体的最大值和最小值。

$$y_{ij} = \frac{x_{ij} - \min(X)}{\max(X) - \min(X)} \tag{11}$$

2. 第 j 项指标下第 i 个样本值的比重 ρ_{ij} 的计算:

$$\rho_{ij} = \frac{x_{ij}}{\sum_{i=1}^{n} x_{ij}} \tag{12}$$

3. 第 j 列指标的信息熵值 e 的计算：

$$k = \frac{1}{\ln(n)} \left(0 \leq e_j < 1\right) \tag{13}$$

$$e_j = -k \sum_{i=1}^{n} \rho_{ij} \times \ln\left(\rho_{ij}\right) \tag{14}$$

4. 计算列指标的权重

   信息效用值 d_j 和指标权重 w_j 的计算：

$$d_j = 1 - e_j \tag{15}$$

$$w_j = \frac{d_j}{\sum_{j=1}^{m} d_j} \tag{16}$$

5. 进行信息熵值得比较

   在对每一个大类资产进行分别分析后，我们得到了如下各类代表指数的信息熵值。通过比较他们 e 的大小，我们便可以得到我们所选的指标。

通过权重的大小比较，我们最终选出**中证 1000 指数、标普高盛商品全收益指数、中债-综合财富 (7-10 年) 指数以及货币基金指数**作为代表。

表 **5**　股票数据的熵权法结果

	上证 50	沪深 300	中证 1000	中证 500
信息熵值 e	0.982	0.983	0.979	0.983
信息效用值 d	0.018	0.017	0.021	0.017
权重 (%)	24.125	22.798	29.281	23.795

表 **6**　大宗商品和债券数据的熵权法结果

	南华指数	标普高盛指数	中债 (1 年)	中债 (3-5 年)	中债 (7-10 年)
信息熵值 e	0.982	0.978	0.967	0.967	0.966
信息效用值 d	0.018	0.022	0.033	0.033	0.034
权重 (%)	44.445	55.555	32.943	33.334	33.723

7.2.2 大类资产指数的风险收益特征

在对大类资产指数进行风险收益特征分析时，我们使用了**期望收益、收益率标准差以及夏普率**这三个分别代表了收益、风险以及风险收益比的指标，并结合第一问划分的时钟周期来进行计算。

1. **期望收益的计算**

对于某一个阶段，我们假设其在投资时钟里被拆分成了 P 个格。我们设某一格为 Y_i，且其共有 n_i 个季度，他的起始指数价格为 Y_s，结束价格为 Y_e，由此我们可以得到每一格的年化率 k_i 以及某一阶段总共的年化率 k_P 的公式为：

$$k_i = \frac{Y_e - Y_s}{Y_s} \times \frac{4}{n} \times 100\% \tag{17}$$

$$k_P = \frac{\sum_{i=1}^{P} k_i \cdot n_i}{\sum_{i=1} n_i} \tag{18}$$

由此我们便可以得到预期年化收益率，并根据其预期收益绘制基于整体值以及某一指数内部的列指标热力图：

阶段	中证1000	标普指数	债券(7-10)	货币基金
1	50.79	24.89	4.8	5.619
2	43.45	-0.3	-5.63	2.459
3	-20.46	-12.38	3.13	9.155
4	30.25	11.36	-0.135	3.939

(a) 整体值热力图

阶段	中证1000	标普指数	债券(7-10)	货币基金
1	50.79	24.89	4.8	5.619
2	43.45	-0.3	-5.63	2.459
3	-20.46	-12.38	3.13	9.155
4	30.25	11.36	-0.135	3.939

(b) 大类列指标热力图

图 **7**　年化预期收益率 **(%)** 热力图

根据上图，我们得出股票的整体波动性远大于其他三个指标，其中在阶段 1、2、4 可以获得较为可观的收益，而大宗商品的标普指数则是在阶段 1 中有最优的表现。对于债券部分而言，除了阶段二会产生一定的回撤，其他阶段均表现较为良好。货币基金的表现十分亮眼，不论在什么阶段中，均能取得正的收益，且仅在第 2 阶段会出现跑输无风险收益的情况。

2. **收益率标准差的计算**

在计算标准差之前，我们首先进行数据处理，并用后一天的指数减去前一天的指数，从而计算出每一天的增量 X_i。对于不同的阶段，我们计算了其阶段平均日期望收益率 $\bar{E}$，并根据如下公式计算了数据的标准差：

$$\sigma = \sqrt{\sum_{i=1}^{n} \left(X_i - \bar{E}\right)^2 \cdot p_i} \tag{19}$$

标准差在一定程度上可以反映出数据之间的波动性，在追求稳定收益的基础上，标准差便成为了我们重要的风险判断依据。标准差的具体计算结果如下表所示：

表 **7**　收益率标准差的计算结果

指标	第一阶段	第二阶段	第三阶段	第四阶段
中证 1000 指数	0.048059	0.047684	0.062372	0.120407
标普高盛商品全收益指数	0.02764	0.02855	0.025663	0.018434
中债-综合财富 (7-10 年) 指数	0.018459	0.009049	0.006944	0.006962
货币基金	0.013822	0.005526	0.005196	0.004358

在风险评估的层面，货币基金与债券具有非常明显的优势。不论在什么样的时期，我们都可以发现四个指标的标准差排序均是：

股票 > 商品 > 债券 > 货币

因此，不论是对于投资者还是投资机构而言，若是不想承受短时过大的回撤，配置一部分资金进入债券以及货币部分是一个非常明智的选择。

3. **夏普率的计算**

夏普率这一指标建立在了“理性的投资者”这一理念上，即在给定风险时追求收益最大化，或是在收益固定时追求最小的风险。因此，当我们在建立有风险的投资组合时，我们便可以通过夏普率来判断该投资组合能否达到无风险投资的回报。夏普率的计算公式如下，其中 $E(R_p)$ 代表投资组合的年化报酬率，而 R_f 代表年化无风险利率，σ_p 代表投资组合年化报酬率的标准差。

$$SharpeRatio = \frac{E(R_p) - R_f}{\sigma_p} \tag{20}$$

夏普率的值与时间跨度以及产品类型有关，在一般情况下主要用于同类产品的比较，由于其为一个相对值，故其数值大小本身是没有意义的。然而，只要夏普比率为正，便意味着组合收益率高于波动风险，其中夏普比率超过 1 的基金是极为罕见的。

根据我们所计算出的各阶段夏普率值，我们绘制了如下的夏普率折线图。

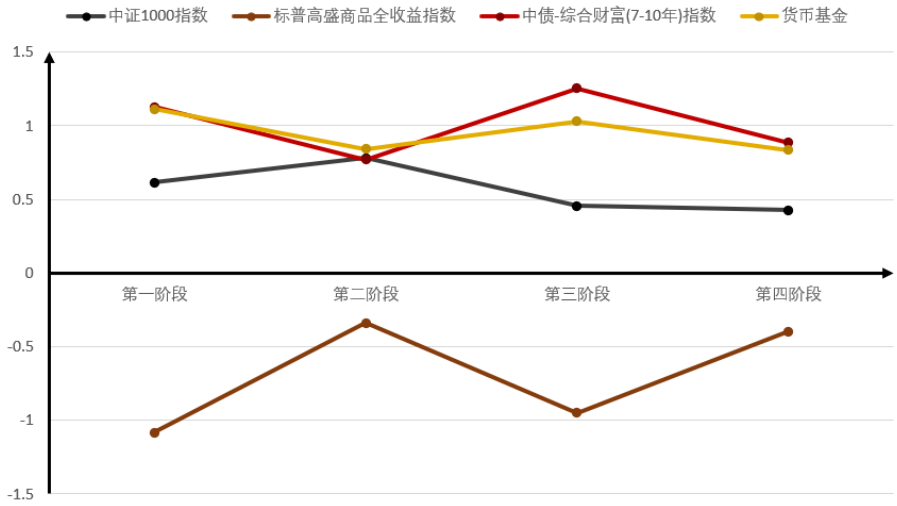


图 **8**　夏普率结果折线图

7.2.3 相关性分析

在对数据进行分析之前，我们首先对数据进行了标准化，绘制了标准化后各项数据之间的关联曲线。

图 **9**　标准化后的大类资产代表指数

为分析指标的相关性，我们首先需要选取一个相关系数进行拟合。由于 Pearson 相关性分析主要用于分析满足正态分布的两个定量变量的关系，故不适合用来对指数进行分析，因此我们使用了 Spearman 相关性分析。

1. 进行 Shapiro-Will 正态性检验

   首先我们对以上数据进行分别排序，绘制出这四个指数的 Q-Q 图，我们所得到的 W 统计量 (P 值) 如下

表 **8**　**Shapiro-Wilk** 检验结果表

	货币指数	中债 7-10	标普商品	中证 1000
S-W 检验	0.924(0.000)	0.941(0.000)	0.934(0.000)	0.978(0.000)

   由于以上四个数据的显著性 P 值均在保留三位小数后变为 0.000，因此水平呈现显著性，拒绝原假设，数据不满足正态分布。因此，我们应该使用 Spearman 相关系数来进行相关性检验。

2. 相关性检验

   (1)X 和 Y 是两组数据，我们首先计算 X_i 与 Y_i 之间的等价差 d_i，并将该数据所在的一列按照从小到大排序后，这个数所在的位置，再计算总数据的个数 n。

(2) 根据如下公式计算出斯皮尔曼相关系数

$$r_s = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n\left(n^2-1\right)} \tag{21}$$

(3) 通过 r_s 的值，判断其之间的相关性。

由于斯皮尔曼相关性分析的分析对象是在两个指标之间的，因此我们需要对四个指数进行两两分析，根据 Spearman 相关性分析的公式，我们可以的到如下的分析结果热力图。

指标	中证指数	标普商品	中债(7-10)	货币指数
中证指数	1	0.987	-0.852	0.787
标普商品	0.987	1	-0.836	0.803
中债(7-10)	-0.852	-0.836	1	-0.785
货币指数	0.787	0.803	-0.785	1

图 **10**　相关性检验结果的热力图

根据热力图我们可以发现，各种商品之间有着非常强的正相关或者负相关性，其中最为特殊的指标是债券，与其他三个指标均有较强的负相关性。因此，债券可以在进行组合配置时对冲一定的风险。

八、 问题四的分析与求解

8.1 问题四的分析

对问题四，我们首先基于问题二预测，挑选了中证 1000、标普商品、中债综合财富指数和货币基金代表四个大类资产指数。考虑到本题目的核心是求解投资组合的配比，因此我们根据风险平价模型原理，将风险平分到每个资产改良为按各资产收益率标准差成比例分配，求解最优化模型，并计算在各个阶段中，组合应有的配置比例。此后，我们在考虑提高风险资产的比例，使用 TRC 变权重的方式对原有算法进行改良。

根据最终的改良的算法，我们可以得到他的投资组合比例，我们据此计算了他在不同阶段的各类风险收益特征，并根据未来五年实际的时间序列预测情况，预测出未来五年的风险特征指标。

8.2 问题四的建立

8.2.1 风险平价模型的建立

问题四要求确立合适的大类资产投资组合，基于模型预测出的国内五年的经济状态，我们首先建立经典的风险平价模型。假设当前由 n 个资产 $(a_1, a_2, \cdots, a_n)$, 其收益率为 $(r_1, r_2, \cdots, r_n)$, 资产组合的权重为 $(x_1, x_2, \cdots, x_n)$ 。由此可以计算出投资组合的收益为：

$$R = x_1 \times r_1 + x_2 \times r_2 + \cdots + x_n \times r_n = \sum_{i=1}^{n} x_i \times r_i \tag{22}$$

为了投资组合的稳定程度，引入风险的概念：风险来自于大类资产价格的波动，更精确的表达为来自于收益率的波动，因此可以用收益率的标准差 Σ 表示风险。收益率，即大类资产在某段时间范围内的相对变化程度，公式如下：

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}} \tag{23}$$

其中 P_{t-1} 为 $t-1$ 时刻的股票价格，P_t 为 t 时刻的股票价格，R_t 为 t 时刻相对 $t-1$ 时刻的股票收益率。那么单一大类资产的风险可以由收益率的波动表示如下：

$$\sigma = \text{Std}(R) \tag{24}$$

而标准差由前公式可以算得，那么将此风险推广到组合，即组合风险的定义应由方差与协方差给出：

$$\sigma(x) = \sqrt{X^\top \Sigma X} = \sqrt{\sum_{i=1}^{n} x_i^2 \times \sigma_i^2 + \sum_{i=1}^{n} \sum_{j \neq i} x_j \times \sigma_{ij}} \tag{25}$$

其中 X 为资产组合权重列向量，Σ 为各资产收益率的协方差矩阵，σ_{ij} 为资产 i 与资产 j 收益率的协方差。可见多个资产配置到一起的总风险不仅考虑了单个资产自身的波动，还考虑到了多个资产之间的相关性。因此，我们的核心问题即转化到了如何确定资产的分配权重 $x_i\,(i = 1, 2, \cdots, n)$ 使得投资组合的收益尽可能高的同时，控制风险尽可能地小。为了建立确定分配权重的风险模型，我们首先需要确定每个资产对于投资组合的风险贡献，资产 i 的边缘风险贡献可由包含 n 个资产的投资组合标准差对各个资产的偏导数给出。为此，将包含 n 个资产的投资组合的标准差改写为

$$\sigma_p = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \sigma_{ij} \tag{26}$$

其中，σ_p 是投资组合的预期标准差，w_i 是资产 i 的权重，σ_{ij}^2 是资产 i 和资产 j 的协方差，σ_{ij} 是资产 i 的标准差，n 是组合中资产的数量。故由上讨论可知资产 i 的边缘风险贡献（MRC）为

$$MRC_i = \frac{\partial \sigma_p}{\partial w_i} = \sum_{i=1}^{n} w_i \sigma_{ij} = \sigma_{ip} \tag{27}$$

该变量描述了单个资产权重的微小变化给投资组合风险所带来的影响，这里的组合风险是以资产收益率的标准差度量的。从而我们可以定义资产 i 的总风险贡献为该资产权重与边际风险贡献的乘积，即

$$TRC_i = w_i \times MRC_i \tag{28}$$

进而，投资组合风险可分解为各资产总体风险贡献的求和，表示如下：

$$\sigma(x) = \sum_{i=1}^{n} TRC_i = \sum_{i=1}^{n} w_i \frac{\partial \sigma_p}{\partial w_i} \tag{29}$$

为了保证数据的可靠性，对各个资产的风险做归一化处理如下:

$$\frac{TRC_i}{\sigma(X)} = \frac{w_i \dfrac{\partial \sigma_p}{\partial w_i}}{\sigma(x)} \tag{30}$$

由此引出风险平价模型的基本定义：将大类投资组合的总风险平均分摊到每个资产上，即将上述关于每种资产 TRC 相等作为约束条件，问题即转化成了以下最优化问题：目标函数

$$x^* = \text{argmin} \left(\sum_{i=1}^{n} \sum_{i=1}^{n} (\text{TRC}_i - TRC_j)^2 \right) \tag{31}$$

其中 $b_i \, (i = 1, 2, \cdots, n)$ 是由问题三中各资产对应年化率的比例。约束条件

$$\sum_{i=1}^{n} x_i = 1, \quad 0 \leqslant X \leqslant 1 \tag{32}$$

但经典的风险平价模型得出的资产配置组合会偏向于集中波动小的资产，以达到平衡风险的条件，这样就难免大大降低了投资组合的整体收益。

表 **9**　风险平价下各阶段指数组合分配权重

指标	第一阶段	第二阶段	第三阶段	第四阶段
中证 1000 指数	7.80%	5.40%	5.00%	1.60%
标普高盛商品全收益指数	38.60%	18.90%	16.10%	25.00%
中债-综合财富 (7-10 年) 指数	22.90%	31.80%	34.90%	29.30%
货币基金	30.70%	43.80%	44.00%	44.10%

可以发现，经典的风险平价模型得出的资产配置组合会偏向于集中波动小的资产，以达到平衡风险的条件，但这样就难免大大降低了投资组合的整体收益。

8.2.2 建立变权重的风险平价模型

经典的风险平价模型正是在约束每种资产 TRC 相等时实现了风险的良好控制，但难免降低了整体收益，我们在任然保持风险的良好控制时，对 TCR 的权重进行适当的改变以提高投资组合的整体收益。在问题一中，我们已经对不同的经济状态进行了划分，得到了四个阶段的各大类资产的价值表现，因此不妨根据该价值表现来设定不同的每种资产风险对总风险的贡献率。

经典的风险平价模型正是在约束每种资产 TRC 相等时实现了风险的良好控制，但难免降低了整体收益，我们在任然保持风险的良好控制时，对 TCR 的权重进行适当的改变以提高投资组合的整体收益。在问题三中，我们已经计算了收益率标准差，得到了四个阶段的各大类资产指数的收益率标准差，因此不妨根据该标准差来设定相应的每种资产风险对总风险的贡献率。

故修正经典风险平价模型，对应以下最优化问题：

$$\text{argmin}\sum_{i=1}^{n}\left(TRC_i - C_i^k \times \sigma_p\right)^2, k = 1,2,3,4 \tag{33}$$

$$\sum_{i=1}^{n} w_i = 1, \sum_{i=1}^{n} C_i^k = 1 \tag{34}$$

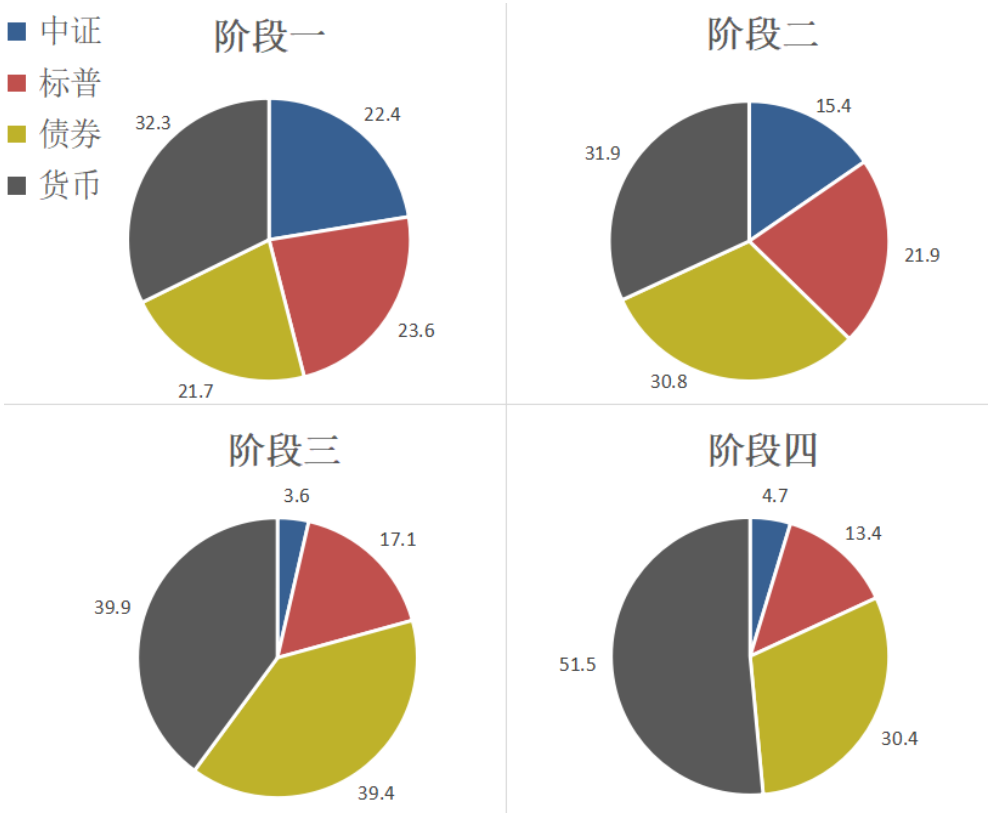


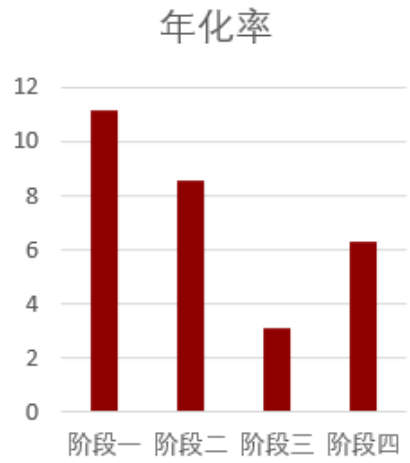
图 **11**　变权重的风险平价下各阶段指数组合分配结果

此后，我们便可以根据此优化模型，对我们的投资组合进行修正，我们依据结果绘制了如上的权重分配结果饼状图。

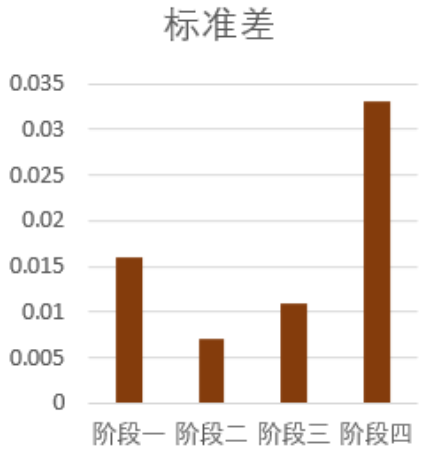
根据此饼状图，我们明显地发现，股票投资所占据的权重在一、二阶段有了大幅度的提升，而其余部分则是呈现出规律不等的增加或是减少。

8.2.3 风险收益特征的构建

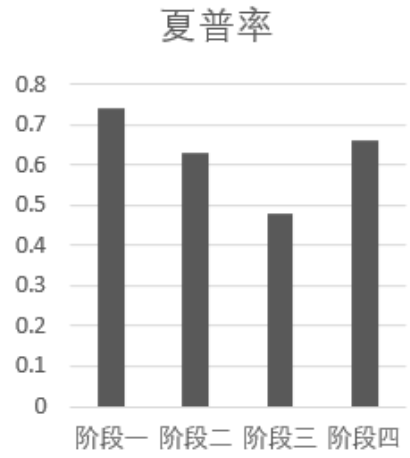
对于我们所建立的变权重风险评价模型，可以根据第二问中的风险收益特征指标以及相应的计算方法对我们的组合基于历史数据进行风险收益特征计算，根据计算结果，我们绘制了如下条形图。



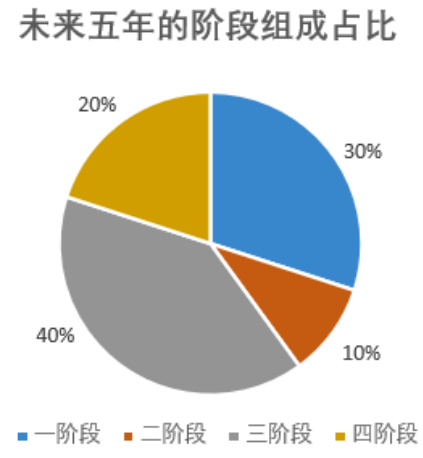
(a) 组合年化率



(b) 组合标准差



(c) 组合夏普率



(d) 未来五年阶段组成占比

从结果导向而言，组合具有非常强的年化收益期望，在预测模型判断正确的条件下，我们的组合可以获得稳定的收益。基于第二问的判断，我们同时分析了未来五年从 2021 年第二季度到 2026 年第一季度的阶段组成，其中第一阶段共有 6 个季度，第二阶段为 2 个季度，第三阶段为 2 年，第四阶段为一年。

最终，按照时间的长度加权平均的方式，我们可以计算我们的组合在未来五年的年化收益率为 6.72%，日标准差为 0.0165，夏普率为 0.609。

九、 模型的评价、改进与推广

9.1 模型评价

优点：1 本文采用改进的美林时钟，基于数据提高了投资时钟的准确性与可靠性，与经典的美林时钟理论有着显著的优越性。2 本文采用变权重的风险平价模型，相较于经典的风险平价模型资产配置集中于波动小的资产，变权重的风险平价模型在保持风险相对稳定的情况下，有效地提高了投资组合的最终收益。缺点：1

9.2 模型的改进与推广

1 除了可以改变不同的资产对总风险的贡献率外，还可以改变风险衡量的类型来改变风险平价模型的权重，例如金融学里的下行标准差。2 该模型可以在题给数据范围内，为大类资产的资源配置机构制定投资组合策略提供参考。同时，因为题给数据涵盖大部分大类资产。在实际情况下，对模型进行改进后，可以将模型推广用于更多大类资产配置策略的构建。

参考文献

[1] 华清, 赵学军, 黄一黎. 资产配置模型的选择: 回报、风险抑或二者兼具 [J]. 统计研究,2018,35(07):62-76.DOI:10.19343/j.cnki.11-1302/c.2018.07.006.

[2] https://max.book118.com/html/2019/1112/8103126025002063.shtm

[3] 文捷. 基于风险平价策略的大类资产配置实证研究 [D]. 浙江大学,2017.

[4] 伟. 全球宏观经济周期与资产轮动——基于中国视角的美林投资时钟修正研究 [J]. 全国流通经济,2021(20):3-7.DOI:10.16834/j.cnki.issn1009-5292.2021.20.001.

[5] 文涛, 董敏杰, 徐灼. 时钟上的波浪——中国金融周期波动与美林时钟的框架结合 [J]. 金融博览,2018(11):38-39.

附录 A　支撑材料目录

- data.csv
- GDP 季度.csv
- 大类资产分区.csv
- 大类资产预测分区.xlsx
- 金融建模.ipynb
- 美林时钟相关指数.csv
- 年化率.xlsx
- 收益率标准差.csv
- 挑选大类指数.csv
- 夏普率.csv
- 新增债务指数与 GDP 平减指数.csv
- 债务未光滑.csv
- 指数分阶段.csv

附录 B　模型计算代码使用说明

2.1　配置总环境

```
import pandas as pd
import numpy as np
import seaborn as sns
from pylab import *
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pylab import *
from scipy.stats import chi2_contingency
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import silhouette_score
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
import statsmodels.api as sm
from sklearn.neighbors import NearestNeighbors
from random import sample
from numpy.random import uniform
from math import isnan
import matplotlib.dates as dates
from scipy.integrate import simps
from scipy import signal
import plotly.io as pio
import plotly.express as px
import plotly.graph_objects as go
import plotly
from plotly.subplots import make_subplots
from scipy.signal import argrelextrema
from sklearn.preprocessing import MinMaxScaler
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号 #有中文出现的情况，需要u'内容'
def rebuild_year(df):
    df=df.rename(columns={'指标名称':'Date'})
    df['Date']=pd.to_datetime(df['Date'])
    df = pd.DataFrame(df).set_index('Date')
    df=df.resample('A').asfreq()
    return df
def rebuild_date(df):
    #重命名第一列
    df=df.rename(columns={'指标名称':'Date'})
    #转化为时间格式
    df['Date']=pd.to_datetime(df['Date'])
    #设置索引
```

```
    df = pd.DataFrame(df).set_index('Date')
    #填充日期，将日期以日为单位填充满
    df=df.resample('M').asfreq()
    #对新填充的数值进行线性拟合
    df=df.interpolate(method='cubic')
    return df
```

2.2 新增债务规模于 GDP 平减指数

```
df_gdp=pd.read_excel('./附件1/国民经济核算/国内生产总值GDP(年).xlsx')
df_cpi=pd.read_excel('./附件1/价格指数/CPI.xlsx')
df_gg=pd.read_excel('./附件1/国民经济核算/中国宏观杠杆率(季).xlsx')
df_gdpr=pd.read_csv('./附件1/GDP不变价new.csv')
df_gdpj=pd.read_csv('./附件1/GDP同减同比.csv')
#数据预处理GDP不变价
df_gdpr=df_gdpr.iloc[:,1:3]
df_gdpr=df_gdpr.rename(columns={'时间':'Date'})
df_gdpr['Date']=df_gdpr['Date'].astype('int64')
df_gdpr['Date']=pd.to_datetime(df_gdpr['Date'],format="%Y")
df_gdpr= pd.DataFrame(df_gdpr).set_index('Date')
df_gdpr=df_gdpr.resample('D').asfreq()
df_gdpr=df_gdpr.interpolate(method='cubic')
df_gdpr=df_gdpr.resample('Q').asfreq()
df_gdpr.plot()
#数据预处理GDP平减指数
df_gdpj=df_gdpj.iloc[:,1:3]
df_gdpj=df_gdpj.rename(columns={'时间':'Date'})
df_gdpj['Date']=df_gdpj['Date'].astype('int64')
df_gdpj['Date']=pd.to_datetime(df_gdpj['Date'],format="%Y")
df_gdpj= pd.DataFrame(df_gdpj).set_index('Date')
df_gdpj=df_gdpj.resample('D').asfreq()
df_gdpj=df_gdpj.interpolate(method='cubic')
df_gdpj=df_gdpj.resample('Q').asfreq()
df_gdpj.plot()
#对杠杆率指数进行预处理，转化为相同的时间索引
df_gg.drop(df_gg.head(3).index,inplace=True)
df_gg.drop(df_gg.tail(2).index,inplace=True)
df_gg=df_gg.iloc[:,0:2]
df_gg['实体经济部门杠杆率']=df_gg['实体经济部门杠杆率'].astype(np.float64)
df_gg_month=rebuild_date(df_gg)
df_gg_month=df_gg_month.resample('Q').asfreq()
df_gg_month.plot()
#合并杠杆率指数和GDp方便计算新增债务规模
df_unit=pd.merge(df_gdpr,df_gg_month,on='Date')
df_unit['new']=""
```

```
for i in range(len(df_unit['GDP:不变价(亿元)'])):
    if(i!=0):
        df_unit.iloc[i,2]=(df_unit.iloc[i,1]*df_unit.iloc[i,0]-df_unit.iloc[i-1,1]*df_unit.iloc[i-1,0])/df_unit.
    #合并GDP不变价和平减指数
df_new = df_unit.iloc[:,2]
df_two=pd.merge(df_new,df_gdpj,on='Date')
df_two.drop(df_two.head(1).index,inplace=True)
df_two.drop(df_two.tail(1).index,inplace=True)
df_smooth=df_two.iloc[:,0]
smooth = signal.savgol_filter(df_smooth,11,3)
new_smooth=pd.Series(smooth.tolist())
smooth = signal.savgol_filter(new_smooth,7,3)
df_new_smooth=pd.DataFrame(smooth)
df_smooth.plot()
df_new_smooth.plot()
df_new_smooth=df_new_smooth.set_index(df_two.index)
dftest=df_two.iloc[:,1]
df_two=pd.merge(dftest,df_new_smooth,on='Date')
#画出新增债务规模与GDP平减指数的趋势图
matplotlib.rcParams['axes.unicode_minus']=False
plt.rcParams['font.sans-serif']=['SimHei']
df_two.plot()
plt.legend(bbox_to_anchor=(1.05,1.0))
plt.show()
#处理现价gdp数据
df_gdp=df_gdp.iloc[:,[0,1]]
df_gdp.drop(df_gdp.head(3).index,inplace=True)
df_gdp.drop(df_gdp.tail(2).index,inplace=True)
df_gdp['GDP:现价']=df_gdp['GDP:现价'].astype(np.float64)
df_gdp=rebuild_date(df_gdp)
df_gdp=df_gdp.resample('Q').asfreq()
#计算现价GDP增速
df_gdp=df_gdp-df_gdp.shift(12)
df_gdp=df_gdp/df_gdp.shift(12)
df_gdp=df_gdp.dropna()
#计算不变价GDP增速
df_gdpr=df_gdpr-df_gdpr.shift(12)
df_gdpr=df_gdpr/df_gdpr.shift(12)
df_gdpr=df_gdpr.dropna()
df1=pd.merge(df_gdpj,df_gdpr,on='Date')
df1=pd.merge(df1,df_gdp,on='Date')
#归一化处理
df1=df1.apply(lambda x: (x - np.min(x)) / (np.max(x) - np.min(x)))
#画出GDP平减指数，GDP不变价和GDP现价之间的关系
fig = plt.figure(figsize=(12,8),dpi=200)
plt.plot(list(df1.index.values),df1.iloc[:,0],label='GDP:平减指数',color='#000000',linewidth =
    4)
```

```
plt.plot(list(df1.index.values),df1.iloc[:,1],label='GDP:不变价',color='#707a80',linewidth = 4)
plt.plot(list(df1.index.values),df1.iloc[:,2],label='GDP:现价',color='#A63F03',linewidth = 4)
plt.grid()
plt.tick_params(labelsize=13)
plt.xlabel('日期',fontsize=20)
x_major_locator=MultipleLocator(1920)
ax=plt.gca()
ax.xaxis.set_major_locator(x_major_locator)
plt.legend()
plt.show()
```

2.3 构建 cpi 与 gdp 增速的关系

```
#预处理
df_gdp.drop(df_gdp.head(3).index,inplace=True)
df_gdp.drop(df_gdp.tail(2).index,inplace=True)
df_cpi.drop(df_cpi.tail(2).index,inplace=True)
df_cpi=df_cpi.iloc[:,0:2]
df_gdp=df_gdp.iloc[:,0:2]
df_gdp['GDP:现价']=df_gdp['GDP:现价'].astype(np.float64)
df_gdp=rebuild_date(df_gdp)
#计算gdp增速
df2=df_gdp-df_gdp.shift(12)
df2=df2/df_gdp.shift(12)
df2=df2.dropna()

# dfgdp3=df2-df2.shift(12)
# dfgdp3=dfgdp3/df2.shift(12)
# dfgdp3=dfgdp3.dropna()
df2.plot()
df_cpinew=rebuild_date(df_cpi)
df_cpinew.plot()
#合并数据归一化后绘制图
dfall=pd.merge(df2,df_cpinew,on='Date')
dfall_one=dfall.apply(lambda x: (x - np.min(x)) / (np.max(x) - np.min(x)))
matplotlib.rcParams['axes.unicode_minus']=False
plt.rcParams['font.sans-serif']=['SimHei']
dfall_one.plot()
plt.legend(bbox_to_anchor=(1.05,1.0))
plt.show()
```

2.4 预测相关指数

```
df_pre=pd.read_csv('美林时钟.csv',encoding='gbk')
```

```
df_pre
Date_real=df_pre.iloc[0:419,0].tolist()
Date_pre=df_pre.iloc[419:-1,0].tolist()
GDP_real=df_pre.iloc[0:419,1].tolist()
GDP_pre=df_pre.iloc[419:-1,1].tolist()
CPI_real=df_pre.iloc[0:419,2].tolist()
CPI_pre=df_pre.iloc[419:-1,2].tolist()
#GDP同比增速预测
fig = plt.figure(figsize=(12,8),dpi=200)
plt.plot(Date_real,GDP_real,label='GDP同比增速真实值',color='#323B38',linewidth = 4)
plt.plot(Date_pre,GDP_pre,label='GDP同比增速预测值',color='#BF6415',linewidth = 4)
plt.grid()
plt.tick_params(labelsize=13)
plt.xlabel('日期',fontsize=20)
plt.ylabel('GDP增速(%)',fontsize=20)
x_major_locator=MultipleLocator(60)
ax=plt.gca()
ax.xaxis.set_major_locator(x_major_locator)
plt.legend()
plt.show()
#CPI指数预测
fig = plt.figure(figsize=(12,8),dpi=200)
plt.plot(Date_real,CPI_real,label='CPI真实值',color='#323B38',linewidth = 4)
plt.plot(Date_pre,CPI_pre,label='CPI预测值',color='#BF6415',linewidth = 4)
plt.grid()
plt.tick_params(labelsize=13)
plt.xlabel('日期',fontsize=20)
plt.ylabel('CPI同比(%)',fontsize=20)
x_major_locator=MultipleLocator(60)
ax=plt.gca()
ax.xaxis.set_major_locator(x_major_locator)
plt.legend()
plt.show()
df_pre_two=pd.read_csv('data.csv',encoding='gbk')
date_real=df_pre_two.iloc[0:109,0].tolist()
date_pre=df_pre_two.iloc[109:-1,0].tolist()
new_real=df_pre_two.iloc[0:109,2].tolist()
new_pre=df_pre_two.iloc[109:-1,2].tolist()
gdp_real=df_pre_two.iloc[0:109,1].tolist()
gdp_pre=df_pre_two.iloc[109:-1,1].tolist()
#新增债务规模与GDP平减指数
fig = plt.figure(figsize=(12,8),dpi=200)
plt.plot(date_real,new_real,label='新增债务规模',color='#000000',linewidth = 4)
plt.plot(date_real,gdp_real,label='GDP平减指数',color='#A63F03',linewidth = 4)
plt.grid()
plt.tick_params(labelsize=13)
plt.xlabel('日期',fontsize=20)
```

```
plt.ylabel('比率(%)',fontsize=20)
x_major_locator=MultipleLocator(20)
ax=plt.gca()
ax.xaxis.set_major_locator(x_major_locator)
plt.legend()
plt.show()
#新增债务规模与GDP平减指数预测
fig = plt.figure(figsize=(12,8),dpi=200)
plt.plot(date_real,new_real,label='新增债务指数真实值',color='#000000',linewidth = 4)
plt.plot(date_pre,new_pre,label='新增债务指数预测值',color='#707a80',linewidth = 4)
plt.plot(date_real,gdp_real,label='GDP平减指数真实值',color='#A63F03',linewidth = 4)
plt.plot(date_pre,gdp_pre,label='GDP平减指数预测值',color='#D9B166',linewidth = 4)
plt.grid()
plt.tick_params(labelsize=13)
plt.xlabel('日期',fontsize=20)
plt.ylabel('比率(%)',fontsize=20)
x_major_locator=MultipleLocator(20)
ax=plt.gca()
ax.xaxis.set_major_locator(x_major_locator)
plt.legend()
plt.show()
#找出极大值极小值
array1=np.array(df_pre_two.iloc[:,2])
peak_indexes = signal.argrelextrema(array1, np.greater, order=3)
peak_indexes = peak_indexes[0]
peak_indexes
valley_indexes = signal.argrelextrema(array1, np.less, order=3)
valley_indexes = valley_indexes[0]
valley_indexes
#找出新增债务规模指数极值
(fig, ax) = plt.subplots()
ax.plot(np.array(df_pre_two.iloc[:,0]), array1)
peak_x = peak_indexes
peak_y = array1[peak_indexes]
ax.scatter(peak_x, peak_y, marker='o', color='red', label="Peaks")

# Plot valleys
valley_x = valley_indexes
valley_y = array1[valley_indexes]
ax.scatter(valley_x, valley_y, marker='o', color='green', label="Valleys")
plt.title('新增债务指数极大极小值')
# 添加图例
plt.legend(loc='best')
x_major_locator=MultipleLocator(20)
ax=plt.gca()
ax.xaxis.set_major_locator(x_major_locator)
# # 保存图像
```

```
# plt.savefig('peaks-valleys.png')
# 显示图像
plt.show()
array2=np.array(df_pre_two.iloc[:,1])
peak_indexes = signal.argrelextrema(array2, np.greater, order=3)
peak_indexes = peak_indexes[0]
peak_indexes
valley_indexes = signal.argrelextrema(array2, np.less, order=3)
valley_indexes = valley_indexes[0]
valley_indexes
#找出GDP平减指数的极值
(fig, ax) = plt.subplots()
ax.plot(np.array(df_pre_two.iloc[:,0]), array2)
peak_x = peak_indexes
peak_y = array2[peak_indexes]
ax.scatter(peak_x, peak_y, marker='o', color='red', label="Peaks")

# Plot valleys
valley_x = valley_indexes
valley_y = array2[valley_indexes]
ax.scatter(valley_x, valley_y, marker='o', color='green', label="Valleys")
plt.title('GDP平减指数极大极小值')
# 添加图例
plt.legend(loc='best')
x_major_locator=MultipleLocator(20)
ax=plt.gca()
ax.xaxis.set_major_locator(x_major_locator)
# # 保存图像
# plt.savefig('peaks-valleys.png')
# 显示图像
plt.show()
df_depart['class']=''
#判断两个指数分为4个阶段
for i in range(len(df_depart['Date'])):
    if(df_depart.iloc[i,3]==1 and df_depart.iloc[i,4]==1):
        df_depart.iloc[i,5]=2
    elif(df_depart.iloc[i,3]==1 and df_depart.iloc[i,4]==0):
        df_depart.iloc[i,5]=3
    elif(df_depart.iloc[i,3]==0 and df_depart.iloc[i,4]==1):
        df_depart.iloc[i,5]=1
    else:
        df_depart.iloc[i,5]=4
df_depart_real=df_depart.iloc[0:108,:]
df_depart_real=df_depart_real.rename(columns={'GDP:赛冲嘶鎸囨暣:錫屾瘪(%)':'GDP平减指数(%)'})
df_depart_real=df_depart_real.rename(columns={'新增债务指数':'新增债务指数(%)'})
df_record=df_depart_real
df_depart_real['Date']=pd.to_datetime(df_depart_real['Date'])
```

```
df_depart_real = pd.DataFrame(df_depart_real).set_index('Date')
df_plt=df_depart_real.iloc[:,[0,1]]
fig = px.line(df_plt)
fig['data'][0]['line']['color']='rgb(0, 0, 0)'
fig['data'][0]['line']['width']=3
fig['data'][1]['line']['color']='rgb(139, 69, 19)'
fig['data'][1]['line']['width']=3
start=df_record.iloc[0,0]
#根据指数分阶段
for i in range(len(df_record['Date'])-1):
    if(df_record.iloc[i,5]!=df_record.iloc[i+1,5]):
        end=df_record.iloc[i,0]
        if(df_record.iloc[i,5]==1):
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="1", annotation_position="top left",
                    fillcolor="red", opacity=0.25, line_width=0)
        elif(df_record.iloc[i,5]==2):
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="2", annotation_position="top left",
                    fillcolor="blue", opacity=0.25, line_width=0)
        elif(df_record.iloc[i,5]==3):
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="3", annotation_position="top left",
                    fillcolor="green", opacity=0.25, line_width=0)
        else:
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="4", annotation_position="top left",
                    fillcolor="yellow", opacity=0.25, line_width=0)
        start=df_record.iloc[i,0]

fig.show()
df_depart_pre=df_depart.iloc[108:-1,:]
df_depart_pre=df_depart_pre.rename(columns={'GDP:骞冲崕鐪囧尯:鍚屾瘮(%)':'GDP平减指数预测值(%)'})
df_depart_pre=df_depart_pre.rename(columns={'新增债务指数':'新增债务指数预测值(%)'})
df_record_pre=df_depart_pre
df_depart_pre['Date']=pd.to_datetime(df_depart_pre['Date'])
df_depart_pre = pd.DataFrame(df_depart_pre).set_index('Date')
df_plt_pre=df_depart_pre.iloc[:,[0,1]]
#根据预测指数分阶段
fig = px.line(df_plt_pre)
fig['data'][0]['line']['color']='rgb(0, 0, 0)'
fig['data'][0]['line']['width']=3
fig['data'][1]['line']['color']='rgb(139, 69, 19)'
fig['data'][1]['line']['width']=3
start=df_record_pre.iloc[0,0]
for i in range(len(df_record_pre['Date'])-1):
    if(df_record_pre.iloc[i,5]!=df_record_pre.iloc[i+1,5]):
```

```
        end=df_record_pre.iloc[i,0]
        if(df_record_pre.iloc[i,5]==1):
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="1", annotation_position="top left",
                    fillcolor="red", opacity=0.25, line_width=0)
        elif(df_record_pre.iloc[i,5]==2):
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="2", annotation_position="top left",
                    fillcolor="blue", opacity=0.25, line_width=0)
        elif(df_record_pre.iloc[i,5]==3):
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="3", annotation_position="top left",
                    fillcolor="green", opacity=0.25, line_width=0)
        else:
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="4", annotation_position="top left",
                    fillcolor="yellow", opacity=0.25, line_width=0)
        start=df_record_pre.iloc[i,0]
fig.add_vrect(x0=df_record_pre.iloc[18,0], x1=df_record_pre.iloc[21,0],
                    annotation_text="2", annotation_position="top left",
                    fillcolor="blue", opacity=0.25, line_width=0)
fig.show()
```

2.5 不同指标在不同阶段的收益特征

```
df_alldata=pd.read_excel('./附件2/大类资产指数行情数据.xlsx')
df_alldata
#东证1000指数阶段图
df_zz1000=df_alldata.iloc[:,[0,4]]
df_zz1000.drop(df_zz1000.head(2).index,inplace=True)
df_zz1000.drop(df_zz1000.tail(2).index,inplace=True)
df_zz1000=df_zz1000.dropna()
df_zz1000 = df_zz1000.drop(df_zz1000[df_zz1000['Unnamed: 4'] ==0].index)
df_zz1000=df_zz1000.rename(columns={'Unnamed: 4':'中证1000指数'})
df_zz1000=df_zz1000.rename(columns={'Unnamed: 0':'Date'})
df_zz1000['Date']=pd.to_datetime(df_zz1000['Date'])
df_zz1000 = pd.DataFrame(df_zz1000).set_index('Date')
fig = px.line(df_zz1000)
fig['data'][0]['line']['color']='rgb(139, 69, 19)'
fig['data'][0]['line']['width']=2
start=df_record.iloc[0,0]
for i in range(len(df_record['Date'])-1):
    if(df_record.iloc[i,5]!=df_record.iloc[i+1,5]):
        end=df_record.iloc[i,0]
        if(df_record.iloc[i,5]==1):
```

```
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="1", annotation_position="top left",
                    fillcolor="red", opacity=0.25, line_width=0)
        elif(df_record.iloc[i,5]==2):
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="2", annotation_position="top left",
                    fillcolor="blue", opacity=0.25, line_width=0)
        elif(df_record.iloc[i,5]==3):
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="3", annotation_position="top left",
                    fillcolor="green", opacity=0.25, line_width=0)
        else:
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="4", annotation_position="top left",
                    fillcolor="yellow", opacity=0.25, line_width=0)
        start=df_record.iloc[i,0]

fig.show()
#标普高盛商品全收益指数阶段图
df_bp=df_alldata.iloc[:,[0,6]]
df_bp.drop(df_bp.head(2).index,inplace=True)
df_bp.drop(df_bp.tail(2).index,inplace=True)
df_bp=df_bp.dropna()
df_bp = df_bp.drop(df_bp[df_bp['Unnamed: 6'] ==0].index)
df_bp=df_bp.rename(columns={'Unnamed: 6':'标普高盛商品全收益指数'})
df_bp=df_bp.rename(columns={'Unnamed: 0':'Date'})
df_bp['Date']=pd.to_datetime(df_bp['Date'])
df_bp = pd.DataFrame(df_bp).set_index('Date')
fig = px.line(df_bp)
fig['data'][0]['line']['color']='rgb(139, 69, 19)'
fig['data'][0]['line']['width']=2
start=df_record.iloc[0,0]
for i in range(len(df_record['Date'])-1):
    if(df_record.iloc[i,5]!=df_record.iloc[i+1,5]):
        end=df_record.iloc[i,0]
        if(df_record.iloc[i,5]==1):
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="1", annotation_position="top left",
                    fillcolor="red", opacity=0.25, line_width=0)
        elif(df_record.iloc[i,5]==2):
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="2", annotation_position="top left",
                    fillcolor="blue", opacity=0.25, line_width=0)
        elif(df_record.iloc[i,5]==3):
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="3", annotation_position="top left",
                    fillcolor="green", opacity=0.25, line_width=0)
```

```
        else:
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="4", annotation_position="top left",
                    fillcolor="yellow", opacity=0.25, line_width=0)
        start=df_record.iloc[i,0]

fig.show()
#中债-综合财富(7-10年)指数阶段图
df_zj=df_alldata.iloc[:,[0,9]]
df_zj.drop(df_zj.head(2).index,inplace=True)
df_zj.drop(df_zj.tail(2).index,inplace=True)
df_zj=df_zj.dropna()
df_zj = df_zj.drop(df_zj[df_zj['Unnamed: 9'] ==0].index)
df_zj=df_zj.rename(columns={'Unnamed: 9':'中债-综合财富(7-10年)指数'})
df_zj=df_zj.rename(columns={'Unnamed: 0':'Date'})
df_zj['Date']=pd.to_datetime(df_zj['Date'])
df_zj = pd.DataFrame(df_zj).set_index('Date')
fig = px.line(df_zj)
fig['data'][0]['line']['color']='rgb(139, 69, 19)'
fig['data'][0]['line']['width']=3
start=df_record.iloc[0,0]
for i in range(len(df_record['Date'])-1):
    if(df_record.iloc[i,5]!=df_record.iloc[i+1,5]):
        end=df_record.iloc[i,0]
        if(df_record.iloc[i,5]==1):
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="1", annotation_position="top left",
                    fillcolor="green", opacity=0.25, line_width=0)
        elif(df_record.iloc[i,5]==2):
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="2", annotation_position="top left",
                    fillcolor="red", opacity=0.25, line_width=0)
        elif(df_record.iloc[i,5]==3):
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="3", annotation_position="top left",
                    fillcolor="blue", opacity=0.25, line_width=0)
        else:
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="4", annotation_position="top left",
                    fillcolor="yellow", opacity=0.25, line_width=0)
        start=df_record.iloc[i,0]

fig.show()
#货币基金指数阶段图
df_hb=df_alldata.iloc[:,[0,10]]
df_hb.drop(df_hb.head(2).index,inplace=True)
df_hb.drop(df_hb.tail(2).index,inplace=True)
```

```
df_hb=df_hb.dropna()
df_hb = df_hb.drop(df_hb[df_hb['Unnamed: 10'] ==0].index)
df_hb=df_hb.rename(columns={'Unnamed: 10':'货币基金'})
df_hb=df_hb.rename(columns={'Unnamed: 0':'Date'})
df_hb['Date']=pd.to_datetime(df_hb['Date'])
df_hb = pd.DataFrame(df_hb).set_index('Date')
fig = px.line(df_hb)
fig['data'][0]['line']['color']='rgb(139, 69, 19)'
fig['data'][0]['line']['width']=3
start=df_record.iloc[0,0]
for i in range(len(df_record['Date'])-1):
    if(df_record.iloc[i,5]!=df_record.iloc[i+1,5]):
        end=df_record.iloc[i,0]
        if(df_record.iloc[i,5]==1):
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="1", annotation_position="top left",
                    fillcolor="red", opacity=0.25, line_width=0)
        elif(df_record.iloc[i,5]==2):
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="2", annotation_position="top left",
                    fillcolor="blue", opacity=0.25, line_width=0)
        elif(df_record.iloc[i,5]==3):
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="3", annotation_position="top left",
                    fillcolor="green", opacity=0.25, line_width=0)
        else:
            fig.add_vrect(x0=start, x1=end,
                    annotation_text="4", annotation_position="top left",
                    fillcolor="yellow", opacity=0.25, line_width=0)
        start=df_record.iloc[i,0]

fig.show()

#归一化后四个指数的走势图
fig = plt.figure(figsize=(12,8),dpi=200)
plt.plot(dateall,df_tocorr_one.iloc[:,0],label='中证1000指数',color='#000000',linewidth = 2.5)
plt.plot(dateall,df_tocorr_one.iloc[:,1],label='标普高盛商品全收益指数',color='#707a80',linewidth
    = 2.5)
plt.plot(dateall,df_tocorr_one.iloc[:,2],label='中债-综合财富(7-10年)指数',color='#A63F03',linewidth
    = 2.5)
plt.plot(dateall,df_tocorr_one.iloc[:,3],label='货币基金',color='#D9B166',linewidth = 2.5)
plt.grid()
plt.tick_params(labelsize=13)
plt.xlabel('日期',fontsize=20)
x_major_locator=MultipleLocator(730)
ax=plt.gca()
ax.xaxis.set_major_locator(x_major_locator)
```

```
plt.legend()
plt.show()
#计算收益率标准差
df_return=df_tocorr.pct_change()
df_revar=np.var(df_return)
temp1=df_depart_real.iloc[:,4]
temp1=temp1.resample('D').asfreq()
temp2=pd.merge(temp1,df_tocorr,on='Date')
# temp2.to_csv('大类资产分区.csv')
df_alldepart=pd.read_csv('大类资产分区.csv',encoding='gbk')
df_alldepart['Date']=pd.to_datetime(df_alldepart['Date'])
   #设置索引
df_alldepart = pd.DataFrame(df_alldepart).set_index('Date')
df_one=df_alldepart[df_alldepart['class'] == 1]
df_two=df_alldepart[df_alldepart['class'] == 2]
df_three=df_alldepart[df_alldepart['class'] == 3]
df_four=df_alldepart[df_alldepart['class'] == 4]
df_onevar=np.std(df_one.pct_change())
df_twovar=np.std(df_two.pct_change())
df_threevar=np.std(df_three.pct_change())
df_fourvar=np.std(df_four.pct_change())
df_std = pd.concat([df_onevar,df_twovar,df_threevar,df_fourvar], axis=1)
df_std=df_std.rename(columns={0:'第一阶段',1:'第二阶段',2:'第三阶段',3:'第四阶段'})
df_std=df_std.iloc[[1,2,3,4],:]

#计算夏普率
# 一般国内无风年利率普遍习采用一年期存款利率，约为3\%，一年有52周，则周化无风险利率为
    (1+3\%)^(1/52)-1
    为0.0569\%。标准差也用当周收益率跟历史周收益均值进行计算。【（1+4\%）的1/365次方
    】-1=0.0001059015326852

df_stdmeanone=np.mean(df_one.pct_change())
df_stdmeantwo=np.mean(df_two.pct_change())
df_stdmeanthree=np.mean(df_three.pct_change())
df_stdmeanfour=np.mean(df_four.pct_change())
df_xpone=(df_stdmeanone-0.0001059015326852)/df_onevar*math.sqrt(255)
df_xptwo=(df_stdmeantwo-0.0001059015326852)/df_twovar*math.sqrt(255)
df_xpthree=(df_stdmeanthree-0.0001059015326852)/df_threevar*math.sqrt(255)
df_xpfour=(df_stdmeanfour-0.0001059015326852)/df_fourvar*math.sqrt(255)
df_xp = pd.concat([df_xpone,df_xptwo,df_xpthree,df_xpfour], axis=1)
df_xp=df_xp.rename(columns={0:'第一阶段',1:'第二阶段',2:'第三阶段',3:'第四阶段'})
df_xp=df_xp.iloc[[1,2,3,4],:]
df_xp.to_csv('夏普率.csv',encoding='utf_8_sig')
```

2.6 变权重风险平价模型

```
from scipy.optimize import minimize
df_pointall=pd.read_excel('大类资产预测分区.xlsx')
df_pointall
def risk_budget_objective(weights,cov):
    weights = np.array(weights) #weights为一维数组
    sigma = np.sqrt(np.dot(weights, np.dot(cov, weights))) #获取组合标准差
    #sigma = np.sqrt(weights@cov@weights)
    MRC = np.dot(cov,weights)/sigma #MRC = cov@weights/sigma
    #MRC = np.dot(weights,cov)/sigma
    TRC = weights * MRC
    delta_TRC = [sum((i - TRC)**2) for i in TRC]
    return sum(delta_TRC)
def total_weight_constraint(x):
    return np.sum(x)-1.0
def long_only_constraint(x):
    return x
def compute(cov,x0):
    bnds = tuple((0,None) for x in x0)
    cons = ({'type': 'eq', 'fun': total_weight_constraint})
    print(bnds)
#     cons = ({'type':'eq', 'fun': lambda x: sum(x) - 1})
    options={'disp':False, 'maxiter':1000, 'ftol':1e-20}
    solution = minimize(risk_budget_objective,x0,args=(cov), bounds=bnds, constraints=cons,
        method='SLSQP', options=options)
    # 求解出权重
    final_weights = solution.x #权重
    for i in range(len(final_weights)):
        print(f'{final_weights[i]:.1\%}投资于{R_cov.columns[i]}')df_one=df_alldepart[df_alldepart['class']
            == 1]
df_two=df_alldepart[df_alldepart['class'] == 2]
df_three=df_alldepart[df_alldepart['class'] == 3]
df_four=df_alldepart[df_alldepart['class'] == 4]
df_one=df_one.iloc[:,1:6]
df_one=df_one.pct_change()
df_one=df_one.iloc[1:,:]
R_cov_one = df_one.cov()
cov_one = np.array(R_cov_one)
x1=[0.45,0.25,0.17,0.13]
compute(cov_one,x1)
df_two=df_two.iloc[:,1:6]
df_two=df_two.pct_change()
df_two=df_two.iloc[1:,:]
R_cov_two = df_two.cov()
cov_two = np.array(R_cov_two)
x2=np.array([0.52,0.31,0.10,0.7])
compute(cov_two,x2)
```

```
df_three=df_three.iloc[:,1:6]
df_three=df_three.pct_change()
df_three=df_three.iloc[1:,:]
R_cov_three = df_three.cov()
cov_three = np.array(R_cov_three)
x3=np.array([0.62,0.25,0.7,0.6])
compute(cov_three,x3)
df_four=df_four.iloc[:,1:6]
df_four=df_four.pct_change()
df_four=df_four.iloc[1:,:]
R_cov_four = df_four.cov()
cov_four = np.array(R_cov_four)
x4=np.array([80,12,5,3])
compute(cov_four,x4)
df_pointall=df_pointall.iloc[:,1:]
df_preone=df_pointall[df_pointall['阶段'] == 1]
df_pretwo=df_pointall[df_pointall['阶段'] == 2]
df_prethree=df_pointall[df_pointall['阶段'] == 3]
df_prefour=df_pointall[df_pointall['阶段'] == 4]
df_preonevar=np.std(df_preone.pct_change())
df_pretwovar=np.std(df_pretwo.pct_change())
df_prethreevar=np.std(df_prethree.pct_change())
df_prefourvar=np.std(df_prefour.pct_change())
df_prestd = pd.concat([df_preonevar,df_pretwovar,df_prethreevar,df_prefourvar], axis=1)
df_prestd=df_prestd.rename(columns={0:'第一阶段',1:'第二阶段',2:'第三阶段',3:'第四阶段'})
df_prestd=df_prestd.iloc[0:4,:]
```