Factor Decay

Johanan Anton Pranesh, Gregory Radu Colonescu, Michael Miller, Roman Mitchell

Introduction **2**
Background Knowledge **3**
Factor Decay & Causes **4**
Data for PCA and factors **6**
PCA **6**
Step 1: Standardization 7
Step 2: Covariance Matrix Computation 8
Step 3: Calculate the eigenvectors and values of the PCA 8
Step 4: Feature Vector 8
Step 5: Recasting along PCA data 8
Feature Engineering **10**
Goals of Experiment **11**
Method **11**
Simplified Method Explanation: 11
In-Depth Explanation: 12
Optimization method **13**
Mathematical formulation explanation 13
Results **14**
Scenarios **19**
Scenario 1 19
Scenario 2 20
Scenario 3 20
Rolling window 20
Conclusion **20**
References **21**
Appendix **21**

Introduction

Factors play a crucial role in asset pricing models, providing insights into risk and return relationships in financial markets. However, factors are not static; they can lose their predictive power over time, a phenomenon known as factor decay. This paper explores the concept of factor decay, its causes, implications, and potential mitigation strategies, along with an analysis using PCA on how to optimize a portfolio using these factors. You can use past data to predict the weight of each of these factors and how correlated it is to a stock. Due to decay, there is a need for diversification in weights of these factors when creating strategies. We will use Python's library Pulp to display a version of optimizing the weights.

Background Knowledge

Some background is needed to establish a baseline understanding of factors in this context. Factors are common attributes corresponding to underlying assets that are thought to have a significant relationship with the assets' risk and returns. Models aim to explain the variation in returns and identify which factors contribute the most to the performance of a portfolio. Therefore, returns of a portfolio or asset can be modeled as a linear combination of factor exposures multiplied by their respective factor risk premiums. Risk corresponds to the variation in asset returns.

Factor models also allow for ex-ante forecasting of asset risk and returns, given factor loadings have been found. However, as discussed in this paper, these factor loadings may change and lose their predictive power.

Examples of common factors used in investment management include market risk (captured by the market return), size (small-cap vs large-cap stocks), value ("high" vs "low" value stocks), and momentum (recent performance). The most well-known factor models are CAPM, Fama-French 3 Factor, Fama-French 5 Factor, and Carhart. CAPM is the original factor model and is still used as a starting point for a general understanding of how the stock market works.

The CAPM equation is $r_i = r_f + \beta(r_m - r_f)$ in idealized scenarios. An epsilon term corresponding to the error term in regression equations is added along with an alpha term to account for excess return that cannot be explained by the chosen factors to yield $r_i - r_f = \alpha_i + \beta(r_m - r_f) + \epsilon_i$. This market factor helps us understand how risky a stock is compared to the market. Fama-French 3 introduces a size (small minus big) and value (high minus low) factor into the CAPM equation, $r_i = r_f + \beta_M(r_m - r_f) + \beta_S(SMB) + B_V(HML)$.

These 3 factors are all used to understand risk. Fama-French 5 introduces additional factors, profitability (robust minus weak) and investment (conservative minus aggressive): $r_i = r_f + \beta_M(r_m - r_f) + \beta_S(SMB) + B_V(HML) + \beta_P(RMW) + \beta_I(CMA)$.

The Carhart 4 model implements a momentum factor because momentum was demonstrated to be predictive of stock returns:

$$r_i = r_f + \beta_M(r_m - r_f) + \beta_S(SMB) + B_V(HML) + \beta_{Mom}(MOM).$$

Momentum is determined by stocks reaching 12-month highs/lows.

Numerous challenges exist to the successful use of factor models. Some of these challenges are data mining bias, factor instability, and model risk. Data mining bias refers to selecting factors based solely on historical performance and can lead to overfitting and poor out-of-sample performance. Factor instability refers to factors losing their predictive power over time due to changes in market dynamics or economic performance. Model risk refers to the fact that factor models may not capture all sources of risk, leading to inaccurate risk assessments and investment decisions. Many more factors and factor models have been proposed.

There are also overarching influences that don't neatly fit in factor models. Two big ones are geography and regime change. Factors may behave differently in different regions due to localized economic and cultural differences. For example, in "Momentum in Japan: The Exception That Proves the Rule," Asness notes research suggesting Japan represents an exception to the generally observed positive momentum effect seen in other major markets and that the anomaly could be attributed to several aspects unique to Japan, such as differences in investor behavior, the influence of keiretsu (conglomerates), or specific market mechanisms that do not reward momentum-based strategies as they do elsewhere. Economic regimes may also impact the effectiveness of factors. As a result, some factors may appear cyclical, or there may be paradigm shifts permanently altering factor performance.

There's many more factors in the market but to optimize the model there should be a limit on the number of factors used. There needs to be an emphasis on the most effective factors across the market historically.

Factor Decay & Causes

Factor decay occurs when a factor ceases to have predictive power, posing a risk-management problem. For example, if employee growth is a factor predictive of equity returns and suddenly stops being predictive, this is factor decay. There are two primary types of factor decay: post-publication/arbitrage and data mining.

Post-publication/Arbitrage occurs when, after publication, a factor loses predictive power because parties in the market start implementing this factor into their trading strategies. According to Falck, Rej, and Thesmar, "The main predictor of decay is the date of publication: every year, the post-publication Sharpe decay of a newly published strategy increases by approximately 5 percentage points." … 6 years of this leads to "We find that the date of publication alone predicts 30% of the variance of decays" (*When do systematic strategies decay?*; Falck, Rej, and Thesmar).

Data mining caused factor decay also becomes evident post-publication, but more immediately. It occurs when researchers find illusory patterns in data. There are 3 primary indicators that data mining is culpable. The first is signal complexity, "as measured by the number of operations involved in computing the predicting variable, predicts stronger post-publication performance decay." "This is consistent with researchers mining for more complex relationships to improve the in-sample backtest" (*When do systematic strategies decay?*; Falck, Rej, and Thesmar).

Next is vulnerability. Falck, Rej, and Thesmar characterize it as "'sensitivity to removing a small subset of stocks' … We construct this measure by computing the range of Sharpe ratios obtained in-sample by randomly removing 10% of the pool over 100 draws. When the range is large, the in-sample performance of the factor is vulnerable to a small number of stocks." This would mean that most of the performance is coming from very few stocks. One way that this would reveal its place is that the place of trading would lead to these factors being nonexistent or place-specific, as Japan is seen as an anomaly for the momentum factor. Here, altering the trading universe could help identify if there's a real pattern or if it's just illusory.

The third is sensitivity. According to Falck, Rej, and Thesmar, “For each anomaly, we compute the performance loss after removing the most influential 0.1% observations. Interestingly, we show that the performance drop is very large on average. Put differently, a big fraction of in-sample returns in classical strategies depends on the presence of a small number of jumps—a feature arising from the non-normality of stock returns.”

All of these could have legitimate reasons why they were done. Perhaps the trading strategy is reliant on trading on certain quarterly results release dates, or there is no easy way to identify the factor. The important part is consistency: Being able to experiment with the model out of sample is incredibly important, and if it works out of sample as well then it’s potentially a good model.

There are a few additional important things to note regarding factor decay. The first is that convergence of factors can exist. like there has been with value. Given a starting portfolio, the average within each group (e.g., high value and low value) will start to converge. There are also different regimes, as mentioned earlier. There have been different time periods where some factors lose their predictive power. Fama-French went through a glut in the 90s and post-2008 recession. This fluctuation over time is seen as related to market cycles. There is also evidence that some factors “enable” other factors, like momentum, which is argued here as multiplying other factors (*Factor Momentum and the Momentum Factor*, EHSANI, LINNAINMAA).

Data for PCA and factors

In this study, the data collection process commenced with accessing the Wharton Research Data Services (WRDS) JKPFactors database, renowned for its comprehensive coverage of financial and market-related data. Specifically, we retrieved datasets encompassing over 25 thematic factors, each offering insights into distinct market dynamics and trends. Additionally, we obtained data pertaining to more than 100 fundamental accounting variables (FAVTOD), which further enriched our dataset. Through meticulous curation, we merged these diverse datasets, comprising a total of 150+ features, to facilitate subsequent feature engineering and analysis.

The process of data collection was critical in laying the foundation for our research objectives. By harnessing the extensive resources available through the WRDS platform, we ensured the inclusion of a broad spectrum of factors and variables relevant to our study. This comprehensive approach not only enhanced the richness of our dataset but also provided a robust basis for subsequent analyses, such as principal component analysis (PCA) and feature engineering. In essence, our methodology for data collection reflects a commitment to leveraging high-quality, comprehensive datasets to explore and uncover underlying patterns and insights in financial markets.

PCA

Furthermore, feature selection played a pivotal role in refining the dataset to include only the most relevant and informative variables. Features exhibiting high similarity to other variables, such as redundant or highly correlated attributes, were identified and removed to mitigate redundancy and multicollinearity issues. This step helped streamline the dataset and improve model performance by focusing on the most discriminative and influential features. Following these preprocessing steps, the dataset was meticulously prepared for principal component analysis (PCA), a dimensionality reduction technique aimed at capturing the underlying structure and patterns in the data while minimizing information loss. By executing these meticulously crafted feature engineering procedures, the study laid a solid foundation for subsequent modeling endeavors, ensuring that the predictive models derived from the data were robust, interpretable, and well-suited for the task at hand.

Principal Component Analysis (PCA) can help confirm factor efficacy and importance as long as they are linear. It can also help weed out datamined factors when dealing with out-of-sample (OOS) data. PCA helps reduce the number of dimensions in large datasets to principal components that retain most of the original information according to the size of their variances. It transforms potentially correlated variables into a smaller set called principal components. PCA is very effective for visualizing and exploring high-dimensional datasets or data with many features, as it can easily identify trends, patterns, or outliers. This reduces model complexity as the addition of each new feature negatively impacts model performance, which is also commonly referred to as the “curse of dimensionality.” PCA minimizes or eliminates common issues such as multicollinearity and overfitting by projecting a high-dimensional dataset into a smaller feature space. Multicollinearity occurs when two or more independent variables are highly correlated, which can be problematic for causal modeling.

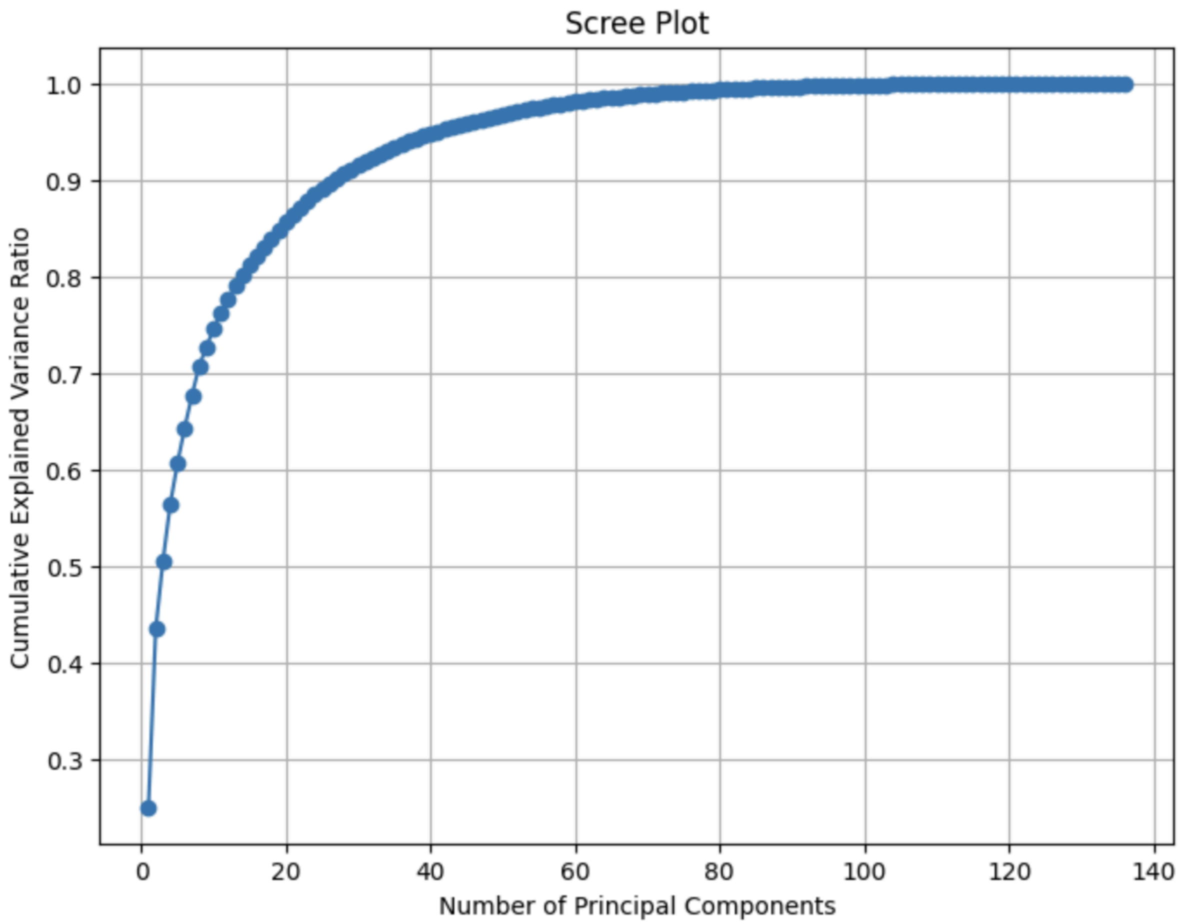
Steps:

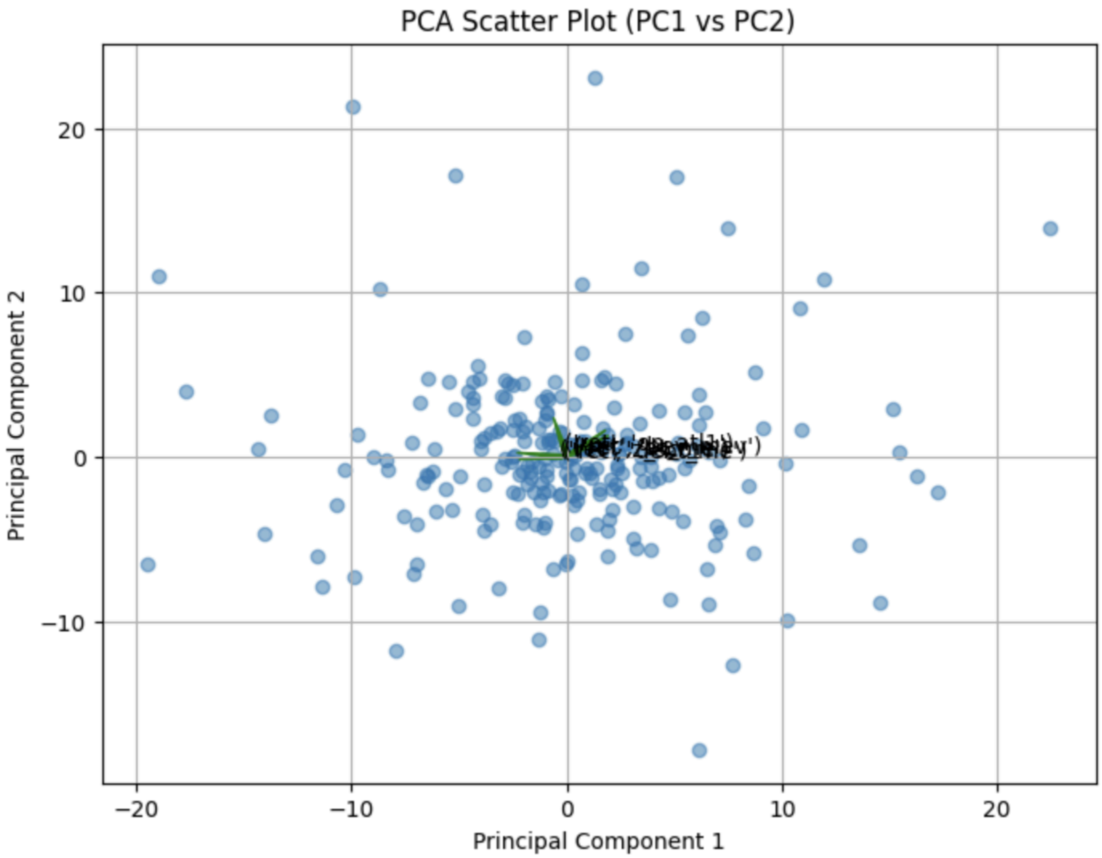
1. Standardization: This step ensures that all variables contribute equally by transforming them to have a mean of zero and a standard deviation of one. In finance, where variables often have different scales, standardization enables fair comparison and analysis across different financial instruments or factors.
2. Covariance Matrix Computation: The covariance matrix summarizes the relationships between pairs of variables in the dataset. It provides insights into the direction and strength of linear relationships between variables. In finance, understanding the covariance structure is essential for analyzing co-movements of asset returns or factor exposures, especially in portfolio management and risk assessment.
3. Calculation of Eigenvectors and Eigenvalues: Eigenvectors and eigenvalues are derived from the covariance matrix. Eigenvectors represent the directions along which the data varies the most, while eigenvalues quantify the variance explained by each principal component. These components help identify the dominant sources of variation in asset returns or factor exposures, allowing for dimensionality reduction and simplification of complex datasets.
4. Feature Vector Generation: The feature vector consists of the eigenvectors corresponding to the highest eigenvalues. It represents the principal components of the dataset and captures the most significant sources of variation. In finance, the feature vector provides a reduced-dimensional representation of asset returns or factor exposures, facilitating portfolio optimization and risk management strategies.
5. Recasting along PCA Data: The original data is projected onto the principal components to obtain the transformed dataset. This transformation enables data visualization, clustering, or classification based on the reduced-dimensional representation. In finance, recasting along PCA data helps identify portfolio risk factors, perform performance attribution analysis, and detect anomalies or trends not captured by traditional factor models.

In summary, PCA offers a powerful tool for analyzing complex financial datasets and identifying dominant sources of variation or risk factors.

Some notes on Factor Decay

Factor decay in finance may manifest in changes in the covariance structure, declining eigenvalues, or shifts in the composition of principal components, highlighting the dynamic nature of financial markets and the need for adaptive risk management and investment strategies.


Scree Plot
Cumulative Explained Variance Ratio
Number of Principal Components
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100
120
140



Feature Engineering

In this study, feature engineering was employed as a critical step to enhance the predictive power and interpretability of the model by transforming and refining the raw input data. Initially, the process involved merging both factor data and thematic data, leveraging a comprehensive range of variables to capture diverse aspects of the underlying phenomenon. Subsequently, the data underwent transformation using the pivot function, facilitating the restructuring of the dataset to a format conducive for analysis and modeling. This pivot transformation facilitated a more structured representation of the data, enabling better exploration and extraction of meaningful insights.

Addressing missing values was a fundamental aspect of the feature engineering pipeline. For this purpose, a systematic approach was adopted: Features with a significant number of missing values, exceeding a predefined threshold of 30, were deemed unsuitable for analysis and were consequently removed from the dataset. Conversely, features with minimal missing values, typically below 5, underwent imputation using the median value to ensure the preservation of data integrity.

For features falling between these extremes, a sophisticated interpolation technique was applied to fill in the missing values, leveraging the temporal and contextual information inherent in the dataset. This comprehensive strategy ensured that the dataset remained robust and informative, while effectively handling missing data scenarios.

Goals of Experiment

We aim to see if the Fama-French and Carhart models are truly good underlying models by seeing if PCA confirms these factors on larger multifactor models. After this, we employ our optimization model to assign us the weight to the equities in our Universe of stocks (500 in this study) and check its performance under different scenarios. We also want to see if some critics are right that momentum, value, and size cover market risk (is market risk necessary?). We can use this to see if our results follow the Efficient Market Hypothesis (i.e. if finding persistent alpha is possible).

Method

Simplified Method Explanation:

We first establish our universe, which we will do multiple times for different geographies and time periods and gather data.

We will take many factors and associated returns, run PCA, break down results based on components, and figure out the variance ratios. Results with more factors will point towards a lack of efficient market, while results with high ratios of Fama-French factors will point towards Fama-French being a good underlying model.

In-Depth Explanation:

1. In our investigation, after transforming data using PCA,, the next step involved sourcing historical data using yfinance, which provided us with a wealth of monthly data encompassing all equities within the S&P 500 index (500 stocks), our chosen universe of stocks. Following data acquisition, we embarked on a process of dataset transformation, wherein we meticulously curated the dataset to retain only the most pertinent information necessary for our analysis, specifically focusing on factors and thematic returns relevant to our research objectives.

2. Subsequently, we leveraged code implementation to gain insights into the newly transformed dataset, enabling us to conduct exploratory data analysis and assess the correlations among various features. Furthermore, we performed multiple linear regression analysis to derive stock betas for the factors, thereby extending our understanding of the underlying relationships within the dataset and facilitating comparisons with Fama-French monthly returns.

3. Despite encountering challenges in computing alpha performance using an optimization portfolio, we pivoted our approach to explore an alternative methodology. Our revised strategy revolved around the postulation that the performance of factors could be discerned by examining their correlation with both the S&P 500 performance and market beta. Specifically, we hypothesized that deviations in correlation metrics relative to historical norms could signify a phenomenon known as factor decay, attributed to shifts in market dynamics.

4. By scrutinizing the correlation patterns between factors and key market indicators, such as the S&P 500 performance and market beta, we gained valuable insights into the evolving nature of factor performance and its implications for portfolio management. The observed trends underscored the significance of monitoring factor behavior in the context of changing market conditions, elucidating potential opportunities and risks for informed decision-making.

5. Next we focused on optimizing the portfolio based on the various scenarios and seeing the various performances over time of the portfolios.

Optimization method

The crown jewel of our project is the optimization model and we created this model with the objective of minimizing the error in achieving our specified target exposures to the factors we identified above as the least impactful by decay with the least transaction cost as well as greater out of sample performance.

Mathematical formulation explanation

We assigned the target factor exposures as per the S&P 500 exposure to these factors in the sample and our hypothesis was that by sticking to these factor exposure, we would achieve outperformance. Then we defined the parameters for the LP problem by setting the parameter budget as 1 million, the maximum stocks as 80, the initial base weights of our portfolio as 0 for all the equities, and the maximum transaction cost of the entire portfolio as 10000.

Then, we created decision variables to measure the error in tracking the factor exposures, the change in the portfolio for each equity in the Universe of stocks (S&P 500, so 500 stocks), the weighting of each equity, to know if a particular equity of the stock universe is part of the portfolio, to know the number of shares of each equity in the portfolio, and finally to measure the total transaction cost for each equity. The objective of the initial model was to minimize the error variable across every in sample period and the constraints on the objective problem were budget constraint, change in the portfolio constraints, constraint to make sure that error in tracking the factor exposure, constraint to restrict the number of stocks in the portfolio for each equity, constraint to restrict the total transaction cost, and to identify the number of stocks for each equity.

Using this model, we were able to determine the number of stocks in the portfolio to achieve the factor exposure as well as the weighting of the portfolio to each equity to achieve the target exposures. We were also able to determine the portfolio factor exposures and its absolute difference from its target exposure in the verification set of our dataset.

```
Target={}
Target["prc"] = -0.248
Target["seas_1_1an"] = -2.224
Target["emp_gr1"] = 1.006
Target["rd_me"] = 0.370
Target["resff3_6_1"] = -0.477
Target["eqnpo_12m"] = -1.510
Target["rd_sale"] = 0.215
Target["aliq_at"] = -1.098
Target["ret_12_7"] = 0.323
Target["ret_3_1"] = -0.659
Target["age"] = 1.740
Target["rmax5_rvol_21d"] = 1.031
budget = 1000000
maxstocks = 100
base_weights = base_w.T*0
max_trcost=10000
```

	SP_500	Target	Optimization	Abs. Diff
ret_12_7	15.712976	-0.248	-0.250616	0.0026
ret_3_1	-1.601758	-2.224	-2.247460	0.0235
rmax5_rvol_21d	-7.881054	0.370	0.373903	0.0039
rd_sale	7.680539	1.006	1.016612	0.0106
emp_gr1	-4.493307	-0.477	-0.482032	0.0050
age	-6.814309	-1.510	-1.525929	0.0159
resff3_6_1	-3.548225	0.215	0.217268	0.0023
seas_1_1an	1.740904	-1.098	-1.109583	0.0116
prc	5.388019	0.323	0.326407	0.0034
rd_me	2.211865	-0.659	-0.665952	0.0070
eqnpo_12m	3.866359	1.740	1.758355	0.0184
aliq_at	4.423794	1.031	1.041876	0.0109

Results

Our first step was establishing what our most important factors were, which we did by finding our principal components and breaking them down into the original factors. We then analyzed the magnitude of the contributions from each of these factors to confirm that each of these factors were showing up across all the 5 most important principal components (which explained 65.0545% of the variance). 18 out of the top 20 factors (scored by summing up the magnitude of factor importance) were present in the top 10 most important factors across the five principal components. Only profit_growth and noa_at didn't show up (i.e. they flew under the radar and weren't in the top 10 most important factors across the five principal components, but were important enough to show up in the top 20). Each of these principal components broken down is below:

```
0 age            0.693812
eqnetis_at     0.286281
fcf_me         0.272676
aliq_mat       0.255446
netis_at       0.239487
chcsho_12m     0.198198
kz_index       0.192621
ret_12_7       0.169719
o_score        0.147355
seas_1_1an     0.119542
dtype: float64
```

```
1 ret_12_7         0.514719
rd_me            0.425526
corr_1260d       0.415214
ebit_sale        0.241034
age              0.240296
seas_1_1an       0.230706
o_score          0.207348
aliq_mat         0.192987
tax_gr1a         0.164268
profit_growth    0.155289
dtype: float64
```

```
2 corr_1260d       0.521515
aliq_mat         0.404586
age              0.339487
kz_index         0.279175
o_score          0.259189
rmax5_rvol_21d   0.210617
ebit_sale        0.197863
netis_at         0.185631
tangibility      0.165997
eqnetis_at       0.149782
dtype: float64
```

```
3 ret_12_7       0.593257
corr_1260d     0.453628
seas_1_1an     0.367594
aliq_mat       0.350078
ppeinv_gr1a    0.177363
rd_me          0.148337
ebit_sale      0.142989
chcsho_12m     0.132439
noa_at         0.115535
tax_gr1a       0.113555
dtype: float64
```

```
4 qmj_growth       0.431720
dgp_dsale        0.334173
aliq_mat         0.321677
ppeinv_gr1a      0.272880
rmax5_rvol_21d   0.263736
netis_at         0.251244
kz_index         0.244278
tangibility      0.224907
fcf_me           0.223473
cowc_gr1a        0.185066
dtype: float64
```

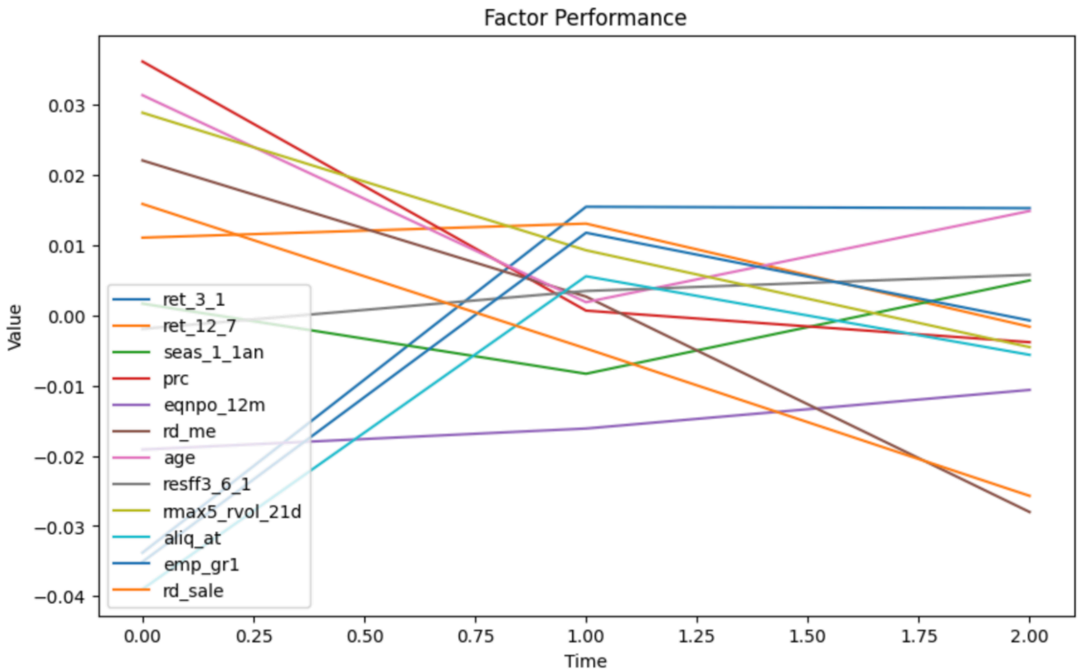
And this is the sum of the magnitudes of the top 20 most important factors:

```
[('cowc_gr1a', 0.18506640287115023),
 ('tax_gr1a', 0.2778228823282733),
 ('chcsho_12m', 0.3306371152130011),
 ('dgp_dsale', 0.3341734488750458),
 ('tangibility', 0.3909042091258319),
 ('qmj_growth', 0.4317198939794481),
 ('eqnetis_at', 0.436063246427539),
 ('ppeinv_gr1a', 0.45024245037316657),
 ('rmax5_rvol_21d', 0.474352334364622),
 ('fcf_me', 0.4961492263629269),
 ('rd_me', 0.5738628320919299),
 ('ebit_sale', 0.5818857179040349),
 ('o_score', 0.6138920469825332),
 ('netis_at', 0.6763617834247944),
 ('kz_index', 0.7160743494392707),
 ('seas_1_1an', 0.717842187962691),
 ('age', 1.2735942768713093),
 ('ret_12_7', 1.2776945582561994),
 ('corr_1260d', 1.3903563538676853),
 ('aliq_mat', 1.524774472664033)]
```

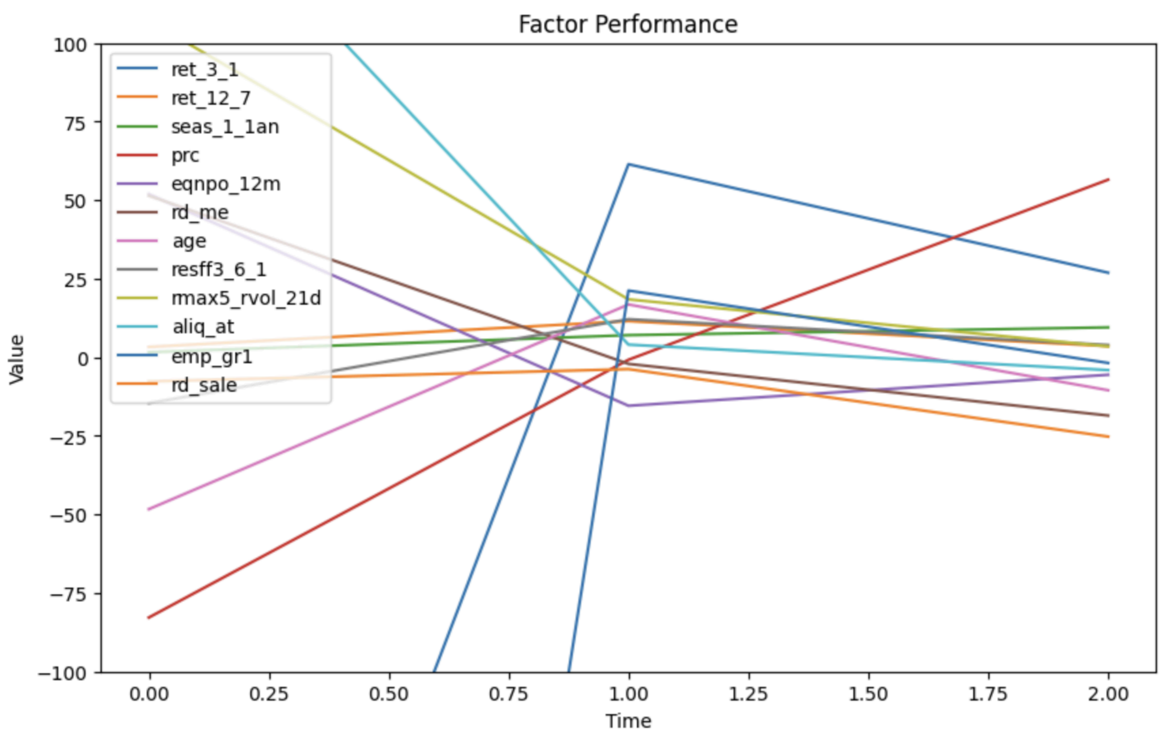
These results indicate that the efficient market hypothesis is not true. We are looking at more than the Fama-French 3 Factors (in fact size and value don't even appear), so our results indicate that these factors aren't important in the grand scheme of things and you can't account for systematic risk with just 3 factors. The first non-market 5 Factor to appear, Investment, is rd_me, or "R&D scaled by Market Equity", the 10th factor on the list.

The next step was to take the factors with available data and see which have decayed. These factors were ret_3_1, ret_12_7, seas_1_1an, prc, eqnpo_12m, rd_me, age, resff3_6_1, rmax5_rvol_21d, aliq_at, emp_gr1, and rd_sale. These translates to Momentum 1-3 Months, Momentum 7-12 Months, 1 Year Annual Seasonality, Price per Share, Net Equity Payout - 12 Month, R&D scaled by Market Equity, Age, Residual Momentum - 6 Month, Max Return to Volatility, Liquidity scaled by lagged Assets, Employee Growth 1 yr, and R&D-to-sales.

In absolute terms:



In relative terms (percentage difference):



Scenarios

Explanation of the variables

weight[t,i]: weight of ticker i at time t

aux[t,i]: auxiliary variable that is the absolute value of the current weight minus the previous weight

monthlyRet[t,i]: return of stock i at time t

FAC[t,j]: (where j is the Target beta)

Target[j]: what our target beta exposure is to "j"

ε[t]: error at time t

sPrice, shares[t,i]: both describe stock-related things for ticker i at time t

trCosts[i]: transaction cost for that ticker

totalTrCosts[t,i]: combined transaction cost for that ticker at that time

bin[t,i]: binary variable for the stock being present in the portfolio

Budget, maxStocks, maxTRCost (max trading cost) should be self-explanatory

	Optimal
Weight_ACGL	7.97%
Weight_AES	3.02%
Weight_BA	8.92%
Weight_BG	11.96%
Weight_COP	0.96%
Weight_DECK	4.21%
Weight_DRI	9.17%
Weight_DUK	-7.10%
Weight_DXCM	10.68%
Weight_KHC	3.35%
Weight_MRNA	4.06%
Weight_PWR	8.63%
Weight_SO	20.02%
Weight_TSCO	14.14%

All of the scenarios implemented two important pieces: the rolling window (ie updating our portfolio trading day to trading day) and quarterly testing. Overall, we generally outperformed the S&P 500 with all the portfolios. All of the portfolios had a large dip in early-mid 2021, but even then the portfolios recovered. Scenario 3's was a more smoothed out version of Scenarios 1 and 2, which lines up with the expectation that reducing error and transaction costs would slightly decrease robustness.

Scenario 1

Here we are focused on reducing the error.

Objective: $\min \sum_t \epsilon_t$

Constraints:

$\text{aux}_{t,i} \geq \text{weight}_{t,i}$ - $\text{weight}_{t-1,i}$ $\forall i, t \in 1..T$

$\text{aux}_{t,i} \geq \text{weight}_{t-1,i}$ - $\text{weight}_{t,i}$ $\forall i, t \in 1..T$

$\sum_i \text{aux}_{t,i} \leq 1 \; \forall t \in 1..T$

$\sum_i \text{weight}_{t,i} = 1 \; \forall t \in 1..T$

$\sum_i \text{weight}_{t,i}$ * $(\text{monthlyRet}_{t,i}\text{-FAC}_{t,RFR})$- $\epsilon_t = \sum_j \text{Target}_j\text{*FAC}_{t,j}$ $\forall$ i, j $\in$ target names, t=1..T

$\text{sPrice}_{t,i}\text{*shares}_{t,i} = \text{aux}_{t,i}$ * budget $\forall i, t \in 1..T$

$\sum_i \frac{aux_{t,i}*budget*trCost_i}{sPrice_{t,i}} = \text{totalTrCost}\ _{t,i} \; \forall i, t \in 1..T$

$\sum_i \text{totalTrCost}_{t,i} \leq \text{maxTrCost}_t \; \forall t \in 1..T$

$\text{weight}_{t,i} \leq \text{bin}_{t,i} \; \forall i, t \in 1..T$

$\sum_i \text{bin}_{t,i} \leq \text{maxStocks} \; \forall t \in 1..T$

Parameters

maxStocks = 80

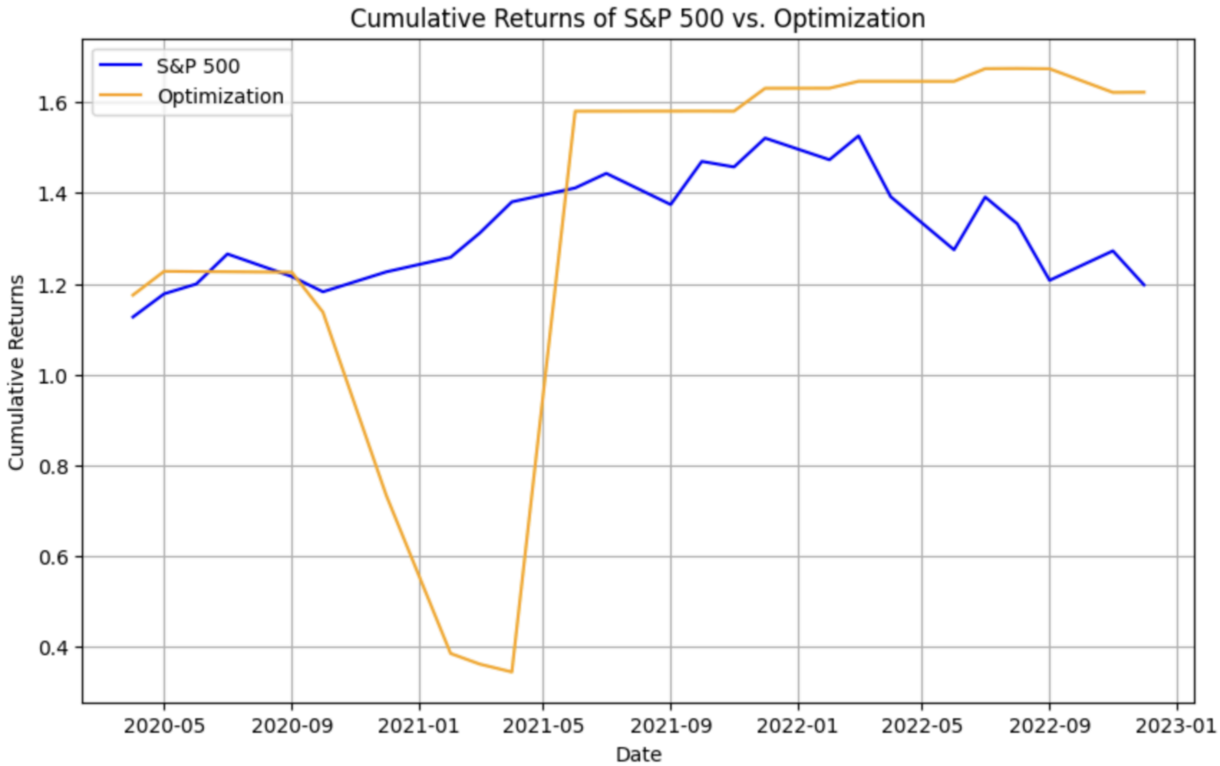
maxTrCost = 2000

budget = 1000000

$$\text{Target} = \begin{bmatrix} \text{prc} & -0.248 \\ \text{seas-1-1an} & -2.224 \\ \text{emp-gr1} & 1.006 \\ \text{rd-me} & 0.370 \\ \text{resff3-6-1} & -0.477 \\ \text{eqnpo-12m} & -1.510 \\ \text{rd-sale} & 0.215 \\ \text{aliq-at} & -1.098 \\ \text{ret-12-7} & 0.323 \\ \text{ret-3-1} & -0.659 \\ \text{age} & 1.740 \\ \text{rmax5-rvol-21d} & 1.031 \end{bmatrix}$$

Note: weights at time 0 are 0.

Result:



The findings reveal that the optimized model, implemented with specific weights, has surpassed the performance of the S&P 500 index. Remarkably, despite the model's capacity to accommodate up to 100 stocks, only 12 were utilized in constructing the portfolio, demonstrating a focused selection approach. While the portfolio exhibited a slight lag initially, attributable to its passive or semi-active strategy, it has exhibited remarkable resilience and momentum over the long term. Leveraging the strengths of factor performance, the portfolio has achieved impressive growth and stability, reflecting the effectiveness of its strategic design.

Scenario 2

In this scenario we were focused on maximizing the return. To that end, we were focused on changing up the portfolio a lot and exposing the portfolio to slightly more risky stocks. We are unsure what exactly causes the huge dip.

Objective: $\max \sum_t \text{totalRet}_t$

Constraints:

$\text{aux}_{t,i} \geq \text{weight}_{t,i}$ - $\text{weight}_{t-1,i}$ $\forall i, t \in 1..T$

$\text{aux}_{t,i} \geq \text{weight}_{t-1,i}$ - $\text{weight}_{t,i}$ $\forall i, t \in 1..T$

$\sum_i \text{aux}_{t,i} \leq 1 \ \forall t \in 1..T$

$\sum_i \text{weight}_{t,i} = 1 \ \forall t \in 1..T$

$\sum_i \text{weight}_{t,i} * (\text{monthlyRet}_{t,i}\text{-FAC}_{t,RFR})\text{-}\ \epsilon_t \leq \sum_j \text{Target}_j\text{*FAC}_{t,j}$ ∀ i, j ∈ target names, t∈1..T

$\sum_i \text{weight}_{t,i} * (\text{monthlyRet}_{t,i}\text{-FAC}_{t,RFR})\text{-}\ \epsilon_t \geq \sum_j \text{Target}_j\text{*FAC}_{t,j}$ ∀ i, j ∈ target names, t∈1..T

$\text{sPrice}_{t,i}\text{*shares}_{t,i} = \text{aux}_{t,i} * \text{budget} \ \forall i, t \in 1..T$

$\sum_i \frac{aux_{t,i}*budget*trCost_i}{sPrice_{t,i}} = \text{totalTrCost}_{\ t,i} \ \forall i, t \in 1..T$

$\sum_i \text{totalTrCost}_{t,i} \leq \text{maxTrCost}_t \ \forall t \in 1..T$

$\text{weight}_{t,i} \leq \text{bin}_{t,i} \ \forall i, t \in 1..T$

$\sum_i \text{bin}_{t,i} \leq \text{maxStocks} \ \forall t \in 1..T$

$\text{totalRet}_t = \sum_i \text{weight}_{t,i} * \text{monthlyRet}_{t,i}$

$-.1 \leq \epsilon_t \leq .1 \ \forall t \in 1..T$

Parameters

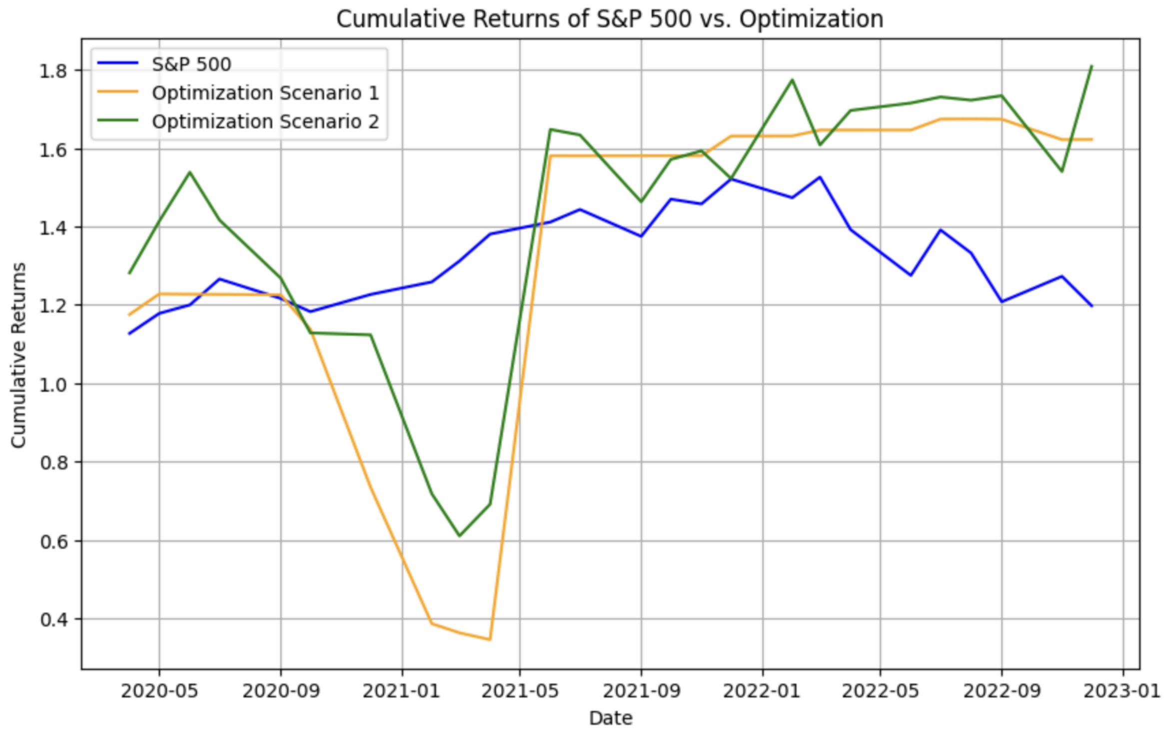
$\text{maxStocks} = 80$

$\text{maxTrCost} = 2000$

$\text{budget} = 1000000$

$$\text{Target} = \begin{bmatrix} \text{prc} & -0.248 \\ \text{seas-1-1an} & -2.224 \\ \text{emp-gr1} & 1.006 \\ \text{rd-me} & 0.370 \\ \text{resff3-6-1} & -0.477 \\ \text{eqnpo-12m} & -1.510 \\ \text{rd-sale} & 0.215 \\ \text{aliq-at} & -1.098 \\ \text{ret-12-7} & 0.323 \\ \text{ret-3-1} & -0.659 \\ \text{age} & 1.740 \\ \text{rmax5-rvol-21d} & 1.031 \end{bmatrix}$$

Result:



The findings reveal that the optimized model, implemented with specific weights, has surpassed the performance of the S&P 500 index and the first optimized model. This can be largely attributed to the fact that our model is designed to maximize return and minimize tracking error in factor exposure. In spite of this, the portfolio still exhibited a slight lag initially, but better than the performance of the first model. This also due to its passive or semi-active strategy, it has still exhibited remarkable resilience and momentum over the long term. Leveraging the strengths of factor performance, the portfolio has achieved even more impressive growth and stability, reflecting the effectiveness.

Scenario 3

Objective: $\min \sum_t \epsilon_t + \sum_t \text{totalTrCost}_t$

Constraints:

$\text{aux}_{t,i} \geq \text{weight}_{t,i}$ - $\text{weight}_{t-1,i}$ $\forall i, t \in 1..T$

$\text{aux}_{t,i} \geq \text{weight}_{t-1,i}$ - $\text{weight}_{t,i}$ $\forall i, t \in 1..T$

$\sum_i \text{aux}_{t,i} \leq 1 \; \forall t \in 1..T$

$\sum_i \text{weight}_{t,i} = 1 \; \forall t \in 1..T$

$\sum_i \text{weight}_{t,i}$ * ($\text{monthlyRet}_{t,i}$-$\text{FAC}_{t,RFR}$)- $\epsilon_t = \sum_j \text{Target}_j$*$\text{FAC}_{t,j}$ ∀ i, j ∈ target names, t=1..T

$\text{sPrice}_{t,i}$*$\text{shares}_{t,i} = \text{aux}_{t,i}$ * budget $\forall i, t \in 1..T$

$\sum_i \frac{aux_{t,i}*budget*trCost_i}{sPrice_{t,i}} = \text{totalTrCost}_{\;t,i} \; \forall i, t \in 1..T$

$\sum_i \text{totalTrCost}_{t,i} \leq \text{maxTrCost}_t \; \forall t \in 1..T$

$\text{weight}_{t,i} \leq \text{bin}_{t,i} \; \forall i, t \in 1..T$

$\sum_i \text{bin}_{t,i} \leq \text{maxStocks} \; \forall t \in 1..T$

Parameters

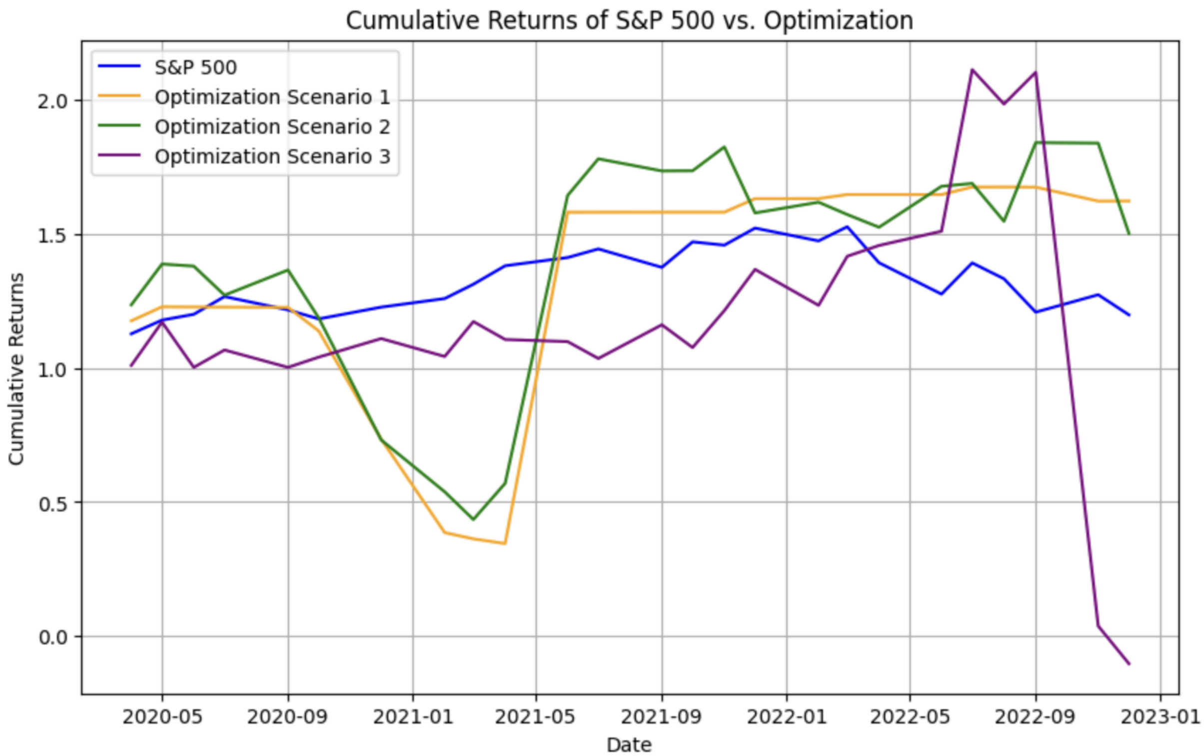
maxStocks = 80

maxTrCost = 2000

budget = 1000000

$$\text{Target} = \begin{bmatrix} \text{prc} & -0.248 \\ \text{seas-1-1an} & -2.224 \\ \text{emp-gr1} & 1.006 \\ \text{rd-me} & 0.370 \\ \text{resff3-6-1} & -0.477 \\ \text{eqnpo-12m} & -1.510 \\ \text{rd-sale} & 0.215 \\ \text{aliq-at} & -1.098 \\ \text{ret-12-7} & 0.323 \\ \text{ret-3-1} & -0.659 \\ \text{age} & 1.740 \\ \text{rmax5-rvol-21d} & 1.031 \end{bmatrix}$$

Results:



The findings reveal that the optimized model in a different scenario, we observed a model where the portfolio initially tracked the index closely but exhibited significantly improved performance towards the end of the test period. However, this model encountered earlier-than-expected factor decay, which impacted its performance. This decay can be attributed to the increasing transaction costs over time, particularly as asset prices rose. Despite the initial setback due to factor decay, the portfolio demonstrated remarkable resilience and momentum in the latter part of the test period, showcasing its potential for long-term growth and stability with appropriate adjustments to mitigate transaction costs

Conclusion

As shown, R&D-related factors experienced significant decay, aligning with expectations following the publication of the Fama-French 5 Factor model in 2013, which diminished the returns from this factor. Several other factors, including Price, Age, and Max Return to Volatility, also exhibited decay, while the impact on others remained ambiguous.

From an optimization perspective, these findings underscore the critical need for continuous monitoring and adjustment in factor-based investment strategies. Factor decay can compromise the efficacy of traditional factor models, necessitating ongoing optimization to maintain performance. Portfolio managers should regularly reassess factor exposures, identify underperforming factors, and recalibrate portfolio allocations to optimize returns and manage risk effectively.

The ambiguous results observed in some factors highlight the complexity and idiosyncrasy inherent in factor dynamics. A sophisticated optimization approach is required, combining quantitative analysis, qualitative judgment, and robust stress testing methodologies. Prioritizing diversification across factors and asset classes can enhance portfolio resilience and mitigate the adverse effects of factor decay.

To further optimize trading strategies, investors might consider incorporating adaptive factor rotation techniques that dynamically adjust factor exposures based on current market conditions and factor performance trends. This strategy enables investors to exploit emerging opportunities while mitigating the risks associated with factor decay. Additionally, employing advanced risk management techniques, such as stop-loss orders and position sizing based on factor volatility, can help optimize portfolio performance amid evolving factor dynamics.

Overall, the observed factor decay underscores the imperative for agile and adaptive optimization practices in factor-based investment strategies. By embracing a proactive approach to monitoring, analyzing, and responding to factor dynamics, investors can enhance portfolio robustness and navigate changing market environments with confidence. The optimized model, with its specific weighting strategy designed to maximize return and minimize tracking error, demonstrates the potential for achieving superior performance and stability compared to traditional models, reflecting the effectiveness of these optimization techniques.

References

Asness, Cliff S., Momentum in Japan: The Exception that Proves the Rule (March 3, 2011). The Journal of Portfolio Management, Forthcoming, Available at SSRN: https://ssrn.com/abstract=1776123

Falck, Antoine and Rej, Adam and Thesmar, David, When Systematic Strategies Decay (May 14, 2021). Available at SSRN: https://ssrn.com/abstract=3845928 or http://dx.doi.org/10.2139/ssrn.3845928

EHSANI, S. and LINNAINMAA, J.T. (2022), Factor Momentum and the Momentum Factor. The Journal of Finance, 77: 1877-1919. https://doi.org/10.1111/jofi.13131

Appendix

AMPL .mod files-

Scenario 1:

```
############################
# SETS
############################
set Time;#monthly dates
set Funds;#tickers
set factors;#factors after PCA and the risk free rate

############################
# PARAMETERS
############################
param exp_ret{Funds};
param base{Funds};
param monthlyRet{Time,Funds};
param factormonthly{Time,factors};
param Target{factors};
param sPrice{Time,Funds};
param budget;
param trcost{Funds};
param maxTrcost;
param maxStocks;

############################
# Variables
############################
var aux{Time,Funds} >= 0;
var weight{Time,Funds} >= 0;
var error{Time} >= 0;
var shares{Time,Funds} >= 0;
var totalTrCost{Time,Funds} >= 0;
var b{Time,Funds} >= 0 binary;

############################
# Objective
############################
minimize sum{t in Time} error[t];

############################
# Constraints
############################
subject to absolute_constraint1{i in Funds}: weight[0,i] = 1;
subject to absolute_constraint2{i in Funds,t in Time}: aux[t,i] >=
weight[t,i]-weight[t-1,i];
subject to absolute_constraint3{i in Funds}: aux[i] >= weight[t-1,i]-weight[t,i];
subject to absolute_constraint4{t in Time}: sum{i in Funds}aux[i] = 0;
subject to obj_constraint1{t in Time}: -error[t] + sum{i in Funds} (weight[t,i] *
(monthlyRet[t,i]-factormonthly[t,0]))= sum{j in factors} (Target[j] *
factormonthly[t,j]);
subject to obj_constraint2: sum{i in Funds} weight[i] = 1;
subject to share_constraint{t in Time,i in Funds}: sPrice[t,i] * shares[t,i] =
aux[t,i] * budget;
subject to transaction_constraint{t in Time}: sum{i in Funds} (aux[t,i] * budget *
trcost[i]/sprice[t,i]) = totalTrCost[t,i];
subject to max_constraint{t in Time}: sum{i in Funds} totalTrCost[t,i] <= maxTrcost;
subject to binary_constraint{t in Time,i in Funds}: weight[i] <= b[t,i];
subject to max_stock_constraint{t in Time}: sum{i in Funds} b[t,i] <= maxStocks;
```

Scenario 2:

```
############################
# SETS
############################
set Time;#monthly dates
set Funds;#tickers
set factors;#factors after PCA and the risk free rate

############################
# PARAMETERS
############################
param exp_ret{Funds};
param base{Funds};
param monthlyRet{Time,Funds};
param factormonthly{Time,factors};
param Target{factors};
param sPrice{Time,Funds};
param budget;
param trcost{Funds};
param maxTrcost;
param maxStocks;

############################
# Variables
############################
var aux{Time,Funds} >= 0;
var weight{Time,Funds} >= 0;
var error{Time} >= 0;
var totalRet{Time} >= 0;
var shares{Time,Funds} >= 0;
var totalTrCost{Time,Funds} >= 0;
var b{Time,Funds} >= 0 binary;

############################
# Objective
############################
maximize sum{t in Time} totalRet[t];

############################
# Constraints
############################
subject to absolute_constraint1{i in Funds}: weight[0,i] = 1;
subject to absolute_constraint2{i in Funds,t in Time}: aux[t,i] >=
weight[t,i]-weight[t-1,i];
subject to absolute_constraint3{i in Funds}: aux[i] >= weight[t-1,i]-weight[t,i];
subject to absolute_constraint4{t in Time}: sum{i in Funds}aux[i] = 0;
subject to obj_constraint1{t in Time}: -error[t] + sum{i in Funds} (weight[t,i] *
(monthlyRet[t,i]-factormonthly[t,0])) <= sum{j in factors} (Target[j] *
factormonthly[t,j]);
subject to obj_constraint2{t in Time}: -error[t] + sum{i in Funds} (weight[t,i] *
(monthlyRet[t,i]-factormonthly[t,0])) >= sum{j in factors} (Target[j] *
factormonthly[t,j]);
subject to obj_constraint3: sum{i in Funds} weight[i] = 1;
subject to share_constraint{t in Time,i in Funds}: sPrice[t,i] * shares[t,i] =
aux[t,i] * budget;
subject to transaction_constraint{t in Time}: sum{i in Funds} (aux[t,i] * budget *
trcost[i]/sprice[t,i]) = totalTrCost[t,i];
subject to max_constraint{t in Time}: sum{i in Funds} totalTrCost[t,i] <= maxTrcost;
subject to binary_constraint{t in Time,i in Funds}: weight[i] <= b[t,i];
subject to max_stock_constraint{t in Time}: sum{i in Funds} b[t,i] <= maxStocks;
subject to return_constraint{t in Time}: totalRet[t] = sum{i in Funds} weight[t,i] *
monthlyRet[t,i];
subject to error_constraint{t in Time}: -0.1 <= error[t] <= 0.1;
```

Scenario 3:

```
#############################
# SETS
#############################
set Time;#monthly dates
set Funds;#tickers
set factors;#factors after PCA and the risk free rate

#############################
# PARAMETERS
#############################
param exp_ret{Funds};
param base{Funds};
param monthlyRet{Time,Funds};
param factormonthly{Time,factors};
param Target{factors};
param sPrice{Time,Funds};
param budget;
param trcost{Funds};
param maxTrcost;
param maxStocks;

#############################
# Variables
#############################
var aux{Time,Funds} >= 0;
var weight{Time,Funds} >= 0;
var error{Time} >= 0;
var totalRet{Time} >= 0;
var shares{Time,Funds} >= 0;
var totalTrCost{Time,Funds} >= 0;
var b{Time,Funds} >= 0 binary;

#############################
# Objective
#############################
maximize sum{t in Time} totalRet[t] + sum{t in Time} error[t];

#############################
# Constraints
#############################
subject to absolute_constraint1{i in Funds}: weight[0,i] = 1;
subject to absolute_constraint2{i in Funds,t in Time}: aux[t,i] >=
weight[t,i]-weight[t-1,i];
subject to absolute_constraint3{i in Funds}: aux[i] >= weight[t-1,i]-weight[t,i];
subject to absolute_constraint4{t in Time}: sum{i in Funds}aux[i] = 0;
subject to obj_constraint1{t in Time}: -error[t] + sum{i in Funds} (weight[t,i] *
(monthlyRet[t,i]-factormonthly[t,0])) <= sum{j in factors} (Target[j] *
factormonthly[t,j]);
subject to obj_constraint2{t in Time}: -error[t] + sum{i in Funds} (weight[t,i] *
(monthlyRet[t,i]-factormonthly[t,0])) >= sum{j in factors} (Target[j] *
factormonthly[t,j]);
subject to obj_constraint3: sum{i in Funds} weight[i] = 1;
subject to share_constraint{t in Time,i in Funds}: sPrice[t,i] * shares[t,i] =
aux[t,i] * budget;
subject to transaction_constraint{t in Time}: sum{i in Funds} (aux[t,i] * budget *
trcost[i]/sprice[t,i]) = totalTrCost[t,i];
subject to max_constraint{t in Time}: sum{i in Funds} totalTrCost[t,i] <= maxTrcost;
subject to binary_constraint{t in Time,i in Funds}: weight[i] <= b[t,i];
subject to max_stock_constraint{t in Time}: sum{i in Funds} b[t,i] <= maxStocks;
```

Python code excerpts:

```
#############
### SETUP ###
#############
factor_world1 = pd.read_csv('[developed]_[all_factors]_[monthly]_[vw_cap].csv')
factor_world2 = pd.read_csv('[developed]_[all_themes]_[monthly]_[vw_cap].csv')
factor_world3 = pd.read_csv('[developed]_[all_factors]_[daily]_[vw_cap].csv')
factor_world4 = pd.read_csv('[developed]_[all_themes]_[daily]_[vw_cap].csv')
factor_world_m = pd.concat([factor_world1, factor_world2], ignore_index=True)
factor_world_d = pd.concat([factor_world3, factor_world4], ignore_index=True)
# factor_world = pd.concat([factor_world1, factor_world2], ignore_index=True)

factor_world_m.rename(columns={"name": "factors"}, inplace=True)
factor_world_m.drop(["location", "n_countries", "freq", "direction"], axis=1,
inplace=True)
factor_world_weighting_monthly = factor_world_m.drop("weighting", axis=1)
factor_world_weighting_monthly['date'] =
pd.to_datetime(factor_world_weighting_monthly['date'])
pivot_df = factor_world_weighting_monthly.pivot(index='date', columns='factors')
factor_world_monthly = pivot_df[12:]

factor_world_d.rename(columns={"name": "factors"}, inplace=True)
factor_world_d.drop(["location", "n_countries", "freq", "direction"], axis=1,
inplace=True)
factor_world_weighting_daily = factor_world_d.drop("weighting",axis=1)
factor_world_weighting_daily['date'] =
pd.to_datetime(factor_world_weighting_daily['date'])
pivot_df_d = factor_world_weighting_daily.pivot(index='date', columns='factors')
factor_world_daily = pivot_df[12:]
```

—---

```
###############
### Here's where we do PCA
###############
full_monthly_data_scaled = scaler.fit_transform(df_interpolated)
pca_1 = PCA()
pca_1.fit(full_monthly_data_scaled)
explained_variance_ratio = pca_1.explained_variance_ratio_
selected_components = pca_1.components_[:n_components]

# Print explained variance ratio
print("Explained Variance Ratio:", explained_variance_ratio, '\n',
len(explained_variance_ratio))

# Get names of features with highest loadings for each principal component
feature_names = df_interpolated.columns
```

```
top_features = []
for component in selected_components:
    component_loadings = pd.Series(component, index=feature_names)

top_features.append(component_loadings.abs().sort_values(ascending=False).index[:n_com
ponents])  # Get top 5 features with highest loadings

print("\nSum of first", n_components, "components in importance to final product")
sum(explained_variance_ratio[:n_components])
```

```
### Work with a plot
# Project data onto the selected principal components
projected_data = np.dot(full_monthly_data_scaled, selected_components.T)

# Scatter plot of the first two principal components
plt.figure(figsize=(8, 6))
plt.scatter(projected_data[:, 0], projected_data[:, 1], alpha=0.5)
plt.title('PCA Scatter Plot (PC1 vs PC2)')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.grid(True)
plt.show()
```

```
# Scree plot to visualize explained variance ratio
plt.figure(figsize=(8, 6))
plt.plot(np.arange(1, len(explained_variance_ratio) + 1),
np.cumsum(explained_variance_ratio), marker='o', linestyle='-')
plt.title('Scree Plot')
plt.xlabel('Number of Principal Components')
plt.ylabel('Cumulative Explained Variance Ratio')
plt.grid(True)
plt.show()
# It's cool that the Pareto principle kind of shows up here
```

```
######################################################
### Here is where we clean up our new monthly data ###
######################################################
# df with columns of how many na's they have
missing_count = factor_world_trading.isnull().sum()

df_interpolated = factor_world_trading.dropna(thresh=len(factor_world_trading) - 30,
axis=1)
```

```
# Identify features with missing values between 5 and 30
interpolate_features_5_30 = missing_count[(missing_count > 5) & (missing_count <=
30)].index
# Interpolate missing values for features with 5 to 30 missing values
df_interpolated[interpolate_features_5_30] =
df_interpolated[interpolate_features_5_30].interpolate(method='linear', axis=0)

# Identify features with 5 or fewer missing values
fill_median_features = missing_count[missing_count <= 5].index
# Fill missing values with median for features with 5 or fewer missing values
for feature in fill_median_features:
    median_value = df_interpolated[feature].median()
    df_interpolated[feature].fillna(median_value, inplace=True)

# Now df_interpolated contains the DataFrame with interpolated and median-filled
missing values

# Drop any further rows containing NaN values
df_cleaned = df_interpolated.dropna(inplace=True)
—-----------------------------------------------------------------------
#########################
### Transaction Costs ###
#########################
random.seed(1)
t_cost = {}
keys       = list(tickers)
#values     = [.15 for i in range(len(keys))]
values = list(random.uniform(.009,.021) for i in range(len(tickers)))
for key,value in zip(keys,values):
    t_cost[key] = value
—----------------------------------------------------------------------
######################
### LP Formulation ###
######################
# Create LP problem
prob = LpProblem("Factor_Decay_Model", LpMinimize)

# Variables
aux = {i: LpVariable(f"Aux_{i}", lowBound=0) for i in nah.index}
weight = {i: LpVariable(f"Weight_{i}", lowBound=0) for i in nah.index}
error = {t: LpVariable(f"Error_{t}", lowBound=0) for t in nahh.index}
shares = {i: LpVariable(f"Shares_{i}", lowBound=0) for i in nah.index}
trcost = {i: LpVariable(f"TrCost_{i}", lowBound=0) for i in nah.index}
b = {i: LpVariable(f"B_{i}", cat='Binary') for i in nah.index}

# Objective
```

```
prob += lpSum(error[t] for t in nahh.index)


# Constraints
for i in nah.index:
    prob += aux.loc[t,i] <= base_weights[i] - weight[i]
    prob += aux.loc[t,i] >= base_weights[i] - weight[i]
    prob += s_price[i] * shares[i] <= aux.loc[t,i] * budget
    prob += s_price[i] * shares[i] >= aux.loc[t,i] * budget
    prob += aux.loc[t,i] * budget * t_cost[i] <= s_price[i] * trcost[i]
    prob += aux.loc[t,i] * budget * t_cost[i] >= s_price[i] * trcost[i]
    prob += b[i] >= weight[i]
    prob += trcost[i] <= max_trcost

for t in nahh.index:
    prob += -error[t] + lpSum(weight[i] * (new_monthly_data.loc[t, i] -
ff3_monthly.loc[t, "RF"]) for i in tickers) <= lpSum(Target[j] * new_dataset.loc[t, j]
for j in ffeature_names)
    prob += -error[t] + lpSum(weight[i] * (new_monthly_data.loc[t, i] -
ff3_monthly.loc[t, "RF"]) for i in tickers) >= lpSum(Target[j] * new_dataset.loc[t, j]
for j in ffeature_names)

prob += lpSum(weight[i] for i in nah.index) == 1
prob += lpSum(b[i] for i in nah.index) <= maxstocks

# Solve the problem
prob.solve()

# Print the results
print("Status:", LpStatus[prob.status])
for v in prob.variables():
    print(v.name, "=", v.varValue)
print("Total Error =", value(prob.objective))
```