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 (Aı¨t-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 returns[1], 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

[1]For 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 factors[2], 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 characteristic[3] 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 dimension[4], and the information related is thus overlapping. Secondly, it is also

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

[3] Zou 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).

[4] For 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 problems[5]. 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.

[5]For 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 WIFR[6], 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

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

$$\omega_{i,t} = \bar{\omega}_{i,t} + \frac{1}{N_t}\theta^{'}\hat{\boldsymbol{y}}_{i,t}, \tag{1}$$

where $\hat{\boldsymbol{y}}_{i,t}$ 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 $\theta^{'}\hat{\boldsymbol{y}}_{i,t}$ 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	*n*	*mean*	*Std.Dev*	*min*	*max*	*mean*	*Std.Dev*	*min*	*max*
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

$$\omega_{i,t} = \frac{\max\left[0, \omega_{i,t}\right]}{\sum_{t=1}^{N_t} \max\left[0, \omega_{i,t}\right]}. \tag{2}$$

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:

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

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

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

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

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

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

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 $\boldsymbol{\mathcal{X}}$ containing n characteristics:

$$\boldsymbol{\mathcal{X}}_{t\times n} = [\boldsymbol{x}_1 \;\; \boldsymbol{x}_2 \;\; \cdots \;\; \boldsymbol{x}_n]_t, \qquad t = 1, 2, ..., T \tag{7}$$

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

$$\boldsymbol{y} = A^T \boldsymbol{x}, \tag{8}$$

where A is the coefficient for each components:

$$A = [\boldsymbol{a}_1 \;\; \boldsymbol{a}_2 \;\; \cdots \;\; \boldsymbol{a}_n] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ . & . & \cdots & . \\ . & . & \cdots & . \\ . & . & \cdots & . \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \tag{9}$$

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

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

In our case, we select $m(m << n)$ 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:

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

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

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 $a_i^T \boldsymbol{x} (i = 2, 3, ..., 8)$ 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 0.049[7], 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-weight 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 and three-component strategies. Our approach of extracting information from characteristics outperform the value-weighted portfolio, however, fail to beat the equal-weighted

[7] The weight is multiplied by 100, hereafter wherever mentioned

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

	All Stocks			Top500 Stocks		
	(1)	(2)	(3)	(4)	(5)	(6)
Variables	Eq. Weighted	Val. Weighted	Optimized	Eq. Weighted	Val. Weighted	Optimized
θ_{mktcap}	-	-	82.49***	-	-	-105.19***
			(4.397)			(4.131)
$\theta_{equityinvcap}$	-	-	-165.90***	-	-	41.02***
			(2.323)			(3.341)
θ_{bm}	-	-	-8.56**	-	-	-96.63***
			(4.327)			(6.574)
θ_{pcf}	-	-	303.61***	-	-	41.99***
			(7.059)			(6.794)
$\theta_{accrual}$	-	-	82.64***	-	-	50.67***
			(2.835)			(4.60)
θ_{cfm}	-	-	-274.68***	-	-	101.16***
			(7.058)			(2.164)
θ_{roa}	-	-	4.70	-	-	-21.39***
			(3.253)			(9.112)
θ_{roe}	-	-	120.30***	-	-	-37.55***
			(7.056)			(5.979)
$\theta_{currratio}$	-	-	-304.64***	-	-	-79.8***
			(9.854)			(4.854)
$\theta_{debttoasset}$	-	-	-420.87***	-	-	-75.20***
			(9.636)			(3.563)
θ_{atturn}	-	-	250.31***	-	-	-146.25***
			(15.022)			(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	All Stocks						
Variables	$pc = 2$	$pc = 3$	$pc = 4$	$pc = 5$	$pc = 6$	$pc = 7$	$pc = 8$
θ_{pc1}	121.73***	133.10***	-126.98***	72.89***	-42.52***	-84.06***	90.41***
	(1.141)	(0.645)	(0.968)	(0.429)	(0.583)	(0.487)	(0.353)
θ_{pc2}	-379.51***	-333.63***	13.16***	-36.09***	-38.87***	82.22***	26.85***
	(0.758)	(0.609)	(0.581)	(0.419)	(0.560)	(0.422)	(0.467)
θ_{pc3}		41.93***	170.65***	70.73***	69.60***	17.66***	-33.17***
		(0.681)	(0.658)	(0.555)	(0.590)	(0.556)	(0.537)
θ_{pc4}			156.72***	53.16***	77.67***	25.06***	-24.51***
			(0.582)	(0.546)	(0.581)	(0.597)	(0.499)
θ_{pc5}				123.51***	-48.44***	-121.19***	31.44***
				(0.466)	(0.618)	(0.505)	(0.340)
θ_{pc6}					134.45***	65.63***	-124.16***
					(0.586)	(0.676)	(0.744)
θ_{pc7}						34.12***	182.29 ***
						(0.474)	(0.720)
θ_{pc8}							119.42***
							(0.396)
$\lvert w_i \rvert \times 100$	0.054	0.054	0.052	0.051	0.051	0.052	0.049
$\max \lvert w_i \rvert \times 100$	7.044	6.462	3.000	3.648	2.679	2.317	3.223
$\min \lvert w_i \rvert \times 100$	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
Sharpe Ratio	0.775	0.763	0.772	0.826	0.779	0.781	0.902
Panel B	Top500 Stocks						
Variable	$pc = 2$	$pc = 3$	$pc = 4$	$pc = 5$	$pc = 6$	$pc = 7$	$pc = 8$
θ_{pc1}	175.79***	-137.48***	-329.87***	-98.90***	-79.01***	-48.85***	-85.78***
	(2.970)	(1.286)	(1.129)	(0.898)	(0.845)	(0.730)	(0.756)
θ_{pc2}	315.25***	192.25***	91.14***	17.15***	48.71***	54.30***	60.60***
	(1.687)	(2.122)	(1.478)	(1.395)	(1.411)	(1.314)	(1.071)
θ_{pc3}		206.05***	120.83***	137.42***	101.43***	93.98***	96.67***
		(1.964)	(1.457)	(1.122)	(1.313)	(1.051)	(1.165)
θ_{pc4}			90.88***	-43.48***	-69.03***	-95.16***	-59.65***
			(1.237)	(1.331)	(1.169)	(0.927)	(1.144)
θ_{pc5}				53.24***	61.23***	45.37***	24.54***
				(1.242)	(1.241)	(0.958)	(0.987)
θ_{pc6}					-40.11***	-37.92***	-33.1***
					(1.193)	(1.077)	(1.021)
θ_{pc7}						28.96***	5.04***
						(0.957)	(0.964)
θ_{pc8}							12.09***
							(1.064)
$\lvert w_i \rvert \times 100$	0.257	0.252	0.262	0.248	0.247	0.243	0.248
$\max \lvert w_i \rvert \times 100$	13.73	12.65	19.81	12.65	10.98	5.814	11.33
$\min \lvert w_i \rvert \times 100$	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
Sharpe Ratio	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-sample[8].

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		Out-of-Sample	
	Val.Weighted	Opt.	Val.Weighted	Opt. Fcst.
θ_{mktcap}	-	2.024	-	2.024
		(1.609)		
$\theta_{equityinvcap}$	-	89.20***	-	89.20
		(1.109)		
θ_{bm}	-	10.20***	-	10.20
		(2.167)		
θ_{pcf}	-	26.36***	-	26.36
		(1.376)		
$\theta_{accrual}$	-	-24.38***	-	-24.38
		(2.030)		
θ_{cfm}	-	164.10***	-	164.10
		(1.609)		
θ_{roa}	-	2.739**	-	2.739
		(1.194)		
θ_{roe}	-	79.44***	-	79.44
		(1.492)		
$\theta_{currratio}$	-	-56.17***	-	-56.17
		(1.365)		
$\theta_{debtasset}$	-	-166.95***	-	-166.95
		(2.152)		
θ_{atturn}	-	-47.85***	-	-47.85
		(1.649)		
$\lvert w_i \rvert \times 100$	0.025	0.051	0.029	0.046
$\max \lvert w_i \rvert \times 100$	8.630	3.921	7.564	4.153
$\min \lvert w_i \rvert \times 100$	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
Sharpe Ratio	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

[8]However, 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	In-sample Estimation (Jan 1970 - Dec 1995)							
Variables	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***
		(1.158)	(0.645)	(0.968)	(0.429)	(0.583)	(0.487)	(0.353)
θ_{pc2}	-	-297.94***	-262.83***	-262.03***	-57.38***	-25.95***	67.45***	31.42***
		(1.108)	(1.003)	(0.822)	(0.537)	(0.706)	(0.700)	(0.704)
θ_{pc3}	-		22.25***	16.37***	49.16***	47.40***	22.04***	-2.27***
			(1.003)	(0.607)	(0.614)	(0.704)	(0.871)	(0.828)
θ_{pc4}	-			16.37***	56.89***	55.82***	38.79***	-20.83***
				(0.607)	(0.606)	(0.726)	(0.776)	(0.653)
θ_{pc5}	-				113.06***	-33.30***	-82.38***	7.53***
					(0.724)	(0.859)	(0.810)	(0.550)
θ_{pc6}	-					114.06***	55.43***	-33.64***
						(0.709)	(0.805)	(1.240)
θ_{pc7}	-						19.10***	84.23 ***
							(0.787)	(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
Sharpe Ratio.	0.384	0.440	0.419	0.426	0.423	0.442	0.403	0.452
Panel B	Out-of-Sample Forecast (Jan 1996 - Dec 2020)							
Variable	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
Sharpe Ratio	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 preference[9]. 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.

[9]Besides, 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
Sharpe Ratio	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
Sharpe Ratio	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
Sharpe Ratio	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
Sharpe Ratio	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 determine 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

Aı̈t-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

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.

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

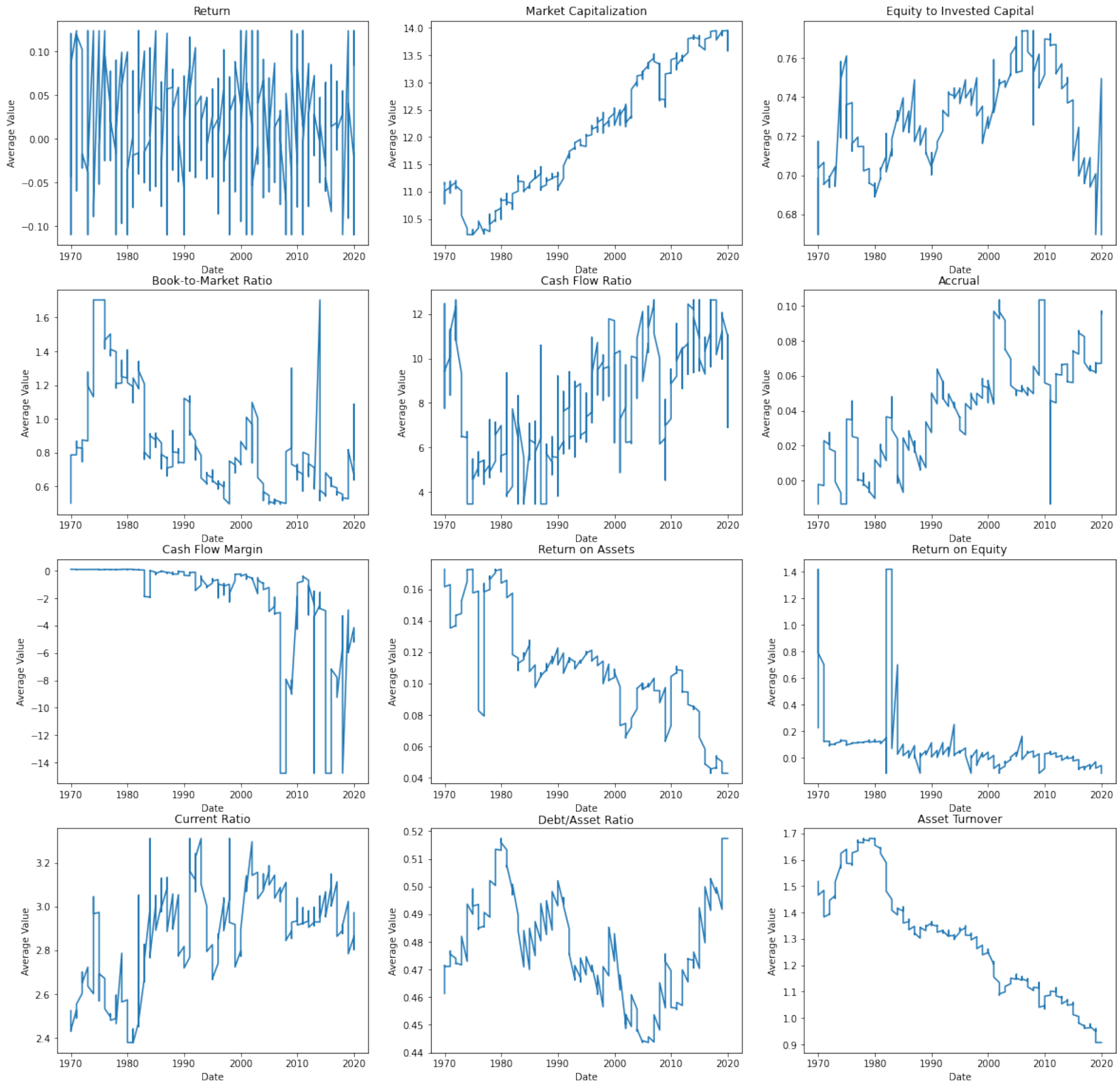
- Asset Turnover (*at turnover*) Ratio: defined as the total revenue divided by the average value of assets during the observation period. Asset turnover ratio measures the operating efficiency of a company by examining the utilization of its assets. Martani and Khairurizka's (2009) study shows that 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)

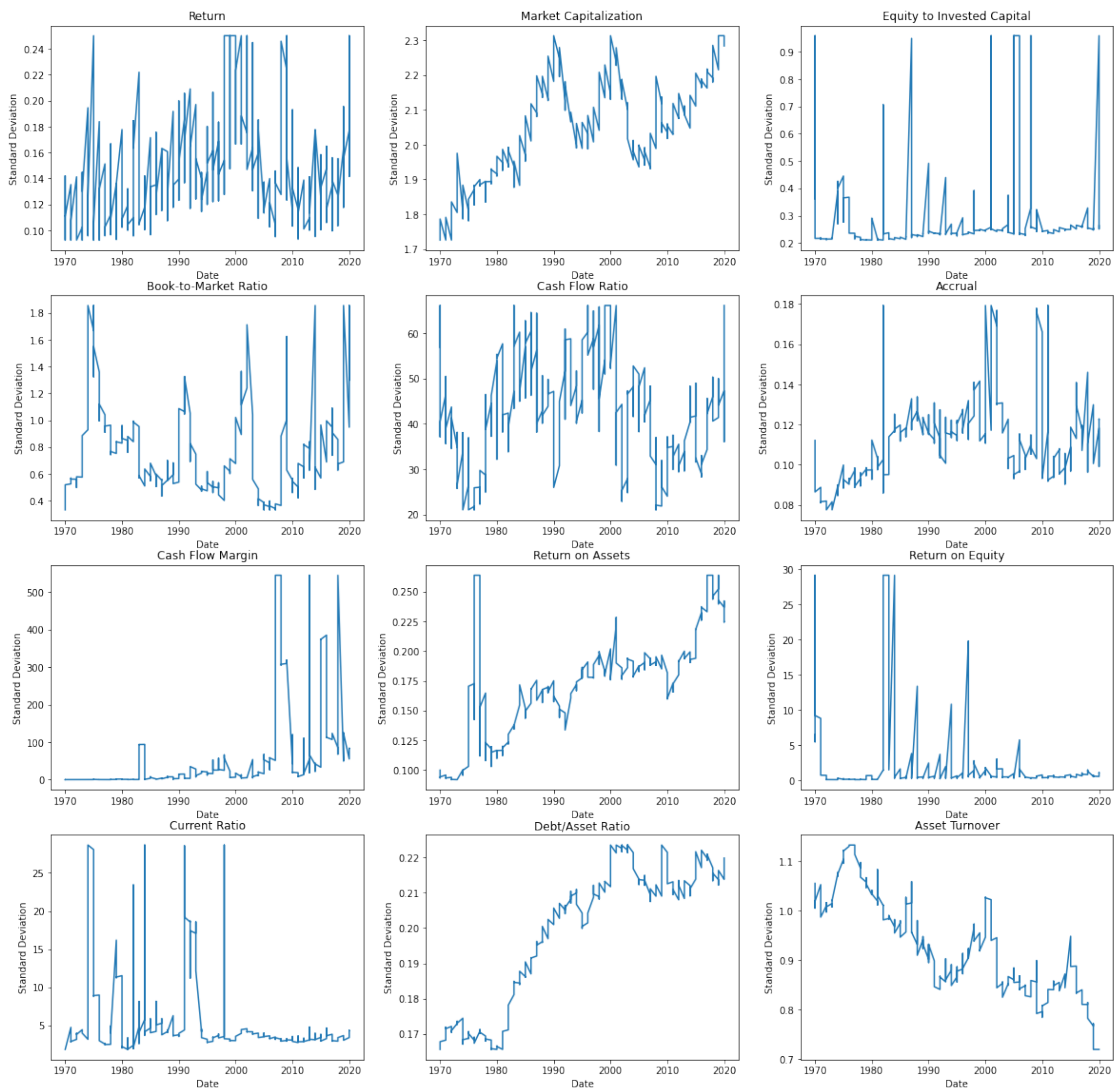
This figure displays the monthly cross-sectional means of returns and firm characteristics Market Capitalization, Equity to Invested Capital Ratio, Book-to-Market Ratio, Cash Flow Ratio, Accrual, Cash Flow Margin, Return on Assets, Return on Equity, Current Ratio, Debt/Asset Ratio, Asset Turnover Ratio 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(*curratio*), Debt/Asset Ratio(*debtto asset*), 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	-	2.024
		(1.609)		
$\theta_{equityinvcap}$	-	89.20***	-	89.20
		(1.109)		
θ_{bm}	-	10.20***	-	10.20
		(2.167)		
θ_{pcf}	-	26.36 ***	-	26.36
		(1.376)		
$\theta_{accrual}$	-	-24.38***	-	-24.38
		(2.030)		
θ_{cfm}	-	164.10***	-	164.10
		(1.609)		
θ_{roa}	-	2.739**	-	2.739
		(1.194)		
θ_{roe}	-	79.44***	-	79.44
		(1.492)		
$\theta_{curratio}$	-	-56.17***	-	-56.17
		(1.365)		
$\theta_{debttoasset}$	-	-166.95***	-	-166.95
		(2.152)		
θ_{atturn}	-	-47.85***	-	-47.85
		(1.649)		
$\|w_i\| \times 100$	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
Sharpe Ratio	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 $pc = n(n = 2, ..., 8)$. 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***	-	97.21
		(1.158)		
θ_{pc2}	-	-297.94***	-	-297.94
		(1.108)		
$\lvert w_i \rvert \times 100$	0.025	0.052	0.029	0.053
$\max \lvert w_i \rvert \times 100$	8.630	3.639	7.564	7.308
$\min \lvert w_i \rvert \times 100$	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
Sharpe Ratio	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***	-	104.43
		(0.899)		
θ_{pc2}	-	-262.83***	-	-262.83
		(1.003)		
$\theta_{pc3}.$	-	22.25***	-	22.25
		(1.003)		
$\lvert w_i \rvert \times 100$	0.025	0.052	0.029	0.053
$\max \lvert w_i \rvert \times 100$	8.630	3.747	7.564	4.280
$\min \lvert w_i \rvert \times 100$	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
Sharpe Ratio	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***	-	102.97
		(0.740)		
θ_{pc2}	-	-262.03***	-	-262.03
		(0.822)		
θ_{pc3}	-	16.37***	-	16.37
		(0.607)		
θ_{pc4}	-	16.37***	-	16.37
		(0.607)		
$\|w_i\| \times 100$	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
Sharpe Ratio	0.384	0.426	0.371	0.338

	In-Sample		Out-of-Sample	
	Val.Weighted	$pc = 5.$	Val.Weighted	$pc = 5$ Fcst.
θ_{pc1}	-	4.162***	-	4.162
		(0.470)		
θ_{pc2}	-	-57.38***	-	-57.38
		(0.537)		
θ_{pc3}	-	49.16***	-	49.16
		(0.614)		
θ_{pc4}	-	56.89***	-	56.88
		(0.606)		
θ_{pc5}	-	113.06***	-	113.06
		(0.724)		
$\|w_i\| \times 100$	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
Sharpe Ratio	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***	-	-10.20
		(0.614)		
θ_{pc2}	-	-25.95***	-	-25.95
		(0.706)		
θ_{pc3}	-	47.04***	-	47.04
		(0.704)		
θ_{pc4}	-	55.82***	-	55.82
		(0.726)		
θ_{pc5}	-	-33.30***	-	-33.30
		(0.859)		
θ_{pc6}	-	114.06***	-	114.06
		(0.709)		
$\lvert w_i \rvert \times 100$	0.025	0.050	0.029	0.048
$\max \lvert w_i \rvert \times 100$	8.630	2.679	7.564	2.759
$\min \lvert w_i \rvert \times 100$	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
Sharpe Ratio	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***	-	-47.98
		(0.562)		
θ_{pc2}	-	67.45***	-	67.45
		(0.700)		
θ_{pc3}	-	22.04***	-	22.04
		(0.871)		
θ_{pc4}	-	38.79***	-	38.79
		(0.776)		
θ_{pc5}	-	-82.38***	-	-82.38
		(0.810)		
θ_{pc6}	-	55.43***	-	55.43
		(0.805)		
θ_{pc7}	-	19.10***	-	19.10
		(0.787)		
$\lvert w_i \rvert \times 100$	0.025	0.051	0.029	0.050
$\max \lvert w_i \rvert \times 100$	8.630	2.317	7.564	1.832
$\min \lvert w_i \rvert \times 100$	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
Sharpe Ratio	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***	-	54.67
		(0.565)		
θ_{pc2}	-	31.42***	-	31.42
		(0.704)		
θ_{pc3}	-	-2.27***	-	-2.27
		(0.828)		
θ_{pc4}	-	-20.83***	-	-20.83
		(0.653)		
θ_{pc5}	-	7.53***	-	7.53
		(0.550)		
θ_{pc6}	-	-33.64***	-	-33.64
		(1.240)		
θ_{pc7}	-	84.23***	-	84.23
		(1.293)		
θ_{pc8}	-	65.00***	-	65.00
		(0.650)		
$\lvert w_i\rvert \times 100$	0.025	0.049	0.029	0.047
$\max \lvert w_i\rvert \times 100$	8.630	2.320	7.564	2.714
$\min \lvert w_i\rvert \times 100$	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
Sharpe Ratio	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	-175.70***	82.49***	-10051.10***	-406.10***
	(0.021)	(3.305)	(4.397)	(51.897)	(0.751)
θ_{bm}	0.00	585.48***	-8.56*	-580.02***	85.51***
	(0.021)	(3.732)	(4.327)	(4.296)	(0.578)
θ_{pcf}	0.00	135.92***	303.61***	-4701.94***	180.93***
	(0.021)	(2.171)	(7.059)	(24.855)	(0.460)
θ_{roa}	0.00	-198.82***	4.70	-1577.19***	-606.33***
	(0.021)	(1.727)	(3.253)	(10.898)	(0.482)
θ_{roe}	0.00	543.32***	120.30***	1497.80***	110.86***
	(0.021)	(2.928)	(7.056)	(7.389)	(0.278)
$\theta_{accrual}$	0.00	-60.20***	82.64***	-1492.13***	420.74***
	(0.021)	(2.116)	(2.835)	(5.659)	(1.204)
$\theta_{equityinvcap}$	0.00	-65.91***	-165.90***	2995.14***	292.91***
	(0.021)	(3.311)	(2.323)	(18.261)	(0.776)
θ_{atturn}	0.00	80.71***	250.31***	-3475.06***	102.06***
	(0.021)	(2.063)	(15.022)	(16.738)	(0.308)
θ_{cfm}	0.00	526.42***	-274.68***	-854.49***	428.27***
	(0.021)	(2.506)	(15.022)	(6.355)	(0.624)
$\theta_{debtasset}$	0.00	-218.79***	-420.87***	-97.66***	110.78***
	(0.021)	(2.430)	(9.636)	(3.985)	(0.270)
$\theta_{currratio}$	0.00	79.43***	-304.64***	-2532.78***	-305.16***
	(0.021)	(1.552)	(9.854)	(12.368)	(0.530)
$\lvert w_i \rvert \times 100$	0.046	0.056	0.049	0.056	0.058
$\max \lvert w_i \rvert \times 100$	0.076	4.674	4.731	3.841	6.212
$\min \lvert w_i \rvert \times 100$	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
Sharpe Ratio	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				
	$pc = 2$				
	$\gamma = 1$	$\gamma = 3$	$\gamma = 5$	$\gamma = 7$	$\gamma = 9$
θ_{pc1}	0.00	151.17***	121.73***	73.75***	61.87***
	(0.021)	(1.811)	(1.141)	(0.719)	(0.562)
θ_{pc2}	0.00	1327.27***	-379.51***	-277.47***	-240.01***
	(0.021)	(1.158)	(0.758)	(0.535)	(0.487)
$\lvert w_i\rvert \times 100$	0.076	0.056	0.054	0.055	0.055
$\max \lvert w_i\rvert \times 100$	0.045	4.674	7.044	7.153	7.216
$\min \lvert w_i\rvert \times 100$	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
Sharpe Ratio	0.944	0.867	0.775	0.750	0.750
	All Stocks				
	$pc = 3$				
	$\gamma = 1$	$\gamma = 3$	$\gamma = 5$	$\gamma = 7$	$\gamma = 9$
θ_{pc1}	0.00	363.85***	133.10***	76.94***	-10386.88***
	(0.021)	(1.448)	(0.645)	0.648)	(0.316)
θ_{pc2}	0.00	-840.13***	-333.63***	-250.34***	14370.29***
	(0.021)	(1.184)	(0.609)	(0.526)	(.497)
θ_{pc3}	0.00	72.83***	41.93***	9.92***	-5019.81***
	(0.021)	(1.346)	(0.681)	(0.580)	(0.223)
$\lvert w_i\rvert \times 100$	0.076	0.054	0.054	0.054	0.054
$\max \lvert w_i\rvert \times 100$	0.045	6.741	7.044	6.870	3.794
$\min \lvert w_i\rvert \times 100$	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
Sharpe Ratio	0.944	0.733	0.775	0.756	0.811

Table continued on next page...

Table continued from the previous page.

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

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

Table continued on next page...

Table continued from the previous page.

	All Stocks $pc = 6$ $\gamma = 1$	$\gamma = 3$	$\gamma = 5$	$\gamma = 7$	$\gamma = 9$
θ_{pc1}	0.00	-267.53***	-42.52***	1681.94***	-260.43***
	(0.021)	(0.848)	(0.583)	(0.419)	(0.725)
θ_{pc2}	0.00	210.87***	-38.87***	-15959.41***	-37.53***
	(0.021)	(1.043)	(0.560)	(0.590)	(0.306)
θ_{pc3}	0.00	14.81***	69.60***	12442.51***	-207.7***1
	(0.021)	(1.146)	(0.590)	(0.916)	(0.700)
θ_{pc4}	0.00	-12.80***	77.67***	5474.78***	197.13***
	(0.021)	(1.170)	(0.581)	(0.270)	(0.769)
θ_{pc5}	0.00	-235.22***	-48.44***	-11750.38***	-95.84***
	(0.021)	(1.181)	(0.618)	(0.464)	(0.484)
θ_{pc6}	0.00	240.54***	134.45***	-14731.39***	63.53***
	(0.021)	(1.448)	(0.586)	(0.723)	(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
Sharpe Ratio	0.944	0.785	0.779	0.786	0.752

Table continued on next page...

Table continued from the previous page.

	All Stocks $pc = 7$ $\gamma = 1$	 $\gamma = 3$	 $\gamma = 5$	 $\gamma = 7$	 $\gamma = 9$
θ_{pc1}	0.00	-257.53***	-84.06***	-142.85***	7.083***
	(0.021)	(0.947)	(0.487)	(0.692)	(0.305)
θ_{pc2}	0.00	20.43***	82.22***	25.07***	-24.12***
	(0.021)	(0.875)	(0.422)	(0.298)	(0.291)
θ_{pc3}	0.00	171.05***	17.66***	-169.79***	27.71***
	(0.021)	(1.193)	(0.556)	(0.737)	(0.300)
θ_{pc4}	0.00	181.02***	25.06***	-398.33***	4.62***
	(0.021)	(1.053)	(0.597)	(1.232)	(0.346)
θ_{pc5}	0.00	-287.25***	-121.19***	319.84***	-13.84***
	(0.021)	(0.981)	(0.505)	(1.123)	(0.276)
θ_{pc6}	0.00	37.00***	65.63***	194.40***	54.95***
	(0.021)	(1.410)	(0.676)	(0.818)	(0.285)
θ_{pc7}	0.00	-201.56***	34.12***	423.95***	-24.49***
	(0.021)	(1.061)	(0.474)	(1.527)	(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
Sharpe Ratio	0.944	0.841	0.781	0.807	0.808

Table continued on next page...

Table continued from the previous page.

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

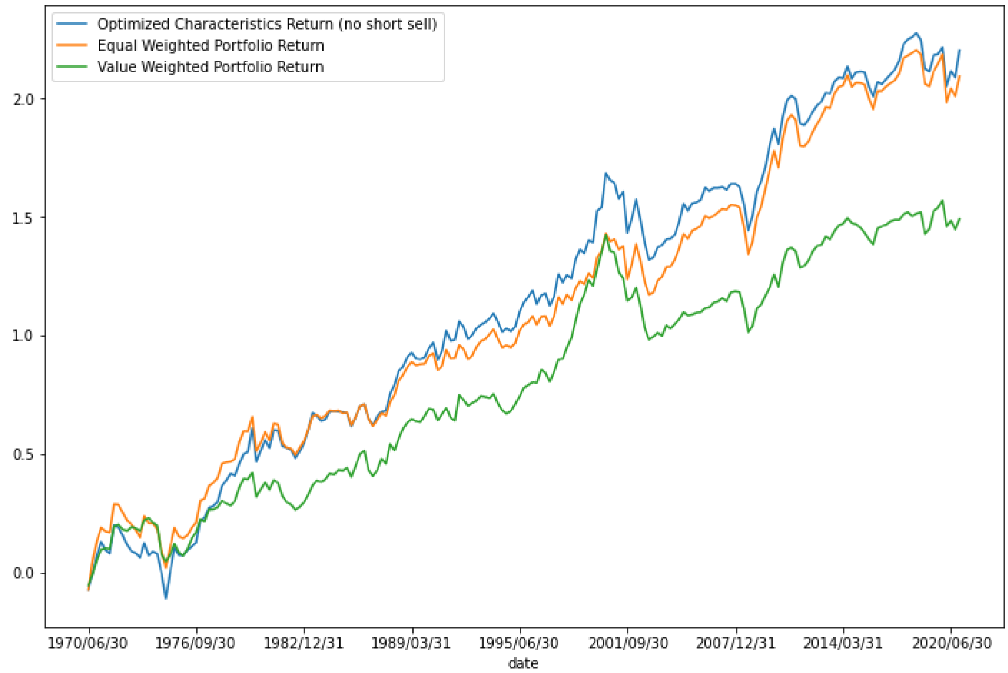


Figure 4.4: Principal Components Optimized Portfolio Cumulative Return, PC Optimized vs. Benchmarks, U.S. Stocks 1970-2020

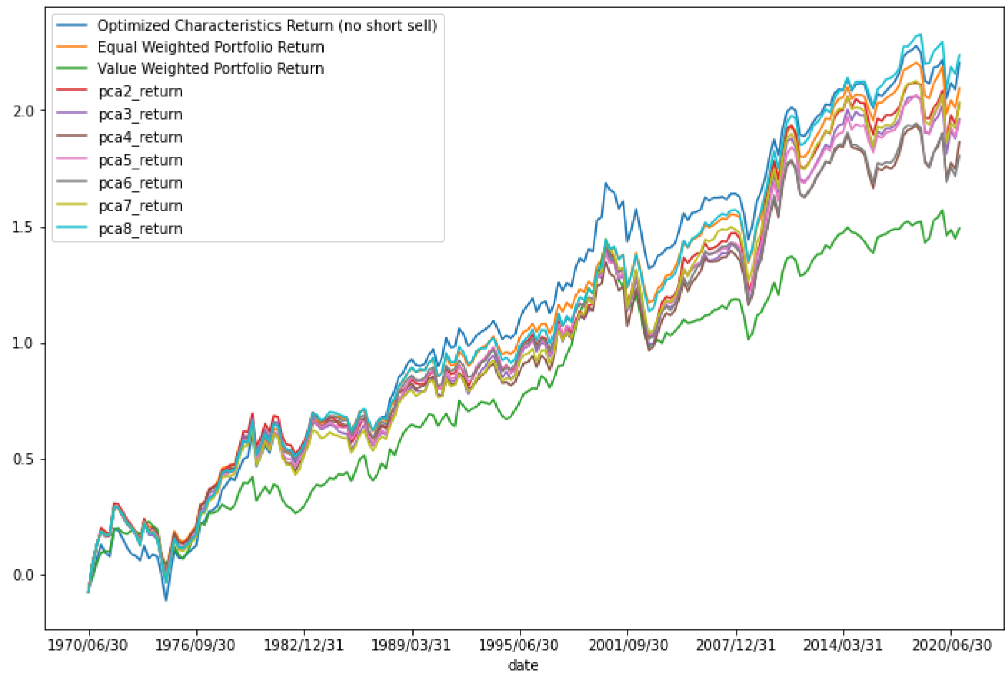


Figure 4.5: Optimized Portfolio Cumulative Return, All vs. Top500 Stocks, U.S. Stocks 1970-2020

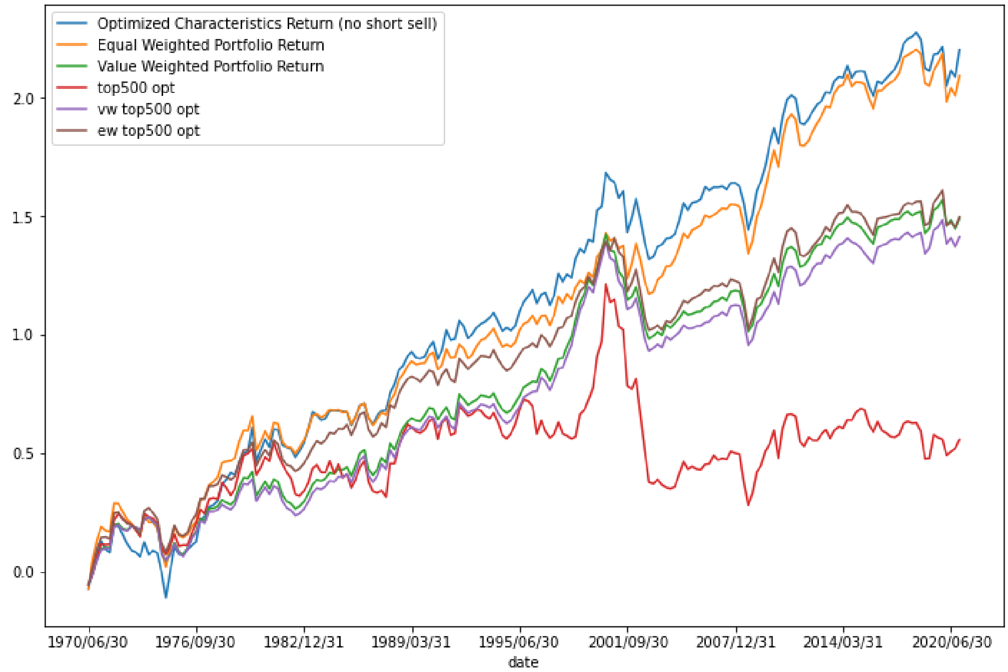


Figure 4.6: Principal Components Optimized Portfolio Cumulative Return, Top500 Stocks, U.S. Stocks 1970-2020

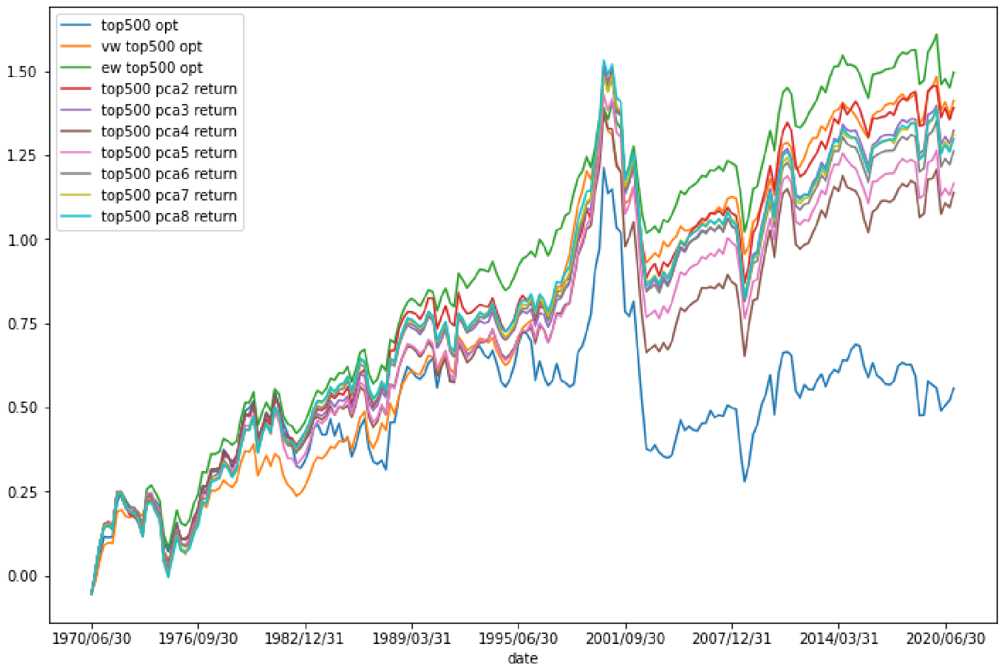
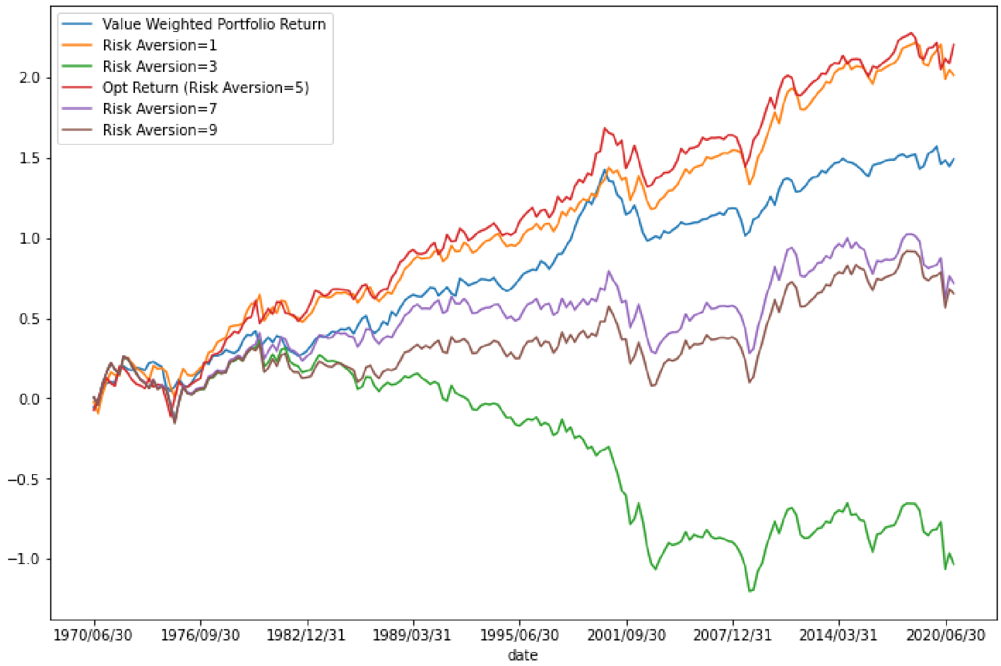


Figure 4.7: Optimized Portfolio Cumulative Return, Different Risk Aversion, U.S. Stocks 1970-2020



Appendix D Codes

```
### Building some useful tools

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
import os

import scipy
from scipy import optimize

def Scale(y, c=True, sc=True):

    x = y.copy()

    if c:
        x -= x.mean()
    if sc and c:
        x /= x.std()
    elif sc:
        x /= np.sqrt(x.pow(2).sum().div(x.count() - 1))
    return x

def check_shape(df1, df2):
    return df1.shape == df2.shape

def delete_company_with_na(df):

    have_null = df.columns[df.isna().any()]
    have_null.append(df.columns[df.isna().any()])
    result_df = df.drop(columns=have_null)

    return result_df

def reshape_dataframe(main_df, reshaped_df):

    n1 = main_df.shape[0]
    n2 = reshaped_df.shape[0]
    df = reshaped_df.iloc[(n2-n1)::,:]
    return df

def descriptive_statistics(df):

    d = {
        "Mean" : df.mean(axis=1),
        "St. Dev" : df.std(axis=1)
    }

    df = pd.DataFrame(d, index=df.index)
    return df

def q4_rolling_return(df):

    q4 = df.rolling(4).sum().dropna()
```

```
    return q4

def m12_rolling_return(df):

    m12 = df.rolling(12).sum().dropna()
    return m12

def find_common_firms(l1, l2):

    s1 = set(l1)
    s2 = set(l2)

    s_result = s1.intersection(s2)

    return list(s_result)

def short_sell_constraints(df):

    no_short_sell_weight = np.zeros(df.shape)
    z_w = np.zeros(df.shape)
    rows = df.shape[0]
    columns = df.shape[1]

    for i in range(rows):
        for j in range(columns):
            z_w[i][j] = max(df.iloc[i,j], 0)

    for i in range(rows):
        for j in range(columns):
            no_short_sell_weight[i][j] = z_w[i][j]/z_w[i].sum()

    no_short_sell_df = pd.DataFrame(no_short_sell_weight, index=df.
   index, columns=df.columns)
    return no_short_sell_df

def value_weights(df):
    w = np.zeros(df.shape)

    for i in range(df.shape[0]):
        for j in range(df.shape[1]):
            w[i][j] = df.iloc[i, j]/df.iloc[i,:].sum()

    value_weight = pd.DataFrame(w, index=df.index, columns=df.
   columns)

    return value_weight

def sharpe_ratio(ret_series, rf):
    SR = (ret_series - rf).mean()/ret_series.std()
    return SR


def make_year_month(df):
    df['year'] = pd.DatetimeIndex(df.index).year
    df['month'] = pd.DatetimeIndex(df.index).month
    return df
```

```python

def rebalancing(company_df, characteristics_df_list, year_list):

    rebalanced_universe = pd.DataFrame(index=year_list, columns=['Companies', 'Number of Companies'])

    for year in year_list:
        company_set = set(list(delete_company_with_na(company_df[company_df['year'] == year]).columns[:-2]))
        print("{} companies in {}".format(len(company_set), year))
        print("---------------------------------------")

        for characteristics_df in characteristics_df_list:
            characteristics_df = delete_company_with_na(characteristics_df[characteristics_df['year'] == year])
            characteristics_set = set(list(characteristics_df.columns[:-2]))
            print("{} companies for the char in {}".format(len(characteristics_set), year))

            company_set.intersection_update(characteristics_set)
            print("Finally, {} companies in {} after rebalancing".format(len(company_set), year))

        rebalanced_universe.loc[year, 'Companies'] = company_set
        number_of_companies = len(company_set)
        rebalanced_universe.loc[year, 'Number of Companies'] = number_of_companies

    return rebalanced_universe

def PPS_base(x, wb, nt, ret, mktcap, bm, roa, roe, accrual, eqinv, atturn, cfm, curr, da, pcf, rr):
    wi = wb + nt * (x[0] * mktcap + x[1] * bm + x[2] * roa + x[3] * roe + x[4] * accrual +
                    x[5] * eqinv + x[6] * atturn  + x[7] * cfm + x[8] * curr +
                    x[9] * da + x[10] * pcf)
    wret = (wi * ret).sum(axis=1)
    ut = ((1 + wret) ** (1 - rr)) / (1 - rr)
    u = -(ut.mean())
    return u

def PPS_pca_2(x, wb, nt, ret, component1, component2, rr):
    wi = wb + nt * (x[0] * component1 + x[1] * component2)
    wret = (wi * ret).sum(axis=1)
    ut = ((1 + wret) ** (1 - rr)) / (1 - rr)
    u = -(ut.mean())
    return u

def PPS_pca_3(x, wb, nt, ret, component1, component2, component3, rr):
    wi = wb + nt * (x[0] * component1 + x[1] * component2 + x[2] * component3)
    wret = (wi * ret).sum(axis=1)
    ut = ((1 + wret) ** (1 - rr)) / (1 - rr)
```

```
    u = -(ut.mean())
    return u

def PPS_pca_4(x, wb, nt, ret, component1, component2, component3, component4, rr):
    wi = wb + nt * (x[0] * component1 + x[1] * component2 + x[2] * component3 + x[3] * component4)
    wret = (wi * ret).sum(axis=1)
    ut = ((1 + wret) ** (1 - rr)) / (1 - rr)
    u = -(ut.mean())
    return u

def PPS_pca_5(x, wb, nt, ret, component1, component2, component3, component4, component5, rr):
    wi = wb + nt * (x[0] * component1 + x[1] * component2 + x[2] * component3 + x[3] * component4
                    + x[4] * component5)
    wret = (wi * ret).sum(axis=1)
    ut = ((1 + wret) ** (1 - rr)) / (1 - rr)
    u = -(ut.mean())
    return u

def PPS_pca_6(x, wb, nt, ret, component1, component2, component3, component4, component5, component6, rr):
    wi = wb + nt * (x[0] * component1 + x[1] * component2 + x[2] * component3 + x[3] * component4
                    + x[4] * component5 + x[5] * component6)
    wret = (wi * ret).sum(axis=1)
    ut = ((1 + wret) ** (1 - rr)) / (1 - rr)
    u = -(ut.mean())
    return u

def PPS_pca_7(x, wb, nt, ret, component1, component2, component3, component4, component5, component6, component7, rr):
    wi = wb + nt * (x[0] * component1 + x[1] * component2 + x[2] * component3 + x[3] * component4
                    + + x[4] * component5 + x[5] * component6 + + x[6] * component7)
    wret = (wi * ret).sum(axis=1)
    ut = ((1 + wret) ** (1 - rr)) / (1 - rr)
    u = -(ut.mean())
    return u

def PPS_pca_8(x, wb, nt, ret, component1, component2, component3, component4, component5, component6, component7,
              component8, rr):
    wi = wb + nt * (x[0] * component1 + x[1] * component2 + x[2] * component3 + x[3] * component4 +
                    + x[4] * component5 + x[5] * component6 + + x[6] * component7 + x[7] * component8)
    wret = (wi * ret).sum(axis=1)
    ut = ((1 + wret) ** (1 - rr)) / (1 - rr)
    u = -(ut.mean())
    return u

def PPS_pca_9(x, wb, nt, ret, component1, component2, component3,
```

```
    component4, component5, component6, component7,
                component8, component9, rr):
    wi = wb + nt * (x[0] * component1 + x[1] * component2 + x[2] *
    component3 + x[3] * component4 +
                    + x[4] * component5 + x[5] * component6 + + x[6]
     * component7 + x[7] * component8 +
                     x[8] * component9)
    wret = (wi * ret).sum(axis=1)
    ut = ((1 + wret) ** (1 - rr)) / (1 - rr)
    u = -(ut.mean())
    return u


def survive(df, number_of_year):

    survivor = []
    stock_list = df.columns
    for stock in stock_list:
        living_year = pd.to_datetime(df[stock].last_valid_index()).
    year - pd.to_datetime(df[stock].first_valid_index()).year
        living_month = pd.to_datetime(df[stock].last_valid_index())
    .month - pd.to_datetime(df[stock].first_valid_index()).month
        if (living_year >= number_of_year) & (living_month >= 0):
            survivor.append(stock)
    return survivor

def separate_for_pca(year_list, char_list, source_char_file_path='
    ./new standardized5/', yearlydata_path='./new Yearly Data/'):

    for year in year_list:
        if not os.path.exists(yearlydata_path+str(year)):
            os.makedirs(yearlydata_path+str(year))

    for year in year_list:
        stock_list = list(pd.read_csv(source_char_file_path + 'ret/
    scaled ret' + str(year) + '.csv').set_index('date').columns)

        for stock in stock_list:
            df = pd.DataFrame()

            for char in char_list:
                char_df = pd.read_csv(source_char_file_path+char+'/
    '+char+str(year)+'.csv').set_index('date')
                df[char] = char_df[stock]

                df.to_csv(yearlydata_path+str(year)+'/'+stock+'.csv
    ')
            print("Done for company {} in {}!".format(stock, year))



def cumulative_return(Weights, Return):

    cumulative_return = np.nansum(Return.values[1:] * Weights.
    values[:-1],axis=1).cumsum()
    return cumulative_return
```

```python

def top500(year_list, make_file=True):

    for year in year_list:
        df = pd.read_csv('./Investing Pool5/mktcap '+str(year)+'.csv').set_index('date')
        top500_index = df.iloc[0,:].sort_values(ascending=False).head(500).index

        df = df[top500_index]
        if make_file:
            df.to_csv('./top500/mktcap '+str(year)+'.csv')
        # return top500_index

def bootstrap_se(theta_sample, B=10000, size=300):

    theta_sample = theta_sample.values

    sample_mean = []
    for _ in range(B):
        sample_n = np.random.choice(theta_sample, size=size)
        sample_mean.append(sample_n.mean())

    se = np.std(sample_mean)/(B**0.5)
    return se

def make_bootstrap_sample(original_sample, B=10000, size=300):

    original_sample = original_sample.values

    new_sample = []
    for _ in range(B):
        sample_n = np.random.choice(original_sample, size=size)
        new_sample.append(sample_n)

    return new_sample

def statistic(returns, weights, coef, se):

    print("mean: {}, std: {}".format(returns.mean(), returns.std()))
    print("mean: {}, max: {}, min: {}".format(weights.mean().mean(), weights.max().max(), weights.min().min()))
    print("Coef: {}".format(coef.mean()))
    print("Se: {}".format(se.mean()))


def in_outofsample_statistic(insampleweight, insamplereturn, insamplecoef,
                                 insamplese, outsampleweight, outsamplereturn, rf=risk_free):
    print('Coef: {}'.format(insamplecoef.mean()))
    print('------------------------------------')
    print('Se: {}'.format(insamplese.mean()))
    print('------------------------------------')
    print('insample weight mean: {}'.format(insampleweight.mean().
```

```
mean()))
    print('insample weight min: {}'.format(insampleweight.min().min()))
    print('insample weight max: {}'.format(insampleweight.max().max()))
    print('-------------------------------------')
    print('outsample weight mean: {}'.format(outsampleweight.mean().mean()))
    print('outsample weight min: {}'.format(outsampleweight.min().min()))
    print('outsample weight max: {}'.format(outsampleweight.max().max()))
    print('-------------------------------------')
    insample_cum_return = cumulative_return(insampleweight, insamplereturn)
    outsample_cum_return = cumulative_return(outsampleweight, outsamplereturn)
    insample_cum_return_mean = insample_cum_return.mean()
    outsample_cum_return_mean = outsample_cum_return.mean()
    insample_cum_return_std = insample_cum_return.std()
    outsample_cum_return_std = outsample_cum_return.std()

    print("insample return mean {}, std {}".format(insample_cum_return_mean, insample_cum_return_std))
    print("outsample return mean {}, std {}".format(outsample_cums_return_mean, outsample_cum_return_std))


    insample_rf = rf[1:104]
    outsample_rf = rf[104:-1]

    print("insample SR: {} | outsample SR: {}".format(((insample_cum_return_mean-insample_rf.mean())/insample_cum_return_std),
                                                       ((outsample_cum_return_mean-outsample_rf.mean())/outsample_cum_return_std)))

### Base Case
## import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import random
import seaborn as sns
from datetime import datetime
from scipy.stats.mstats import winsorize

from Portfolio import *

# shortselling is not allow
BaseWeights = pd.DataFrame()
BaseReturn = pd.DataFrame()

BaseCoef = pd.DataFrame(np.zeros(11)).T
BaseSE = pd.DataFrame()
```

```python

year_list = range(1970, 2021)

for year in year_list:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/scaled ret' + str(year) + '.csv').set_index('date')
    scaled_mktcap = pd.read_csv(scaled_data_folder + 'mktcap/mktcap' + str(year) + '.csv').set_index('date')
    scaled_bm = pd.read_csv(scaled_data_folder + 'bm/bm' + str(year) + '.csv').set_index('date')
    scaled_roa = pd.read_csv(scaled_data_folder + 'roa/roa' + str(year) + '.csv').set_index('date')
    scaled_roe = pd.read_csv(scaled_data_folder + 'roe/roe' + str(year) + '.csv').set_index('date')
    scaled_accrual = pd.read_csv(scaled_data_folder + 'accrual/accrual' + str(year) + '.csv').set_index('date')
    scaled_cfm = pd.read_csv(scaled_data_folder + 'cfm/cfm' + str(year) + '.csv').set_index('date')
    scaled_eqinv = pd.read_csv(scaled_data_folder + 'equity invcap/equity invcap' + str(year) + '.csv').set_index('date')
    scaled_atturn = pd.read_csv(scaled_data_folder + 'at turn/at turn' + str(year) + '.csv').set_index('date')
    scaled_pcf = pd.read_csv(scaled_data_folder + 'pcf/pcf' + str(year) + '.csv').set_index('date')
    scaled_da = pd.read_csv(scaled_data_folder + 'debt asset/debt asset' + str(year) + '.csv').set_index('date')
    scaled_curr = pd.read_csv(scaled_data_folder + 'curr ratio/curr ratio' + str(year) + '.csv').set_index('date')

    quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(year)+'/09/30',str(year)+'/12/31']
    df_ret = df_ret.loc[quarter_index, :]
    scaled_ret = scaled_ret.loc[quarter_index, :]
    scaled_mktcap = scaled_mktcap.loc[quarter_index, :]
    scaled_bm = scaled_bm.loc[quarter_index, :]
    scaled_roa = scaled_roa.loc[quarter_index, :]
    scaled_roe = scaled_roe.loc[quarter_index, :]
    scaled_accrual = scaled_accrual.loc[quarter_index, :]
    scaled_cfm = scaled_cfm.loc[quarter_index, :]
    scaled_eqinv = scaled_eqinv.loc[quarter_index, :]
    scaled_atturn = scaled_atturn.loc[quarter_index, :]
    scaled_pcf = scaled_pcf.loc[quarter_index, :]
    scaled_da = scaled_da.loc[quarter_index, :]
    scaled_curr = scaled_curr.loc[quarter_index, :]

    BaseReturn = BaseReturn.append(df_ret)

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

    Base_results = []
    Base_weights = []
```

```
    Base_SE = []
    init_points = list(BaseCoef.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_base,
            init_points,
            method="BFGS",
            args=(
                wb,
                nt,
                scaled_ret.iloc[0 : i, :],
                scaled_mktcap.iloc[0 : i, :],
                scaled_bm.iloc[0 : i, :],
                scaled_roa.iloc[0 : i, :],
                scaled_roe.iloc[0 : i, :],
                scaled_accrual.iloc[0 : i, :],
                scaled_eqinv.iloc[0 : i, :],
                scaled_atturn.iloc[0 : i, :],
                scaled_cfm.iloc[0 : i, :],
                scaled_curr.iloc[0 : i, :],
                scaled_da.iloc[0 : i, :],
                scaled_pcf.iloc[0 : i, :],
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
        print("The value:", opt["x"])
        Base_results.append(list(opt["x"]))

        Base_SE.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
        weight = wb + nt * (
            + opt["x"][0] * scaled_mktcap.iloc[i, :]
            + opt["x"][1] * scaled_bm.iloc[i, :]
            + opt["x"][2] * scaled_roa.iloc[i, :]
            + opt["x"][3] * scaled_roe.iloc[i, :]
            + opt["x"][4] * scaled_accrual.iloc[i, :]
            + opt["x"][5] * scaled_eqinv.iloc[i, :]
            + opt["x"][6] * scaled_atturn.iloc[i, :]
            + opt["x"][7] * scaled_cfm.iloc[i, :]
            + opt["x"][8] * scaled_curr.iloc[i, :]
            + opt["x"][9] * scaled_da.iloc[i, :]
            + opt["x"][10] * scaled_pcf.iloc[i, :]
        )
        print(weight)
        Base_weights.append(weight)

    BaseWeights = BaseWeights.append(short_sell_constraints(pd.DataFrame(Base_weights)))
    BaseCoef = BaseCoef.append(pd.DataFrame(Base_results))
    BaseSE = BaseSE.append(pd.DataFrame(Base_SE))

# Value Weighted and Equal weighted Portfolio

vwWeight = pd.DataFrame()
ewWeight = pd.DataFrame()
```

```
for year in year_list:

    mktcap_source_file = './Investing Pool5/mktcap ' + str(year) +
    '.csv'

    mktcap_df = pd.read_csv(mktcap_source_file).set_index('date')
    mktcap_df = mktcap_df.iloc[[2,5,8,11], :]

    vwWeight = vwWeight.append(value_weights(mktcap_df))

for year in year_list:

    portfolio_source_file = './new char5/ret/ret' + str(year) + '.
    csv'

    portfolio_df = pd.read_csv(portfolio_source_file).set_index('
    date')
    portfolio_df = portfolio_df.iloc[[1,4,7,10], :]
    M = portfolio_df.shape[0]
    N = portfolio_df.shape[1]

    ewWeight = ewWeight.append(pd.DataFrame(np.ones((M,N))/N, index
    =portfolio_df.index, columns=portfolio_df.columns))

index = BaseReturn.index[1:]

portfolio_return_df = pd.DataFrame(index=index)

opt_return = np.nansum((BaseReturn.values[1:] * BaseWeights.values
    [:-1]), axis=1).cumsum()
ew_return = np.nansum((BaseReturn.values[1:] * ewWeight.values
    [:-1]), axis=1).cumsum()
vw_return = np.nansum((BaseReturn.values[1:] * vwWeight.values
    [:-1]), axis=1).cumsum()

portfolio_return_df['Optimized Characteristics Return (no short
    sell)'] = opt_return
portfolio_return_df['Equal Weighted Portfolio Return'] = ew_return
portfolio_return_df['Value Weighted Portfolio Return'] = vw_return

portfolio_return_df.plot(figsize=(12,8))

# risk-free rate

riskfree = pd.read_csv('riskfree.csv').set_index('caldt')
riskfree_rate = riskfree['t30ret']

portfolio_return_df['rf'] = riskfree_rate[1:]

## Top500 case

Top500Weights = pd.DataFrame()
Top500Return = pd.DataFrame()
Top500SE = pd.DataFrame()

```

```python
Top500Coef = pd.DataFrame(np.zeros(11)).T

year_list = range(1970, 2021)

for year in year_list:

    df_ret = pd.read_csv('./top500/ret'+str(year)+'.csv').set_index('date')
    stock_list = df_ret.columns

    scaled_data_folder = './new standardized5/'
    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/scaled ret' + str(year) + '.csv').set_index('date')[stock_list]
    scaled_mktcap = pd.read_csv(scaled_data_folder + 'mktcap/mktcap' + str(year) + '.csv').set_index('date')[stock_list]
    scaled_bm = pd.read_csv(scaled_data_folder + 'bm/bm' + str(year) + '.csv').set_index('date')[stock_list]
    scaled_roa = pd.read_csv(scaled_data_folder + 'roa/roa' + str(year) + '.csv').set_index('date')[stock_list]
    scaled_roe = pd.read_csv(scaled_data_folder + 'roe/roe' + str(year) + '.csv').set_index('date')[stock_list]
    scaled_accrual = pd.read_csv(scaled_data_folder + 'accrual/accrual' + str(year) + '.csv').set_index('date')[stock_list]
    scaled_cfm = pd.read_csv(scaled_data_folder + 'cfm/cfm' + str(year) + '.csv').set_index('date')[stock_list]
    scaled_eqinv = pd.read_csv(scaled_data_folder + 'equity invcap/equity invcap' + str(year) + '.csv').set_index('date')[stock_list]
    scaled_atturn = pd.read_csv(scaled_data_folder + 'at turn/at turn' + str(year) + '.csv').set_index('date')[stock_list]
    scaled_pcf = pd.read_csv(scaled_data_folder + 'pcf/pcf' + str(year) + '.csv').set_index('date')[stock_list]
    scaled_da = pd.read_csv(scaled_data_folder + 'debt asset/debt asset' + str(year) + '.csv').set_index('date')[stock_list]
    scaled_curr = pd.read_csv(scaled_data_folder + 'curr ratio/curr ratio' + str(year) + '.csv').set_index('date')[stock_list]

    quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(year)+'/09/30',str(year)+'/12/31']
    df_ret = df_ret.loc[quarter_index, :]
    scaled_ret = scaled_ret.loc[quarter_index, :]
    scaled_mktcap = scaled_mktcap.loc[quarter_index, :]
    scaled_bm = scaled_bm.loc[quarter_index, :]
    scaled_roa = scaled_roa.loc[quarter_index, :]
    scaled_roe = scaled_roe.loc[quarter_index, :]
    scaled_accrual = scaled_accrual.loc[quarter_index, :]
    scaled_cfm = scaled_cfm.loc[quarter_index, :]
    scaled_eqinv = scaled_eqinv.loc[quarter_index, :]
    scaled_atturn = scaled_atturn.loc[quarter_index, :]
    scaled_pcf = scaled_pcf.loc[quarter_index, :]
    scaled_da = scaled_da.loc[quarter_index, :]
    scaled_curr = scaled_curr.loc[quarter_index, :]

    Top500Return = Top500Return.append(df_ret)

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

```
    top500_results = []
    top500_weights = []
    top500_se = []
    init_points = list(Top500Coef.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_base,
            init_points,
            method="BFGS",
            args=(
                wb,
                nt,
                scaled_ret.iloc[0 : i, :],
                scaled_mktcap.iloc[0 : i, :],
                scaled_bm.iloc[0 : i, :],
                scaled_roa.iloc[0 : i, :],
                scaled_roe.iloc[0 : i, :],
                scaled_accrual.iloc[0 : i, :],
                scaled_eqinv.iloc[0 : i, :],
                scaled_atturn.iloc[0 : i, :],
                scaled_cfm.iloc[0 : i, :],
                scaled_curr.iloc[0 : i, :],
                scaled_da.iloc[0 : i, :],
                scaled_pcf.iloc[0 : i, :],
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
        print("The value:", opt["x"])
        top500_results.append(list(opt["x"]))
        top500_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
        weight = wb + nt * (
            + opt["x"][0] * scaled_mktcap.iloc[i, :]
            + opt["x"][1] * scaled_bm.iloc[i, :]
            + opt["x"][2] * scaled_roa.iloc[i, :]
            + opt["x"][3] * scaled_roe.iloc[i, :]
            + opt["x"][4] * scaled_accrual.iloc[i, :]
            + opt["x"][5] * scaled_eqinv.iloc[i, :]
            + opt["x"][6] * scaled_atturn.iloc[i, :]
            + opt["x"][7] * scaled_cfm.iloc[i, :]
            + opt["x"][8] * scaled_curr.iloc[i, :]
            + opt["x"][9] * scaled_da.iloc[i, :]
            + opt["x"][10] * scaled_pcf.iloc[i, :]
        )
        print(weight)
        top500_weights.append(weight)

    Top500Weights = Top500Weights.append(short_sell_constraints(pd.
   DataFrame(top500_weights)))
    Top500Coef = Top500Coef.append(pd.DataFrame(top500_results))
    Top500SE = Top500SE.append(pd.DataFrame(top500_se))

# Top500 Value Weighted and Equal weighted Portfolio

```

```
vwWeight500 = pd.DataFrame()
ewWeight500 = pd.DataFrame()

for year in year_list:

    mktcap_source_file = './top500/mktcap ' + str(year) + '.csv'

    mktcap_df = pd.read_csv(mktcap_source_file).set_index('date')
    mktcap_df = mktcap_df.iloc[[2,5,8,11], :]

    vwWeight500 = vwWeight500.append(value_weights(mktcap_df))

for year in year_list:

    portfolio_source_file = './top500/ret' + str(year) + '.csv'

    portfolio_df = pd.read_csv(portfolio_source_file).set_index('
   date')
    portfolio_df = portfolio_df.iloc[[2,5,8,11], :]
    M = portfolio_df.shape[0]
    N = portfolio_df.shape[1]

    ewWeight500 = ewWeight500.append(pd.DataFrame(np.ones((M,N))/N,
    index=portfolio_df.index, columns=portfolio_df.columns))

## PCA Cases

PCA2Weights = pd.DataFrame()
PCA2Return = pd.DataFrame()
PCA2SE = pd.DataFrame()

PCA2Coef = pd.DataFrame(np.zeros(2)).T
rr = 5
year_list = range(1970, 2021)

for year in year_list:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
   set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_PCA2_folder = './PCA Case/2 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
   ret' + str(year) + '.csv').set_index('date')
    scaled_component1 = pd.read_csv(scaled_PCA2_folder + str(year)
   + '/component 1.csv').set_index('date')
    scaled_component2 = pd.read_csv(scaled_PCA2_folder + str(year)
   + '/component 2.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, :]
    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

    PCA2Return = PCA2Return.append(df_ret)

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

    PCA2_results = []
    PCA2_weights = []
    PCA2_se = []
    init_points = list(PCA2Coef.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_pca_2,
            init_points,
            method="BFGS",
            args=(
                wb,
                nt,
                scaled_ret.iloc[0 : i, :],
                scaled_component1.iloc[0 : i, :],
                scaled_component2.iloc[0 : i, :],
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
        print("The value:", opt["x"])
        PCA2_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
        PCA2_results.append(list(opt["x"]))
        weight = wb + nt * (
            opt["x"][0] * scaled_component1.iloc[i, :]
            + opt["x"][1] * scaled_component2.iloc[i, :]
        )
        print(weight)
        PCA2_weights.append(weight)

    PCA2Weights = PCA2Weights.append(short_sell_constraints(pd.
   DataFrame(PCA2_weights)))
    PCA2Coef = PCA2Coef.append(pd.DataFrame(PCA2_results))
    PCA2SE = PCA2SE.append(PCA2_se)

PCA3Weights = pd.DataFrame()
PCA3Return = pd.DataFrame()
PCA3SE = pd.DataFrame()

PCA3Coef = pd.DataFrame(np.zeros(3)).T
rr = 5
year_list = range(1970, 2021)

for year in year_list:

    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

    PCA3Return = PCA3Return.append(df_ret)

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

    PCA3_results = []
    PCA3_weights = []
    PCA3_se = []
    init_points = list(PCA3Coef.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 * (
            opt["x"][0] * scaled_component1.iloc[i, :]
            + opt["x"][1] * scaled_component2.iloc[i, :]
            + opt["x"][2] * scaled_component3.iloc[i, :]
```

```python
        )
        print(weight)
        PCA3_weights.append(weight)

    PCA3Weights = PCA3Weights.append(short_sell_constraints(pd.DataFrame(PCA3_weights)))
    PCA3Coef = PCA3Coef.append(pd.DataFrame(PCA3_results))
    PCA3SE = PCA3SE.append(pd.DataFrame(PCA3_se))

PCA4Weights = pd.DataFrame()
PCA4Return = pd.DataFrame()
PCA4SE = pd.DataFrame()

PCA4Coef = pd.DataFrame(np.zeros(4)).T
rr = 5
year_list = range(1970, 2021)

for year in year_list:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_PCA4_folder = './PCA Case/4 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled ret' + str(year) + '.csv').set_index('date')
    scaled_component1 = pd.read_csv(scaled_PCA4_folder + str(year) + '/component 1.csv').set_index('date')
    scaled_component2 = pd.read_csv(scaled_PCA4_folder + str(year) + '/component 2.csv').set_index('date')
    scaled_component3 = pd.read_csv(scaled_PCA4_folder + str(year) + '/component 3.csv').set_index('date')
    scaled_component4 = pd.read_csv(scaled_PCA4_folder + str(year) + '/component 4.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, :]
    scaled_component4 = scaled_component4.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
    scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T

    PCA4Return = PCA4Return.append(df_ret)

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

    PCA4_results = []
    PCA4_weights = []
    PCA4_se = []
```

```
    init_points = list(PCA4Coef.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_pca_4,
            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, :],
                scaled_component4.iloc[0 : i, :],
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
        print("The value:", opt["x"])
        PCA4_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))

        PCA4_results.append(list(opt["x"]))
        weight = wb + nt * (
            opt["x"][0] * scaled_component1.iloc[i, :]
            + opt["x"][1] * scaled_component2.iloc[i, :]
            + opt["x"][2] * scaled_component3.iloc[i, :]
            + opt["x"][3] * scaled_component4.iloc[i, :]
        )
        print(weight)
        PCA4_weights.append(weight)

    PCA4Weights = PCA4Weights.append(short_sell_constraints(pd.
   DataFrame(PCA4_weights)))
    PCA4Coef = PCA4Coef.append(pd.DataFrame(PCA4_results))
    PCA4SE = PCA4SE.append(pd.DataFrame(PCA4_se))

PCA5Weights = pd.DataFrame()
PCA5Return = pd.DataFrame()
PCA5SE = pd.DataFrame()

PCA5Coef = pd.DataFrame(np.zeros(5)).T
rr = 5
year_list = range(1970, 2021)

for year in year_list:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
   set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_PCA5_folder = './PCA Case/5 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
   ret' + str(year) + '.csv').set_index('date')
    scaled_component1 = pd.read_csv(scaled_PCA5_folder + str(year)
```

```
+ '/component 1.csv').set_index('date')
    scaled_component2 = pd.read_csv(scaled_PCA5_folder + str(year)
+ '/component 2.csv').set_index('date')
    scaled_component3 = pd.read_csv(scaled_PCA5_folder + str(year)
+ '/component 3.csv').set_index('date')
    scaled_component4 = pd.read_csv(scaled_PCA5_folder + str(year)
+ '/component 4.csv').set_index('date')
    scaled_component5 = pd.read_csv(scaled_PCA5_folder + str(year)
+ '/component 5.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, :]
    scaled_component4 = scaled_component4.loc[quarter_index, :]
    scaled_component5 = scaled_component5.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
    scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T
    scaled_component5 =  pd.DataFrame(Scale(scaled_component5.T)).T

    PCA5Return = PCA5Return.append(df_ret)

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

    PCA5_results = []
    PCA5_weights = []
    PCA5_se = []
    init_points = list(PCA5Coef.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_pca_5,
            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, :],
                scaled_component4.iloc[0 : i, :],
                scaled_component5.iloc[0 : i, :],
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
        print("The value:", opt["x"])
        PCA5_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
        PCA5_results.append(list(opt["x"]))
        weight = wb + nt * (
```

```
            opt["x"][0] * scaled_component1.iloc[i, :]
            + opt["x"][1] * scaled_component2.iloc[i, :]
            + opt["x"][2] * scaled_component3.iloc[i, :]
            + opt["x"][3] * scaled_component4.iloc[i, :]
            + opt["x"][4] * scaled_component5.iloc[i, :]
        )
        print(weight)
        PCA5_weights.append(weight)

    PCA5Weights = PCA5Weights.append(short_sell_constraints(pd.
    DataFrame(PCA5_weights)))
    PCA5Coef = PCA5Coef.append(pd.DataFrame(PCA5_results))
    PCA5SE = PCA5SE.append(pd.DataFrame(PCA5_se))


PCA6Weights = pd.DataFrame()
PCA6Return = pd.DataFrame()
PCA6SE = pd.DataFrame()

PCA6Coef = pd.DataFrame(np.zeros(6)).T
rr = 5
year_list = range(1970, 2021)

for year in year_list:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
    set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_PCA6_folder = './PCA Case/6 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
    ret' + str(year) + '.csv').set_index('date')
    scaled_component1 = pd.read_csv(scaled_PCA6_folder + str(year)
    + '/component 1.csv').set_index('date')
    scaled_component2 = pd.read_csv(scaled_PCA6_folder + str(year)
    + '/component 2.csv').set_index('date')
    scaled_component3 = pd.read_csv(scaled_PCA6_folder + str(year)
    + '/component 3.csv').set_index('date')
    scaled_component4 = pd.read_csv(scaled_PCA6_folder + str(year)
    + '/component 4.csv').set_index('date')
    scaled_component5 = pd.read_csv(scaled_PCA6_folder + str(year)
    + '/component 5.csv').set_index('date')
    scaled_component6 = pd.read_csv(scaled_PCA6_folder + str(year)
    + '/component 6.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, :]
    scaled_component4 = scaled_component4.loc[quarter_index, :]
    scaled_component5 = scaled_component5.loc[quarter_index, :]
    scaled_component6 = scaled_component6.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
    scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T
    scaled_component5 =  pd.DataFrame(Scale(scaled_component5.T)).T
    scaled_component6 =  pd.DataFrame(Scale(scaled_component6.T)).T

    PCA6Return = PCA6Return.append(df_ret)

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

    PCA6_results = []
    PCA6_weights = []
    PCA6_se = []
    init_points = list(PCA6Coef.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_pca_6,
            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, :],
                scaled_component4.iloc[0 : i, :],
                scaled_component5.iloc[0 : i, :],
                scaled_component6.iloc[0 : i, :],
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
        print("The value:", opt["x"])
        PCA6_results.append(list(opt["x"]))
        PCA6_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))

        weight = wb + nt * (
            opt["x"][0] * scaled_component1.iloc[i, :]
            + opt["x"][1] * scaled_component2.iloc[i, :]
            + opt["x"][2] * scaled_component3.iloc[i, :]
            + opt["x"][3] * scaled_component4.iloc[i, :]
            + opt["x"][4] * scaled_component5.iloc[i, :]
            + opt["x"][5] * scaled_component6.iloc[i, :]
        )
        print(weight)
        PCA6_weights.append(weight)

    PCA6Weights = PCA6Weights.append(short_sell_constraints(pd.
   DataFrame(PCA6_weights)))
    PCA6Coef = PCA6Coef.append(pd.DataFrame(PCA6_results))
    PCA6SE = PCA6SE.append(pd.DataFrame(PCA6_se))

PCA7Weights = pd.DataFrame()
```

```
PCA7Return = pd.DataFrame()
PCA7SE = pd.DataFrame()

PCA7Coef = pd.DataFrame(np.zeros(7)).T
rr = 5
year_list = range(1970, 2021)

for year in year_list:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
    set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_PCA7_folder = './PCA Case/7 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
    ret' + str(year) + '.csv').set_index('date')
    scaled_component1 = pd.read_csv(scaled_PCA7_folder + str(year)
    + '/component 1.csv').set_index('date')
    scaled_component2 = pd.read_csv(scaled_PCA7_folder + str(year)
    + '/component 2.csv').set_index('date')
    scaled_component3 = pd.read_csv(scaled_PCA7_folder + str(year)
    + '/component 3.csv').set_index('date')
    scaled_component4 = pd.read_csv(scaled_PCA7_folder + str(year)
    + '/component 4.csv').set_index('date')
    scaled_component5 = pd.read_csv(scaled_PCA7_folder + str(year)
    + '/component 5.csv').set_index('date')
    scaled_component6 = pd.read_csv(scaled_PCA7_folder + str(year)
    + '/component 6.csv').set_index('date')
    scaled_component7 = pd.read_csv(scaled_PCA7_folder + str(year)
    + '/component 7.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, :]
    scaled_component4 = scaled_component4.loc[quarter_index, :]
    scaled_component5 = scaled_component5.loc[quarter_index, :]
    scaled_component6 = scaled_component6.loc[quarter_index, :]
    scaled_component7 = scaled_component7.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
    scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T
    scaled_component5 =  pd.DataFrame(Scale(scaled_component5.T)).T
    scaled_component6 =  pd.DataFrame(Scale(scaled_component6.T)).T
    scaled_component7 =  pd.DataFrame(Scale(scaled_component7.T)).T

    PCA7Return = PCA7Return.append(df_ret)

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

```

```
    PCA7_results = []
    PCA7_weights = []
    PCA7_se = []
    init_points = list(PCA7Coef.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_pca_7,
            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, :],
                scaled_component4.iloc[0 : i, :],
                scaled_component5.iloc[0 : i, :],
                scaled_component6.iloc[0 : i, :],
                scaled_component7.iloc[0 : i, :],
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
        print("The value:", opt["x"])
        PCA7_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))

        PCA7_results.append(list(opt["x"]))
        weight = wb + nt * (
            opt["x"][0] * scaled_component1.iloc[i, :]
            + opt["x"][1] * scaled_component2.iloc[i, :]
            + opt["x"][2] * scaled_component3.iloc[i, :]
            + opt["x"][3] * scaled_component4.iloc[i, :]
            + opt["x"][4] * scaled_component5.iloc[i, :]
            + opt["x"][5] * scaled_component6.iloc[i, :]
            + opt["x"][6] * scaled_component7.iloc[i, :]
        )
        print(weight)
        PCA7_weights.append(weight)

    PCA7Weights = PCA7Weights.append(short_sell_constraints(pd.
   DataFrame(PCA7_weights)))
    PCA7Coef = PCA7Coef.append(pd.DataFrame(PCA7_results))
    PCA7SE = PCA7SE.append(pd.DataFrame(PCA7_se))


PCA8Weights = pd.DataFrame()
PCA8Return = pd.DataFrame()
PCA8SE = pd.DataFrame()

PCA8Coef = pd.DataFrame(np.zeros(8)).T
rr = 5
year_list = range(1970, 2021)

for year in year_list:
```

```
    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
    set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_PCA8_folder = './PCA Case/8 mpc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
    ret' + str(year) + '.csv').set_index('date')
    scaled_component1 = pd.read_csv(scaled_PCA8_folder + str(year)
    + '/component 1.csv').set_index('date')
    scaled_component2 = pd.read_csv(scaled_PCA8_folder + str(year)
    + '/component 2.csv').set_index('date')
    scaled_component3 = pd.read_csv(scaled_PCA8_folder + str(year)
    + '/component 3.csv').set_index('date')
    scaled_component4 = pd.read_csv(scaled_PCA8_folder + str(year)
    + '/component 4.csv').set_index('date')
    scaled_component5 = pd.read_csv(scaled_PCA8_folder + str(year)
    + '/component 5.csv').set_index('date')
    scaled_component6 = pd.read_csv(scaled_PCA8_folder + str(year)
    + '/component 6.csv').set_index('date')
    scaled_component7 = pd.read_csv(scaled_PCA8_folder + str(year)
    + '/component 7.csv').set_index('date')
    scaled_component8 = pd.read_csv(scaled_PCA8_folder + str(year)
    + '/component 8.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, :]
    scaled_component4 = scaled_component4.loc[quarter_index, :]
    scaled_component5 = scaled_component5.loc[quarter_index, :]
    scaled_component6 = scaled_component6.loc[quarter_index, :]
    scaled_component7 = scaled_component7.loc[quarter_index, :]
    scaled_component8 = scaled_component8.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
    scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T
    scaled_component5 =  pd.DataFrame(Scale(scaled_component5.T)).T
    scaled_component6 =  pd.DataFrame(Scale(scaled_component6.T)).T
    scaled_component7 =  pd.DataFrame(Scale(scaled_component7.T)).T
    scaled_component8 =  pd.DataFrame(Scale(scaled_component8.T)).T

    PCA8Return = PCA8Return.append(df_ret)

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

    PCA8_results = []
    PCA8_weights = []
    PCA8_se = []
    init_points = list(PCA8Coef.iloc[-1,:].values)
```

```
    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_pca_8,
            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, :],
                scaled_component4.iloc[0 : i, :],
                scaled_component5.iloc[0 : i, :],
                scaled_component6.iloc[0 : i, :],
                scaled_component7.iloc[0 : i, :],
                scaled_component8.iloc[0 : i, :],
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
        print("The value:", opt["x"])
        PCA8_results.append(list(opt["x"]))
        PCA8_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))

        weight = wb + nt * (
            opt["x"][0] * scaled_component1.iloc[i, :]
            + opt["x"][1] * scaled_component2.iloc[i, :]
            + opt["x"][2] * scaled_component3.iloc[i, :]
            + opt["x"][3] * scaled_component4.iloc[i, :]
            + opt["x"][4] * scaled_component5.iloc[i, :]
            + opt["x"][5] * scaled_component6.iloc[i, :]
            + opt["x"][6] * scaled_component7.iloc[i, :]
            + opt["x"][7] * scaled_component8.iloc[i, :]
        )
        print(weight)
        PCA8_weights.append(weight)

    PCA8Weights = PCA8Weights.append(short_sell_constraints(pd.
   DataFrame(PCA8_weights)))
    PCA8Coef = PCA8Coef.append(pd.DataFrame(PCA8_results))
    PCA8SE = PCA8SE.append(pd.DataFrame(PCA8_se))

## Top500 PCA Cases

PCA2Weights500 = pd.DataFrame()
PCA2Return500 = pd.DataFrame()
PCA2SE500 = pd.DataFrame()

PCA2Coef500 = pd.DataFrame(np.zeros(2)).T
rr = 5
year_list = range(1970, 2021)

for year in year_list:

    df_ret = pd.read_csv('./top500/ret'+str(year)+'.csv').set_index
```

```
('date')
    stock_list = list(df_ret.columns)

    scaled_data_folder = './new standardized5/'
    scaled_PCA2_folder = './PCA Case/2 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled ret' + str(year) + '.csv').set_index('date')[stock_list]
    scaled_component1 = pd.read_csv(scaled_PCA2_folder + str(year) + '/component 1.csv').set_index('date')[stock_list]
    scaled_component2 = pd.read_csv(scaled_PCA2_folder + str(year) + '/component 2.csv').set_index('date')[stock_list]

    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, :]
    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

    PCA2Return500 = PCA2Return500.append(df_ret)

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

    PCA2_results_500 = []
    PCA2_weights_500 = []
    PCA2_se_500 = []
    init_points = list(PCA2Coef500.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_pca_2,
            init_points,
            method="BFGS",
            args=(
                wb,
                nt,
                scaled_ret.iloc[0 : i, :],
                scaled_component1.iloc[0 : i, :],
                scaled_component2.iloc[0 : i, :],
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
        print("The value:", opt["x"])
        PCA2_results_500.append(list(opt["x"]))
        PCA2_se_500.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
        weight = wb + nt * (
            opt["x"][0] * scaled_component1.iloc[i, :]
            + opt["x"][1] * scaled_component2.iloc[i, :]
        )
        print(weight)
        PCA2_weights_500.append(weight)

```

```
    PCA2Weights500 = PCA2Weights500.append(short_sell_constraints(
   pd.DataFrame(PCA2_weights_500)))
    PCA2Coef500 = PCA2Coef500.append(pd.DataFrame(PCA2_results_500)
   )
    PCA2SE500 = PCA2SE500.append(pd.DataFrame(PCA2_se_500))

PCA3Weights500 = pd.DataFrame()
PCA3Return500 = pd.DataFrame()
PCA3SE500 = pd.DataFrame()

PCA3Coef500 = pd.DataFrame(np.zeros(3)).T
rr = 5
year_list = range(1970, 2021)

for year in year_list:

    df_ret = pd.read_csv('./top500/ret'+str(year)+'.csv').set_index
   ('date')
    stock_list = list(df_ret.columns)

    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')[stock_list]
    scaled_component1 = pd.read_csv(scaled_PCA3_folder + str(year)
   + '/component 1.csv').set_index('date')[stock_list]
    scaled_component2 = pd.read_csv(scaled_PCA3_folder + str(year)
   + '/component 2.csv').set_index('date')[stock_list]
    scaled_component3 = pd.read_csv(scaled_PCA3_folder + str(year)
   + '/component 3.csv').set_index('date')[stock_list]

    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


    PCA3Return500 = PCA3Return500.append(df_ret)

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

    PCA3_results_500 = []
    PCA3_weights_500 = []
    PCA3_se_500 = []
    init_points = list(PCA3Coef500.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_results_500.append(list(opt["x"]))
        PCA3_se_500.append(list(((np.diag(opt.hess_inv))*nt)**0.5))

        weight = wb + nt * (
            opt["x"][0] * scaled_component1.iloc[i, :]
            + opt["x"][1] * scaled_component2.iloc[i, :]
            + opt["x"][2] * scaled_component3.iloc[i, :]
        )
        print(weight)
        PCA3_weights_500.append(weight)

    PCA3Weights500 = PCA3Weights500.append(short_sell_constraints(
    pd.DataFrame(PCA3_weights_500)))
    PCA3Coef500 = PCA3Coef500.append(pd.DataFrame(PCA3_results_500)
    )
    PCA3SE500 = PCA3SE500.append(pd.DataFrame(PCA3_se_500))


PCA4Weights500 = pd.DataFrame()
PCA4Return500 = pd.DataFrame()
PCA4SE500 = pd.DataFrame()

PCA4Coef500 = pd.DataFrame(np.zeros(4)).T
rr = 5
year_list = range(1970, 2021)

for year in year_list:

    df_ret = pd.read_csv('./top500/ret'+str(year)+'.csv').set_index
    ('date')
    stock_list = list(df_ret.columns)

    scaled_data_folder = './new standardized5/'
    scaled_PCA4_folder = './PCA Case/4 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
    ret' + str(year) + '.csv').set_index('date')[stock_list]
    scaled_component1 = pd.read_csv(scaled_PCA4_folder + str(year)
    + '/component 1.csv').set_index('date')[stock_list]
    scaled_component2 = pd.read_csv(scaled_PCA4_folder + str(year)
    + '/component 2.csv').set_index('date')[stock_list]
    scaled_component3 = pd.read_csv(scaled_PCA4_folder + str(year)
```

```
    + '/component 3.csv').set_index('date')[stock_list]
    scaled_component4 = pd.read_csv(scaled_PCA4_folder + str(year)
    + '/component 4.csv').set_index('date')[stock_list]


    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, :]
    scaled_component4 = scaled_component4.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
    scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T

    PCA4Return500 = PCA4Return500.append(df_ret)

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

    PCA4_results_500 = []
    PCA4_weights_500 = []
    PCA4_se_500 = []
    init_points = list(PCA4Coef500.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_pca_4,
            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, :],
                scaled_component4.iloc[0 : i, :],
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
        print("The value:", opt["x"])
        PCA4_results_500.append(list(opt["x"]))
        PCA4_se_500.append(list(((np.diag(opt.hess_inv))*nt)**0.5))

        weight = wb + nt * (
            opt["x"][0] * scaled_component1.iloc[i, :]
            + opt["x"][1] * scaled_component2.iloc[i, :]
            + opt["x"][2] * scaled_component3.iloc[i, :]
            + opt["x"][3] * scaled_component4.iloc[i, :]
        )
        print(weight)
        PCA4_weights_500.append(weight)
```

```
    PCA4Weights500 = PCA4Weights500.append(short_sell_constraints(pd.DataFrame(PCA4_weights_500)))
    PCA4Coef500 = PCA4Coef500.append(pd.DataFrame(PCA4_results_500))
    PCA4SE500 = PCA4SE500.append(pd.DataFrame(PCA4_se_500))

PCA5Weights500 = pd.DataFrame()
PCA5Return500 = pd.DataFrame()
PCA5SE500 = pd.DataFrame()

PCA5Coef500 = pd.DataFrame(np.zeros(5)).T
rr = 5
year_list = range(1970, 2021)

for year in year_list:

    df_ret = pd.read_csv('./top500/ret'+str(year)+'.csv').set_index('date')
    stock_list = list(df_ret.columns)

    scaled_data_folder = './new standardized5/'
    scaled_PCA5_folder = './PCA Case/5 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled ret' + str(year) + '.csv').set_index('date')[stock_list]
    scaled_component1 = pd.read_csv(scaled_PCA5_folder + str(year) + '/component 1.csv').set_index('date')[stock_list]
    scaled_component2 = pd.read_csv(scaled_PCA5_folder + str(year) + '/component 2.csv').set_index('date')[stock_list]
    scaled_component3 = pd.read_csv(scaled_PCA5_folder + str(year) + '/component 3.csv').set_index('date')[stock_list]
    scaled_component4 = pd.read_csv(scaled_PCA5_folder + str(year) + '/component 4.csv').set_index('date')[stock_list]
    scaled_component5 = pd.read_csv(scaled_PCA5_folder + str(year) + '/component 5.csv').set_index('date')[stock_list]


    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, :]
    scaled_component4 = scaled_component4.loc[quarter_index, :]
    scaled_component5 = scaled_component5.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
    scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T
    scaled_component5 =  pd.DataFrame(Scale(scaled_component5.T)).T

    PCA5Return500 = PCA5Return500.append(df_ret)
```

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

    PCA5_results_500 = []
    PCA5_weights_500 = []
    PCA5_se_500 = []
    init_points = list(PCA5Coef500.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_pca_5,
            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, :],
                scaled_component4.iloc[0 : i, :],
                scaled_component5.iloc[0 : i, :],
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
        print("The value:", opt["x"])
        PCA5_results_500.append(list(opt["x"]))
        PCA5_se_500.append(list(((np.diag(opt.hess_inv))*nt)**0.5))

        weight = wb + nt * (
            opt["x"][0] * scaled_component1.iloc[i, :]
            + opt["x"][1] * scaled_component2.iloc[i, :]
            + opt["x"][2] * scaled_component3.iloc[i, :]
            + opt["x"][3] * scaled_component4.iloc[i, :]
            + opt["x"][4] * scaled_component5.iloc[i, :]
        )
        print(weight)
        PCA5_weights_500.append(weight)

    PCA5Weights500 = PCA5Weights500.append(short_sell_constraints(
    pd.DataFrame(PCA5_weights_500)))
    PCA5Coef500 = PCA5Coef500.append(pd.DataFrame(PCA5_results_500)
    )
    PCA5SE500 = PCA5SE500.append(pd.DataFrame(PCA5_se_500))

PCA6Weights500 = pd.DataFrame()
PCA6Return500 = pd.DataFrame()
PCA6SE500 = pd.DataFrame()

PCA6Coef500 = pd.DataFrame(np.zeros(6)).T
rr = 5
year_list = range(1970, 2021)

for year in year_list:

    df_ret = pd.read_csv('./top500/ret'+str(year)+'.csv').set_index
```

```
('date')
    stock_list = list(df_ret.columns)

    scaled_data_folder = './new standardized5/'
    scaled_PCA6_folder = './PCA Case/6 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled ret' + str(year) + '.csv').set_index('date')[stock_list]
    scaled_component1 = pd.read_csv(scaled_PCA6_folder + str(year) + '/component 1.csv').set_index('date')[stock_list]
    scaled_component2 = pd.read_csv(scaled_PCA6_folder + str(year) + '/component 2.csv').set_index('date')[stock_list]
    scaled_component3 = pd.read_csv(scaled_PCA6_folder + str(year) + '/component 3.csv').set_index('date')[stock_list]
    scaled_component4 = pd.read_csv(scaled_PCA6_folder + str(year) + '/component 4.csv').set_index('date')[stock_list]
    scaled_component5 = pd.read_csv(scaled_PCA6_folder + str(year) + '/component 5.csv').set_index('date')[stock_list]
    scaled_component6 = pd.read_csv(scaled_PCA6_folder + str(year) + '/component 6.csv').set_index('date')[stock_list]

    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, :]
    scaled_component4 = scaled_component4.loc[quarter_index, :]
    scaled_component5 = scaled_component5.loc[quarter_index, :]
    scaled_component6 = scaled_component6.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
    scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T
    scaled_component5 =  pd.DataFrame(Scale(scaled_component5.T)).T
    scaled_component6 =  pd.DataFrame(Scale(scaled_component6.T)).T

    PCA6Return500 = PCA6Return500.append(df_ret)

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

    PCA6_results_500 = []
    PCA6_weights_500 = []
    PCA6_se_500 = []
    init_points = list(PCA6Coef500.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_pca_6,
            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, :],
                    scaled_component4.iloc[0 : i, :],
                    scaled_component5.iloc[0 : i, :],
                    scaled_component6.iloc[0 : i, :],
                    rr,
                ),
            )
            print("The {} window for year {}".format(i+1, year))
            print("The value:", opt["x"])
            PCA6_results_500.append(list(opt["x"]))
            PCA6_se_500.append(list(((np.diag(opt.hess_inv))*nt)**0.5))

            weight = wb + nt * (
                opt["x"][0] * scaled_component1.iloc[i, :]
                + opt["x"][1] * scaled_component2.iloc[i, :]
                + opt["x"][2] * scaled_component3.iloc[i, :]
                + opt["x"][3] * scaled_component4.iloc[i, :]
                + opt["x"][4] * scaled_component5.iloc[i, :]
                + opt["x"][5] * scaled_component6.iloc[i, :]
            )
            print(weight)
            PCA6_weights_500.append(weight)

        PCA6Weights500 = PCA6Weights500.append(short_sell_constraints(
    pd.DataFrame(PCA6_weights_500)))
        PCA6Coef500 = PCA6Coef500.append(pd.DataFrame(PCA6_results_500)
    )
        PCA6SE500 = PCA6SE500.append(pd.DataFrame(PCA6_se_500))


PCA7Weights500 = pd.DataFrame()
PCA7Return500 = pd.DataFrame()
PCA7SE500 = pd.DataFrame()

PCA7Coef500 = pd.DataFrame(np.zeros(7)).T
rr = 5
year_list = range(1970, 2021)

for year in year_list:

    df_ret = pd.read_csv('./top500/ret'+str(year)+'.csv').set_index
    ('date')
    stock_list = list(df_ret.columns)

    scaled_data_folder = './new standardized5/'
    scaled_PCA7_folder = './PCA Case/7 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
    ret' + str(year) + '.csv').set_index('date')[stock_list]
    scaled_component1 = pd.read_csv(scaled_PCA7_folder + str(year)
    + '/component 1.csv').set_index('date')[stock_list]
    scaled_component2 = pd.read_csv(scaled_PCA7_folder + str(year)
    + '/component 2.csv').set_index('date')[stock_list]
```

```
    scaled_component3 = pd.read_csv(scaled_PCA7_folder + str(year)
    + '/component 3.csv').set_index('date')[stock_list]
    scaled_component4 = pd.read_csv(scaled_PCA7_folder + str(year)
    + '/component 4.csv').set_index('date')[stock_list]
    scaled_component5 = pd.read_csv(scaled_PCA7_folder + str(year)
    + '/component 5.csv').set_index('date')[stock_list]
    scaled_component6 = pd.read_csv(scaled_PCA7_folder + str(year)
    + '/component 6.csv').set_index('date')[stock_list]
    scaled_component7 = pd.read_csv(scaled_PCA7_folder + str(year)
    + '/component 7.csv').set_index('date')[stock_list]

    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, :]
    scaled_component4 = scaled_component4.loc[quarter_index, :]
    scaled_component5 = scaled_component5.loc[quarter_index, :]
    scaled_component6 = scaled_component6.loc[quarter_index, :]
    scaled_component7 = scaled_component7.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
    scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T
    scaled_component5 =  pd.DataFrame(Scale(scaled_component5.T)).T
    scaled_component6 =  pd.DataFrame(Scale(scaled_component6.T)).T
    scaled_component7 =  pd.DataFrame(Scale(scaled_component7.T)).T

    PCA7Return500 = PCA7Return500.append(df_ret)

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

    PCA7_results_500 = []
    PCA7_weights_500 = []
    PCA7_se_500 = []
    init_points = list(PCA7Coef500.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_pca_7,
            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, :],
                scaled_component4.iloc[0 : i, :],
                scaled_component5.iloc[0 : i, :],
                scaled_component6.iloc[0 : i, :],
```

```
                    scaled_component7.iloc[0 : i, :],
                    rr,
                ),
            )
            print("The {} window for year {}".format(i+1, year))
            print("The value:", opt["x"])
            PCA7_results_500.append(list(opt["x"]))
            PCA7_se_500.append(list(((np.diag(opt.hess_inv))*nt)**0.5))

            weight = wb + nt * (
                opt["x"][0] * scaled_component1.iloc[i, :]
                + opt["x"][1] * scaled_component2.iloc[i, :]
                + opt["x"][2] * scaled_component3.iloc[i, :]
                + opt["x"][3] * scaled_component4.iloc[i, :]
                + opt["x"][4] * scaled_component5.iloc[i, :]
                + opt["x"][5] * scaled_component6.iloc[i, :]
                + opt["x"][6] * scaled_component7.iloc[i, :]
            )
            print(weight)
            PCA7_weights_500.append(weight)

    PCA7Weights500 = PCA7Weights500.append(short_sell_constraints(
pd.DataFrame(PCA7_weights_500)))
    PCA7Coef500 = PCA7Coef500.append(pd.DataFrame(PCA7_results_500)
)
    PCA7SE500 = PCA7SE500.append(pd.DataFrame(PCA7_se_500))

PCA8Weights500 = pd.DataFrame()
PCA8Return500 = pd.DataFrame()
PCA8SE500 = pd.DataFrame()

PCA8Coef500 = pd.DataFrame(np.zeros(8)).T
rr = 5
year_list = range(1970, 2021)

for year in year_list:

    df_ret = pd.read_csv('./top500/ret'+str(year)+'.csv').set_index
('date')
    stock_list = list(df_ret.columns)

    scaled_data_folder = './new standardized5/'
    scaled_PCA8_folder = './PCA Case/8 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
ret' + str(year) + '.csv').set_index('date')[stock_list]
    scaled_component1 = pd.read_csv(scaled_PCA8_folder + str(year)
+ '/component 1.csv').set_index('date')[stock_list]
    scaled_component2 = pd.read_csv(scaled_PCA8_folder + str(year)
+ '/component 2.csv').set_index('date')[stock_list]
    scaled_component3 = pd.read_csv(scaled_PCA8_folder + str(year)
+ '/component 3.csv').set_index('date')[stock_list]
    scaled_component4 = pd.read_csv(scaled_PCA8_folder + str(year)
+ '/component 4.csv').set_index('date')[stock_list]
    scaled_component5 = pd.read_csv(scaled_PCA8_folder + str(year)
+ '/component 5.csv').set_index('date')[stock_list]
```

```
    scaled_component6 = pd.read_csv(scaled_PCA8_folder + str(year)
    + '/component 6.csv').set_index('date')[stock_list]
    scaled_component7 = pd.read_csv(scaled_PCA8_folder + str(year)
    + '/component 7.csv').set_index('date')[stock_list]
    scaled_component8 = pd.read_csv(scaled_PCA8_folder + str(year)
    + '/component 8.csv').set_index('date')[stock_list]

    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, :]
    scaled_component4 = scaled_component4.loc[quarter_index, :]
    scaled_component5 = scaled_component5.loc[quarter_index, :]
    scaled_component6 = scaled_component6.loc[quarter_index, :]
    scaled_component7 = scaled_component7.loc[quarter_index, :]
    scaled_component8 = scaled_component8.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
    scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T
    scaled_component5 =  pd.DataFrame(Scale(scaled_component5.T)).T
    scaled_component6 =  pd.DataFrame(Scale(scaled_component6.T)).T
    scaled_component7 =  pd.DataFrame(Scale(scaled_component7.T)).T
    scaled_component8 =  pd.DataFrame(Scale(scaled_component8.T)).T

    PCA8Return500 = PCA8Return500.append(df_ret)

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

    PCA8_results_500 = []
    PCA8_weights_500 = []
    PCA8_se_500 =[]
    init_points = list(PCA8Coef500.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_pca_8,
            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, :],
                scaled_component4.iloc[0 : i, :],
                scaled_component5.iloc[0 : i, :],
                scaled_component6.iloc[0 : i, :],
                scaled_component7.iloc[0 : i, :],
                scaled_component8.iloc[0 : i, :],
```

```
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
        print("The value:", opt["x"])
        PCA8_results_500.append(list(opt["x"]))
        PCA8_se_500.append(list(((np.diag(opt.hess_inv))*nt)**0.5))

        weight = wb + nt * (
            opt["x"][0] * scaled_component1.iloc[i, :]
            + opt["x"][1] * scaled_component2.iloc[i, :]
            + opt["x"][2] * scaled_component3.iloc[i, :]
            + opt["x"][3] * scaled_component4.iloc[i, :]
            + opt["x"][4] * scaled_component5.iloc[i, :]
            + opt["x"][5] * scaled_component6.iloc[i, :]
            + opt["x"][6] * scaled_component7.iloc[i, :]
            + opt["x"][7] * scaled_component8.iloc[i, :]
        )
        print(weight)
        PCA8_weights_500.append(weight)

    PCA8Weights500 = PCA8Weights500.append(short_sell_constraints(
    pd.DataFrame(PCA8_weights_500)))
    PCA8Coef500 = PCA8Coef500.append(pd.DataFrame(PCA8_results_500)
    )
    PCA8SE500 = PCA8SE500.append(pd.DataFrame(PCA8_se_500))

## in-sample and out-of-sample performance
# Base Case in-sample and out-of-sample performance

in_sample = range(1970, 1996)
out_of_sample = range(1996, 2021)

InsampleWeights = pd.DataFrame()
InsampleReturn = pd.DataFrame()
InsampleCoef = pd.DataFrame(np.zeros(11)).T
InsampleSE = pd.DataFrame()

char_name = ['mktcap', 'bm', 'roa', 'roe', 'accrual', 'equity
    invcap', 'at turn',
             'cfm', 'pcf' ,'debt asset', 'curr ratio']

rr = 5


for year in in_sample:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
    set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/scaled ret'
    + str(year) + '.csv').set_index('date')
    scaled_mktcap = pd.read_csv(scaled_data_folder + 'mktcap/mktcap
    ' + str(year) + '.csv').set_index('date')
    scaled_bm = pd.read_csv(scaled_data_folder + 'bm/bm' + str(year
```

```
) + '.csv').set_index('date')
    scaled_roa = pd.read_csv(scaled_data_folder + 'roa/roa' + str(year) + '.csv').set_index('date')
    scaled_roe = pd.read_csv(scaled_data_folder + 'roe/roe' + str(year) + '.csv').set_index('date')
    scaled_accrual = pd.read_csv(scaled_data_folder + 'accrual/accrual' + str(year) + '.csv').set_index('date')
    scaled_cfm = pd.read_csv(scaled_data_folder + 'cfm/cfm' + str(year) + '.csv').set_index('date')
    scaled_eqinv = pd.read_csv(scaled_data_folder + 'equity invcap/equity invcap' + str(year) + '.csv').set_index('date')
    scaled_atturn = pd.read_csv(scaled_data_folder + 'at turn/at turn' + str(year) + '.csv').set_index('date')
    scaled_pcf = pd.read_csv(scaled_data_folder + 'pcf/pcf' + str(year) + '.csv').set_index('date')
    scaled_da = pd.read_csv(scaled_data_folder + 'debt asset/debt asset' + str(year) + '.csv').set_index('date')
    scaled_curr = pd.read_csv(scaled_data_folder + 'curr ratio/curr ratio' + str(year) + '.csv').set_index('date')

    quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(year)+'/09/30',str(year)+'/12/31']
    df_ret = df_ret.loc[quarter_index, :]
    scaled_ret = scaled_ret.loc[quarter_index, :]
    scaled_mktcap = scaled_mktcap.loc[quarter_index, :]
    scaled_bm = scaled_bm.loc[quarter_index, :]
    scaled_roa = scaled_roa.loc[quarter_index, :]
    scaled_roe = scaled_roe.loc[quarter_index, :]
    scaled_accrual = scaled_accrual.loc[quarter_index, :]
    scaled_cfm = scaled_cfm.loc[quarter_index, :]
    scaled_eqinv = scaled_eqinv.loc[quarter_index, :]
    scaled_atturn = scaled_atturn.loc[quarter_index, :]
    scaled_pcf = scaled_pcf.loc[quarter_index, :]
    scaled_da = scaled_da.loc[quarter_index, :]
    scaled_curr = scaled_curr.loc[quarter_index, :]

    InsampleReturn = InsampleReturn.append(df_ret)

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

    insample_results = []
    insample_weights = []
    insample_se = []
    init_points = list(InsampleCoef.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_base,
            init_points,
            method="BFGS",
            args=(
                wb,
                nt,
                scaled_ret.iloc[0 : i, :],
                scaled_mktcap.iloc[0 : i, :],
                scaled_bm.iloc[0 : i, :],
```

```
                scaled_roa.iloc[0 : i, :],
                scaled_roe.iloc[0 : i, :],
                scaled_accrual.iloc[0 : i, :],
                scaled_eqinv.iloc[0 : i, :],
                scaled_atturn.iloc[0 : i, :],
                scaled_cfm.iloc[0 : i, :],
                scaled_curr.iloc[0 : i, :],
                scaled_da.iloc[0 : i, :],
                scaled_pcf.iloc[0 : i, :],
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
        print("The value:", opt["x"])
        insample_results.append(list(opt["x"]))
        insample_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))

        weight = wb + nt * (
            + opt["x"][0] * scaled_mktcap.iloc[i, :]
            + opt["x"][1] * scaled_bm.iloc[i, :]
            + opt["x"][2] * scaled_roa.iloc[i, :]
            + opt["x"][3] * scaled_roe.iloc[i, :]
            + opt["x"][4] * scaled_accrual.iloc[i, :]
            + opt["x"][5] * scaled_eqinv.iloc[i, :]
            + opt["x"][6] * scaled_atturn.iloc[i, :]
            + opt["x"][7] * scaled_cfm.iloc[i, :]
            + opt["x"][8] * scaled_curr.iloc[i, :]
            + opt["x"][9] * scaled_da.iloc[i, :]
            + opt["x"][10] * scaled_pcf.iloc[i, :]
        )
        print(weight)
        insample_weights.append(weight)

    InsampleWeights = InsampleWeights.append(short_sell_constraints(pd.DataFrame(insample_weights)))
    InsampleCoef = InsampleCoef.append(pd.DataFrame(insample_results))
    InsampleSE = InsampleSE.append(insample_se)

## PCA cases in-sample and out-of-sample performance

PCA2InsampleWeights = pd.DataFrame()
PCA2InsampleReturn = pd.DataFrame()
PCA2InsampleCoef = pd.DataFrame(np.zeros(2)).T
PCA2InsampleSE = pd.DataFrame()

rr = 5


for year in in_sample:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_PCA2_folder = './PCA Case/2 npc/'
```

```
    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
ret' + str(year) + '.csv').set_index('date')
    scaled_component1 = pd.read_csv(scaled_PCA2_folder + str(year)
+ '/component 1.csv').set_index('date')
    scaled_component2 = pd.read_csv(scaled_PCA2_folder + str(year)
+ '/component 2.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, :]
    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

    PCA2InsampleReturn = PCA2InsampleReturn.append(df_ret)

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

    PCA2_results = []
    PCA2_weights = []
    PCA2_se = []
    init_points = list(PCA2InsampleCoef.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_pca_2,
            init_points,
            method="BFGS",
            args=(
                wb,
                nt,
                scaled_ret.iloc[0 : i, :],
                scaled_component1.iloc[0 : i, :],
                scaled_component2.iloc[0 : i, :],
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
        print("The value:", opt["x"])
        PCA2_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
        PCA2_results.append(list(opt["x"]))
        weight = wb + nt * (
            opt["x"][0] * scaled_component1.iloc[i, :]
            + opt["x"][1] * scaled_component2.iloc[i, :]
        )
        print(weight)
        PCA2_weights.append(weight)

    PCA2InsampleWeights = PCA2InsampleWeights.append(
short_sell_constraints(pd.DataFrame(PCA2_weights)))
    PCA2InsampleCoef = PCA2InsampleCoef.append(pd.DataFrame(
PCA2_results))
    PCA2InsampleSE = PCA2InsampleSE.append(PCA2_se)
```

```
pca2insample_coef = PCA2InsampleCoef.mean()
PCA2OutofSampleWeights = pd.DataFrame()
PCA2OutofSampleReturn = pd.DataFrame()
for year in out_of_sample:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
    set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_PCA2_folder = './PCA Case/2 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
    ret' + str(year) + '.csv').set_index('date')
    scaled_component1 = pd.read_csv(scaled_PCA2_folder + str(year)
    + '/component 1.csv').set_index('date')
    scaled_component2 = pd.read_csv(scaled_PCA2_folder + str(year)
    + '/component 2.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, :]
    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

    PCA2OutofSampleReturn = PCA2OutofSampleReturn.append(df_ret)

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

    outofsample_weights = []

    for i in range(len(df_ret)):
        weight = wb + nt * (
            + pca2insample_coef[0] * scaled_component1.iloc[i, :]
            + pca2insample_coef[1] * scaled_component2.iloc[i, :]
        )
        outofsample_weights.append(weight)
        print(weight)

    PCA2OutofSampleWeights = PCA2OutofSampleWeights.append(
    short_sell_constraints(pd.DataFrame(outofsample_weights)))

PCA3InsampleWeights = pd.DataFrame()
PCA3InsampleReturn = pd.DataFrame()
PCA3InsampleCoef = pd.DataFrame(np.zeros(3)).T
PCA3InsampleSE = pd.DataFrame()

rr = 5


for year in in_sample:

    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 * (
```

```
            opt["x"][0] * scaled_component1.iloc[i, :]
            + opt["x"][1] * scaled_component2.iloc[i, :]
            + opt["x"][2] * scaled_component3.iloc[i, :]
        )
        print(weight)
        PCA3_weights.append(weight)

    PCA3InsampleWeights = PCA3InsampleWeights.append(short_sell_constraints(pd.DataFrame(PCA3_weights)))
    PCA3InsampleCoef = PCA3InsampleCoef.append(pd.DataFrame(PCA3_results))
    PCA3InsampleSE = PCA3InsampleSE.append(PCA3_se)

pca3insample_coef = PCA3InsampleCoef.mean()
PCA3OutofSampleWeights = pd.DataFrame()
PCA3OutofSampleReturn = pd.DataFrame()
for year in out_of_sample:

    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

    PCA3OutofSampleReturn = PCA3OutofSampleReturn.append(df_ret)

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

    outofsample_weights = []

    for i in range(len(df_ret)):
        weight = wb + nt * (
            + pca3insample_coef[0] * scaled_component1.iloc[i, :]
            + pca3insample_coef[1] * scaled_component2.iloc[i, :]
            + pca3insample_coef[1] * scaled_component3.iloc[i, :]
        )
```

```
        outofsample_weights.append(weight)
        print(weight)

    PCA3OutofSampleWeights = PCA3OutofSampleWeights.append(short_sell_constraints(pd.DataFrame(outofsample_weights)))

PCA4InsampleWeights = pd.DataFrame()
PCA4InsampleReturn = pd.DataFrame()
PCA4InsampleCoef = pd.DataFrame(np.zeros(4)).T
PCA4InsampleSE = pd.DataFrame()

rr = 5


for year in in_sample:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_PCA4_folder = './PCA Case/4 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled ret' + str(year) + '.csv').set_index('date')
    scaled_component1 = pd.read_csv(scaled_PCA4_folder + str(year) + '/component 1.csv').set_index('date')
    scaled_component2 = pd.read_csv(scaled_PCA4_folder + str(year) + '/component 2.csv').set_index('date')
    scaled_component3 = pd.read_csv(scaled_PCA4_folder + str(year) + '/component 3.csv').set_index('date')
    scaled_component4 = pd.read_csv(scaled_PCA4_folder + str(year) + '/component 4.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, :]
    scaled_component4 = scaled_component4.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
    scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T

    PCA4InsampleReturn = PCA4InsampleReturn.append(df_ret)

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

    PCA4_results = []
    PCA4_weights = []
    PCA4_se = []
```

```
    init_points = list(PCA4InsampleCoef.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_pca_4,
            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, :],
                scaled_component4.iloc[0 : i, :],
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
        print("The value:", opt["x"])
        PCA4_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
        PCA4_results.append(list(opt["x"]))
        weight = wb + nt * (
            opt["x"][0] * scaled_component1.iloc[i, :]
            + opt["x"][1] * scaled_component2.iloc[i, :]
            + opt["x"][2] * scaled_component3.iloc[i, :]
            + opt["x"][3] * scaled_component4.iloc[i, :]
        )
        print(weight)
        PCA4_weights.append(weight)

    PCA4InsampleWeights = PCA4InsampleWeights.append(short_sell_constraints(pd.DataFrame(PCA4_weights)))
    PCA4InsampleCoef = PCA4InsampleCoef.append(pd.DataFrame(PCA4_results))
    PCA4InsampleSE = PCA4InsampleSE.append(PCA4_se)

pca4insample_coef = PCA4InsampleCoef.mean()
PCA4OutofSampleWeights = pd.DataFrame()
PCA4OutofSampleReturn = pd.DataFrame()
for year in out_of_sample:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_PCA4_folder = './PCA Case/4 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled ret' + str(year) + '.csv').set_index('date')
    scaled_component1 = pd.read_csv(scaled_PCA4_folder + str(year) + '/component 1.csv').set_index('date')
    scaled_component2 = pd.read_csv(scaled_PCA4_folder + str(year) + '/component 2.csv').set_index('date')
    scaled_component3 = pd.read_csv(scaled_PCA4_folder + str(year) + '/component 3.csv').set_index('date')
```

```
    scaled_component4 = pd.read_csv(scaled_PCA4_folder + str(year) + '/component 4.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, :]
    scaled_component4 = scaled_component4.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
    scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T


    PCA4OutofSampleReturn = PCA4OutofSampleReturn.append(df_ret)

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

    outofsample_weights = []

    for i in range(len(df_ret)):
        weight = wb + nt * (
            + pca4insample_coef[0] * scaled_component1.iloc[i, :]
            + pca4insample_coef[1] * scaled_component2.iloc[i, :]
            + pca4insample_coef[2] * scaled_component3.iloc[i, :]
            + pca4insample_coef[3] * scaled_component4.iloc[i, :]
        )
        outofsample_weights.append(weight)
        print(weight)

    PCA4OutofSampleWeights = PCA4OutofSampleWeights.append(short_sell_constraints(pd.DataFrame(outofsample_weights)))


PCA5InsampleWeights = pd.DataFrame()
PCA5InsampleReturn = pd.DataFrame()
PCA5InsampleCoef = pd.DataFrame(np.zeros(5)).T
PCA5InsampleSE = pd.DataFrame()

rr = 5


for year in in_sample:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_PCA5_folder = './PCA Case/5 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled ret' + str(year) + '.csv').set_index('date')
```

```
    scaled_component1 = pd.read_csv(scaled_PCA5_folder + str(year) + '/component 1.csv').set_index('date')
    scaled_component2 = pd.read_csv(scaled_PCA5_folder + str(year) + '/component 2.csv').set_index('date')
    scaled_component3 = pd.read_csv(scaled_PCA5_folder + str(year) + '/component 3.csv').set_index('date')
    scaled_component4 = pd.read_csv(scaled_PCA5_folder + str(year) + '/component 4.csv').set_index('date')
    scaled_component5 = pd.read_csv(scaled_PCA5_folder + str(year) + '/component 5.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, :]
    scaled_component4 = scaled_component4.loc[quarter_index, :]
    scaled_component5 = scaled_component5.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
    scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T
    scaled_component5 =  pd.DataFrame(Scale(scaled_component5.T)).T

    PCA5InsampleReturn = PCA5InsampleReturn.append(df_ret)

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

    PCA5_results = []
    PCA5_weights = []
    PCA5_se = []
    init_points = list(PCA5InsampleCoef.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_pca_5,
            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, :],
                scaled_component4.iloc[0 : i, :],
                scaled_component5.iloc[0 : i, :],
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
```

```
        print("The value:", opt["x"])
        PCA5_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
        PCA5_results.append(list(opt["x"]))
        weight = wb + nt * (
            opt["x"][0] * scaled_component1.iloc[i, :]
            + opt["x"][1] * scaled_component2.iloc[i, :]
            + opt["x"][2] * scaled_component3.iloc[i, :]
            + opt["x"][3] * scaled_component4.iloc[i, :]
            + opt["x"][4] * scaled_component5.iloc[i, :]
        )
        print(weight)
        PCA5_weights.append(weight)

    PCA5InsampleWeights = PCA5InsampleWeights.append(short_sell_constraints(pd.DataFrame(PCA5_weights)))
    PCA5InsampleCoef = PCA5InsampleCoef.append(pd.DataFrame(PCA5_results))
    PCA5InsampleSE = PCA5InsampleSE.append(PCA5_se)

pca5insample_coef = PCA5InsampleCoef.mean()
PCA5OutofSampleWeights = pd.DataFrame()
PCA5OutofSampleReturn = pd.DataFrame()
for year in out_of_sample:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_PCA5_folder = './PCA Case/5 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled ret' + str(year) + '.csv').set_index('date')
    scaled_component1 = pd.read_csv(scaled_PCA5_folder + str(year) + '/component 1.csv').set_index('date')
    scaled_component2 = pd.read_csv(scaled_PCA5_folder + str(year) + '/component 2.csv').set_index('date')
    scaled_component3 = pd.read_csv(scaled_PCA5_folder + str(year) + '/component 3.csv').set_index('date')
    scaled_component4 = pd.read_csv(scaled_PCA5_folder + str(year) + '/component 4.csv').set_index('date')
    scaled_component5 = pd.read_csv(scaled_PCA5_folder + str(year) + '/component 5.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, :]
    scaled_component4 = scaled_component4.loc[quarter_index, :]
    scaled_component5 = scaled_component5.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
```

```
    scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T
    scaled_component5 =  pd.DataFrame(Scale(scaled_component5.T)).T

    PCA5OutofSampleReturn = PCA5OutofSampleReturn.append(df_ret)

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

    outofsample_weights = []

    for i in range(len(df_ret)):
        weight = wb + nt * (
            + pca5insample_coef[0] * scaled_component1.iloc[i, :]
            + pca5insample_coef[1] * scaled_component2.iloc[i, :]
            + pca5insample_coef[2] * scaled_component3.iloc[i, :]
            + pca5insample_coef[3] * scaled_component4.iloc[i, :]
            + pca5insample_coef[4] * scaled_component5.iloc[i, :]
        )
        outofsample_weights.append(weight)
        print(weight)

    PCA5OutofSampleWeights = PCA5OutofSampleWeights.append(short_sell_constraints(pd.DataFrame(outofsample_weights)))

PCA6InsampleWeights = pd.DataFrame()
PCA6InsampleReturn = pd.DataFrame()
PCA6InsampleCoef = pd.DataFrame(np.zeros(6)).T
PCA6InsampleSE = pd.DataFrame()

rr = 5


for year in in_sample:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_PCA6_folder = './PCA Case/6 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled ret' + str(year) + '.csv').set_index('date')
    scaled_component1 = pd.read_csv(scaled_PCA6_folder + str(year) + '/component 1.csv').set_index('date')
    scaled_component2 = pd.read_csv(scaled_PCA6_folder + str(year) + '/component 2.csv').set_index('date')
    scaled_component3 = pd.read_csv(scaled_PCA6_folder + str(year) + '/component 3.csv').set_index('date')
    scaled_component4 = pd.read_csv(scaled_PCA6_folder + str(year) + '/component 4.csv').set_index('date')
    scaled_component5 = pd.read_csv(scaled_PCA6_folder + str(year) + '/component 5.csv').set_index('date')
    scaled_component6 = pd.read_csv(scaled_PCA6_folder + str(year) + '/component 6.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, :]
    scaled_component4 = scaled_component4.loc[quarter_index, :]
    scaled_component5 = scaled_component5.loc[quarter_index, :]
    scaled_component6 = scaled_component6.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
    scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T
    scaled_component5 =  pd.DataFrame(Scale(scaled_component5.T)).T
    scaled_component6 =  pd.DataFrame(Scale(scaled_component6.T)).T
    PCA6InsampleReturn = PCA6InsampleReturn.append(df_ret)

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

    PCA6_results = []
    PCA6_weights = []
    PCA6_se = []
    init_points = list(PCA6InsampleCoef.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_pca_6,
            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, :],
                scaled_component4.iloc[0 : i, :],
                scaled_component5.iloc[0 : i, :],
                scaled_component6.iloc[0 : i, :],
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
        print("The value:", opt["x"])
        PCA6_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
        PCA6_results.append(list(opt["x"]))
        weight = wb + nt * (
            opt["x"][0] * scaled_component1.iloc[i, :]
            + opt["x"][1] * scaled_component2.iloc[i, :]
            + opt["x"][2] * scaled_component3.iloc[i, :]
            + opt["x"][3] * scaled_component4.iloc[i, :]
            + opt["x"][4] * scaled_component5.iloc[i, :]
            + opt["x"][5] * scaled_component6.iloc[i, :]
        )
        print(weight)
```

```
        PCA6_weights.append(weight)

    PCA6InsampleWeights = PCA6InsampleWeights.append(
    short_sell_constraints(pd.DataFrame(PCA6_weights)))
    PCA6InsampleCoef = PCA6InsampleCoef.append(pd.DataFrame(
    PCA6_results))
    PCA6InsampleSE = PCA6InsampleSE.append(PCA6_se)

pca6insample_coef = PCA6InsampleCoef.mean()
PCA6OutofSampleWeights = pd.DataFrame()
PCA6OutofSampleReturn = pd.DataFrame()
for year in out_of_sample:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
    set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_PCA6_folder = './PCA Case/6 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
    ret' + str(year) + '.csv').set_index('date')
    scaled_component1 = pd.read_csv(scaled_PCA6_folder + str(year)
    + '/component 1.csv').set_index('date')
    scaled_component2 = pd.read_csv(scaled_PCA6_folder + str(year)
    + '/component 2.csv').set_index('date')
    scaled_component3 = pd.read_csv(scaled_PCA6_folder + str(year)
    + '/component 3.csv').set_index('date')
    scaled_component4 = pd.read_csv(scaled_PCA6_folder + str(year)
    + '/component 4.csv').set_index('date')
    scaled_component5 = pd.read_csv(scaled_PCA6_folder + str(year)
    + '/component 5.csv').set_index('date')
    scaled_component6 = pd.read_csv(scaled_PCA6_folder + str(year)
    + '/component 6.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, :]
    scaled_component4 = scaled_component4.loc[quarter_index, :]
    scaled_component5 = scaled_component5.loc[quarter_index, :]
    scaled_component6 = scaled_component6.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
    scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T
    scaled_component5 =  pd.DataFrame(Scale(scaled_component5.T)).T
    scaled_component6 =  pd.DataFrame(Scale(scaled_component6.T)).T

    PCA6OutofSampleReturn = PCA6OutofSampleReturn.append(df_ret)

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

```

```
    outofsample_weights = []

    for i in range(len(df_ret)):
        weight = wb + nt * (
            + pca6insample_coef[0] * scaled_component1.iloc[i, :]
            + pca6insample_coef[1] * scaled_component2.iloc[i, :]
            + pca6insample_coef[2] * scaled_component3.iloc[i, :]
            + pca6insample_coef[3] * scaled_component4.iloc[i, :]
            + pca6insample_coef[4] * scaled_component5.iloc[i, :]
            + pca6insample_coef[5] * scaled_component6.iloc[i, :]
        )
        outofsample_weights.append(weight)
        print(weight)

    PCA6OutofSampleWeights = PCA6OutofSampleWeights.append(short_sell_constraints(pd.DataFrame(outofsample_weights)))

PCA7InsampleWeights = pd.DataFrame()
PCA7InsampleReturn = pd.DataFrame()
PCA7InsampleCoef = pd.DataFrame(np.zeros(7)).T
PCA7InsampleSE = pd.DataFrame()

rr = 5


for year in in_sample:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_PCA7_folder = './PCA Case/7 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled ret' + str(year) + '.csv').set_index('date')
    scaled_component1 = pd.read_csv(scaled_PCA7_folder + str(year) + '/component 1.csv').set_index('date')
    scaled_component2 = pd.read_csv(scaled_PCA7_folder + str(year) + '/component 2.csv').set_index('date')
    scaled_component3 = pd.read_csv(scaled_PCA7_folder + str(year) + '/component 3.csv').set_index('date')
    scaled_component4 = pd.read_csv(scaled_PCA7_folder + str(year) + '/component 4.csv').set_index('date')
    scaled_component5 = pd.read_csv(scaled_PCA7_folder + str(year) + '/component 5.csv').set_index('date')
    scaled_component6 = pd.read_csv(scaled_PCA7_folder + str(year) + '/component 6.csv').set_index('date')
    scaled_component7 = pd.read_csv(scaled_PCA7_folder + str(year) + '/component 7.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, :]
    scaled_component4 = scaled_component4.loc[quarter_index, :]
```

```
    scaled_component5 = scaled_component5.loc[quarter_index, :]
    scaled_component6 = scaled_component6.loc[quarter_index, :]
    scaled_component7 = scaled_component7.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
    scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T
    scaled_component5 =  pd.DataFrame(Scale(scaled_component5.T)).T
    scaled_component6 =  pd.DataFrame(Scale(scaled_component6.T)).T
    scaled_component7 =  pd.DataFrame(Scale(scaled_component7.T)).T
    PCA7InsampleReturn = PCA7InsampleReturn.append(df_ret)

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

    PCA7_results = []
    PCA7_weights = []
    PCA7_se = []
    init_points = list(PCA7InsampleCoef.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_pca_7,
            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, :],
                scaled_component4.iloc[0 : i, :],
                scaled_component5.iloc[0 : i, :],
                scaled_component6.iloc[0 : i, :],
                scaled_component7.iloc[0 : i, :],
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
        print("The value:", opt["x"])
        PCA7_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
        PCA7_results.append(list(opt["x"]))
        weight = wb + nt * (
            opt["x"][0] * scaled_component1.iloc[i, :]
            + opt["x"][1] * scaled_component2.iloc[i, :]
            + opt["x"][2] * scaled_component3.iloc[i, :]
            + opt["x"][3] * scaled_component4.iloc[i, :]
            + opt["x"][4] * scaled_component5.iloc[i, :]
            + opt["x"][5] * scaled_component6.iloc[i, :]
            + opt["x"][6] * scaled_component7.iloc[i, :]
        )
        print(weight)
        PCA7_weights.append(weight)
```

```python

    PCA7InsampleWeights = PCA7InsampleWeights.append(short_sell_constraints(pd.DataFrame(PCA7_weights)))
    PCA7InsampleCoef = PCA7InsampleCoef.append(pd.DataFrame(PCA7_results))
    PCA7InsampleSE = PCA7InsampleSE.append(PCA7_se)

pca7insample_coef = PCA7InsampleCoef.mean()
PCA7OutofSampleWeights = pd.DataFrame()
PCA7OutofSampleReturn = pd.DataFrame()
for year in out_of_sample:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_PCA7_folder = './PCA Case/7 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled ret' + str(year) + '.csv').set_index('date')
    scaled_component1 = pd.read_csv(scaled_PCA7_folder + str(year) + '/component 1.csv').set_index('date')
    scaled_component2 = pd.read_csv(scaled_PCA7_folder + str(year) + '/component 2.csv').set_index('date')
    scaled_component3 = pd.read_csv(scaled_PCA7_folder + str(year) + '/component 3.csv').set_index('date')
    scaled_component4 = pd.read_csv(scaled_PCA7_folder + str(year) + '/component 4.csv').set_index('date')
    scaled_component5 = pd.read_csv(scaled_PCA7_folder + str(year) + '/component 5.csv').set_index('date')
    scaled_component6 = pd.read_csv(scaled_PCA7_folder + str(year) + '/component 6.csv').set_index('date')
    scaled_component7 = pd.read_csv(scaled_PCA7_folder + str(year) + '/component 7.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, :]
    scaled_component4 = scaled_component4.loc[quarter_index, :]
    scaled_component5 = scaled_component5.loc[quarter_index, :]
    scaled_component6 = scaled_component6.loc[quarter_index, :]
    scaled_component7 = scaled_component7.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
    scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T
    scaled_component5 =  pd.DataFrame(Scale(scaled_component5.T)).T
    scaled_component6 =  pd.DataFrame(Scale(scaled_component6.T)).T
    scaled_component7 =  pd.DataFrame(Scale(scaled_component7.T)).T

    PCA7OutofSampleReturn = PCA7OutofSampleReturn.append(df_ret)
```

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

    outofsample_weights = []

    for i in range(len(df_ret)):
        weight = wb + nt * (
            + pca7insample_coef[0] * scaled_component1.iloc[i, :]
            + pca7insample_coef[1] * scaled_component2.iloc[i, :]
            + pca7insample_coef[2] * scaled_component3.iloc[i, :]
            + pca7insample_coef[3] * scaled_component4.iloc[i, :]
            + pca7insample_coef[4] * scaled_component5.iloc[i, :]
            + pca7insample_coef[5] * scaled_component6.iloc[i, :]
            + pca7insample_coef[6] * scaled_component7.iloc[i, :]
        )
        outofsample_weights.append(weight)
        print(weight)

    PCA7OutofSampleWeights = PCA7OutofSampleWeights.append(
    short_sell_constraints(pd.DataFrame(outofsample_weights)))


PCA8InsampleWeights = pd.DataFrame()
PCA8InsampleReturn = pd.DataFrame()
PCA8InsampleCoef = pd.DataFrame(np.zeros(8)).T
PCA8InsampleSE = pd.DataFrame()

rr = 5


for year in in_sample:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
    set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_PCA8_folder = './PCA Case/8 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled
    ret' + str(year) + '.csv').set_index('date')
    scaled_component1 = pd.read_csv(scaled_PCA8_folder + str(year)
    + '/component 1.csv').set_index('date')
    scaled_component2 = pd.read_csv(scaled_PCA8_folder + str(year)
    + '/component 2.csv').set_index('date')
    scaled_component3 = pd.read_csv(scaled_PCA8_folder + str(year)
    + '/component 3.csv').set_index('date')
    scaled_component4 = pd.read_csv(scaled_PCA8_folder + str(year)
    + '/component 4.csv').set_index('date')
    scaled_component5 = pd.read_csv(scaled_PCA8_folder + str(year)
    + '/component 5.csv').set_index('date')
    scaled_component6 = pd.read_csv(scaled_PCA8_folder + str(year)
    + '/component 6.csv').set_index('date')
    scaled_component7 = pd.read_csv(scaled_PCA8_folder + str(year)
    + '/component 7.csv').set_index('date')
    scaled_component8 = pd.read_csv(scaled_PCA8_folder + str(year)
    + '/component 8.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, :]
    scaled_component4 = scaled_component4.loc[quarter_index, :]
    scaled_component5 = scaled_component5.loc[quarter_index, :]
    scaled_component6 = scaled_component6.loc[quarter_index, :]
    scaled_component7 = scaled_component7.loc[quarter_index, :]
    scaled_component8 = scaled_component8.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
    scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T
    scaled_component5 =  pd.DataFrame(Scale(scaled_component5.T)).T
    scaled_component6 =  pd.DataFrame(Scale(scaled_component6.T)).T
    scaled_component7 =  pd.DataFrame(Scale(scaled_component7.T)).T
    scaled_component8 =  pd.DataFrame(Scale(scaled_component8.T)).T
    PCA8InsampleReturn = PCA8InsampleReturn.append(df_ret)

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

    PCA8_results = []
    PCA8_weights = []
    PCA8_se = []
    init_points = list(PCA8InsampleCoef.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_pca_8,
            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, :],
                scaled_component4.iloc[0 : i, :],
                scaled_component5.iloc[0 : i, :],
                scaled_component6.iloc[0 : i, :],
                scaled_component7.iloc[0 : i, :],
                scaled_component8.iloc[0 : i, :],
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
        print("The value:", opt["x"])
        PCA8_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
        PCA8_results.append(list(opt["x"]))
```

```
        weight = wb + nt * (
            opt["x"][0] * scaled_component1.iloc[i, :]
            + opt["x"][1] * scaled_component2.iloc[i, :]
            + opt["x"][2] * scaled_component3.iloc[i, :]
            + opt["x"][3] * scaled_component4.iloc[i, :]
            + opt["x"][4] * scaled_component5.iloc[i, :]
            + opt["x"][5] * scaled_component6.iloc[i, :]
            + opt["x"][6] * scaled_component7.iloc[i, :]
            + opt["x"][7] * scaled_component8.iloc[i, :]
        )
        print(weight)
        PCA8_weights.append(weight)

    PCA8InsampleWeights = PCA8InsampleWeights.append(short_sell_constraints(pd.DataFrame(PCA8_weights)))
    PCA8InsampleCoef = PCA8InsampleCoef.append(pd.DataFrame(PCA8_results))
    PCA8InsampleSE = PCA8InsampleSE.append(PCA8_se)

pca8insample_coef = PCA8InsampleCoef.mean()
PCA8OutofSampleWeights = pd.DataFrame()
PCA8OutofSampleReturn = pd.DataFrame()
for year in out_of_sample:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_PCA8_folder = './PCA Case/8 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled ret' + str(year) + '.csv').set_index('date')
    scaled_component1 = pd.read_csv(scaled_PCA8_folder + str(year) + '/component 1.csv').set_index('date')
    scaled_component2 = pd.read_csv(scaled_PCA8_folder + str(year) + '/component 2.csv').set_index('date')
    scaled_component3 = pd.read_csv(scaled_PCA8_folder + str(year) + '/component 3.csv').set_index('date')
    scaled_component4 = pd.read_csv(scaled_PCA8_folder + str(year) + '/component 4.csv').set_index('date')
    scaled_component5 = pd.read_csv(scaled_PCA8_folder + str(year) + '/component 5.csv').set_index('date')
    scaled_component6 = pd.read_csv(scaled_PCA8_folder + str(year) + '/component 6.csv').set_index('date')
    scaled_component7 = pd.read_csv(scaled_PCA8_folder + str(year) + '/component 7.csv').set_index('date')
    scaled_component8 = pd.read_csv(scaled_PCA8_folder + str(year) + '/component 8.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, :]
    scaled_component4 = scaled_component4.loc[quarter_index, :]
```

```
    scaled_component5 = scaled_component5.loc[quarter_index, :]
    scaled_component6 = scaled_component6.loc[quarter_index, :]
    scaled_component7 = scaled_component7.loc[quarter_index, :]
    scaled_component8 = scaled_component8.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
    scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T
    scaled_component5 =  pd.DataFrame(Scale(scaled_component5.T)).T
    scaled_component6 =  pd.DataFrame(Scale(scaled_component6.T)).T
    scaled_component7 =  pd.DataFrame(Scale(scaled_component7.T)).T
    scaled_component8 =  pd.DataFrame(Scale(scaled_component8.T)).T

    PCA8OutofSampleReturn = PCA8OutofSampleReturn.append(df_ret)

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

    outofsample_weights = []

    for i in range(len(df_ret)):
        weight = wb + nt * (
            + pca8insample_coef[0] * scaled_component1.iloc[i, :]
            + pca8insample_coef[1] * scaled_component2.iloc[i, :]
            + pca8insample_coef[2] * scaled_component3.iloc[i, :]
            + pca8insample_coef[3] * scaled_component4.iloc[i, :]
            + pca8insample_coef[4] * scaled_component5.iloc[i, :]
            + pca8insample_coef[5] * scaled_component6.iloc[i, :]
            + pca8insample_coef[6] * scaled_component7.iloc[i, :]
            + pca8insample_coef[7] * scaled_component8.iloc[i, :]
        )
        outofsample_weights.append(weight)
        print(weight)

    PCA8OutofSampleWeights = PCA8OutofSampleWeights.append(
   short_sell_constraints(pd.DataFrame(outofsample_weights)))

### Risk Aversion
## Base Case

rr = 1

BaseWeights1 = pd.DataFrame()
BaseReturn1 = pd.DataFrame()

BaseCoef1 = pd.DataFrame(np.zeros(11)).T
BaseSE1 = pd.DataFrame()

year_list = range(1970, 2021)

for year in year_list:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').
   set_index('date')

```

```
    scaled_data_folder = './new standardized5/'
    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/scaled ret' + str(year) + '.csv').set_index('date')
    scaled_mktcap = pd.read_csv(scaled_data_folder + 'mktcap/mktcap' + str(year) + '.csv').set_index('date')
    scaled_bm = pd.read_csv(scaled_data_folder + 'bm/bm' + str(year) + '.csv').set_index('date')
    scaled_roa = pd.read_csv(scaled_data_folder + 'roa/roa' + str(year) + '.csv').set_index('date')
    scaled_roe = pd.read_csv(scaled_data_folder + 'roe/roe' + str(year) + '.csv').set_index('date')
    scaled_accrual = pd.read_csv(scaled_data_folder + 'accrual/accrual' + str(year) + '.csv').set_index('date')
    scaled_cfm = pd.read_csv(scaled_data_folder + 'cfm/cfm' + str(year) + '.csv').set_index('date')
    scaled_eqinv = pd.read_csv(scaled_data_folder + 'equity invcap/equity invcap' + str(year) + '.csv').set_index('date')
    scaled_atturn = pd.read_csv(scaled_data_folder + 'at turn/at turn' + str(year) + '.csv').set_index('date')
    scaled_pcf = pd.read_csv(scaled_data_folder + 'pcf/pcf' + str(year) + '.csv').set_index('date')
    scaled_da = pd.read_csv(scaled_data_folder + 'debt asset/debt asset' + str(year) + '.csv').set_index('date')
    scaled_curr = pd.read_csv(scaled_data_folder + 'curr ratio/curr ratio' + str(year) + '.csv').set_index('date')

    quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(year)+'/09/30',str(year)+'/12/31']
    df_ret = df_ret.loc[quarter_index, :]
    scaled_ret = scaled_ret.loc[quarter_index, :]
    scaled_mktcap = scaled_mktcap.loc[quarter_index, :]
    scaled_bm = scaled_bm.loc[quarter_index, :]
    scaled_roa = scaled_roa.loc[quarter_index, :]
    scaled_roe = scaled_roe.loc[quarter_index, :]
    scaled_accrual = scaled_accrual.loc[quarter_index, :]
    scaled_cfm = scaled_cfm.loc[quarter_index, :]
    scaled_eqinv = scaled_eqinv.loc[quarter_index, :]
    scaled_atturn = scaled_atturn.loc[quarter_index, :]
    scaled_pcf = scaled_pcf.loc[quarter_index, :]
    scaled_da = scaled_da.loc[quarter_index, :]
    scaled_curr = scaled_curr.loc[quarter_index, :]

    BaseReturn1 = BaseReturn1.append(df_ret)

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

    Base_results = []
    Base_weights = []
    Base_SE = []
    init_points = list(BaseCoef1.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_base,
            init_points,
            method="BFGS",
```

```
            args=(
                wb,
                nt,
                scaled_ret.iloc[0 : i, :],
                scaled_mktcap.iloc[0 : i, :],
                scaled_bm.iloc[0 : i, :],
                scaled_roa.iloc[0 : i, :],
                scaled_roe.iloc[0 : i, :],
                scaled_accrual.iloc[0 : i, :],
                scaled_eqinv.iloc[0 : i, :],
                scaled_atturn.iloc[0 : i, :],
                scaled_cfm.iloc[0 : i, :],
                scaled_curr.iloc[0 : i, :],
                scaled_da.iloc[0 : i, :],
                scaled_pcf.iloc[0 : i, :],
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
        print("The value:", opt["x"])
        Base_results.append(list(opt["x"]))

        Base_SE.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
        weight = wb + nt * (
            + opt["x"][0] * scaled_mktcap.iloc[i, :]
            + opt["x"][1] * scaled_bm.iloc[i, :]
            + opt["x"][2] * scaled_roa.iloc[i, :]
            + opt["x"][3] * scaled_roe.iloc[i, :]
            + opt["x"][4] * scaled_accrual.iloc[i, :]
            + opt["x"][5] * scaled_eqinv.iloc[i, :]
            + opt["x"][6] * scaled_atturn.iloc[i, :]
            + opt["x"][7] * scaled_cfm.iloc[i, :]
            + opt["x"][8] * scaled_curr.iloc[i, :]
            + opt["x"][9] * scaled_da.iloc[i, :]
            + opt["x"][10] * scaled_pcf.iloc[i, :]
        )
        print(weight)
        Base_weights.append(weight)

    BaseWeights1 = BaseWeights1.append(short_sell_constraints(pd.
   DataFrame(Base_weights)))
    BaseCoef1 = BaseCoef1.append(pd.DataFrame(Base_results))
    BaseSE1 = BaseSE1.append(pd.DataFrame(Base_SE))

rr = 3

BaseWeights3 = pd.DataFrame()
BaseReturn3 = pd.DataFrame()

BaseCoef3 = pd.DataFrame(np.zeros(11)).T
BaseSE3 = pd.DataFrame()

year_list = range(1970, 2021)

for year in year_list:

```

```
    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/scaled ret' + str(year) + '.csv').set_index('date')
    scaled_mktcap = pd.read_csv(scaled_data_folder + 'mktcap/mktcap' + str(year) + '.csv').set_index('date')
    scaled_bm = pd.read_csv(scaled_data_folder + 'bm/bm' + str(year) + '.csv').set_index('date')
    scaled_roa = pd.read_csv(scaled_data_folder + 'roa/roa' + str(year) + '.csv').set_index('date')
    scaled_roe = pd.read_csv(scaled_data_folder + 'roe/roe' + str(year) + '.csv').set_index('date')
    scaled_accrual = pd.read_csv(scaled_data_folder + 'accrual/accrual' + str(year) + '.csv').set_index('date')
    scaled_cfm = pd.read_csv(scaled_data_folder + 'cfm/cfm' + str(year) + '.csv').set_index('date')
    scaled_eqinv = pd.read_csv(scaled_data_folder + 'equity invcap/equity invcap' + str(year) + '.csv').set_index('date')
    scaled_atturn = pd.read_csv(scaled_data_folder + 'at turn/at turn' + str(year) + '.csv').set_index('date')
    scaled_pcf = pd.read_csv(scaled_data_folder + 'pcf/pcf' + str(year) + '.csv').set_index('date')
    scaled_da = pd.read_csv(scaled_data_folder + 'debt asset/debt asset' + str(year) + '.csv').set_index('date')
    scaled_curr = pd.read_csv(scaled_data_folder + 'curr ratio/curr ratio' + str(year) + '.csv').set_index('date')

    quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(year)+'/09/30',str(year)+'/12/31']
    df_ret = df_ret.loc[quarter_index, :]
    scaled_ret = scaled_ret.loc[quarter_index, :]
    scaled_mktcap = scaled_mktcap.loc[quarter_index, :]
    scaled_bm = scaled_bm.loc[quarter_index, :]
    scaled_roa = scaled_roa.loc[quarter_index, :]
    scaled_roe = scaled_roe.loc[quarter_index, :]
    scaled_accrual = scaled_accrual.loc[quarter_index, :]
    scaled_cfm = scaled_cfm.loc[quarter_index, :]
    scaled_eqinv = scaled_eqinv.loc[quarter_index, :]
    scaled_atturn = scaled_atturn.loc[quarter_index, :]
    scaled_pcf = scaled_pcf.loc[quarter_index, :]
    scaled_da = scaled_da.loc[quarter_index, :]
    scaled_curr = scaled_curr.loc[quarter_index, :]

    BaseReturn3 = BaseReturn3.append(df_ret)

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

    Base_results = []
    Base_weights = []
    Base_SE = []
    init_points = list(BaseCoef3.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
```

```
                PPS_base,
                init_points,
                method="BFGS",
                args=(
                    wb,
                    nt,
                    scaled_ret.iloc[0 : i, :],
                    scaled_mktcap.iloc[0 : i, :],
                    scaled_bm.iloc[0 : i, :],
                    scaled_roa.iloc[0 : i, :],
                    scaled_roe.iloc[0 : i, :],
                    scaled_accrual.iloc[0 : i, :],
                    scaled_eqinv.iloc[0 : i, :],
                    scaled_atturn.iloc[0 : i, :],
                    scaled_cfm.iloc[0 : i, :],
                    scaled_curr.iloc[0 : i, :],
                    scaled_da.iloc[0 : i, :],
                    scaled_pcf.iloc[0 : i, :],
                    rr,
                ),
            )
            print("The {} window for year {}".format(i+1, year))
            print("The value:", opt["x"])
            Base_results.append(list(opt["x"]))

            Base_SE.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
            weight = wb + nt * (
                + opt["x"][0] * scaled_mktcap.iloc[i, :]
                + opt["x"][1] * scaled_bm.iloc[i, :]
                + opt["x"][2] * scaled_roa.iloc[i, :]
                + opt["x"][3] * scaled_roe.iloc[i, :]
                + opt["x"][4] * scaled_accrual.iloc[i, :]
                + opt["x"][5] * scaled_eqinv.iloc[i, :]
                + opt["x"][6] * scaled_atturn.iloc[i, :]
                + opt["x"][7] * scaled_cfm.iloc[i, :]
                + opt["x"][8] * scaled_curr.iloc[i, :]
                + opt["x"][9] * scaled_da.iloc[i, :]
                + opt["x"][10] * scaled_pcf.iloc[i, :]
            )
            print(weight)
            Base_weights.append(weight)

    BaseWeights3 = BaseWeights3.append(short_sell_constraints(pd.
  DataFrame(Base_weights)))
    BaseCoef3 = BaseCoef3.append(pd.DataFrame(Base_results))
    BaseSE3 = BaseSE3.append(pd.DataFrame(Base_SE))

rr = 7

BaseWeights7 = pd.DataFrame()
BaseReturn7 = pd.DataFrame()

BaseCoef7 = pd.DataFrame(np.zeros(11)).T
BaseSE7 = pd.DataFrame()

year_list = range(1970, 2021)
```

```python

for year in year_list:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/scaled ret' + str(year) + '.csv').set_index('date')
    scaled_mktcap = pd.read_csv(scaled_data_folder + 'mktcap/mktcap' + str(year) + '.csv').set_index('date')
    scaled_bm = pd.read_csv(scaled_data_folder + 'bm/bm' + str(year) + '.csv').set_index('date')
    scaled_roa = pd.read_csv(scaled_data_folder + 'roa/roa' + str(year) + '.csv').set_index('date')
    scaled_roe = pd.read_csv(scaled_data_folder + 'roe/roe' + str(year) + '.csv').set_index('date')
    scaled_accrual = pd.read_csv(scaled_data_folder + 'accrual/accrual' + str(year) + '.csv').set_index('date')
    scaled_cfm = pd.read_csv(scaled_data_folder + 'cfm/cfm' + str(year) + '.csv').set_index('date')
    scaled_eqinv = pd.read_csv(scaled_data_folder + 'equity invcap/equity invcap' + str(year) + '.csv').set_index('date')
    scaled_atturn = pd.read_csv(scaled_data_folder + 'at turn/at turn' + str(year) + '.csv').set_index('date')
    scaled_pcf = pd.read_csv(scaled_data_folder + 'pcf/pcf' + str(year) + '.csv').set_index('date')
    scaled_da = pd.read_csv(scaled_data_folder + 'debt asset/debt asset' + str(year) + '.csv').set_index('date')
    scaled_curr = pd.read_csv(scaled_data_folder + 'curr ratio/curr ratio' + str(year) + '.csv').set_index('date')

    quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(year)+'/09/30',str(year)+'/12/31']
    df_ret = df_ret.loc[quarter_index, :]
    scaled_ret = scaled_ret.loc[quarter_index, :]
    scaled_mktcap = scaled_mktcap.loc[quarter_index, :]
    scaled_bm = scaled_bm.loc[quarter_index, :]
    scaled_roa = scaled_roa.loc[quarter_index, :]
    scaled_roe = scaled_roe.loc[quarter_index, :]
    scaled_accrual = scaled_accrual.loc[quarter_index, :]
    scaled_cfm = scaled_cfm.loc[quarter_index, :]
    scaled_eqinv = scaled_eqinv.loc[quarter_index, :]
    scaled_atturn = scaled_atturn.loc[quarter_index, :]
    scaled_pcf = scaled_pcf.loc[quarter_index, :]
    scaled_da = scaled_da.loc[quarter_index, :]
    scaled_curr = scaled_curr.loc[quarter_index, :]

    BaseReturn7 = BaseReturn7.append(df_ret)

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

    Base_results = []
    Base_weights = []
    Base_SE = []
    init_points = list(BaseCoef7.iloc[-1,:].values)
```

```
    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_base,
            init_points,
            method="BFGS",
            args=(
                wb,
                nt,
                scaled_ret.iloc[0 : i, :],
                scaled_mktcap.iloc[0 : i, :],
                scaled_bm.iloc[0 : i, :],
                scaled_roa.iloc[0 : i, :],
                scaled_roe.iloc[0 : i, :],
                scaled_accrual.iloc[0 : i, :],
                scaled_eqinv.iloc[0 : i, :],
                scaled_atturn.iloc[0 : i, :],
                scaled_cfm.iloc[0 : i, :],
                scaled_curr.iloc[0 : i, :],
                scaled_da.iloc[0 : i, :],
                scaled_pcf.iloc[0 : i, :],
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
        print("The value:", opt["x"])
        Base_results.append(list(opt["x"]))

        Base_SE.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
        weight = wb + nt * (
            + opt["x"][0] * scaled_mktcap.iloc[i, :]
            + opt["x"][1] * scaled_bm.iloc[i, :]
            + opt["x"][2] * scaled_roa.iloc[i, :]
            + opt["x"][3] * scaled_roe.iloc[i, :]
            + opt["x"][4] * scaled_accrual.iloc[i, :]
            + opt["x"][5] * scaled_eqinv.iloc[i, :]
            + opt["x"][6] * scaled_atturn.iloc[i, :]
            + opt["x"][7] * scaled_cfm.iloc[i, :]
            + opt["x"][8] * scaled_curr.iloc[i, :]
            + opt["x"][9] * scaled_da.iloc[i, :]
            + opt["x"][10] * scaled_pcf.iloc[i, :]
        )
        print(weight)
        Base_weights.append(weight)

    BaseWeights7 = BaseWeights7.append(short_sell_constraints(pd.
   DataFrame(Base_weights)))
    BaseCoef7 = BaseCoef7.append(pd.DataFrame(Base_results))
    BaseSE7 = BaseSE7.append(pd.DataFrame(Base_SE))

rr = 9

BaseWeights9 = pd.DataFrame()
BaseReturn9 = pd.DataFrame()

BaseCoef9 = pd.DataFrame(np.zeros(11)).T
```

```
BaseSE9 = pd.DataFrame()

year_list = range(1970, 2021)

for year in year_list:

    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/scaled ret' + str(year) + '.csv').set_index('date')
    scaled_mktcap = pd.read_csv(scaled_data_folder + 'mktcap/mktcap' + str(year) + '.csv').set_index('date')
    scaled_bm = pd.read_csv(scaled_data_folder + 'bm/bm' + str(year) + '.csv').set_index('date')
    scaled_roa = pd.read_csv(scaled_data_folder + 'roa/roa' + str(year) + '.csv').set_index('date')
    scaled_roe = pd.read_csv(scaled_data_folder + 'roe/roe' + str(year) + '.csv').set_index('date')
    scaled_accrual = pd.read_csv(scaled_data_folder + 'accrual/accrual' + str(year) + '.csv').set_index('date')
    scaled_cfm = pd.read_csv(scaled_data_folder + 'cfm/cfm' + str(year) + '.csv').set_index('date')
    scaled_eqinv = pd.read_csv(scaled_data_folder + 'equity invcap/equity invcap' + str(year) + '.csv').set_index('date')
    scaled_atturn = pd.read_csv(scaled_data_folder + 'at turn/at turn' + str(year) + '.csv').set_index('date')
    scaled_pcf = pd.read_csv(scaled_data_folder + 'pcf/pcf' + str(year) + '.csv').set_index('date')
    scaled_da = pd.read_csv(scaled_data_folder + 'debt asset/debt asset' + str(year) + '.csv').set_index('date')
    scaled_curr = pd.read_csv(scaled_data_folder + 'curr ratio/curr ratio' + str(year) + '.csv').set_index('date')

    quarter_index = [str(year)+'/03/31', str(year)+'/06/30', str(year)+'/09/30',str(year)+'/12/31']
    df_ret = df_ret.loc[quarter_index, :]
    scaled_ret = scaled_ret.loc[quarter_index, :]
    scaled_mktcap = scaled_mktcap.loc[quarter_index, :]
    scaled_bm = scaled_bm.loc[quarter_index, :]
    scaled_roa = scaled_roa.loc[quarter_index, :]
    scaled_roe = scaled_roe.loc[quarter_index, :]
    scaled_accrual = scaled_accrual.loc[quarter_index, :]
    scaled_cfm = scaled_cfm.loc[quarter_index, :]
    scaled_eqinv = scaled_eqinv.loc[quarter_index, :]
    scaled_atturn = scaled_atturn.loc[quarter_index, :]
    scaled_pcf = scaled_pcf.loc[quarter_index, :]
    scaled_da = scaled_da.loc[quarter_index, :]
    scaled_curr = scaled_curr.loc[quarter_index, :]

    BaseReturn9 = BaseReturn9.append(df_ret)

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

    Base_results = []
```

```
    Base_weights = []
    Base_SE = []
    init_points = list(BaseCoef9.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_base,
            init_points,
            method="BFGS",
            args=(
                wb,
                nt,
                scaled_ret.iloc[0 : i, :],
                scaled_mktcap.iloc[0 : i, :],
                scaled_bm.iloc[0 : i, :],
                scaled_roa.iloc[0 : i, :],
                scaled_roe.iloc[0 : i, :],
                scaled_accrual.iloc[0 : i, :],
                scaled_eqinv.iloc[0 : i, :],
                scaled_atturn.iloc[0 : i, :],
                scaled_cfm.iloc[0 : i, :],
                scaled_curr.iloc[0 : i, :],
                scaled_da.iloc[0 : i, :],
                scaled_pcf.iloc[0 : i, :],
                rr,
            ),
        )
        print("The {} window for year {}".format(i+1, year))
        print("The value:", opt["x"])
        Base_results.append(list(opt["x"]))

        Base_SE.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
        weight = wb + nt * (
            + opt["x"][0] * scaled_mktcap.iloc[i, :]
            + opt["x"][1] * scaled_bm.iloc[i, :]
            + opt["x"][2] * scaled_roa.iloc[i, :]
            + opt["x"][3] * scaled_roe.iloc[i, :]
            + opt["x"][4] * scaled_accrual.iloc[i, :]
            + opt["x"][5] * scaled_eqinv.iloc[i, :]
            + opt["x"][6] * scaled_atturn.iloc[i, :]
            + opt["x"][7] * scaled_cfm.iloc[i, :]
            + opt["x"][8] * scaled_curr.iloc[i, :]
            + opt["x"][9] * scaled_da.iloc[i, :]
            + opt["x"][10] * scaled_pcf.iloc[i, :]
        )
        print(weight)
        Base_weights.append(weight)

    BaseWeights9 = BaseWeights9.append(short_sell_constraints(pd.DataFrame(Base_weights)))
    BaseCoef9 = BaseCoef9.append(pd.DataFrame(Base_results))
    BaseSE9 = BaseSE9.append(pd.DataFrame(Base_SE))



### Risk Aversion PCA Cases
```

```
# pc = 2

rr = [1,3,7,9]
year_list = range(1970, 2021)

for r in rr:

    rr = r
    PCA2Weights = pd.DataFrame()
    PCA2Return = pd.DataFrame()
    PCA2SE = pd.DataFrame()

    PCA2Coef = pd.DataFrame(np.zeros(2)).T

    for year in year_list:

        df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv'
    ).set_index('date')

        scaled_data_folder = './new standardized5/'
        scaled_PCA2_folder = './PCA Case/2 npc/'

        scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + '
    scaled ret' + str(year) + '.csv').set_index('date')
        scaled_component1 = pd.read_csv(scaled_PCA2_folder + str(
    year) + '/component 1.csv').set_index('date')
        scaled_component2 = pd.read_csv(scaled_PCA2_folder + str(
    year) + '/component 2.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, :]
        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

        PCA2Return = PCA2Return.append(df_ret)

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

        PCA2_results = []
        PCA2_weights = []
        PCA2_se = []
        init_points = list(PCA2Coef.iloc[-1,:].values)

        for i in range(4):
            opt = scipy.optimize.minimize(
                PPS_pca_2,
                init_points,
                method="BFGS",
                args=(
```

```
                    wb,
                    nt,
                    scaled_ret.iloc[0 : i, :],
                    scaled_component1.iloc[0 : i, :],
                    scaled_component2.iloc[0 : i, :],
                    rr,
                ),
            )
#                 print("The {} window for year {}".format(i+1, year))
#                 print("The value:", opt["x"])
            PCA2_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
            PCA2_results.append(list(opt["x"]))
            weight = wb + nt * (
                opt["x"][0] * scaled_component1.iloc[i, :]
                + opt["x"][1] * scaled_component2.iloc[i, :]
                )
#                 print(weight)
            PCA2_weights.append(weight)

        PCA2Weights = PCA2Weights.append(short_sell_constraints(pd.DataFrame(PCA2_weights)))
        PCA2Coef = PCA2Coef.append(pd.DataFrame(PCA2_results))
        PCA2SE = PCA2SE.append(PCA2_se)

    print('-------------  RISK AVERSION = {} -----------------'.format(r))
    print('Max weight = {}; Min weight = {}; Average weight = {}'.format(PCA2Weights.max().max(),
         PCA2Weights.min().min(),
         PCA2Weights.mean().mean()))
    print('Coef 1 = {}, Coef 2 = {}'.format(PCA2Coef.mean()[0], PCA2Coef.mean()[1]))
    print('SE 1 = {}, SE 2 = {}'.format(PCA2SE.mean()[0], PCA2SE.mean()[1]))
    print('Average Return = {}'.format(cumulative_return(PCA2Return, PCA2Weights).mean()*0.16))
    print('Standard deviation = {}'.format(np.nansum(PCA2Return.values[1:]*PCA2Weights.values[:-1], axis=1).std()*(12**0.5)))
    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))))

# pc = 3

rr = [1,3,7,9]
year_list = range(1970, 2021)

for r in rr:

    PCA3Weights = pd.DataFrame()
    PCA3Return = pd.DataFrame()
    PCA3SE = pd.DataFrame()

```

```
    PCA3Coef = pd.DataFrame(np.zeros(3)).T
    rr = r

    for year in year_list:

        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

        PCA3Return = PCA3Return.append(df_ret)

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

        PCA3_results = []
        PCA3_weights = []
        PCA3_se = []
        init_points = list(PCA3Coef.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 * (
                opt["x"][0] * scaled_component1.iloc[i, :]
                + opt["x"][1] * scaled_component2.iloc[i, :]
                + opt["x"][2] * scaled_component3.iloc[i, :]
            )
#             print(weight)
            PCA3_weights.append(weight)

        PCA3Weights = PCA3Weights.append(short_sell_constraints(pd.
    DataFrame(PCA3_weights)))
        PCA3Coef = PCA3Coef.append(pd.DataFrame(PCA3_results))
        PCA3SE = PCA3SE.append(pd.DataFrame(PCA3_se))

    print('-------------  RISK AVERSION = {} -----------------'.
    format(r))
    print('Max weight = {}; Min weight = {}; Average weight = {}'.
    format(PCA3Weights.max().max(),

          PCA3Weights.min().min(),

          PCA3Weights.mean().mean()))
    print('Coef 1 = {}, Coef 2 = {}, Coef 3 = {}'.format(PCA3Coef.
    mean()[0], PCA3Coef.mean()[1], PCA3Coef.mean()[2]))
    print('SE 1 = {}, SE 2 = {}, SE 3 = {}'.format(PCA3SE.mean()
    [0], PCA3SE.mean()[1], PCA3SE.mean()[2]))
    print('Average Return = {}'.format(cumulative_return(PCA3Return
    , PCA3Weights).mean()*0.16))
    print('Standard deviation = {}'.format(np.nansum(PCA3Return.
    values[1:]*PCA3Weights.values[:-1], axis=1).std()*(12**0.5)))
    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))))

# pc = 4

rr = [1,3,7,9]

for r in rr:

    rr = r
    PCA4Weights = pd.DataFrame()
    PCA4Return = pd.DataFrame()
    PCA4SE = pd.DataFrame()

    PCA4Coef = pd.DataFrame(np.zeros(4)).T

    for year in year_list:

```

```
    df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').set_index('date')

    scaled_data_folder = './new standardized5/'
    scaled_PCA4_folder = './PCA Case/4 npc/'

    scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled ret' + str(year) + '.csv').set_index('date')
    scaled_component1 = pd.read_csv(scaled_PCA4_folder + str(year) + '/component 1.csv').set_index('date')
    scaled_component2 = pd.read_csv(scaled_PCA4_folder + str(year) + '/component 2.csv').set_index('date')
    scaled_component3 = pd.read_csv(scaled_PCA4_folder + str(year) + '/component 3.csv').set_index('date')
    scaled_component4 = pd.read_csv(scaled_PCA4_folder + str(year) + '/component 4.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, :]
    scaled_component4 = scaled_component4.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
    scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T

    PCA4Return = PCA4Return.append(df_ret)

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

    PCA4_results = []
    PCA4_weights = []
    PCA4_se = []
    init_points = list(PCA4Coef.iloc[-1,:].values)

    for i in range(4):
        opt = scipy.optimize.minimize(
            PPS_pca_4,
            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, :],
                scaled_component4.iloc[0 : i, :],
```

```
                    rr,
                ),
            )
#               print("The {} window for year {}".format(i+1, year))
#               print("The value:", opt["x"])
            PCA4_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))

            PCA4_results.append(list(opt["x"]))
            weight = wb + nt * (
                opt["x"][0] * scaled_component1.iloc[i, :]
                + opt["x"][1] * scaled_component2.iloc[i, :]
                + opt["x"][2] * scaled_component3.iloc[i, :]
                + opt["x"][3] * scaled_component4.iloc[i, :]
            )
#               print(weight)
            PCA4_weights.append(weight)

        PCA4Weights = PCA4Weights.append(short_sell_constraints(pd.DataFrame(PCA4_weights)))
        PCA4Coef = PCA4Coef.append(pd.DataFrame(PCA4_results))
        PCA4SE = PCA4SE.append(pd.DataFrame(PCA4_se))

    print('-------------  RISK AVERSION = {} -----------------'.format(r))
    print('Max weight = {}; Min weight = {}; Average weight = {}'.format(PCA4Weights.max().max(),

          PCA4Weights.min().min(),

          PCA4Weights.mean().mean()))
    print('Coef 1 = {}, Coef 2 = {}, Coef 3 = {}, Coef 4 = {}'.format(PCA4Coef.mean()[0],

       PCA4Coef.mean()[1],

       PCA4Coef.mean()[2],

       PCA4Coef.mean()[3]))
    print('SE 1 = {}, SE 2 = {}, SE 3 = {}, SE 4 = {}'.format(PCA4SE.mean()[0],
                                                                PCA4SE.mean()[1],
                                                                PCA4SE.mean()[2],
                                                                PCA4SE.mean()[3]))
    print('Average Return = {}'.format(cumulative_return(PCA4Return, PCA4Weights).mean()*0.16))
    print('Standard deviation = {}'.format(np.nansum(PCA4Return.values[1:]*PCA4Weights.values[:-1], axis=1).std()*(12**0.5)))
    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))))

# pc = 5
rr = [1,3,7,9]
```

```
for r in rr:

    PCA5Weights = pd.DataFrame()
    PCA5Return = pd.DataFrame()
    PCA5SE = pd.DataFrame()

    PCA5Coef = pd.DataFrame(np.zeros(5)).T
    rr = r

    for year in year_list:

        df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv'
    ).set_index('date')

        scaled_data_folder = './new standardized5/'
        scaled_PCA5_folder = './PCA Case/5 npc/'

        scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + '
    scaled ret' + str(year) + '.csv').set_index('date')
        scaled_component1 = pd.read_csv(scaled_PCA5_folder + str(
    year) + '/component 1.csv').set_index('date')
        scaled_component2 = pd.read_csv(scaled_PCA5_folder + str(
    year) + '/component 2.csv').set_index('date')
        scaled_component3 = pd.read_csv(scaled_PCA5_folder + str(
    year) + '/component 3.csv').set_index('date')
        scaled_component4 = pd.read_csv(scaled_PCA5_folder + str(
    year) + '/component 4.csv').set_index('date')
        scaled_component5 = pd.read_csv(scaled_PCA5_folder + str(
    year) + '/component 5.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, :]
        scaled_component4 = scaled_component4.loc[quarter_index, :]
        scaled_component5 = scaled_component5.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
        scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T
    )).T
        scaled_component5 =  pd.DataFrame(Scale(scaled_component5.T
    )).T

        PCA5Return = PCA5Return.append(df_ret)

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

        PCA5_results = []
```

```
        PCA5_weights = []
        PCA5_se = []
        init_points = list(PCA5Coef.iloc[-1,:].values)

        for i in range(4):
            opt = scipy.optimize.minimize(
                PPS_pca_5,
                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, :],
                    scaled_component4.iloc[0 : i, :],
                    scaled_component5.iloc[0 : i, :],
                    rr,
                ),
            )
#             print("The {} window for year {}".format(i+1, year))
#             print("The value:", opt["x"])
            PCA5_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))
            PCA5_results.append(list(opt["x"]))
            weight = wb + nt * (
                opt["x"][0] * scaled_component1.iloc[i, :]
                + opt["x"][1] * scaled_component2.iloc[i, :]
                + opt["x"][2] * scaled_component3.iloc[i, :]
                + opt["x"][3] * scaled_component4.iloc[i, :]
                + opt["x"][4] * scaled_component5.iloc[i, :]
            )
#             print(weight)
            PCA5_weights.append(weight)

        PCA5Weights = PCA5Weights.append(short_sell_constraints(pd.DataFrame(PCA5_weights)))
        PCA5Coef = PCA5Coef.append(pd.DataFrame(PCA5_results))
        PCA5SE = PCA5SE.append(pd.DataFrame(PCA5_se))
    print('-------------  RISK AVERSION = {} -----------------'.format(r))
    print('Max weight = {}; Min weight = {}; Average weight = {}'.format(PCA5Weights.max().max(),
         PCA5Weights.min().min(),
         PCA5Weights.mean().mean()))
    print('Coef 1 = {}, Coef 2 = {}, Coef 3 = {}, Coef 4 = {}, Coef 5 = {}'.format(PCA5Coef.mean()[0],
      PCA5Coef.mean()[1],
      PCA5Coef.mean()[2],
      PCA5Coef.mean()[3],

```

```
    PCA5Coef.mean()[4]))
    print('SE 1 = {}, SE 2 = {}, SE 3 = {}, SE 4 = {}, SE 5 = {}'.
    format(PCA5SE.mean()[0],
                                                                PCA5SE.mean()
    [1],
                                                                PCA5SE.mean()
    [2],
                                                                PCA5SE.mean()
    [3],
                                                                PCA5SE.mean()
    [4]))
    print('Average Return = {}'.format(cumulative_return(PCA5Return
    , PCA5Weights).mean()*0.16))
    print('Standard deviation = {}'.format(np.nansum(PCA5Return.
    values[1:]*PCA5Weights.values[:-1], axis=1).std()*(12**0.5)))
    print('Sharpe Ratio = {}'.format(((cumulative_return(PCA5Return
    , PCA5Weights).mean()*0.16)-0.012)/(np.nansum(PCA5Return.values
    [1:]*PCA5Weights.values[:-1], axis=1).std()*(12**0.5))))

# pc = 6
rr = [1,3,7,9]

for r in rr:
    PCA6Weights = pd.DataFrame()
    PCA6Return = pd.DataFrame()
    PCA6SE = pd.DataFrame()

    PCA6Coef = pd.DataFrame(np.zeros(6)).T
    rr = r

    for year in year_list:

        df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv'
    ).set_index('date')

        scaled_data_folder = './new standardized5/'
        scaled_PCA6_folder = './PCA Case/6 npc/'

        scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + '
    scaled ret' + str(year) + '.csv').set_index('date')
        scaled_component1 = pd.read_csv(scaled_PCA6_folder + str(
    year) + '/component 1.csv').set_index('date')
        scaled_component2 = pd.read_csv(scaled_PCA6_folder + str(
    year) + '/component 2.csv').set_index('date')
        scaled_component3 = pd.read_csv(scaled_PCA6_folder + str(
    year) + '/component 3.csv').set_index('date')
        scaled_component4 = pd.read_csv(scaled_PCA6_folder + str(
    year) + '/component 4.csv').set_index('date')
        scaled_component5 = pd.read_csv(scaled_PCA6_folder + str(
    year) + '/component 5.csv').set_index('date')
        scaled_component6 = pd.read_csv(scaled_PCA6_folder + str(
    year) + '/component 6.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, :]
        scaled_component4 = scaled_component4.loc[quarter_index, :]
        scaled_component5 = scaled_component5.loc[quarter_index, :]
        scaled_component6 = scaled_component6.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
        scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T
        scaled_component5 =  pd.DataFrame(Scale(scaled_component5.T)).T
        scaled_component6 =  pd.DataFrame(Scale(scaled_component6.T)).T

        PCA6Return = PCA6Return.append(df_ret)

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

        PCA6_results = []
        PCA6_weights = []
        PCA6_se = []
        init_points = list(PCA6Coef.iloc[-1,:].values)

        for i in range(4):
            opt = scipy.optimize.minimize(
                PPS_pca_6,
                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, :],
                    scaled_component4.iloc[0 : i, :],
                    scaled_component5.iloc[0 : i, :],
                    scaled_component6.iloc[0 : i, :],
                    rr,
                ),
            )
#             print("The {} window for year {}".format(i+1, year))
#             print("The value:", opt["x"])
            PCA6_results.append(list(opt["x"]))
            PCA6_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))

            weight = wb + nt * (
                opt["x"][0] * scaled_component1.iloc[i, :]
                + opt["x"][1] * scaled_component2.iloc[i, :]
                + opt["x"][2] * scaled_component3.iloc[i, :]
```

```
                + opt["x"][3] * scaled_component4.iloc[i, :]
                + opt["x"][4] * scaled_component5.iloc[i, :]
                + opt["x"][5] * scaled_component6.iloc[i, :]
            )
#               print(weight)
            PCA6_weights.append(weight)

        PCA6Weights = PCA6Weights.append(short_sell_constraints(pd.DataFrame(PCA6_weights)))
        PCA6Coef = PCA6Coef.append(pd.DataFrame(PCA6_results))
        PCA6SE = PCA6SE.append(pd.DataFrame(PCA6_se))

    print('-------------  RISK AVERSION = {} -----------------'.format(r))
    print('Max weight = {}; Min weight = {}; Average weight = {}'.format(PCA6Weights.max().max(),
                                                                        PCA6Weights.min().min(),
                                                                        PCA6Weights.mean().mean()))
    print('Coef 1 = {}, Coef 2 = {}, Coef 3 = {}, Coef 4 = {}, Coef 5 = {}, Coef 6 = {}'.format(PCA6Coef.mean()[0],
                                                                                            PCA6Coef.mean()[1],
                                                                                            PCA6Coef.mean()[2],
                                                                                            PCA6Coef.mean()[3],
                                                                                            PCA6Coef.mean()[4],
                                                                                            PCA6Coef.mean()[5]))
    print('SE 1 = {}, SE 2 = {}, SE 3 = {}, SE 4 = {}, SE 5 = {}, SE 6 = {}'.format(PCA6SE.mean()[0],
                                                                                  PCA6SE.mean()[1],
                                                                                  PCA6SE.mean()[2],
                                                                                  PCA6SE.mean()[3],
                                                                                  PCA6SE.mean()[4],
                                                                                  PCA6SE.mean()[5]))
    print('Average Return = {}'.format(cumulative_return(PCA6Return, PCA5Weights).mean()*0.16))
    print('Standard deviation = {}'.format(np.nansum(PCA6Return.values[1:]*PCA6Weights.values[:-1], axis=1).std()*(12**0.5)))
    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))))

# pc = 7

rr = [1,3,7,9]

```

```
for r in rr:

    PCA7Weights = pd.DataFrame()
    PCA7Return = pd.DataFrame()
    PCA7SE = pd.DataFrame()

    PCA7Coef = pd.DataFrame(np.zeros(7)).T
    rr = r

    for year in year_list:

        df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv'
    ).set_index('date')

        scaled_data_folder = './new standardized5/'
        scaled_PCA7_folder = './PCA Case/7 npc/'

        scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + '
    scaled ret' + str(year) + '.csv').set_index('date')
        scaled_component1 = pd.read_csv(scaled_PCA7_folder + str(
    year) + '/component 1.csv').set_index('date')
        scaled_component2 = pd.read_csv(scaled_PCA7_folder + str(
    year) + '/component 2.csv').set_index('date')
        scaled_component3 = pd.read_csv(scaled_PCA7_folder + str(
    year) + '/component 3.csv').set_index('date')
        scaled_component4 = pd.read_csv(scaled_PCA7_folder + str(
    year) + '/component 4.csv').set_index('date')
        scaled_component5 = pd.read_csv(scaled_PCA7_folder + str(
    year) + '/component 5.csv').set_index('date')
        scaled_component6 = pd.read_csv(scaled_PCA7_folder + str(
    year) + '/component 6.csv').set_index('date')
        scaled_component7 = pd.read_csv(scaled_PCA7_folder + str(
    year) + '/component 7.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, :]
        scaled_component4 = scaled_component4.loc[quarter_index, :]
        scaled_component5 = scaled_component5.loc[quarter_index, :]
        scaled_component6 = scaled_component6.loc[quarter_index, :]
        scaled_component7 = scaled_component7.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
        scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T
    )).T
        scaled_component5 =  pd.DataFrame(Scale(scaled_component5.T
    )).T
```

```
        scaled_component6 =  pd.DataFrame(Scale(scaled_component6.T)).T
        scaled_component7 =  pd.DataFrame(Scale(scaled_component7.T)).T

        PCA7Return = PCA7Return.append(df_ret)

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

        PCA7_results = []
        PCA7_weights = []
        PCA7_se = []
        init_points = list(PCA7Coef.iloc[-1,:].values)

        for i in range(4):
            opt = scipy.optimize.minimize(
                PPS_pca_7,
                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, :],
                    scaled_component4.iloc[0 : i, :],
                    scaled_component5.iloc[0 : i, :],
                    scaled_component6.iloc[0 : i, :],
                    scaled_component7.iloc[0 : i, :],
                    rr,
                ),
            )
#             print("The {} window for year {}".format(i+1, year))
#             print("The value:", opt["x"])
            PCA7_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))

            PCA7_results.append(list(opt["x"]))
            weight = wb + nt * (
                opt["x"][0] * scaled_component1.iloc[i, :]
                + opt["x"][1] * scaled_component2.iloc[i, :]
                + opt["x"][2] * scaled_component3.iloc[i, :]
                + opt["x"][3] * scaled_component4.iloc[i, :]
                + opt["x"][4] * scaled_component5.iloc[i, :]
                + opt["x"][5] * scaled_component6.iloc[i, :]
                + opt["x"][6] * scaled_component7.iloc[i, :]
            )
#             print(weight)
            PCA7_weights.append(weight)

        PCA7Weights = PCA7Weights.append(short_sell_constraints(pd.DataFrame(PCA7_weights)))
        PCA7Coef = PCA7Coef.append(pd.DataFrame(PCA7_results))
        PCA7SE = PCA7SE.append(pd.DataFrame(PCA7_se))
    print('-------------  RISK AVERSION = {} -----------------'.format(r))
```

```
    print('Max weight = {}; Min weight = {}; Average weight = {}'.
   format(PCA7Weights.max().max(),

          PCA7Weights.min().min(),

          PCA7Weights.mean().mean()))
    print('Coef 1 = {}, Coef 2 = {}, Coef 3 = {}, Coef 4 = {}, Coef
    5 = {}, Coef 6 = {}, Coef 7 = {}'.format(PCA7Coef.mean()[0],

      PCA7Coef.mean()[1],

      PCA7Coef.mean()[2],

      PCA7Coef.mean()[3],

      PCA7Coef.mean()[4],

      PCA7Coef.mean()[5],

      PCA7Coef.mean()[6]))
    print('SE 1 = {}, SE 2 = {}, SE 3 = {}, SE 4 = {}, SE 5 = {},
   SE 6 = {}, SE 7 = {}'.format(PCA7SE.mean()[0],
                                                    PCA7SE.mean()
   [1],
                                                    PCA7SE.mean()
   [2],
                                                    PCA7SE.mean()
   [3],
                                                    PCA7SE.mean()
   [4],
                                                    PCA7SE.mean()
   [5],
                                                    PCA7SE.mean()
   [6]))
    print('Average Return = {}'.format(cumulative_return(PCA7Return
   , PCA7Weights).mean()*0.16))
    print('Standard deviation = {}'.format(np.nansum(PCA7Return.
   values[1:]*PCA7Weights.values[:-1], axis=1).std()*(12**0.5)))
    print('Sharpe Ratio = {}'.format(((cumulative_return(PCA7Return
   , PCA7Weights).mean()*0.16)-0.012)/(np.nansum(PCA7Return.values
   [1:]*PCA7Weights.values[:-1], axis=1).std()*(12**0.5))))

# pc = 8

rr = [1,3,7,9]

for r in rr:
    rr = r
    PCA8Weights = pd.DataFrame()
    PCA8Return = pd.DataFrame()
    PCA8SE = pd.DataFrame()

    PCA8Coef = pd.DataFrame(np.zeros(8)).T


    for year in year_list:
```

```

        df_ret = pd.read_csv('./new char5/ret/ret'+str(year)+'.csv').set_index('date')

        scaled_data_folder = './new standardized5/'
        scaled_PCA8_folder = './PCA Case/8 mpc/'

        scaled_ret = pd.read_csv(scaled_data_folder + 'ret/' + 'scaled ret' + str(year) + '.csv').set_index('date')
        scaled_component1 = pd.read_csv(scaled_PCA8_folder + str(year) + '/component 1.csv').set_index('date')
        scaled_component2 = pd.read_csv(scaled_PCA8_folder + str(year) + '/component 2.csv').set_index('date')
        scaled_component3 = pd.read_csv(scaled_PCA8_folder + str(year) + '/component 3.csv').set_index('date')
        scaled_component4 = pd.read_csv(scaled_PCA8_folder + str(year) + '/component 4.csv').set_index('date')
        scaled_component5 = pd.read_csv(scaled_PCA8_folder + str(year) + '/component 5.csv').set_index('date')
        scaled_component6 = pd.read_csv(scaled_PCA8_folder + str(year) + '/component 6.csv').set_index('date')
        scaled_component7 = pd.read_csv(scaled_PCA8_folder + str(year) + '/component 7.csv').set_index('date')
        scaled_component8 = pd.read_csv(scaled_PCA8_folder + str(year) + '/component 8.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, :]
        scaled_component4 = scaled_component4.loc[quarter_index, :]
        scaled_component5 = scaled_component5.loc[quarter_index, :]
        scaled_component6 = scaled_component6.loc[quarter_index, :]
        scaled_component7 = scaled_component7.loc[quarter_index, :]
        scaled_component8 = scaled_component8.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
        scaled_component4 =  pd.DataFrame(Scale(scaled_component4.T)).T
        scaled_component5 =  pd.DataFrame(Scale(scaled_component5.T)).T
        scaled_component6 =  pd.DataFrame(Scale(scaled_component6.T)).T
        scaled_component7 =  pd.DataFrame(Scale(scaled_component7.T)).T
        scaled_component8 =  pd.DataFrame(Scale(scaled_component8.T)).T

        PCA8Return = PCA8Return.append(df_ret)
```

```python

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

        PCA8_results = []
        PCA8_weights = []
        PCA8_se = []
        init_points = list(PCA8Coef.iloc[-1,:].values)

        for i in range(4):
            opt = scipy.optimize.minimize(
                PPS_pca_8,
                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, :],
                    scaled_component4.iloc[0 : i, :],
                    scaled_component5.iloc[0 : i, :],
                    scaled_component6.iloc[0 : i, :],
                    scaled_component7.iloc[0 : i, :],
                    scaled_component8.iloc[0 : i, :],
                    rr,
                ),
            )
#             print("The {} window for year {}".format(i+1, year))
#             print("The value:", opt["x"])
            PCA8_results.append(list(opt["x"]))
            PCA8_se.append(list(((np.diag(opt.hess_inv))*nt)**0.5))

            weight = wb + nt * (
                opt["x"][0] * scaled_component1.iloc[i, :]
                + opt["x"][1] * scaled_component2.iloc[i, :]
                + opt["x"][2] * scaled_component3.iloc[i, :]
                + opt["x"][3] * scaled_component4.iloc[i, :]
                + opt["x"][4] * scaled_component5.iloc[i, :]
                + opt["x"][5] * scaled_component6.iloc[i, :]
                + opt["x"][6] * scaled_component7.iloc[i, :]
                + opt["x"][7] * scaled_component8.iloc[i, :]
            )
#             print(weight)
            PCA8_weights.append(weight)

        PCA8Weights = PCA8Weights.append(short_sell_constraints(pd.
    DataFrame(PCA8_weights)))
        PCA8Coef = PCA8Coef.append(pd.DataFrame(PCA8_results))
        PCA8SE = PCA8SE.append(pd.DataFrame(PCA8_se))

     print('-------------  RISK AVERSION = {} -----------------'.
    format(r))
     print('Max weight = {}; Min weight = {}; Average weight = {}'.
    format(PCA8Weights.max().max(),

```

```
        PCA8Weights.min().min(),

        PCA8Weights.mean().mean()))
print('Coef 1 = {}, Coef 2 = {}, Coef 3 = {}, Coef 4 = {}, Coef 5 = {}, Coef 6 = {}, Coef 7 = {}, Coef 8 ={}'.format(PCA8Coef.mean()[0],

    PCA8Coef.mean()[1],

    PCA8Coef.mean()[2],

    PCA8Coef.mean()[3],

    PCA8Coef.mean()[4],

    PCA8Coef.mean()[5],

    PCA8Coef.mean()[6],

    PCA8Coef.mean()[7]))
print('SE 1 = {}, SE 2 = {}, SE 3 = {}, SE 4 = {}, SE 5 = {}, SE 6 = {}, SE 7 = {}, SE 8 = {}'.format(PCA8SE.mean()[0],
                                                                PCA8SE.mean()[1],
                                                                PCA8SE.mean()[2],
                                                                PCA8SE.mean()[3],
                                                                PCA8SE.mean()[4],
                                                                PCA8SE.mean()[5],
                                                                PCA8SE.mean()[6],
                                                                PCA8SE.mean()[7]))
print('Average Return = {}'.format(cumulative_return(PCA8Return, PCA8Weights).mean()*0.16))
print('Standard deviation = {}'.format(np.nansum(PCA8Return.values[1:]*PCA8Weights.values[:-1], axis=1).std()*(12**0.5)))
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))))
```