

UNIVERSITY OF GRONINGEN



RESEARCH MASTER THESIS

A Modified Parametric Approach for Portfolio Optimization Problem

Authors:

SHUYI WANG

s3068145

Supervisor:

PROF. DR. LAMMERTJAN DAM

August 31, 2021

Abstract

This paper investigates the influence of firm characteristics on portfolio selection by modifying the parametric portfolio policy proposed by Brandt et al. (2009). We select eleven firm characteristics that represent as much as the company operating condition. Firm characteristics are able to explain the cross-section return, and we assume that they contribute to asset weighting under the framework of Brandt. By applying principal components analysis (PCA), we construct multiple new “characteristics”, a linear transformation of the original firm characteristics set, model the asset weights as function of them and build the corresponding principal components portfolios (pc-portfolios). Besides, we impose cross-sectional short-selling limits on each asset of the portfolio. We compare the results of a base case portfolio (formed by using the eleven firm characteristics) and a benchmark (value-weighted) portfolio with our pc-portfolios. We find that one pc-portfolio outperforms the base case and the benchmark, however, it has higher volatility. We assess the pc-portfolios performance under various risk aversion levels and examine the profit stability across in-sample and out-of-sample experiments. We show that our strategies are robust out-of-sample or do not have in-sample overfitting. Moreover, compared to the benchmark portfolio, our findings indicate that the pc-portfolios are not easily affected by different risk aversion levels.

Keywords: *Portfolio Optimization, Firm Characteristics, PCA, Asset Allocation, Parametric Portfolio Weights*

1 Introduction

This paper explores the relationship between the information provided by firm characteristics and the asset weights. We are aiming to construct new portfolio policies by refining and extending the existed parametric method proposed by Brandt et al. (2009). A portfolio is defined as a collection of securities, including stocks, bonds, commodities, cash and cash equivalents, that are able to generate profits. Individual and institutional investors carefully make their portfolio selection to gain excess return in the financial market. The classic portfolio selection problem consists of assets allocation and the corresponding weights determination. A parametric approach is to modeling asset weights with observable economic variables, such as macroeconomic states or firm characteristics, and find the parameters by maximizing the objective expected utility function. We are aiming to solve for a large-scale cross-sectional portfolio selection problem, which contains all the stocks in the investable universe. Our study modifies this process by constructing new variables that represent firms' operating conditions, and demonstrates that these variables can capture the information related to assets weighting more efficiently.

The pioneering work of Markowitz (1952, 1960) describes the optimal portfolio selection problem as a process to minimize the portfolio variance at a prescribed return and to solve for assets weighting. He introduces the mean-variance (MV) methodology and quantitatively frames the optimization process. MV theory treats individual asset return as random variables and, by assessing the corresponding expected value and variance, is able to quantify the return and risk at the portfolio level. He proposes that the covariance matrix can reflect the dependence among the assets and that, by achieving the MV efficiency, investors are able to diversify the portfolio and gain expected return at given risk level. However, holding a mean-variance efficient portfolio is not a widely applied strategy by active portfolio managers (L. K. Chan et al., 1999) since the MV approach has practical problems. First, the mathematical optimization process is sensitive to budget constraints. The model obtains unreasonable large negative (short) positions on many assets (Black

and Litterman, 1992) if no constraints imposed. The shorts positions require sufficient liquidity of assets, however, small companies usually are not capable of providing such liquidity. Besides, when imposing weights constraints, such as short sell limitations, the method over-weights stocks with small market capitalization. Second, active portfolio managers hold different views of expected return of an asset over time, which substantially leads to frequent relocation of assets and thus high transactions costs. Hence, further research focus on economic variables that affect investors' subjective views of conditional return distribution (Aït-Sahali and Brandt, 2001). For instance, investors may refer to single or multiple firm characteristics, such as book-to-market ratio (*bm*), market capitalization (*mktcap*) (e.g., Eugene and French, 1992, 1996), presented by financial or analyst reports to make investment decisions. These firm characteristics reflect the operating conditions of a company and are proven useful to predict returns¹, however, translating the characteristics directly to investment advice is plausible. Although the firm characteristics provide information associated with stocks expected return, variance and covariance with other stocks (L. K. C. Chan et al., 1998), modeling the joint distribution of returns, variance, covariance and the characteristics requires the covariance matrix to be positive definite. Michaud (1989) also shows that the MV procedure does not yield stable results. In addition to the econometric requirements of the covariance matrix, the modelling process will cause substantial computational burden when applying the universe of all assets. Accordingly, for the past decades, researchers and professional assets managers have sought for new methods either because of the formidable requirements of covariance matrix or because they intend to discover superior returns from other sources (Black and Litterman, 1992).

Under the framework of MV theory, Brandt et al. (2009) develop a parametric method handling the firm characteristics as variables to directly quantify the asset

¹For instance, Avramov (2002) shows that, in a Bayesian framework, dividend yield, book-to-market ratio and earnings yield both in-sample and out-of-sample predictability; Campbell and Viceira (1999) indicate that investors who face risk-less interest rate (Treasury Bill yield) and time-varying equity premium have hedging demands.

weights in a portfolio, and the method avoids the econometric assumptions of modeling the joint distribution of return and firm characteristics and is able to yield consistent econometric inference. Besides, the method directly maximizes the investors' expected utility function, and thus we can tune the risk preference related parameter. The parametric method simplifies the computation when considering the universe of all assets. Besides, over the investment period, the asset allocation is determined by the coefficients of firm characteristics and is thus convenient to impose constraints such as short selling constraints (Jagannathan and Ma, 2003). Modeling with size, value and winner factors², Brandt's policy of portfolio obtains significant positive excess return both in-sample and out-of-sample. Besides, the approach outperforms a passive benchmark (value-weight portfolio), and the authors argue that the method can be applied to multiple asset classes.

Our research contributes to portfolio optimization literature and the relevant methodology. Since a firm characteristic³ could represent a dimension that implicitly reveals the performance of a company, such as valuation or profitability, we argue that multiple characteristics cover more dimensions and explain the performance more effectively and that firm characteristics are able to affect asset weights within the framework of Brandt. Nevertheless, the related literature is scarce. One possible concern is that applying many firm characteristics as explanatory variables to capture the variation of asset weights could overfit and cause over-weighting on particular assets. To be specific, from a statistical perspective, more characteristics are able to explain the more variance. However, given the calculation of firm characteristics, they might be related, especially if they are describing the same dimension⁴, and the information related is thus overlapping. Secondly, it is also

²They are market capitalization, book-to-market ratio and past 12-month moving average returns.

³Zou and Stan (1998) use the firm characteristics to depict the demographics and managerial situations. The characteristic variables includes size, leverage, turnover, growth, ownership structure and even board characteristics (e.g., Subrahmanyam and Titman, 1998, McKnight and Weir, 2009, Kogan and Tian, 2012).

⁴For example, we use book-to-market Ratio and cash flow ratio to describe *Valuation*, they demonstrate the value of a company from market and operating perspectives respectively. Likely,

possible that some of characteristics do not necessarily contribute to asset weights and are (partially) noise. To solve the two problems, we impose Principal Component Analysis (PCA) (Pearson, 1901) to the information set of firm characteristics. We intend to extract factors that explain the most of the variance of firm characteristics and wisely ignore noise. We substitute the firm characteristics with their principal components, accordingly we use the principal components to compute for asset weights and building principal component optimized (pc-optimized) portfolios.

PCA is a widely applied dimension reduction technique which synthesizes information from the provided information space, construct new variables, and form the asset weights accordingly. The goal of dimension reduction is to transform the high-dimensional dataset to lower dimension representation that retains the original properties. Many similar methods are proposed to solve multivariate problems⁵. In finance, PCA (Pearson, 1901) is the most commonly applied method to reduce dimension and construct new pricing factors(e.g., Fujiwara et al., 2006, Jothimani et al., 2017, Han et al., 2018, Jiang et al., 2018, Suh et al., 2014). Besides, PCA has potentials to reveal “latent” factors (Giglio and Xiu, 2021). Our study uses a high-dimensional dataset involving eleven firm characteristics and applies PCA to reduce dimension by the projection of the data points onto a few given components. We obtain such components and form our new “characteristics” for the asset weights problem. The new characteristics preserve as much variation of the original data as possible. By applying PCA, we implicitly assume that characteristic set is a combination of a desired informational set and a noisy set. This incorporates with the implication that the aggregation of various firm characteristics share the return-related information, however, they are (partially) noisy since they contain information directing differently about expect return (Light et al., 2017). Although not directly solving the correlation with expected return, we apply PCA

return on assets and return on equity describe *Profitability* from the efficiency of a company using asset or equity , respectively, to generate profit. Table 2.1 shows more details.

⁵For instance, Hotelling (1935) propose canonical correlation; Wold (1975) use partial least square to build latent variables; Blei et al. (2003) apply latent dirichlet allocation to solve classification problems

on the firm characteristics aiming to extract the most relevant information and to show the importance of the corresponding components. By using the information of firm characteristics more efficiently and constructing input variables for the optimal weights problem, we are aiming to build portfolios outperforming the Brandt et al.'s portfolio, equal-weighted and value-weighted portfolios. Another advantage of applying PCA is that we avoid optimizing the objective function with the whole characteristics set (eleven characteristics in our sample) and reduce the computational complexity. The constructed PCA variables are only the linear transformation of the original firm characteristics and thus do not change the original structure of the optimization problem (see in problem (12)).

This paper extends the main method of Brandt et al. (2009) with principal components and short sell constraints. Our modified method presents desired results. Firstly, we show that our parameterization of asset weights as function of principal components generates higher cumulative return than the value-weighted portfolio over the investment period. However, our methods face higher volatility, resulting in lower Sharpe ratio. Secondly, we argue that our model yields stable economic benefits and does not overfit the data. We examine the performance between two sub-samples, namely in-sample and out-of-sample. We split the datasets, both stock return and firm-level characteristics, equally into two parts. Given that the future prices are not observable, we set the first half of the data as in-sample, and the second half of the data as out-of-sample. We find that, with eight components, the out-of-sample portfolio generate higher return and Sharpe ratio than the in-sample. Thirdly, we perform experiments with various risk aversion coefficients since the method depends highly on the investors' risk preference. Our results indicate that the sign and magnitude of coefficients vary across different risk aversions. However, all the pc-optimized portfolios show similar weight distribution and stable return and volatility under different risk preferences. This implies that risk preference do not easily affect pc-optimized portfolios.

The remainder of the paper is organized as follows. We provide a description of the

basic methodology with our extensions and data in Section 2, we apply our method and present the empirical results in Section 3, including base case and pc-optimized cases. In Section 4 conclusion can be found.

2 Data and Methodology

2.1 Data

Our sample contains monthly stock price from CRSP and the firm-level characteristics from CRSP Industry Financial Ratios (WIFR hereafter) dataset, from January 1970 to December 2020. For each firm, we also calculate monthly stock return and market capitalization (*mktcap*), as one of the firm-level characteristics. We define *mktcap* as the log of the the current price per share times the total outstanding number of shares. Instead of intentionally choosing “profitable” firm characteristics, we tend to select characteristics covering as many dimensions as possible for a firm. As defined by the CRSP WIFR⁶, seven commonly applied categories of company characteristics: *Capitalization*, *Valuation*, *Financial Soundness/Solvency*, *Profitability*, *Liquidity*, *Efficiency*, *other*. Our selection of firm-level characteristics are based on these seven categories and described in Table 2.1:

Practical work of active portfolio manager face unbalanced investing pools since the number of tradable companies varies over the investment period. In order to provide sufficient solution for a portfolio choice problem, we also take into account the case that the companies may not survive through the 51-year period, either it is caused by delisting or it is due to data missing. Besides, we select companies that have been listing for more than 5 years (included). The investing pools are rebalanced at the end of each year. The average annual growth rate of firm number is 1.6%, with the fewest firms in 1970 (1317 firms) and the most firms in 1998 (2732 firms). The average number of tradable firms across the investing pools is 1814. We obtain one-month Treasury bill rate from CRSP database as the risk-free rate, and the rate

⁶Find more detail on https://wrds-www.wharton.upenn.edu/documents/793/WRDS_Industry_Financial_Ratio_Manual.pdf

Table 2.1: WIFR Firm Characteristics Definition, U.S. Stock, 1970-2020

Firm Characteristics	Description	Category
Market Capitalization (<i>mktcap</i>)	Log of Market Capitalization	Capitalization
Equity to Invested Capital (<i>equity invcap</i>)	Common Equity as a fraction of Invested Capital	Capitalization
Book to Market Ratio (<i>bm</i>)	Book Value of Equity as a fraction of Market value of Equity	Valuation
Cash Flow Ratio (<i>pcf</i>)	Multiple of Market Value of Equity to Net Cash Flow from Operating Activities	Valuation
Accrual (<i>accrual</i>)	Accruals as a fraction of average Total Assets based on most recent two periods	Financial Soundness
Cash Flow Margin (<i>cfm</i>)	Income before Extraordinary Items and Depreciation as a fraction of Sales	Financial Soundness
Return on Asset (<i>roa</i>)	Return on Asset	Profitability
Return on Equity (<i>roe</i>)	Return on Equity	Profitability
Current Ratio (<i>curr ratio</i>)	Current Assets as a fraction of Current Liabilities	Liquidity
Debt/Asset Ratio (<i>debt to asset</i>)	Total Debt as a fraction of Total Assets	Solvency
Asset Turnover Ratio (<i>at turnover</i>)	Sales as a fraction of the average Total Assets based on the most recent two periods	Efficiency

is scaled with the same period as the our sample.

Since most firm characteristics are based on quarterly-updated financial fundamentals data, we scale data quarterly for further analysis. We found multiple outliers for characteristics around 2001, and thus we winsorize the firm characteristics at level 2.5% and 97.5% to minimize the effect of extreme values. The following Table 2.2 demonstrates the descriptive statistics for the cross-sectional mean and volatility. Panel A summarizes the statistics for the return and the firm-level characteristics and Panel B shows those for the principal components. We also display the cross-sectional mean and standard deviation over time in the Figure A4.1 and Figure A4.2 shown in **Appendix A**.

2.2 Methodology

2.2.1 Parametric Weights

We apply the weight parameterization method in Brandt et al. (2009), which assigns each asset weight $\omega_{i,t}$ for stock i at date t to the sum of a benchmark portfolio weight and a vector of estimates of firm characteristics:

$$\omega_{i,t} = \bar{\omega}_{i,t} + \frac{1}{N_t} \theta' \hat{\mathbf{y}}_{i,t}, \quad (1)$$

where $\hat{\mathbf{y}}_{i,t}$ is a vector of firm characteristics and $\bar{\omega}_{i,t}$ is the weight of stock i at t in a benchmark portfolio, equal-weighted portfolio in the following case. θ is the corresponding time-invariant coefficients for each firm characteristics. It is obvious that $\theta' \hat{\mathbf{y}}_{i,t}$ is treated as deviation from the benchmark portfolio weights. In order to ensure the weights sum to one, the firm characteristics are standardized cross-sectionally. Besides, we can compare the magnitude of the estimated coefficients. We also set no-short sell constraint for the asset weights since large-scale portfolio management does face short sell constraints in the real world. In order to ensure the parameterized weights still sum to one, we impose the constraint as follow:

Table 2.2: Cross-sectional Monthly Firm Characteristics, Principal Components, Descriptive Statistics, U.S. 1970-2020

Panel A		Cross-Sectional Mean				Cross-Sectional Volatility			
Variables	<i>n</i>	<i>mean</i>	<i>Std.Dev</i>	<i>min</i>	<i>max</i>	<i>mean</i>	<i>Std.Dev</i>	<i>min</i>	<i>max</i>
Return	612.0	0.013	0.060	-0.281	0.305	0.139	0.040	0.070	0.397
Market Capitalization	612.0	12.063	1.201	9.828	14.158	2.039	0.148	1.629	2.391
Equity to Invested Capital(%)	612.0	0.727	0.027	0.619	0.776	0.284	0.228	0.207	4.138
Book to Market Ratio(%)	612.0	0.849	0.322	0.462	2.300	0.914	2.535	0.301	60.173
Cash Flow Ratio(%)	612.0	8.121	2.531	2.356	15.055	42.828	12.508	15.162	119.130
Accrual(%)	612.0	0.036	0.088	-1.143	0.131	0.381	3.786	0.072	54.278
Cash Flow Margin(%)	612.0	-2.174	4.582	-45.710	0.126	60.010	170.456	0.095	1938.477
Return on Asset(%)	612.0	0.110	0.035	0.020	0.178	0.169	0.045	0.087	0.358
Return on Equity(%)	612.0	0.095	0.396	-0.770	5.216	2.872	12.614	0.114	152.537
Current Ratio(%)	612.0	2.877	0.243	2.295	3.865	5.481	6.049	1.486	44.233
Debt/Asset Ratio(%)	612.0	0.479	0.019	0.440	0.539	0.199	0.022	0.164	0.352
Asset Turnover Ratio(%)	612.0	1.289	0.217	0.800	1.694	0.926	0.101	0.643	1.150
Risk-free Return	612.0	0.011	0.008	0.000	0.038				
Panel B									
Principal Component 1	612.0	0.0	0.182	-0.367	0.989	1.252	0.267	0.666	2.244
Principal Component 2	612.0	0.0	0.080	-0.104	0.363	0.364	0.100	0.138	0.712
Principal Component 3	612.0	0.0	0.038	-0.053	0.191	0.161	0.043	0.068	0.379
Principal Component 4	612.0	0.0	0.013	-0.029	0.072	0.078	0.020	0.034	0.163
Principal Component 5	612.0	0.0	0.004	-0.015	0.035	0.036	0.010	0.009	0.089
Principal Component 6	612.0	0.0	0.002	-0.008	0.016	0.016	0.006	0.003	0.043
Principal Component 7	612.0	0.0	0.001	-0.002	0.004	0.008	0.003	0.001	0.019
Principal Component 8	612.0	0.0	0.000	-0.001	0.002	0.004	0.002	0.000	0.010

$$\omega_{i,t} = \frac{\max[0, \omega_{i,t}]}{\sum_{t=1}^{N_t} \max[0, \omega_{i,t}]} \quad (2)$$

Using this parametric approach avoids assuming the joint distribution of returns and each firm characteristics. Instead, it tends to estimate the optimal portfolio weights by directly maximize investors' utility function. We assume that investors have constant relative risk aversion (CRRA) preference. With respect to different γ , we can estimate the characteristics coefficients with different level of risk aversion:

$$u(r_{p,t+1}) = \frac{(1 + r_{p,t+1})^{1-\gamma}}{1 - \gamma}, \quad (3)$$

where u is the objective utility function taking the portfolio return $r_{p,t+1}$ as input variable:

$$r_{p,t+1} = \sum_{i=1}^{N_t} \left(\bar{\omega}_{i,t} + \frac{1}{N_t} \theta' \hat{\mathbf{y}}_{i,t} \right) r_{i,t+1}. \quad (4)$$

Hence, we can write down the maximization problem as follows:

$$\max_{\{\omega_{i,t}\}_{i=1}^{N_t}} \mathbf{E}_t [u(r_{p,t+1})] = \mathbf{E}_t \left[u \left(\sum_{i=1}^{N_t} \omega_{i,t} r_{i,t+1} \right) \right] \quad (5)$$

$$= \mathbf{E}_t \left[u \left(\sum_{i=1}^{N_t} \left(\bar{\omega}_{i,t} + \frac{1}{N_t} \theta' \hat{\mathbf{y}}_{i,t} \right) r_{i,t+1} \right) \right]. \quad (6)$$

2.2.2 Principal Components

We assume that the firm characteristics capture multiple dimensions of a company. For example, book-to-market (*bm*) ratio is commonly considered as one of the valuation metrics, and the ratio indicates if the company is over- or under-valued. This directs investors to take different positions of the corresponding stock and thus the

asset weights in a portfolio. Our intention is to include multiple firm characteristics that provide multi-dimensional information that contribute to asset weights. However, it is possible that these information are overlapping or noisy. Therefore, we propose Principal Component Analysis (PCA) (Pearson, 1901) to the weight parametric method, expecting to extract useful information for the optimized problem.

PCA aims to decompose multivariate dataset in a set of successive orthogonal components that explain a maximum amount of the variance (Pedregosa et al., 2011). We formalize the process of the PCA extension to the parametric method as follows:

Suppose that we have the (observed) firm characteristics space \mathcal{X} containing n characteristics:

$$\mathcal{X}_{t \times n} = [\mathbf{x}_1 \ \mathbf{x}_2 \ \cdots \ \mathbf{x}_n]_t, \quad t = 1, 2, \dots, T \quad (7)$$

where each \mathbf{x} is a T -dimension vector representing the observation for company i . We assume that $\hat{\mathbf{y}}_{i,t} \in \mathbb{R}^m$ in (2) is a m -dimension vector containing the principal components from the firm characteristics space \mathcal{X} for company i at time t ($n \gg m$). We can write down a linear transformation for \mathbf{y} :

$$\mathbf{y} = A^T \mathbf{x}, \quad (8)$$

where A is the coefficient for each components:

$$A = [\mathbf{a}_1 \ \mathbf{a}_2 \ \cdots \ \mathbf{a}_n] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad (9)$$

and $a_i = (a_{1i}, a_{2i}, \dots, a_{ni})^T, i = 1, 2, \dots, n$. Therefore, a linear transformation of \mathbf{x} without losing any information is:

$$\mathbf{y} = a_i^T \mathbf{x} = a_{1i}\mathbf{x}_1 + a_{2i}\mathbf{x}_2 + \dots + a_{ni}\mathbf{x}_n, \quad i = 1, 2, \dots, n. \quad (10)$$

In our case, we select $m(m \ll n)$ components that explain a high percentage of variance and investigate if such extraction can assign asset weights more efficiently. We can rewrite the maximization problem:

$$\max_{\theta} \mathbf{E}_t[u(r_{p,t+1})] = \mathbf{E}_t \left[u \left(\sum_{i=1}^{N_t} \left(\bar{\omega}_{i,t} + \frac{1}{N_t} \theta' (a_i^T \mathbf{x}) \right) r_{i,t+1} \right) \right] \quad (11)$$

$$= \frac{1}{T} \sum_{t=0}^{T-1} u \left(\sum_{i=1}^{N_t} \left(\bar{\omega}_{i,t} + \frac{1}{N_t} \theta' (a_i^T \mathbf{x}) \right) r_{i,t+1} \right) \quad (12)$$

PCA presents good linear properties that do not affect the structure of the objective function. Besides, taking into account all the firm characteristics causes tremendous computational burden, and PCA solves this issue by applying less but more efficient components of observed characteristic set. In our practical cases, we construct new variables from the eleven firm characteristics. We specify from two to eight principal components for $a_i^T \mathbf{x} (i = 2, 3, \dots, 8)$ in problem (12).

3 Empirical Application

We produce the empirical results from four perspectives. First, we start from the base case which includes the eleven firm-level characteristics as variables to determine the asset weights, and subsequently, we compare this optimized portfolio policy with benchmark portfolios, such as value-weighted portfolio. To illustrate the advantages of substituting firm characteristics with their principal components, we construct the pc-portfolios involving from two to eight principal components and show a comprehensive comparison between these portfolios. Second, we compare the

performance between two investable universes, namely the All Stocks and Top500 Stock universes. This part serves as a robustness check and reflects the impact of different investable pools. Third, to further illustrate the effectiveness and robustness of our approach, we perform the in-sample and out-of-sample experiments. Such experiments are widely applied in portfolio optimization literature and are crucial steps to prove the effectiveness of a strategy. Unless stated, we assume the investors' CRRA preference and a relative risk aversion of five. In the fourth part, we examine the risk preference influence on the original portfolio policy and pc-portfolios by showing the portfolio performances under a range of risk preference quantities. It should be noted that we impose the short sell constraints to the asset weights, specified in equation (2). In other words, we only consider long-only portfolios.

We organize all the tables as follows. The upper few rows describe the estimates of parameters and the associated standard errors are derived from the hessian matrix, which represents the second derivative of the estimates of utility function. Since the variables are cross-sectionally standardized, the magnitude of coefficients can be compared. The middle few rows present the asset weight information, including average weight, maximum weight and minimum weight across the firms and over the investment period. The bottom few rows assess the performance of the portfolios by showing the average return, standard deviation and Sharpe ratio. For simplicity, the measures in the bottom rows are annualized.

3.1 Base and PCA Cases

We display the results of base case optimized portfolio, relative to equal-weighted and value-weighted portfolios in Table 3.3 (from column (1) to (3)). For equal-weighted portfolio, the asset weights are only scaled by the number of stocks of the year. For the value-weighted portfolio, the asset weights depend on the firm's share of the whole market capitalization of the year. Therefore, the short sell constraints do not affect these two portfolios. Since the investing pool is rebalanced annually, the asset weights vary over the investment period. For the optimized portfolio, the

average asset weight is 0.049⁷, which is slightly higher than that of equal-weighted portfolio (0.046) and differs relatively much from that of value-weighted portfolio. In our setting, the benchmark weight \bar{w} in equation (1) is the average weight of the year. This is the reason that the average optimized weight is close to the equal weight. Most of coefficients from the third columns are statistically significant. We find that the strategy over-weights the companies with higher large market capitalization, cash flow ratio, ROA, ROE, accrual ratio and asset turnover ratio. This conceptually incorporated with accounting literature. We find the greatest positive coefficient of cash flow ratio and negative coefficient of debt-to-asset ratio, which indicate that investors tend to increase the weight of a company with higher cash flow ratio and decrease that with higher debt-to-asset ratio. More importantly, this negative effect is larger than the positive one. From the bottom few rows, the optimized portfolio has a higher average return than that of the equal-weighted and value-weighted portfolios, 18.3% versus 17.8% and 13.1%, respectively. We can visually find this in Figure C4.3 about the cumulative return of the three portfolios. We find that the optimized portfolio outperforms other two portfolios over the most of investment period. However, the optimized portfolio is exposed to more risk, given that it has higher volatility of 18.0% as opposed to 17.1% and 13.7% for the equal-weighted and value-weighted portfolios, respectively.

We subsequently present the results of our extension for the parametric method with the principal components shown in Table 3.4 Panel A. We list the coefficients and the standard error in parentheses. The principal components do not have economic meanings and only represent variables that explain variance. All the coefficients in each specification are significant. The statistics show that the average weight decrease as we involve more components. The portfolio with eight components achieves the highest average return of 17.8% and relatively lower volatility 18.4%, as opposed to the 20.0% and 19.8% volatility of two-component and three-component strategies. Our approach of extracting information from characteristics outperform the value-weighted portfolio, however, fail to beat the equal-weighted

⁷The weight is multiplied by 100, hereafter wherever mentioned

Table 3.3: Optimized Portfolio Performances, All Stocks vs. Top500 Stocks, U.S. Stocks 1970-2020

Variables	All Stocks		Top500 Stocks			
	(1) Eq. Weighted	(2) Val. Weighted	(3) Optimized	(4) Eq. Weighted	(5) Val. Weighted	(6) Optimized
θ_{mktcap}	-	-	82.49*** (4.397)	-	-	-105.19*** (4.131)
$\theta_{equityinvcap}$	-	-	-165.90*** (2.323)	-	-	41.02*** (3.341)
θ_{bm}	-	-	-8.56** (4.327)	-	-	-96.63*** (6.574)
θ_{pcf}	-	-	303.61*** (7.059)	-	-	41.99*** (6.794)
$\theta_{accrual}$	-	-	82.64*** (2.835)	-	-	50.67*** (4.60)
θ_{cfm}	-	-	-274.68*** (7.058)	-	-	101.16*** (2.164)
θ_{roa}	-	-	4.70 (3.253)	-	-	-21.39*** (9.112)
θ_{roe}	-	-	120.30*** (7.056)	-	-	-37.55*** (5.979)
$\theta_{currratio}$	-	-	-304.64*** (9.854)	-	-	-79.8*** (4.854)
$\theta_{debttoasset}$	-	-	-420.87*** (9.636)	-	-	-75.20*** (3.563)
θ_{atturn}	-	-	250.31*** (15.022)	-	-	-146.25*** (1.988)
$ w_i \times 100$	0.046	0.024	0.049	0.046	0.114	0.041
$\max w_i \times 100$	0.076	8.630	4.731	0.200	9.209	3.347
$\min w_i \times 100$	0.037	0.001*	0.000	0.200	0.001*	0.000
\bar{r}	0.178	0.131	0.183	0.145	0.124	0.079
$\sigma(r)$	0.171	0.137	0.180	0.150	0.135	0.194
Sharpe Ratio	0.971	0.867	0.950	0.887	0.830	0.345

* The minimum weight is too small, we only approximate it to 0.001

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

and the original optimized portfolios. We display this fact in Figure C4.4. The PCA cumulative returns, over most of the investment period, are not above the optimized portfolio return.

3.2 Top500 Stocks Performance

Instead of using the universe of all the stocks, we investigate how the approach perform in the investing pool containing the largest 500 companies (Top500), defined by the top 500 market capitalization and rebalanced every year. We demonstrate the statistics of the optimized portfolio with the equal-weighted and value-weighted portfolios in Table 3.3 (from column (4) to (6)). This subset contains the high-quality and high-liquidity companies of the market. Therefore, it is of interest to use this subset in practice for large scale active portfolio management. By comparing column (3) and (6), we find that the optimized strategy weights assets differently in the two pools. The opposed signs of the same coefficients indicates that, in different pools, the strategy evaluates firms based on different characteristics. The portfolio has 7.9% average return, compared to 14.5% and 12.4% of equal-weighted and value-weighted portfolio. However, its standard deviation (19.4%) is unfortunately higher, which indicates the optimized strategy endure more risk, and leads to 0.345 Sharpe ratio. This result implies that, given the investing pool does not contain small companies, the optimized strategy highly weights the small companies and small companies generate more profits. Figure C4.5 displays the cumulative returns for all the portfolios. It is visually clear that the Top500 stock related strategies under-perform those with all the stocks.

We assess the performance of pc-optimized strategies in the Top500 stocks pool, and Table 3.4 presents the results. The pc-optimized portfolios demonstrate relatively stable and better results. The weight distributions are similar, given that the average weight for the seven specifications are around 0.250. These strategies generate higher returns as most of them achieve more than 12.5% average return. With two components involved, we achieve the highest average return and volatility of 13.1% and 18.1%. These combine into the highest Sharpe ratio, among the

Table 3.4: Principal Components Optimized Portfolio Performances, All Stocks vs. Top500 Stocks, U.S. Stocks 1970-2020

Panel A Variables	All Stocks						
	$pc = 2$	$pc = 3$	$pc = 4$	$pc = 5$	$pc = 6$	$pc = 7$	$pc = 8$
θ_{pc1}	121.73*** (1.141)	133.10*** (0.645)	-126.98*** (0.968)	72.89*** (0.429)	-42.52*** (0.583)	-84.06*** (0.487)	90.41*** (0.353)
θ_{pc2}	-379.51*** (0.758)	-333.63*** (0.609)	13.16*** (0.581)	-36.09*** (0.419)	-38.87*** (0.560)	82.22*** (0.422)	26.85*** (0.467)
θ_{pc3}		41.93*** (0.681)	170.65*** (0.658)	70.73*** (0.555)	69.60*** (0.590)	17.66*** (0.556)	-33.17*** (0.537)
θ_{pc4}			156.72*** (0.582)	53.16*** (0.546)	77.67*** (0.581)	25.06*** (0.597)	-24.51*** (0.499)
θ_{pc5}				123.51*** (0.466)	-48.44*** (0.618)	-121.19*** (0.505)	31.44*** (0.340)
θ_{pc6}					134.45*** (0.586)	65.63*** (0.676)	-124.16*** (0.744)
θ_{pc7}						34.12*** (0.474)	182.29 *** (0.720)
θ_{pc8}							119.42*** (0.396)
$ w_i \times 100$	0.054	0.054	0.052	0.051	0.051	0.052	0.049
$\max w_i \times 100$	7.044	6.462	3.000	3.648	2.679	2.317	3.223
$\min w_i \times 100$	0.000	0.000	0.000	0.000	0.000	0.000	0.000
\bar{r}	0.167	0.163	0.158	0.164	0.160	0.165	0.178
$\sigma(r)$	0.200	0.198	0.189	0.184	0.190	0.196	0.184
<i>Sharpe Ratio</i>	0.775	0.763	0.772	0.826	0.779	0.781	0.902

Panel B Variable	Top500 Stocks						
	$pc = 2$	$pc = 3$	$pc = 4$	$pc = 5$	$pc = 6$	$pc = 7$	$pc = 8$
θ_{pc1}	175.79*** (2.970)	-137.48*** (1.286)	-329.87*** (1.129)	-98.90*** (0.898)	-79.01*** (0.845)	-48.85*** (0.730)	-85.78*** (0.756)
θ_{pc2}	315.25*** (1.687)	192.25*** (2.122)	91.14*** (1.478)	17.15*** (1.395)	48.71*** (1.411)	54.30*** (1.314)	60.60*** (1.071)
θ_{pc3}		206.05*** (1.964)	120.83*** (1.457)	137.42*** (1.122)	101.43*** (1.313)	93.98*** (1.051)	96.67*** (1.165)
θ_{pc4}			90.88*** (1.237)	-43.48*** (1.331)	-69.03*** (1.169)	-95.16*** (0.927)	-59.65*** (1.144)
θ_{pc5}				53.24*** (1.242)	61.23*** (1.241)	45.37*** (0.958)	24.54*** (0.987)
θ_{pc6}					-40.11*** (1.193)	-37.92*** (1.077)	-33.1*** (1.021)
θ_{pc7}						28.96*** (0.957)	5.04*** (0.964)
θ_{pc8}							12.09*** (1.064)
$ w_i \times 100$	0.257	0.252	0.262	0.248	0.247	0.243	0.248
$\max w_i \times 100$	13.73	12.65	19.81	12.65	10.98	5.814	11.33
$\min w_i \times 100$	0.000	0.000	0.000	0.000	0.000	0.000	0.000
\bar{r}	0.131	0.126	0.112	0.117	0.124	0.126	0.127
$\sigma(r)$	0.181	0.180	0.179	0.177	0.176	0.175	0.179
<i>Sharpe Ratio</i>	0.657	0.633	0.559	0.591	0.638	0.653	0.642

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

seven models, of 0.657. We can argue that, in the Top500 pool, the pc-optimized strategies are more efficiently capture the useful information provided by the firm characteristics. However, these specifications do not outperform the original strategy and pc-optimized strategies using all the stocks. In Figure C4.6, we display the cumulative returns for the Top500 pool. We can see that the pc-optimized curves are above the original curve.

3.3 In- and Out-of-Sample Performance

We establish the robustness of our approach through the out-of-sample experiments. Our intention is to test if the approach is able to avoid over-fitting since we estimate a large number of variables to optimize the portfolio. The experiments for the base case are shown in Table 3.5. We focus on the in-sample and out-of-sample performance of pc-portfolios and present the results of experiments for each specification in Table 3.6. The whole dataset is split into two parts equally. We define the first half as the in-sample from January 1970 through December 1995, define the second half as the out-of-sample from January 1996 through December 2020. We estimate the coefficients for the in-sample and apply the estimated coefficients to the out-of-sample to compute the asset weights and to construct the portfolio. To be specific, the coefficients in the out-of-sample is not re-estimated but directly used from the in-sample estimation, and we forecast the average return and volatility of the out-of-sample. The estimates and performance for the in-sample and out-of-sample are displayed in Panel A and B respectively. We also list the results of value-weighted portfolio for both in-sample and out-of-sample. Given the nature of time-series that we can only observe the past information of the in-sample and that the future information of out-of-sample is not observable, we only use the estimates from the in-sample and not from the out-of-sample. We firstly calculate the out-of-sample annualized volatility (21%), which is nearly double that of in-sample volatility, approximately 12.5%. Therefore, we have to face the facts that any approach will suffer from higher volatility out-of-sample and that the out-of-sample Sharpe ratios

are likely lower than in-sample⁸.

Table 3.5: Optimized Portfolio Performance, In- and Out-of-Sample Experiments, U.S. Stocks (In-sample: 1970-1995; Out-of-Sample: 1996-2020)

	All Stocks			
	In-Sample Val.Weighted	Opt.	Out-of-Sample Val.Weighted	Opt. Fcst.
θ_{mktcap}	-	2.024 (1.609)	-	2.024
$\theta_{equityinvcap}$	-	89.20*** (1.109)	-	89.20
θ_{bm}	-	10.20*** (2.167)	-	10.20
θ_{pcf}	-	26.36*** (1.376)	-	26.36
$\theta_{accrual}$	-	-24.38*** (2.030)	-	-24.38
θ_{cfm}	-	164.10*** (1.609)	-	164.10
θ_{roa}	-	2.739** (1.194)	-	2.739
θ_{roe}	-	79.44*** (1.492)	-	79.44
$\theta_{currratio}$	-	-56.17*** (1.365)	-	-56.17
$\theta_{debtasset}$	-	-166.95*** (2.152)	-	-166.95
θ_{atturn}	-	-47.85*** (1.649)	-	-47.85
$ w_i \times 100$	0.025	0.051	0.029	0.046
$\max w_i \times 100$	8.630	3.921	7.564	4.153
$\min w_i \times 100$	0.001	0.000	0.001	0.000
\bar{r}	0.065	0.091	0.083	0.083
$\sigma(r)$	0.125	0.161	0.210	0.207
<i>Sharpe Ratio</i>	0.384	0.460	0.371	0.377

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Summarizing from the base case results (Table 3.5), we are able to conclude that the method is robust since the asset weight distribution and performance metrics are

⁸However, this does not necessarily indicate the approach is a failure, since we encounter more crisis, such as dot-com crisis (1999-2001), subprime mortgage crisis (2007-2008) and the most recent Covid-related crisis (2019-present).

similar. We observe that, for the optimized portfolio, the average weights are 0.051 and 0.046 for the in-sample and out-of-sample, respectively. More importantly, the out-of-sample optimized portfolio shows 8.3% and 20.7% for the return and volatility, and the in-sample results show a slightly higher return of 9.1% and a slightly lower volatility of 16.1%. For the in-sample, the portfolio policy gain 2.6% more than the benchmark, however, it does not gain more profit than the benchmark out-of-sample.

Turning to our pc-optimized portfolios in- and out-of-sample performance (Table 3.6), we find that, with various principal components involved, the weight distribution are remarkably similar across two samples. Particularly, most portfolios are able to obtain profit than the benchmark out-of-sample. For the specifications of $pc = 7$ and $pc = 8$, we find astonishing results, in addition to the outperformance over the value-weighted portfolio (both in-sample and out-of-sample), that they achieve higher out-of-sample average return of 9.0% and 9.6% than in-sample average return of 8.4% and 9.2%. We argue that the optimized strategy yields stable and robust gains as the in-sample and out-of-sample performances are remarkably close. The forecast results of pc-optimized portfolios also possess this advantage. Besides, they are able to obtain more profit out-of-sample when including a proper number of components. Hence, we can conclude that the principal components are capable of capturing more information that contributes to asset weights, compared to the equivalent number of firm characteristics.

3.4 Risk Aversion

Our extended approach also depends on the assumption about the investors' preference of risk tolerance. We begin with assuming the coefficient of constant relative risk aversion $\gamma = 5$ for the CRRA utility function specified in equation (6). In this setting, $\gamma = 0$ means that the investors are risk-neutral and do not react to gains or losses. We further report the portfolios with a range of risk aversion level, $\gamma = 1, 3, 7, 9$. For $\gamma = 1$, we turn to the log utility function. The increasing γ indicates that the investors become more risk averse and are more sensitive to losses.

Table 3.6: Principal Components Optimized Portfolio Performances, In-Sample vs. Out-of-sample, U.S. Stocks (In-sample: 1970-1995; Out-of-Sample: 1996-2020)

Panel A Variables	In-sample Estimation (Jan 1970 - Dec 1995)							
	Val. Weighted	$pc = 2$	$pc = 3$	$pc = 4$	$pc = 5$	$pc = 6$	$pc = 7$	$pc = 8$
θ_{pc1}	-	97.21*** (1.158)	104.43*** (0.645)	102.97*** (0.968)	4.162*** (0.429)	-10.20*** (0.583)	-47.98*** (0.487)	54.67*** (0.353)
θ_{pc2}	-	-297.94*** (1.108)	-262.83*** (1.003)	-262.03*** (0.822)	-57.38*** (0.537)	-25.95*** (0.706)	67.45*** (0.700)	31.42*** (0.704)
θ_{pc3}	-		22.25*** (1.003)	16.37*** (0.607)	49.16*** (0.614)	47.40*** (0.704)	22.04*** (0.871)	-2.27*** (0.828)
θ_{pc4}	-			16.37*** (0.607)	56.89*** (0.606)	55.82*** (0.726)	38.79*** (0.776)	-20.83*** (0.653)
θ_{pc5}	-				113.06*** (0.724)	-33.30*** (0.859)	-82.38*** (0.810)	7.53*** (0.550)
θ_{pc6}	-					114.06*** (0.709)	55.43*** (0.805)	-33.64*** (1.240)
θ_{pc7}	-						19.10*** (0.787)	84.23*** (1.293)
θ_{pc8}	-							65.00*** (0.650)
$ w_i \times 100$	0.025	0.052	0.052	0.051	0.050	0.050	0.051	0.049
$\max w_i \times 100$	8.630	3.639	3.747	2.449	2.248	2.679	2.317	2.320
$\min w_i \times 100$	0.001*	0.000	0.000	0.000	0.000	0.000	0.000	0.000
\bar{r}	0.065	0.091	0.087	0.086	0.087	0.091	0.084	0.092
$\sigma(r)$	0.125	0.168	0.167	0.162	0.163	0.289	0.166	0.166
<i>Sharpe Ratio</i>	0.384	0.440	0.419	0.426	0.423	0.442	0.403	0.452
Panel B Variable	Out-of-Sample Forecast (Jan 1996 - Dec 2020)							
	Val. Weighted	$pc = 2$	$pc = 3$	$pc = 4$	$pc = 5$	$pc = 6$	$pc = 7$	$pc = 8$
θ_{pc1}	-	97.21	104.43	102.97	4.162	-10.20	-47.98	54.67
θ_{pc2}	-	-297.94	-262.83	-262.03	-57.38	-25.95	67.45	31.42
θ_{pc3}	-		22.25	16.37	49.16	47.40	22.04	-2.27
θ_{pc4}	-			16.37	56.89	55.82	38.79	-20.83
θ_{pc5}	-				113.06	-33.30	-82.38	7.53
θ_{pc6}	-					114.06	55.43	-33.64
θ_{pc7}	-						19.10	84.23
θ_{pc8}	-							65.00
$ w_i \times 100$	0.029	0.053	0.053	0.050	0.049	0.048	0.050	0.047
$\max w_i \times 100$	7.564	7.308	4.280	2.791	1.890	2.759	1.832	2.714
$\min w_i \times 100$	0.001*	0.000	0.000	0.000	0.000	0.000	0.000	0.000
\bar{r}	0.083	0.085	0.086	0.076	0.080	0.070	0.090	0.096
$\sigma(r)$	0.210	0.229	0.230	0.210	0.210	0.210	0.220	0.200
<i>Sharpe Ratio</i>	0.371	0.349	0.3352	0.338	0.357	0.310	0.386	0.455

* The minimum weight is too small, we only approximate it to 0.001

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Table 3.7 summarizes the weight information and performance metrics for the optimized base case and the pc-portfolios under the different risk preference⁹. For $\gamma = 1$, we find that the investors are not sensitive to risks and that the coefficients are zeros. This explicitly shows that such risk aversion level is almost risk-neutral and leads to no effects on asset weights, and subsequently, the portfolio becomes the equal-weighted portfolio. Surprisingly, for $\gamma = 3$, the portfolio does not generate positive average return (-4.6%) and has relatively high volatility of 19.0%. Besides, with increasing γ , the signs and magnitude of coefficients vary widely across difference specifications. For $\gamma = 9$, the portfolio only gains 5.6% of average return and volatility of 19.2%, combining into 0.231 Sharpe ratio. We display the cumulative return in Figure C4.7 for the variant risk aversion coefficients, with value-weighted portfolio as a reference. For the pc-optimized portfolios, we show that the coefficients are also zero when $\gamma = 1$. This is consistent with the base case results and implying that investors are not sensitive under the case of log-utility. Our pc-optimized portfolios generate stable results since weight distribution and performance metrics are similar across different number of variables and γ . This implies the pc-optimized approach is not easily affected by the investors' risk preference.

4 Conclusion

We extend the portfolio optimization approach with short sell constraints and substitute eleven firm characteristics with their principal components. In our extension, the asset weights are a function of arbitrary principal components. We use from two to eight principal components, fewer than the number of firm characteristics, as variables to determine for each asset weight. The coefficient of each principal component is found through the optimization process of the investors' utility function. To compare among different strategies, we list the estimation results and performances for each specification, including two benchmark portfolios.

⁹Besides, more comprehensive tables reporting the coefficients and standard errors are displayed Table B4.10 and B4.11 in the Appendix B.

Table 3.7: Principal Components Optimized Portfolio Performances, Different Risk Aversion, U.S. Stocks, 1970-2020

	Optimized					$pc = 2$				
	$\gamma = 1$	$\gamma = 3$	$\gamma = 5$	$\gamma = 7$	$\gamma = 9$	$\gamma = 1$	$\gamma = 3$	$\gamma = 5$	$\gamma = 7$	$\gamma = 9$
$ w_i \times 100$	0.046	0.056	0.049	0.056	0.058	0.046	0.056	0.054	0.055	0.055
$\max w_i \times 100$	0.076	4.674	4.731	3.841	6.212	0.076	4.674	7.044	7.153	7.216
$\min w_i \times 100$	0.037	0.000	0.000	0.000	0.000	0.037	0.000	0.000	0.000	0.000
\bar{r}	0.178	-0.046	0.183	0.081	0.056	0.174	0.184	0.167	0.162	0.162
$\sigma(r)$	0.171	0.190	0.180	0.198	0.192	0.171	0.199	0.200	0.200	0.200
<i>Sharpe Ratio</i>	0.971	-0.313	0.950	0.347	0.231	0.944	0.867	0.775	0.750	0.750
	$pc = 3$					$pc = 4$				
	$\gamma = 1$	$\gamma = 3$	$\gamma = 5$	$\gamma = 7$	$\gamma = 9$	$\gamma = 1$	$\gamma = 3$	$\gamma = 5$	$\gamma = 7$	$\gamma = 9$
$ w_i \times 100$	0.076	0.054	0.054	0.054	0.054	0.076	0.054	0.052	0.053	0.054
$\max w_i \times 100$	0.045	6.741	7.044	6.870	3.794	0.045	4.880	3.000	2.968	2.752
$\min w_i \times 100$	0.037	0.000	0.000	0.000	0.000	0.037	0.000	0.000	0.000	0.000
\bar{r}	0.174	0.157	0.167	0.162	0.171	0.174	0.153	0.158	0.162	0.193
$\sigma(r)$	0.171	0.197	0.200	0.199	0.196	0.171	0.199	0.189	0.199	0.199
<i>Sharpe Ratio</i>	0.944	0.733	0.775	0.756	0.811	0.944	0.710	0.772	0.779	0.910
	$pc = 5$					$pc = 6$				
	$\gamma = 1$	$\gamma = 3$	$\gamma = 5$	$\gamma = 7$	$\gamma = 9$	$\gamma = 1$	$\gamma = 3$	$\gamma = 5$	$\gamma = 7$	$\gamma = 9$
$ w_i \times 100$	0.076	0.052	0.051	0.053	0.052	0.076	0.052	0.051	0.052	0.053
$\max w_i \times 100$	0.045	3.413	3.648	4.064	2.653	0.045	2.665	2.679	2.656	2.846
$\min w_i \times 100$	0.037	0.000	0.000	0.000	0.000	0.037	0.000	0.000	0.000	0.000
\bar{r}	0.174	0.165	0.164	0.172	0.153	0.174	0.153	0.160	0.153	0.153
$\sigma(r)$	0.171	0.187	0.184	0.198	0.189	0.171	0.198	0.190	0.196	0.197
<i>Sharpe Ratio</i>	0.944	0.817	0.826	0.806	0.746	0.944	0.785	0.779	0.786	0.752
	$pc = 7$					$pc = 8$				
	$\gamma = 1$	$\gamma = 3$	$\gamma = 5$	$\gamma = 7$	$\gamma = 9$	$\gamma = 1$	$\gamma = 3$	$\gamma = 5$	$\gamma = 7$	$\gamma = 9$
$ w_i \times 100$	0.076	0.052	0.052	0.051	0.051	0.076	0.051	0.049	0.051	0.051
$\max w_i \times 100$	0.045	2.291	2.317	2.712	2.687	0.045	2.387	3.223	2.459	2.270
$\min w_i \times 100$	0.037	0.000	0.000	0.000	0.000	0.037	0.000	0.000	0.000	0.000
\bar{r}	0.174	0.174	0.165	0.164	0.164	0.174	0.173	0.178	0.163	0.170
$\sigma(r)$	0.171	0.192	0.196	0.188	0.188	0.171	0.190	0.184	0.188	0.19
<i>Sharpe Ratio</i>	0.944	0.841	0.781	0.807	0.808	0.944	0.850	0.902	0.805	0.828

We demonstrate that, with the increasing number of principal components involved, the pc-optimized portfolios generate stable average return (around 16.5%). In addition to using all the stocks, we compare performances between different stock pools, one of which contains the largest 500 companies. We show that, by eliminating small companies, the average returns are lower than those in the all stock investing pool. Besides, in the Top500 stocks pool, the pc-optimized portfolios suffer the same level of volatility as they do in the all stocks pool. We argue that the small companies provide more economic benefits and that the strategy over weights the small companies.

We show that a subset of principal components is able to capture weight information and perform closely to the original policy since the return and volatility of the corresponding portfolios are similar to the original policy. Hence, we conclude that our method is more efficiently capture the weighting information. Moreover, our results show that the extended method has two advantageous features that the original policy does not possess. One feature is that the strategy is robust and the model does not overfit. In our in- and out-of-sample experiments, we display that the performances of in-sample and out-of-sample are close. Some forecast results are even better performed out-of-sample than in-sample. The original policy does not gain more profit out-of-sample than in-sample. Another advantage is that, given various level of risk aversion, the pc-optimized portfolios are not easily affected. The performance of the original portfolio is highly affected by the risk aversion coefficients. Although the sign and magnitude of coefficients are not consistent across given different risk aversion coefficients, the pc-optimized portfolios are able to gain stable returns. We did not discover clear pattern indicating the best combination of risk aversion level and number of principal components involved.

This study has limitations. First, the extended approach is not able to produce economic interpretation for the parameterization since the variables are only projection of the original data. The components are ambiguous since each one of them contains information from the eleven firm characteristics. It is thus difficult to de-

termine the number of principal components to be involved without experiments. Second, although the first few of components capture high proportion of the variance, however, this does not indicate the last few components are irrelevant. For instance, in our study, the deliberately omitted ninth, tenth and eleventh components may possess important information related to assets weight even they do not explain as much variance as the former components. Third, we lose “prior knowledge” when applying PCA. The cross-sectional mean of market capitalization and return on asset show a upward trend, however, PCA tend not to incorporate with this feature. In addition to involve more data and more relevant variables, further research could parameterize the PCA in order to capture the prior knowledge of the data.

References

- Arıt-sahali, Y., & Brandt, M. W. (2001). Variable selection for portfolio choice. *The Journal of Finance*, 56(4), 1297–1351.
- Avramov, D. (2002). Stock return predictability and model uncertainty. *Journal of Financial Economics*, 64(3), 423–458.
- Black, F., & Litterman, R. (1992). Global portfolio optimization. *Financial analysts journal*, 48(5), 28–43.
- Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3, 993–1022.
- Brandt, M. W., Santa-Clara, P., & Valkanov, R. (2009). Parametric portfolio policies: Exploiting characteristics in the cross-section of equity returns. *The Review of Financial Studies*, 22(9), 3411–3447.
- Cai, J., & Zhang, Z. (2011). Leverage change, debt overhang, and stock prices. *Journal of Corporate Finance*, 17(3), 391–402.
- Campbell, J. Y., & Viceira, L. M. (1999). Consumption and portfolio decisions when expected returns are time varying. *The Quarterly Journal of Economics*, 114(2), 433–495.
- Chan, L. K. C., Karceski, J., & Lakonishok, J. (1998). The risk and return from factors. *Journal of Financial and Quantitative Analysis*, 33(2), 159–188. <https://doi.org/10.2307/2331306>
- Chan, L. K., Karceski, J., & Lakonishok, J. (1999). On portfolio optimization: Forecasting covariances and choosing the risk model. *The review of Financial studies*, 12(5), 937–974.
- Clubb, C., & Naffi, M. (2007). The usefulness of book-to-market and roe expectations for explaining uk stock returns. *Journal of Business Finance & Accounting*, 34(1-2), 1–32.
- DuCharme, L. L., Malatesta, P. H., & Sefcik, S. E. (2004). Earnings management, stock issues, and shareholder lawsuits. *Journal of financial economics*, 71(1), 27–49.

- Eugene, F., & French, K. (1992). The cross-section of expected stock returns. *Journal of Finance*, 47(2), 427–465.
- Fama, E. F., & French, K. R. (1996). The capm is wanted, dead or alive. *The Journal of Finance*, 51(5), 1947–1958.
- Fávero, L. P., & Belfiore, P. (2011). Cash flow, earnings ratio and stock returns in emerging global regions: Evidence from longitudinal data. *Global Economy and Finance Journal*, 4(1), 32–43.
- Fujiwara, Y., Souma, W., Murasato, H., & Yoon, H. (2006). Application of pca and random matrix theory to passive fund management. *Practical fruits of econophysics* (pp. 226–230). Springer.
- Giglio, S., & Xiu, D. (2021). Asset pricing with omitted factors. *Journal of Political Economy*, 129(7), 000–000.
- Han, Y., He, A., Rapach, D., & Zhou, G. (2018). What firm characteristics drive us stock returns. *Available at SSRN*.
- Hotelling, H. (1935). The most predictable criterion. *Journal of educational Psychology*, 26(2), 139.
- Jagannathan, R., & Ma, T. (2003). Risk reduction in large portfolios: Why imposing the wrong constraints helps. *The Journal of Finance*, 58(4), 1651–1683.
- Jiang, F., Tang, G., & Zhou, G. (2018). Firm characteristics and chinese stocks. *Journal of Management Science and Engineering*, 3(4), 259–283.
- Johnson, R., & Soenen, L. (2003). Indicators of successful companies. *European Management Journal*, 21(3), 364–369. [https://doi.org/https://doi.org/10.1016/S0263-2373\(03\)00050-1](https://doi.org/https://doi.org/10.1016/S0263-2373(03)00050-1)
- Jothimani, D., Shankar, R., & Yadav, S. S. (2017). A pca-dea framework for stock selection in indian stock market. *Journal of Modelling in Management*.
- Kogan, L., & Tian, M. H. (2012). Firm characteristics and empirical factor models: A data-mining experiment. *FRB International Finance discussion paper*, (1070).
- Light, N., Maslov, D., & Rytchkov, O. (2017). Aggregation of information about the cross section of stock returns: A latent variable approach. *The Review of Financial Studies*, 30(4), 1339–1381.

- Markowitz, H. M. (1952). Portfolio selection. *Journal of Finance*, (7), 77–91.
- Martani, D., & Khairurizka, R. (2009). The effect of financial ratios, firm size, and cash flow from operating activities in the interim report to the stock return. *Chinese Business Review*, 8(6), 44.
- McKnight, P. J., & Weir, C. (2009). Agency costs, corporate governance mechanisms and ownership structure in large uk publicly quoted companies: A panel data analysis. *The quarterly review of economics and finance*, 49(2), 139–158.
- Michaud, R. O. (1989). The markowitz optimization enigma: Is “optimized” optimal? *Financial analysts journal*, 45(1), 31–42.
- Miller, M. H. (1960). Portfolio selection: Efficient diversification of investments.
- Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11), 559–572.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- Subrahmanyam, A., & Titman, S. (1998). Feedback from stock prices to cash flows “(formerly called “real effects of financial market trading”).
- Suh, S., Song, W., & Lee, B.-S. (2014). A new method for forming asset pricing factors from firm characteristics. *Applied Economics*, 46(28), 3463–3482.
- Wold, H. (1975). Soft modelling by latent variables: The non-linear iterative partial least squares (nipals) approach. *Journal of Applied Probability*, 12(S1), 117–142.
- Zou, S., & Stan, S. (1998). The determinants of export performance: A review of the empirical literature between 1987 and 1997. *International marketing review*.

APPENDIX

Appendix A Firm Characteristics Description

- Market Capitalization (*mktcap*): calculated by the log of price per share times the number of outstanding shares. The definition is commonly applied in asset pricing literature (e.g., Fama and French, 1996).

$$market\ capitalization_t = \log(price_t \times number\ of\ outstanding\ shares_t)$$

- Equity to Invested Capital (*equity invcap*): defined as the value of common equity divided by the value of invested capital. Invested capital refers to the money raised from issuing equity and debts from bondholders.

$$equity\ to\ invested\ capital = \frac{common\ equity}{invested\ capital}$$

- Book-to-Market Ratio (*bm*): defined as the value of shareholders' equity (value of assets minus the value of liabilities) divided by the market capitalization (market price per share times the number of outstanding shares). Eugene and French (1992) show that the ratio is able to explain the variance of cross-sectional stock return.

$$book\ to\ market\ ratio = \frac{common\ shareholders'\ equity}{market\ capitalization}.$$

- Cash Flow Ratio (*pcf*): defined as the price divided by operating cash flow. As one of valuation metrics, cash flow ratio is preferred over price to earning (PE) since it wipe out the expense. Cash flow ratio explains stock return more significantly than earning estimators (Fávero and Belfiore, 2011).

$$price\ to\ cash\ flow\ ratio = \frac{share\ price}{operating\ cash\ flow\ per\ share}$$

- Accrual (*accrual*): defined as accruals divided by the value of average assets. The ratio is to measure the quality of the earnings (DuCharme et al., 2004). To investor or analysts, the ratio provides information about possibility changes. A company may change its accounting practice in order to improve its financial results. Therefore, the ratio needs to be evaluated over time to detect

the possible possibility changes and the intention of company to cover up its financially-stressed situation.

$$accrual = \frac{\Delta Working\ Capital - \Delta Cash - \Delta Depreciation}{\Delta Total\ Assets}.$$

- Cash Flow Margin (*cfm*): defined as the operating income divided by the sales. WIFR use *cfm* to describe the financial soundness, however, analysts consider this metric a profitability metric. The ratio shows the efficiency of a company using its revenue to generate profit. A low cash flow margin may also illustrate the incapability, caused by financial stress, of a company making money using the revenue.

$$cash\ flow\ margin = \frac{cash\ flow\ from\ operations}{net\ sales}$$

- Return on Assets (*roa*): defined as the company's net income divided by the value of total assets. The ratio indicates the effectiveness or efficiency of company in using its assets. Higher ratio indicates the profitability of the company (Johnson and Soenen, 2003).

$$return\ on\ assets = \frac{net\ income}{total\ assets}.$$

- Return on Equity (*roe*): defined as the company's net income divided by the value of equities. The ratio measures investment return. Clubb and Naffi (2007) find that, combining with book-to-market ratio, *roe* explains a significant portion of variation of the future cross-sectional stock return.

$$return\ on\ equity = \frac{net\ income}{total\ equity}.$$

- Current Ratio (*curr ratio*): defined as current asset divided by current liabilities. The ratio reveals the capability of a company to cover its short-term debt with its current assets. The ratio also has potential to show the risk of distress or default of companies within the same industry if the values are lower than the industrial average.

$$current\ ratio = \frac{current\ asset}{current\ liabilities}$$

- Debt/Asset Ratio (*debt to asset*): defined as the total debt divided by total asset. The ratio is also call leverage ratio, indicating the share of asset financed with debt. Since high leverage cause risks for repaying the debt, investors use this indicator to determine if the company is solvent. Cai and Zhang (2011) show that increasing leverage ratio has significant negative effect on stock return.

$$debt\ to\ asset\ ratio = \frac{total\ debt}{total\ asset}$$

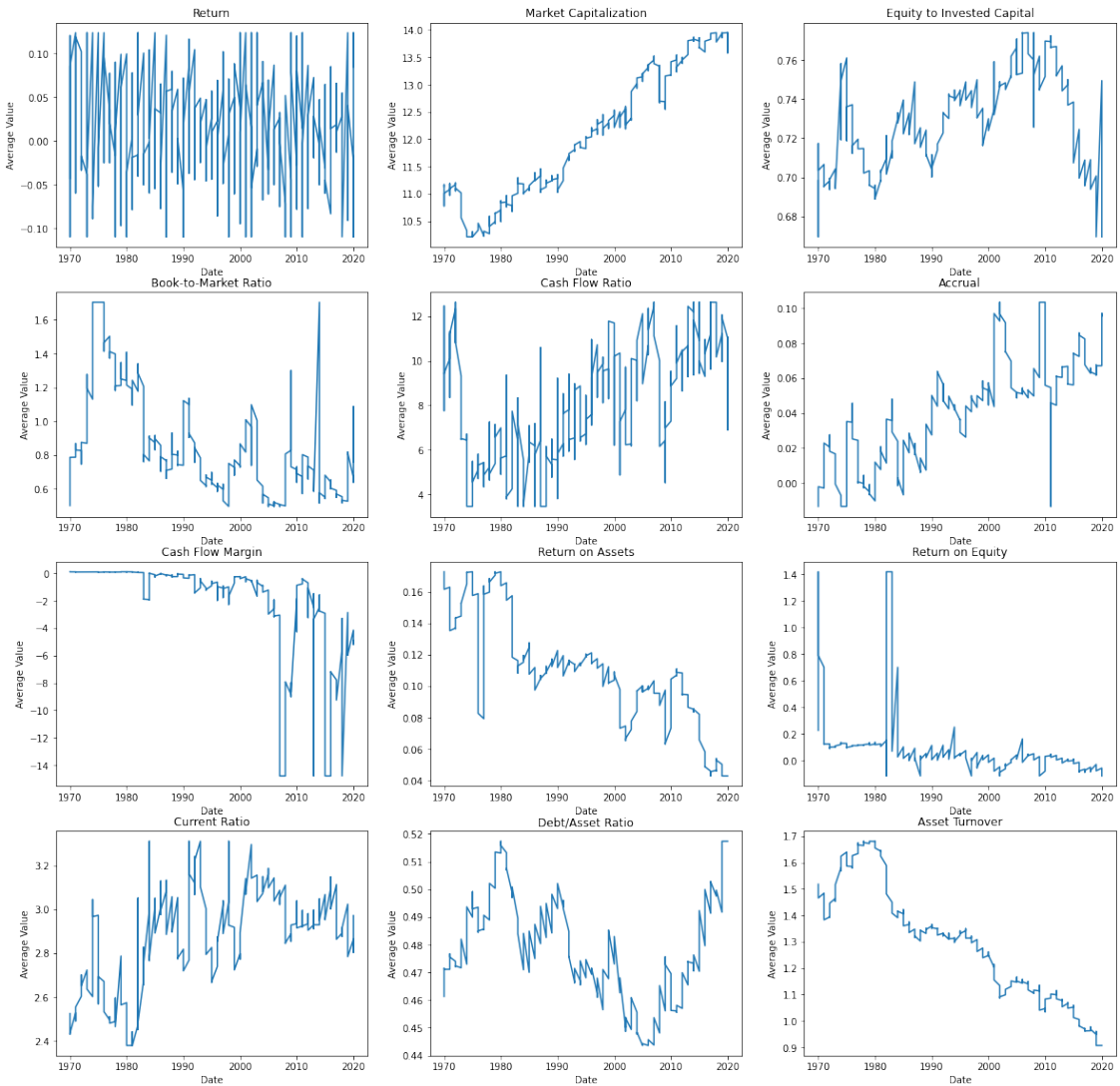
- Asset Turnover (*at turnover*) Ratio: defined as the total revenue divided by the average value of assets during the observation period. Asset turnover ratio measures the operating efficiency of a company by examining the utilization of its assets. Martani and Khairurizka's (2009) study shows that asset turnover rate contribute significantly to stock return across industries. Moreover, they also find that asset turnover rate are cointegrated with stock return at I(1) level.

$$asset\ turnover = \frac{(sales)revenue}{\frac{beginning\ asset+ending\ asset}{2}}$$

Mean and Volatility of Firm Characteristics (Non-standardized)

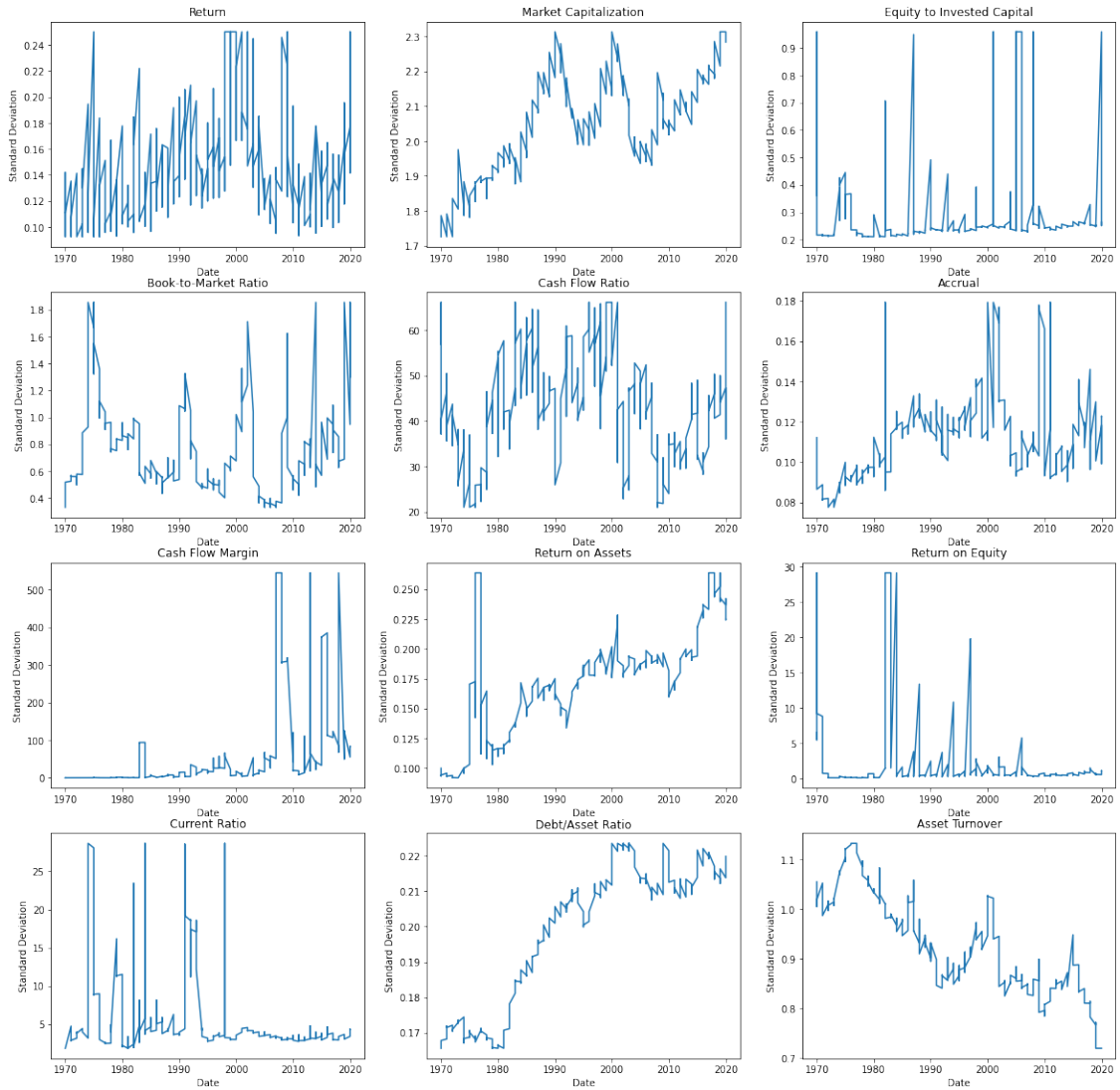
This figure displays the monthly cross-sectional means of returns and firm characteristics Market Capitalization, Equity to Invested Capital Ratio, Book-to-Market Ratio, Cash Flow Ratio, Accrual, Cash Flow Margin, Return on Assets, Return on Equity, Current Ratio, Debt/Asset Ratio, Asset Turnover Ratio as well as Return, from January 1970 to December 2020. The return data is downloaded from Compustat and the firm characteristics are from WIFR. Market capitalization is calculated by adjusted price times the outstanding shares. To eliminate the extreme value impacts, the characteristics are winsorized at level 2.5% and 97.5%.

Figure 4.1: Cross-sectional Mean Summary Statistics, Return & Firm Characteristics, U.S. Stock 1970-2020



This figure displays the monthly cross-sectional volatility of returns and firm characteristics Market Capitalization, Equity to Invested Capital Ratio, Book-to-Market Ratio, Cash Flow Ratio, Accrual, Cash Flow Margin, Return on Assets, Return on Equity, Current Ratio, Debt/Asset Ratio, Asset Turnover Ratio as well as Return, from January 1970 to December 2020. The return data is downloaded from Compustat and the firm characteristics are from WIFR. Market capitalization is calculated by adjusted price times the outstanding shares. To eliminate the extreme value impacts, the characteristics are winsorized at level 2.5% and 97.5%.

Figure 4.2: Cross-sectional Volatility Summary Statistics, Return & Firm Characteristics, U.S. Stock 1970-2020



Appendix B Comprehensive Tables

In- and Out-of-Sample Performance - Base Case

The following table demonstrates the In- and Out-of-Sample performances of the base case with characteristics, Market Capitalization(*mktcap*), Equity to Invested Capital Ratio(*equityinvcap*), Book-to-Market Ratio(*bm*), Cash Flow Ratio(*pcf*), Accrual(*accrual*), Cash Flow Margin(*cfm*), Return on Assets(*roa*), Return on Equity(*roe*), Current Ratio(*currratio*), Debt/Asset Ratio(*debttoasset*), Asset Turnover Ratio(*atturn*). The two sub-samples are equally divided. The in-sample starts from January 1970 to December 1995, and the out-of-sample starts from January 1996 to December 2020. We display statistics of in-sample, however, in the out-of-sample, the coefficient of each characteristic is not re-estimated. Instead, we use the parameters of in-sample to forecast asset weights and return metrics in out-of-sample and assess the performance (shown in column Opt. Fcst.). We also present the valued-weighted portfolio for both sub-samples. The risk-free rate is split for the two sub-samples as well. The average (annualized) risk-free rates are 0.017 and 0.005 for in-sample and out-of-sample, respectively.

Table 4.8: Comprehensive Optimized Portfolio Performance, In- & Out-of-Sample Experiments, U.S. Stocks (In-Sample: 1970-1995; Out-of-Sample: 1996-2020)

	All Stocks			
	In-Sample		Out-of-Sample	
	Val.Weighted	Opt.	Val.Weighted	Opt. Fcst.
θ_{mktcap}	-	2.024 (1.609)	-	2.024
$\theta_{equityinvcap}$	-	89.20*** (1.109)	-	89.20
θ_{bm}	-	10.20*** (2.167)	-	10.20
θ_{pcf}	-	26.36 *** (1.376)	-	26.36
$\theta_{accrual}$	-	-24.38*** (2.030)	-	-24.38
θ_{cfm}	-	164.10*** (1.609)	-	164.10
θ_{roa}	-	2.739** (1.194)	-	2.739
θ_{roe}	-	79.44*** (1.492)	-	79.44
$\theta_{currratio}$	-	-56.17*** (1.365)	-	-56.17
$\theta_{debttoasset}$	-	-166.95*** (2.152)	-	-166.95
θ_{atturn}	-	-47.85*** (1.649)	-	-47.85
$ w_i \times 100$	0.025	0.051	0.029	0.046
$\max w_i \times 100$	8.630	3.921	7.564	4.153
$\min w_i \times 100$	0.001*	0.000	0.001*	0.000
\bar{r}	0.065	0.091	0.083	0.083
$\sigma(r)$	0.125	0.161	0.210	0.207
<i>Sharpe Ratio</i>	0.384	0.460	0.371	0.377

* The minimum weight is too small, we only approximate it to 0.001

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

In- and Out-of-Sample Performance - PCA Cases

The following tables display the In- and Out-of-Sample performance of the PCA cases with from two to eight principal components parameterized for asset weights. The two sub-samples are equally divided. The in-sample starts from January 1970 to December 1995, and the out-of-sample starts from January 1996 to December 2020. We display statistics of in-sample, however, in the out-of-sample, the coefficient of each characteristic is not re-estimated. Instead, we use the parameters of in-sample to forecast asset weights and return metrics in out-of-sample and assess the performance (shown in column $pc = n(n = 2, \dots, 8)$. Fcst.). We also present the valued-weighted portfolio for both sub-samples. The risk-free rate is split for the two sub-samples as well. The average (annualized) risk-free rates are 0.017 and 0.005 for in-sample and out-of-sample, respectively.

Table continued on next page...

Table 4.9: Comprehensive Principal Components Optimized Portfolio Performance, In- & Out-of-Sample Experiments, U.S. Stocks (In-Sample: 1970-1995; Out-of-Sample: 1996-2020)

	All Stocks			
	In-Sample		Out-of-Sample	
	Val.Weighted	$pc = 2.$	Val.Weighted	$pc = 2$ Fcst.
θ_{pc1}	-	97.21*** (1.158)	-	97.21
θ_{pc2}	-	-297.94*** (1.108)	-	-297.94
$ w_i \times 100$	0.025	0.052	0.029	0.053
$\max w_i \times 100$	8.630	3.639	7.564	7.308
$\min w_i \times 100$	0.001*	0.000	0.001*	0.000
\bar{r}	0.065	0.091	0.083	0.085
$\sigma(r)$	0.125	0.168	0.210	0.229
<i>Sharpe Ratio</i>	0.384	0.440	0.371	0.349
	In-Sample		Out-of-Sample	
	Val.Weighted	$pc = 3.$	Val.Weighted	$pc = 3$ Fcst.
θ_{pc1}	-	104.43*** (0.899)	-	104.43
θ_{pc2}	-	-262.83*** (1.003)	-	-262.83
θ_{pc3}	-	22.25*** (1.003)	-	22.25
$ w_i \times 100$	0.025	0.052	0.029	0.053
$\max w_i \times 100$	8.630	3.747	7.564	4.280
$\min w_i \times 100$	0.001*	0.000	0.001*	0.000
\bar{r}	0.065	0.087	0.083	0.086
$\sigma(r)$	0.125	0.167	0.210	0.230
<i>Sharpe Ratio</i>	0.384	0.419	0.371	0.352

* The minimum weight is too small, we only approximate it to 0.001

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Table continued from the previous page.

	All Stocks			
	In-Sample		Out-of-Sample	
	Val.Weighted	$pc = 4.$	Val.Weighted	$pc = 4$ Fcst.
θ_{pc1}	-	102.97*** (0.740)	-	102.97
θ_{pc2}	-	-262.03*** (0.822)	-	-262.03
θ_{pc3}	-	16.37*** (0.607)	-	16.37
θ_{pc4}	-	16.37*** (0.607)	-	16.37
$ w_i \times 100$	0.025	0.051	0.029	0.050
$\max w_i \times 100$	8.630	2.499	7.564	2.791
$\min w_i \times 100$	0.001*	0.000	0.001*	0.000
\bar{r}	0.065	0.086	0.083	0.076
$\sigma(r)$	0.125	0.162	0.210	0.210
<i>Sharpe Ratio</i>	0.384	0.426	0.371	0.338
	In-Sample		Out-of-Sample	
	Val.Weighted	$pc = 5.$	Val.Weighted	$pc = 5$ Fcst.
	Val.Weighted	$pc = 5.$	Val.Weighted	$pc = 5$ Fcst.
θ_{pc1}	-	4.162*** (0.470)	-	4.162
θ_{pc2}	-	-57.38*** (0.537)	-	-57.38
θ_{pc3}	-	49.16*** (0.614)	-	49.16
θ_{pc4}	-	56.89*** (0.606)	-	56.88
θ_{pc5}	-	113.06*** (0.724)	-	113.06
$ w_i \times 100$	0.025	0.050	0.029	0.049
$\max w_i \times 100$	8.630	2.248	7.564	1.890
$\min w_i \times 100$	0.001*	0.000	0.001*	0.000
\bar{r}	0.065	0.087	0.083	0.080
$\sigma(r)$	0.125	0.163	0.210	0.210
<i>Sharpe Ratio</i>	0.384	0.423	0.371	0.357

Table continued on next page...

Table continued from the previous page.

	All Stocks			
	In-Sample		Out-of-Sample	
	Val.Weighted	$pc = 6.$	Val.Weighted	$pc = 6$ Fcst.
θ_{pc1}	-	-10.20*** (0.614)	-	-10.20
θ_{pc2}	-	-25.95*** (0.706)	-	-25.95
θ_{pc3}	-	47.04*** (0.704)	-	47.04
θ_{pc4}	-	55.82*** (0.726)	-	55.82
θ_{pc5}	-	-33.30*** (0.859)	-	-33.30
θ_{pc6}	-	114.06*** (0.709)	-	114.06
$ w_i \times 100$	0.025	0.050	0.029	0.048
$\max w_i \times 100$	8.630	2.679	7.564	2.759
$\min w_i \times 100$	0.001*	0.000	0.001*	0.000
\bar{r}	0.065	0.091	0.083	0.070
$\sigma(r)$	0.125	0.289	0.210	0.210
<i>Sharpe Ratio</i>	0.384	0.442	0.371	0.310

Table continued on next page...

Table continued from the previous page.

	All Stocks			
	In-Sample		Out-of-Sample	
	Val.Weighted	$pc = 7.$	Val.Weighted	$pc = 7$ Fcst.
θ_{pc1}	-	-47.98*** (0.562)	-	-47.98
θ_{pc2}	-	67.45*** (0.700)	-	67.45
θ_{pc3}	-	22.04*** (0.871)	-	22.04
θ_{pc4}	-	38.79*** (0.776)	-	38.79
θ_{pc5}	-	-82.38*** (0.810)	-	-82.38
θ_{pc6}	-	55.43*** (0.805)	-	55.43
θ_{pc7}	-	19.10*** (0.787)	-	19.10
$ w_i \times 100$	0.025	0.051	0.029	0.050
$\max w_i \times 100$	8.630	2.317	7.564	1.832
$\min w_i \times 100$	0.001*	0.000	0.001*	0.000
\bar{r}	0.065	0.084	0.083	0.090
$\sigma(r)$	0.125	0.166	0.210	0.220
<i>Sharpe Ratio</i>	0.384	0.403	0.371	0.386

Table continued on next page...

Table continued from the previous page.

	All Stocks			
	In-Sample		Out-of-Sample	
	Val.Weighted	$pc = 8$.	Val.Weighted	$pc = 8$ Fcst.
θ_{pc1}	-	54.67*** (0.565)	-	54.67
θ_{pc2}	-	31.42*** (0.704)	-	31.42
θ_{pc3}	-	-2.27*** (0.828)	-	-2.27
θ_{pc4}	-	-20.83*** (0.653)	-	-20.83
θ_{pc5}	-	7.53*** (0.550)	-	7.53
θ_{pc6}	-	-33.64*** (1.240)	-	-33.64
θ_{pc7}	-	84.23*** (1.293)	-	84.23
θ_{pc8}	-	65.00*** (0.650)	-	65.00
$ w_i \times 100$	0.025	0.049	0.029	0.047
$\max w_i \times 100$	8.630	2.320	7.564	2.714
$\min w_i \times 100$	0.001※	0.000	0.001※	0.000
\bar{r}	0.065	0.092	0.083	0.096
$\sigma(r)$	0.125	0.166	0.210	0.200
<i>Sharpe Ratio</i>	0.384	0.452	0.371	0.455

※ The minimum weight is too small, we only approximate it to 0.001

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Risk Aversion - Base Case

This Table displays the estimate of the coefficients for the eleven firm characteristics: *mktcap*, *bm*, *pcf*, *roa*, *roe*, *accrual*, *equityinvcap*, *atturn*, *cfm*, *debtasset* and *currratio*, specified in equation (1), under different risk aversion coefficients (of $\gamma = 1, 3, 5, 7, 9$, respectively). In the upper few rows, we show the coefficients with the corresponding standard error in parentheses for each specification. In the middle few rows, we show the average weights ($|w_i|$) as well as the min and max in the three portfolios. In the bottom few rows, we present the average return, standard deviation and Sharpe Ratio ($\bar{r}, \sigma(r)$ and *Sharpe Ratio*, respectively). The average risk-free rate across the sample is 0.012 (annualized).

Table 4.10: Comprehensive Optimized Portfolio Performance, Different Risk Aversion, U.S. Stocks, 1970-2020 (Risk Aversion Coefficient $\gamma = 1, 3, 5, 7, 9$)

	All Stocks				
	$\gamma = 1$	$\gamma = 3$	$\gamma = 5$	$\gamma = 7$	$\gamma = 9$
θ_{mktcap}	0.00 (0.021)	-175.70*** (3.305)	82.49*** (4.397)	-10051.10*** (51.897)	-406.10*** (0.751)
θ_{bm}	0.00 (0.021)	585.48*** (3.732)	-8.56* (4.327)	-580.02*** (4.296)	85.51*** (0.578)
θ_{pcf}	0.00 (0.021)	135.92*** (2.171)	303.61*** (7.059)	-4701.94*** (24.855)	180.93*** (0.460)
θ_{roa}	0.00 (0.021)	-198.82*** (1.727)	4.70 (3.253)	-1577.19*** (10.898)	-606.33*** (0.482)
θ_{roe}	0.00 (0.021)	543.32*** (2.928)	120.30*** (7.056)	1497.80*** (7.389)	110.86*** (0.278)
$\theta_{accrual}$	0.00 (0.021)	-60.20*** (2.116)	82.64*** (2.835)	-1492.13*** (5.659)	420.74*** (1.204)
$\theta_{equityinvcap}$	0.00 (0.021)	-65.91*** (3.311)	-165.90*** (2.323)	2995.14*** (18.261)	292.91*** (0.776)
θ_{atturn}	0.00 (0.021)	80.71*** (2.063)	250.31*** (15.022)	-3475.06*** (16.738)	102.06*** (0.308)
θ_{cfm}	0.00 (0.021)	526.42*** (2.506)	-274.68*** (15.022)	-854.49*** (6.355)	428.27*** (0.624)
$\theta_{debtasset}$	0.00 (0.021)	-218.79*** (2.430)	-420.87*** (9.636)	-97.66*** (3.985)	110.78*** (0.270)
$\theta_{currratio}$	0.00 (0.021)	79.43*** (1.552)	-304.64*** (9.854)	-2532.78*** (12.368)	-305.16*** (0.530)
$ w_i \times 100$	0.046	0.056	0.049	0.056	0.058
$\max w_i \times 100$	0.076	4.674	4.731	3.841	6.212
$\min w_i \times 100$	0.037	0.000	0.000	0.000	0.000
\bar{r}	0.178	-0.046	0.183	0.081	0.056
$\sigma(r)$	0.171	0.190	0.180	0.198	0.192
<i>Sharpe Ratio</i>	0.971	-0.313	0.950	0.347	0.231

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Risk Aversion - PCA Cases

This Table displays the estimates of the coefficients for the pc-optimized portfolios with from two to eight principal components. For each specification, we present the estimated results under different risk aversion coefficients (of $\gamma = 1, 3, 5, 7, 9$, respectively). In the upper few rows, we show the coefficients with the corresponding standard error in parentheses for each specification. In the middle few rows, we show the average weights ($|w_i|$) as well as the min and max in the three portfolios. In the bottom few rows, we present the average return, standard deviation and Sharpe Ratio ($\bar{r}, \sigma(r)$ and *Sharpe Ratio*, respectively). The average risk-free rate across the sample is 0.012 (annualized).

Table 4.11: Comprehensive Principal Components Optimized Portfolio Performance, Different Risk Aversion, U.S. Stocks, 1970-2020 (Risk Aversion Coefficient $\gamma = 1, 3, 5, 7, 9$)

	All Stocks <i>pc = 2</i>				
	$\gamma = 1$	$\gamma = 3$	$\gamma = 5$	$\gamma = 7$	$\gamma = 9$
θ_{pc1}	0.00 (0.021)	151.17*** (1.811)	121.73*** (1.141)	73.75*** (0.719)	61.87*** (0.562)
θ_{pc2}	0.00 (0.021)	1327.27*** (1.158)	-379.51*** (0.758)	-277.47*** (0.535)	-240.01*** (0.487)
$ w_i \times 100$	0.076	0.056	0.054	0.055	0.055
$\max w_i \times 100$	0.045	4.674	7.044	7.153	7.216
$\min w_i \times 100$	0.037	0.000	0.000	0.000	0.000
\bar{r}	0.174	0.184	0.167	0.162	0.162
$\sigma(r)$	0.171	0.199	0.200	0.200	0.200
<i>Sharpe Ratio</i>	0.944	0.867	0.775	0.750	0.750
	All Stocks <i>pc = 3</i>				
	$\gamma = 1$	$\gamma = 3$	$\gamma = 5$	$\gamma = 7$	$\gamma = 9$
θ_{pc1}	0.00 (0.021)	363.85*** (1.448)	133.10*** (0.645)	76.94*** (0.648)	-10386.88*** (0.316)
θ_{pc2}	0.00 (0.021)	-840.13*** (1.184)	-333.63*** (0.609)	-250.34*** (0.526)	14370.29*** (.497)
θ_{pc3}	0.00 (0.021)	72.83*** (1.346)	41.93*** (0.681)	9.92*** (0.580)	-5019.81*** (0.223)
$ w_i \times 100$	0.076	0.054	0.054	0.054	0.054
$\max w_i \times 100$	0.045	6.741	7.044	6.870	3.794
$\min w_i \times 100$	0.037	0.000	0.000	0.000	0.000
\bar{r}	0.174	0.157	0.167	0.162	0.171
$\sigma(r)$	0.171	0.197	0.200	0.199	0.196
<i>Sharpe Ratio</i>	0.944	0.733	0.775	0.756	0.811

Table continued on next page...

Table continued from the previous page.

	All Stocks				
	$\gamma = 1$	$\gamma = 3$	$pc = 4$ $\gamma = 5$	$\gamma = 7$	$\gamma = 9$
θ_{pc1}	0.00 (0.021)	-92.30*** (1.016)	-126.98*** (0.968)	-15886.02*** (0.281)	-115.69*** (0.201)
θ_{pc2}	0.00 (0.021)	-554.79*** (0.958)	13.16*** (0.581)	4815.26*** (0.321)	40.15*** (0.315)
θ_{pc3}	0.00 (0.021)	-164.70*** (1.071)	170.65*** (0.658)	-11711.83*** (0.359)	45.47*** (0.261)
θ_{pc4}	0.00 (0.021)	311.79*** (1.128)	156.72*** (0.582)	-12155.85*** (0.374)	-91.26*** (0.273)
$ w_i \times 100$	0.076	0.054	0.052	0.053	0.054
$\max w_i \times 100$	0.045	4.880	3.000	2.968	2.752
$\min w_i \times 100$	0.037	0.000	0.000	0.000	0.000
\bar{r}	0.174	0.153	0.158	0.162	0.193
$\sigma(r)$	0.171	0.199	0.189	0.199	0.199
<i>Sharpe Ratio</i>	0.944	0.710	0.772	0.779	0.910

	All Stocks				
	$\gamma = 1$	$\gamma = 3$	$pc = 5$ $\gamma = 5$	$\gamma = 7$	$\gamma = 9$
θ_{pc1}	0.00 (0.021)	126.48*** (1.001)	72.89*** (0.429)	601.94*** (0.160)	-40.96*** (0.351)
θ_{pc2}	0.00 (0.021)	-204.98*** (0.816)	-36.09*** (0.419)	3428.69*** (0.228)	-30.62*** (0.286)
θ_{pc3}	0.00 (0.021)	322.36*** (1.081)	70.73*** (0.555)	-4087.68*** (0.225)	10.03*** (0.311)
θ_{pc4}	0.00 (0.021)	16.49*** (1.056)	53.16*** (0.546)	-1933.95*** (0.211)	97.56*** (0.319)
θ_{pc5}	0.00 (0.021)	284.33*** (1.001)	123.51*** (0.466)	-2548.56*** (0.241)	53.82*** (0.327)
$ w_i \times 100$	0.076	0.052	0.051	0.053	0.052
$\max w_i \times 100$	0.045	3.413	3.648	4.064	2.653
$\min w_i \times 100$	0.037	0.000	0.000	0.000	0.000
\bar{r}	0.174	0.165	0.164	0.172	0.153
$\sigma(r)$	0.171	0.187	0.184	0.198	0.189
<i>Sharpe Ratio</i>	0.944	0.817	0.826	0.806	0.746

Table continued on next page...

Table continued from the previous page.

	All Stocks				
	$\gamma = 1$	$\gamma = 3$	$pc = 6$ $\gamma = 5$	$\gamma = 7$	$\gamma = 9$
θ_{pc1}	0.00 (0.021)	-267.53*** (0.848)	-42.52*** (0.583)	1681.94*** (0.419)	-260.43*** (0.725)
θ_{pc2}	0.00 (0.021)	210.87*** (1.043)	-38.87*** (0.560)	-15959.41*** (0.590)	-37.53*** (0.306)
θ_{pc3}	0.00 (0.021)	14.81*** (1.146)	69.60*** (0.590)	12442.51*** (0.916)	-207.7***1 (0.700)
θ_{pc4}	0.00 (0.021)	-12.80*** (1.170)	77.67*** (0.581)	5474.78*** (0.270)	197.13*** (0.769)
θ_{pc5}	0.00 (0.021)	-235.22*** (1.181)	-48.44*** (0.618)	-11750.38*** (0.464)	-95.84*** (0.484)
θ_{pc6}	0.00 (0.021)	240.54*** (1.448)	134.45*** (0.586)	-14731.39*** (0.723)	63.53*** (0.352)
$ w_i \times 100$	0.076	0.052	0.051	0.052	0.053
$\max w_i \times 100$	0.045	2.665	2.679	2.656	2.846
$\min w_i \times 100$	0.037	0.000	0.000	0.000	0.000
\bar{r}	0.174	0.153	0.160	0.153	0.153
$\sigma(r)$	0.171	0.198	0.190	0.196	0.197
<i>Sharpe Ratio</i>	0.944	0.785	0.779	0.786	0.752

Table continued on next page...

Table continued from the previous page.

	All Stocks				
	$\gamma = 1$	$\gamma = 3$	$pc = 7$ $\gamma = 5$	$\gamma = 7$	$\gamma = 9$
θ_{pc1}	0.00 (0.021)	-257.53*** (0.947)	-84.06*** (0.487)	-142.85*** (0.692)	7.083*** (0.305)
θ_{pc2}	0.00 (0.021)	20.43*** (0.875)	82.22*** (0.422)	25.07*** (0.298)	-24.12*** (0.291)
θ_{pc3}	0.00 (0.021)	171.05*** (1.193)	17.66*** (0.556)	-169.79*** (0.737)	27.71*** (0.300)
θ_{pc4}	0.00 (0.021)	181.02*** (1.053)	25.06*** (0.597)	-398.33*** (1.232)	4.62*** (0.346)
θ_{pc5}	0.00 (0.021)	-287.25*** (0.981)	-121.19*** (0.505)	319.84*** (1.123)	-13.84*** (0.276)
θ_{pc6}	0.00 (0.021)	37.00*** (1.410)	65.63*** (0.676)	194.40*** (0.818)	54.95*** (0.285)
θ_{pc7}	0.00 (0.021)	-201.56*** (1.061)	34.12*** (0.474)	423.95*** (1.527)	-24.49*** (0.255)
$ w_i \times 100$	0.076	0.052	0.052	0.051	0.051
$\max w_i \times 100$	0.045	2.291	2.317	2.712	2.687
$\min w_i \times 100$	0.037	0.000	0.000	0.000	0.000
\bar{r}	0.174	0.174	0.165	0.164	0.164
$\sigma(r)$	0.171	0.192	0.196	0.188	0.188
<i>Sharpe Ratio</i>	0.944	0.841	0.781	0.807	0.808

Table continued on next page...

Table continued from the previous page.

	All Stocks				
	$\gamma = 1$	$\gamma = 3$	$pc = 8$ $\gamma = 5$	$\gamma = 7$	$\gamma = 9$
θ_{pc1}	0.00 (0.021)	33.92*** (0.846)	90.41*** (0.353)	18.70*** (0.203)	11.21*** (0.323)
θ_{pc2}	0.00 (0.021)	-56.06*** (0.837)	26.85*** (0.467)	211.01*** (0.726)	-5.73*** (0.336)
θ_{pc3}	0.00 (0.021)	119.26*** (0.869)	-33.17*** (0.537)	351.82*** (1.082)	18.58*** (0.365)
θ_{pc4}	0.00 (0.021)	-215.31*** (0.838)	-24.51*** (0.499)	16.00*** (0.519)	-65.62*** (0.345)
θ_{pc5}	0.00 (0.021)	-10.29*** (0.863)	31.44*** (0.340)	-225.95*** (0.791)	7.05*** (0.375)
θ_{pc6}	0.00 (0.021)	266.01*** (0.911)	-124.16*** (0.744)	149.17*** (0.565)	45.14*** (0.451)
θ_{pc7}	0.00 (0.021)	-196.90*** (0.730)	182.29*** (0.720)	-209.74*** (0.712)	-31.48*** (0.413)
θ_{pc8}	0.00 (0.021)	74.27*** (0.589)	119.42*** (0.396)	32.6***2 (0.249)	15.06*** (0.358)
$ w_i \times 100$	0.076	0.051	0.049	0.051	0.051
$\max w_i \times 100$	0.045	2.387	3.223	2.459	2.270
$\min w_i \times 100$	0.037	0.000	0.000	0.000	0.000
\bar{r}	0.174	0.173	0.178	0.163	0.170
$\sigma(r)$	0.171	0.190	0.184	0.188	0.191
<i>Sharpe Ratio</i>	0.944	0.850	0.902	0.805	0.828

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Appendix C Graphs

Figure 4.3: Optimized Portfolio Cumulative Return, Optimized vs. Benchmarks, U.S. Stocks 1970-2020

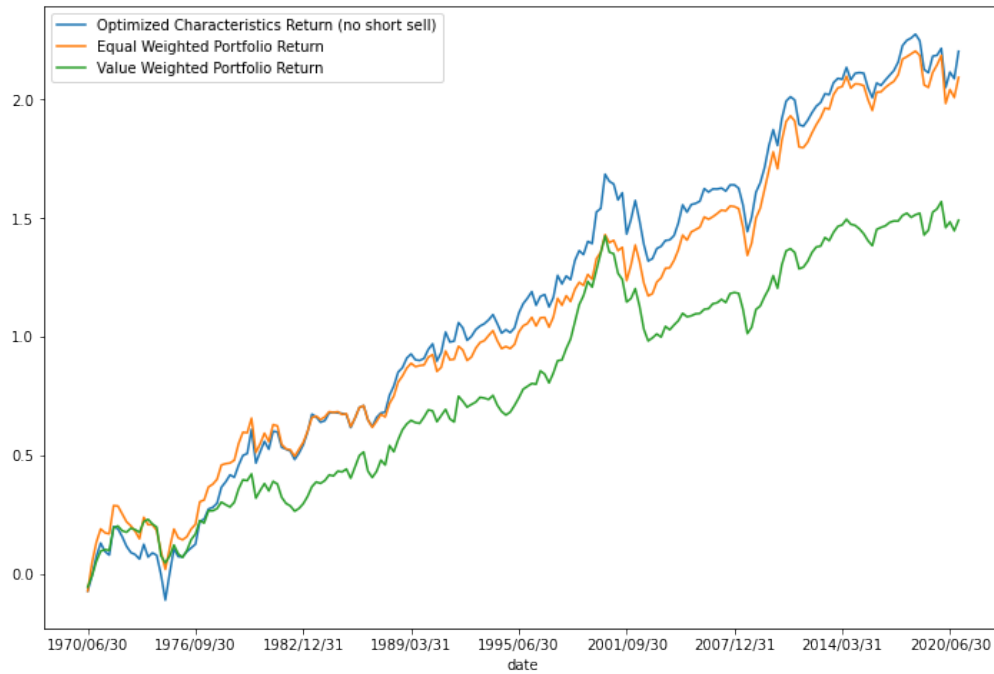


Figure 4.4: Principal Components Optimized Portfolio Cumulative Return, PC Optimized vs. Benchmarks, U.S. Stocks 1970-2020

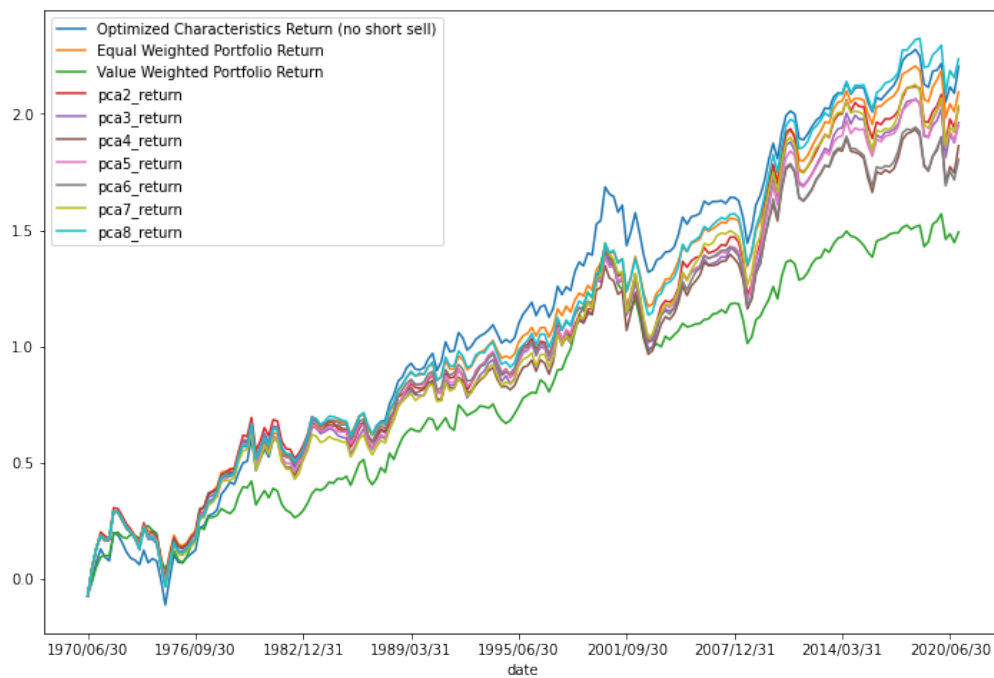


Figure 4.5: Optimized Portfolio Cumulative Return, All vs. Top500 Stocks, U.S. Stocks 1970-2020

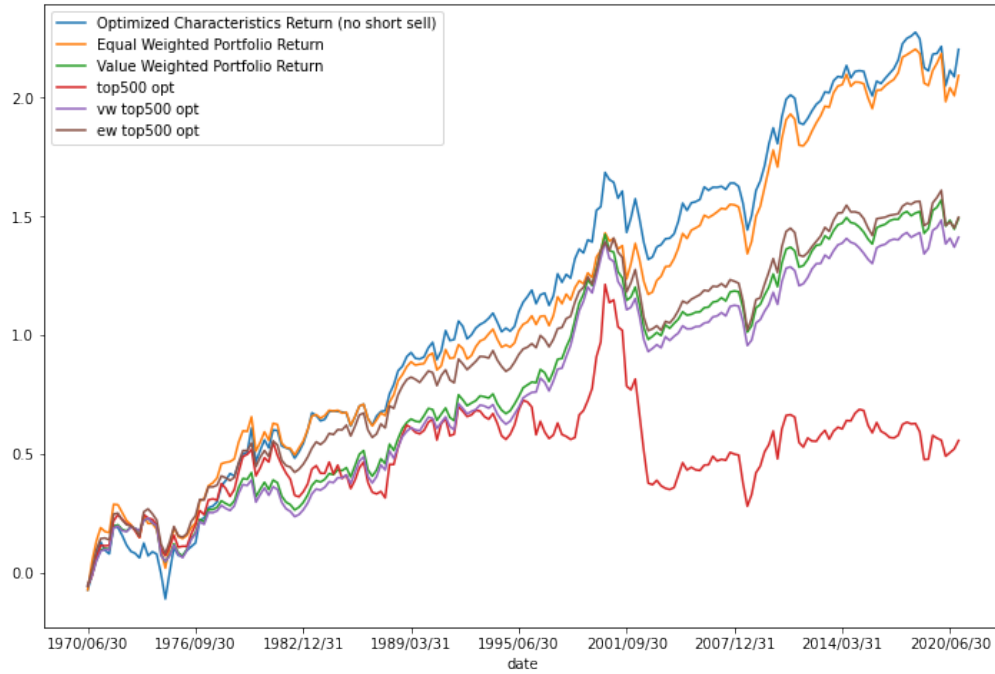


Figure 4.6: Principal Components Optimized Portfolio Cumulative Return, Top500 Stocks, U.S. Stocks 1970-2020

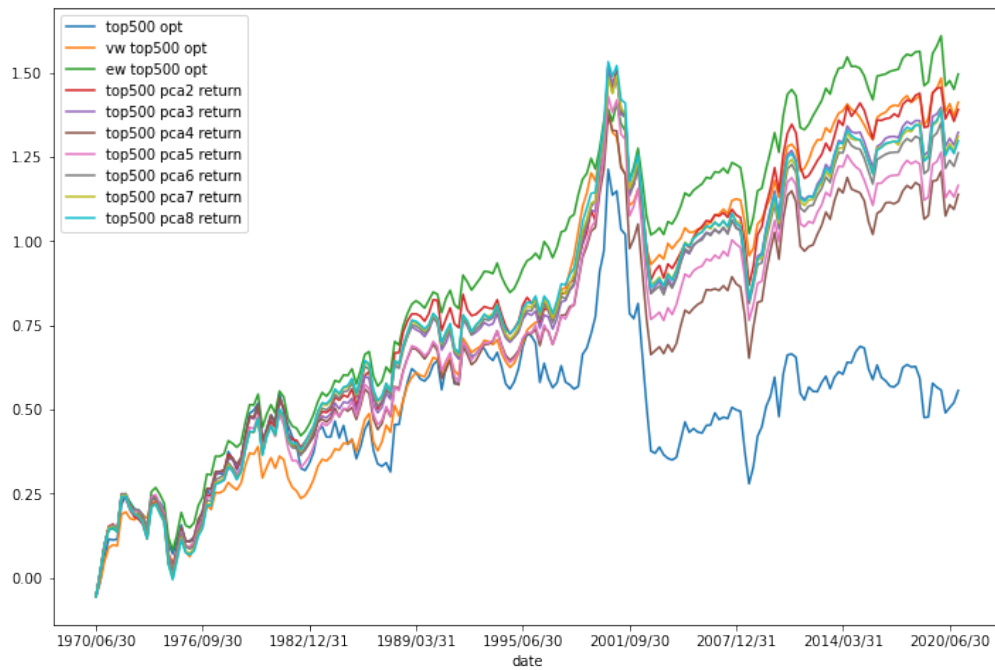
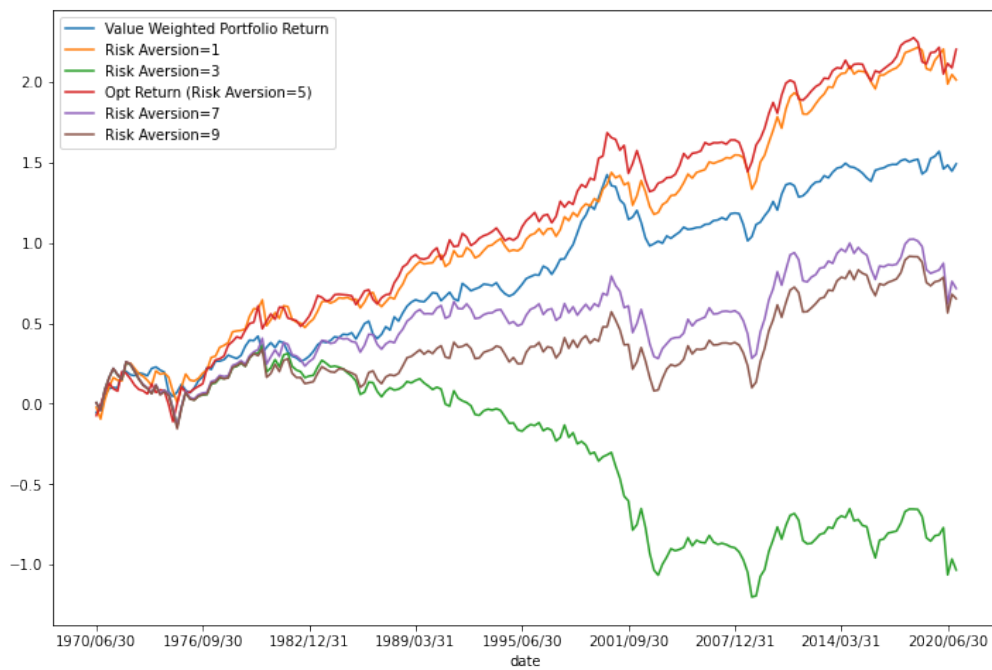


Figure 4.7: Optimized Portfolio Cumulative Return, Different Risk Aversion, U.S. Stocks 1970-2020



Appendix D Codes

```
1 ### Building some useful tools
2
3 import pandas as pd
4 import numpy as np
5 import matplotlib.pyplot as plt
6 from sklearn.decomposition import PCA
7 import os
8
9 import scipy
10 from scipy import optimize
11
12 def Scale(y, c=True, sc=True):
13
14     x = y.copy()
15
16     if c:
17         x -= x.mean()
18     if sc and c:
19         x /= x.std()
20     elif sc:
21         x /= np.sqrt(x.pow(2).sum().div(x.count() - 1))
22     return x
23
24 def check_shape(df1, df2):
25     return df1.shape == df2.shape
26
27 def delete_company_with_na(df):
28
29     have_null = df.columns[df.isna().any()]
30     have_null.append(df.columns[df.isna().any()])
31     result_df = df.drop(columns=have_null)
32
33     return result_df
34
35 def reshape_dataframe(main_df, reshaped_df):
36
37     n1 = main_df.shape[0]
38     n2 = reshaped_df.shape[0]
39     df = reshaped_df.iloc[(n2-n1):,:]
40     return df
41
42 def descriptive_statistics(df):
43
44     d = {
45         "Mean" : df.mean(axis=1),
46         "St. Dev" : df.std(axis=1)
47     }
48
49     df = pd.DataFrame(d, index=df.index)
50     return df
51
52 def q4_rolling_return(df):
53
54     q4 = df.rolling(4).sum().dropna()
```

```

55     return q4
56
57 def m12_rolling_return(df):
58
59     m12 = df.rolling(12).sum().dropna()
60     return m12
61
62 def find_common_firms(l1, l2):
63
64     s1 = set(l1)
65     s2 = set(l2)
66
67     s_result = s1.intersection(s2)
68
69     return list(s_result)
70
71 def short_sell_constraints(df):
72
73     no_short_sell_weight = np.zeros(df.shape)
74     z_w = np.zeros(df.shape)
75     rows = df.shape[0]
76     columns = df.shape[1]
77
78     for i in range(rows):
79         for j in range(columns):
80             z_w[i][j] = max(df.iloc[i,j], 0)
81
82     for i in range(rows):
83         for j in range(columns):
84             no_short_sell_weight[i][j] = z_w[i][j]/z_w[i].sum()
85
86     no_short_sell_df = pd.DataFrame(no_short_sell_weight, index=df.
index, columns=df.columns)
87     return no_short_sell_df
88
89 def value_weights(df):
90     w = np.zeros(df.shape)
91
92     for i in range(df.shape[0]):
93         for j in range(df.shape[1]):
94             w[i][j] = df.iloc[i, j]/df.iloc[i,:].sum()
95
96     value_weight = pd.DataFrame(w, index=df.index, columns=df.
columns)
97
98     return value_weight
99
100 def sharpe_ratio(ret_series, rf):
101     SR = (ret_series - rf).mean()/ret_series.std()
102     return SR
103
104
105 def make_year_month(df):
106     df['year'] = pd.DatetimeIndex(df.index).year
107     df['month'] = pd.DatetimeIndex(df.index).month
108     return df

```

```

109
110 def rebalancing(company_df, characteristics_df_list, year_list):
111
112     rebalanced_universe = pd.DataFrame(index=year_list, columns=['
Companies', 'Number of Companies'])
113
114     for year in year_list:
115         company_set = set(list(delete_company_with_na(company_df[
company_df['year'] == year]).columns[:-2]))
116         print("{} companies in {}".format(len(company_set), year))
117         print("-----")
118
119         for characteristics_df in characteristics_df_list:
120             characteristics_df = delete_company_with_na(
characteristics_df[characteristics_df['year'] == year])
121             characteristics_set = set(list(characteristics_df.
columns[:-2]))
122             print("{} companies for the char in {}".format(len(
characteristics_set), year))
123
124             company_set.intersection_update(characteristics_set)
125             print("Finally, {} companies in {} after rebalancing".
format(len(company_set), year))
126
127             rebalanced_universe.loc[year, 'Companies'] = company_set
128             number_of_companies = len(company_set)
129             rebalanced_universe.loc[year, 'Number of Companies'] =
number_of_companies
130
131     return rebalanced_universe
132
133 def PPS_base(x, wb, nt, ret, mktcap, bm, roa, roe, accrual, eqinv,
atturn, cfm, curr, da, pcf, rr):
134     wi = wb + nt * (x[0] * mktcap + x[1] * bm + x[2] * roa + x[3] *
roe + x[4] * accrual +
135         x[5] * eqinv + x[6] * atturn + x[7] * cfm + x
[8] * curr +
136         x[9] * da + x[10] * pcf)
137     wret = (wi * ret).sum(axis=1)
138     ut = ((1 + wret) ** (1 - rr)) / (1 - rr)
139     u = -(ut.mean())
140     return u
141
142 def PPS_pca_2(x, wb, nt, ret, component1, component2, rr):
143     wi = wb + nt * (x[0] * component1 + x[1] * component2)
144     wret = (wi * ret).sum(axis=1)
145     ut = ((1 + wret) ** (1 - rr)) / (1 - rr)
146     u = -(ut.mean())
147     return u
148
149 def PPS_pca_3(x, wb, nt, ret, component1, component2, component3,
rr):
150     wi = wb + nt * (x[0] * component1 + x[1] * component2 + x[2] *
component3)
151     wret = (wi * ret).sum(axis=1)
152     ut = ((1 + wret) ** (1 - rr)) / (1 - rr)

```

```

153     u = -(ut.mean())
154     return u
155
156 def PPS_pca_4(x, wb, nt, ret, component1, component2, component3,
157 component4, rr):
158     wi = wb + nt * (x[0] * component1 + x[1] * component2 + x[2] *
159 component3 + x[3] * component4)
160     wret = (wi * ret).sum(axis=1)
161     ut = ((1 + wret) ** (1 - rr)) / (1 - rr)
162     u = -(ut.mean())
163     return u
164
165 def PPS_pca_5(x, wb, nt, ret, component1, component2, component3,
166 component4, component5, rr):
167     wi = wb + nt * (x[0] * component1 + x[1] * component2 + x[2] *
168 component3 + x[3] * component4
169 + x[4] * component5)
170     wret = (wi * ret).sum(axis=1)
171     ut = ((1 + wret) ** (1 - rr)) / (1 - rr)
172     u = -(ut.mean())
173     return u
174
175 def PPS_pca_6(x, wb, nt, ret, component1, component2, component3,
176 component4, component5, component6, rr):
177     wi = wb + nt * (x[0] * component1 + x[1] * component2 + x[2] *
178 component3 + x[3] * component4
179 + x[4] * component5 + x[5] * component6)
180     wret = (wi * ret).sum(axis=1)
181     ut = ((1 + wret) ** (1 - rr)) / (1 - rr)
182     u = -(ut.mean())
183     return u
184
185 def PPS_pca_7(x, wb, nt, ret, component1, component2, component3,
186 component4, component5, component6, component7, rr):
187     wi = wb + nt * (x[0] * component1 + x[1] * component2 + x[2] *
188 component3 + x[3] * component4
189 + + x[4] * component5 + x[5] * component6 + + x
190 [6] * component7)
191     wret = (wi * ret).sum(axis=1)
192     ut = ((1 + wret) ** (1 - rr)) / (1 - rr)
193     u = -(ut.mean())
194     return u
195
196 def PPS_pca_8(x, wb, nt, ret, component1, component2, component3,
197 component4, component5, component6, component7,
198 component8, rr):
199     wi = wb + nt * (x[0] * component1 + x[1] * component2 + x[2] *
200 component3 + x[3] * component4 +
201 + x[4] * component5 + x[5] * component6 + + x[6]
202 * component7 + x[7] * component8)
203     wret = (wi * ret).sum(axis=1)
204     ut = ((1 + wret) ** (1 - rr)) / (1 - rr)
205     u = -(ut.mean())
206     return u
207
208 def PPS_pca_9(x, wb, nt, ret, component1, component2, component3,

```

```

197     component4, component5, component6, component7,
198     component8, component9, rr):
199     wi = wb + nt * (x[0] * component1 + x[1] * component2 + x[2] *
200     component3 + x[3] * component4 +
201     + x[4] * component5 + x[5] * component6 + + x[6]
202     * component7 + x[7] * component8 +
203     x[8] * component9)
204     wret = (wi * ret).sum(axis=1)
205     ut = ((1 + wret) ** (1 - rr)) / (1 - rr)
206     u = -(ut.mean())
207     return u
208
209 def survive(df, number_of_year):
210     survivor = []
211     stock_list = df.columns
212     for stock in stock_list:
213         living_year = pd.to_datetime(df[stock].last_valid_index()).
214         year - pd.to_datetime(df[stock].first_valid_index()).year
215         living_month = pd.to_datetime(df[stock].last_valid_index()).
216         .month - pd.to_datetime(df[stock].first_valid_index()).month
217         if (living_year >= number_of_year) & (living_month >= 0):
218             survivor.append(stock)
219     return survivor
220
221 def separate_for_pca(year_list, char_list, source_char_file_path='
222 ./new standardized5/', yearlydata_path='./new Yearly Data/'):
223
224     for year in year_list:
225         if not os.path.exists(yearlydata_path+str(year)):
226             os.makedirs(yearlydata_path+str(year))
227
228     for year in year_list:
229         stock_list = list(pd.read_csv(source_char_file_path + 'ret/
230 scaled ret' + str(year) + '.csv').set_index('date').columns)
231
232         for stock in stock_list:
233             df = pd.DataFrame()
234
235             for char in char_list:
236                 char_df = pd.read_csv(source_char_file_path+char+'/'
237 '+char+str(year)+'.csv').set_index('date')
238                 df[char] = char_df[stock]
239
240                 df.to_csv(yearlydata_path+str(year)+'/'+stock+'.csv
241 ')
242                 print("Done for company {} in {}".format(stock, year))
243
244 def cumulative_return(Weights, Return):
245
246     cumulative_return = np.nansum(Return.values[1:] * Weights.
247 values[:-1],axis=1).cumsum()
248     return cumulative_return

```

```

243
244 def top500(year_list, make_file=True):
245
246     for year in year_list:
247         df = pd.read_csv('./Investing Pool5/mktcap '+str(year)+'.'.
248             csv').set_index('date')
249         top500_index = df.iloc[0,:].sort_values(ascending=False).
250             head(500).index
251
252         df = df[top500_index]
253         if make_file:
254             df.to_csv('./top500/mktcap '+str(year)+'.'.csv')
255         # return top500_index
256
257 def bootstrap_se(theta_sample, B=10000, size=300):
258
259     theta_sample = theta_sample.values
260
261     sample_mean = []
262     for _ in range(B):
263         sample_n = np.random.choice(theta_sample, size=size)
264         sample_mean.append(sample_n.mean())
265
266     se = np.std(sample_mean)/(B**0.5)
267     return se
268
269 def make_bootstrap_sample(original_sample, B=10000, size=300):
270
271     original_sample = original_sample.values
272
273     new_sample = []
274     for _ in range(B):
275         sample_n = np.random.choice(original_sample, size=size)
276         new_sample.append(sample_n)
277
278     return new_sample
279
280 def statistic(returns, weights, coef, se):
281
282     print("mean: {}, std: {}".format(returns.mean(), returns.std())
283 )
284     print("mean: {}, max: {}, min: {}".format(weights.mean().mean()
285 , weights.max().max(), weights.min().min()))
286     print("Coef: {}".format(coef.mean()))
287     print("Se: {}".format(se.mean()))
288
289
290 def in_outofsample_statistic(insampleweight, insamplereturn,
291     insamplecoef,
292     insamplese, outsampleweight,
293     outsamplereturn, rf=risk_free):
294     print('Coef: {}'.format(insamplecoef.mean()))
295     print('-----')
296     print('Se: {}'.format(insamplese.mean()))
297     print('-----')
298     print('insample weight mean: {}'.format(insampleweight.mean().

```



```

mean()))
293     print('insample weight min: {}'.format(insampleweight.min().min
        ()))
294     print('insample weight max: {}'.format(insampleweight.max().max
        ()))
295     print('-----')
296     print('outsample weight mean: {}'.format(outsampleweight.mean()
        .mean()))
297     print('outsample weight min: {}'.format(outsampleweight.min().
        min()))
298     print('outsample weight max: {}'.format(outsampleweight.max().
        max()))
299     print('-----')
300     insample_cum_return = cumulative_return(insampleweight,
        insamplereturn)
301     outsample_cum_return = cumulative_return(outsampleweight,
        outsamplereturn)
302     insample_cum_return_mean = insample_cum_return.mean()
303     outsample_cum_return_mean = outsample_cum_return.mean()
304     insample_cum_return_std = insample_cum_return.std()
305     outsample_cum_return_std = outsample_cum_return.std()
306
307     print("insample return mean {}, std {}".format(
        insample_cum_return_mean, insample_cum_return_std))
308     print("outsample return mean {}, std {}".format(
        outsample_cums_return_mean, outsample_cum_return_std))
309
310
311     insample_rf = rf[1:104]
312     outsample_rf = rf[104:-1]
313
314     print("insample SR: {} | outsample SR: {}".format(((
        insample_cum_return_mean-insample_rf.mean())/
        insample_cum_return_std),
315                                                         ((
        outsample_cum_return_mean-outsample_rf.mean())/
        outsample_cum_return_std)))
316
317 ### Base Case
318 ## import libraries
319 import numpy as np
320 import pandas as pd
321 import matplotlib.pyplot as plt
322 import random
323 import seaborn as sns
324 from datetime import datetime
325 from scipy.stats.mstats import winsorize
326
327 from Portfolio import *
328
329 # shortselling is not allow
330 BaseWeights = pd.DataFrame()
331 BaseReturn = pd.DataFrame()
332
333 BaseCoef = pd.DataFrame(np.zeros(11)).T
334 BaseSE = pd.DataFrame()

```

```

335
336 year_list = range(1970, 2021)
337
338 for year in year_list:
339
340     df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
        set_index('date')
341
342     scaled_data_folder = './new standardized5/'
343     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/scaled ret'
        + str(year) + '.csv').set_index('date')
344     scaled_mktcap = pd.read_csv(scaled_data_folder + 'mktcap/mktcap'
        + str(year) + '.csv').set_index('date')
345     scaled_bm = pd.read_csv(scaled_data_folder + 'bm/bm' + str(year)
        + '.csv').set_index('date')
346     scaled_roa = pd.read_csv(scaled_data_folder + 'roa/roa' + str(
        year) + '.csv').set_index('date')
347     scaled_roe = pd.read_csv(scaled_data_folder + 'roe/roe' + str(
        year) + '.csv').set_index('date')
348     scaled_accrual = pd.read_csv(scaled_data_folder + 'accrual/
        accrual' + str(year) + '.csv').set_index('date')
349     scaled_cfm = pd.read_csv(scaled_data_folder + 'cfm/cfm' + str(
        year) + '.csv').set_index('date')
350     scaled_eqinv = pd.read_csv(scaled_data_folder + 'equity invcap/
        equity invcap' + str(year) + '.csv').set_index('date')
351     scaled_atturn = pd.read_csv(scaled_data_folder + 'at turn/at
        turn' + str(year) + '.csv').set_index('date')
352     scaled_pcf = pd.read_csv(scaled_data_folder + 'pcf/pcf' + str(
        year) + '.csv').set_index('date')
353     scaled_da = pd.read_csv(scaled_data_folder + 'debt asset/debt
        asset' + str(year) + '.csv').set_index('date')
354     scaled_curr = pd.read_csv(scaled_data_folder + 'curr ratio/curr
        ratio' + str(year) + '.csv').set_index('date')
355
356     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
        year)+'/09/30',str(year)+'/12/31']
357     df_ret = df_ret.loc[quarter_index, :]
358     scaled_ret = scaled_ret.loc[quarter_index, :]
359     scaled_mktcap = scaled_mktcap.loc[quarter_index, :]
360     scaled_bm = scaled_bm.loc[quarter_index, :]
361     scaled_roa = scaled_roa.loc[quarter_index, :]
362     scaled_roe = scaled_roe.loc[quarter_index, :]
363     scaled_accrual = scaled_accrual.loc[quarter_index, :]
364     scaled_cfm = scaled_cfm.loc[quarter_index, :]
365     scaled_eqinv = scaled_eqinv.loc[quarter_index, :]
366     scaled_atturn = scaled_atturn.loc[quarter_index, :]
367     scaled_pcf = scaled_pcf.loc[quarter_index, :]
368     scaled_da = scaled_da.loc[quarter_index, :]
369     scaled_curr = scaled_curr.loc[quarter_index, :]
370
371     BaseReturn = BaseReturn.append(df_ret)
372
373     nt = wb = 1 / df_ret.shape[1]
374
375     Base_results = []
376     Base_weights = []

```

```

377 Base_SE = []
378 init_points = list(BaseCoef.iloc[-1,:].values)
379
380 for i in range(4):
381     opt = scipy.optimize.minimize(
382         PPS_base,
383         init_points,
384         method="BFGS",
385         args=(
386             wb,
387             nt,
388             scaled_ret.iloc[0 : i, :],
389             scaled_mktcap.iloc[0 : i, :],
390             scaled_bm.iloc[0 : i, :],
391             scaled_roa.iloc[0 : i, :],
392             scaled_roe.iloc[0 : i, :],
393             scaled_accrual.iloc[0 : i, :],
394             scaled_eqinv.iloc[0 : i, :],
395             scaled_atturn.iloc[0 : i, :],
396             scaled_cfm.iloc[0 : i, :],
397             scaled_curr.iloc[0 : i, :],
398             scaled_da.iloc[0 : i, :],
399             scaled_pcf.iloc[0 : i, :],
400             rr,
401         ),
402     )
403     print("The {} window for year {}".format(i+1, year))
404     print("The value:", opt["x"])
405     Base_results.append(list(opt["x"]))
406
407     Base_SE.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
408     weight = wb + nt * (
409         + opt["x"][0] * scaled_mktcap.iloc[i, :]
410         + opt["x"][1] * scaled_bm.iloc[i, :]
411         + opt["x"][2] * scaled_roa.iloc[i, :]
412         + opt["x"][3] * scaled_roe.iloc[i, :]
413         + opt["x"][4] * scaled_accrual.iloc[i, :]
414         + opt["x"][5] * scaled_eqinv.iloc[i, :]
415         + opt["x"][6] * scaled_atturn.iloc[i, :]
416         + opt["x"][7] * scaled_cfm.iloc[i, :]
417         + opt["x"][8] * scaled_curr.iloc[i, :]
418         + opt["x"][9] * scaled_da.iloc[i, :]
419         + opt["x"][10] * scaled_pcf.iloc[i, :]
420     )
421     print(weight)
422     Base_weights.append(weight)
423
424     BaseWeights = BaseWeights.append(short_sell_constraints(pd.
DataFrame(Base_weights)))
425     BaseCoef = BaseCoef.append(pd.DataFrame(Base_results))
426     BaseSE = BaseSE.append(pd.DataFrame(Base_SE))
427
428 # Value Weighted and Equal weighted Portfolio
429
430 vwWeight = pd.DataFrame()
431 ewWeight = pd.DataFrame()

```

```

432
433 for year in year_list:
434
435     mktcap_source_file = './Investing Pool5/mktcap ' + str(year) +
436         '.csv'
437
438     mktcap_df = pd.read_csv(mktcap_source_file).set_index('date')
439     mktcap_df = mktcap_df.iloc[[2,5,8,11], :]
440
441     vwWeight = vwWeight.append(value_weights(mktcap_df))
442
443 for year in year_list:
444
445     portfolio_source_file = './new char5/ret/ret' + str(year) + '.
446         csv'
447
448     portfolio_df = pd.read_csv(portfolio_source_file).set_index('
449         date')
450     portfolio_df = portfolio_df.iloc[[1,4,7,10], :]
451     M = portfolio_df.shape[0]
452     N = portfolio_df.shape[1]
453
454     ewWeight = ewWeight.append(pd.DataFrame(np.ones((M,N))/N, index
455         =portfolio_df.index, columns=portfolio_df.columns))
456
457 index = BaseReturn.index[1:]
458
459 portfolio_return_df = pd.DataFrame(index=index)
460
461 opt_return = np.nansum((BaseReturn.values[1:] * BaseWeights.values
462    [:-1]), axis=1).cumsum()
463 ew_return = np.nansum((BaseReturn.values[1:] * ewWeight.values
464    [:-1]), axis=1).cumsum()
465 vw_return = np.nansum((BaseReturn.values[1:] * vwWeight.values
466    [:-1]), axis=1).cumsum()
467
468 portfolio_return_df['Optimized Characteristics Return (no short
469     sell)'] = opt_return
470 portfolio_return_df['Equal Weighted Portfolio Return'] = ew_return
471 portfolio_return_df['Value Weighted Portfolio Return'] = vw_return
472
473 portfolio_return_df.plot(figsize=(12,8))
474
475 # risk-free rate
476
477 riskfree = pd.read_csv('riskfree.csv').set_index('caldt')
478 riskfree_rate = riskfree['t30ret']
479
480 portfolio_return_df['rf'] = riskfree_rate[1:]
481
482 ## Top500 case
483
484 Top500Weights = pd.DataFrame()
485 Top500Return = pd.DataFrame()
486 Top500SE = pd.DataFrame()
487
488

```

```

480 Top500Coef = pd.DataFrame(np.zeros(11)).T
481
482 year_list = range(1970, 2021)
483
484 for year in year_list:
485
486     df_ret = pd.read_csv('./top500/ret'+str(year)+'.csv').set_index(
487         ('date'))
488     stock_list = df_ret.columns
489
490     scaled_data_folder = './new standardized5/'
491     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/scaled ret'
492 + str(year) + '.csv').set_index('date')[stock_list]
493     scaled_mktcap = pd.read_csv(scaled_data_folder + 'mktcap/mktcap'
494 + str(year) + '.csv').set_index('date')[stock_list]
495     scaled_bm = pd.read_csv(scaled_data_folder + 'bm/bm' + str(year)
496 + str(year) + '.csv').set_index('date')[stock_list]
497     scaled_roa = pd.read_csv(scaled_data_folder + 'roa/roa' + str(
498 year) + '.csv').set_index('date')[stock_list]
499     scaled_roe = pd.read_csv(scaled_data_folder + 'roe/roe' + str(
500 year) + '.csv').set_index('date')[stock_list]
501     scaled_accrual = pd.read_csv(scaled_data_folder + 'accrual/
502 accrual' + str(year) + '.csv').set_index('date')[stock_list]
503     scaled_cfm = pd.read_csv(scaled_data_folder + 'cfm/cfm' + str(
504 year) + '.csv').set_index('date')[stock_list]
505     scaled_eqinv = pd.read_csv(scaled_data_folder + 'equity invcap/
506 equity invcap' + str(year) + '.csv').set_index('date')[
507 stock_list]
508     scaled_atturn = pd.read_csv(scaled_data_folder + 'at turn/at
509 turn' + str(year) + '.csv').set_index('date')[stock_list]
510     scaled_pcf = pd.read_csv(scaled_data_folder + 'pcf/pcf' + str(
511 year) + '.csv').set_index('date')[stock_list]
512     scaled_da = pd.read_csv(scaled_data_folder + 'debt asset/debt
513 asset' + str(year) + '.csv').set_index('date')[stock_list]
514     scaled_curr = pd.read_csv(scaled_data_folder + 'curr ratio/curr
515 ratio' + str(year) + '.csv').set_index('date')[stock_list]
516
517     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
518 year)+'/09/30',str(year)+'/12/31']
519     df_ret = df_ret.loc[quarter_index, :]
520     scaled_ret = scaled_ret.loc[quarter_index, :]
521     scaled_mktcap = scaled_mktcap.loc[quarter_index, :]
522     scaled_bm = scaled_bm.loc[quarter_index, :]
523     scaled_roa = scaled_roa.loc[quarter_index, :]
524     scaled_roe = scaled_roe.loc[quarter_index, :]
525     scaled_accrual = scaled_accrual.loc[quarter_index, :]
526     scaled_cfm = scaled_cfm.loc[quarter_index, :]
527     scaled_eqinv = scaled_eqinv.loc[quarter_index, :]
528     scaled_atturn = scaled_atturn.loc[quarter_index, :]
529     scaled_pcf = scaled_pcf.loc[quarter_index, :]
530     scaled_da = scaled_da.loc[quarter_index, :]
531     scaled_curr = scaled_curr.loc[quarter_index, :]
532
533     Top500Return = Top500Return.append(df_ret)
534
535     nt = wb = 1 / df_ret.shape[1]

```

```

521
522 top500_results = []
523 top500_weights = []
524 top500_se = []
525 init_points = list(Top500Coef.iloc[-1,:].values)
526
527 for i in range(4):
528     opt = scipy.optimize.minimize(
529         PPS_base,
530         init_points,
531         method="BFGS",
532         args=(
533             wb,
534             nt,
535             scaled_ret.iloc[0 : i, :],
536             scaled_mktcap.iloc[0 : i, :],
537             scaled_bm.iloc[0 : i, :],
538             scaled_roa.iloc[0 : i, :],
539             scaled_roe.iloc[0 : i, :],
540             scaled_accrual.iloc[0 : i, :],
541             scaled_eqinv.iloc[0 : i, :],
542             scaled_atturn.iloc[0 : i, :],
543             scaled_cfm.iloc[0 : i, :],
544             scaled_curr.iloc[0 : i, :],
545             scaled_da.iloc[0 : i, :],
546             scaled_pcf.iloc[0 : i, :],
547             rr,
548         ),
549     )
550     print("The {} window for year {}".format(i+1, year))
551     print("The value:", opt["x"])
552     top500_results.append(list(opt["x"]))
553     top500_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
554     weight = wb + nt * (
555         + opt["x"][0] * scaled_mktcap.iloc[i, :]
556         + opt["x"][1] * scaled_bm.iloc[i, :]
557         + opt["x"][2] * scaled_roa.iloc[i, :]
558         + opt["x"][3] * scaled_roe.iloc[i, :]
559         + opt["x"][4] * scaled_accrual.iloc[i, :]
560         + opt["x"][5] * scaled_eqinv.iloc[i, :]
561         + opt["x"][6] * scaled_atturn.iloc[i, :]
562         + opt["x"][7] * scaled_cfm.iloc[i, :]
563         + opt["x"][8] * scaled_curr.iloc[i, :]
564         + opt["x"][9] * scaled_da.iloc[i, :]
565         + opt["x"][10] * scaled_pcf.iloc[i, :]
566     )
567     print(weight)
568     top500_weights.append(weight)
569
570 Top500Weights = Top500Weights.append(short_sell_constraints(pd.
DataFrame(top500_weights)))
571 Top500Coef = Top500Coef.append(pd.DataFrame(top500_results))
572 Top500SE = Top500SE.append(pd.DataFrame(top500_se))
573
574 # Top500 Value Weighted and Equal weighted Portfolio
575

```

```

576 vwWeight500 = pd.DataFrame()
577 ewWeight500 = pd.DataFrame()
578
579 for year in year_list:
580
581     mktcap_source_file = './top500/mktcap ' + str(year) + '.csv'
582
583     mktcap_df = pd.read_csv(mktcap_source_file).set_index('date')
584     mktcap_df = mktcap_df.iloc[[2,5,8,11], :]
585
586     vwWeight500 = vwWeight500.append(value_weights(mktcap_df))
587
588 for year in year_list:
589
590     portfolio_source_file = './top500/ret' + str(year) + '.csv'
591
592     portfolio_df = pd.read_csv(portfolio_source_file).set_index('
date')
593     portfolio_df = portfolio_df.iloc[[2,5,8,11], :]
594     M = portfolio_df.shape[0]
595     N = portfolio_df.shape[1]
596
597     ewWeight500 = ewWeight500.append(pd.DataFrame(np.ones((M,N))/N,
index=portfolio_df.index, columns=portfolio_df.columns))
598
599 ## PCA Cases
600
601 PCA2Weights = pd.DataFrame()
602 PCA2Return = pd.DataFrame()
603 PCA2SE = pd.DataFrame()
604
605 PCA2Coef = pd.DataFrame(np.zeros(2)).T
606 rr = 5
607 year_list = range(1970, 2021)
608
609 for year in year_list:
610
611     df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
set_index('date')
612
613     scaled_data_folder = './new standardized5/'
614     scaled_PCA2_folder = './PCA Case/2 npc/'
615
616     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
ret' + str(year) + '.csv').set_index('date')
617     scaled_component1 = pd.read_csv(scaled_PCA2_folder + str(year)
+ '/component 1.csv').set_index('date')
618     scaled_component2 = pd.read_csv(scaled_PCA2_folder + str(year)
+ '/component 2.csv').set_index('date')
619
620     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
year)+'/09/30', str(year)+'/12/31']
621     scaled_component1 = scaled_component1.loc[quarter_index, :]
622     scaled_component2 = scaled_component2.loc[quarter_index, :]
623     df_ret = df_ret.loc[quarter_index, :]
624

```

```

625 scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
626 scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
627
628 PCA2Return = PCA2Return.append(df_ret)
629
630 nt = wb = 1 / df_ret.shape[1]
631
632 PCA2_results = []
633 PCA2_weights = []
634 PCA2_se = []
635 init_points = list(PCA2Coef.iloc[-1,:].values)
636
637 for i in range(4):
638     opt = scipy.optimize.minimize(
639         PPS_pca_2,
640         init_points,
641         method="BFGS",
642         args=(
643             wb,
644             nt,
645             scaled_ret.iloc[0 : i, :],
646             scaled_component1.iloc[0 : i, :],
647             scaled_component2.iloc[0 : i, :],
648             rr,
649         ),
650     )
651     print("The {} window for year {}".format(i+1, year))
652     print("The value:", opt["x"])
653     PCA2_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
654     PCA2_results.append(list(opt["x"]))
655     weight = wb + nt * (
656         opt["x"][0] * scaled_component1.iloc[i, :]
657         + opt["x"][1] * scaled_component2.iloc[i, :]
658     )
659     print(weight)
660     PCA2_weights.append(weight)
661
662     PCA2Weights = PCA2Weights.append(short_sell_constraints(pd.
DataFrame(PCA2_weights)))
663     PCA2Coef = PCA2Coef.append(pd.DataFrame(PCA2_results))
664     PCA2SE = PCA2SE.append(PCA2_se)
665
666 PCA3Weights = pd.DataFrame()
667 PCA3Return = pd.DataFrame()
668 PCA3SE = pd.DataFrame()
669
670 PCA3Coef = pd.DataFrame(np.zeros(3)).T
671 rr = 5
672 year_list = range(1970, 2021)
673
674 for year in year_list:
675
676     df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
set_index('date')
677
678     scaled_data_folder = './new standardized5/'

```



```

679     scaled_PCA3_folder = './PCA Case/3 npc/'
680
681     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
ret' + str(year) + '.csv').set_index('date')
682     scaled_component1 = pd.read_csv(scaled_PCA3_folder + str(year)
+ '/component 1.csv').set_index('date')
683     scaled_component2 = pd.read_csv(scaled_PCA3_folder + str(year)
+ '/component 2.csv').set_index('date')
684     scaled_component3 = pd.read_csv(scaled_PCA3_folder + str(year)
+ '/component 3.csv').set_index('date')
685
686     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
year)+'/09/30', str(year)+'/12/31']
687     scaled_component1 = scaled_component1.loc[quarter_index, :]
688     scaled_component2 = scaled_component2.loc[quarter_index, :]
689     scaled_component3 = scaled_component3.loc[quarter_index, :]
690     df_ret = df_ret.loc[quarter_index, :]
691
692
693     scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
694     scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
695     scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
696
697     PCA3Return = PCA3Return.append(df_ret)
698
699     nt = wb = 1 / df_ret.shape[1]
700
701     PCA3_results = []
702     PCA3_weights = []
703     PCA3_se = []
704     init_points = list(PCA3Coef.iloc[-1,:].values)
705
706     for i in range(4):
707         opt = scipy.optimize.minimize(
708             PPS_pca_3,
709             init_points,
710             method="BFGS",
711             args=(
712                 wb,
713                 nt,
714                 scaled_ret.iloc[0 : i, :],
715                 scaled_component1.iloc[0 : i, :],
716                 scaled_component2.iloc[0 : i, :],
717                 scaled_component3.iloc[0 : i, :],
718                 rr,
719             ),
720         )
721         print("The {} window for year {}".format(i+1, year))
722         print("The value:", opt["x"])
723
724         PCA3_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
725         PCA3_results.append(list(opt["x"]))
726         weight = wb + nt * (
727             opt["x"][0] * scaled_component1.iloc[i, :]
728             + opt["x"][1] * scaled_component2.iloc[i, :]
729             + opt["x"][2] * scaled_component3.iloc[i, :]

```

```

730     )
731     print(weight)
732     PCA3_weights.append(weight)
733
734     PCA3Weights = PCA3Weights.append(short_sell_constraints(pd.
DataFrame(PCA3_weights)))
735     PCA3Coef = PCA3Coef.append(pd.DataFrame(PCA3_results))
736     PCA3SE = PCA3SE.append(pd.DataFrame(PCA3_se))
737
738 PCA4Weights = pd.DataFrame()
739 PCA4Return = pd.DataFrame()
740 PCA4SE = pd.DataFrame()
741
742 PCA4Coef = pd.DataFrame(np.zeros(4)).T
743 rr = 5
744 year_list = range(1970, 2021)
745
746 for year in year_list:
747
748     df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
set_index('date')
749
750     scaled_data_folder = './new standardized5/'
751     scaled_PCA4_folder = './PCA Case/4 npc/'
752
753     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
ret' + str(year) + '.csv').set_index('date')
754     scaled_component1 = pd.read_csv(scaled_PCA4_folder + str(year)
+ '/component 1.csv').set_index('date')
755     scaled_component2 = pd.read_csv(scaled_PCA4_folder + str(year)
+ '/component 2.csv').set_index('date')
756     scaled_component3 = pd.read_csv(scaled_PCA4_folder + str(year)
+ '/component 3.csv').set_index('date')
757     scaled_component4 = pd.read_csv(scaled_PCA4_folder + str(year)
+ '/component 4.csv').set_index('date')
758
759     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
year)+'/09/30',str(year)+'/12/31']
760     scaled_component1 = scaled_component1.loc[quarter_index, :]
761     scaled_component2 = scaled_component2.loc[quarter_index, :]
762     scaled_component3 = scaled_component3.loc[quarter_index, :]
763     scaled_component4 = scaled_component4.loc[quarter_index, :]
764     df_ret = df_ret.loc[quarter_index, :]
765
766     scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
767     scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
768     scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
769     scaled_component4 = pd.DataFrame(Scale(scaled_component4.T)).T
770
771     PCA4Return = PCA4Return.append(df_ret)
772
773     nt = wb = 1 / df_ret.shape[1]
774
775     PCA4_results = []
776     PCA4_weights = []
777     PCA4_se = []

```

```

778     init_points = list(PCA4Coef.iloc[-1,:].values)
779
780     for i in range(4):
781         opt = scipy.optimize.minimize(
782             PPS_pca_4,
783             init_points,
784             method="BFGS",
785             args=(
786                 wb,
787                 nt,
788                 scaled_ret.iloc[0 : i, :],
789                 scaled_component1.iloc[0 : i, :],
790                 scaled_component2.iloc[0 : i, :],
791                 scaled_component3.iloc[0 : i, :],
792                 scaled_component4.iloc[0 : i, :],
793                 rr,
794             ),
795         )
796         print("The {} window for year {}".format(i+1, year))
797         print("The value:", opt["x"])
798         PCA4_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
799
800         PCA4_results.append(list(opt["x"]))
801         weight = wb + nt * (
802             opt["x"][0] * scaled_component1.iloc[i, :]
803             + opt["x"][1] * scaled_component2.iloc[i, :]
804             + opt["x"][2] * scaled_component3.iloc[i, :]
805             + opt["x"][3] * scaled_component4.iloc[i, :]
806         )
807         print(weight)
808         PCA4_weights.append(weight)
809
810         PCA4Weights = PCA4Weights.append(short_sell_constraints(pd.
DataFrame(PCA4_weights)))
811         PCA4Coef = PCA4Coef.append(pd.DataFrame(PCA4_results))
812         PCA4SE = PCA4SE.append(pd.DataFrame(PCA4_se))
813
814     PCA5Weights = pd.DataFrame()
815     PCA5Return = pd.DataFrame()
816     PCA5SE = pd.DataFrame()
817
818     PCA5Coef = pd.DataFrame(np.zeros(5)).T
819     rr = 5
820     year_list = range(1970, 2021)
821
822     for year in year_list:
823
824         df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
set_index('date')
825
826         scaled_data_folder = './new standardized5/'
827         scaled_PCA5_folder = './PCA Case/5 npc/'
828
829         scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
ret' + str(year) + '.csv').set_index('date')
830         scaled_component1 = pd.read_csv(scaled_PCA5_folder + str(year)

```

```

+ '/component 1.csv').set_index('date')
831 scaled_component2 = pd.read_csv(scaled_PCA5_folder + str(year)
+ '/component 2.csv').set_index('date')
832 scaled_component3 = pd.read_csv(scaled_PCA5_folder + str(year)
+ '/component 3.csv').set_index('date')
833 scaled_component4 = pd.read_csv(scaled_PCA5_folder + str(year)
+ '/component 4.csv').set_index('date')
834 scaled_component5 = pd.read_csv(scaled_PCA5_folder + str(year)
+ '/component 5.csv').set_index('date')
835
836 quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
year)+'/09/30', str(year)+'/12/31']
837 scaled_component1 = scaled_component1.loc[quarter_index, :]
838 scaled_component2 = scaled_component2.loc[quarter_index, :]
839 scaled_component3 = scaled_component3.loc[quarter_index, :]
840 scaled_component4 = scaled_component4.loc[quarter_index, :]
841 scaled_component5 = scaled_component5.loc[quarter_index, :]
842 df_ret = df_ret.loc[quarter_index, :]
843
844 scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
845 scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
846 scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
847 scaled_component4 = pd.DataFrame(Scale(scaled_component4.T)).T
848 scaled_component5 = pd.DataFrame(Scale(scaled_component5.T)).T
849
850 PCA5Return = PCA5Return.append(df_ret)
851
852 nt = wb = 1 / df_ret.shape[1]
853
854 PCA5_results = []
855 PCA5_weights = []
856 PCA5_se = []
857 init_points = list(PCA5Coef.iloc[-1,:].values)
858
859 for i in range(4):
860     opt = scipy.optimize.minimize(
861         PPS_pca_5,
862         init_points,
863         method="BFGS",
864         args=(
865             wb,
866             nt,
867             scaled_ret.iloc[0 : i, :],
868             scaled_component1.iloc[0 : i, :],
869             scaled_component2.iloc[0 : i, :],
870             scaled_component3.iloc[0 : i, :],
871             scaled_component4.iloc[0 : i, :],
872             scaled_component5.iloc[0 : i, :],
873             rr,
874         ),
875     )
876     print("The {} window for year {}".format(i+1, year))
877     print("The value:", opt["x"])
878     PCA5_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
879     PCA5_results.append(list(opt["x"]))
880     weight = wb + nt * (

```

```

881         opt["x"][0] * scaled_component1.iloc[i, :]
882         + opt["x"][1] * scaled_component2.iloc[i, :]
883         + opt["x"][2] * scaled_component3.iloc[i, :]
884         + opt["x"][3] * scaled_component4.iloc[i, :]
885         + opt["x"][4] * scaled_component5.iloc[i, :]
886     )
887     print(weight)
888     PCA5_weights.append(weight)
889
890     PCA5Weights = PCA5Weights.append(short_sell_constraints(pd.
DataFrame(PCA5_weights)))
891     PCA5Coef = PCA5Coef.append(pd.DataFrame(PCA5_results))
892     PCA5SE = PCA5SE.append(pd.DataFrame(PCA5_se))
893
894
895 PCA6Weights = pd.DataFrame()
896 PCA6Return = pd.DataFrame()
897 PCA6SE = pd.DataFrame()
898
899 PCA6Coef = pd.DataFrame(np.zeros(6)).T
900 rr = 5
901 year_list = range(1970, 2021)
902
903 for year in year_list:
904
905     df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
set_index('date')
906
907     scaled_data_folder = './new standardized5/'
908     scaled_PCA6_folder = './PCA Case/6 npc/'
909
910     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
ret' + str(year) + '.csv').set_index('date')
911     scaled_component1 = pd.read_csv(scaled_PCA6_folder + str(year)
+ '/component 1.csv').set_index('date')
912     scaled_component2 = pd.read_csv(scaled_PCA6_folder + str(year)
+ '/component 2.csv').set_index('date')
913     scaled_component3 = pd.read_csv(scaled_PCA6_folder + str(year)
+ '/component 3.csv').set_index('date')
914     scaled_component4 = pd.read_csv(scaled_PCA6_folder + str(year)
+ '/component 4.csv').set_index('date')
915     scaled_component5 = pd.read_csv(scaled_PCA6_folder + str(year)
+ '/component 5.csv').set_index('date')
916     scaled_component6 = pd.read_csv(scaled_PCA6_folder + str(year)
+ '/component 6.csv').set_index('date')
917
918     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
year)+'/09/30', str(year)+'/12/31']
919     scaled_component1 = scaled_component1.loc[quarter_index, :]
920     scaled_component2 = scaled_component2.loc[quarter_index, :]
921     scaled_component3 = scaled_component3.loc[quarter_index, :]
922     scaled_component4 = scaled_component4.loc[quarter_index, :]
923     scaled_component5 = scaled_component5.loc[quarter_index, :]
924     scaled_component6 = scaled_component6.loc[quarter_index, :]
925     df_ret = df_ret.loc[quarter_index, :]
926

```

```

927 scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
928 scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
929 scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
930 scaled_component4 = pd.DataFrame(Scale(scaled_component4.T)).T
931 scaled_component5 = pd.DataFrame(Scale(scaled_component5.T)).T
932 scaled_component6 = pd.DataFrame(Scale(scaled_component6.T)).T
933
934 PCA6Return = PCA6Return.append(df_ret)
935
936 nt = wb = 1 / df_ret.shape[1]
937
938 PCA6_results = []
939 PCA6_weights = []
940 PCA6_se = []
941 init_points = list(PCA6Coef.iloc[-1,:].values)
942
943 for i in range(4):
944     opt = scipy.optimize.minimize(
945         PPS_pca_6,
946         init_points,
947         method="BFGS",
948         args=(
949             wb,
950             nt,
951             scaled_ret.iloc[0 : i, :],
952             scaled_component1.iloc[0 : i, :],
953             scaled_component2.iloc[0 : i, :],
954             scaled_component3.iloc[0 : i, :],
955             scaled_component4.iloc[0 : i, :],
956             scaled_component5.iloc[0 : i, :],
957             scaled_component6.iloc[0 : i, :],
958             rr,
959         ),
960     )
961     print("The {} window for year {}".format(i+1, year))
962     print("The value:", opt["x"])
963     PCA6_results.append(list(opt["x"]))
964     PCA6_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
965
966     weight = wb + nt * (
967         opt["x"][0] * scaled_component1.iloc[i, :]
968         + opt["x"][1] * scaled_component2.iloc[i, :]
969         + opt["x"][2] * scaled_component3.iloc[i, :]
970         + opt["x"][3] * scaled_component4.iloc[i, :]
971         + opt["x"][4] * scaled_component5.iloc[i, :]
972         + opt["x"][5] * scaled_component6.iloc[i, :]
973     )
974     print(weight)
975     PCA6_weights.append(weight)
976
977 PCA6Weights = PCA6Weights.append(short_sell_constraints(pd.
DataFrame(PCA6_weights)))
978 PCA6Coef = PCA6Coef.append(pd.DataFrame(PCA6_results))
979 PCA6SE = PCA6SE.append(pd.DataFrame(PCA6_se))
980
981 PCA7Weights = pd.DataFrame()

```

```

982 PCA7Return = pd.DataFrame()
983 PCA7SE = pd.DataFrame()
984
985 PCA7Coef = pd.DataFrame(np.zeros(7)).T
986 rr = 5
987 year_list = range(1970, 2021)
988
989 for year in year_list:
990
991     df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
        set_index('date')
992
993     scaled_data_folder = './new standardized5/'
994     scaled_PCA7_folder = './PCA Case/7 npc/'
995
996     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
        ret' + str(year) + '.csv').set_index('date')
997     scaled_component1 = pd.read_csv(scaled_PCA7_folder + str(year)
        + '/component 1.csv').set_index('date')
998     scaled_component2 = pd.read_csv(scaled_PCA7_folder + str(year)
        + '/component 2.csv').set_index('date')
999     scaled_component3 = pd.read_csv(scaled_PCA7_folder + str(year)
        + '/component 3.csv').set_index('date')
1000     scaled_component4 = pd.read_csv(scaled_PCA7_folder + str(year)
        + '/component 4.csv').set_index('date')
1001     scaled_component5 = pd.read_csv(scaled_PCA7_folder + str(year)
        + '/component 5.csv').set_index('date')
1002     scaled_component6 = pd.read_csv(scaled_PCA7_folder + str(year)
        + '/component 6.csv').set_index('date')
1003     scaled_component7 = pd.read_csv(scaled_PCA7_folder + str(year)
        + '/component 7.csv').set_index('date')
1004
1005
1006     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
        year)+'/09/30', str(year)+'/12/31']
1007     scaled_component1 = scaled_component1.loc[quarter_index, :]
1008     scaled_component2 = scaled_component2.loc[quarter_index, :]
1009     scaled_component3 = scaled_component3.loc[quarter_index, :]
1010     scaled_component4 = scaled_component4.loc[quarter_index, :]
1011     scaled_component5 = scaled_component5.loc[quarter_index, :]
1012     scaled_component6 = scaled_component6.loc[quarter_index, :]
1013     scaled_component7 = scaled_component7.loc[quarter_index, :]
1014     df_ret = df_ret.loc[quarter_index, :]
1015
1016     scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
1017     scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
1018     scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
1019     scaled_component4 = pd.DataFrame(Scale(scaled_component4.T)).T
1020     scaled_component5 = pd.DataFrame(Scale(scaled_component5.T)).T
1021     scaled_component6 = pd.DataFrame(Scale(scaled_component6.T)).T
1022     scaled_component7 = pd.DataFrame(Scale(scaled_component7.T)).T
1023
1024     PCA7Return = PCA7Return.append(df_ret)
1025
1026     nt = wb = 1 / df_ret.shape[1]
1027

```

```

1028 PCA7_results = []
1029 PCA7_weights = []
1030 PCA7_se = []
1031 init_points = list(PCA7Coef.iloc[-1,:].values)
1032
1033 for i in range(4):
1034     opt = scipy.optimize.minimize(
1035         PPS_pca_7,
1036         init_points,
1037         method="BFGS",
1038         args=(
1039             wb,
1040             nt,
1041             scaled_ret.iloc[0 : i, :],
1042             scaled_component1.iloc[0 : i, :],
1043             scaled_component2.iloc[0 : i, :],
1044             scaled_component3.iloc[0 : i, :],
1045             scaled_component4.iloc[0 : i, :],
1046             scaled_component5.iloc[0 : i, :],
1047             scaled_component6.iloc[0 : i, :],
1048             scaled_component7.iloc[0 : i, :],
1049             rr,
1050         ),
1051     )
1052     print("The {} window for year {}".format(i+1, year))
1053     print("The value:", opt["x"])
1054     PCA7_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
1055
1056     PCA7_results.append(list(opt["x"]))
1057     weight = wb + nt * (
1058         opt["x"][0] * scaled_component1.iloc[i, :]
1059         + opt["x"][1] * scaled_component2.iloc[i, :]
1060         + opt["x"][2] * scaled_component3.iloc[i, :]
1061         + opt["x"][3] * scaled_component4.iloc[i, :]
1062         + opt["x"][4] * scaled_component5.iloc[i, :]
1063         + opt["x"][5] * scaled_component6.iloc[i, :]
1064         + opt["x"][6] * scaled_component7.iloc[i, :]
1065     )
1066     print(weight)
1067     PCA7_weights.append(weight)
1068
1069     PCA7Weights = PCA7Weights.append(short_sell_constraints(pd.
DataFrame(PCA7_weights)))
1070     PCA7Coef = PCA7Coef.append(pd.DataFrame(PCA7_results))
1071     PCA7SE = PCA7SE.append(pd.DataFrame(PCA7_se))
1072
1073
1074 PCA8Weights = pd.DataFrame()
1075 PCA8Return = pd.DataFrame()
1076 PCA8SE = pd.DataFrame()
1077
1078 PCA8Coef = pd.DataFrame(np.zeros(8)).T
1079 rr = 5
1080 year_list = range(1970, 2021)
1081
1082 for year in year_list:

```



```

1083
1084     df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
set_index('date')
1085
1086     scaled_data_folder = './new standardized5/'
1087     scaled_PCA8_folder = './PCA Case/8 mpc/'
1088
1089     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
ret' + str(year) + '.csv').set_index('date')
1090     scaled_component1 = pd.read_csv(scaled_PCA8_folder + str(year)
+ '/component 1.csv').set_index('date')
1091     scaled_component2 = pd.read_csv(scaled_PCA8_folder + str(year)
+ '/component 2.csv').set_index('date')
1092     scaled_component3 = pd.read_csv(scaled_PCA8_folder + str(year)
+ '/component 3.csv').set_index('date')
1093     scaled_component4 = pd.read_csv(scaled_PCA8_folder + str(year)
+ '/component 4.csv').set_index('date')
1094     scaled_component5 = pd.read_csv(scaled_PCA8_folder + str(year)
+ '/component 5.csv').set_index('date')
1095     scaled_component6 = pd.read_csv(scaled_PCA8_folder + str(year)
+ '/component 6.csv').set_index('date')
1096     scaled_component7 = pd.read_csv(scaled_PCA8_folder + str(year)
+ '/component 7.csv').set_index('date')
1097     scaled_component8 = pd.read_csv(scaled_PCA8_folder + str(year)
+ '/component 8.csv').set_index('date')
1098
1099     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
year)+'/09/30',str(year)+'/12/31']
1100     scaled_component1 = scaled_component1.loc[quarter_index, :]
1101     scaled_component2 = scaled_component2.loc[quarter_index, :]
1102     scaled_component3 = scaled_component3.loc[quarter_index, :]
1103     scaled_component4 = scaled_component4.loc[quarter_index, :]
1104     scaled_component5 = scaled_component5.loc[quarter_index, :]
1105     scaled_component6 = scaled_component6.loc[quarter_index, :]
1106     scaled_component7 = scaled_component7.loc[quarter_index, :]
1107     scaled_component8 = scaled_component8.loc[quarter_index, :]
1108     df_ret = df_ret.loc[quarter_index, :]
1109
1110     scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
1111     scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
1112     scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
1113     scaled_component4 = pd.DataFrame(Scale(scaled_component4.T)).T
1114     scaled_component5 = pd.DataFrame(Scale(scaled_component5.T)).T
1115     scaled_component6 = pd.DataFrame(Scale(scaled_component6.T)).T
1116     scaled_component7 = pd.DataFrame(Scale(scaled_component7.T)).T
1117     scaled_component8 = pd.DataFrame(Scale(scaled_component8.T)).T
1118
1119     PCA8Return = PCA8Return.append(df_ret)
1120
1121     nt = wb = 1 / df_ret.shape[1]
1122
1123     PCA8_results = []
1124     PCA8_weights = []
1125     PCA8_se = []
1126     init_points = list(PCA8Coef.iloc[-1,:].values)
1127

```

```

1128     for i in range(4):
1129         opt = scipy.optimize.minimize(
1130             PPS_pca_8,
1131             init_points,
1132             method="BFGS",
1133             args=(
1134                 wb,
1135                 nt,
1136                 scaled_ret.iloc[0 : i, :],
1137                 scaled_component1.iloc[0 : i, :],
1138                 scaled_component2.iloc[0 : i, :],
1139                 scaled_component3.iloc[0 : i, :],
1140                 scaled_component4.iloc[0 : i, :],
1141                 scaled_component5.iloc[0 : i, :],
1142                 scaled_component6.iloc[0 : i, :],
1143                 scaled_component7.iloc[0 : i, :],
1144                 scaled_component8.iloc[0 : i, :],
1145                 rr,
1146             ),
1147         )
1148         print("The {} window for year {}".format(i+1, year))
1149         print("The value:", opt["x"])
1150         PCA8_results.append(list(opt["x"]))
1151         PCA8_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
1152
1153         weight = wb + nt * (
1154             opt["x"][0] * scaled_component1.iloc[i, :]
1155             + opt["x"][1] * scaled_component2.iloc[i, :]
1156             + opt["x"][2] * scaled_component3.iloc[i, :]
1157             + opt["x"][3] * scaled_component4.iloc[i, :]
1158             + opt["x"][4] * scaled_component5.iloc[i, :]
1159             + opt["x"][5] * scaled_component6.iloc[i, :]
1160             + opt["x"][6] * scaled_component7.iloc[i, :]
1161             + opt["x"][7] * scaled_component8.iloc[i, :]
1162         )
1163         print(weight)
1164         PCA8_weights.append(weight)
1165
1166         PCA8Weights = PCA8Weights.append(short_sell_constraints(pd.
DataFrame(PCA8_weights)))
1167         PCA8Coef = PCA8Coef.append(pd.DataFrame(PCA8_results))
1168         PCA8SE = PCA8SE.append(pd.DataFrame(PCA8_se))
1169
1170     ## Top500 PCA Cases
1171
1172     PCA2Weights500 = pd.DataFrame()
1173     PCA2Return500 = pd.DataFrame()
1174     PCA2SE500 = pd.DataFrame()
1175
1176     PCA2Coef500 = pd.DataFrame(np.zeros(2)).T
1177     rr = 5
1178     year_list = range(1970, 2021)
1179
1180     for year in year_list:
1181
1182         df_ret = pd.read_csv('./top500/ret'+str(year)+''.csv').set_index

```

```

('date')
1183 stock_list = list(df_ret.columns)
1184
1185 scaled_data_folder = './new standardized5/'
1186 scaled_PCA2_folder = './PCA Case/2 npc/'
1187
1188 scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
ret' + str(year) + '.csv').set_index('date')[stock_list]
1189 scaled_component1 = pd.read_csv(scaled_PCA2_folder + str(year)
+ '/component 1.csv').set_index('date')[stock_list]
1190 scaled_component2 = pd.read_csv(scaled_PCA2_folder + str(year)
+ '/component 2.csv').set_index('date')[stock_list]
1191
1192 quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
year)+'/09/30', str(year)+'/12/31']
1193 scaled_component1 = scaled_component1.loc[quarter_index, :]
1194 scaled_component2 = scaled_component2.loc[quarter_index, :]
1195 df_ret = df_ret.loc[quarter_index, :]
1196
1197 scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
1198 scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
1199
1200 PCA2Return500 = PCA2Return500.append(df_ret)
1201
1202 nt = wb = 1 / df_ret.shape[1]
1203
1204 PCA2_results_500 = []
1205 PCA2_weights_500 = []
1206 PCA2_se_500 = []
1207 init_points = list(PCA2Coef500.iloc[-1,:].values)
1208
1209 for i in range(4):
1210     opt = scipy.optimize.minimize(
1211         PPS_pca_2,
1212         init_points,
1213         method="BFGS",
1214         args=(
1215             wb,
1216             nt,
1217             scaled_ret.iloc[0 : i, :],
1218             scaled_component1.iloc[0 : i, :],
1219             scaled_component2.iloc[0 : i, :],
1220             rr,
1221         ),
1222     )
1223     print("The {} window for year {}".format(i+1, year))
1224     print("The value:", opt["x"])
1225     PCA2_results_500.append(list(opt["x"]))
1226     PCA2_se_500.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
1227     weight = wb + nt * (
1228         opt["x"][0] * scaled_component1.iloc[i, :]
1229         + opt["x"][1] * scaled_component2.iloc[i, :]
1230     )
1231     print(weight)
1232     PCA2_weights_500.append(weight)
1233

```

```

1234     PCA2Weights500 = PCA2Weights500.append(short_sell_constraints(
1235         pd.DataFrame(PCA2_weights_500)))
1236     PCA2Coef500 = PCA2Coef500.append(pd.DataFrame(PCA2_results_500)
1237 )
1238     PCA2SE500 = PCA2SE500.append(pd.DataFrame(PCA2_se_500))
1239
1240 PCA3Weights500 = pd.DataFrame()
1241 PCA3Return500 = pd.DataFrame()
1242 PCA3SE500 = pd.DataFrame()
1243
1244 PCA3Coef500 = pd.DataFrame(np.zeros(3)).T
1245 rr = 5
1246 year_list = range(1970, 2021)
1247
1248 for year in year_list:
1249     df_ret = pd.read_csv('./top500/ret'+str(year)+'.csv').set_index(
1250         'date')
1251     stock_list = list(df_ret.columns)
1252
1253     scaled_data_folder = './new standardized5/'
1254     scaled_PCA3_folder = './PCA Case/3 npc/'
1255
1256     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
1257 ret' + str(year) + '.csv').set_index('date')[stock_list]
1258     scaled_component1 = pd.read_csv(scaled_PCA3_folder + str(year)
1259 + '/component 1.csv').set_index('date')[stock_list]
1260     scaled_component2 = pd.read_csv(scaled_PCA3_folder + str(year)
1261 + '/component 2.csv').set_index('date')[stock_list]
1262     scaled_component3 = pd.read_csv(scaled_PCA3_folder + str(year)
1263 + '/component 3.csv').set_index('date')[stock_list]
1264
1265     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
1266 year)+'/09/30',str(year)+'/12/31']
1267     scaled_component1 = scaled_component1.loc[quarter_index, :]
1268     scaled_component2 = scaled_component2.loc[quarter_index, :]
1269     scaled_component3 = scaled_component3.loc[quarter_index, :]
1270     df_ret = df_ret.loc[quarter_index, :]
1271
1272     scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
1273     scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
1274     scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
1275
1276 PCA3Return500 = PCA3Return500.append(df_ret)
1277
1278 nt = wb = 1 / df_ret.shape[1]
1279
1280 PCA3_results_500 = []
1281 PCA3_weights_500 = []
1282 PCA3_se_500 = []
1283 init_points = list(PCA3Coef500.iloc[-1,:].values)
1284
1285 for i in range(4):
1286     opt = scipy.optimize.minimize(
1287         PPS_pca_3,

```

```

1282         init_points,
1283         method="BFGS",
1284         args=(
1285             wb,
1286             nt,
1287             scaled_ret.iloc[0 : i, :],
1288             scaled_component1.iloc[0 : i, :],
1289             scaled_component2.iloc[0 : i, :],
1290             scaled_component3.iloc[0 : i, :],
1291             rr,
1292         ),
1293     )
1294     print("The {} window for year {}".format(i+1, year))
1295     print("The value:", opt["x"])
1296     PCA3_results_500.append(list(opt["x"]))
1297     PCA3_se_500.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
1298
1299     weight = wb + nt * (
1300         opt["x"][0] * scaled_component1.iloc[i, :]
1301         + opt["x"][1] * scaled_component2.iloc[i, :]
1302         + opt["x"][2] * scaled_component3.iloc[i, :]
1303     )
1304     print(weight)
1305     PCA3_weights_500.append(weight)
1306
1307     PCA3Weights500 = PCA3Weights500.append(short_sell_constraints(
1308         pd.DataFrame(PCA3_weights_500)))
1309     PCA3Coef500 = PCA3Coef500.append(pd.DataFrame(PCA3_results_500))
1310
1311     PCA3SE500 = PCA3SE500.append(pd.DataFrame(PCA3_se_500))
1312
1313     PCA4Weights500 = pd.DataFrame()
1314     PCA4Return500 = pd.DataFrame()
1315     PCA4SE500 = pd.DataFrame()
1316     PCA4Coef500 = pd.DataFrame(np.zeros(4)).T
1317     rr = 5
1318     year_list = range(1970, 2021)
1319
1320     for year in year_list:
1321
1322         df_ret = pd.read_csv('./top500/ret'+str(year)+'.csv').set_index(
1323             'date')
1324         stock_list = list(df_ret.columns)
1325
1326         scaled_data_folder = './new standardized5/'
1327         scaled_PCA4_folder = './PCA Case/4 npc/'
1328
1329         scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
1330             ret' + str(year) + '.csv').set_index('date')[stock_list]
1331         scaled_component1 = pd.read_csv(scaled_PCA4_folder + str(year)
1332             + '/component 1.csv').set_index('date')[stock_list]
1333         scaled_component2 = pd.read_csv(scaled_PCA4_folder + str(year)
1334             + '/component 2.csv').set_index('date')[stock_list]
1335         scaled_component3 = pd.read_csv(scaled_PCA4_folder + str(year)

```

```

1332 + '/component 3.csv').set_index('date')[stock_list]
1333 scaled_component4 = pd.read_csv(scaled_PCA4_folder + str(year)
1334 + '/component 4.csv').set_index('date')[stock_list]
1335
1336 quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
1337 year)+'/09/30',str(year)+'/12/31']
1338 scaled_component1 = scaled_component1.loc[quarter_index, :]
1339 scaled_component2 = scaled_component2.loc[quarter_index, :]
1340 scaled_component3 = scaled_component3.loc[quarter_index, :]
1341 scaled_component4 = scaled_component4.loc[quarter_index, :]
1342 df_ret = df_ret.loc[quarter_index, :]
1343
1344 scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
1345 scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
1346 scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
1347 scaled_component4 = pd.DataFrame(Scale(scaled_component4.T)).T
1348
1349 PCA4Return500 = PCA4Return500.append(df_ret)
1350
1351 nt = wb = 1 / df_ret.shape[1]
1352
1353 PCA4_results_500 = []
1354 PCA4_weights_500 = []
1355 PCA4_se_500 = []
1356 init_points = list(PCA4Coef500.iloc[-1,:].values)
1357
1358 for i in range(4):
1359     opt = scipy.optimize.minimize(
1360         PPS_pca_4,
1361         init_points,
1362         method="BFGS",
1363         args=(
1364             wb,
1365             nt,
1366             scaled_ret.iloc[0 : i, :],
1367             scaled_component1.iloc[0 : i, :],
1368             scaled_component2.iloc[0 : i, :],
1369             scaled_component3.iloc[0 : i, :],
1370             scaled_component4.iloc[0 : i, :],
1371             rr,
1372         ),
1373     )
1374     print("The {} window for year {}".format(i+1, year))
1375     print("The value:", opt["x"])
1376     PCA4_results_500.append(list(opt["x"]))
1377     PCA4_se_500.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
1378
1379     weight = wb + nt * (
1380         opt["x"][0] * scaled_component1.iloc[i, :]
1381         + opt["x"][1] * scaled_component2.iloc[i, :]
1382         + opt["x"][2] * scaled_component3.iloc[i, :]
1383         + opt["x"][3] * scaled_component4.iloc[i, :]
1384     )
1385     print(weight)
1386     PCA4_weights_500.append(weight)

```

```

1385
1386     PCA4Weights500 = PCA4Weights500.append(short_sell_constraints(
1387         pd.DataFrame(PCA4_weights_500)))
1388     PCA4Coef500 = PCA4Coef500.append(pd.DataFrame(PCA4_results_500)
1389 )
1390     PCA4SE500 = PCA4SE500.append(pd.DataFrame(PCA4_se_500))
1391
1392 PCA5Weights500 = pd.DataFrame()
1393 PCA5Return500 = pd.DataFrame()
1394 PCA5SE500 = pd.DataFrame()
1395
1396 PCA5Coef500 = pd.DataFrame(np.zeros(5)).T
1397 rr = 5
1398 year_list = range(1970, 2021)
1399
1400 for year in year_list:
1401
1402     df_ret = pd.read_csv('./top500/ret'+str(year)+'.csv').set_index(
1403         'date')
1404     stock_list = list(df_ret.columns)
1405
1406     scaled_data_folder = './new standardized5/'
1407     scaled_PCA5_folder = './PCA Case/5 npc/'
1408
1409     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
1410 ret' + str(year) + '.csv').set_index('date')[stock_list]
1411     scaled_component1 = pd.read_csv(scaled_PCA5_folder + str(year)
1412 + '/component 1.csv').set_index('date')[stock_list]
1413     scaled_component2 = pd.read_csv(scaled_PCA5_folder + str(year)
1414 + '/component 2.csv').set_index('date')[stock_list]
1415     scaled_component3 = pd.read_csv(scaled_PCA5_folder + str(year)
1416 + '/component 3.csv').set_index('date')[stock_list]
1417     scaled_component4 = pd.read_csv(scaled_PCA5_folder + str(year)
1418 + '/component 4.csv').set_index('date')[stock_list]
1419     scaled_component5 = pd.read_csv(scaled_PCA5_folder + str(year)
1420 + '/component 5.csv').set_index('date')[stock_list]
1421
1422     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
1423 year)+'/09/30',str(year)+'/12/31']
1424     scaled_component1 = scaled_component1.loc[quarter_index, :]
1425     scaled_component2 = scaled_component2.loc[quarter_index, :]
1426     scaled_component3 = scaled_component3.loc[quarter_index, :]
1427     scaled_component4 = scaled_component4.loc[quarter_index, :]
1428     scaled_component5 = scaled_component5.loc[quarter_index, :]
1429
1430     df_ret = df_ret.loc[quarter_index, :]
1431
1432     scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
1433     scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
1434     scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
1435     scaled_component4 = pd.DataFrame(Scale(scaled_component4.T)).T
1436     scaled_component5 = pd.DataFrame(Scale(scaled_component5.T)).T
1437
1438     PCA5Return500 = PCA5Return500.append(df_ret)
1439

```

```

1431     nt = wb = 1 / df_ret.shape[1]
1432
1433     PCA5_results_500 = []
1434     PCA5_weights_500 = []
1435     PCA5_se_500 = []
1436     init_points = list(PCA5Coef500.iloc[-1,:].values)
1437
1438     for i in range(4):
1439         opt = scipy.optimize.minimize(
1440             PPS_pca_5,
1441             init_points,
1442             method="BFGS",
1443             args=(
1444                 wb,
1445                 nt,
1446                 scaled_ret.iloc[0 : i, :],
1447                 scaled_component1.iloc[0 : i, :],
1448                 scaled_component2.iloc[0 : i, :],
1449                 scaled_component3.iloc[0 : i, :],
1450                 scaled_component4.iloc[0 : i, :],
1451                 scaled_component5.iloc[0 : i, :],
1452                 rr,
1453             ),
1454         )
1455         print("The {} window for year {}".format(i+1, year))
1456         print("The value:", opt["x"])
1457         PCA5_results_500.append(list(opt["x"]))
1458         PCA5_se_500.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
1459
1460         weight = wb + nt * (
1461             opt["x"][0] * scaled_component1.iloc[i, :]
1462             + opt["x"][1] * scaled_component2.iloc[i, :]
1463             + opt["x"][2] * scaled_component3.iloc[i, :]
1464             + opt["x"][3] * scaled_component4.iloc[i, :]
1465             + opt["x"][4] * scaled_component5.iloc[i, :]
1466         )
1467         print(weight)
1468         PCA5_weights_500.append(weight)
1469
1470     PCA5Weights500 = PCA5Weights500.append(short_sell_constraints(
1471         pd.DataFrame(PCA5_weights_500)))
1472     PCA5Coef500 = PCA5Coef500.append(pd.DataFrame(PCA5_results_500))
1473
1474     PCA5SE500 = PCA5SE500.append(pd.DataFrame(PCA5_se_500))
1475
1476     PCA6Weights500 = pd.DataFrame()
1477     PCA6Return500 = pd.DataFrame()
1478     PCA6SE500 = pd.DataFrame()
1479
1480     PCA6Coef500 = pd.DataFrame(np.zeros(6)).T
1481     rr = 5
1482     year_list = range(1970, 2021)
1483
1484     for year in year_list:
1485         df_ret = pd.read_csv('./top500/ret'+str(year)+''.csv').set_index

```



```

('date')
1485     stock_list = list(df_ret.columns)
1486
1487     scaled_data_folder = './new standardized5/'
1488     scaled_PCA6_folder = './PCA Case/6 npc/'
1489
1490     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
ret' + str(year) + '.csv').set_index('date')[stock_list]
1491     scaled_component1 = pd.read_csv(scaled_PCA6_folder + str(year)
+ '/component 1.csv').set_index('date')[stock_list]
1492     scaled_component2 = pd.read_csv(scaled_PCA6_folder + str(year)
+ '/component 2.csv').set_index('date')[stock_list]
1493     scaled_component3 = pd.read_csv(scaled_PCA6_folder + str(year)
+ '/component 3.csv').set_index('date')[stock_list]
1494     scaled_component4 = pd.read_csv(scaled_PCA6_folder + str(year)
+ '/component 4.csv').set_index('date')[stock_list]
1495     scaled_component5 = pd.read_csv(scaled_PCA6_folder + str(year)
+ '/component 5.csv').set_index('date')[stock_list]
1496     scaled_component6 = pd.read_csv(scaled_PCA6_folder + str(year)
+ '/component 6.csv').set_index('date')[stock_list]
1497
1498     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
year)+'/09/30', str(year)+'/12/31']
1499     scaled_component1 = scaled_component1.loc[quarter_index, :]
1500     scaled_component2 = scaled_component2.loc[quarter_index, :]
1501     scaled_component3 = scaled_component3.loc[quarter_index, :]
1502     scaled_component4 = scaled_component4.loc[quarter_index, :]
1503     scaled_component5 = scaled_component5.loc[quarter_index, :]
1504     scaled_component6 = scaled_component6.loc[quarter_index, :]
1505
1506     df_ret = df_ret.loc[quarter_index, :]
1507
1508     scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
1509     scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
1510     scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
1511     scaled_component4 = pd.DataFrame(Scale(scaled_component4.T)).T
1512     scaled_component5 = pd.DataFrame(Scale(scaled_component5.T)).T
1513     scaled_component6 = pd.DataFrame(Scale(scaled_component6.T)).T
1514
1515     PCA6Return500 = PCA6Return500.append(df_ret)
1516
1517     nt = wb = 1 / df_ret.shape[1]
1518
1519     PCA6_results_500 = []
1520     PCA6_weights_500 = []
1521     PCA6_se_500 = []
1522     init_points = list(PCA6Coef500.iloc[-1,:].values)
1523
1524     for i in range(4):
1525         opt = scipy.optimize.minimize(
1526             PPS_pca_6,
1527             init_points,
1528             method="BFGS",
1529             args=(
1530                 wb,
1531                 nt,

```

```

1532         scaled_ret.iloc[0 : i, :],
1533         scaled_component1.iloc[0 : i, :],
1534         scaled_component2.iloc[0 : i, :],
1535         scaled_component3.iloc[0 : i, :],
1536         scaled_component4.iloc[0 : i, :],
1537         scaled_component5.iloc[0 : i, :],
1538         scaled_component6.iloc[0 : i, :],
1539         rr,
1540     ),
1541 )
1542 print("The {} window for year {}".format(i+1, year))
1543 print("The value:", opt["x"])
1544 PCA6_results_500.append(list(opt["x"]))
1545 PCA6_se_500.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
1546
1547 weight = wb + nt * (
1548     opt["x"][0] * scaled_component1.iloc[i, :]
1549     + opt["x"][1] * scaled_component2.iloc[i, :]
1550     + opt["x"][2] * scaled_component3.iloc[i, :]
1551     + opt["x"][3] * scaled_component4.iloc[i, :]
1552     + opt["x"][4] * scaled_component5.iloc[i, :]
1553     + opt["x"][5] * scaled_component6.iloc[i, :]
1554 )
1555 print(weight)
1556 PCA6_weights_500.append(weight)
1557
1558 PCA6Weights500 = PCA6Weights500.append(short_sell_constraints(
1559     pd.DataFrame(PCA6_weights_500)))
1560 PCA6Coef500 = PCA6Coef500.append(pd.DataFrame(PCA6_results_500))
1561 )
1562 PCA6SE500 = PCA6SE500.append(pd.DataFrame(PCA6_se_500))
1563
1564 PCA7Weights500 = pd.DataFrame()
1565 PCA7Return500 = pd.DataFrame()
1566 PCA7SE500 = pd.DataFrame()
1567 PCA7Coef500 = pd.DataFrame(np.zeros(7)).T
1568 rr = 5
1569 year_list = range(1970, 2021)
1570
1571 for year in year_list:
1572
1573     df_ret = pd.read_csv('./top500/ret'+str(year)+'.csv').set_index('date')
1574     stock_list = list(df_ret.columns)
1575
1576     scaled_data_folder = './new standardized5/'
1577     scaled_PCA7_folder = './PCA Case/7 npc/'
1578
1579     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
1580     ret' + str(year) + '.csv').set_index('date')[stock_list]
1581     scaled_component1 = pd.read_csv(scaled_PCA7_folder + str(year)
1582     + '/component 1.csv').set_index('date')[stock_list]
1583     scaled_component2 = pd.read_csv(scaled_PCA7_folder + str(year)
1584     + '/component 2.csv').set_index('date')[stock_list]

```

```

1582     scaled_component3 = pd.read_csv(scaled_PCA7_folder + str(year)
1583     + '/component 3.csv').set_index('date')[stock_list]
1584     scaled_component4 = pd.read_csv(scaled_PCA7_folder + str(year)
1585     + '/component 4.csv').set_index('date')[stock_list]
1586     scaled_component5 = pd.read_csv(scaled_PCA7_folder + str(year)
1587     + '/component 5.csv').set_index('date')[stock_list]
1588     scaled_component6 = pd.read_csv(scaled_PCA7_folder + str(year)
1589     + '/component 6.csv').set_index('date')[stock_list]
1590     scaled_component7 = pd.read_csv(scaled_PCA7_folder + str(year)
1591     + '/component 7.csv').set_index('date')[stock_list]
1592
1593     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
1594     year)+'/09/30', str(year)+'/12/31']
1595     scaled_component1 = scaled_component1.loc[quarter_index, :]
1596     scaled_component2 = scaled_component2.loc[quarter_index, :]
1597     scaled_component3 = scaled_component3.loc[quarter_index, :]
1598     scaled_component4 = scaled_component4.loc[quarter_index, :]
1599     scaled_component5 = scaled_component5.loc[quarter_index, :]
1600     scaled_component6 = scaled_component6.loc[quarter_index, :]
1601     scaled_component7 = scaled_component7.loc[quarter_index, :]
1602
1603     df_ret = df_ret.loc[quarter_index, :]
1604
1605     scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
1606     scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
1607     scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
1608     scaled_component4 = pd.DataFrame(Scale(scaled_component4.T)).T
1609     scaled_component5 = pd.DataFrame(Scale(scaled_component5.T)).T
1610     scaled_component6 = pd.DataFrame(Scale(scaled_component6.T)).T
1611     scaled_component7 = pd.DataFrame(Scale(scaled_component7.T)).T
1612
1613     PCA7Return500 = PCA7Return500.append(df_ret)
1614
1615     nt = wb = 1 / df_ret.shape[1]
1616
1617     PCA7_results_500 = []
1618     PCA7_weights_500 = []
1619     PCA7_se_500 = []
1620     init_points = list(PCA7Coef500.iloc[-1,:].values)
1621
1622     for i in range(4):
1623         opt = scipy.optimize.minimize(
1624             PPS_pca_7,
1625             init_points,
1626             method="BFGS",
1627             args=(
1628                 wb,
1629                 nt,
1630                 scaled_ret.iloc[0 : i, :],
1631                 scaled_component1.iloc[0 : i, :],
1632                 scaled_component2.iloc[0 : i, :],
1633                 scaled_component3.iloc[0 : i, :],
1634                 scaled_component4.iloc[0 : i, :],
1635                 scaled_component5.iloc[0 : i, :],
1636                 scaled_component6.iloc[0 : i, :],

```

```

1632         scaled_component7.iloc[0 : i, :],
1633         rr,
1634     ),
1635 )
1636 print("The {} window for year {}".format(i+1, year))
1637 print("The value:", opt["x"])
1638 PCA7_results_500.append(list(opt["x"]))
1639 PCA7_se_500.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
1640
1641 weight = wb + nt * (
1642     opt["x"][0] * scaled_component1.iloc[i, :]
1643     + opt["x"][1] * scaled_component2.iloc[i, :]
1644     + opt["x"][2] * scaled_component3.iloc[i, :]
1645     + opt["x"][3] * scaled_component4.iloc[i, :]
1646     + opt["x"][4] * scaled_component5.iloc[i, :]
1647     + opt["x"][5] * scaled_component6.iloc[i, :]
1648     + opt["x"][6] * scaled_component7.iloc[i, :]
1649 )
1650 print(weight)
1651 PCA7_weights_500.append(weight)
1652
1653 PCA7Weights500 = PCA7Weights500.append(short_sell_constraints(
1654     pd.DataFrame(PCA7_weights_500)))
1655 PCA7Coef500 = PCA7Coef500.append(pd.DataFrame(PCA7_results_500))
1656 )
1657 PCA7SE500 = PCA7SE500.append(pd.DataFrame(PCA7_se_500))
1658
1659 PCA8Weights500 = pd.DataFrame()
1660 PCA8Return500 = pd.DataFrame()
1661 PCA8SE500 = pd.DataFrame()
1662
1663 PCA8Coef500 = pd.DataFrame(np.zeros(8)).T
1664 rr = 5
1665 year_list = range(1970, 2021)
1666
1667 for year in year_list:
1668     df_ret = pd.read_csv('./top500/ret'+str(year)+'.csv').set_index('date')
1669     stock_list = list(df_ret.columns)
1670
1671     scaled_data_folder = './new standardized5/'
1672     scaled_PCA8_folder = './PCA Case/8 npc/'
1673
1674     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
1675     ret' + str(year) + '.csv').set_index('date')[stock_list]
1676     scaled_component1 = pd.read_csv(scaled_PCA8_folder + str(year)
1677     + '/component 1.csv').set_index('date')[stock_list]
1678     scaled_component2 = pd.read_csv(scaled_PCA8_folder + str(year)
1679     + '/component 2.csv').set_index('date')[stock_list]
1680     scaled_component3 = pd.read_csv(scaled_PCA8_folder + str(year)
1681     + '/component 3.csv').set_index('date')[stock_list]
1682     scaled_component4 = pd.read_csv(scaled_PCA8_folder + str(year)
1683     + '/component 4.csv').set_index('date')[stock_list]
1684     scaled_component5 = pd.read_csv(scaled_PCA8_folder + str(year)
1685     + '/component 5.csv').set_index('date')[stock_list]

```

```

1679     scaled_component6 = pd.read_csv(scaled_PCA8_folder + str(year)
1680     + '/component 6.csv').set_index('date')[stock_list]
1681     scaled_component7 = pd.read_csv(scaled_PCA8_folder + str(year)
1682     + '/component 7.csv').set_index('date')[stock_list]
1683     scaled_component8 = pd.read_csv(scaled_PCA8_folder + str(year)
1684     + '/component 8.csv').set_index('date')[stock_list]
1685
1686     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
1687     year)+'/09/30',str(year)+'/12/31']
1688     scaled_component1 = scaled_component1.loc[quarter_index, :]
1689     scaled_component2 = scaled_component2.loc[quarter_index, :]
1690     scaled_component3 = scaled_component3.loc[quarter_index, :]
1691     scaled_component4 = scaled_component4.loc[quarter_index, :]
1692     scaled_component5 = scaled_component5.loc[quarter_index, :]
1693     scaled_component6 = scaled_component6.loc[quarter_index, :]
1694     scaled_component7 = scaled_component7.loc[quarter_index, :]
1695     scaled_component8 = scaled_component8.loc[quarter_index, :]
1696
1697     df_ret = df_ret.loc[quarter_index, :]
1698
1699     scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
1700     scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
1701     scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
1702     scaled_component4 = pd.DataFrame(Scale(scaled_component4.T)).T
1703     scaled_component5 = pd.DataFrame(Scale(scaled_component5.T)).T
1704     scaled_component6 = pd.DataFrame(Scale(scaled_component6.T)).T
1705     scaled_component7 = pd.DataFrame(Scale(scaled_component7.T)).T
1706     scaled_component8 = pd.DataFrame(Scale(scaled_component8.T)).T
1707
1708     PCA8Return500 = PCA8Return500.append(df_ret)
1709
1710     nt = wb = 1 / df_ret.shape[1]
1711
1712     PCA8_results_500 = []
1713     PCA8_weights_500 = []
1714     PCA8_se_500 = []
1715     init_points = list(PCA8Coef500.iloc[-1,:].values)
1716
1717     for i in range(4):
1718         opt = scipy.optimize.minimize(
1719             PPS_pca_8,
1720             init_points,
1721             method="BFGS",
1722             args=(
1723                 wb,
1724                 nt,
1725                 scaled_ret.iloc[0 : i, :],
1726                 scaled_component1.iloc[0 : i, :],
1727                 scaled_component2.iloc[0 : i, :],
1728                 scaled_component3.iloc[0 : i, :],
1729                 scaled_component4.iloc[0 : i, :],
1730                 scaled_component5.iloc[0 : i, :],
1731                 scaled_component6.iloc[0 : i, :],
1732                 scaled_component7.iloc[0 : i, :],
1733                 scaled_component8.iloc[0 : i, :],

```

```

1731         rr,
1732     ),
1733 )
1734 print("The {} window for year {}".format(i+1, year))
1735 print("The value:", opt["x"])
1736 PCA8_results_500.append(list(opt["x"]))
1737 PCA8_se_500.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
1738
1739 weight = wb + nt * (
1740     opt["x"][0] * scaled_component1.iloc[i, :]
1741     + opt["x"][1] * scaled_component2.iloc[i, :]
1742     + opt["x"][2] * scaled_component3.iloc[i, :]
1743     + opt["x"][3] * scaled_component4.iloc[i, :]
1744     + opt["x"][4] * scaled_component5.iloc[i, :]
1745     + opt["x"][5] * scaled_component6.iloc[i, :]
1746     + opt["x"][6] * scaled_component7.iloc[i, :]
1747     + opt["x"][7] * scaled_component8.iloc[i, :]
1748 )
1749 print(weight)
1750 PCA8_weights_500.append(weight)
1751
1752 PCA8Weights500 = PCA8Weights500.append(short_sell_constraints(
1753     pd.DataFrame(PCA8_weights_500)))
1754 PCA8Coef500 = PCA8Coef500.append(pd.DataFrame(PCA8_results_500))
1755 )
1756 PCA8SE500 = PCA8SE500.append(pd.DataFrame(PCA8_se_500))
1757
1758 ## in-sample and out-of-sample performance
1759 # Base Case in-sample and out-of-sample performance
1760
1761 in_sample = range(1970, 1996)
1762 out_of_sample = range(1996, 2021)
1763
1764 InsampleWeights = pd.DataFrame()
1765 InsampleReturn = pd.DataFrame()
1766 InsampleCoef = pd.DataFrame(np.zeros(11)).T
1767 InsampleSE = pd.DataFrame()
1768
1769 char_name = ['mktcap', 'bm', 'roa', 'roe', 'accrual', 'equity',
1770             'invcap', 'at turn',
1771             'cfm', 'pcf', 'debt asset', 'curr ratio']
1772
1773 rr = 5
1774
1775 for year in in_sample:
1776     df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
1777         set_index('date')
1778
1779     scaled_data_folder = './new standardized5/'
1780     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/scaled ret'
1781 + str(year) + '.csv').set_index('date')
1782     scaled_mktcap = pd.read_csv(scaled_data_folder + 'mktcap/mktcap'
1783 + str(year) + '.csv').set_index('date')
1784     scaled_bm = pd.read_csv(scaled_data_folder + 'bm/bm' + str(year)

```

```

) + '.csv').set_index('date')
1781 scaled_roa = pd.read_csv(scaled_data_folder + 'roa/roa' + str(
year) + '.csv').set_index('date')
1782 scaled_roe = pd.read_csv(scaled_data_folder + 'roe/roe' + str(
year) + '.csv').set_index('date')
1783 scaled_accrual = pd.read_csv(scaled_data_folder + 'accrual/
accrual' + str(year) + '.csv').set_index('date')
1784 scaled_cfm = pd.read_csv(scaled_data_folder + 'cfm/cfm' + str(
year) + '.csv').set_index('date')
1785 scaled_eqinv = pd.read_csv(scaled_data_folder + 'equity invcap/
equity invcap' + str(year) + '.csv').set_index('date')
1786 scaled_atturn = pd.read_csv(scaled_data_folder + 'at turn/at
turn' + str(year) + '.csv').set_index('date')
1787 scaled_pcf = pd.read_csv(scaled_data_folder + 'pcf/pcf' + str(
year) + '.csv').set_index('date')
1788 scaled_da = pd.read_csv(scaled_data_folder + 'debt asset/debt
asset' + str(year) + '.csv').set_index('date')
1789 scaled_curr = pd.read_csv(scaled_data_folder + 'curr ratio/curr
ratio' + str(year) + '.csv').set_index('date')
1790
1791 quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
year)+'/09/30',str(year)+'/12/31']
1792 df_ret = df_ret.loc[quarter_index, :]
1793 scaled_ret = scaled_ret.loc[quarter_index, :]
1794 scaled_mktcap = scaled_mktcap.loc[quarter_index, :]
1795 scaled_bm = scaled_bm.loc[quarter_index, :]
1796 scaled_roa = scaled_roa.loc[quarter_index, :]
1797 scaled_roe = scaled_roe.loc[quarter_index, :]
1798 scaled_accrual = scaled_accrual.loc[quarter_index, :]
1799 scaled_cfm = scaled_cfm.loc[quarter_index, :]
1800 scaled_eqinv = scaled_eqinv.loc[quarter_index, :]
1801 scaled_atturn = scaled_atturn.loc[quarter_index, :]
1802 scaled_pcf = scaled_pcf.loc[quarter_index, :]
1803 scaled_da = scaled_da.loc[quarter_index, :]
1804 scaled_curr = scaled_curr.loc[quarter_index, :]
1805
1806 InsampleReturn = InsampleReturn.append(df_ret)
1807
1808 nt = wb = 1 / df_ret.shape[1]
1809
1810 insample_results = []
1811 insample_weights = []
1812 insample_se = []
1813 init_points = list(InsampleCoef.iloc[-1,:].values)
1814
1815 for i in range(4):
1816     opt = scipy.optimize.minimize(
1817         PPS_base,
1818         init_points,
1819         method="BFGS",
1820         args=(
1821             wb,
1822             nt,
1823             scaled_ret.iloc[0 : i, :],
1824             scaled_mktcap.iloc[0 : i, :],
1825             scaled_bm.iloc[0 : i, :],

```

```

1826         scaled_roa.iloc[0 : i, :],
1827         scaled_roe.iloc[0 : i, :],
1828         scaled_accrual.iloc[0 : i, :],
1829         scaled_eqinv.iloc[0 : i, :],
1830         scaled_atturn.iloc[0 : i, :],
1831         scaled_cfm.iloc[0 : i, :],
1832         scaled_curr.iloc[0 : i, :],
1833         scaled_da.iloc[0 : i, :],
1834         scaled_pcf.iloc[0 : i, :],
1835         rr,
1836     ),
1837 )
1838 print("The {} window for year {}".format(i+1, year))
1839 print("The value:", opt["x"])
1840 insample_results.append(list(opt["x"]))
1841 insample_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
1842
1843 weight = wb + nt * (
1844     + opt["x"][0] * scaled_mktcap.iloc[i, :]
1845     + opt["x"][1] * scaled_bm.iloc[i, :]
1846     + opt["x"][2] * scaled_roa.iloc[i, :]
1847     + opt["x"][3] * scaled_roe.iloc[i, :]
1848     + opt["x"][4] * scaled_accrual.iloc[i, :]
1849     + opt["x"][5] * scaled_eqinv.iloc[i, :]
1850     + opt["x"][6] * scaled_atturn.iloc[i, :]
1851     + opt["x"][7] * scaled_cfm.iloc[i, :]
1852     + opt["x"][8] * scaled_curr.iloc[i, :]
1853     + opt["x"][9] * scaled_da.iloc[i, :]
1854     + opt["x"][10] * scaled_pcf.iloc[i, :]
1855 )
1856 print(weight)
1857 insample_weights.append(weight)
1858
1859 InsampleWeights = InsampleWeights.append(short_sell_constraints
1860 (pd.DataFrame(insample_weights)))
1861 InsampleCoef = InsampleCoef.append(pd.DataFrame(
1862 insample_results))
1863 InsampleSE = InsampleSE.append(insample_se)
1864
1865 ## PCA cases in-sample and out-of-sample performance
1866
1867 PCA2InsampleWeights = pd.DataFrame()
1868 PCA2InsampleReturn = pd.DataFrame()
1869 PCA2InsampleCoef = pd.DataFrame(np.zeros(2)).T
1870 PCA2InsampleSE = pd.DataFrame()
1871
1872 rr = 5
1873
1874 for year in in_sample:
1875     df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
1876     set_index('date')
1877
1878     scaled_data_folder = './new standardized5/'
1879     scaled_PCA2_folder = './PCA Case/2 npc/'

```



```

1879
1880     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
scaled_ret' + str(year) + '.csv').set_index('date')
1881     scaled_component1 = pd.read_csv(scaled_PCA2_folder + str(year)
+ '/component 1.csv').set_index('date')
1882     scaled_component2 = pd.read_csv(scaled_PCA2_folder + str(year)
+ '/component 2.csv').set_index('date')
1883
1884     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
year)+'/09/30',str(year)+'/12/31']
1885     scaled_component1 = scaled_component1.loc[quarter_index, :]
1886     scaled_component2 = scaled_component2.loc[quarter_index, :]
1887     df_ret = df_ret.loc[quarter_index, :]
1888
1889     scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
1890     scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
1891
1892     PCA2InsampleReturn = PCA2InsampleReturn.append(df_ret)
1893
1894     nt = wb = 1 / df_ret.shape[1]
1895
1896     PCA2_results = []
1897     PCA2_weights = []
1898     PCA2_se = []
1899     init_points = list(PCA2InsampleCoef.iloc[-1,:].values)
1900
1901     for i in range(4):
1902         opt = scipy.optimize.minimize(
1903             PPS_pca_2,
1904             init_points,
1905             method="BFGS",
1906             args=(
1907                 wb,
1908                 nt,
1909                 scaled_ret.iloc[0 : i, :],
1910                 scaled_component1.iloc[0 : i, :],
1911                 scaled_component2.iloc[0 : i, :],
1912                 rr,
1913             ),
1914         )
1915         print("The {} window for year {}".format(i+1, year))
1916         print("The value:", opt["x"])
1917         PCA2_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
1918         PCA2_results.append(list(opt["x"]))
1919         weight = wb + nt * (
1920             opt["x"][0] * scaled_component1.iloc[i, :]
1921             + opt["x"][1] * scaled_component2.iloc[i, :]
1922         )
1923         print(weight)
1924         PCA2_weights.append(weight)
1925
1926     PCA2InsampleWeights = PCA2InsampleWeights.append(
short_sell_constraints(pd.DataFrame(PCA2_weights)))
1927     PCA2InsampleCoef = PCA2InsampleCoef.append(pd.DataFrame(
PCA2_results))
1928     PCA2InsampleSE = PCA2InsampleSE.append(PCA2_se)

```

```

1929
1930 pca2insample_coef = PCA2InsampleCoef.mean()
1931 PCA2OutofSampleWeights = pd.DataFrame()
1932 PCA2OutofSampleReturn = pd.DataFrame()
1933 for year in out_of_sample:
1934
1935     df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
        set_index('date')
1936
1937     scaled_data_folder = './new standardized5/'
1938     scaled_PCA2_folder = './PCA Case/2 npc/'
1939
1940     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
        ret' + str(year) + '.csv').set_index('date')
1941     scaled_component1 = pd.read_csv(scaled_PCA2_folder + str(year)
        + '/component 1.csv').set_index('date')
1942     scaled_component2 = pd.read_csv(scaled_PCA2_folder + str(year)
        + '/component 2.csv').set_index('date')
1943
1944     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
        year)+'/09/30',str(year)+'/12/31']
1945     scaled_component1 = scaled_component1.loc[quarter_index, :]
1946     scaled_component2 = scaled_component2.loc[quarter_index, :]
1947     df_ret = df_ret.loc[quarter_index, :]
1948
1949     scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
1950     scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
1951
1952     PCA2OutofSampleReturn = PCA2OutofSampleReturn.append(df_ret)
1953
1954     nt = wb = 1 / df_ret.shape[1]
1955
1956     outofsample_weights = []
1957
1958     for i in range(len(df_ret)):
1959         weight = wb + nt * (
1960             + pca2insample_coef[0] * scaled_component1.iloc[i, :]
1961             + pca2insample_coef[1] * scaled_component2.iloc[i, :]
1962         )
1963         outofsample_weights.append(weight)
1964         print(weight)
1965
1966     PCA2OutofSampleWeights = PCA2OutofSampleWeights.append(
        short_sell_constraints(pd.DataFrame(outofsample_weights)))
1967
1968 PCA3InsampleWeights = pd.DataFrame()
1969 PCA3InsampleReturn = pd.DataFrame()
1970 PCA3InsampleCoef = pd.DataFrame(np.zeros(3)).T
1971 PCA3InsampleSE = pd.DataFrame()
1972
1973 rr = 5
1974
1975
1976 for year in in_sample:
1977
1978     df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').

```

```

set_index('date')

scaled_data_folder = './new standardized5/'
scaled_PCA3_folder = './PCA Case/3 npc/'

scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
ret' + str(year) + '.csv').set_index('date')
scaled_component1 = pd.read_csv(scaled_PCA3_folder + str(year)
+ '/component 1.csv').set_index('date')
scaled_component2 = pd.read_csv(scaled_PCA3_folder + str(year)
+ '/component 2.csv').set_index('date')
scaled_component3 = pd.read_csv(scaled_PCA3_folder + str(year)
+ '/component 3.csv').set_index('date')

quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
year)+'/09/30',str(year)+'/12/31']
scaled_component1 = scaled_component1.loc[quarter_index, :]
scaled_component2 = scaled_component2.loc[quarter_index, :]
scaled_component3 = scaled_component3.loc[quarter_index, :]

df_ret = df_ret.loc[quarter_index, :]

scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T

PCA3InsampleReturn = PCA3InsampleReturn.append(df_ret)

nt = wb = 1 / df_ret.shape[1]

PCA3_results = []
PCA3_weights = []
PCA3_se = []
init_points = list(PCA3InsampleCoef.iloc[-1,:].values)

for i in range(4):
    opt = scipy.optimize.minimize(
        PPS_pca_3,
        init_points,
        method="BFGS",
        args=(
            wb,
            nt,
            scaled_ret.iloc[0 : i, :],
            scaled_component1.iloc[0 : i, :],
            scaled_component2.iloc[0 : i, :],
            scaled_component3.iloc[0 : i, :],
            rr,
        ),
    )
    print("The {} window for year {}".format(i+1, year))
    print("The value:", opt["x"])
    PCA3_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
    PCA3_results.append(list(opt["x"]))
    weight = wb + nt * (

```

```

2029         opt["x"][0] * scaled_component1.iloc[i, :]
2030         + opt["x"][1] * scaled_component2.iloc[i, :]
2031         + opt["x"][2] * scaled_component3.iloc[i, :]
2032     )
2033     print(weight)
2034     PCA3_weights.append(weight)
2035
2036     PCA3InsampleWeights = PCA3InsampleWeights.append(
2037         short_sell_constraints(pd.DataFrame(PCA3_weights)))
2038     PCA3InsampleCoef = PCA3InsampleCoef.append(pd.DataFrame(
2039         PCA3_results))
2040     PCA3InsampleSE = PCA3InsampleSE.append(PCA3_se)
2041
2042     pca3insample_coef = PCA3InsampleCoef.mean()
2043     PCA3OutofSampleWeights = pd.DataFrame()
2044     PCA3OutofSampleReturn = pd.DataFrame()
2045     for year in out_of_sample:
2046
2047         df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
2048             set_index('date')
2049
2050         scaled_data_folder = './new standardized5/'
2051         scaled_PCA3_folder = './PCA Case/3 npc/'
2052
2053         scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
2054         ret' + str(year) + '.csv').set_index('date')
2055         scaled_component1 = pd.read_csv(scaled_PCA3_folder + str(year)
2056         + '/component 1.csv').set_index('date')
2057         scaled_component2 = pd.read_csv(scaled_PCA3_folder + str(year)
2058         + '/component 2.csv').set_index('date')
2059         scaled_component3 = pd.read_csv(scaled_PCA3_folder + str(year)
2060         + '/component 3.csv').set_index('date')
2061
2062         quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
2063         year)+'/09/30',str(year)+'/12/31']
2064         scaled_component1 = scaled_component1.loc[quarter_index, :]
2065         scaled_component2 = scaled_component2.loc[quarter_index, :]
2066         scaled_component3 = scaled_component3.loc[quarter_index, :]
2067         df_ret = df_ret.loc[quarter_index, :]
2068
2069         scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
2070         scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
2071         scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
2072
2073         PCA3OutofSampleReturn = PCA3OutofSampleReturn.append(df_ret)
2074
2075         nt = wb = 1 / df_ret.shape[1]
2076
2077         outofsample_weights = []
2078
2079         for i in range(len(df_ret)):
2080             weight = wb + nt * (
2081                 + pca3insample_coef[0] * scaled_component1.iloc[i, :]
2082                 + pca3insample_coef[1] * scaled_component2.iloc[i, :]
2083                 + pca3insample_coef[1] * scaled_component3.iloc[i, :]
2084             )

```

```

2077         outofsample_weights.append(weight)
2078         print(weight)
2079
2080     PCA3OutofSampleWeights = PCA3OutofSampleWeights.append(
2081         short_sell_constraints(pd.DataFrame(outofsample_weights)))
2082
2083     PCA4InsampleWeights = pd.DataFrame()
2084     PCA4InsampleReturn = pd.DataFrame()
2085     PCA4InsampleCoef = pd.DataFrame(np.zeros(4)).T
2086     PCA4InsampleSE = pd.DataFrame()
2087
2088     rr = 5
2089
2090     for year in in_sample:
2091
2092         df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
2093             set_index('date')
2094
2095         scaled_data_folder = './new standardized5/'
2096         scaled_PCA4_folder = './PCA Case/4 npc/'
2097
2098         scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
2099         ret' + str(year) + '.csv').set_index('date')
2100         scaled_component1 = pd.read_csv(scaled_PCA4_folder + str(year)
2101         + '/component 1.csv').set_index('date')
2102         scaled_component2 = pd.read_csv(scaled_PCA4_folder + str(year)
2103         + '/component 2.csv').set_index('date')
2104         scaled_component3 = pd.read_csv(scaled_PCA4_folder + str(year)
2105         + '/component 3.csv').set_index('date')
2106         scaled_component4 = pd.read_csv(scaled_PCA4_folder + str(year)
2107         + '/component 4.csv').set_index('date')
2108
2109         quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
2110         year)+'/09/30',str(year)+'/12/31']
2111         scaled_component1 = scaled_component1.loc[quarter_index, :]
2112         scaled_component2 = scaled_component2.loc[quarter_index, :]
2113         scaled_component3 = scaled_component3.loc[quarter_index, :]
2114         scaled_component4 = scaled_component4.loc[quarter_index, :]
2115
2116         df_ret = df_ret.loc[quarter_index, :]
2117
2118         scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
2119         scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
2120         scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
2121         scaled_component4 = pd.DataFrame(Scale(scaled_component4.T)).T
2122
2123         PCA4InsampleReturn = PCA4InsampleReturn.append(df_ret)
2124
2125     nt = wb = 1 / df_ret.shape[1]
2126
2127     PCA4_results = []
2128     PCA4_weights = []
2129     PCA4_se = []

```

```

2125     init_points = list(PCA4InsampleCoef.iloc[-1,:].values)
2126
2127     for i in range(4):
2128         opt = scipy.optimize.minimize(
2129             PPS_pca_4,
2130             init_points,
2131             method="BFGS",
2132             args=(
2133                 wb,
2134                 nt,
2135                 scaled_ret.iloc[0 : i, :],
2136                 scaled_component1.iloc[0 : i, :],
2137                 scaled_component2.iloc[0 : i, :],
2138                 scaled_component3.iloc[0 : i, :],
2139                 scaled_component4.iloc[0 : i, :],
2140                 rr,
2141             ),
2142         )
2143         print("The {} window for year {}".format(i+1, year))
2144         print("The value:", opt["x"])
2145         PCA4_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
2146         PCA4_results.append(list(opt["x"]))
2147         weight = wb + nt * (
2148             opt["x"][0] * scaled_component1.iloc[i, :]
2149             + opt["x"][1] * scaled_component2.iloc[i, :]
2150             + opt["x"][2] * scaled_component3.iloc[i, :]
2151             + opt["x"][3] * scaled_component4.iloc[i, :]
2152         )
2153         print(weight)
2154         PCA4_weights.append(weight)
2155
2156     PCA4InsampleWeights = PCA4InsampleWeights.append(
2157         short_sell_constraints(pd.DataFrame(PCA4_weights)))
2158     PCA4InsampleCoef = PCA4InsampleCoef.append(pd.DataFrame(
2159         PCA4_results))
2160     PCA4InsampleSE = PCA4InsampleSE.append(PCA4_se)
2161
2162     pca4insample_coef = PCA4InsampleCoef.mean()
2163     PCA4OutofSampleWeights = pd.DataFrame()
2164     PCA4OutofSampleReturn = pd.DataFrame()
2165     for year in out_of_sample:
2166         df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
2167             set_index('date')
2168
2169         scaled_data_folder = './new standardized5/'
2170         scaled_PCA4_folder = './PCA Case/4 npc/'
2171
2172         scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
2173             ret' + str(year) + '.csv').set_index('date')
2174         scaled_component1 = pd.read_csv(scaled_PCA4_folder + str(year)
2175             + '/component 1.csv').set_index('date')
2176         scaled_component2 = pd.read_csv(scaled_PCA4_folder + str(year)
2177             + '/component 2.csv').set_index('date')
2178         scaled_component3 = pd.read_csv(scaled_PCA4_folder + str(year)
2179             + '/component 3.csv').set_index('date')

```

```

2174     scaled_component4 = pd.read_csv(scaled_PCA4_folder + str(year)
2175     + '/component 4.csv').set_index('date')
2176
2177     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
2178     year)+'/09/30',str(year)+'/12/31']
2179     scaled_component1 = scaled_component1.loc[quarter_index, :]
2180     scaled_component2 = scaled_component2.loc[quarter_index, :]
2181     scaled_component3 = scaled_component3.loc[quarter_index, :]
2182     scaled_component4 = scaled_component4.loc[quarter_index, :]
2183     df_ret = df_ret.loc[quarter_index, :]
2184
2185     scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
2186     scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
2187     scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
2188     scaled_component4 = pd.DataFrame(Scale(scaled_component4.T)).T
2189
2190     PCA4OutofSampleReturn = PCA4OutofSampleReturn.append(df_ret)
2191
2192     nt = wb = 1 / df_ret.shape[1]
2193
2194     outofsample_weights = []
2195
2196     for i in range(len(df_ret)):
2197         weight = wb + nt * (
2198             + pca4insample_coef[0] * scaled_component1.iloc[i, :]
2199             + pca4insample_coef[1] * scaled_component2.iloc[i, :]
2200             + pca4insample_coef[2] * scaled_component3.iloc[i, :]
2201             + pca4insample_coef[3] * scaled_component4.iloc[i, :]
2202         )
2203         outofsample_weights.append(weight)
2204         print(weight)
2205
2206     PCA4OutofSampleWeights = PCA4OutofSampleWeights.append(
2207     short_sell_constraints(pd.DataFrame(outofsample_weights)))
2208
2209     PCA5InsampleWeights = pd.DataFrame()
2210     PCA5InsampleReturn = pd.DataFrame()
2211     PCA5InsampleCoef = pd.DataFrame(np.zeros(5)).T
2212     PCA5InsampleSE = pd.DataFrame()
2213
2214     rr = 5
2215
2216
2217     for year in in_sample:
2218
2219         df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
2220         set_index('date')
2221
2222         scaled_data_folder = './new standardized5/'
2223         scaled_PCA5_folder = './PCA Case/5 npc/'
2224
2225         scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
2226         ret' + str(year) + '.csv').set_index('date')

```

```

2225     scaled_component1 = pd.read_csv(scaled_PCA5_folder + str(year)
2226     + '/component 1.csv').set_index('date')
2227     scaled_component2 = pd.read_csv(scaled_PCA5_folder + str(year)
2228     + '/component 2.csv').set_index('date')
2229     scaled_component3 = pd.read_csv(scaled_PCA5_folder + str(year)
2230     + '/component 3.csv').set_index('date')
2231     scaled_component4 = pd.read_csv(scaled_PCA5_folder + str(year)
2232     + '/component 4.csv').set_index('date')
2233     scaled_component5 = pd.read_csv(scaled_PCA5_folder + str(year)
2234     + '/component 5.csv').set_index('date')
2235
2236     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
2237     year)+'/09/30',str(year)+'/12/31']
2238     scaled_component1 = scaled_component1.loc[quarter_index, :]
2239     scaled_component2 = scaled_component2.loc[quarter_index, :]
2240     scaled_component3 = scaled_component3.loc[quarter_index, :]
2241     scaled_component4 = scaled_component4.loc[quarter_index, :]
2242     scaled_component5 = scaled_component5.loc[quarter_index, :]
2243
2244     df_ret = df_ret.loc[quarter_index, :]
2245
2246     scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
2247     scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
2248     scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
2249     scaled_component4 = pd.DataFrame(Scale(scaled_component4.T)).T
2250     scaled_component5 = pd.DataFrame(Scale(scaled_component5.T)).T
2251
2252     PCA5InsampleReturn = PCA5InsampleReturn.append(df_ret)
2253
2254     nt = wb = 1 / df_ret.shape[1]
2255
2256     PCA5_results = []
2257     PCA5_weights = []
2258     PCA5_se = []
2259     init_points = list(PCA5InsampleCoef.iloc[-1,:].values)
2260
2261     for i in range(4):
2262         opt = scipy.optimize.minimize(
2263             PPS_pca_5,
2264             init_points,
2265             method="BFGS",
2266             args=(
2267                 wb,
2268                 nt,
2269                 scaled_ret.iloc[0 : i, :],
2270                 scaled_component1.iloc[0 : i, :],
2271                 scaled_component2.iloc[0 : i, :],
2272                 scaled_component3.iloc[0 : i, :],
2273                 scaled_component4.iloc[0 : i, :],
2274                 scaled_component5.iloc[0 : i, :],
2275                 rr,
2276             ),
2277         )
2278         print("The {} window for year {}".format(i+1, year))

```



```

2275     print("The value:", opt["x"])
2276     PCA5_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
2277     PCA5_results.append(list(opt["x"]))
2278     weight = wb + nt * (
2279         opt["x"][0] * scaled_component1.iloc[i, :]
2280         + opt["x"][1] * scaled_component2.iloc[i, :]
2281         + opt["x"][2] * scaled_component3.iloc[i, :]
2282         + opt["x"][3] * scaled_component4.iloc[i, :]
2283         + opt["x"][4] * scaled_component5.iloc[i, :]
2284     )
2285     print(weight)
2286     PCA5_weights.append(weight)
2287
2288     PCA5InsampleWeights = PCA5InsampleWeights.append(
2289         short_sell_constraints(pd.DataFrame(PCA5_weights)))
2290     PCA5InsampleCoef = PCA5InsampleCoef.append(pd.DataFrame(
2291         PCA5_results))
2292     PCA5InsampleSE = PCA5InsampleSE.append(PCA5_se)
2293
2294     pca5insample_coef = PCA5InsampleCoef.mean()
2295     PCA5OutofSampleWeights = pd.DataFrame()
2296     PCA5OutofSampleReturn = pd.DataFrame()
2297     for year in out_of_sample:
2298
2299         df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
2300             set_index('date')
2301
2302         scaled_data_folder = './new standardized5/'
2303         scaled_PCA5_folder = './PCA Case/5 npc/'
2304
2305         scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
2306         ret' + str(year) + '.csv').set_index('date')
2307         scaled_component1 = pd.read_csv(scaled_PCA5_folder + str(year)
2308         + '/component 1.csv').set_index('date')
2309         scaled_component2 = pd.read_csv(scaled_PCA5_folder + str(year)
2310         + '/component 2.csv').set_index('date')
2311         scaled_component3 = pd.read_csv(scaled_PCA5_folder + str(year)
2312         + '/component 3.csv').set_index('date')
2313         scaled_component4 = pd.read_csv(scaled_PCA5_folder + str(year)
2314         + '/component 4.csv').set_index('date')
2315         scaled_component5 = pd.read_csv(scaled_PCA5_folder + str(year)
2316         + '/component 5.csv').set_index('date')
2317
2318         quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
2319         year)+'/09/30',str(year)+'/12/31']
2320         scaled_component1 = scaled_component1.loc[quarter_index, :]
2321         scaled_component2 = scaled_component2.loc[quarter_index, :]
2322         scaled_component3 = scaled_component3.loc[quarter_index, :]
2323         scaled_component4 = scaled_component4.loc[quarter_index, :]
2324         scaled_component5 = scaled_component5.loc[quarter_index, :]
2325         df_ret = df_ret.loc[quarter_index, :]
2326
2327         scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
2328         scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
2329         scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T

```

```

2321 scaled_component4 = pd.DataFrame(Scale(scaled_component4.T)).T
2322 scaled_component5 = pd.DataFrame(Scale(scaled_component5.T)).T
2323
2324 PCA5OutofSampleReturn = PCA5OutofSampleReturn.append(df_ret)
2325
2326 nt = wb = 1 / df_ret.shape[1]
2327
2328 outofsample_weights = []
2329
2330 for i in range(len(df_ret)):
2331     weight = wb + nt * (
2332         + pca5insample_coef[0] * scaled_component1.iloc[i, :]
2333         + pca5insample_coef[1] * scaled_component2.iloc[i, :]
2334         + pca5insample_coef[2] * scaled_component3.iloc[i, :]
2335         + pca5insample_coef[3] * scaled_component4.iloc[i, :]
2336         + pca5insample_coef[4] * scaled_component5.iloc[i, :]
2337     )
2338     outofsample_weights.append(weight)
2339     print(weight)
2340
2341 PCA5OutofSampleWeights = PCA5OutofSampleWeights.append(
2342     short_sell_constraints(pd.DataFrame(outofsample_weights)))
2343
2344 PCA6InsampleWeights = pd.DataFrame()
2345 PCA6InsampleReturn = pd.DataFrame()
2346 PCA6InsampleCoef = pd.DataFrame(np.zeros(6)).T
2347 PCA6InsampleSE = pd.DataFrame()
2348
2349 rr = 5
2350
2351 for year in in_sample:
2352
2353     df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
2354     set_index('date')
2355
2356     scaled_data_folder = './new standardized5/'
2357     scaled_PCA6_folder = './PCA Case/6 npc/'
2358
2359     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
2360     ret' + str(year) + '.csv').set_index('date')
2361     scaled_component1 = pd.read_csv(scaled_PCA6_folder + str(year)
2362     + '/component 1.csv').set_index('date')
2363     scaled_component2 = pd.read_csv(scaled_PCA6_folder + str(year)
2364     + '/component 2.csv').set_index('date')
2365     scaled_component3 = pd.read_csv(scaled_PCA6_folder + str(year)
2366     + '/component 3.csv').set_index('date')
2367     scaled_component4 = pd.read_csv(scaled_PCA6_folder + str(year)
2368     + '/component 4.csv').set_index('date')
2369     scaled_component5 = pd.read_csv(scaled_PCA6_folder + str(year)
2370     + '/component 5.csv').set_index('date')
2371     scaled_component6 = pd.read_csv(scaled_PCA6_folder + str(year)
2372     + '/component 6.csv').set_index('date')
2373
2374     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
2375     year)+'/09/30', str(year)+'/12/31']

```

```

2367 scaled_component1 = scaled_component1.loc[quarter_index, :]
2368 scaled_component2 = scaled_component2.loc[quarter_index, :]
2369 scaled_component3 = scaled_component3.loc[quarter_index, :]
2370 scaled_component4 = scaled_component4.loc[quarter_index, :]
2371 scaled_component5 = scaled_component5.loc[quarter_index, :]
2372 scaled_component6 = scaled_component6.loc[quarter_index, :]
2373
2374
2375 df_ret = df_ret.loc[quarter_index, :]
2376
2377 scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
2378 scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
2379 scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
2380 scaled_component4 = pd.DataFrame(Scale(scaled_component4.T)).T
2381 scaled_component5 = pd.DataFrame(Scale(scaled_component5.T)).T
2382 scaled_component6 = pd.DataFrame(Scale(scaled_component6.T)).T
2383 PCA6InsampleReturn = PCA6InsampleReturn.append(df_ret)
2384
2385 nt = wb = 1 / df_ret.shape[1]
2386
2387 PCA6_results = []
2388 PCA6_weights = []
2389 PCA6_se = []
2390 init_points = list(PCA6InsampleCoef.iloc[-1,:].values)
2391
2392 for i in range(4):
2393     opt = scipy.optimize.minimize(
2394         PPS_pca_6,
2395         init_points,
2396         method="BFGS",
2397         args=(
2398             wb,
2399             nt,
2400             scaled_ret.iloc[0 : i, :],
2401             scaled_component1.iloc[0 : i, :],
2402             scaled_component2.iloc[0 : i, :],
2403             scaled_component3.iloc[0 : i, :],
2404             scaled_component4.iloc[0 : i, :],
2405             scaled_component5.iloc[0 : i, :],
2406             scaled_component6.iloc[0 : i, :],
2407             rr,
2408         ),
2409     )
2410     print("The {} window for year {}".format(i+1, year))
2411     print("The value:", opt["x"])
2412     PCA6_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
2413     PCA6_results.append(list(opt["x"]))
2414     weight = wb + nt * (
2415         opt["x"][0] * scaled_component1.iloc[i, :]
2416         + opt["x"][1] * scaled_component2.iloc[i, :]
2417         + opt["x"][2] * scaled_component3.iloc[i, :]
2418         + opt["x"][3] * scaled_component4.iloc[i, :]
2419         + opt["x"][4] * scaled_component5.iloc[i, :]
2420         + opt["x"][5] * scaled_component6.iloc[i, :]
2421     )
2422     print(weight)

```

```

2423         PCA6_weights.append(weight)
2424
2425         PCA6InsampleWeights = PCA6InsampleWeights.append(
2426             short_sell_constraints(pd.DataFrame(PCA6_weights)))
2427         PCA6InsampleCoef = PCA6InsampleCoef.append(pd.DataFrame(
2428             PCA6_results))
2429         PCA6InsampleSE = PCA6InsampleSE.append(PCA6_se)
2430
2431     pca6insample_coef = PCA6InsampleCoef.mean()
2432     PCA6OutofSampleWeights = pd.DataFrame()
2433     PCA6OutofSampleReturn = pd.DataFrame()
2434     for year in out_of_sample:
2435
2436         df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
2437             set_index('date')
2438
2439         scaled_data_folder = './new standardized5/'
2440         scaled_PCA6_folder = './PCA Case/6 npc/'
2441
2442         scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
2443             ret' + str(year) + '.csv').set_index('date')
2444         scaled_component1 = pd.read_csv(scaled_PCA6_folder + str(year)
2445             + '/component 1.csv').set_index('date')
2446         scaled_component2 = pd.read_csv(scaled_PCA6_folder + str(year)
2447             + '/component 2.csv').set_index('date')
2448         scaled_component3 = pd.read_csv(scaled_PCA6_folder + str(year)
2449             + '/component 3.csv').set_index('date')
2450         scaled_component4 = pd.read_csv(scaled_PCA6_folder + str(year)
2451             + '/component 4.csv').set_index('date')
2452         scaled_component5 = pd.read_csv(scaled_PCA6_folder + str(year)
2453             + '/component 5.csv').set_index('date')
2454         scaled_component6 = pd.read_csv(scaled_PCA6_folder + str(year)
2455             + '/component 6.csv').set_index('date')
2456
2457         quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
2458             year)+'/09/30',str(year)+'/12/31']
2459         scaled_component1 = scaled_component1.loc[quarter_index, :]
2460         scaled_component2 = scaled_component2.loc[quarter_index, :]
2461         scaled_component3 = scaled_component3.loc[quarter_index, :]
2462         scaled_component4 = scaled_component4.loc[quarter_index, :]
2463         scaled_component5 = scaled_component5.loc[quarter_index, :]
2464         scaled_component6 = scaled_component6.loc[quarter_index, :]
2465         df_ret = df_ret.loc[quarter_index, :]
2466
2467         scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
2468         scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
2469         scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
2470         scaled_component4 = pd.DataFrame(Scale(scaled_component4.T)).T
2471         scaled_component5 = pd.DataFrame(Scale(scaled_component5.T)).T
2472         scaled_component6 = pd.DataFrame(Scale(scaled_component6.T)).T
2473
2474         PCA6OutofSampleReturn = PCA6OutofSampleReturn.append(df_ret)
2475
2476         nt = wb = 1 / df_ret.shape[1]

```

```

2468     outofsample_weights = []
2469
2470     for i in range(len(df_ret)):
2471         weight = wb + nt * (
2472             + pca6insample_coef[0] * scaled_component1.iloc[i, :]
2473             + pca6insample_coef[1] * scaled_component2.iloc[i, :]
2474             + pca6insample_coef[2] * scaled_component3.iloc[i, :]
2475             + pca6insample_coef[3] * scaled_component4.iloc[i, :]
2476             + pca6insample_coef[4] * scaled_component5.iloc[i, :]
2477             + pca6insample_coef[5] * scaled_component6.iloc[i, :]
2478         )
2479         outofsample_weights.append(weight)
2480         print(weight)
2481
2482     PCA6OutofSampleWeights = PCA6OutofSampleWeights.append(
2483         short_sell_constraints(pd.DataFrame(outofsample_weights)))
2484
2485     PCA7InsampleWeights = pd.DataFrame()
2486     PCA7InsampleReturn = pd.DataFrame()
2487     PCA7InsampleCoef = pd.DataFrame(np.zeros(7)).T
2488     PCA7InsampleSE = pd.DataFrame()
2489
2490     rr = 5
2491
2492     for year in in_sample:
2493
2494         df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
2495             set_index('date')
2496
2497         scaled_data_folder = './new standardized5/'
2498         scaled_PCA7_folder = './PCA Case/7 npc/'
2499
2500         scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
2501             ret' + str(year) + '.csv').set_index('date')
2502         scaled_component1 = pd.read_csv(scaled_PCA7_folder + str(year)
2503             + '/component 1.csv').set_index('date')
2504         scaled_component2 = pd.read_csv(scaled_PCA7_folder + str(year)
2505             + '/component 2.csv').set_index('date')
2506         scaled_component3 = pd.read_csv(scaled_PCA7_folder + str(year)
2507             + '/component 3.csv').set_index('date')
2508         scaled_component4 = pd.read_csv(scaled_PCA7_folder + str(year)
2509             + '/component 4.csv').set_index('date')
2510         scaled_component5 = pd.read_csv(scaled_PCA7_folder + str(year)
2511             + '/component 5.csv').set_index('date')
2512         scaled_component6 = pd.read_csv(scaled_PCA7_folder + str(year)
2513             + '/component 6.csv').set_index('date')
2514         scaled_component7 = pd.read_csv(scaled_PCA7_folder + str(year)
2515             + '/component 7.csv').set_index('date')
2516
2517         quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
2518             year)+'/09/30', str(year)+'/12/31']
2519         scaled_component1 = scaled_component1.loc[quarter_index, :]
2520         scaled_component2 = scaled_component2.loc[quarter_index, :]
2521         scaled_component3 = scaled_component3.loc[quarter_index, :]
2522         scaled_component4 = scaled_component4.loc[quarter_index, :]

```

```

2513 scaled_component5 = scaled_component5.loc[quarter_index, :]
2514 scaled_component6 = scaled_component6.loc[quarter_index, :]
2515 scaled_component7 = scaled_component7.loc[quarter_index, :]
2516
2517 df_ret = df_ret.loc[quarter_index, :]
2518
2519 scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
2520 scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
2521 scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
2522 scaled_component4 = pd.DataFrame(Scale(scaled_component4.T)).T
2523 scaled_component5 = pd.DataFrame(Scale(scaled_component5.T)).T
2524 scaled_component6 = pd.DataFrame(Scale(scaled_component6.T)).T
2525 scaled_component7 = pd.DataFrame(Scale(scaled_component7.T)).T
2526 PCA7InsampleReturn = PCA7InsampleReturn.append(df_ret)
2527
2528 nt = wb = 1 / df_ret.shape[1]
2529
2530 PCA7_results = []
2531 PCA7_weights = []
2532 PCA7_se = []
2533 init_points = list(PCA7InsampleCoef.iloc[-1,:].values)
2534
2535 for i in range(4):
2536     opt = scipy.optimize.minimize(
2537         PPS_pca_7,
2538         init_points,
2539         method="BFGS",
2540         args=(
2541             wb,
2542             nt,
2543             scaled_ret.iloc[0 : i, :],
2544             scaled_component1.iloc[0 : i, :],
2545             scaled_component2.iloc[0 : i, :],
2546             scaled_component3.iloc[0 : i, :],
2547             scaled_component4.iloc[0 : i, :],
2548             scaled_component5.iloc[0 : i, :],
2549             scaled_component6.iloc[0 : i, :],
2550             scaled_component7.iloc[0 : i, :],
2551             rr,
2552         ),
2553     )
2554     print("The {} window for year {}".format(i+1, year))
2555     print("The value:", opt["x"])
2556     PCA7_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
2557     PCA7_results.append(list(opt["x"]))
2558     weight = wb + nt * (
2559         opt["x"][0] * scaled_component1.iloc[i, :]
2560         + opt["x"][1] * scaled_component2.iloc[i, :]
2561         + opt["x"][2] * scaled_component3.iloc[i, :]
2562         + opt["x"][3] * scaled_component4.iloc[i, :]
2563         + opt["x"][4] * scaled_component5.iloc[i, :]
2564         + opt["x"][5] * scaled_component6.iloc[i, :]
2565         + opt["x"][6] * scaled_component7.iloc[i, :]
2566     )
2567     print(weight)
2568     PCA7_weights.append(weight)

```

```

2569
2570     PCA7InsampleWeights = PCA7InsampleWeights.append(
short_sell_constraints(pd.DataFrame(PCA7_weights)))
2571     PCA7InsampleCoef = PCA7InsampleCoef.append(pd.DataFrame(
PCA7_results))
2572     PCA7InsampleSE = PCA7InsampleSE.append(PCA7_se)
2573
2574     pca7insample_coef = PCA7InsampleCoef.mean()
2575     PCA7OutofSampleWeights = pd.DataFrame()
2576     PCA7OutofSampleReturn = pd.DataFrame()
2577     for year in out_of_sample:
2578
2579         df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
set_index('date')
2580
2581         scaled_data_folder = './new standardized5/'
2582         scaled_PCA7_folder = './PCA Case/7 npc/'
2583
2584         scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
ret' + str(year) + '.csv').set_index('date')
2585         scaled_component1 = pd.read_csv(scaled_PCA7_folder + str(year)
+ '/component 1.csv').set_index('date')
2586         scaled_component2 = pd.read_csv(scaled_PCA7_folder + str(year)
+ '/component 2.csv').set_index('date')
2587         scaled_component3 = pd.read_csv(scaled_PCA7_folder + str(year)
+ '/component 3.csv').set_index('date')
2588         scaled_component4 = pd.read_csv(scaled_PCA7_folder + str(year)
+ '/component 4.csv').set_index('date')
2589         scaled_component5 = pd.read_csv(scaled_PCA7_folder + str(year)
+ '/component 5.csv').set_index('date')
2590         scaled_component6 = pd.read_csv(scaled_PCA7_folder + str(year)
+ '/component 6.csv').set_index('date')
2591         scaled_component7 = pd.read_csv(scaled_PCA7_folder + str(year)
+ '/component 7.csv').set_index('date')
2592
2593
2594         quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
year)+'/09/30',str(year)+'/12/31']
2595         scaled_component1 = scaled_component1.loc[quarter_index, :]
2596         scaled_component2 = scaled_component2.loc[quarter_index, :]
2597         scaled_component3 = scaled_component3.loc[quarter_index, :]
2598         scaled_component4 = scaled_component4.loc[quarter_index, :]
2599         scaled_component5 = scaled_component5.loc[quarter_index, :]
2600         scaled_component6 = scaled_component6.loc[quarter_index, :]
2601         scaled_component7 = scaled_component7.loc[quarter_index, :]
2602         df_ret = df_ret.loc[quarter_index, :]
2603
2604         scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
2605         scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
2606         scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
2607         scaled_component4 = pd.DataFrame(Scale(scaled_component4.T)).T
2608         scaled_component5 = pd.DataFrame(Scale(scaled_component5.T)).T
2609         scaled_component6 = pd.DataFrame(Scale(scaled_component6.T)).T
2610         scaled_component7 = pd.DataFrame(Scale(scaled_component7.T)).T
2611
2612         PCA7OutofSampleReturn = PCA7OutofSampleReturn.append(df_ret)

```



```

2613
2614     nt = wb = 1 / df_ret.shape[1]
2615
2616     outofsample_weights = []
2617
2618     for i in range(len(df_ret)):
2619         weight = wb + nt * (
2620             + pca7insample_coef[0] * scaled_component1.iloc[i, :]
2621             + pca7insample_coef[1] * scaled_component2.iloc[i, :]
2622             + pca7insample_coef[2] * scaled_component3.iloc[i, :]
2623             + pca7insample_coef[3] * scaled_component4.iloc[i, :]
2624             + pca7insample_coef[4] * scaled_component5.iloc[i, :]
2625             + pca7insample_coef[5] * scaled_component6.iloc[i, :]
2626             + pca7insample_coef[6] * scaled_component7.iloc[i, :]
2627         )
2628         outofsample_weights.append(weight)
2629         print(weight)
2630
2631     PCA7OutOfSampleWeights = PCA7OutOfSampleWeights.append(
2632         short_sell_constraints(pd.DataFrame(outofsample_weights)))
2633
2634     PCA8InsampleWeights = pd.DataFrame()
2635     PCA8InsampleReturn = pd.DataFrame()
2636     PCA8InsampleCoef = pd.DataFrame(np.zeros(8)).T
2637     PCA8InsampleSE = pd.DataFrame()
2638
2639     rr = 5
2640
2641
2642     for year in in_sample:
2643
2644         df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
2645             set_index('date')
2646
2647         scaled_data_folder = './new standardized5/'
2648         scaled_PCA8_folder = './PCA Case/8 npc/'
2649
2650         scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
2651         ret' + str(year) + '.csv').set_index('date')
2652         scaled_component1 = pd.read_csv(scaled_PCA8_folder + str(year)
2653         + '/component 1.csv').set_index('date')
2654         scaled_component2 = pd.read_csv(scaled_PCA8_folder + str(year)
2655         + '/component 2.csv').set_index('date')
2656         scaled_component3 = pd.read_csv(scaled_PCA8_folder + str(year)
2657         + '/component 3.csv').set_index('date')
2658         scaled_component4 = pd.read_csv(scaled_PCA8_folder + str(year)
2659         + '/component 4.csv').set_index('date')
2660         scaled_component5 = pd.read_csv(scaled_PCA8_folder + str(year)
2661         + '/component 5.csv').set_index('date')
2662         scaled_component6 = pd.read_csv(scaled_PCA8_folder + str(year)
2663         + '/component 6.csv').set_index('date')
2664         scaled_component7 = pd.read_csv(scaled_PCA8_folder + str(year)
2665         + '/component 7.csv').set_index('date')
2666         scaled_component8 = pd.read_csv(scaled_PCA8_folder + str(year)
2667         + '/component 8.csv').set_index('date')

```



```

2658
2659
2660     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
2661 year)+'/09/30',str(year)+'/12/31']
2662     scaled_component1 = scaled_component1.loc[quarter_index, :]
2663     scaled_component2 = scaled_component2.loc[quarter_index, :]
2664     scaled_component3 = scaled_component3.loc[quarter_index, :]
2665     scaled_component4 = scaled_component4.loc[quarter_index, :]
2666     scaled_component5 = scaled_component5.loc[quarter_index, :]
2667     scaled_component6 = scaled_component6.loc[quarter_index, :]
2668     scaled_component7 = scaled_component7.loc[quarter_index, :]
2669     scaled_component8 = scaled_component8.loc[quarter_index, :]
2670
2671     df_ret = df_ret.loc[quarter_index, :]
2672
2673     scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
2674     scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
2675     scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
2676     scaled_component4 = pd.DataFrame(Scale(scaled_component4.T)).T
2677     scaled_component5 = pd.DataFrame(Scale(scaled_component5.T)).T
2678     scaled_component6 = pd.DataFrame(Scale(scaled_component6.T)).T
2679     scaled_component7 = pd.DataFrame(Scale(scaled_component7.T)).T
2680     scaled_component8 = pd.DataFrame(Scale(scaled_component8.T)).T
2681     PCA8InsampleReturn = PCA8InsampleReturn.append(df_ret)
2682
2683     nt = wb = 1 / df_ret.shape[1]
2684
2685     PCA8_results = []
2686     PCA8_weights = []
2687     PCA8_se = []
2688     init_points = list(PCA8InsampleCoef.iloc[-1,:].values)
2689
2690     for i in range(4):
2691         opt = scipy.optimize.minimize(
2692             PPS_pca_8,
2693             init_points,
2694             method="BFGS",
2695             args=(
2696                 wb,
2697                 nt,
2698                 scaled_ret.iloc[0 : i, :],
2699                 scaled_component1.iloc[0 : i, :],
2700                 scaled_component2.iloc[0 : i, :],
2701                 scaled_component3.iloc[0 : i, :],
2702                 scaled_component4.iloc[0 : i, :],
2703                 scaled_component5.iloc[0 : i, :],
2704                 scaled_component6.iloc[0 : i, :],
2705                 scaled_component7.iloc[0 : i, :],
2706                 scaled_component8.iloc[0 : i, :],
2707                 rr,
2708             ),
2709             print("The {} window for year {}".format(i+1, year))
2710             print("The value:", opt["x"])
2711             PCA8_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
2712             PCA8_results.append(list(opt["x"]))

```

```

2713     weight = wb + nt * (
2714         opt["x"][0] * scaled_component1.iloc[i, :]
2715         + opt["x"][1] * scaled_component2.iloc[i, :]
2716         + opt["x"][2] * scaled_component3.iloc[i, :]
2717         + opt["x"][3] * scaled_component4.iloc[i, :]
2718         + opt["x"][4] * scaled_component5.iloc[i, :]
2719         + opt["x"][5] * scaled_component6.iloc[i, :]
2720         + opt["x"][6] * scaled_component7.iloc[i, :]
2721         + opt["x"][7] * scaled_component8.iloc[i, :]
2722     )
2723     print(weight)
2724     PCA8_weights.append(weight)
2725
2726     PCA8InsampleWeights = PCA8InsampleWeights.append(
2727         short_sell_constraints(pd.DataFrame(PCA8_weights)))
2728     PCA8InsampleCoef = PCA8InsampleCoef.append(pd.DataFrame(
2729         PCA8_results))
2730     PCA8InsampleSE = PCA8InsampleSE.append(PCA8_se)
2731
2732     pca8insample_coef = PCA8InsampleCoef.mean()
2733     PCA8OutOfSampleWeights = pd.DataFrame()
2734     PCA8OutOfSampleReturn = pd.DataFrame()
2735     for year in out_of_sample:
2736
2737         df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
2738             set_index('date')
2739
2740         scaled_data_folder = './new standardized5/'
2741         scaled_PCA8_folder = './PCA Case/8 npc/'
2742
2743         scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
2744         ret' + str(year) + '.csv').set_index('date')
2745         scaled_component1 = pd.read_csv(scaled_PCA8_folder + str(year)
2746         + '/component 1.csv').set_index('date')
2747         scaled_component2 = pd.read_csv(scaled_PCA8_folder + str(year)
2748         + '/component 2.csv').set_index('date')
2749         scaled_component3 = pd.read_csv(scaled_PCA8_folder + str(year)
2750         + '/component 3.csv').set_index('date')
2751         scaled_component4 = pd.read_csv(scaled_PCA8_folder + str(year)
2752         + '/component 4.csv').set_index('date')
2753         scaled_component5 = pd.read_csv(scaled_PCA8_folder + str(year)
2754         + '/component 5.csv').set_index('date')
2755         scaled_component6 = pd.read_csv(scaled_PCA8_folder + str(year)
2756         + '/component 6.csv').set_index('date')
2757         scaled_component7 = pd.read_csv(scaled_PCA8_folder + str(year)
2758         + '/component 7.csv').set_index('date')
2759         scaled_component8 = pd.read_csv(scaled_PCA8_folder + str(year)
2760         + '/component 8.csv').set_index('date')
2761
2762         quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
2763         year)+'/09/30', str(year)+'/12/31']
2764         scaled_component1 = scaled_component1.loc[quarter_index, :]
2765         scaled_component2 = scaled_component2.loc[quarter_index, :]
2766         scaled_component3 = scaled_component3.loc[quarter_index, :]
2767         scaled_component4 = scaled_component4.loc[quarter_index, :]

```

```

2756 scaled_component5 = scaled_component5.loc[quarter_index, :]
2757 scaled_component6 = scaled_component6.loc[quarter_index, :]
2758 scaled_component7 = scaled_component7.loc[quarter_index, :]
2759 scaled_component8 = scaled_component8.loc[quarter_index, :]
2760 df_ret = df_ret.loc[quarter_index, :]
2761
2762 scaled_component1 = pd.DataFrame(Scale(scaled_component1.T)).T
2763 scaled_component2 = pd.DataFrame(Scale(scaled_component2.T)).T
2764 scaled_component3 = pd.DataFrame(Scale(scaled_component3.T)).T
2765 scaled_component4 = pd.DataFrame(Scale(scaled_component4.T)).T
2766 scaled_component5 = pd.DataFrame(Scale(scaled_component5.T)).T
2767 scaled_component6 = pd.DataFrame(Scale(scaled_component6.T)).T
2768 scaled_component7 = pd.DataFrame(Scale(scaled_component7.T)).T
2769 scaled_component8 = pd.DataFrame(Scale(scaled_component8.T)).T
2770
2771 PCA8OutofSampleReturn = PCA8OutofSampleReturn.append(df_ret)
2772
2773 nt = wb = 1 / df_ret.shape[1]
2774
2775 outofsample_weights = []
2776
2777 for i in range(len(df_ret)):
2778     weight = wb + nt * (
2779         + pca8insample_coef[0] * scaled_component1.iloc[i, :]
2780         + pca8insample_coef[1] * scaled_component2.iloc[i, :]
2781         + pca8insample_coef[2] * scaled_component3.iloc[i, :]
2782         + pca8insample_coef[3] * scaled_component4.iloc[i, :]
2783         + pca8insample_coef[4] * scaled_component5.iloc[i, :]
2784         + pca8insample_coef[5] * scaled_component6.iloc[i, :]
2785         + pca8insample_coef[6] * scaled_component7.iloc[i, :]
2786         + pca8insample_coef[7] * scaled_component8.iloc[i, :]
2787     )
2788     outofsample_weights.append(weight)
2789     print(weight)
2790
2791 PCA8OutofSampleWeights = PCA8OutofSampleWeights.append(
2792     short_sell_constraints(pd.DataFrame(outofsample_weights)))
2793
2794 ### Risk Aversion
2795 ## Base Case
2796
2797 rr = 1
2798
2799 BaseWeights1 = pd.DataFrame()
2800 BaseReturn1 = pd.DataFrame()
2801
2802 BaseCoef1 = pd.DataFrame(np.zeros(11)).T
2803 BaseSE1 = pd.DataFrame()
2804
2805 year_list = range(1970, 2021)
2806
2807 for year in year_list:
2808     df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
2809     set_index('date')

```

```

2810     scaled_data_folder = './new standardized5/'
2811     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/scaled ret'
+ str(year) + '.csv').set_index('date')
2812     scaled_mktcap = pd.read_csv(scaled_data_folder + 'mktcap/mktcap'
+ str(year) + '.csv').set_index('date')
2813     scaled_bm = pd.read_csv(scaled_data_folder + 'bm/bm' + str(year)
+ '.csv').set_index('date')
2814     scaled_roa = pd.read_csv(scaled_data_folder + 'roa/roa' + str(
year) + '.csv').set_index('date')
2815     scaled_roe = pd.read_csv(scaled_data_folder + 'roe/roe' + str(
year) + '.csv').set_index('date')
2816     scaled_accrual = pd.read_csv(scaled_data_folder + 'accrual/
accrual' + str(year) + '.csv').set_index('date')
2817     scaled_cfm = pd.read_csv(scaled_data_folder + 'cfm/cfm' + str(
year) + '.csv').set_index('date')
2818     scaled_eqinv = pd.read_csv(scaled_data_folder + 'equity invcap/
equity invcap' + str(year) + '.csv').set_index('date')
2819     scaled_atturn = pd.read_csv(scaled_data_folder + 'at turn/at
turn' + str(year) + '.csv').set_index('date')
2820     scaled_pcf = pd.read_csv(scaled_data_folder + 'pcf/pcf' + str(
year) + '.csv').set_index('date')
2821     scaled_da = pd.read_csv(scaled_data_folder + 'debt asset/debt
asset' + str(year) + '.csv').set_index('date')
2822     scaled_curr = pd.read_csv(scaled_data_folder + 'curr ratio/curr
ratio' + str(year) + '.csv').set_index('date')
2823
2824     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
year)+'/09/30',str(year)+'/12/31']
2825     df_ret = df_ret.loc[quarter_index, :]
2826     scaled_ret = scaled_ret.loc[quarter_index, :]
2827     scaled_mktcap = scaled_mktcap.loc[quarter_index, :]
2828     scaled_bm = scaled_bm.loc[quarter_index, :]
2829     scaled_roa = scaled_roa.loc[quarter_index, :]
2830     scaled_roe = scaled_roe.loc[quarter_index, :]
2831     scaled_accrual = scaled_accrual.loc[quarter_index, :]
2832     scaled_cfm = scaled_cfm.loc[quarter_index, :]
2833     scaled_eqinv = scaled_eqinv.loc[quarter_index, :]
2834     scaled_atturn = scaled_atturn.loc[quarter_index, :]
2835     scaled_pcf = scaled_pcf.loc[quarter_index, :]
2836     scaled_da = scaled_da.loc[quarter_index, :]
2837     scaled_curr = scaled_curr.loc[quarter_index, :]
2838
2839     BaseReturn1 = BaseReturn1.append(df_ret)
2840
2841     nt = wb = 1 / df_ret.shape[1]
2842
2843     Base_results = []
2844     Base_weights = []
2845     Base_SE = []
2846     init_points = list(BaseCoef1.iloc[-1,:].values)
2847
2848     for i in range(4):
2849         opt = scipy.optimize.minimize(
2850             PPS_base,
2851             init_points,
2852             method="BFGS",

```

```

2853         args=(
2854             wb,
2855             nt,
2856             scaled_ret.iloc[0 : i, :],
2857             scaled_mktcap.iloc[0 : i, :],
2858             scaled_bm.iloc[0 : i, :],
2859             scaled_roa.iloc[0 : i, :],
2860             scaled_roe.iloc[0 : i, :],
2861             scaled_accrual.iloc[0 : i, :],
2862             scaled_eqinv.iloc[0 : i, :],
2863             scaled_atturn.iloc[0 : i, :],
2864             scaled_cfm.iloc[0 : i, :],
2865             scaled_curr.iloc[0 : i, :],
2866             scaled_da.iloc[0 : i, :],
2867             scaled_pcf.iloc[0 : i, :],
2868             rr,
2869         ),
2870     )
2871     print("The {} window for year {}".format(i+1, year))
2872     print("The value:", opt["x"])
2873     Base_results.append(list(opt["x"]))
2874
2875     Base_SE.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
2876     weight = wb + nt * (
2877         + opt["x"][0] * scaled_mktcap.iloc[i, :]
2878         + opt["x"][1] * scaled_bm.iloc[i, :]
2879         + opt["x"][2] * scaled_roa.iloc[i, :]
2880         + opt["x"][3] * scaled_roe.iloc[i, :]
2881         + opt["x"][4] * scaled_accrual.iloc[i, :]
2882         + opt["x"][5] * scaled_eqinv.iloc[i, :]
2883         + opt["x"][6] * scaled_atturn.iloc[i, :]
2884         + opt["x"][7] * scaled_cfm.iloc[i, :]
2885         + opt["x"][8] * scaled_curr.iloc[i, :]
2886         + opt["x"][9] * scaled_da.iloc[i, :]
2887         + opt["x"][10] * scaled_pcf.iloc[i, :]
2888     )
2889     print(weight)
2890     Base_weights.append(weight)
2891
2892     BaseWeights1 = BaseWeights1.append(short_sell_constraints(pd.
2893     DataFrame(Base_weights)))
2894     BaseCoef1 = BaseCoef1.append(pd.DataFrame(Base_results))
2895     BaseSE1 = BaseSE1.append(pd.DataFrame(Base_SE))
2896
2897     rr = 3
2898
2899     BaseWeights3 = pd.DataFrame()
2900     BaseReturn3 = pd.DataFrame()
2901
2902     BaseCoef3 = pd.DataFrame(np.zeros(11)).T
2903     BaseSE3 = pd.DataFrame()
2904
2905     year_list = range(1970, 2021)
2906     for year in year_list:
2907

```

```

2908     df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
        set_index('date')
2909
2910     scaled_data_folder = './new standardized5/'
2911     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/scaled ret'
        + str(year) + '.csv').set_index('date')
2912     scaled_mktcap = pd.read_csv(scaled_data_folder + 'mktcap/mktcap'
        + str(year) + '.csv').set_index('date')
2913     scaled_bm = pd.read_csv(scaled_data_folder + 'bm/bm' + str(year)
        + '.csv').set_index('date')
2914     scaled_roa = pd.read_csv(scaled_data_folder + 'roa/roa' + str(
        year) + '.csv').set_index('date')
2915     scaled_roe = pd.read_csv(scaled_data_folder + 'roe/roe' + str(
        year) + '.csv').set_index('date')
2916     scaled_accrual = pd.read_csv(scaled_data_folder + 'accrual/
        accrual' + str(year) + '.csv').set_index('date')
2917     scaled_cfm = pd.read_csv(scaled_data_folder + 'cfm/cfm' + str(
        year) + '.csv').set_index('date')
2918     scaled_eqinv = pd.read_csv(scaled_data_folder + 'equity invcap/
        equity invcap' + str(year) + '.csv').set_index('date')
2919     scaled_atturn = pd.read_csv(scaled_data_folder + 'at turn/at
        turn' + str(year) + '.csv').set_index('date')
2920     scaled_pcf = pd.read_csv(scaled_data_folder + 'pcf/pcf' + str(
        year) + '.csv').set_index('date')
2921     scaled_da = pd.read_csv(scaled_data_folder + 'debt asset/debt
        asset' + str(year) + '.csv').set_index('date')
2922     scaled_curr = pd.read_csv(scaled_data_folder + 'curr ratio/curr
        ratio' + str(year) + '.csv').set_index('date')
2923
2924     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
        year)+'/09/30',str(year)+'/12/31']
2925     df_ret = df_ret.loc[quarter_index, :]
2926     scaled_ret = scaled_ret.loc[quarter_index, :]
2927     scaled_mktcap = scaled_mktcap.loc[quarter_index, :]
2928     scaled_bm = scaled_bm.loc[quarter_index, :]
2929     scaled_roa = scaled_roa.loc[quarter_index, :]
2930     scaled_roe = scaled_roe.loc[quarter_index, :]
2931     scaled_accrual = scaled_accrual.loc[quarter_index, :]
2932     scaled_cfm = scaled_cfm.loc[quarter_index, :]
2933     scaled_eqinv = scaled_eqinv.loc[quarter_index, :]
2934     scaled_atturn = scaled_atturn.loc[quarter_index, :]
2935     scaled_pcf = scaled_pcf.loc[quarter_index, :]
2936     scaled_da = scaled_da.loc[quarter_index, :]
2937     scaled_curr = scaled_curr.loc[quarter_index, :]
2938
2939     BaseReturn3 = BaseReturn3.append(df_ret)
2940
2941     nt = wb = 1 / df_ret.shape[1]
2942
2943     Base_results = []
2944     Base_weights = []
2945     Base_SE = []
2946     init_points = list(BaseCoef3.iloc[-1,:].values)
2947
2948     for i in range(4):
2949         opt = scipy.optimize.minimize(

```

```

2950     PPS_base,
2951     init_points,
2952     method="BFGS",
2953     args=(
2954         wb,
2955         nt,
2956         scaled_ret.iloc[0 : i, :],
2957         scaled_mktcap.iloc[0 : i, :],
2958         scaled_bm.iloc[0 : i, :],
2959         scaled_roa.iloc[0 : i, :],
2960         scaled_roe.iloc[0 : i, :],
2961         scaled_accrual.iloc[0 : i, :],
2962         scaled_eqinv.iloc[0 : i, :],
2963         scaled_atturn.iloc[0 : i, :],
2964         scaled_cfm.iloc[0 : i, :],
2965         scaled_curr.iloc[0 : i, :],
2966         scaled_da.iloc[0 : i, :],
2967         scaled_pcf.iloc[0 : i, :],
2968         rr,
2969     ),
2970 )
2971 print("The {} window for year {}".format(i+1, year))
2972 print("The value:", opt["x"])
2973 Base_results.append(list(opt["x"]))
2974
2975 Base_SE.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
2976 weight = wb + nt * (
2977     + opt["x"][0] * scaled_mktcap.iloc[i, :]
2978     + opt["x"][1] * scaled_bm.iloc[i, :]
2979     + opt["x"][2] * scaled_roa.iloc[i, :]
2980     + opt["x"][3] * scaled_roe.iloc[i, :]
2981     + opt["x"][4] * scaled_accrual.iloc[i, :]
2982     + opt["x"][5] * scaled_eqinv.iloc[i, :]
2983     + opt["x"][6] * scaled_atturn.iloc[i, :]
2984     + opt["x"][7] * scaled_cfm.iloc[i, :]
2985     + opt["x"][8] * scaled_curr.iloc[i, :]
2986     + opt["x"][9] * scaled_da.iloc[i, :]
2987     + opt["x"][10] * scaled_pcf.iloc[i, :]
2988 )
2989 print(weight)
2990 Base_weights.append(weight)
2991
2992 BaseWeights3 = BaseWeights3.append(short_sell_constraints(pd.
DataFrame(Base_weights)))
2993 BaseCoef3 = BaseCoef3.append(pd.DataFrame(Base_results))
2994 BaseSE3 = BaseSE3.append(pd.DataFrame(Base_SE))
2995
2996 rr = 7
2997
2998 BaseWeights7 = pd.DataFrame()
2999 BaseReturn7 = pd.DataFrame()
3000
3001 BaseCoef7 = pd.DataFrame(np.zeros(11)).T
3002 BaseSE7 = pd.DataFrame()
3003
3004 year_list = range(1970, 2021)

```

```

3005
3006 for year in year_list:
3007
3008     df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
        set_index('date')
3009
3010     scaled_data_folder = './new standardized5/'
3011     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/scaled ret'
        + str(year) + '.csv').set_index('date')
3012     scaled_mktcap = pd.read_csv(scaled_data_folder + 'mktcap/mktcap'
        + str(year) + '.csv').set_index('date')
3013     scaled_bm = pd.read_csv(scaled_data_folder + 'bm/bm' + str(year)
        + '.csv').set_index('date')
3014     scaled_roa = pd.read_csv(scaled_data_folder + 'roa/roa' + str(
        year) + '.csv').set_index('date')
3015     scaled_roe = pd.read_csv(scaled_data_folder + 'roe/roe' + str(
        year) + '.csv').set_index('date')
3016     scaled_accrual = pd.read_csv(scaled_data_folder + 'accrual/
        accrual' + str(year) + '.csv').set_index('date')
3017     scaled_cfm = pd.read_csv(scaled_data_folder + 'cfm/cfm' + str(
        year) + '.csv').set_index('date')
3018     scaled_eqinv = pd.read_csv(scaled_data_folder + 'equity invcap/
        equity invcap' + str(year) + '.csv').set_index('date')
3019     scaled_atturn = pd.read_csv(scaled_data_folder + 'at turn/at
        turn' + str(year) + '.csv').set_index('date')
3020     scaled_pcf = pd.read_csv(scaled_data_folder + 'pcf/pcf' + str(
        year) + '.csv').set_index('date')
3021     scaled_da = pd.read_csv(scaled_data_folder + 'debt asset/debt
        asset' + str(year) + '.csv').set_index('date')
3022     scaled_curr = pd.read_csv(scaled_data_folder + 'curr ratio/curr
        ratio' + str(year) + '.csv').set_index('date')
3023
3024     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
        year)+'/09/30',str(year)+'/12/31']
3025     df_ret = df_ret.loc[quarter_index, :]
3026     scaled_ret = scaled_ret.loc[quarter_index, :]
3027     scaled_mktcap = scaled_mktcap.loc[quarter_index, :]
3028     scaled_bm = scaled_bm.loc[quarter_index, :]
3029     scaled_roa = scaled_roa.loc[quarter_index, :]
3030     scaled_roe = scaled_roe.loc[quarter_index, :]
3031     scaled_accrual = scaled_accrual.loc[quarter_index, :]
3032     scaled_cfm = scaled_cfm.loc[quarter_index, :]
3033     scaled_eqinv = scaled_eqinv.loc[quarter_index, :]
3034     scaled_atturn = scaled_atturn.loc[quarter_index, :]
3035     scaled_pcf = scaled_pcf.loc[quarter_index, :]
3036     scaled_da = scaled_da.loc[quarter_index, :]
3037     scaled_curr = scaled_curr.loc[quarter_index, :]
3038
3039     BaseReturn7 = BaseReturn7.append(df_ret)
3040
3041     nt = wb = 1 / df_ret.shape[1]
3042
3043     Base_results = []
3044     Base_weights = []
3045     Base_SE = []
3046     init_points = list(BaseCoef7.iloc[-1,:].values)

```



```

3047
3048     for i in range(4):
3049         opt = scipy.optimize.minimize(
3050             PPS_base,
3051             init_points,
3052             method="BFGS",
3053             args=(
3054                 wb,
3055                 nt,
3056                 scaled_ret.iloc[0 : i, :],
3057                 scaled_mktcap.iloc[0 : i, :],
3058                 scaled_bm.iloc[0 : i, :],
3059                 scaled_roa.iloc[0 : i, :],
3060                 scaled_roe.iloc[0 : i, :],
3061                 scaled_accrual.iloc[0 : i, :],
3062                 scaled_eqinv.iloc[0 : i, :],
3063                 scaled_atturn.iloc[0 : i, :],
3064                 scaled_cfm.iloc[0 : i, :],
3065                 scaled_curr.iloc[0 : i, :],
3066                 scaled_da.iloc[0 : i, :],
3067                 scaled_pcf.iloc[0 : i, :],
3068                 rr,
3069             ),
3070         )
3071         print("The {} window for year {}".format(i+1, year))
3072         print("The value:", opt["x"])
3073         Base_results.append(list(opt["x"]))
3074
3075         Base_SE.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
3076         weight = wb + nt * (
3077             + opt["x"][0] * scaled_mktcap.iloc[i, :]
3078             + opt["x"][1] * scaled_bm.iloc[i, :]
3079             + opt["x"][2] * scaled_roa.iloc[i, :]
3080             + opt["x"][3] * scaled_roe.iloc[i, :]
3081             + opt["x"][4] * scaled_accrual.iloc[i, :]
3082             + opt["x"][5] * scaled_eqinv.iloc[i, :]
3083             + opt["x"][6] * scaled_atturn.iloc[i, :]
3084             + opt["x"][7] * scaled_cfm.iloc[i, :]
3085             + opt["x"][8] * scaled_curr.iloc[i, :]
3086             + opt["x"][9] * scaled_da.iloc[i, :]
3087             + opt["x"][10] * scaled_pcf.iloc[i, :]
3088         )
3089         print(weight)
3090         Base_weights.append(weight)
3091
3092         BaseWeights7 = BaseWeights7.append(short_sell_constraints(pd.
DataFrame(Base_weights)))
3093         BaseCoef7 = BaseCoef7.append(pd.DataFrame(Base_results))
3094         BaseSE7 = BaseSE7.append(pd.DataFrame(Base_SE))
3095
3096     rr = 9
3097
3098     BaseWeights9 = pd.DataFrame()
3099     BaseReturn9 = pd.DataFrame()
3100
3101     BaseCoef9 = pd.DataFrame(np.zeros(11)).T

```

```

3102 BaseSE9 = pd.DataFrame()
3103
3104 year_list = range(1970, 2021)
3105
3106 for year in year_list:
3107
3108     df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
        set_index('date')
3109
3110     scaled_data_folder = './new standardized5/'
3111     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/scaled ret'
        + str(year) + '.csv').set_index('date')
3112     scaled_mktcap = pd.read_csv(scaled_data_folder + 'mktcap/mktcap'
        + str(year) + '.csv').set_index('date')
3113     scaled_bm = pd.read_csv(scaled_data_folder + 'bm/bm' + str(year)
        + '.csv').set_index('date')
3114     scaled_roa = pd.read_csv(scaled_data_folder + 'roa/roa' + str(
        year) + '.csv').set_index('date')
3115     scaled_roe = pd.read_csv(scaled_data_folder + 'roe/roe' + str(
        year) + '.csv').set_index('date')
3116     scaled_accrual = pd.read_csv(scaled_data_folder + 'accrual/
        accrual' + str(year) + '.csv').set_index('date')
3117     scaled_cfm = pd.read_csv(scaled_data_folder + 'cfm/cfm' + str(
        year) + '.csv').set_index('date')
3118     scaled_eqinv = pd.read_csv(scaled_data_folder + 'equity invcap/
        equity invcap' + str(year) + '.csv').set_index('date')
3119     scaled_atturn = pd.read_csv(scaled_data_folder + 'at turn/at
        turn' + str(year) + '.csv').set_index('date')
3120     scaled_pcf = pd.read_csv(scaled_data_folder + 'pcf/pcf' + str(
        year) + '.csv').set_index('date')
3121     scaled_da = pd.read_csv(scaled_data_folder + 'debt asset/debt
        asset' + str(year) + '.csv').set_index('date')
3122     scaled_curr = pd.read_csv(scaled_data_folder + 'curr ratio/curr
        ratio' + str(year) + '.csv').set_index('date')
3123
3124     quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(
        year)+'/09/30',str(year)+'/12/31']
3125     df_ret = df_ret.loc[quarter_index, :]
3126     scaled_ret = scaled_ret.loc[quarter_index, :]
3127     scaled_mktcap = scaled_mktcap.loc[quarter_index, :]
3128     scaled_bm = scaled_bm.loc[quarter_index, :]
3129     scaled_roa = scaled_roa.loc[quarter_index, :]
3130     scaled_roe = scaled_roe.loc[quarter_index, :]
3131     scaled_accrual = scaled_accrual.loc[quarter_index, :]
3132     scaled_cfm = scaled_cfm.loc[quarter_index, :]
3133     scaled_eqinv = scaled_eqinv.loc[quarter_index, :]
3134     scaled_atturn = scaled_atturn.loc[quarter_index, :]
3135     scaled_pcf = scaled_pcf.loc[quarter_index, :]
3136     scaled_da = scaled_da.loc[quarter_index, :]
3137     scaled_curr = scaled_curr.loc[quarter_index, :]
3138
3139     BaseReturn9 = BaseReturn9.append(df_ret)
3140
3141     nt = wb = 1 / df_ret.shape[1]
3142
3143     Base_results = []

```

```

3144 Base_weights = []
3145 Base_SE = []
3146 init_points = list(BaseCoef9.iloc[-1,:].values)
3147
3148 for i in range(4):
3149     opt = scipy.optimize.minimize(
3150         PPS_base,
3151         init_points,
3152         method="BFGS",
3153         args=(
3154             wb,
3155             nt,
3156             scaled_ret.iloc[0 : i, :],
3157             scaled_mktcap.iloc[0 : i, :],
3158             scaled_bm.iloc[0 : i, :],
3159             scaled_roa.iloc[0 : i, :],
3160             scaled_roe.iloc[0 : i, :],
3161             scaled_accrual.iloc[0 : i, :],
3162             scaled_eqinv.iloc[0 : i, :],
3163             scaled_atturn.iloc[0 : i, :],
3164             scaled_cfm.iloc[0 : i, :],
3165             scaled_curr.iloc[0 : i, :],
3166             scaled_da.iloc[0 : i, :],
3167             scaled_pcf.iloc[0 : i, :],
3168             rr,
3169         ),
3170     )
3171     print("The {} window for year {}".format(i+1, year))
3172     print("The value:", opt["x"])
3173     Base_results.append(list(opt["x"]))
3174
3175     Base_SE.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
3176     weight = wb + nt * (
3177         + opt["x"][0] * scaled_mktcap.iloc[i, :]
3178         + opt["x"][1] * scaled_bm.iloc[i, :]
3179         + opt["x"][2] * scaled_roa.iloc[i, :]
3180         + opt["x"][3] * scaled_roe.iloc[i, :]
3181         + opt["x"][4] * scaled_accrual.iloc[i, :]
3182         + opt["x"][5] * scaled_eqinv.iloc[i, :]
3183         + opt["x"][6] * scaled_atturn.iloc[i, :]
3184         + opt["x"][7] * scaled_cfm.iloc[i, :]
3185         + opt["x"][8] * scaled_curr.iloc[i, :]
3186         + opt["x"][9] * scaled_da.iloc[i, :]
3187         + opt["x"][10] * scaled_pcf.iloc[i, :]
3188     )
3189     print(weight)
3190     Base_weights.append(weight)
3191
3192     BaseWeights9 = BaseWeights9.append(short_sell_constraints(pd.
DataFrame(Base_weights)))
3193     BaseCoef9 = BaseCoef9.append(pd.DataFrame(Base_results))
3194     BaseSE9 = BaseSE9.append(pd.DataFrame(Base_SE))
3195
3196
3197
3198 ### Risk Aversion PCA Cases

```

```

3199
3200 # pc = 2
3201
3202 rr = [1,3,7,9]
3203 year_list = range(1970, 2021)
3204
3205 for r in rr:
3206
3207     rr = r
3208     PCA2Weights = pd.DataFrame()
3209     PCA2Return = pd.DataFrame()
3210     PCA2SE = pd.DataFrame()
3211
3212     PCA2Coef = pd.DataFrame(np.zeros(2)).T
3213
3214     for year in year_list:
3215
3216         df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv')
3217         .set_index('date')
3218
3219         scaled_data_folder = './new standardized5/'
3220         scaled_PCA2_folder = './PCA Case/2 npc/'
3221
3222         scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + '
scaled ret' + str(year) + '.csv').set_index('date')
3223         scaled_component1 = pd.read_csv(scaled_PCA2_folder + str(
year) + '/component 1.csv').set_index('date')
3224         scaled_component2 = pd.read_csv(scaled_PCA2_folder + str(
year) + '/component 2.csv').set_index('date')
3225
3226         quarter_index = [str(year)+'/03/31', str(year)+'/06/30',
str(year)+'/09/30',str(year)+'/12/31']
3227         scaled_component1 = scaled_component1.loc[quarter_index, :]
3228         scaled_component2 = scaled_component2.loc[quarter_index, :]
3229         df_ret = df_ret.loc[quarter_index, :]
3230
3231         scaled_component1 = pd.DataFrame(Scale(scaled_component1.T
)).T
3232         scaled_component2 = pd.DataFrame(Scale(scaled_component2.T
)).T
3233
3234         PCA2Return = PCA2Return.append(df_ret)
3235
3236         nt = wb = 1 / df_ret.shape[1]
3237
3238         PCA2_results = []
3239         PCA2_weights = []
3240         PCA2_se = []
3241         init_points = list(PCA2Coef.iloc[-1,:].values)
3242
3243         for i in range(4):
3244             opt = scipy.optimize.minimize(
PPS_pca_2,
3245             init_points,
3246             method="BFGS",
3247             args=(

```

```

3248         wb,
3249         nt,
3250         scaled_ret.iloc[0 : i, :],
3251         scaled_component1.iloc[0 : i, :],
3252         scaled_component2.iloc[0 : i, :],
3253         rr,
3254     ),
3255 )
3256 #         print("The {} window for year {}".format(i+1,
year))
3257 #         print("The value:", opt["x"])
3258 PCA2_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
3259 PCA2_results.append(list(opt["x"]))
3260 weight = wb + nt * (
3261     opt["x"][0] * scaled_component1.iloc[i, :]
3262     + opt["x"][1] * scaled_component2.iloc[i, :]
3263 )
3264 #         print(weight)
3265 PCA2_weights.append(weight)
3266
3267 PCA2Weights = PCA2Weights.append(short_sell_constraints(pd.
DataFrame(PCA2_weights)))
3268 PCA2Coef = PCA2Coef.append(pd.DataFrame(PCA2_results))
3269 PCA2SE = PCA2SE.append(PCA2_se)
3270
3271 print('----- RISK AVERSION = {} -----'.
format(r))
3272 print('Max weight = {}; Min weight = {}; Average weight = {}'.
format(PCA2Weights.max().max(),
3273
PCA2Weights.min().min(),
3274
PCA2Weights.mean().mean()))
3275 print('Coef 1 = {}, Coef 2 = {}'.format(PCA2Coef.mean()[0],
PCA2Coef.mean()[1]))
3276 print('SE 1 = {}, SE 2 = {}'.format(PCA2SE.mean()[0], PCA2SE.
mean()[1]))
3277 print('Average Return = {}'.format(cumulative_return(PCA2Return
, PCA2Weights).mean()*0.16))
3278 print('Standard deviation = {}'.format(np.nansum(PCA2Return.
values[1:]*PCA2Weights.values[:-1], axis=1).std()*(12**0.5)))
3279 print('Sharpe Ratio = {}'.format(((cumulative_return(PCA2Return
, PCA2Weights).mean()*0.16)-0.012)/(np.nansum(PCA2Return.values
[1:]*PCA2Weights.values[:-1], axis=1).std()*(12**0.5))))
3280
3281 # pc = 3
3282
3283 rr = [1,3,7,9]
3284 year_list = range(1970, 2021)
3285
3286 for r in rr:
3287
3288     PCA3Weights = pd.DataFrame()
3289     PCA3Return = pd.DataFrame()
3290     PCA3SE = pd.DataFrame()
3291

```

```

3292 PCA3Coef = pd.DataFrame(np.zeros(3)).T
3293 rr = r
3294
3295 for year in year_list:
3296
3297     df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv'
3298     ).set_index('date')
3299
3300     scaled_data_folder = './new standardized5/'
3301     scaled_PCA3_folder = './PCA Case/3 npc/'
3302
3303     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + '
3304     scaled_ret' + str(year) + '.csv').set_index('date')
3305     scaled_component1 = pd.read_csv(scaled_PCA3_folder + str(
3306     year) + '/component 1.csv').set_index('date')
3307     scaled_component2 = pd.read_csv(scaled_PCA3_folder + str(
3308     year) + '/component 2.csv').set_index('date')
3309     scaled_component3 = pd.read_csv(scaled_PCA3_folder + str(
3310     year) + '/component 3.csv').set_index('date')
3311
3312     quarter_index = [str(year)+'/03/31', str(year)+'/06/30',
3313     str(year)+'/09/30',str(year)+'/12/31']
3314     scaled_component1 = scaled_component1.loc[quarter_index, :]
3315     scaled_component2 = scaled_component2.loc[quarter_index, :]
3316     scaled_component3 = scaled_component3.loc[quarter_index, :]
3317     df_ret = df_ret.loc[quarter_index, :]
3318
3319     scaled_component1 = pd.DataFrame(Scale(scaled_component1.T
3320     )).T
3321     scaled_component2 = pd.DataFrame(Scale(scaled_component2.T
3322     )).T
3323     scaled_component3 = pd.DataFrame(Scale(scaled_component3.T
3324     )).T
3325
3326     PCA3Return = PCA3Return.append(df_ret)
3327
3328     nt = wb = 1 / df_ret.shape[1]
3329
3330     PCA3_results = []
3331     PCA3_weights = []
3332     PCA3_se = []
3333     init_points = list(PCA3Coef.iloc[-1,:].values)
3334
3335     for i in range(4):
3336         opt = scipy.optimize.minimize(
3337             PPS_pca_3,
3338             init_points,
3339             method="BFGS",
3340             args=(
3341                 wb,
3342                 nt,
3343                 scaled_ret.iloc[0 : i, :],
3344                 scaled_component1.iloc[0 : i, :],
3345                 scaled_component2.iloc[0 : i, :],
3346                 scaled_component3.iloc[0 : i, :],

```

```

3339         rr,
3340     ),
3341 )
3342 #     print("The {} window for year {}".format(i+1, year))
3343 #     print("The value:", opt["x"])
3344
3345     PCA3_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
3346     PCA3_results.append(list(opt["x"]))
3347     weight = wb + nt * (
3348         opt["x"][0] * scaled_component1.iloc[i, :]
3349         + opt["x"][1] * scaled_component2.iloc[i, :]
3350         + opt["x"][2] * scaled_component3.iloc[i, :]
3351     )
3352 #     print(weight)
3353     PCA3_weights.append(weight)
3354
3355     PCA3Weights = PCA3Weights.append(short_sell_constraints(pd.
DataFrame(PCA3_weights)))
3356     PCA3Coef = PCA3Coef.append(pd.DataFrame(PCA3_results))
3357     PCA3SE = PCA3SE.append(pd.DataFrame(PCA3_se))
3358
3359     print('----- RISK AVERSION = {} -----'.
format(r))
3360     print('Max weight = {}; Min weight = {}; Average weight = {}'.
format(PCA3Weights.max().max(),
3361
        PCA3Weights.min().min(),
3362
        PCA3Weights.mean().mean()))
3363     print('Coef 1 = {}, Coef 2 = {}, Coef 3 = {}'.format(PCA3Coef.
mean()[0], PCA3Coef.mean()[1], PCA3Coef.mean()[2]))
3364     print('SE 1 = {}, SE 2 = {}, SE 3 = {}'.format(PCA3SE.mean()
[0], PCA3SE.mean()[1], PCA3SE.mean()[2]))
3365     print('Average Return = {}'.format(cumulative_return(PCA3Return
, PCA3Weights).mean()*0.16))
3366     print('Standard deviation = {}'.format(np.nansum(PCA3Return.
values[1:]*PCA3Weights.values[:-1], axis=1).std()*(12**0.5)))
3367     print('Sharpe Ratio = {}'.format(((cumulative_return(PCA3Return
, PCA3Weights).mean()*0.16)-0.012)/(np.nansum(PCA3Return.values
[1:]*PCA3Weights.values[:-1], axis=1).std()*(12**0.5))))
3368
3369 # pc = 4
3370
3371 rr = [1,3,7,9]
3372
3373 for r in rr:
3374
3375     rr = r
3376     PCA4Weights = pd.DataFrame()
3377     PCA4Return = pd.DataFrame()
3378     PCA4SE = pd.DataFrame()
3379
3380     PCA4Coef = pd.DataFrame(np.zeros(4)).T
3381
3382     for year in year_list:
3383

```

```

3384         df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv'
3385                                ).set_index('date')
3386
3387         scaled_data_folder = './new standardized5/'
3388         scaled_PCA4_folder = './PCA Case/4 npc/'
3389
3390         scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + '
3391         scaled_ret' + str(year) + '.csv').set_index('date')
3392         scaled_component1 = pd.read_csv(scaled_PCA4_folder + str(
3393         year) + '/component 1.csv').set_index('date')
3394         scaled_component2 = pd.read_csv(scaled_PCA4_folder + str(
3395         year) + '/component 2.csv').set_index('date')
3396         scaled_component3 = pd.read_csv(scaled_PCA4_folder + str(
3397         year) + '/component 3.csv').set_index('date')
3398         scaled_component4 = pd.read_csv(scaled_PCA4_folder + str(
3399         year) + '/component 4.csv').set_index('date')
3400
3401         quarter_index = [str(year)+'/03/31', str(year)+'/06/30',
3402         str(year)+'/09/30',str(year)+'/12/31']
3403         scaled_component1 = scaled_component1.loc[quarter_index, :]
3404         scaled_component2 = scaled_component2.loc[quarter_index, :]
3405         scaled_component3 = scaled_component3.loc[quarter_index, :]
3406         scaled_component4 = scaled_component4.loc[quarter_index, :]
3407         df_ret = df_ret.loc[quarter_index, :]
3408
3409         scaled_component1 = pd.DataFrame(Scale(scaled_component1.T
3410         )).T
3411         scaled_component2 = pd.DataFrame(Scale(scaled_component2.T
3412         )).T
3413         scaled_component3 = pd.DataFrame(Scale(scaled_component3.T
3414         )).T
3415         scaled_component4 = pd.DataFrame(Scale(scaled_component4.T
3416         )).T
3417
3418         PCA4Return = PCA4Return.append(df_ret)
3419
3420         nt = wb = 1 / df_ret.shape[1]
3421
3422         PCA4_results = []
3423         PCA4_weights = []
3424         PCA4_se = []
3425         init_points = list(PCA4Coef.iloc[-1,:].values)
3426
3427         for i in range(4):
3428             opt = scipy.optimize.minimize(
3429                 PPS_pca_4,
3430                 init_points,
3431                 method="BFGS",
3432                 args=(
3433                     wb,
3434                     nt,
3435                     scaled_ret.iloc[0 : i, :],
3436                     scaled_component1.iloc[0 : i, :],
3437                     scaled_component2.iloc[0 : i, :],
3438                     scaled_component3.iloc[0 : i, :],
3439                     scaled_component4.iloc[0 : i, :],

```



```

3429         rr,
3430     ),
3431 )
3432 #     print("The {} window for year {}".format(i+1, year))
3433 #     print("The value:", opt["x"])
3434 PCA4_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
3435
3436 PCA4_results.append(list(opt["x"]))
3437 weight = wb + nt * (
3438     opt["x"][0] * scaled_component1.iloc[i, :]
3439     + opt["x"][1] * scaled_component2.iloc[i, :]
3440     + opt["x"][2] * scaled_component3.iloc[i, :]
3441     + opt["x"][3] * scaled_component4.iloc[i, :]
3442 )
3443 #     print(weight)
3444 PCA4_weights.append(weight)
3445
3446 PCA4Weights = PCA4Weights.append(short_sell_constraints(pd.
DataFrame(PCA4_weights)))
3447 PCA4Coef = PCA4Coef.append(pd.DataFrame(PCA4_results))
3448 PCA4SE = PCA4SE.append(pd.DataFrame(PCA4_se))
3449
3450 print('----- RISK AVERSION = {} -----'.
format(r))
3451 print('Max weight = {}; Min weight = {}; Average weight = {}'.
format(PCA4Weights.max().max(),
3452
PCA4Weights.min().min(),
3453
PCA4Weights.mean().mean()))
3454 print('Coef 1 = {}, Coef 2 = {}, Coef 3 = {}, Coef 4 = {}'.
format(PCA4Coef.mean()[0],
3455
PCA4Coef.mean()[1],
3456
PCA4Coef.mean()[2],
3457
PCA4Coef.mean()[3]))
3458 print('SE 1 = {}, SE 2 = {}, SE 3 = {}, SE 4 = {}'.format(
PCA4SE.mean()[0],
3459
PCA4SE.mean()
[1],
3460
PCA4SE.mean()
[2],
3461
PCA4SE.mean()
[3]))
3462 print('Average Return = {}'.format(cumulative_return(PCA4Return
, PCA4Weights).mean()*0.16))
3463 print('Standard deviation = {}'.format(np.nansum(PCA4Return.
values[1:]*PCA4Weights.values[:-1], axis=1).std()*(12**0.5)))
3464 print('Sharpe Ratio = {}'.format(((cumulative_return(PCA4Return
, PCA4Weights).mean()*0.16)-0.012)/(np.nansum(PCA4Return.values
[1:]*PCA4Weights.values[:-1], axis=1).std()*(12**0.5))))
3465
3466 # pc = 5
3467 rr = [1,3,7,9]

```

```

3468
3469 for r in rr:
3470
3471     PCA5Weights = pd.DataFrame()
3472     PCA5Return = pd.DataFrame()
3473     PCA5SE = pd.DataFrame()
3474
3475     PCA5Coef = pd.DataFrame(np.zeros(5)).T
3476     rr = r
3477
3478     for year in year_list:
3479
3480         df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv')
3481         .set_index('date')
3482
3483         scaled_data_folder = './new standardized5/'
3484         scaled_PCA5_folder = './PCA Case/5 npc/'
3485
3486         scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + '
scaled ret' + str(year) + '.csv').set_index('date')
3487         scaled_component1 = pd.read_csv(scaled_PCA5_folder + str(
year) + '/component 1.csv').set_index('date')
3488         scaled_component2 = pd.read_csv(scaled_PCA5_folder + str(
year) + '/component 2.csv').set_index('date')
3489         scaled_component3 = pd.read_csv(scaled_PCA5_folder + str(
year) + '/component 3.csv').set_index('date')
3490         scaled_component4 = pd.read_csv(scaled_PCA5_folder + str(
year) + '/component 4.csv').set_index('date')
3491         scaled_component5 = pd.read_csv(scaled_PCA5_folder + str(
year) + '/component 5.csv').set_index('date')
3492
3493         quarter_index = [str(year)+'/03/31', str(year)+'/06/30',
str(year)+'/09/30',str(year)+'/12/31']
3494         scaled_component1 = scaled_component1.loc[quarter_index, :]
3495         scaled_component2 = scaled_component2.loc[quarter_index, :]
3496         scaled_component3 = scaled_component3.loc[quarter_index, :]
3497         scaled_component4 = scaled_component4.loc[quarter_index, :]
3498         scaled_component5 = scaled_component5.loc[quarter_index, :]
3499         df_ret = df_ret.loc[quarter_index, :]
3500
3501         scaled_component1 = pd.DataFrame(Scale(scaled_component1.T
)).T
3502         scaled_component2 = pd.DataFrame(Scale(scaled_component2.T
)).T
3503         scaled_component3 = pd.DataFrame(Scale(scaled_component3.T
)).T
3504         scaled_component4 = pd.DataFrame(Scale(scaled_component4.T
)).T
3505         scaled_component5 = pd.DataFrame(Scale(scaled_component5.T
)).T
3506
3507         PCA5Return = PCA5Return.append(df_ret)
3508
3509         nt = wb = 1 / df_ret.shape[1]
3510
3511         PCA5_results = []

```

```

3511     PCA5_weights = []
3512     PCA5_se = []
3513     init_points = list(PCA5Coef.iloc[-1,:].values)
3514
3515     for i in range(4):
3516         opt = scipy.optimize.minimize(
3517             PPS_pca_5,
3518             init_points,
3519             method="BFGS",
3520             args=(
3521                 wb,
3522                 nt,
3523                 scaled_ret.iloc[0 : i, :],
3524                 scaled_component1.iloc[0 : i, :],
3525                 scaled_component2.iloc[0 : i, :],
3526                 scaled_component3.iloc[0 : i, :],
3527                 scaled_component4.iloc[0 : i, :],
3528                 scaled_component5.iloc[0 : i, :],
3529                 rr,
3530             ),
3531         )
3532     #     print("The {} window for year {}".format(i+1, year))
3533     #     print("The value:", opt["x"])
3534     PCA5_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
3535     PCA5_results.append(list(opt["x"]))
3536     weight = wb + nt * (
3537         opt["x"][0] * scaled_component1.iloc[i, :]
3538         + opt["x"][1] * scaled_component2.iloc[i, :]
3539         + opt["x"][2] * scaled_component3.iloc[i, :]
3540         + opt["x"][3] * scaled_component4.iloc[i, :]
3541         + opt["x"][4] * scaled_component5.iloc[i, :]
3542     )
3543     #     print(weight)
3544     PCA5_weights.append(weight)
3545
3546     PCA5Weights = PCA5Weights.append(short_sell_constraints(pd.
DataFrame(PCA5_weights)))
3547     PCA5Coef = PCA5Coef.append(pd.DataFrame(PCA5_results))
3548     PCA5SE = PCA5SE.append(pd.DataFrame(PCA5_se))
3549     print('----- RISK AVERSION = {} -----'.
format(r))
3550     print('Max weight = {}; Min weight = {}; Average weight = {}'.
format(PCA5Weights.max().max(),
3551
PCA5Weights.min().min(),
3552
PCA5Weights.mean().mean()))
3553     print('Coef 1 = {}, Coef 2 = {}, Coef 3 = {}, Coef 4 = {}, Coef
5 = {}'.format(PCA5Coef.mean()[0],
3554
PCA5Coef.mean()[1],
3555
PCA5Coef.mean()[2],
3556
PCA5Coef.mean()[3],
3557

```

```

PCA5Coef.mean()[4]))
3558 print('SE 1 = {}, SE 2 = {}, SE 3 = {}, SE 4 = {}, SE 5 = {}'.
format(PCA5SE.mean()[0],
3559 PCA5SE.mean()
[1],
3560 PCA5SE.mean()
[2],
3561 PCA5SE.mean()
[3],
3562 PCA5SE.mean()
[4]))
3563 print('Average Return = {}'.format(cumulative_return(PCA5Return
, PCA5Weights).mean()*0.16))
3564 print('Standard deviation = {}'.format(np.nanstd(PCA5Return.
values[1:]*PCA5Weights.values[:-1], axis=1).std()*(12**0.5)))
3565 print('Sharpe Ratio = {}'.format(((cumulative_return(PCA5Return
, PCA5Weights).mean()*0.16)-0.012)/(np.nanstd(PCA5Return.values
[1:]*PCA5Weights.values[:-1], axis=1).std()*(12**0.5))))
3566
3567 # pc = 6
3568 rr = [1,3,7,9]
3569
3570 for r in rr:
3571     PCA6Weights = pd.DataFrame()
3572     PCA6Return = pd.DataFrame()
3573     PCA6SE = pd.DataFrame()
3574
3575     PCA6Coef = pd.DataFrame(np.zeros(6)).T
3576     rr = r
3577
3578     for year in year_list:
3579
3580         df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv'
).set_index('date')
3581
3582         scaled_data_folder = './new standardized5/'
3583         scaled_PCA6_folder = './PCA Case/6 npc/'
3584
3585         scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + '
scaled ret' + str(year) + '.csv').set_index('date')
3586         scaled_component1 = pd.read_csv(scaled_PCA6_folder + str(
year) + '/component 1.csv').set_index('date')
3587         scaled_component2 = pd.read_csv(scaled_PCA6_folder + str(
year) + '/component 2.csv').set_index('date')
3588         scaled_component3 = pd.read_csv(scaled_PCA6_folder + str(
year) + '/component 3.csv').set_index('date')
3589         scaled_component4 = pd.read_csv(scaled_PCA6_folder + str(
year) + '/component 4.csv').set_index('date')
3590         scaled_component5 = pd.read_csv(scaled_PCA6_folder + str(
year) + '/component 5.csv').set_index('date')
3591         scaled_component6 = pd.read_csv(scaled_PCA6_folder + str(
year) + '/component 6.csv').set_index('date')
3592
3593         quarter_index = [str(year)+'/03/31', str(year)+'/06/30',
str(year)+'/09/30',str(year)+'/12/31']
3594         scaled_component1 = scaled_component1.loc[quarter_index, :]

```

```

3595     scaled_component2 = scaled_component2.loc[quarter_index, :]
3596     scaled_component3 = scaled_component3.loc[quarter_index, :]
3597     scaled_component4 = scaled_component4.loc[quarter_index, :]
3598     scaled_component5 = scaled_component5.loc[quarter_index, :]
3599     scaled_component6 = scaled_component6.loc[quarter_index, :]
3600     df_ret = df_ret.loc[quarter_index, :]
3601
3602     scaled_component1 = pd.DataFrame(Scale(scaled_component1.T
3603 ))).T
3604     scaled_component2 = pd.DataFrame(Scale(scaled_component2.T
3605 ))).T
3606     scaled_component3 = pd.DataFrame(Scale(scaled_component3.T
3607 ))).T
3608     scaled_component4 = pd.DataFrame(Scale(scaled_component4.T
3609 ))).T
3610     scaled_component5 = pd.DataFrame(Scale(scaled_component5.T
3611 ))).T
3612     scaled_component6 = pd.DataFrame(Scale(scaled_component6.T
3613 ))).T
3614
3615     PCA6Return = PCA6Return.append(df_ret)
3616
3617     nt = wb = 1 / df_ret.shape[1]
3618
3619     PCA6_results = []
3620     PCA6_weights = []
3621     PCA6_se = []
3622     init_points = list(PCA6Coef.iloc[-1,:].values)
3623
3624     for i in range(4):
3625         opt = scipy.optimize.minimize(
3626             PPS_pca_6,
3627             init_points,
3628             method="BFGS",
3629             args=(
3630                 wb,
3631                 nt,
3632                 scaled_ret.iloc[0 : i, :],
3633                 scaled_component1.iloc[0 : i, :],
3634                 scaled_component2.iloc[0 : i, :],
3635                 scaled_component3.iloc[0 : i, :],
3636                 scaled_component4.iloc[0 : i, :],
3637                 scaled_component5.iloc[0 : i, :],
3638                 scaled_component6.iloc[0 : i, :],
3639                 rr,
3640             ),
3641         )
3642         print("The {} window for year {}".format(i+1, year))
3643         print("The value:", opt["x"])
3644         PCA6_results.append(list(opt["x"]))
3645         PCA6_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
3646
3647         weight = wb + nt * (
3648             opt["x"][0] * scaled_component1.iloc[i, :]
3649             + opt["x"][1] * scaled_component2.iloc[i, :]
3650             + opt["x"][2] * scaled_component3.iloc[i, :]

```

```

3645         + opt["x"][3] * scaled_component4.iloc[i, :]
3646         + opt["x"][4] * scaled_component5.iloc[i, :]
3647         + opt["x"][5] * scaled_component6.iloc[i, :]
3648     )
3649 #     print(weight)
3650     PCA6_weights.append(weight)
3651
3652     PCA6Weights = PCA6Weights.append(short_sell_constraints(pd.
DataFrame(PCA6_weights)))
3653     PCA6Coef = PCA6Coef.append(pd.DataFrame(PCA6_results))
3654     PCA6SE = PCA6SE.append(pd.DataFrame(PCA6_se))
3655
3656     print('----- RISK AVERSION = {} -----'.
format(r))
3657     print('Max weight = {}; Min weight = {}; Average weight = {}'.
format(PCA6Weights.max().max(),
3658
PCA6Weights.min().min(),
3659
PCA6Weights.mean().mean()))
3660     print('Coef 1 = {}, Coef 2 = {}, Coef 3 = {}, Coef 4 = {}, Coef
5 = {}, Coef 6 = {}'.format(PCA6Coef.mean()[0],
3661
PCA6Coef.mean()[1],
3662
PCA6Coef.mean()[2],
3663
PCA6Coef.mean()[3],
3664
PCA6Coef.mean()[4],
3665
PCA6Coef.mean()[5]))
3666     print('SE 1 = {}, SE 2 = {}, SE 3 = {}, SE 4 = {}, SE 5 = {},
SE 6 = {}'.format(PCA6SE.mean()[0],
3667
PCA6SE.mean()
[1],
3668
PCA6SE.mean()
[2],
3669
PCA6SE.mean()
[3],
3670
PCA6SE.mean()
[4],
3671
PCA6SE.mean()
[5]))
3672     print('Average Return = {}'.format(cumulative_return(PCA6Return
, PCA5Weights).mean()*0.16))
3673     print('Standard deviation = {}'.format(np.nansum(PCA6Return.
values[1:]*PCA6Weights.values[:-1], axis=1).std()*(12**0.5)))
3674     print('Sharpe Ratio = {}'.format(((cumulative_return(PCA6Return
, PCA6Weights).mean()*0.16) - 0.012)/(np.nansum(PCA6Return.values
[1:]*PCA6Weights.values[:-1], axis=1).std()*(12**0.5))))
3675
3676 # pc = 7
3677
3678 rr = [1,3,7,9]
3679

```

```

3680 for r in rr:
3681
3682     PCA7Weights = pd.DataFrame()
3683     PCA7Return = pd.DataFrame()
3684     PCA7SE = pd.DataFrame()
3685
3686     PCA7Coef = pd.DataFrame(np.zeros(7)).T
3687     rr = r
3688
3689     for year in year_list:
3690
3691         df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv',
3692                               ).set_index('date')
3693
3694         scaled_data_folder = './new standardized5/'
3695         scaled_PCA7_folder = './PCA Case/7 npc/'
3696
3697         scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + '
3698         scaled_ret' + str(year) + '.csv').set_index('date')
3699         scaled_component1 = pd.read_csv(scaled_PCA7_folder + str(
3700         year) + '/component 1.csv').set_index('date')
3701         scaled_component2 = pd.read_csv(scaled_PCA7_folder + str(
3702         year) + '/component 2.csv').set_index('date')
3703         scaled_component3 = pd.read_csv(scaled_PCA7_folder + str(
3704         year) + '/component 3.csv').set_index('date')
3705         scaled_component4 = pd.read_csv(scaled_PCA7_folder + str(
3706         year) + '/component 4.csv').set_index('date')
3707         scaled_component5 = pd.read_csv(scaled_PCA7_folder + str(
3708         year) + '/component 5.csv').set_index('date')
3709         scaled_component6 = pd.read_csv(scaled_PCA7_folder + str(
3710         year) + '/component 6.csv').set_index('date')
3711         scaled_component7 = pd.read_csv(scaled_PCA7_folder + str(
3712         year) + '/component 7.csv').set_index('date')
3713
3714         quarter_index = [str(year)+'/03/31', str(year)+'/06/30',
3715         str(year)+'/09/30',str(year)+'/12/31']
3716         scaled_component1 = scaled_component1.loc[quarter_index, :]
3717         scaled_component2 = scaled_component2.loc[quarter_index, :]
3718         scaled_component3 = scaled_component3.loc[quarter_index, :]
3719         scaled_component4 = scaled_component4.loc[quarter_index, :]
3720         scaled_component5 = scaled_component5.loc[quarter_index, :]
3721         scaled_component6 = scaled_component6.loc[quarter_index, :]
3722         scaled_component7 = scaled_component7.loc[quarter_index, :]
3723         df_ret = df_ret.loc[quarter_index, :]
3724
3725         scaled_component1 = pd.DataFrame(Scale(scaled_component1.T
3726         )),.T
3727         scaled_component2 = pd.DataFrame(Scale(scaled_component2.T
3728         )),.T
3729         scaled_component3 = pd.DataFrame(Scale(scaled_component3.T
3730         )),.T
3731         scaled_component4 = pd.DataFrame(Scale(scaled_component4.T
3732         )),.T
3733         scaled_component5 = pd.DataFrame(Scale(scaled_component5.T
3734         )),.T

```

```

3721     scaled_component6 = pd.DataFrame(Scale(scaled_component6.T
3722     scaled_component7 = pd.DataFrame(Scale(scaled_component7.T
3723     ))).T
3724     scaled_component7 = pd.DataFrame(Scale(scaled_component7.T
3725     ))).T
3726
3727     PCA7Return = PCA7Return.append(df_ret)
3728
3729     nt = wb = 1 / df_ret.shape[1]
3730
3731     PCA7_results = []
3732     PCA7_weights = []
3733     PCA7_se = []
3734     init_points = list(PCA7Coef.iloc[-1,:].values)
3735
3736     for i in range(4):
3737         opt = scipy.optimize.minimize(
3738             PPS_pca_7,
3739             init_points,
3740             method="BFGS",
3741             args=(
3742                 wb,
3743                 nt,
3744                 scaled_ret.iloc[0 : i, :],
3745                 scaled_component1.iloc[0 : i, :],
3746                 scaled_component2.iloc[0 : i, :],
3747                 scaled_component3.iloc[0 : i, :],
3748                 scaled_component4.iloc[0 : i, :],
3749                 scaled_component5.iloc[0 : i, :],
3750                 scaled_component6.iloc[0 : i, :],
3751                 scaled_component7.iloc[0 : i, :],
3752                 rr,
3753             ),
3754         )
3755         # print("The {} window for year {}".format(i+1, year))
3756         # print("The value:", opt["x"])
3757         PCA7_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
3758
3759         PCA7_results.append(list(opt["x"]))
3760         weight = wb + nt * (
3761             opt["x"][0] * scaled_component1.iloc[i, :]
3762             + opt["x"][1] * scaled_component2.iloc[i, :]
3763             + opt["x"][2] * scaled_component3.iloc[i, :]
3764             + opt["x"][3] * scaled_component4.iloc[i, :]
3765             + opt["x"][4] * scaled_component5.iloc[i, :]
3766             + opt["x"][5] * scaled_component6.iloc[i, :]
3767             + opt["x"][6] * scaled_component7.iloc[i, :]
3768         )
3769         # print(weight)
3770         PCA7_weights.append(weight)
3771
3772         PCA7Weights = PCA7Weights.append(short_sell_constraints(pd.
3773         DataFrame(PCA7_weights)))
3774         PCA7Coef = PCA7Coef.append(pd.DataFrame(PCA7_results))
3775         PCA7SE = PCA7SE.append(pd.DataFrame(PCA7_se))
3776
3777         print('----- RISK AVERSION = {} -----'.
3778         format(r))

```



```

3773     print('Max weight = {}; Min weight = {}; Average weight = {}'.
3774           format(PCA7Weights.max().max(),
3775                  PCA7Weights.min().min(),
3776                  PCA7Weights.mean().mean()))
3777     print('Coef 1 = {}, Coef 2 = {}, Coef 3 = {}, Coef 4 = {}, Coef
3778           5 = {}, Coef 6 = {}, Coef 7 = {}'.format(PCA7Coef.mean()[0],
3779           PCA7Coef.mean()[1],
3780           PCA7Coef.mean()[2],
3781           PCA7Coef.mean()[3],
3782           PCA7Coef.mean()[4],
3783           PCA7Coef.mean()[5],
3784           PCA7Coef.mean()[6]))
3785     print('SE 1 = {}, SE 2 = {}, SE 3 = {}, SE 4 = {}, SE 5 = {},
3786           SE 6 = {}, SE 7 = {}'.format(PCA7SE.mean()[0],
3787           PCA7SE.mean()[1],
3788           PCA7SE.mean()[2],
3789           PCA7SE.mean()[3],
3790           PCA7SE.mean()[4],
3791           PCA7SE.mean()[5],
3792           PCA7SE.mean()[6]))
3793     print('Average Return = {}'.format(cumulative_return(PCA7Return
3794     , PCA7Weights).mean()*0.16))
3795     print('Standard deviation = {}'.format(np.nansum(PCA7Return.
3796     values[1:]*PCA7Weights.values[:-1], axis=1).std()*(12**0.5)))
3797     print('Sharpe Ratio = {}'.format(((cumulative_return(PCA7Return
3798     , PCA7Weights).mean()*0.16)-0.012)/(np.nansum(PCA7Return.values
3799     [1:]*PCA7Weights.values[:-1], axis=1).std()*(12**0.5))))
3800 # pc = 8
3801 rr = [1,3,7,9]
3802 for r in rr:
3803     rr = r
3804     PCA8Weights = pd.DataFrame()
3805     PCA8Return = pd.DataFrame()
3806     PCA8SE = pd.DataFrame()
3807     PCA8Coef = pd.DataFrame(np.zeros(8)).T
3808     for year in year_list:

```

```

3808
3809     df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv'
3810                          ).set_index('date')
3811
3812     scaled_data_folder = './new standardized5/'
3813     scaled_PCA8_folder = './PCA Case/8 mpc/'
3814
3815     scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + '
3816                             scaled ret' + str(year) + '.csv').set_index('date')
3817     scaled_component1 = pd.read_csv(scaled_PCA8_folder + str(
3818     year) + '/component 1.csv').set_index('date')
3819     scaled_component2 = pd.read_csv(scaled_PCA8_folder + str(
3820     year) + '/component 2.csv').set_index('date')
3821     scaled_component3 = pd.read_csv(scaled_PCA8_folder + str(
3822     year) + '/component 3.csv').set_index('date')
3823     scaled_component4 = pd.read_csv(scaled_PCA8_folder + str(
3824     year) + '/component 4.csv').set_index('date')
3825     scaled_component5 = pd.read_csv(scaled_PCA8_folder + str(
3826     year) + '/component 5.csv').set_index('date')
3827     scaled_component6 = pd.read_csv(scaled_PCA8_folder + str(
3828     year) + '/component 6.csv').set_index('date')
3829     scaled_component7 = pd.read_csv(scaled_PCA8_folder + str(
3830     year) + '/component 7.csv').set_index('date')
3831     scaled_component8 = pd.read_csv(scaled_PCA8_folder + str(
3832     year) + '/component 8.csv').set_index('date')
3833
3834     quarter_index = [str(year)+'/03/31', str(year)+'/06/30',
3835                      str(year)+'/09/30',str(year)+'/12/31']
3836     scaled_component1 = scaled_component1.loc[quarter_index, :]
3837     scaled_component2 = scaled_component2.loc[quarter_index, :]
3838     scaled_component3 = scaled_component3.loc[quarter_index, :]
3839     scaled_component4 = scaled_component4.loc[quarter_index, :]
3840     scaled_component5 = scaled_component5.loc[quarter_index, :]
3841     scaled_component6 = scaled_component6.loc[quarter_index, :]
3842     scaled_component7 = scaled_component7.loc[quarter_index, :]
3843     scaled_component8 = scaled_component8.loc[quarter_index, :]
3844     df_ret = df_ret.loc[quarter_index, :]
3845
3846     scaled_component1 = pd.DataFrame(Scale(scaled_component1.T
3847     )).T
3848     scaled_component2 = pd.DataFrame(Scale(scaled_component2.T
3849     )).T
3850     scaled_component3 = pd.DataFrame(Scale(scaled_component3.T
3851     )).T
3852     scaled_component4 = pd.DataFrame(Scale(scaled_component4.T
3853     )).T
3854     scaled_component5 = pd.DataFrame(Scale(scaled_component5.T
3855     )).T
3856     scaled_component6 = pd.DataFrame(Scale(scaled_component6.T
3857     )).T
3858     scaled_component7 = pd.DataFrame(Scale(scaled_component7.T
3859     )).T
3860     scaled_component8 = pd.DataFrame(Scale(scaled_component8.T
3861     )).T
3862
3863     PCA8Return = PCA8Return.append(df_ret)

```

```

3845
3846     nt = wb = 1 / df_ret.shape[1]
3847
3848     PCA8_results = []
3849     PCA8_weights = []
3850     PCA8_se = []
3851     init_points = list(PCA8Coef.iloc[-1,:].values)
3852
3853     for i in range(4):
3854         opt = scipy.optimize.minimize(
3855             PPS_pca_8,
3856             init_points,
3857             method="BFGS",
3858             args=(
3859                 wb,
3860                 nt,
3861                 scaled_ret.iloc[0 : i, :],
3862                 scaled_component1.iloc[0 : i, :],
3863                 scaled_component2.iloc[0 : i, :],
3864                 scaled_component3.iloc[0 : i, :],
3865                 scaled_component4.iloc[0 : i, :],
3866                 scaled_component5.iloc[0 : i, :],
3867                 scaled_component6.iloc[0 : i, :],
3868                 scaled_component7.iloc[0 : i, :],
3869                 scaled_component8.iloc[0 : i, :],
3870                 rr,
3871             ),
3872         )
3873     #     print("The {} window for year {}".format(i+1, year))
3874     #     print("The value:", opt["x"])
3875     PCA8_results.append(list(opt["x"]))
3876     PCA8_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
3877
3878     weight = wb + nt * (
3879         opt["x"][0] * scaled_component1.iloc[i, :]
3880         + opt["x"][1] * scaled_component2.iloc[i, :]
3881         + opt["x"][2] * scaled_component3.iloc[i, :]
3882         + opt["x"][3] * scaled_component4.iloc[i, :]
3883         + opt["x"][4] * scaled_component5.iloc[i, :]
3884         + opt["x"][5] * scaled_component6.iloc[i, :]
3885         + opt["x"][6] * scaled_component7.iloc[i, :]
3886         + opt["x"][7] * scaled_component8.iloc[i, :]
3887     )
3888     #     print(weight)
3889     PCA8_weights.append(weight)
3890
3891     PCA8Weights = PCA8Weights.append(short_sell_constraints(pd.
DataFrame(PCA8_weights)))
3892     PCA8Coef = PCA8Coef.append(pd.DataFrame(PCA8_results))
3893     PCA8SE = PCA8SE.append(pd.DataFrame(PCA8_se))
3894
3895     print('----- RISK AVERSION = {} -----'.
format(r))
3896     print('Max weight = {}; Min weight = {}; Average weight = {}'.
format(PCA8Weights.max().max(),

```

```

PCA8Weights.min().min(),
3898
    PCA8Weights.mean().mean()))
3899 print('Coef 1 = {}, Coef 2 = {}, Coef 3 = {}, Coef 4 = {}, Coef
    5 = {}, Coef 6 = {}, Coef 7 = {}, Coef 8 = {}'.format(PCA8Coef.
mean()[0],
3900
    PCA8Coef.mean()[1],
3901
    PCA8Coef.mean()[2],
3902
    PCA8Coef.mean()[3],
3903
    PCA8Coef.mean()[4],
3904
    PCA8Coef.mean()[5],
3905
    PCA8Coef.mean()[6],
3906
    PCA8Coef.mean()[7]))
3907 print('SE 1 = {}, SE 2 = {}, SE 3 = {}, SE 4 = {}, SE 5 = {},
SE 6 = {}, SE 7 = {}, SE 8 = {}'.format(PCA8SE.mean()[0],
3908
    PCA8SE.mean()
[1],
3909
    PCA8SE.mean()
[2],
3910
    PCA8SE.mean()
[3],
3911
    PCA8SE.mean()
[4],
3912
    PCA8SE.mean()
[5],
3913
    PCA8SE.mean()
[6],
3914
    PCA8SE.mean()
[7]))
3915 print('Average Return = {}'.format(cumulative_return(PCA8Return
, PCA8Weights).mean()*0.16))
3916 print('Standard deviation = {}'.format(np.nansum(PCA8Return.
values[1:]*PCA8Weights.values[:-1], axis=1).std()*(12**0.5)))
3917 print('Sharpe Ratio = {}'.format(((cumulative_return(PCA8Return
, PCA8Weights).mean()*0.16)-0.012)/(np.nansum(PCA8Return.values
[1:]*PCA8Weights.values[:-1], axis=1).std()*(12**0.5))))

```