CMOR 462 Project 2: Construction of an Index Fund

Alex Zalles, Michael Khalfin, Matthew Banschbach

October 2023

1 Introduction

Index funds are one of the most important aspects of the Stock Market as we know it today. By combining many stocks under one umbrella fund, variance in individual returns is accounted for, and consumers are more encouraged to purchase and participate in the market. However, there are many assumptions that go into the formulation of an index fund and processes that occur behind the scenes in its generation. Our goal for this project is to replicate this process, outlining the assumptions we make along the way and discovering the methods that result in index funds of greatest return, least volatility, and more. Through this replication, we hope to not only better understand the processes but also acquire the capabilities to create models with tangible impact on the stock market.

This process begins with data scrubbing, where we take historical data from the CRSP database and translate it into a more usable format. We then use the single factor model to estimate the covariances between securities in the form of covariance matrices–a process that requires running multiple linear-regressions. After estimating inputs and converting them to show correlations, we select the entries of the matrix with the largest capitalization for each month which we are analyzing, and then cluster stocks by similarity in their correlations. However, the implications of similarity depend on the metric used to measure it. Thus, the first step in our experimentation is altering the distance measure used for clustering and analyzing how the outputs vary. After clustering the stocks, we choose a representative stock from each cluster, with this set of selected securities comprising the index fund. Here, the question becomes that of selecting the representative stock from each cluster. We want the optimal selection method, and so another aspect of our experimentation involves evaluating different stock selection methods and their impacts. Finally, we want to determine how these representative stocks should be weighted in the fund itself. Should each stock be assigned a uniform weight, or should their weights be determined by the market-value of the stocks they represent?

The clustering, selection, and allocation processes represent different branches of analysis for our problem. Through further exploration, we hope to determine

which precise set of methodologies will contribute to the most robust and successful fund, and possibly understand the reasoning behind this improved performance. With this knowledge, we could ultimately translate such decisions to other stock groups and problems with similar concerns and questions.

2 Methods

2.1 Covariance Matrices, Expected Returns, and Capitalizations Data

The covariances and expected returns of securities were estimated using the single factor model. In order to do so, it was first necessary to put the CRSP database files into actionable formats, particularly the one providing periodic return and capitalization data for each stock. This was completed using Python and the Pandas library, and produced two files. One file lists each stock's *return* during each period, and the other lists each stock's *capitalization* during each period. From here, the single factor model requires a linear regression to be run for each stock, decomposing observed stock returns into a market component and a residual component. From this we yield an estimation for α, corresponding to the residual component, and β, corresponding to the market component. Using these estimates, along with that for residual error, we construct the covariance matrix and expected returns vector.

2.1.1 Logic of the Single Factor Model

The single-factor model (1) assumes that the returns of an asset can be decomposed into a market component and a residual component. Stated mathematically,

$$r_i = \beta_i r_M + \theta_i \tag{1}$$

where β_i is factor exposure of asset i, where the performance of the market is the factor, and θ_i is the residual, uncorrelated return. For a single period, given a vector of asset returns, $\mathbf{r}$ and the broad market return of that period, r_M, the single factor model assumes the following:

$$\mathbf{r} = \boldsymbol{\beta} r_M + \boldsymbol{\theta} \tag{2}$$

Based on these assumptions, we can estimate the expected returns as follows:

$$\mathbb{E}(\mathbf{r}) = \boldsymbol{\beta}\mathbb{E}(r_M) + \mathbb{E}(\boldsymbol{\theta}) \tag{3}$$

and the covariance matrix $\mathbf{V}$ can be estimated as follows:

$$\mathbf{V} = \sigma_M^2 \boldsymbol{\beta}\boldsymbol{\beta}^{\boldsymbol{T}} + \mathbf{D} \tag{4}$$

for $\sigma_M^2 = Var(r_M)$ and $\mathbf{D} = diag(\omega_1^2, ..., \omega_n^2) = Cov(\theta)$. In the next section, we describe the process of estimating $\mathbb{E}(\boldsymbol{\theta})$, $\mathbf{D}$, $\boldsymbol{\beta}$, and σ_M^2 using historical data.

R Code for Assembling Covariance Matrix

```
# Initialize the diagonal matrix
D_matrix = matrix(0, nrow=num_securities, ncol=num_securities)

# Add the values to the diagonal matrix
for(i in 1:num_securities){
    D_matrix[i,i] = resid_errors[i] ^ 2
}

covariance_matrix = matrix(betas) %*% t(matrix(betas))
covariance_matrix = sigma_m_2 * covariance_matrix
covariance_matrix = covariance_matrix + D_matrix
```

2.1.2 Linear Regressions for Estimation

We utilize historical stock and market returns to estimate the above parameters. We elaborate on the specific lag-times and other estimation practices later, so for now consider an abritrary number of assets, n, and an arbitrary number of periods, T. Then, we have $\mathbf{r}(1)...\mathbf{r}(T)$, where $\mathbf{r}(k) \in \mathbb{R}^n$ is the vector of returns for each asset during period k, for all $1 \leq k \leq T$. We also have the corresponding market returns, $r_M(1)...r_M(T)$.

We then run n linear regressions, one for each stock. More specifically, we utilize each stock's returns against the market return across the T periods. The regression equation for asset i is:

$$r_i = \alpha_i + \beta_i r_M + \epsilon_i \tag{5}$$

where β_i is the beta for asset i, α_i is $\mathbb{E}(\theta_i)$, and ϵ_i corresponds to the residual error (that is, $\text{var}(\epsilon_i) = \text{var}(\theta_i)$). Each regression then produces the corresponding estimates $\hat{\beta}_i$, $\hat{\alpha}_i$, and ω_i. We also estimate $\hat{\sigma_M^2}$ using the historical market return data. With these estimates for each asset, we construct the covariance matrix and expected returns according to the equations above.

R Code for Linear Regression

```
alphas = c()
```

```
betas = c()
resid_errors = c()
num_securities = dim(sec_ret)[1]

for(k in 1:num_securities){
    sec_i_regression = lm(security ~ market, data=sec_i_reg_data)
    # Runs the regression
    reg_sum_i = summary(sec_i_regression)  # Gets regression estimates
    a_i = reg_sum_i$coefficients[1,1]
    b_i = reg_sum_i$coefficients[1,1]
    sigma_i = reg_sum_i$sigma

    alphas = append(alphas, a_i)
    betas = append(betas, b_i)
    resid_errors = append(resid_errors, sigma_i)
}
```

Note that this block is run for each period under consideration for the given covariance matrix and expected return vector.

2.1.3 Data Considered for Input Generation

Given our initial parameter choice of $T = 60$, we estimate the first covariance matrix and expected-return vector based on sixty months of observed returns. Practically, we observe data from January 2017 through December 2021, and generate a corresponding covariance matrix and expected returns vector. We want to replicate the benchmark portfolio over the course of a year (specifically, the year of 2022). We therefore run the selection and allocation algorithms for each period in that year. Given a period in this year–say May 2022–we will use the 60 periods of in-sample data, but it would also be beneficial to utilize more recently observed returns, like those of April 2022 and March 2022. We therefore run the estimation process twelve times, generating model inputs based on 60 periods (for the initial model run) through 71 periods (corresponding to December 2022) of observations.

For the sake of experimentation, we also produced estimations based on fewer periods of historical data. Covariance matrices and expected returns vectors were computed with 60 periods of data (as described above), 30 periods of data, and 12 periods of data. In general, the logic of doing so is to evaluate the impact of considering older data by comparing model outputs. That is, we are interested in knowing if using recent data yields vastly different results, which might indicate change in the market over time; it is possible that using recent data represents the market factor and its impact more accurately.

2.2 Clustering Process

The proceeding sections of the paper are in Python.

We cluster the stocks based on their "distances" from each other. The smaller the distance between two stocks, the greater the similarity in the performance of the stocks with respect to time. We endeavor to group the most similar stocks together. Our motivation for creating these groups is that we ultimately pick one representative stock for each group to include in our index fund per time period.

Before we can begin the clustering process for any given time period, we need to perform some preprocessing on the covariance matrix. We convert the covariance matrix to a correlation matrix; then, convert the correlation matrix to a distance matrix. The distance matrix is the input to our mixed integer program (MIP) used to cluster the stocks.

2.2.1 Correlation Matrix

First, we convert the covariance matrix to a correlation matrix:

```
std_devs = np.sqrt(np.diag(self.covariance_matrix))
return (self.covariance_matrix / np.outer(std_devs, std_devs))
```

The first line calculates the standard deviations of each variable in our data set. It does this by taking the square root (np.sqrt) of the diagonal elements (np.diag) of the covariance matrix. The diagonal elements of the covariance matrix represent the variances of each variable, and the standard deviation is the square root of the variance.

The second line calculates the correlation matrix. It does this by dividing each element of the covariance matrix by the outer product (np.outer) of the standard deviations vector with itself. The outer product of the standard deviations creates a matrix where each element is the product of the standard deviations of the corresponding variables. Dividing the covariance matrix by this matrix scales it to create the correlation matrix.

2.2.2 Distance Matrix

Next, we convert the correlation matrix to a distance matrix:

```
distance_matrix = np.zeros_like(correlation_matrix)
for i in range(correlation_matrix.shape[0]):
    for j in range(correlation_matrix.shape[1]):
        distance_matrix[i, j] = self.distance_bw_vectors(
          correlation_matrix[i], correlation_matrix[j])
```

In our process, we systematically traverse the correlation matrix, row by row and column by column. For each pair of indices, say the i^{th} row and j^{th} column, we extract the corresponding vectors from the correlation matrix. Next, we calculate the distance between these vectors, which we then record in the corresponding position in our distance matrix.

Note that we explore several methods for computing the distance between vectors. By default, we employ the standard Euclidean distance. We discuss the other methods we experimented with in Section 2.4.

2.2.3 Clustering

We wish to partition n stocks into k groups based on their distances from each other. This gives rise to the following MIP (1):

Parameters:
ρ_{ij}: distance between stocks i and j

Variables:
y_j: binary variable indicating whether j is a centroid for $j = 1, \ldots, n$
x_{ij}: binary variable indicating whether i is represented by j

Objective:
$\min \sum_{j=1}^{n} \sum_{i=1}^{n} \rho_{ij} x_{ij}$

Constraints:
$\sum_{j=1}^{n} y_j = k$ (choose k centroids)
$\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i = 1, \ldots, n$ (each object is represented by one centroid)
$x_{ij} \leq y_j \quad \forall i, j = 1, \ldots, n$ (i is represented by j only if j is a centroid)
$\sum_{i=1}^{n} x_{ij} \leq \lfloor n/k \rfloor \quad \forall j = 1, \ldots, n$ (Each centroid has $\lfloor n/k \rfloor$ elements associated with it)
$x_{ij}, y_j \in \{0, 1\} \quad \forall i, j = 1, \ldots, n$ (binary variables)

We solve this MIP using the gurobipy package in Python (2). Then, we do some post-processing based on the values of the y's and x's to determine the clusters which are associated with each cluster point.

2.2.4 Stock Selection

Finally, we pick the largest cap stocks from each cluster. These are the stocks which make up our index for the desired time period.

2.2.5 Valuing the Index

Once the stocks are selected, we need to figure out how much weight each stock should have in the index fund. We calculate these percent values $\sigma(i)$ for stocks $i = 1, \ldots, n$ based on the formula:

$$\sigma(i) = \frac{\text{cap}(i)}{\sum_{j=1}^{n} \text{cap}(j)} \tag{6}$$

Finally, we obtain the value of our portfolio Γ_P using the formula:

$$\Gamma_P = \sum_{i=1}^{n} \sigma(i)\rho(i) \tag{7}$$

where the $\rho(i)$ are the expected returns of the stocks.

2.3 Running Experiments

There are four methodologies/metrics that vary across experimentation: (1) distance metric, (2) stock selection methodology, (3) weighting methodology, and (4) number of groups. Practically, we run the model with different metrics by altering the input permutation of parameters. This permutation is a sequence of four integers, each integer corresponding to an argument and corresponding parameter. This practical implementation is elaborated on below.

The distance metric is used to determine the similarity between stocks, with similar stocks being placed into clusters. For our purposes, we recognize that the technique used for measuring distance may present different implications for the result, and so we implement three different techniques for measuring distance and evaluate each's impact on final performance. The three distance metrics used are: (0) Euclidian distance, (1) Absolute Distance, and (2) Cosine Distance.

The next varied argument is the stock selection method. Within each cluster, we want to select the stock that best represents the cluster and its performance. We test three methods for doing so with respect to a security's expected return. One method is to (0) select the stock with the greatest expected return of the next interval; the second method is to (1) select the stock with the lowest expected return of the next interval; the final method is to (2) select the stock with an expected return closest to the average expected return of the cluster.

Once clustering and selection has been handled, the securities must be allocated to form the index fund. For this, we evaluate two methods that mimic the allocation for the benchmark portfolio. These two methods are (0) capitalization weighting and (1) uniform weighting. A capitalization weighting allocates securities based on the ratio of their capitalization to the total capitalization of all securities in question. For example, if we are interested in building a portfolio based on two stocks, with stock 1 having a capitalization double that of stock 2, we would invest twice as much into stock 1 as stock 2. A uniform weighting assigns an equal portfolio weight to each security. That is, each stock's proportion of the portfolio is equal.

Finally, we vary the number of clusters formed in the clustering process, which in turn causes more stocks to be included in the final index fund. This is equivalent to altering values of k in accordance with the project specification. The numbers of clusters tested are: 5, 10, and 15.

For experimentation, we pass a permutation of integers indicating which of these methodologies are used. What follows are tables detailing which of said integers map to different methods.

i	Distance Metric
0	Euclidian Distance
1	Absolute Distance
2	Cosine Distance

i	Selection Method
0	Highest Expected Return
1	Lowest Expected Return
2	Average Expected Return

i	Allocation Method
0	Capitalization Weighted
1	Uniform Weighted

i	Number of Clusters
5	Five Groups
10	Ten Groups
15	Fifteen Groups

Note that this is a combinatorial approach. There are $3^2 \cdot 2 \cdot 3 = 54$ possible permutations; an example permutation is (0,2,0,10). If we had many possible permutations (in the order of 10^3 or 10^4), we could use the "hitting set" algorithm to reduce these permutations to the ones which produced significantly different results. We felt that the combinatorial approach was the best way for our experiments to be extensible and modular as we were trying to follow good object oriented programming practices.

3 Results

As we explained in the methodology, the novel part of our research was varying the inputs to the clustering process. We sought to answer questions like:

- How do our portfolio performances differ when we train over different lengths of past data?
- What happens if we measure the distance between the entries of the covariance differently?
- What if we select the stocks based on different metrics?
- How does the Stock Allocation method impact the portfolio returns?
- How does changing the number of clusters change the fund's composition and the overall returns?

3.1 Lag Impact on Covariance Matrix

Figure 1 demonstrates the 3 different lag choices for the creation of the covariance matrix. For a lag of 12, we train over the last 12 months, for a lag of 30 the last 30 and so on. Brighter entries of the heatmap correspond to larger entries of the covariance matrix. Each covariance has the largest results concentrated in different areas. Considering that this covariance matrix is sorted, meaning

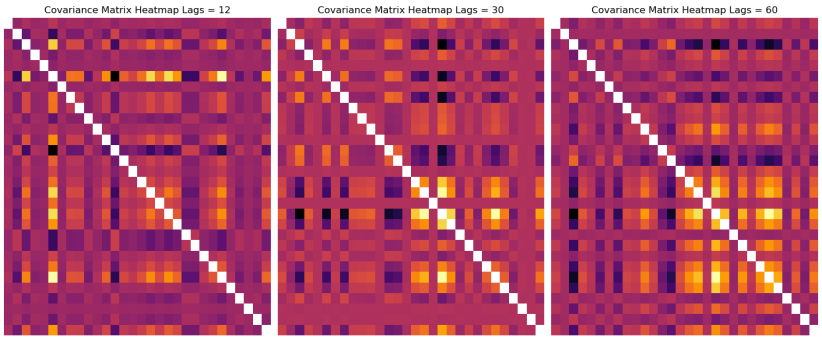

Figure 1: Covariance matrices comparing "Lags" 12, 30, and 60 for the first out-of-sample period.

earlier rows and columns correspond to the largest capitalization stocks in the universe, we see in the lag-60 and lag-30 matrix a concentration of higher values in the lower capitalization stocks. This makes sense when we consider the pandemic, and how these two matrices are the ones created with that data included in its training set. During the pandemic, the largest companies were the ones that were able to maintain production and performance, particularly large tech companies whose products were used for virtual working and learning. Thus, the lower return stocks had more variance in their outputs over the pandemic, and the matrices have higher values in these areas. For the lag-12 matrix, we actually see a concentration of the large variances in some of the larger capitalization stocks particularly.

These two different locations for volatility in the matrices have two different impacts, both uniquely impacting the performance of our clustering method. The 12 lag covariance will likely isolate the high variance stocks with large returns, possibly ignoring them in the clustering process. The lag-30 and 60 covariances will cluster the smaller return stocks together, possibly losing out on some of the diversity in the cluster process because of that. While these are not inherently negative impacts on portfolio performance, considering the role they possibly play can reveal more information about the intricacies of our model.

3.2 Varying Inputs and Training Period

As can be seen in Figure 2, there are many different results in terms of value and expected return in response to the permutations of combinations. These results are for the first period exclusively. Some inputs result in values and volatilities near the benchmarks, while others have drastically larger volatilities or returns. For the 15 clusters, some of the combinations have the same output, because when we have clusters with 2 stocks in each, the average expected return stock is either the largest or the smallest. This is why the third picture has less points,

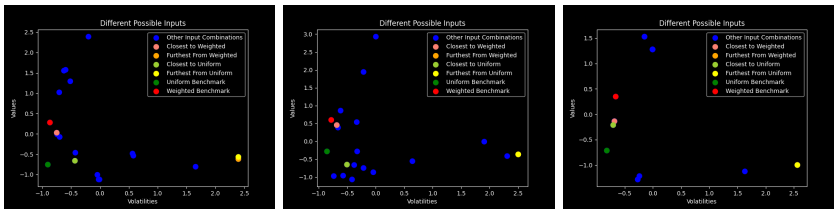

Figure 2: Scatter plots for each of the group sizes (5 groups of 6, 10 groups of 3, 15 groups of 2) displaying the trade-offs between different possible input Permutations with the x-axis "Volatilities" and the y-axis "Values". We include the uniform benchmark and weighted benchmark as baseline results.

as there are less unique clustering processes and portfolios. Also, as can be seen, there are certain portfolios which seem to perform very well, meaning they have very large returns and small variances, but it is unknown if these particular inputs will have continued performance in other intervals.

While this visual is helpful in understanding the layout of our analysis space for a single period, we want the capability of computing and deciding upon benchmarks in a more robust method. Thus, future analyses will focus on both selection of possible inputs which have varying quality of results, i.e. higher portfolio return, less variance in results, etc. And, through analyses of the sharpe ratio of our different portfolios, we can have a measurement of overall best performing inputs for each number of clusters.

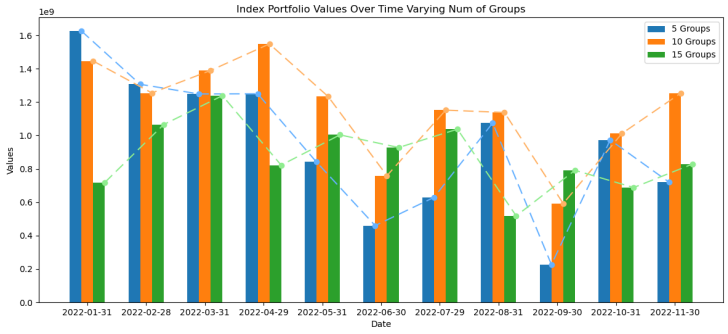

Figure 3: Fluctuation of the index portfolio values over time for each of the group sizes (5 groups of 6, 10 groups of 3, 15 groups of 2) with a fixed lag of 30.

Figure 3 records the portfolio value over time with a fixed lag of 30, further broken down by the number of clusters (aka groups). Recall that we ran tests in which the number of groups $k = 5, 10, \text{and } 15$. We observe that forming 15 groups of stocks (with 2 stocks in each group), the portfolio value over time is the least volatile. That is, it is the most consistent across the periods. Using

5 groups of stocks yielded the most volatile portfolio, although it is also true that the greatest portfolio value was attained using this parameter. Although the performance of the 10-group portfolio was more volatile than that of the 15-group, it delivered relatively high returns more consistently than that of the 5-group. Based on this visualization of portfolio performance under this model configuration and accompanying assumptions, the 10-group method appears to be the ideal balance of risk and return, although practically speaking, risk tolerance is subject to the circumstances of the investor.

3.3 Specific Input Performance Analysis

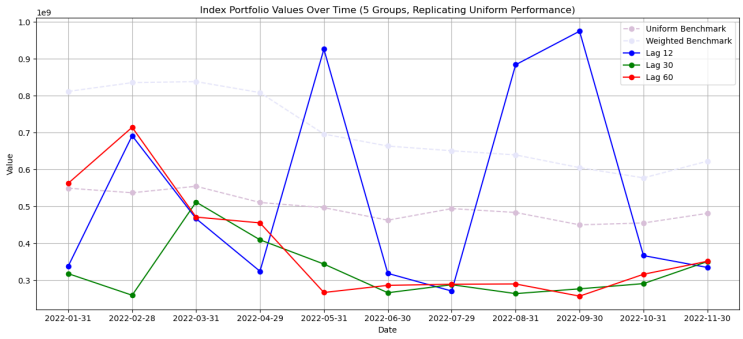

Figure 4: "Bad" Selection: Fluctuation of the index portfolio values over time for each of the lag sizes (12, 30, and 60 periods) with a fixed permutation (2, 0, 0, 5) compared to the uniform and benchmark "Baseline" portfolios.

We now discuss a selection of experiment results with respect to the different lags. Figure 4 displays the portfolio values of the parameter configuration (2, 0, 0, 5). This means that the distance metric was cosine distance, the selection method was highest expected returns, the allocation method was capitalization weighted, and the number of groups was 5. We observe that, for each of the lags (representing different amounts of historical data used to compute the model inputs), our portfolio generally underperforms with respect to both benchmark portfolio configurations. The exceptions are a handful of periods in the lag-12 portfolio, as well as one period in the lag-60 portfolio. However, it is clear that the lag-12 portfolio is highly volatile and except for these few periods delivered relatively poor results. While both the 30 and 60 period-lag portfolios were significantly less volatile, they also underperformed considerably, especially with respect to the capitalization weighted portfolio.

Figure 5 displays our portfolio values under a parameter configuration of (1, 0, 1, 10). This means that the distance metric was absolute distance, the selection method was highest expected return, the allocation method was uniform weighting, and the number of groups was 10. Under this configuration, the lag-

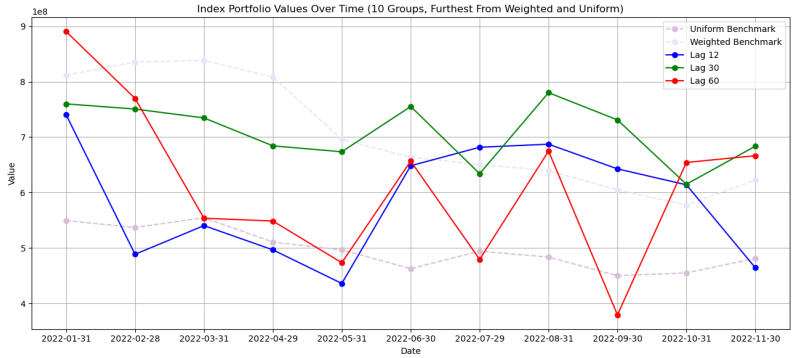

Figure 5: "Medium" Selection: Fluctuation of the index portfolio values over time for each of the lag sizes (12, 30, and 60 periods) with a fixed permutation (1, 0, 1, 10) compared to the uniform and benchmark "Baseline" portfolios.

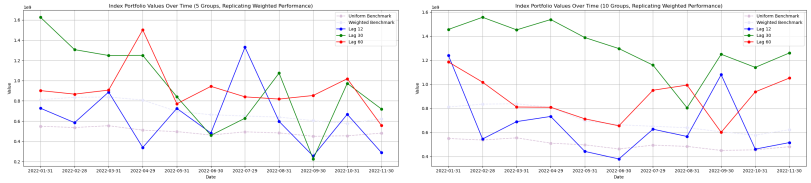

Figure 6: "Good" Selections: Fluctuation of the index portfolio values over time for each of the lag sizes (12, 30, and 60 periods) with fixed permutations (0, 0, 0, 10) and (1, 1, 0, 10) compared to the uniform and benchmark "Baseline" portfolios.

30 portfolio performed the best with respect to the benchmarks, consistently outperforming the other two and the uniform benchmark. Both the lag-12 and lag-60 portfolios were considerably more volatile than the lag-30, but it is worth noting that they generally track with or outperform the uniform benchmark. These results suggest that this configuration yields better performance than that used to produce the results of Figure 4.

Figure 6 displays two experiments we feel to demonstrate good configurations. The figure on the left uses a (1,1,0,10) configuration. Here, we observe that the lag-12 and lag-30 portfolios are more volatile, with the lag-60 being the most stable. The lag-60 also consistently tracks or outperforms both benchmark portfolios. Although the lag-12 peaks and troughs frequently, it is interesting that the lag-30 portfolio experiences such a large drop from initial valuation.

The figure on the right uses a (0, 0, 0, 10) configuration, meaning it uses a Euclidean distance metric, highest expected return selection method, capitalization weighted allocation method, and 10 groups of stocks. This configuration yields particularly good results for lag-60 and lag-30 portfolios, which consis-

tently either replicate or outperform the benchmark portfolio values. The lag-12 portfolio is more volatile, but still tracks the uniform portfolio relatively well, outperforming it occasionally. This configuration appears to be the ideal one based on our analyses, a conclusion that is elaborated on in the discussion section.

3.4 Sharpe Ratio Analysis of Performance

Finally, in an effort to summarize these results and compute the best performing inputs for each number of clusters, we computed the sharpe ratio for our index with all possible combinations of inputs, averaged them across the testing period, and found the inputs which resulted in the largest average sharpe ratio. Thus, this portfolio should, over the testing period, have the highest expected return in relation to its volatility, when weighting the expected return and standard deviation of that return equally.

Figure 7: lag-60 Sharpe Ratio Results

Portfolio Label	Sharpe Ratio	Corresponding Inputs
5 Cluster Index	0.42687	(1, 0, 0)
10 Cluster Index	0.55166	(2, 0, 0)
15 Cluster Index	0.634287	(2, 0, 0)
Uniform Benchmark	0.80000	N/A
Value-Weighted Benchmark	1.11789	N/A

Figure 8: lag-30 Sharpe Ratio Results

Portfolio Label	Sharpe Ratio	Corresponding Inputs
5 Cluster Index	0.39447775	(2, 0, 0)
10 Cluster Index	0.49236164	(1, 0, 0)
15 Cluster Index	0.56819263	(2, 0, 0)

Figure 9: lag-12 Sharpe Ratio Results

Portfolio Label	Sharpe Ratio	Corresponding Inputs
5 Cluster Index	0.44154972	(2, 1, 0)
10 Cluster Index	0.57074568	(2, 1, 0)
15 Cluster Index	0.60859157	(1, 1, 0)

In the table for 60 lags we have the sharpe ratio and inputs which resulted in these inputs, meaning in all cases the best method is to select the stock with the largest expected return for each cluster and to weight the clusters by their

corresponding represented stocks. However, the best distance measures vary, with 5 clusters having the best results with a Manhattan (absolute) distance, while 10 and 15 clusters perform best with a Cosine distance. Also, these are computed using 60 lags, meaning that we are training our model over the last 60 months of returns. We then compute the same values for lag-30 and lag-12 models, with interesting results. Firstly, it seems that the lag-60 results in the overall best performing portfolio, which makes sense with the well known idea that training over the most possible data will lead to the best possible results. Additionally, for the lag-12 model it seems that selecting the stock with the lowest expected return is the best method, which in some ways aligns with our analysis of the covariance matrix earlier, where the higher capitalization stocks have the largest variance for the 12 month training period. These results can apply to these different instances, if only certain amounts of data are available, as interestingly there are some methods which are overall best, like weighting the representative stocks with the returns of the stocks they represent, but there are some differences in certain categories.

4 Implications and Conclusions

This project set out to assess various metrics and methodologies involved in the process of index fund creation. The variation in our results suggests examining such methods is a worthwhile endeavor. Financial institutions and other providers of investment services and products should rigorously examine the effectiveness of these and other methods when creating products, in order to deliver the highest quality products to their clients. Along with stock selection methodology and other fund parameters, financial institutions and investors must make sound decisions regarding their use of historical data and estimation methodologies.

In this project, we demonstrate the impact that different ranges of historical data have on model results. This historical data is used for the estimation of stock covariances and returns, parameters that are essential in the clustering and stock selection processes, as well as calculations pertaining to risk/volatility and fund/portfolio return. Practically, it is sensible to consider different ranges of historical data. The objective of estimation is to capture current market and stock conditions, and the inclusion of too much older data could potentially influence selection and allocation in unproductive ways. Of the results we display here, portfolios allocated using 12-lag data tended to be more volatile and 30-lag portfolios generally provided a balance of return and risk, at least for our first testing period. Although not rigorously studied, it is possible that impacts to the market as a result of COVID-19 countermeasures are best captured by the 30-month data set.

One key parameter we manipulated was the number of stocks in the final fund. Portfolio performance with respect to the different inputs was displayed in Figure 2. Portfolios with 5 stocks tended to be the most volatile, and portfolios with 15 stocks tended to be the least volatile. 15-stock portfolios also yielded

generally lower returns. The 10-stock portfolios appear to provide considerably high returns on a consistent basis–as a task for future research this should be evaluated over a longer period of time. Although the 10-stock portfolio seems to provide the best of both worlds, individual investors have different time-horizons and risk tolerances. If one is interested in maintaining value, for example, a fund composed of more stocks might be beneficial. This, along with other metrics, was only tested over a 12 month period–testing over a longer period of of time would be necessary for financial institutions. Additionally, results from our sharpe ratio experiments indicate that this performance could be dependent on the inputs to our models when forming these graphs, and instead overall highest performing portfolios may occur when creating 15 clusters.

Another tested metric was the allocation method. We replicate the allocation methods used in benchmark portfolios, and find that the capitalization weighting method tends to outperform the uniform weighting method. This is not especially surprising, as the capitalization benchmark portfolio tends to outperform the uniform benchmark portfolio. For clustering and stock selection, it appears that the use of euclidean distance and selection based on highest expected return yield the best results, although again sharpe ratio results suggest this can change depending on lags or number of clusters.

For further research, incorporating multiple factors into parameter estimation would be beneficial and sensible given the financial industry's use of such multi-factor models. The clustering and selection methods tested here are far from exhaustive, and there are also portfolio allocation models that might be combined with stock clustering and selection here. For each period, we examine the thirty largest cap stocks, effectively limiting this index fund to the mega-cap sector in the US. Expanding the universe of stocks to the whole S&P500, for example, or another index containing small or mid-cap securities like the Russell 1000 would be insightful as well.

5 Software

The code for our project can be found at this Github Repository link: `https://github.com/michael-khalfin/notAPonziScheme`. The Repository contains R and Python code. We used the numpy, pandas, and matplotlib packages in Python. Our IDE was Visual Studio Code.

6 Acknowledgements

Thank you Dr. Shiqian Ma for a wonderful semester!

7 References

1. Cornuéjols, G., Peña, J., & Tütüncü, R. (2018). Mixed Integer Programming: Theory and Algorithms. In Optimization Methods in Finance (pp. 90-123, 140-

160). Cambridge: Cambridge University Press. doi:10.1017/9781107297340.009
2. Gurobi Optimization, LLC. (2023). Gurobi Optimizer Reference Manual. Retrieved from `https://www.gurobi.com`