Zurich University of Applied Sciences

School of Management and Law

Bachelor's Thesis

A Python Integration of Practical Asset Allocation Based on Modern Portfolio Theory and Its Advancements

Author:
Philipp Dubach (17-659-863)

Supervisor:
Dr. Norbert Hilber

June 9, 2021

Management Summary

In 1990, Harry Markowitz was awarded with the Nobel Prize for his work on modern portfolio theory. To this day, the mean-variance framework is the preferred method to pick investments for many retail and institutional investors. Meanwhile, big data and the real-time economy have created new challenges and opportunities in the field of asset allocation. These developments are well documented, and portfolio management firms continuously implement the findings into sophisticated models. Nevertheless, only a few comprehensive software models are available publicly to use, study, or modify.

We tackle this issue by engineering practical tools for asset allocation and implementing them in the Python programming language. With its clear syntax, efficient development, and usability, Python provides an ideal framework for this thesis. We turn to convex optimization to formulate specific portfolio optimization problems and incorporate different investment constraints. Even though convex optimization proves to offer a restricted class of optimization problems, its fundamental advantages become apparent throughout this thesis. We consistently examine our problems by solving them analytically or with numerical examples. The focus is to keep the tools simple enough for interested practitioners to understand the underlying theory yet provide adequate numerical solutions. For this reason, we provide code snippets of the accompanying routines as well as valuable visuals to describe the input data and the obtained results.

We extend the original mean-variance model by going beyond the first two moments of the return distribution. Particularly we set up optimization problems with more advanced risk measures such as expected shortfall. We show how estimation errors in practical asset allocation can be reduced by combining the sample covariance matrix with a more structured estimator through a process called shrinkage. The effect of the implemented routines becomes apparent in the out-of-sample optimization results. Additionally, we provide a discussion on methods that did not demonstrate the anticipated improved results or did not meet our standard of efficiency and comprehensibility.

We find that most optimization problems can be expressed in convex form and therefore be implemented and solved efficiently using available Python modules to create portfolios from real-world data. Finally, we demonstrate how even in an environment with high correlation, achieving a competitive return with a lower expected shortfall and lower excess risk than the given benchmark over multiple periods is possible. We underline this through various studies with historical data from the Swiss equity market.

Contents

List of Figures

List of Tables

Abbreviations

ARA	Absolute Risk Aversion
BL	Black-Litterman (Model)
CAPM	Capital Asset Pricing Model
CML	Capital Market Line
COV	Covariance
CVaR	Conditional Value at Risk
ES	Expected Shortfall
ETF	Exchange Traded Fund
MPT	Modern Portfolio Theory
MV	Mean Variance (Model)
MVP	Minimum Variance Portfolio
QQ	Quantile Quantile (Plot)
RP	Risk Premium
SIX	Swiss Sock Exchange
SMI	Swiss Market Index
SPI	Swiss Performance Index
S&P 500	Standard and Poor's 500 (Index)
STD	Standard Deviation
TR	Total Return
VAR	Variance
VaR	Value at Risk

Notation

t	Generic time
T	Time horizon
W	Wealth; Investor's budget
$P_i(t)$	Price of asset i
r_f	Risk free rate of return
r_i	Linear return of asset i
R_i	Compounded return of asset i
$E(r_i)$	Expected return of asset i
σ_i^2	Variance of returns of asset i
σ_i	Standard deviation of returns of asset i
V	Value of the portfolio
k	Number of securities
w_i	Relative weight of asset i
ω	Vector of all portfolio weights
w^*	Optimal vector of all portfolio weights
M	Vector of all expected returns
Σ	Covariance matrix
S	Sample covariance Matrix
F	Structured estimator of the covariance matrix
$\mathbb{1}_m$	Matrix of ones
δ	Shrinkage constant
U	Investors Utility (function)
A	Investors Risk Aversion (function)
γ	Arrow-Pratt risk aversion parameter (Lagrange Coefficient)
γ^*	Optimal arrow-Pratt risk aversion
∇	Gradient operator
λ	Lagrange multiplier
β	Confidence level
α	Portfolio VaR

1 Introduction

Every investor, be it a trader, a mutual fund, or a private investor, has to decide how to allocate their given resources. Even the unconscious decision of holding a cash-only portfolio is ultimately an investment decision. While one can rely solely on his or her intuition, follow trends, or take advice from other parties, institutional investors are most often bound by regulations, prospectus, and the need to generate excess returns compared to a given benchmark. The key is to not only choose one fitting asset but find a combination of securities that complement each other. The practice of spreading among different investments to reduce risk is known as diversification.

Asset Allocation involves assigning different weights (portions of wealth) to different asset categories such as stocks, bonds, or cash. While doing so, the investors are looking for a combination that best suits their needs in an environment they cannot predict. To determine the optimum allocation, one has to estimate, assess, model, and manage uncertainty. Since this is no trivial task with little to no given variables, the investor must rely on quantities and function estimates.

One of the best-known optimization models in finance is the portfolio selection model developed by Harry Markowitz [Mar52] which forms the foundation of modern portfolio theory. The mean-variance approach by Markowitz led to significant developments in financial economics such as Tobin's [Tob58] mutual fund separation theorem and Sharpe's [Sha64] Capital Asset Pricing Model (CAPM). Markowitz was awarded the 1990 Nobel Prize for Economics for the enormous influence on both theory and practice.

For this reason, there have been many advances and modifications of the original model in the past decades. While the theoretical findings are well documented and portfolio management firms continuously implement these findings into sophisticated models, only a few comprehensive software models are available publicly to use, study, or modify. For this reason, this thesis aims to engineer and implement practical tools for asset allocation with Python. The tools have to be simple enough for interested practitioners to understand the underlying theory and sufficient enough to provide proper numerical solutions. We answer the question of how to apply the principles of portfolio theory and the mathematics of convex optimization efficiently and comprehensibly in practical asset allocation as follows.

1.1 Structure, Focus, and Methods

In Section 2 of this thesis, we establish the building blocks needed to model the securities market. Section 3 presents the mean-variance framework introduced by Markowitz [Mar52] describing the basic principles of portfolio optimization. We extend the model by adding constraints to different aspects of the portfolio. Further, we develop a measure of the investor's objectives and an understanding of optimality. In Section 4, we turn to mathematical optimization, namely convex optimization, to formulate specific asset allocation problems. In the first part, we work with analytical solutions where possible. We then apply the principles in Section 5 as we implement the problems with Python to solve them numerically. Mainly because it is not possible to determine analytical solutions to most non-trivial optimization problems. Moreover, we can use historical time series to see how the optimized portfolios perform in empirical backtesting. Python offers an ideal environment to both optimize, backtest and visualize our methodology and results. Section 6 and 7 provide a discussion on the results with a focus on the Swiss stock market.

2 Asset Allocation

We consider a financial market where N assets $i = 1, 2, ..., N$ are traded. Typical assets are common stocks, bonds, or domestic and foreign cash. Generally, the term "asset" can be associated with any financial instrument that can be bought or sold.

2.1 Asset Prices and Returns

Every asset is described by its return as a profit on an investment over a defined period of time, in proportion to the original investment.

Let $P_i(t)$ be the Price of an asset i at time t and $P_i(t-1)$ the respective price at time $t-1$. Since stock prices can only take a certain set of values the variable $P_i(t)$ is a positive, discrete random variable. The linear return is defined as:

$$r_i = \frac{P_i(t)}{P_i(t-1)} - 1 \tag{2.1}$$

Because the distribution of linear returns is not symmetrical, in finance many practitioners use the logarithmic or *compounded return* which we define as:

$$R_i = \ln(r_i + 1) = \ln\left(\frac{P_i(t)}{P_i(t-1)}\right) \tag{2.2}$$

Unlike the linear returns, the distribution of the compounded returns, considering them independent and identically distributed random variables, can be easily projected to any horizon. A broader discussion on this topic is to be found in Chapter 6 of this thesis. For returns that are close to zero, the compounded return is almost identical to the linear return. This is congruent with what was found when solving the problems in this thesis for linear and logarithmic returns; the difference in results was minimal. Nevertheless, as Hudson and Gregoriou [HG15] find, in the context of investigations into the terminal wealth of investors, it seems clear that simple return is the most appropriate measure to use. Apart from time aggregation or otherwise mentioned, we will work with linear returns in this thesis.

2.2 Distribution of Returns

Before we can estimate or forecast the expected return we have to look at the distribution of r_i. The most straightforward way to describe a random variable's distribution is by using the probability density function. Intuitively, the function shows a peak where the measurement result is most likely to occur. Formally the probability density function is defined as follows:

$$\mathbb{P}[X \in a, b] = \int_a^b f_X(x)\, dx \tag{2.3}$$

Whereas the probability $\mathbb{P}$ is a measurement of the likelihood that X takes place in an interval $[a, b]$. Most academic finance theory, including Sharpe's [Sha64] CAPM and the Black–Scholes [BS73] model for option pricing, rests on the assumption that stock returns are normally distributed. The normal distribution with its bell shape is the most widely used distribution. It is characterized by two parameters μ and σ^2. The parameter μ is a location parameter known as the expected value. The dispersion parameter σ^2 is known as the variance. The following notation is used to indicate that X is normally distributed:

$$X \sim N(\mu, \sigma^2) \tag{2.4}$$

To test if the normal distribution is an adequate representation of stock returns, we plot a histogram showing the monthly returns of the Swiss Performance Index SPI[1] for 18 consecutive years in Figure 2.1. On top of the Histogram, we overlay the theoretical normal distribution. As expected, it looks approximately normally distributed with some breakouts at both ends. These extreme events that occur more frequently than assumed by the normal distribution are known as "fat tails".

According to Newbold et al. [NCT13], one of the most straightforward ways to test for normality is to use a QQ probability plot. With this graphical method, we compare the actual distribution with the theoretical (normal) distribution by plotting their quantiles against each other. This yields a similar result in Figure 2.2. For most values, the normal assumption holds. However, the more we move towards the ends, the greater the deviation becomes. Also, the SPI returns are distributed skinnier in its center compared to the normal distribution. The same can be observed when analyzing other indices, sectors, or asset classes. It also has to be noted that the normality assumption is more accurate for longer intervals. The distribution of monthly returns is closer resembled by the normal

[1]The SPI is a total-return index that tracks the performance of more than 200 of the largest Swiss companies.

distribution than weekly or daily returns. The discussion in Section 6.1 compares different intervals and gives more insight into how returns are distributed.

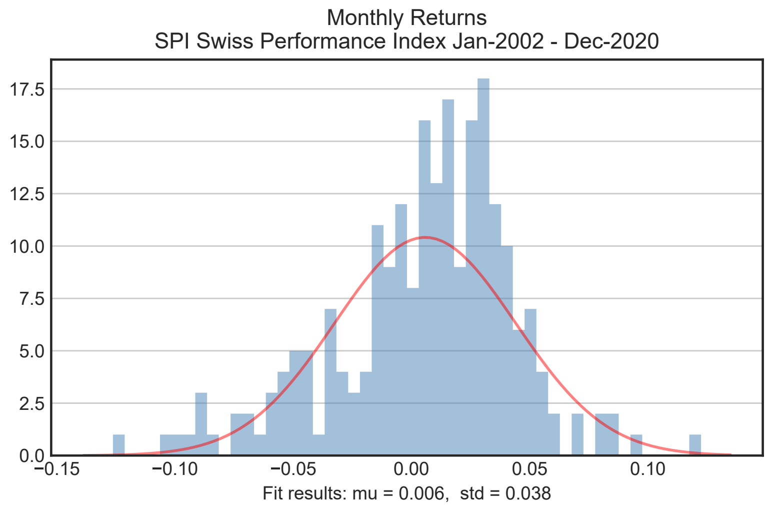


Figure 2.1: Histogram of normalized actual return distribution with normal distribution overlaid (Data: Thomson Reuters).

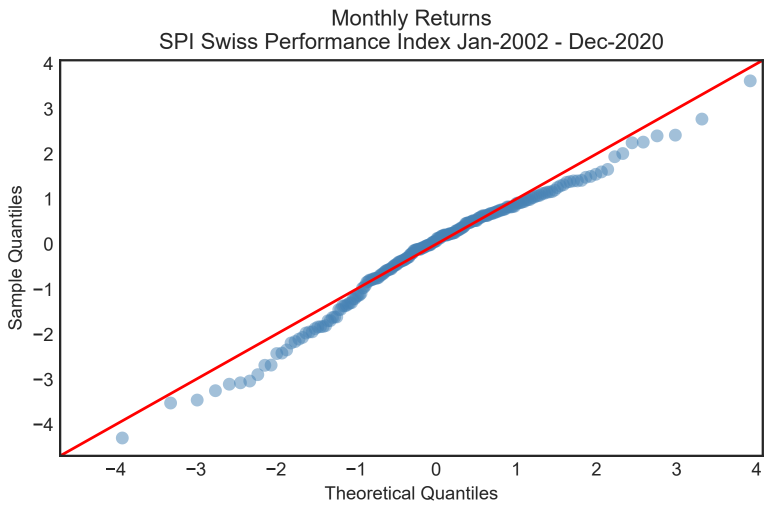


Figure 2.2: Normality testing QQ Plot (Data: Thomson Reuters).

Given these properties, Peiró and Amado [Pei94] detail different alternative distributions. Nevertheless, mean-variance models still rely on the normality assumption for different, mostly practical, reasons. One of them is that the normal distribution is a good enough representation of the actual distribution. Another reason is that the sum of two normally distributed random variables is also normally distributed [BGP20], and it is characterized by only two parameters that are easy to estimate. In this thesis, we will therefore assume that asset returns are distributed normally, if not otherwise stated.

2.3 Estimating Returns

As for any model, the output is only as good as its input. Therefore, much of the effort should go into composing the anticipations or estimates, especially of the expected return $E(r)$. A good way to approach the expected return is to create a list of possible scenarios s and specify both the probability of each scenario and the associated return. Then the expected return $E(r_i)$ for asset i is defined as:

$$E(r_i) = \overline{\mu}_i = \sum_{s=1}^{S} r_i(s)\mathbb{P}(r_i(s)) \tag{2.5}$$

The notation indicates that the summation extends over all possible scenarios.

Since neither this thesis nor Markowitz [Mar52] himself in his introduction of the mean-variance framework goes into detail on how these anticipated returns should be decided upon, we consider the estimates as given. The most straightforward way to do so is by using the arithmetic mean of historic returns:

$$E(r_i) = \overline{\mu}_i = \frac{\sum\limits_{t=1}^{T} r_i(t)}{T} \tag{2.6}$$

According to Meucci [Meu05, p. 102], sample estimates only make sense if the quantities to estimate are market invariants. That is if they display the same statistical behavior independently across different periods. In equity-like securities, the returns are approximately invariant. This is why the mean-variance approach is usually set in terms of returns.

The sample mean $\overline{\mu}$ can be considered a good estimator of the population mean μ if the number of provided historic single-period returns is large.

2.4 Risk

Financial theory refers to risk as the degree of uncertainty or the chance that an outcome will differ from what the investor expected. In reality, most investors are only concerned about downside risk, meaning that the outcome is worse than expected. Even though the variance is a symmetric measure, the mean-variance framework and many other financial models consider variance one of the main risk measures. Mathematically the expectation of the squared deviations about the mean $(X - \overline{\mu})^2$ known as the variance σ^2 is given by:

$$\sigma^2 = E[(X - \overline{\mu})^2] \tag{2.7}$$

Equivalent to the mean of historic returns (2.6) we calculate the historic standard deviation σ_i^2 as a proxy of the deviation of the return r_i from its mean μ_i as follows:

$$\sigma_i^2 = \frac{\sum_{t=1}^{T} \left(r_i(t) - \mu_i \right)^2}{T} \tag{2.8}$$

Again, this is only true if we assume that the sample mean is a good estimator of the population mean. The standard deviation σ is the positive square root of the variance. An asset that is considered to be risk-free therefore has a standard deviation of zero.

2.5 Portfolio definition

A portfolio is a distribution of a given amount of initial capital across a combination of different assets. The assets do not necessarily have to be of different asset classes. In financial theory, a distinction is made between risky assets and risk-free assets (see Section 2.4). In this thesis, if not otherwise stated, all assets are considered to be associated with some risk. Furthermore, we consider all portfolios as self-financing, which means that there is no addition or withdrawal of funds over the time of observation. The acquisition of a new asset must be funded through the sale of an existing asset.

2.5.1 Portfolio weights

We consider a portfolio with n ($n \in N$) different assets. The Value $V_i(t)$ invested in asset i at time t can be expressed as:

$$V_i(t) = k_i P_i(t) \tag{2.9}$$

Where k_i is the number of securities of the i-th asset. The Value $V(t)$ invested in the total Portfolio at time t is given by:

$$V(t) = \sum_{i=1}^{n} k_i P_i(t) = \sum_{i=1}^{n} V_i(t) \tag{2.10}$$

From (2.9) an (2.10) we can now express the relative weight $w_i(t)$ of asset i at time t:

$$w_i(t) = \frac{k_i P_i(t)}{\sum_{i=1}^{n} k_i P_i(t)} = \frac{V_i(t)}{V(t)} \tag{2.11}$$

With no further constraints (see Section 3.3) such as limits on leverage or short selling, the relative weight $w_i(t)$ of an asset can theoretically take any value in the space of real numbers $\mathbb{R}$. It can be seen in (2.11) that the relative weights, all other things being equal, changes over time with the associated asset price. In this Thesis weights are always understood as relative weights. Note, that in the Markowitz [Mar52] approach to portfolio creation the relative weights always have sum up to one:

$$w(t) = \sum_{i=1}^{n} w_i(t) = 1 \tag{2.12}$$

This requirement must be taken into account when formulating the constrained optimization problem in Section 4.2.

2.5.2 Expectations and Covariances of Returns

From (2.6) and (2.11) we now define the expected return on the portfolio as:

$$E(r) = \mu = \sum_{i=1}^{n} w_i E\big(r_i(t)\big) = \sum_{i=1}^{n} w_i \mu_i \tag{2.13}$$

The variance of the portfolio return, is calculated as follows:

$$\begin{aligned}
\sigma^2 &= E\Big[(r_i - \mu_i)^2\Big] \\
&= E\Big[\big(\sum_{i=1}^{n} w_i(r_i - \mu_i)\big)^2\Big] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} E[w_i w_j (r_i - \mu_i)(r_j - \mu_j)] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j covar(r_i, r_j) \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \sigma_{i,j}
\end{aligned} \tag{2.14}$$

2.5.3 Matrix Notation

Matrices provide handy ways to organize data sets together; transforming and modifying them becomes much more straightforward than working with each matrix constituent separately.

From (2.11) we denote $\omega = [w_1, w_2 \ldots, w_n]^T$ a one-column vector containing all portfolio weights. Similarly $M = [\mu_1, \mu_1 \ldots, \mu_n]^T$ contains all the expected returns arranged into a one-column vector. The covariance Matrix is setup as:

$$\Sigma = [\sigma_{i,j}] = \begin{bmatrix} \sigma_{11} & \dots & \dots & \sigma_{1n} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \sigma_{n1} & \dots & \dots & \sigma_{nn} \end{bmatrix} \tag{2.15}$$

Note that the diagonal elements $\Sigma = [\sigma_{i,i}]$ of the covariance matrix are the variances of returns.

The portfolio then has an expected return:

$$E(r) = \omega^T M \tag{2.16}$$

With a standard deviation:

$$\sigma = \sqrt{\omega^T \Sigma \omega} \tag{2.17}$$

With (2.16) and (2.17) we have established the two main parameters of the mean-variance framework.

3 Mean-Variance Framework

The mean-variance framework pioneered by Markowitz [Mar52] is the most common approach to asset allocation, in which the investor seeks to optimize the portfolio's expected return for a given degree of variance and a given set of investment constraints. Under the assumptions made in Chapter 2, it is possible to estimate the market parameters that feed the model and then solve the resulting optimization problem. The underlying assumptions to the model are: (1) All assets are arbitrarily interchangeable at any time. (2) Investors consider an equal time horizon. (3) Prices reflect all available information. (4) Investors base their decisions on expected return and risk.

3.1 The general Approach

The observation that diversification can reduce risk is fundamental to any portfolio optimization model. By incorporating a combination of risky assets with different correlations, the portfolio's overall risk can be reduced. In Figure 3.1 the pairwise correlation of

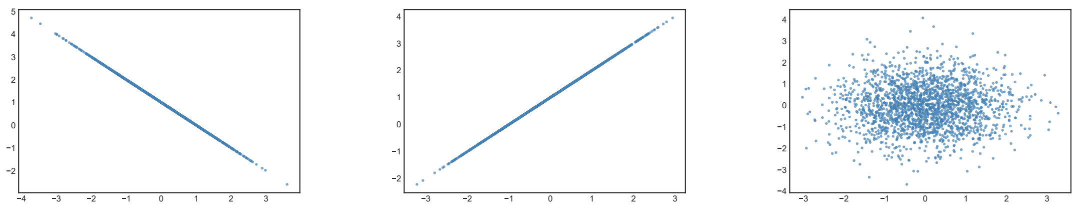


Figure 3.1: Strong negative correlation, strong positive correlation and uncorrelated returns (from L to R).

different sets with differently correlated returns can be seen. In theory, if the returns show perfect negative correlation, perfect diversification could be achieved. This would mean that the variance of returns, the measure of risk for the investor, would be zero. On the other hand, if all assets were perfectly independent, the risk would tend toward zero as more and more assets were added. In reality, asset returns are neither perfectly correlated nor perfectly independent. As noted by William Sharpe in his work on the Capital Asset Pricing Model [Sha64, p. 439], risk can be reduced by adding more assets to a portfolio, but only to the limit of the average covariance. Average covariance therefore becomes a measure of the market risk which cannot be reduced by diversification. It is also true that the mitigation of risk generally leads to lower expected returns. Nevertheless, a look at the possible combinations of assets reveals that some portfolios dominate others in terms of the expected return for a level of standard deviation.

```python
def random_portfolio(timeseries):
    #Return Vector (3x1)
    M = pd.DataFrame(timeseries.mean())
    #Weight Vector (3x1)
    w = pd.DataFrame(gen_weights(timeseries.shape[1]),
    index=['A', 'B', 'C'])
    #CovMatrix
    S = pd.DataFrame(timeseries.cov())

    mu = w.T@M
    sigma = np.sqrt(w.T@S@w)

    return mu, sigma, w

for _ in range(300):
      sim = sim.append(pd.DataFrame(np.concatenate(
      random_portfolio(df_returns)).T,
      columns=['Mean', 'Std', 'wA', 'wB', 'wC'])
      )
```

Listing 3.1: Python code generating a set of 300 portfolios with random weights for three random arbitrary risky assets A, B and C.

The Python code in Listing 3.1 lets us generate and visualize a set of 300 portfolios consisting of three arbitrary assets with a random proportional weight assigned to each asset. As we plot the means and standard deviations of the portfolios generated, in Figure 3.2 the described effect can be seen clearly. It is observable that there are different portfolios with equal means and different standard deviations and vice versa. Furthermore, there seems to be an invisible boundary on both ends of the mean distribution. The top boundary of this area, which looks like it is part of a hyperbola is called ”the efficient frontier”. All portfolios on the efficient frontier are Pareto optimal, meaning that no other combination of assets yields a higher return for a given amount of risk. On the contrary, the lower part of the frontier contains all inefficient portfolios, meaning that another combination of the same assets yields a higher return for the same amount of risk. In this thesis, we adjust the weights of assets in a portfolio to achieve such Pareto efficiency.

A special case of an optimal portfolio is the minimum variance portfolio, located at the global minima of the hyperbola surrounding all portfolios (red dot in Figure 3.2). This portfolio is not particularly desirable; it is merely characterized by the fact that it is the single portfolio with the lowest variance. According to Merton’s Analytic Derivation of

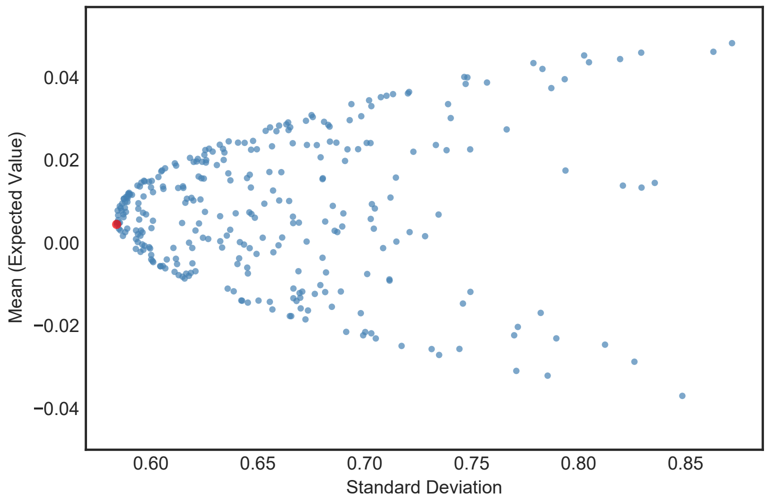


Figure 3.2: Plot of standard deviation and mean as a measure of risk and return for the generated portfolios.

the Efficient Portfolio Frontier [Mer72] the minimum variance portfolio marks the beginning of the efficient frontier. The portfolio with the highest return marks the end of the frontier. This portfolio naturally only consists of the one single asset with the highest return.

3.2 Utility

Utility is a measure of how much happiness someone derives from something. We introduce this concept here because an investor's actions have direct effects on wealth which in turn affects utility. Before we can approach utility, we have to define the investor's objective. Meucci [Meu05, p. 239] differentiates between the following three common objectives: Relative wealth, and net profits, absolute wealth. Relative wealth is an objective sought after, for example, by fund managers who evaluate their performance against a benchmark over a specific horizon. Opposite, traders who focus on their daily profit and loss most likely have a net profit objective for their time horizon, which is a single trading day. Most investors nevertheless focus on the value at the horizon of the portfolio and therefore have an absolute wealth objective. Every investor can have one or more objectives. If multiple objectives exist, it has to be analyzed if an order exists such that one stochastically dominates or is dominated by another. In this thesis, we consider only investors who have a single, absolute wealth objective.

As mentioned before, a utility function describes how much happiness someone derives from a generic outcome regarding their objective. Therefore the expected utility can be calculated by weighting the utility from every possible outcome by the probability of that outcome. This is the Von Neumann-Morgenstern [VM44] specification of expected utility. Further, it states that people (investors) have preferences. When the investor has to choose, he or she can tell whether portfolio A is preferred to $B, U(A) > U(B)$; is indifferent between A and $B, U(A) = U(B)$; or prefers B to $A, U(A) < U(B)$. Those choices are transitive: If someone prefers A to B and prefers B to C, it follows that portfolio A is preferred to C. Portfolios with equal utility are equally desirable. From the assessment of these preferences, the individual utility function can be formulated. If an investor gets positive marginal utility from taking on more risk, he or she is called risk-seeking. Positive but diminishing marginal utility is one of the characteristics in most models. The marginal utility is diminishing because as the investor takes on more and more risk, he or she enjoys each additional unit less.

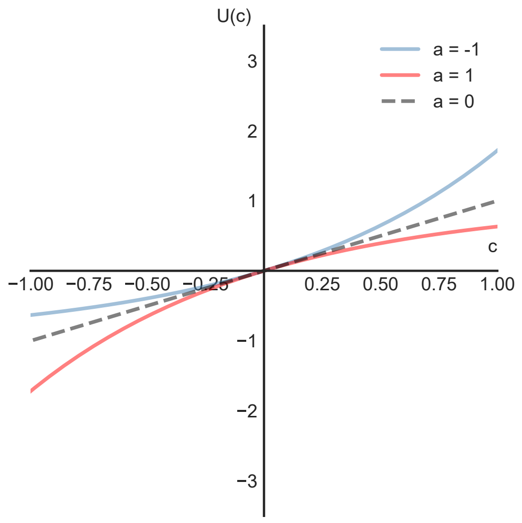


Figure 3.3: Exponential Utility Function for risk-averse (a > 0), risk-neutral (a = 0) and risk-seeking investors (a < 0).

As it can be seen, utility is still an abstract concept, and there is no exact measure even in ex post-analysis. Therefore, much of the research, such as prospect theory developed by Kahneman and Tversky [KT79] is based on results from controlled studies. This thesis does not go into further detail on how the individual utility function is determined. For the sake of comprehension, we consider an exponential utility function with a being a constant describing the investor's risk profile and c the variable that the investor prefers more of, such as consumption.

$$U(c) = \left\{ \begin{array}{ll} (1 - e^{-ac})/a & a \neq 0 \\ c & a = 0 \end{array} \right\} \tag{3.1}$$

In financial literature, this variable is often denoted as W since the utility U depends on the investor's wealth. Other types of utility functions, such as isoelastic utility or power utility functions, are considered more realistic since they exhibit decreasing absolute risk aversion. Yet, due to its concavity and the relatively simple mathematical form, the quadratic utility function as a simplifying assumption has prevailed in most academic literature. Figure 3.3 displays the utility function (3.1) for different risk profiles. It can be seen that the risk-averse investor prefers the portfolio that has less risk. In contrast, the risk-seeking investor chooses the investment with more risk since he or she gains utility from increased risk. The Risk-neutral investor is indifferent among the investments as long as their expected returns are the same. This shows how the investor's utility function influences the investment decisions.

3.2.1 Absolute risk aversion

Since the second derivative of the utility function is not invariant to positive linear transformations of the utility function this cannot be used as local measure of risk aversion. Therefore Arrow [Arr71] and Pratt [Pra64] came up with a measure of risk-aversion that would remain the same even after an affine transformation of the utility function:

$$A(c) = -\frac{U''(c)}{U'(c)} \tag{3.2}$$

The negative sign in front of the division of the second derivative by the first derivative ensures that a larger number indicates a more risk-averse investor. This also implies that the Arrow-Pratt risk aversion is positive only if the investor is locally risk-averse. We will use a risk aversion parameter similar to $A(c)$ to scale the relative importance of the estimated return and the estimated risk when formulating the optimization problem in Chapter 4.

3.3 Constraints

To determine the optimal allocation, the investor's constraints have to be known first. Constraints can arise for many different reasons. They can be imposed by law or corporate policies in the case of institutional investors. Other times they are chosen voluntarily to avoid certain undesirable portfolios. A constraint that was already introduced (2.12) is that in the Markowitz style portfolio, all weights have to sum up to one. Following we list some of the most common constraints.

3.3.1 No-hold constraints

Constraint that forbids the holding of a specific asset i. This can be extended to multiple assets or a class of assets:

$$w_i = 0 \tag{3.3}$$

3.3.2 Long/short constraints

Short sales can be achieved by borrowing an asset and then selling it without actually owning it. This constraint ensures that only long positions are entered. Meaning that assets cannot be sold if the investor does not own them:

$$w_i \geq 0 \tag{3.4}$$

3.3.3 Budget constraints

The initial Value of the Portfolio V_0 (2.10) cannot exceed the Budget as in initial wealth W_0 provided by the investor:

$$V_0 \leq W_0 \tag{3.5}$$

3.3.4 Concentration constraints

The holding of each asset i can be limited to a maximum fraction of the portfolio:

$$w_i \leq w_{max} \tag{3.6}$$

Another form of this constraint is to limit the weight of the sum of the K largest positions:

$$\sum_{i=1}^{K} w_i \leq w_{max} \tag{3.7}$$

3.3.5 Leverage constraints

We measure leverage L in excess of one. The leverage can therefore be limited with the constraint:

$$\sum_{i=1}^{n} |w_i| - 1 \leq L_{max} \tag{3.8}$$

Note that to implement a leverage constraint effectively, the base constraint (2.12) has to be lifted. Also, if a budget constraint (3.5) is already implemented, the leverage constraint does not affect the optimization since the budget constraint already implies zero leverage.

Furthermore, the usage of leverage presupposes that lending and borrowing are available without any restrictions.

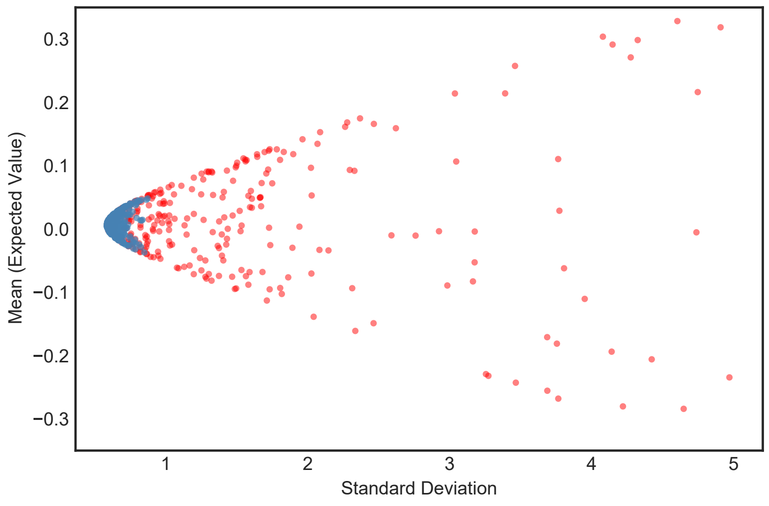


Figure 3.4: Possible random portfolios with (blue) and without (red) constraints

The above-listed constraints are not exhaustive. Every constraint restricts the choice of possible feasible portfolios. This also limits the number of feasible optimal portfolios, which means that an unconstrained optimization might yield a portfolio that better fits the investor's profile. This has to be kept in mind when implementing constraints. To visualize the effect of constraints we again generate several random portfolios in Figure 3.4. For the set of blue portfolios only positive weights (long-only) were considered. For the set of red portfolios this constraint was not applied. While the constraints mentioned above all limit holdings, there are also constraints limiting trading which are especially important in multi-period models. Such constraints can impose turnover limits, limits relative to trading volume, no-buy, no-sell, or trade restriction.

4 Optimization

In the previous chapters we have collected information on the market and projected them into the future by estimating the anticipated returns and their variances for individual assets as well as on the portfolio level. Furthermore, we have collected information on the investor by defining the utility function and therefore setting a measure for risk aversion. We are now setting up the function to be optimized (either minimized or maximized) to derive the optimal portfolio, given these inputs. Mathematically, an optimization problem has the following form:

$$\begin{aligned} &\text{minimize} && f_0(x) \\ &\text{subject to} && f_i(x) \geq b_i, \quad i = 1, ..., m \end{aligned} \tag{4.1}$$

The vector $x = (x_1, ..., x_n)$ is the optimization variable of the problem, the function f_0 is the objective function, the functions f_i are the (inequality) constraint functions, and the constants $b_1, ..., b_m$ are the limits, or bounds, for the constraints. For a minimization problem, a vector x^* is called optimal, or a solution of the problem (4.1), if it has the smallest objective value among all vectors that satisfy the constraints. Characterized by specific forms of the objective and constraint functions optimization problems are organized into different classes.

As we will find, general optimization is difficult, in some extreme cases even impossible. This for many reasons: First, the number of feasible candidates grows exponentially in the search space. Secondly, the search algorithms often get trapped in local minima or saddle points. In general, complex feasible regions are complicated to navigate [BV04, p. 128]. An exception is when the objective function $f(x)$ is convex, and the feasible region satisfying all constraints is a convex set. Only then, local extrema are also guaranteed to be global extrema, which allows efficient algorithmic solving [BV04, p. 8].

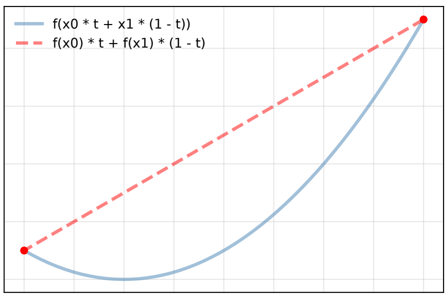
Least-squares problems are optimization problems with no constraints and an objective which is a sum of squares. A solution can be found by solving a set of linear equations. With linear programming, we have another class in which the objective and all constraint functions are linear. Compared to the least-squares problems, there is no simple analytical formula for the solution. Convex optimization problems are of the same form as (4.1) where the functions $f_0, ..., f_m$ are convex. Meaning that the objective function, as well as all the inequality constraints, have to be convex. As with linear programming, there is often no analytical formula for the solution of convex optimization problems. Nevertheless, there are certain algorithms such as interior-point methods [Wri04] that can solve these problems in a small number of iterations. As we will see, the convex problem can be solved efficiently and reliably using available software.

4.1 Convex Optimization

According to Boyd et al. [BV04] the first and most challenging step in using convex optimization is in recognizing and formulating the problem. A function $f(x)$ is convex if:

$$f(x_0 t + x_1(1-t)) \leq f(x_0)t + f(x_1)(1-t), \quad t \in [0,1] \tag{4.2}$$

As it can be seen in the plot on the left of Figure 4.1 the inequality (4.2) signifies that the line segment between $[x_0, f(x_0)]$ and $[x_1, f(x_1)]$, has to lie above the graph of the function. A function f is strictly convex if (4.2) holds whenever $x_0 \neq x_1$ and $0 < t < 0$. A function f is concave if $-f$ is convex. Because of the definition of convexity, convex



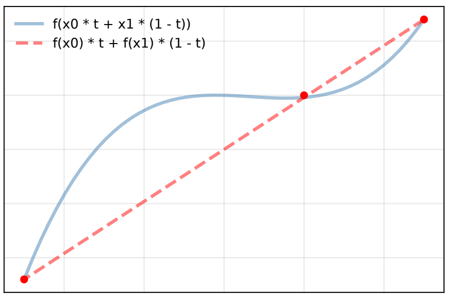


Figure 4.1: For a Convex function (left) the line segment (between any two points) lies above the graph. This property is not fulfilled by the non-convex function (right).

optimization is a restricted class of optimization problems. Nevertheless, it is an instrumental class that offers many advantages that justify its heavy use in different kinds of applications such as machine learning, control, signal and image processing, networking, and many more.

For one, solutions to convex problems guarantee that the found local extrema are also global extrema. Further, it offers good ways to analyze the sensitivity of the solution to small changes to the constraints. As we will see in Chapter 5, there are already many high-quality solvers available to tackle convex problems. Also are ready to use convex modeling languages, which transform mathematical syntax into solver language just as a compiler automates the translation into machine language.

4.2 Constrained Optimization

The Markowitz [Mar70, p. 175] optimization problem can be formulated in three different ways which lead to equivalent results. The following formulation yields the portfolio with the least variance for a given target (mean) return r_P, desired by the investor:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} w^T \Sigma w \\ \text{subject to} \quad & w^T M = r_P \\ & w^T \mathbb{1}_m = 1 \end{aligned} \tag{4.3}$$

In Section 3.3 different constraints were introduced. We are now applying them to our objective function. The first constraint ensures that the achieved return is equivalent to the desired target return. The second constraint is the base constraint (2.12) that applies to all Markowitz style optimization problems. The constant $\frac{1}{2}$ is added for convenience in calculation and does in no way affect the solution. For the second variation of this problem the return is being maximized given an upper limit on the variance σ_P^2 of the portfolio:

$$\begin{aligned} \text{maximize} \quad & w^T M \\ \text{subject to} \quad & w^T \Sigma w \leq \sigma_P^2 \\ & w^T \mathbb{1}_m = 1 \end{aligned} \tag{4.4}$$

If the first constraint would be voided, the optimization would logically lead the investor to invest all his or her wealth $w_n = 1$ in the one asset with the highest return, leaving all other possible assets without an investment.

None of the previous mentioned formulation can directly reflect the investor's objective to put weights on the conflicting interest of minimal risk and maximum return. Only by adding a scalar variable γ gauging the trade-off between risk and return, the risk-adjusted return can be calculated by solving the following quadratic program:

$$\begin{aligned} \text{maximize} \quad & w^T M - \frac{\gamma}{2} w^T \Sigma w \\ \text{subject to} \quad & w^T \mathbb{1}_m = 1 \end{aligned} \tag{4.5}$$

The Lagrange coefficient $\gamma \geq 0$ that solves (4.5) can be interpreted as the Arrow-Pratt risk aversion (3.2) and is essentially a scalar that puts more weight on risk as the risk aversion index grows. As Meucci [Meu05, p. 339] points out, it would be incorrect to consider the risk aversion γ^* as a feature of the investor that is independent of the market. The opposite is true since the same investor shows a different risk aversion when facing different markets. Therefore, the Problem (4.5) has to be solved in a two-step process. First $w^*(\gamma)$ the curve on which the optimal allocation lies has to be calculated. From there, a one-dimensional search for the optimal level of risk aversion γ^* can be performed according to the investor's objective. If the investor's objective is to achieve a certain level of risk (aversion), which might have been predetermined through psychological analysis (see Section 3.2), only then $\gamma^* = \gamma$ and the second step becomes obsolete.

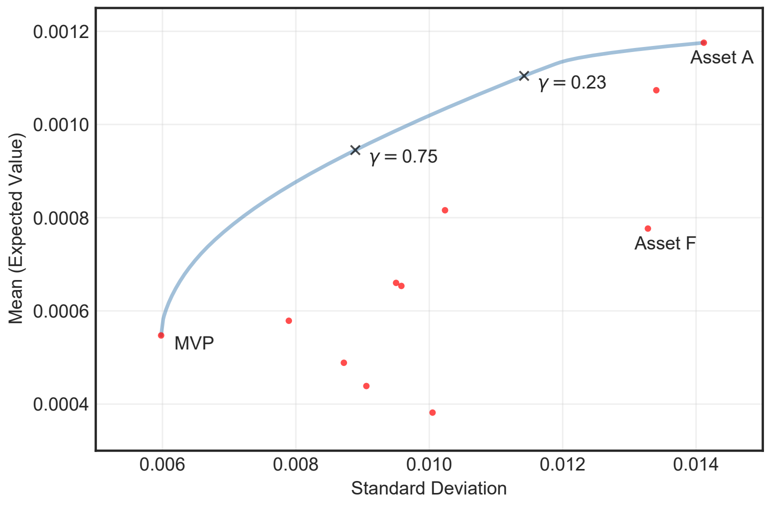


Figure 4.2: Effect of the risk aversion value on the position of the optimal portfolio on the efficient frontier.

We observe that for $\gamma = 0$, the optimal portfolio only consists of the one asset with the highest return, disregarding all other assets as described in the comment to (4.4). As the risk aversion increases $\gamma \to +\infty$, the scalar variable begins to push downwards toward the Minimum Variance Portfolio (MVP)(4.3), which is located at the beginning of the efficient frontier.

In Figure 4.3 we compute the solution to (4.5) for the arbitrary assets from Figure 4.2 and plot the relative weights of each asset against the corresponding value of risk aversion. As risk aversion decreases (from left to right), we move further away from the MVP and closer to the Asset A with the highest mean return. This shift can be observed clearly in the change of relative asset weights.

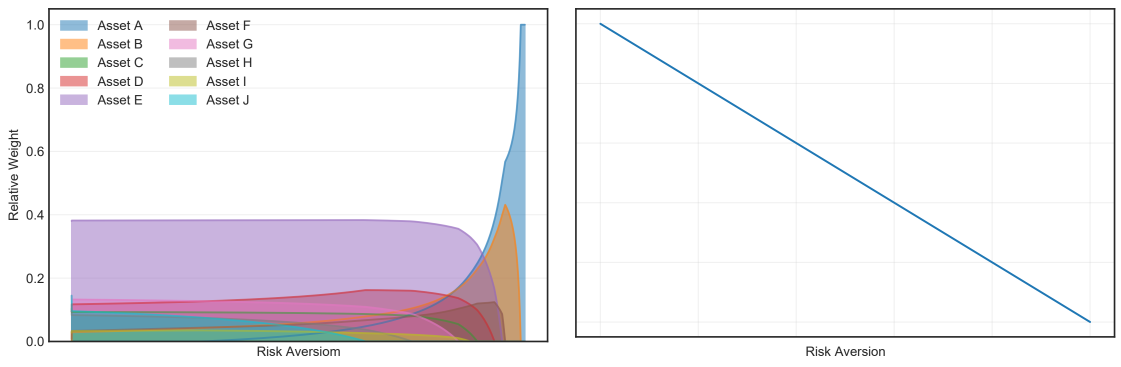


Figure 4.3: Relative portfolio composition as we move the risk aversion up along the efficient frontier.

4.2.1 Solving Constrained Optimization Problems

An essential tool in solving constrained optimization problems is the Lagrange function. Lagrangian multipliers can be used to find the extrema of a multivariate function subject to one or many constraints [Str91, p. 514]. First, the Lagrangian function L (4.6) is being set up by associating a so-called Lagrange multiplier λ with each of the constraints. With the lagrange multiplier, we are making use of the fact that when our objective function is tangential to the constraint, the two gradients that are perpendicular to the curve are pointing in the same direction. Therefore, we have $\nabla f = \lambda \nabla g$; that is, one is a multiple of the other. In a further step to find the desired extrema, we compute the derivative of the Lagrangian function and find points where the derivative is 0 (critical points) and evaluate the function at these points.

To solve the previously defined problem (4.3) we define the Lagrangian:

$$L(w, \lambda_1, \lambda_2) = \frac{1}{2} w^T \Sigma w + \lambda_1 (r_P - w^T M) + \lambda_2 (1 - w^T \mathbb{1}_m) \tag{4.6}$$

Derive the first-order conditions:

$$\begin{aligned} \frac{\partial L}{\partial w} &= \Sigma w - \lambda_1 M - \lambda_2 \mathbb{1}_m &&= 0_m \\ \frac{\partial L}{\partial \lambda_1} &= r_P - w^T M &&= 0 \\ \frac{\partial L}{\partial \lambda_2} &= 1 - w^T \mathbb{1}_m &&= 0 \end{aligned} \tag{4.7}$$

By substitution λ_1, λ_2 for w we arrive at:

$$\begin{bmatrix} r_P \\ 1 \end{bmatrix} = \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} \tag{4.8}$$

with

$$a = M^T\Sigma^{-1}M, \quad b = M^T\Sigma^{-1}\mathbb{1}_m, \quad c = \mathbb{1}_m^T\Sigma^{-1}\mathbb{1}_m \tag{4.9}$$

With the Lagrange multiplier λ_1 and λ_2 the vector of weights for the optimal portfolio can be formulated as:

$$w_0 = \lambda_1\Sigma^{-1}M + \lambda_2\Sigma^{-1}\mathbb{1}_m \tag{4.10}$$

Since the variance is always non-negative, we assume Σ to be a positive definite matrix. This is a necessary condition for the optimization to yield a global optimum. Since the inverse of a positive definite matrix is also a positive definite matrix, this can be assumed for Σ^{-1} as well.

Therefore, the variance of the minimum-variance target portfolio can be calculated as follows:

$$\begin{aligned}
\sigma_0^2 &= w_0^T\Sigma w_0 \\
&= \lambda_1^2(M^T\Sigma^{-1}M) + 2\lambda_1\lambda_2(M^T\Sigma^{-1}\mathbb{1}_m) + \lambda_2^2(1_m^T\Sigma^{-1}\mathbb{1}_m) \\
&= \begin{bmatrix} r_P \\ 1 \end{bmatrix}^T \begin{bmatrix} a & b \\ b & c \end{bmatrix}^{-1} = \begin{bmatrix} r_P \\ 1 \end{bmatrix} \\
&= \frac{1}{ac-b^2}(c(r_P)^2 - 2b(r_p) + a)
\end{aligned} \tag{4.11}$$

The problems (4.4) and (4.5) can be solved by equivalent Lagrangians. From these problems we can then find the efficient frontier which is the collection of all possible solutions. This can be done by ranging the target return (4.3), the maximum variance (4.4) or the risk aversion coefficient (4.5) amongst all feasible values. All these problems are convex quadratic optimization problems. By adding inequality constraints such as (3.4) to the basic mean-variance model, the problems become more complex, which means that they generally do not have an analytical closed-form solution anymore.

4.2.2 Maximizing Sharpe Ratio

As an alternative or addition to the previously proposed problems, we consider the problem of finding the efficient portfolio with the highest reward-to-variability ratio. This ratio is also known as the Sharpe ratio and can be useful especially if there is no clear index of satisfaction that maximizes the investor's utility. The common formulation for this problem is the following:

$$\begin{aligned} \text{maximize} \quad & \frac{w^T M}{\sqrt{w^T \Sigma w}} \\ \text{subject to} \quad & w^T \mathbb{1}_m = 1 \\ & w_i \geq 0 \end{aligned} \tag{4.12}$$

Notice that the long only constraint $w_i \geq 0$ is not necessarily a feature of the Sharpe Ratio formulation. This problem, as it stands, is not a quadratic optimization problem, and the objective function is not convex. Nevertheless, this problem can be recast as a quadratic convex optimization problem via a suitable homogenization [CPT18, p. 103] under the assumption that a vector w exists satisfying all constraints such that $w^T M > 0$. Meaning that at least one combination of assets is able to yield a positive mean return. By doing so we arrive at the following equivalent formulation with a convex objective function:

$$\begin{aligned} \text{minimize} \quad & z^T \Sigma z \\ \text{subject to} \quad & z^T M = 1 \\ & \sum_{i=1}^{N} z_i = \kappa \\ & \kappa \geq 0 \\ & w_i = \frac{z_i}{\kappa} \end{aligned} \tag{4.13}$$

Where z is the set of unscaled weights and κ the scaling vector. For a more comprehensive mathematical proof of equivalency see Section A.1.

A more straightforward approach would be to consider a two-step process where first the efficient frontier is defined by solving (4.5) and then perform a one-dimensional search for γ^* to find the allocation σ^*, r^* which yields the portfolio with the maximum Sharpe ratio. We will see in Chapter 5, that the result is equivalent for both methods as long as the mentioned assumptions hold.

4.2.3 Semivariance Models

For the classic mean-variance portfolio optimization, we have assumed that the investor is only concerned about the first two moments of the return distribution: Mean and variance. However, actual market returns are not entirely normally distributed but display skewness, which means that the variance above and below the mean is not equal. If this is the case, a measure of downside risk (downside deviation; which measures variance only below a benchmark) such as the excess return to downside variance ratio proposed by Sortino [SP94] should be implemented. Consequently mean-semivariance portfolio optimization should yield results superior to the more popular mean-variance optimization. However, since the semicovariance matrix is endogenous, meaning its terms change if the portfolio weights change, the optimization problems become intractable. Estrada [Est07] among others proposed a heuristic solution: The elements of the semicovariance matrix are computed with respect to when the single assets unperformed the benchmark and not the portfolio as a whole. This again yields a symmetric and exogenous semicovariance matrix. Nevertheless, even with a feasible semicovariance matrix Cheremushkin [Che09] shows that this approach can lead to significant approximation errors, especially when assets are negatively correlated. Even if the assets are uncorrelated, substantial errors can occur. The author concludes that this is due to the fact that the heuristic devised by Estrada [Est07] does not account for upside returns of one asset that could compensate the downside returns of another asset. The problem is to be found in the notion of downside risk, not necessarily in the measure of downside risk. It conditions the investor to estimate the necessary inputs with a fraction of the available information. For these reasons, semivariance models have not gained acceptance.

4.2.4 Downside Risk Measures (VaR / CVaR)

Another risk measures, which aims to address the shortcomings and pitfalls of the classic mean-variance model is a measure called Value at Risk (VaR). This concept was first introduced by a team at J.P. Morgan [JPM96]. VaR indicates the worst possible loss to a portfolio, at a given confidence level (typically 99% or 95%), within a given period of time. Following the notation in [CPT18, p. 183] the random variable Y describes the loss function of the portfolio and $\beta \in (0, 1)$ the confidence level. Then the β value at risk of Y is the $(1 - \beta)$ quantile of Y that is, the value ξ such that:

$$\mathbb{P}(Y \geq \xi) = 1 - \beta \tag{4.14}$$

Both a higher confidence level and a longer period of observation imply, all other things being equal, a higher VaR. Supposed the loss function is normally distributed, so that $Y \sim N(\mu, \sigma^2)$, the VaR can be computed via the known quantiles of the normal distribution. Even though the VaR function is monotonic (entirely non-increasing, or entirely non-decreasing), homogeneous (multiplicative scaling behavior), and translation invariant (translation with a displacement vector does not change the value of the function), it is not sub-additive. That is $VaR(r_{P1} + r_{P2}) \leq VaR(r_{P1}) + VaR(r_{P2})$ and therefore does not improve with diversification, diversification can actually increase VaR. Only fulfilling three of the four criteria set by Artzner et al. [Art+99, p. 210] it cannot be considered a coherent risk measure. Another shortcoming is that VaR does not provide any measure for losses beyond its threshold. However, there is a modification of VaR that is coherent and provides a loss measure beyond the VaR.

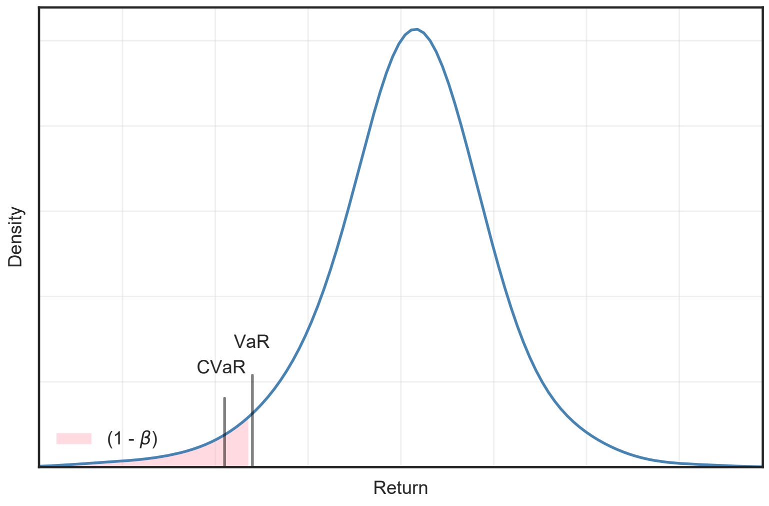


Figure 4.4: Probability density function with illustrative distinction between VaR and CVaR.

Namely the Tail Conditional Expectation known as Conditional Value at Risk (CVaR) or Expected Shortfall. Introduced by Rockafellar and Uryasev [RU00] the CVaR denotes the average loss, when VaR is exceeded. More formally: Given a random variable Y describing the loss function and confidence level $\beta \in (0,1)$ the conditional value at risk is the expected loss Y, under the condition that the loss is at least $VaR_\beta(Y)$:

$$\mathbb{E}(Y \mid Y \geq VaR_\beta(Y)) \tag{4.15}$$

Instead of using the loss function, CVaR and VaR can also be defined with a return function. The random variable $Z := -Y$ then describes the return function of the portfolio over the period under consideration.

A key property of CVaR as Rockafellar and Uryasev [RU00, p. 27] find, is that it can be expressed as the following equivalent convex function. This allows to solve CVaR portfolio optimization problems via convex optimization.

$$F_\beta(w,\alpha) = \alpha + \frac{1}{1-\beta} \int [-w^T r - \alpha]^+ p(r) dr \tag{4.16}$$

Where w is the vector of weights, r a vector of daily asset returns with probability distribution $p(r)$, $-w^T r$ the loss (negative return) of the portfolio and α the portfolio VaR with confidence β. We further suppose that $[x]^+ = max(x,0)$.

The authors then prove that minimizing $F_\beta(w, \alpha)$ over all w, α successfully minimizes the CVaR. For a set of T daily returns the integral in (4.16) can be approximated. In terms of auxiliary real variables u_t it is equivalent to minimizing the following linear expression:

$$\begin{aligned} \text{minimize} \quad & \alpha + \frac{1}{T(1-\beta)} \sum_{t=1}^{T} u_t \\ \text{subject to} \quad & u_t \geq 0 \\ & u_t \geq -w^T r_t - \alpha \end{aligned} \tag{4.17}$$

Note, although VaR and CVaR are usually defined in terms of money, here we follow the mentioned literature and intentionally use percentage terms. The optimization in (4.17) yields a vector of weights minimizing portfolio CVaR. Although, formally, the method only minimizes CVaR, numerical experiments [RU00, p. 37] indicate that it also lowers VaR because CVaR $\geq$ VaR as it can be seen in Figure 4.4. Like minimizing absolute portfolio variance minimizing absolute CVaR might not be a desirable goal for most investors. Additional constraints can be added to optimize for other goals. By adding the constraint:

$$M^T w = r_P \tag{4.18}$$

Analogous to (4.3) CVaR will be minimized for a given target return. A common constraint is to maximize portfolio return while the portfolio CVaR should be kept below or at a given value:

$$CVaR \leq CVaR_P \tag{4.19}$$

If returns are considered independent and identically distributed random variables, the square-root rule applies, meaning that shorter-term VaR and CVaR grow at the square root of the longer-term horizon.

4.2.5 Effects of adding a Risk-free Asset

For now, we have always considered all assets to be risky, equity-like assets. Meaning that for every expected return, we have an associated uncertainty in the form of variance. We considered the variance of returns or its square root, the standard deviation, the investor's measure for risk.

Nevertheless, there are also assets that are considered to be risk-free r_f. A risk-free asset has a certain future return; that is, the value of the return is given, and the variance is zero. In academic literature, it is widely accepted to consider any government debt issued by a stable nation to be risk-free. We now suppose the investor can invest in n risky assets as well as the risk-free asset. Since the variance is zero, the risk-free asset is to be found

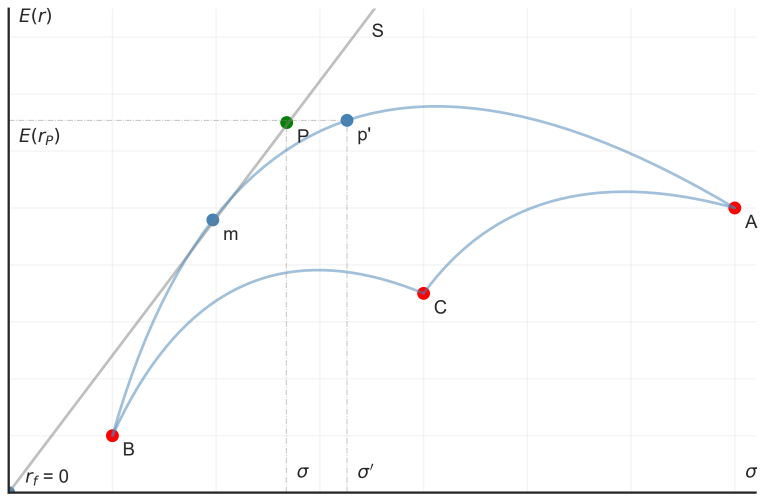


Figure 4.5: Efficient portfolio with the risk-free asset.

on the y-axis of our mean-variance plot in Figure 4.5. If we combine this asset with any of the portfolios on the efficient frontier, the feasible set will be any point along the line S which is a tangent to the efficient frontier and is often called the Capital Market Line. This combination will lead to a Portfolio with the expected mean that is the weighted sum of the risky assets and the risk-free asset. As we can see, we are now able to achieve an improvement over the points on the efficient frontier with either a higher return for the same amount of variance or less variance for the same expected return. The problem corresponding with 4.3 in this case can be formulated as:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} w^T \Sigma w \\ \text{subject to} \quad & w^T M + (1 - w^T \mathbb{1}_m) r_f = r_P \\ & w^T \mathbb{1}_m = 1 \end{aligned} \tag{4.20}$$

This problem can again be solved with a Lagrangian objective function. Note that the addition of the risk-free asset to the portfolio consisting of risky assets does not change portfolio variance since $w_f \sigma_f^2 = 0$ with $\sigma_f^2 = 0$. By changing (reducing) the weight on the portfolio of risky assets, we achieve the same expected return but with a reduced standard deviation. In our formulation of the problem (4.20) this happens inevitably since the constraint $w^T \mathbb{1}_m = 1$ demands the sum of weights remain unchanged. The described tangency portfolio is also relevant for Tobin's [Tob58] Mutual Fund Theorem, where the author concludes that every optimal portfolio is a combination of the risk-free asset and the Market (tangency) Portfolio. Nevertheless, this thesis does not further regard optimization problems with a risk-free asset.

5 Implementation

This chapter forms the core of the thesis, in that we put the previously established theoretical findings into practice. We do so by comparing different approaches and modifications to evaluate which model offers the most significant benefit regarding efficiency and comprehensibility in practical asset allocation. This requires that the tools we employ can process large amounts of data aptly. Meanwhile, the syntax and semantics should not impose a barrier but enable the user to work more efficiently. Among the most used solutions for analysis and optimization in finance are R and Python, alongside languages such as C++, C#, and Java. One of the significant advantages of Python is that it is a high-level programming language with a simple syntax that in many ways resembles the English language. Its simplicity does not limit the functionality. Because of its open-source license, programmers can add specific features to the language. Mathematicians and programmers have made available some very complex structures and algorithms in Python for practitioners to use. In addition, with the rise of machine learning and artificial intelligence, many algorithms developed by university researchers are written in Python. For this reason, we decided to use Python and its available packages for the implementation part of this thesis.

5.1 Python in Finance

Many of the underlying principles and findings of portfolio theory and optimization have been around for many decades. Meanwhile, as Hilpisch [Hil19] points out, two things have changed drastically: Big data and the real-time economy. The unprecedented availability of data is a challenge and an opportunity to tune the models and gain new insights. Meanwhile, it demands increased computing power and an overhaul of legacy systems. With the rise of real-time data, models had to be adjusted to interpret and assess data in seconds or fractions of a second [Duh09]. The Python programming language is predestined to bring a solution to both of those challenges. Any algorithmic statement can be translated into a single line of python code. Code that is to be reused is organized in modules that can be called anywhere inside a project. Available performance packages provide essential pre-compiled functions to ensure reduced run-time, contributing to the efficiency of Python. Characterized by its benefits, such as the elegant syntax, efficient development, and usability for prototyping and production, Python became an ideal framework for the financial industry. We will introduce some of the packages that prove to be most helpful in solving the Problems introduced in this thesis.

5.1.1 NumPy

NumPy is a package or library, adding support for large, multi-dimensional arrays and matrices. It incorporates fundamental array concepts [Har+20] such as vectorization, broadcasting or indexing. This comes along with high-level mathematical functions to operate on these arrays. Providing stability and speed, NumPy has established itself in the industry, academia and finance.

5.1.2 Pandas

With its support for numerical tables and time series, Pandas became the most important Python library for data analysis and data science. It enables the user to import data from numerous file formats such as CSV, JSON, SQL, or Microsoft Excel. Further, Pandas also enables modification operations such as selecting, reshaping or merging, making data easier to read or more structured. By wrapping functionality from other packages, mathematical operations can be performed on a tabular level using Pandas.

5.1.3 Matplotlib

Visualizing data is an integral part of data analysis, especially in academia, where findings are often best presented graphically. Matplotlib allows the creation of static 2D and 3D bitmap plots. There are other libraries that offer interactive plots. We find that the outputs offered by Matplotlib are, in our case, often sufficient. Pandas already provides a wrapper around Matplotlib, enabling basic visualization of a pandas table with a single line of code.

5.1.4 CVXPY

According to Diamond and Boyd [DS16], the authors of cvxpy, their modeling language allows users to express convex optimization problems in a natural syntax that follows the math, rather than in the restrictive standard form required by solvers. The cvxpy modeling language makes it easy to combine convex optimization with high-level features of Python, such as parallelism and object-oriented design. Nevertheless, cvxpy is not a solver but attempts to find a suitable solver for a supplied problem [Agr+18]. In Subsection 5.2.3 we take a closer look at the different solvers and their characteristics.

All of the mentioned libraries are widely used in many different fields within and outside of finance. They are open-source, performance-oriented, and continuously developed.

5.2 Python Optimization Problem

To prove the solidity of our optimization problem we consider a simple example with a set of three arbitrary assets. We first calculated the solution using the Lagrangian defined in (4.6). We then implement the problem with cvxpy and compare the result obtained by the solver to the result from the Lagrange function. The assets under consideration have the following properties:

$$\mu = [\mu_i] = \begin{bmatrix} 0.017 \\ 0.002 \\ 0.008 \end{bmatrix}, \quad \Sigma = [\sigma_{i,j}] = \begin{bmatrix} 0.402 & 0.065 & -0.022 \\ 0.065 & 0.136 & 0.008 \\ -0.022 & 0.008 & 0.003 \end{bmatrix}$$

Further we define a target return for the Portfolio $r_P = 1.25\%$ to be achieved.

5.2.1 Solving the Lagrangian equation

From (4.8) we solve for λ_1 and λ_2:

$$\lambda_1 = \begin{bmatrix} - & 0.348 \\ & 0.812 \\ & 0.536 \end{bmatrix}, \quad \lambda_2 = \begin{bmatrix} & 45.708 \\ - & 98.105 \\ & 52.397 \end{bmatrix}$$

Only then we are able to calculate:

$$w = \begin{bmatrix} 0.224 & -0.414 & 1.190 \end{bmatrix}$$

Solving (4.11) we arrive at:

$$\sigma_0^2 = 0.01606$$

From this, we have proof that the satisfactory target return $r_P = 1.25\%$ can be achieved. Note that this illustrative example could only be solved in such a manner because we did not deal with any inequality constraints.

5.2.2 Optimizer

We set up the simple optimization problem defined in (4.3) using the cvxpy syntax as follows:

```python
w = cp.Variable(len(df_assets.columns),)
ret = w.T@get_mu(df_assets)
sigma_square = cp.quad_form(w, get_cov(df_assets))
objective = cp.Minimize(sigma_square)
constraints = [ret == 0.0125,
               cp.sum(w) == 1]

prob = cp.Problem(objective, constraints)
```

Listing 5.1: Minimum variance optimization Problem (4.3) formulated using cvxpy.

A detailed description of the setup and definition can be found in Subsection 5.4.1. Note again, that since there are no inequality constraints in this version of our optimization problem it can be solved with a linear system solver. In this case we use the qdldl routine from OSQP (Subsection 5.2.3). Running the optimizer with the convex problem defined in Listing 5.1 yields the following result:

```
status:               solved
solution polish:      successful
number of iterations: 25
optimal objective:    0.01606
w.value:              [ 0.22369437 -0.41445845  1.19076409 ]
run time:             7.06e-04s
```

As we can see, the solver returns σ_0^2 as our optimal objective. The result matches with the value we obtained by solving the Lagrangian (Subsection 5.2.1). To see if the constraints are being met, we calculate the return using the weights calculated by the solver and validate if the sum of the weight is equal to one. This proves that we successfully set up our optimization problem in Python.

5.2.3 Solver

Distributed with cvxpy are the open-source solvers ECOS, OSQP, and SCS [Agr+18]. Many other solvers can be called by CVXPY if installed separately[1]. As we will see, most of our problems, we are able to solve using OSQP [Ste+20] a Quadratic solver developed by a Team at the University of Oxford. This solver runs infeasibility detection proofs using the ADMM [Ban+17] algorithm.

[1]A comprehensive overview can be found in the cvxpy documentation.

5.3 Portfolio Construction

In the first step of the optimization process, we have to define our portfolio, that is, the data set from which the input to the optimization routine is taken. Further, we have to set a benchmark against which the success of our optimization can be measured. We choose to set our primary focus on risky assets, specifically on the Swiss stock market. The Swiss stock market is dominated by three companies: Nestlé, Roche, and Novartis. As of April 2021, these companies alone accounted for about 48% of the market in terms of capitalization. This imbalance has existed unchanged for more than 15 years. The same assets can also be found among the largest listed European companies, ranking second, third, and sixth. Furthermore, all major indices provided by SIX Swiss Exchange are weighted by market capitalization, amplifying this concentration. This setting poses a number of challenges for an investor considering this market, particularly with regard to diversification.

For our analysis, we chose 85 of the largest companies listed on the SIX Swiss Exchange by market capitalization and volume as of April 2021. We do not include securities with a free float of less than 20% or shares of investment companies. Further, we only included securities that were listed every day of the considered period. As a Benchmark, the SIX Swiss Exchange Index ”SPI Mid & Large Total Return” provides a good representation of the chosen set.

5.3.1 Data Sourcing

In order to collect information on the market, we turn to a data provider (Thomson Reuters Refinitiv) retrieving time-series of daily stock prices for the period of nine years starting from 2011. To do so, we made use of the subscription-based API. To gather more than 190’000 data points and still comply with the API limits, the datareader script shown in Listing 5.2 had to be set up.

The change in stock price is not always a proper reflection of an investor’s return. One example are dividend payouts; they negatively impact the stock price, yet this effect is offset by the cash the investor receives and can potentially reinvest. To address this issue, most stock exchanges provide adjusted closing prices. This data is unavailable for some of the stocks under consideration. Therefore, to account for dividends and stock splits and to ensure a fair comparison with the benchmark index, the daily total return calculated by Thomson Reuters Refinitiv was also pulled as a single time-series for each asset. The total return incorporates the price change and any relevant dividends for the specified period. This methodology is also known as the dividend reinvested total return. The daily total return data of the ”SPI Mid & Large Total Return” Index and the daily total return

```python
def eikon_datareader_TR(RICs, Start, End, Frequency):
"""
RICs : List of Thomson Reuters Instrument Code
Start: Last day of requested time series yyyy-mm-dd
End: First day of requested time series yyyy-mm-dd
Frequency: D, W, M, Q, Y
"""

    df, err = ek.get_data([RICs[0]],fields=[
        "TR.TotalReturn(SDate={},EDate={},Frq={}).date"
        .format(Start, End, Frequency)])
    df_store = pd.DataFrame(df["Date"].str[:10])

    for i in range(len(RICs)):
        df, err = ek.get_data([RICs[i]],fields=[
            "TR.TotalReturn(SDate={},EDate={},Frq={}).date"
            .format(Start, End, Frequency),
            "TR.TotalReturn(SDate={},EDate={},Frq={})"
            .format(Start, End, Frequency)])

        df_returns_temp = pd.DataFrame(df["Total Return"])
        df_returns_temp.columns = [df.iloc[1][0]]
        df_store = pd.concat([df_store, df_returns_temp], axis=1)
        i += 1

    return df_store
```

Listing 5.2: Script to request a total return time series from Thomson Reuters Eikon API.

data for each of the 85 Assets for the mentioned eight-year period build the basis of our optimization process.

5.3.2 Data Processing

The time series contains 85 Assets with 2256 daily observations each. The annual returns are between -20.44% and 25.62%, the annual volatility ranges between 8.52% and 50.51% for our period of observation. Meanwhile, the benchmark index yields a return of 9.39% with a volatility of 14.44% per year. Note that in the context of stock returns, the term *volatility* is often used to describe the standard deviation. This wording is in line with academic literature; therefore, the terms will be handled equally from now on. All assets in our data set are positively correlated. Since the chosen period of observation includes different market phases as well as extreme tail events, it is ideal for testing different optimization methods and models.

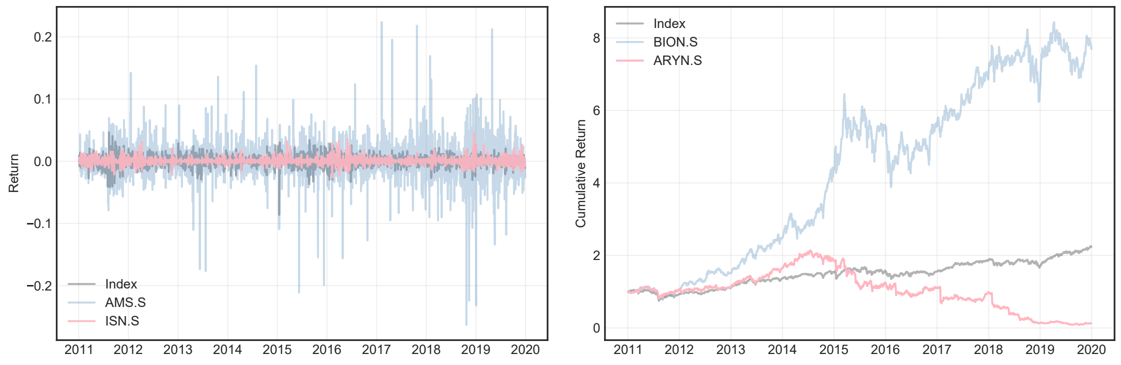


Figure 5.1: Most volatile asset (AMS) vs least volatile asset (ISN) and asset with highest return (BION) vs asset with lowest return (ARYN).

5.4 Portfolio Optimization

In this step of the process, the data will be applied to the problems established in Section 4.2 to solve for numerical solutions. In addition, the models will be extended and further developed based on the analytical findings. The main focus stays to provide efficient, accurate, and easily understandable python routines and to gain insights into the state of the Swiss equity market.

5.4.1 Reducing Volatility vs. Benchmark

To demonstrate the effect of mean-variance optimization an adaption of the minimum variance optimization problem (4.3) is being considered. To ensure a fair comparison, we add a long only constraint since this restriction also applies to the benchmark index. By doing so the following question is being answered: How can the weights of a portfolio be adjusted to achieve the same return as the market or benchmark but with less volatility? Using cvxpy we set up the following optimization problem:

```python
w = cp.Variable(len(df_assets.columns),)
ret = w.T@get_mu(df_assets)
sigma_square = cp.quad_form(w, get_cov(df_assets))
objective = cp.Minimize(sigma_square)
constraints = [ ret == get_mu(df_index),
                cp.sum(w) == 1.0,
                w >= 0.0]

prob = cp.Problem(objective, constraints)
```

Listing 5.3: Minimum variance optimization with three linear constraints concerning desired return, the sum of weights and a long only constraint.

The problem is constructed (9) by defining the objective and the applicable constraints. On line (1), the viable constructor from cvxpy is used to set up the variable w, representing the vector of weights. We then set the expected portfolio return (2) as a function of the weights vector and the expected returns vector. Further the expected portfolio variance (3) is setup as a function of the covariance matrix and the weights vector. The objective (4) is then set to minimize the previously defined expression using the corresponding cvxpy constructor. On line (5) and following, we set a list of constraints. The first constraint ensures that the return of the optimized portfolio is equal to the desired expected return, in this case, the return of the benchmark index. The second constraint represents the general requirement (2.11) that in Markowitz style portfolio creation, the relative weights always have to sum up to one. With the last constraint, for the reasons already mentioned, short selling (3.4) is being prevented. The Helper Functions used in the code, such as *get_mu*, are user-defined and can be found in Section B.2. Solving this problem with the previously developed methods leads to an optimized portfolio containing only 18 of the available 85 assets. The optimization appears to be successful as the volatility is being reduced by more

Table 5.1: Benchmark compared to volatility optimized portfolio.

	Index	**MV Optimized**
Expected Annual Return	9.386%	9.386%
Annual Volatility	14.445%	5.574%
Sharpe Ratio	0.650	1.684
Number of Assets	85	18

than 60% while attaining an equivalent return of 9.386% annually. However, this result must be taken with caution since the optimization was done entirely in-sample, meaning that all historical data-points were available to the optimizer. Therefore, the portfolio is not guaranteed to be optimal in the future or any other scenario. One could argue that

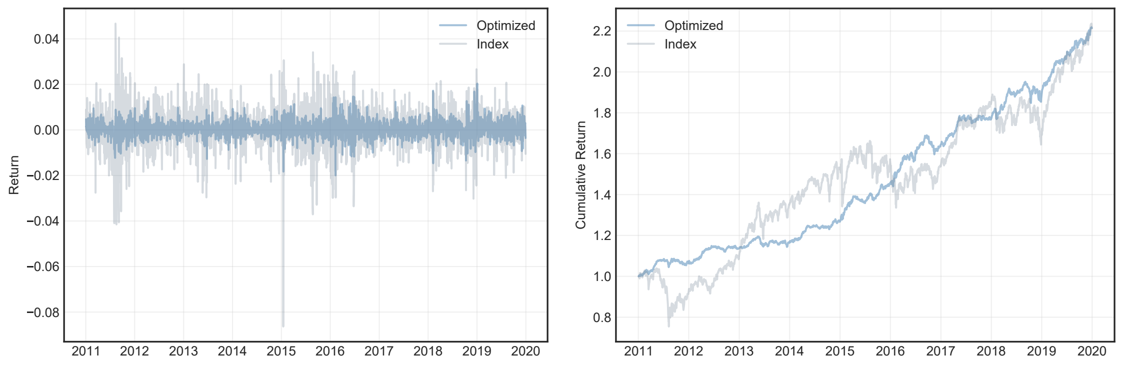


Figure 5.2: Time series of the results shown in Table 5.1.

it is straightforward to find an optimal portfolio with the benefit of hindsight. However, this is beyond the point of this step of the process. The result merely proves that with the methods described in this thesis, it is possible to make a weighted selection of assets resulting in more desirable parameters, in this case, lower volatility of returns. A closer look at the resulting weights reveals that 85% of the optimized portfolio is made up by only six of the 18 assets. This finding raises the question of whether a satisfactory result could also be achieved with fewer assets. When the optimization is performed again, this

Table 5.2: Single asset weights from optimization with all available assets compared to optimization only with assets w >5% in the first run.

	Weights run 1	**Weights run 2**	**Difference**
ISN.S	0.346	0.469	+35.5%
BEKN.S	0.114	0.072	-36.8%
ZUGER.S	0.112	0.120	+7.1%
GRKP.S	0.101	0.087	-13.9%
LUKN.S	0.093	0.100	+7.5%
ALLN.S	0.089	0.153	+71.9%

time only with assets that have reached a weighting >5%, the annual volatility increases by only 0.233 percentage points to 5.807%. The observed difference is consistent with the finding that diversification's impact on portfolio risk has diminishing benefits, even when transaction and holding costs are ignored. First discovered by Evans and Archer [EA68] over 50 years ago, the limiting number was set to ten assets. Since then, following the evolution of capital markets, different statements have been made about the number of stocks needed for efficient diversification [Sta87][CLM][DLR06]. Wherever the number was found to be, the underlying assumption stayed the same.

A look at the change in asset weights reveals that the proportions have not scaled linearly. This observation ultimately leads to the conjecture that a more satisfying result could have been obtained by limiting the number of assets with a constraint on minimum holding size or the number of assets held. Nevertheless, when expressing such constraints in a way that ensures convexity new challenges arise, which we further discuss in Section 6.3.

Removing Long Only Constraint As noted in Subsection 3.3.5 each constraint limits the number of feasible optimal portfolios. Therefore lifting the long-only constraint $w \geq 0$ from the previous example should naturally lead to a "better optimized" portfolio. In this case, equal or less volatility for an equal expected return compared to the first optimization run. Allowing short positions means that the calculated weights can be less than zero. In practice, to be short of an asset the investor has to borrow this specific asset and immediately sell it again. The investor is then short of this asset because he or she has to repurchase this asset to fulfill the obligation towards a creditor. Since the portfolio weights still have to add up to one, this allows to overweight the portfolio's long component, in this case, assets with a mean-variance profile more desirable to the investor.

Table 5.3: Benchmark compared to volatility optimized portfolio (long-only) and an optimized portfolio with short-selling allowed.

	Index	**MV optimized**	**Long/Short optimized**
Expected Annual Return	9.386%	9.386%	9.386%
Annual Volatility	14.445%	5.574%	5.276%
Sharpe Ratio	0.650	1.684	1.779
Number of Assets	85	18	46 (39)

From the results in Table 5.3 it can be seen that the volatility can be further reduced by allowing short positions. Nevertheless, if transaction and holding costs were introduced into the model, the benefit from reduced volatility could be offset by the additional number of positions (46 long and 39 short). There are two apparent reasons for the minimal effect on the expected volatility from allowing short-positions. The main reason lies in the optimization problem under consideration: The first constraint still limits the portfolio return to equal the benchmark index's return. The other explanation can be found in the positive long-term correlation of the available assets from our data-set.

Even with short positions allowed, the top long holdings are represented by the same positions, as in the previous long-only optimization, with only slightly different weightings. The largest short positions are all represented by assets previously not included in the 18 assets long-only portfolio. It also has to be noted that the number and size of short positions are substantially smaller than the number and size of long positions. Lifting the long-only constraint leads to an absolute sum of weights of 1.662. This number can be interpreted as leverage of 66.20% on the portfolio level.

5.4.2 Enhancing Return vs. Benchmark

Another approach is to optimize for maximum return (4.4) instead of minimum volatility. Here a different question is being answered: How can the weights of a portfolio be adjusted to achieve the highest possible return while limiting the volatility to not exceed the benchmark volatility? Similar to the problem set up in Listing 5.3 we use cvxpy to implement the problem shown in Listing 5.4. Again, to ensure a fair comparison with the benchmark index a long-only constraint is added at first. The main difference regarding the cvxpy setup is that we now maximize a concave problem. Equivalently we could also formulate this as a convex minimization problem. The results in Table 5.4 show that the optimized portfolio includes only 11 of the 85 available assets. The previous remarks on diversification apply here as well. Since many of the assets have a high positive pairwise

```python
w = cp.Variable(len(df_assets.columns),)
ret = w.T@get_mu(df_assets)
sigma_square = cp.quad_form(w, get_cov(df_assets))
objective = cp.Maximize(ret)
constraints = [ sigma_square <= get_var(df_index),
                cp.sum(w) == 1.0,
                w >= 0.0]

prob = cp.Problem(objective, constraints)
```

Listing 5.4: Maximum return optimization with three linear constraints concerning desired variance, the sum of weights and a long only constraint.

Table 5.4: Benchmark compared to return optimized portfolio.

	Index	**Return Optimized**
Expected Annual Return	9.386%	23.397%
Annual Volatility	14.445%	14.445%
Sharpe Ratio	0.650	1.620
Number of Assets	85	11

correlation, no further diversification can be achieved by adding more assets from the same basket. Nevertheless, an excess return of 432.23% over the period of observation compared to the benchmark index is earned by the investor with the optimized portfolio. Meanwhile, the volatility of both portfolios is equal. In Figure 5.3 the diversion of returns becomes clear. An important observation here is that compared to the minimum variance optimization in Subsection 5.4.1 the expected/desired volatility is 8.91 percentage points higher, which is a factor of almost 2.6. This allows for more volatile assets with higher expected returns to be included. On the downside, the Sharpe ratio, which reflects the re-

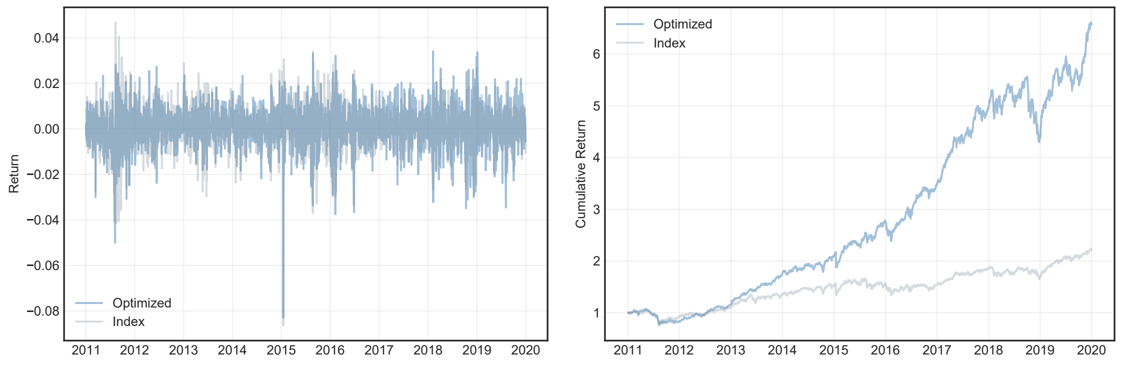


Figure 5.3: Time series of the results shown in Table 5.4

lationship between reward to variability, decreases from 1.684 for the minimum variance portfolio to 1.620 for the maximum return portfolio. Meaning the investor accepts a small amount of excess risk for the higher returns. This disparity can be addressed by explicitly solving for the optimal Sharpe ratio.

Removing Long Only Constraint Removing the long-only constraint has a much more significant effect when optimizing for maximum returns. Throughout observation, the return would grow almost 25 times more compared to the benchmark. This makes it clear that being able to enter short positions is more advantageous when aiming to increase returns than when limiting volatility. However, the investor must be aware that giving up the long-only constraint can lead to a highly leveraged portfolio. In this example, the leverage is at 476.8%, which in reality would bring new challenges such as that a margin account with adequate collateral is needed. It must be pointed out again that the complete optimization was done in-sample.

Table 5.5: Benchmark compared to volatility optimized portfolio (long-only) and an optimized portfolio with short-selling allowed.

	Index	**Return optimized**	**Long/Short optimized**
Expected Annual Return	9.386%	23.397%	49.859%
Annual Volatility	14.445%	14.445%	14.445%
Sharpe Ratio	0.650	1.620	3.452
Number of Assets	85	11	46 (39)

5.4.3 Sharpe Ratio Optimization

An Investor aiming to combine the objectives of minimum variance and maximum return naturally arrives at the Sharpe ratio (4.12) asking the following question: How much additional return do I get for an additional unit of risk? To maximize the Sharpe ratio as convex problem the variable transformations which have been worked out in Subsection 4.2.2 have to be applied. Here z is considered the transformed variable:

```python
z = cp.Variable(len(df_assets.columns),)
k = cp.Variable()
ret = z.T@get_mu(df_assets)
sigma_square = cp.quad_form(z, get_cov(df_assets))
objective = cp.Minimize(sigma_square)
constraints = [ ret == 1.0,
                cp.sum(z) == k,
                z >= 0.0,
                k >= 0.0 ]

prob = cp.Problem(objective, constraints)
```

Listing 5.5: Transformed Sharpe Ratio maximization problem.

Once the problem is solved, the variable transformation has to be reversed; this is done by dividing the transformed variable z by the scalar k. The results in Table 5.6 show the success of the optimization. The optimized portfolio has an expected return higher than the benchmark index, as well as a lower volatility.

Table 5.6: Benchmark compared to sharpe optimized portfolio.

	Index	**Sharpe Optimized**
Expected Annual Return	9.386%	15.676%
Annual Volatility	14.445%	7.347%
Sharpe Ratio	0.650	2.134
Number of Assets	85	21

Lifting the long-only constraint $z \geq 0$ results in a portfolio which includes all of the 85 available assets. Again, the portfolio is highly leveraged with 336.8% and has a Sharpe ratio of 3.368, which is higher than the long/short optimized maximum return portfolio. These results confirm the thesis in Subsection 5.4.2 were we stated that the investor with the maximum return portfolio is taking on too much excess risk. Further descriptions and variations of the problem are omitted since the results behave similarly to the previous examples.

5.4.4 Risk Aversion Optimization

Implementing the risk aversion optimization problem (4.5) poses one major challenge: The constant γ has to be implemented in a way that the optimization problem can be solved for w as in the previous examples and a value to the scalar γ can be assigned after construction of the problem. The cvxpy *parameters* expression allows us to specify the value of the parameter after the problem is created. Therefore, the parameters expression proves to be ideal for computing trade-off curves such as the efficient frontier.

```python
w = cp.Variable(len(df_assets.columns),)
ret = w.T@get_mu(df_assets)
sigma_square = cp.quad_form(w, get_cov(df_assets))
gamma = cp.Parameter(nonneg=True)
objective = cp.Maximize(ret - gamma*sigma_square)
constraints = [cp.sum(w) == 1.0,
                w >= 0.0]

prob = cp.Problem(objective, constraints)
```

Listing 5.6: Risk aversion optimization problem.

The goal of this optimization is twofold. Firstly, risk aversion optimization allows us to construct the efficient frontier and optimize for the investor’s risk aversion. Secondly, the efficient frontier then allows us to derive the maximum Sharpe ratio and therefore prove the validity of the solution to our problem in Subsection 5.4.3. To obtain the entire curve, the optimization process is being iterated with a varied risk aversion parameter. The parameters range from 0, the position of the maximum return portfolio, up to a parameter value that is no more than $5 \cdot 10^{-4}$ larger than the respective risk aversion of the minimum variance portfolio. With this routine, a smooth curve can usually be plotted with less than 1'000 iterations. A practitioner's approach to finding the optimal risk aversion parameter

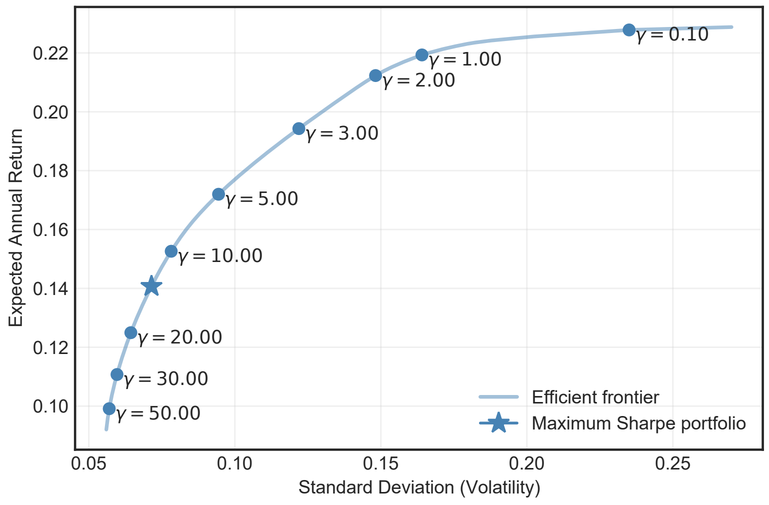


Figure 5.4: Efficient frontier with different risk aversion parameters γ highlighted.

γ^* would be to present the investor with the efficient frontier in Figure 5.4. Since the optimal allocation has to lie on the presented curve, that is, the efficient frontier, the investor could then pick a point on that curve. He or she would be able to do so because it becomes apparent that a 10% annual expected volatility, as an example, bears the potential of an 18% annual return.

From this statement, the investor's optimal risk aversion $\gamma^* \approx 4.25$ would then be derived. In Figure 5.5 we plot the return distributions of the portfolios for different levels of risk aversion. By comparing the different distributions, the effect of reduced variance for higher levels of risk aversion can be clearly recognized.

Another approach is the two-step mean-variance heuristic presented by Meucci [Meu05, pp. 249, 338]. Akin to the approach presented here in the first step, the mean-variance efficient frontier is computed. Instead of presenting the investor with the efficient frontier, a function called the index of satisfaction S is introduced. This function translates the return distribution of the portfolio into the investor's personal preference ranking. Therefore, the

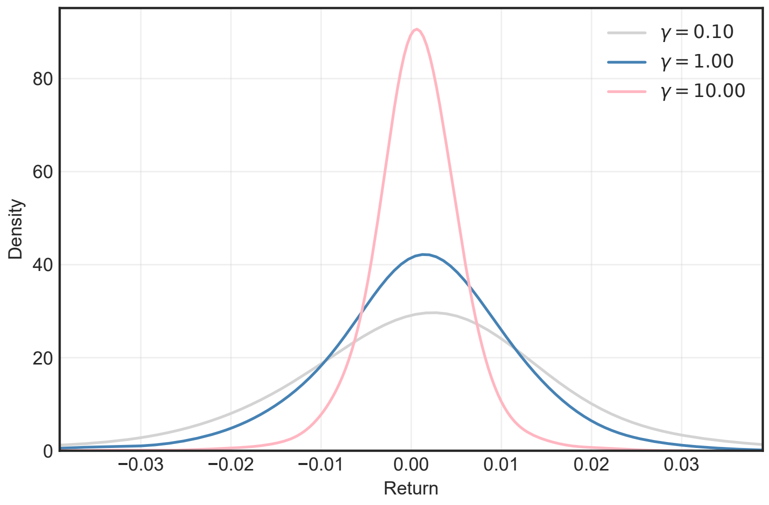


Figure 5.5: Return distributions for different levels of risk aversion.

optimal Portfolio is selected by the following one-dimensional search:

$$w^* = argmax\{S(w_\gamma)\} \tag{5.1}$$

Meucci [Meu11, p. 21] does not fail to mention that the choice of the most suitable index of satisfaction S varies widely depending on the profile of the securities return distribution, the investment horizon, and other features of the market and the investor. To keep the model as general as possible, we refrain from further detailing this two-step process.

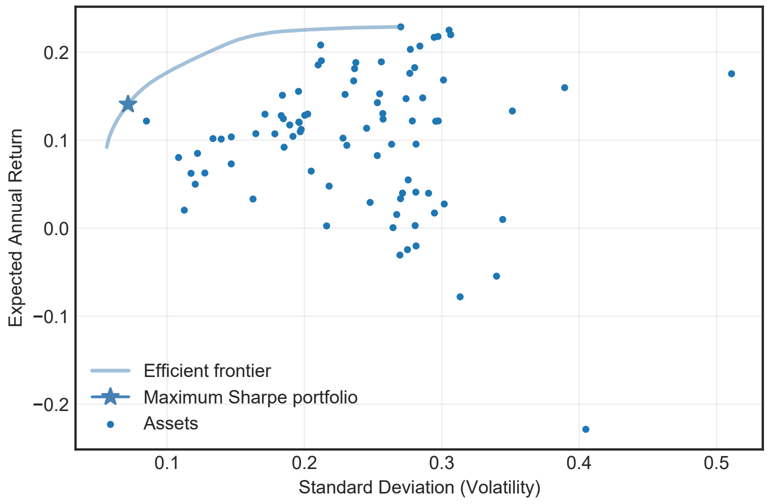


Figure 5.6: Scatter diagram detailing the mean-variance characteristic for each asset in relation to the efficient frontier.

On the mean-variance diagram in Figure 5.6, it can be seen how the different assets are positioned relative to the curve. We calculate the Sharpe ratios and search for the portfolio with the highest ratio from the efficient portfolios that form the frontier. The portfolio, which is also highlighted in Figure 5.6 has a Sharpe ratio of 2.134. The ratio is equivalent to the one that was calculated in Subsection 5.4.3 thus proving the validity of both approaches.

5.4.5 Conditional Value at Risk Optimization

We now suppose that the investor is concerned with risk in quantities of Expected Shortfall / Conditional Value at Risk instead of the symmetric measure of standard deviation. The Python code in Listing 5.7 is a variation of the problem (4.15), with the additional constraint (4.18) the CVaR is minimized for a given target return.

```python
w = cp.Variable(len(df_assets.columns),)
alpha = cp.Variable()
u = cp.Variable(len(df_assets))
beta = 0.95

return_target = get_mu(df_index)
ret = w.T@get_mu(df_assets)

objective = cp.Minimize(alpha +1.0 / (len(df_assets)*(1 - beta))*cp.sum(u))
constraints = [
                cp.sum(w) == 1,
                u >= 0.0,
                w >= 0.0,
                df_assets.values@w + alpha + u >= 0.0,
                ret >= return_target
                ]
prob = cp.Problem(objective, constraints)
```

Listing 5.7: Optimization problem minimizing CVaR for a given target Return.

In line with the previous examples, we set the desired target return to equal the benchmark's return. With a confidence of 95% ($\beta = 0.95$) the benchmark index will not lose more than 1.407% (VaR). The historically expected shortfall (CVaR) is 2.174%. The results in Table 5.7 show that the CVaR and therefore the VaR could be reduced substantially even at a higher confidence level.

Table 5.7: 1-Day minimum VaR and CVaR obtained for an expected return equal to the benchmark.

	$\beta = 0.90$	$\beta = 0.95$	$\beta = 0.99$
VaR	0.355%	0.512%	0.866%
CVaR	0.578%	0.730%	1.084%

The accuracy of the solution is confirmed by comparing the solver result with the historic CVaR of the corresponding portfolio.

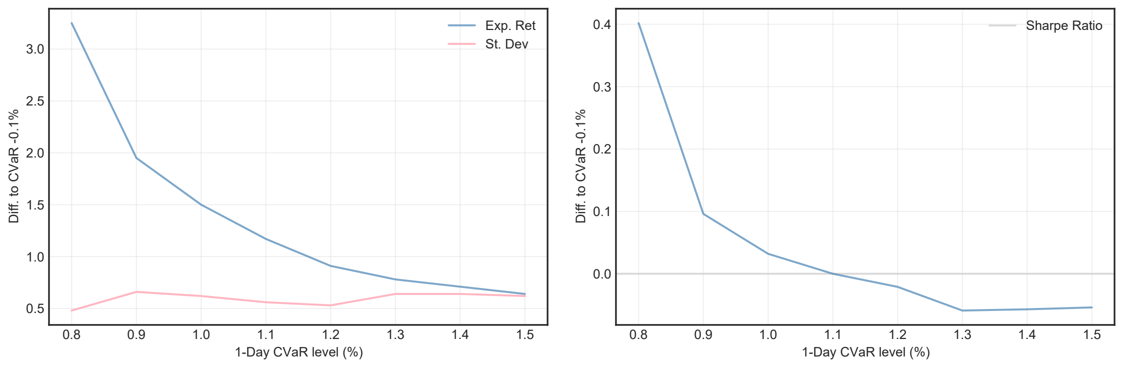


Figure 5.7: Diminishing excess return from higher CVaR (β=0.95) and therefore diminishing Sharpe ratio.

We further find that when increasing the risk level, additional volatility grows disproportionately to the additional expected return. Figure 5.7 shows that there is no linear relationship between the two measures, which should be kept in mind when comparing them in the following section.

5.5 Results

In this section, we combine and extend the Python routines previously introduced to provide some practical insights. For this purpose, we think of an investor willing to take risk, but only if he or she is adequately compensated for the risk. We consider the period from January 2011 to December 2017 as the in-sample period. In the previous chapters, we found risk measures more sophisticated than the classical mean-variance model; therefore, we consider the expected shortfall (CVaR) as our measure of risk.

We further found that it is challenging to quantify the investor's risk aversion. For this reason, we propose a different approach; in the first step, we calculate the efficient frontier for different levels of CVaR starting from the minimum CVaR portfolio. As a modification of the problem in Listing 5.7, we replace the constraint that requires the optimized portfolio to achieve a target return. Instead, we implement a constraint to require a desired CVaR level. We then perform a one-dimensional search on the efficient frontier to obtain the portfolio with the highest Sharpe ratio. In Table 5.8 the results for the confidence level $\beta = 0.95$ can be found. In the second step, we find that the maximum Sharpe Ratio is achieved at a 1.00% CVaR level.

Table 5.8: Portfolio compositions for different 1-day CVaR levels with $\beta = 0.95$; Assets with zero weights for any portfolio not included in list. Period: Jan 2011 - Dec 2017.

CVaR, %	**0.68**	**0.70**	**0.75**	**0.80**	**0.85**	**0.90**	**1.00**	**1.10**	**1.20**
Exp. Ret, %	9.82	11.59	14.02	15.56	16.76	17.67	19.12	20.39	21.52
St.Dev, %	5.53	5.67	6.17	6.59	6.96	7.27	7.83	8.42	9.05
Sharpe	1.777	2.045	2.271	2.362	2.408	2.432	2.44	2.42	2.38
ISN.S	44.84	44.51	47.89	50.37	53.43	52.88	50.25	49.37	45.15
LUKN.S	14.07	14.35	18.06	17.77	13.15	9.2	3.88	0	0
COTNE.S	0.92	1.72	3.45	6.22	7.81	9.9	12.17	13.44	16.08
CONC.S	3.63	3.92	5.74	6.85	7.31	7.65	7.8	7.76	7.49
BANB.S	3.89	4.71	5.3	5.25	5.78	5.12	4.61	4.25	3.08
LEHN.S	0.54	1.26	2.59	2.91	3.08	4.04	5.29	5.68	6.71
ALSN.S	1.34	1.31	2.21	3.01	3.2	3.19	3.96	4.87	6.42
JFN.S	0	0.73	2.36	3.48	2.94	3.7	4.24	2.82	1.37
GRKP.S	7.31	7.29	3.39	2.22	0	0	0	0	0
BEKN.S	9.67	7.57	1.32	0	0	0	0	0	0
ZUGER.S	6.39	5.17	3.39	0.53	0	0	0	0	0
EMSN.S	0	0	0	0.03	0.31	1.66	3.25	4.08	4.86
ALLN.S	5.72	5.34	1.67	0	0	0	0	0	0
AMS.S	0	0	0.15	0.1	0.51	0.64	1.09	1.78	2.89
BOS.S	0	0	0	0	0	0.24	1.11	1.5	2.35
INRN.S	0	0	0	0.45	0.83	0.65	0.74	1.5	0.86
SFZN.S	0	0.73	0.49	0	0	0	1.22	0.78	1.55
PGHN.S	0	0	0.25	0.53	1.67	1.13	0.41	0	0.23
VZN.S	0.78	0.89	0.99	0.28	0	0	0	0	0
BION.S	0	0	0	0	0	0	0	1.5	0.92
BARN.S	0.57	0.5	0.75	0	0	0	0	0	0
KARN.S	0	0	0	0	0	0	0	0.67	0.02
GIVN.S	0.34	0	0	0	0	0	0	0	0
Total w, %	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0

It can be seen from the results that this optimization process leads to large positions in single assets, which in turn leads to undesirable cluster risk. To prevent unwanted large positions, we add a constraint (3.6) to limit the maximum size of a single position to 10% of the portfolio. Restrictions such as this can also be found to be regulatory limits for mutual funds in many markets. The constraint inevitably leads to a greater number of positions from which the portfolio is composed. In our example, the number of positions increased from 14 to 18, with the four largest positions being capped at 10% weighting.

We have already proven that optimization carried out in this way achieves excellent in-sample results. We now put our routines to the test using the weights calculated from the given in-sample historical data to calculate the out-of-sample performance. The years 2018 and 2019 show two different market phases and can provide valuable insights. We therefore define these two periods as our out-of-sample periods. For the second out-of-sample period (2019), the returns from the previous period (2018) were included in-sample, which is equal to an annual rebalancing.

Table 5.9: Out-of-sample results for two different market phases.

	Index		**Optimized**	
	2018	2019	2018	2019
Annual Return	-8.46%	31.50%	-8.34%	24.14%
Annual Volatility	13.96%	10.23%	11.17%	6.01%
Sharpe Ratio		3.08		4.01
1-Year 95% VaR:	22.80%	16.72%	19.70%	7.70%
1-Year 95% CVaR:	32.04%	22.29%	26.15%	11.14%

The results in Table 5.9 are consistent with the performance measured for a larger number of periods. The CVaR of the optimized portfolio was reduced in all cases compared to the benchmark. Only in 10 out of 16 cases, the return of the optimized portfolio was higher. However, in 14 out of 16 cases, the Sharpe Ratio of the optimized portfolio was higher.

5.5.1 Remarks

We end this section with some general remarks. Though rooted directly in our work, these observations may have implications beyond the field of asset allocation. Nevertheless, they serve as a complement to the quantitative findings noted in this chapter.

Diversification The classic Markowitz mean-variance model is primarily based on the premise that because of different correlations between assets, the portfolio's overall volatility can be reduced by combining assets. However, we found limitations to diversification within our data set because of the high average correlation among the different assets.

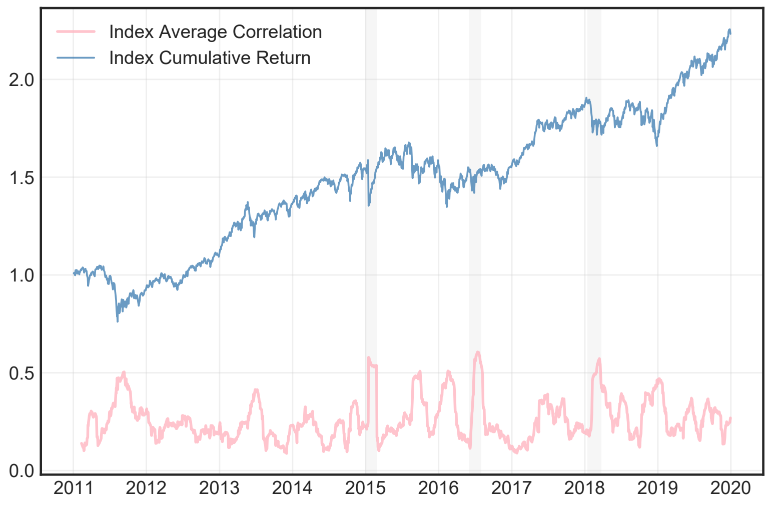


Figure 5.8: 30-Day average pairwise correlation for all 85 assets in the data set compared to the cumulative return for the years 2011 - 2019.

It can be seen in Figure 5.8 that the average correlation in our data set rose to levels above 50% during market phases with extreme downward movements. However, also under normal conditions, correlation stayed at high levels. A recent study by S&P Global [CL16, p. 4] has shown that the rolling 12-Month average correlation increased from around 25% before the turn of the millennium to around 45% for the years following the financial crisis. This increase can also be seen among other asset classes. One explanation is the development in basket trading and Exchange Traded Funds. Large baskets that make up or reflect benchmark indices are traded simultaneously, regardless of expert recommendations for their relative performance. In the light of these developments, the possibilities of diversification ”à la Markowitz” must be evaluated carefully for each model.

Short Positions Another finding from the empirical analysis is that being able to enter short positions is much more beneficial when optimizing for maximum return than it is when seeking to mitigate the overall risk. Naturally, any constraint limits the set of feasible assets or portfolios; therefore, in principle, only an unconstrained optimization would yield a real Pareto optimum. In reality, however, many investors are bound by some long-only constraint. When optimizing for minimum variance with a return target or a given Conditional Value at Risk, the results obtained without a constraint on short positions were only slightly better. In reality, this effect would often be canceled out by the adverse consequences of short positions, such as a greater amount of smaller positions or the obligation to deposit cash or assets as collateral.

6 Discussion

We have found that many models leave room for interpretation and argumentation. Moreover, the models evolve based on scientific research and practical experience. For example, many of the risk models used over the past decade were found to be inadequate after the financial crisis. In this section, we briefly discuss some of the variations, challenges, and developments that did not fit into the scope of previous chapters.

6.1 Estimation of Inputs

The estimation of input parameters, particularly the vector of total expected returns M and the corresponding covariance matrix of returns S, is one of the most critical and challenging steps in the use of mean-variance models. Following Meucci's recipe of ten sequential steps [Meu11], we point out some of the key insights that were applied when setting up the user-defined functions for this thesis. When estimating the distribution of the considered invariants, which in the case of equities are the compounded returns, an appropriate time frequency has to be chosen. As it can be seen in Figure 6.1 and Figure 6.2, different time frequencies also have different distributions. Since the mean-variance model assumes normal distribution, longer intervals would be a better fit because the distribution of monthly returns is closer resembled by the normal distribution.

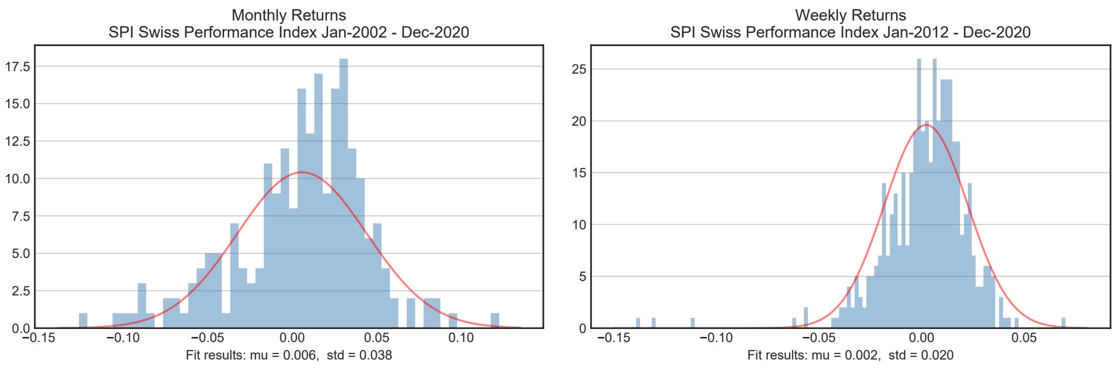


Figure 6.1: Probability density function for monthly and weekly returns (Data: Thomson Reuters)

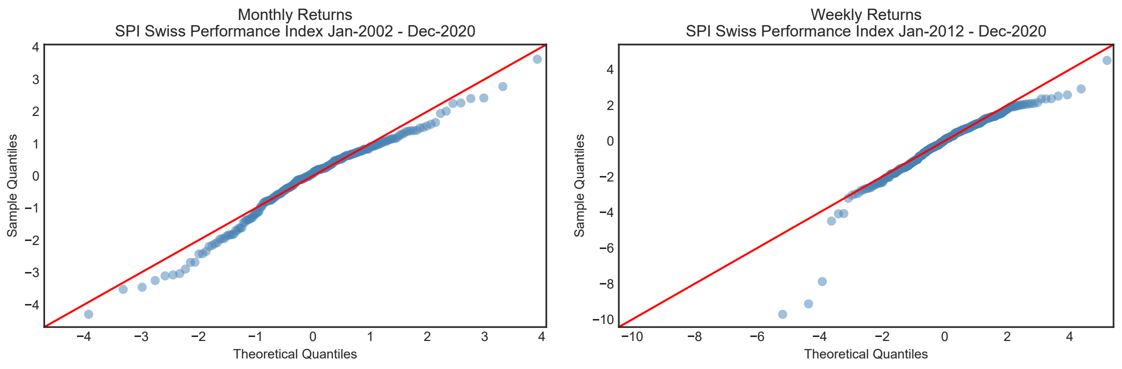


Figure 6.2: QQ Plot for monthly and weekly returns (Data: Thomson Reuters)

On the other hand, a greater number of observations is more desirable since it is more representative of the population and limits the influence of extreme observations. Another challenge arises when attempting to project the invariants to the investment horizon: The projection of the returns leads to the inevitable choice between mean returns calculated using logarithmic returns and those calculated using simple returns. From the academic literature consulted, it appears that the advantages of using logarithmic returns have been widely accepted. Primarily because using logarithmic returns is beneficial when considering multi-period returns as the continuously compounded multi-period return is simply the sum of continuously compounded single period returns.

Nevertheless, as Hudson and Gregoriou [HG15, p. 7] point out, there is no direct relationship between mean logarithmic and mean simple returns. It becomes clear from their empirical examples [HG15, p. 16] that the return under consideration in any research exercise could be defined as either the logarithmic return or the simple return, and each of these would give an internally consistent logical framework to address the problem. Nonetheless, in the context of investigations into the terminal wealth of investors, it seems clear that the simple return is the most appropriate measure to use.

Going back to the previously mentioned projection of the invariant's distribution to the investment horizon, another pitfall has to be avoided. As Meucci points out [Meu10], it is incorrect to estimate the means μ_i of the compounded returns and then project them to the investment horizon T such as $\mu(T) = T\mu$. This procedure would lead to sub-optimal allocations; among other consequences, the efficient frontier would not depend on the investment horizon. The solution to this is to aggregates the logarithmic returns across time, establish the distribution, and project it to the investment horizon T. Only after the distribution at the investment horizon is established the compounded returns $R_i(T)$ can be mapped to linear returns:

$$e^{R_i(T)} - 1 = r_i(T) \tag{6.1}$$

Therefore Meucci's [Meu10] approach is to estimate the horizon means and covariances $M(T)$ and $\Sigma(T)$ of the linear returns and use these as inputs to the mean-variance optimization. We incorporated these findings into the projection of invariants for this thesis.

6.1.1 Factor Models

Another challenge that might arise when estimating inputs to the mean-variance model comes from the number of available assets. That is because for n available assets there are $\frac{1}{2}n(n+3)$ parameters to estimate. The problem does not occur when n is a small number, such as in the examples in this thesis, but it becomes unfeasible in security selection from a basket of thousands of securities. A solution to this is offered by factor models. Single factor models go back to Sharpe's finding that every asset return can be decomposed into two components. The first component correlates with the market return by a factor β, the second component, called residual return is uncorrelated with the market. Multiple-factor models take this principle and assume that each return can be explained by a small number of different factors k. Factors usually represent economic groups such as countries or industrial sectors. For example, a "telecommunications factor" would have an exposure of 1 for telecommunications stocks and 0 for assets in other sectors. The respective factor exposures can be found by analysis of historical data and is usually sold to institutional investors by third-party analytic firms. Mathematically, the multiple-factor covariance estimate Σ_F is calculated as follows:

$$\Sigma_F = B\hat{\Sigma}B^T + D \tag{6.2}$$

With B the factor exposure matrix and $\hat{\Sigma}$ an estimate of the covariance of the vector of factor returns. The non-negative diagonal matrix D accounts for the additional variance in individual asset returns beyond that predicted by the factor model, also known as idiosyncratic risk.

Table 6.1: Single thread time used by cvxpy (with OSQP) for 3000 assets or 50 factors; Leverage limit: 2.0 applied. Source: [Boy+17].

covariance	**solve time**
single matrix	173.30 s
factor model	0.85 s

When factor models are used, the required soling time can be reduced substantially. The increase in solving speed can be seen in Table 6.1 with a reduction achieved by the factor model from n to k of 60:1, the solver is more than 200 times faster.

6.2 Estimation Risk

One of the major challenges of mean-variance models is their sensitivity to input parameters. If an asset has desirable properties for the optimization goal, the optimization routine will try to exploit this advantage by overweighting this position. Unfortunately, the actual sample mean, and sample covariance are challenging to estimate and will inevitably contain estimation errors amplified by the mean-variance optimizer. Nevertheless, there are approaches to mitigate this problem.

6.2.1 Black-Litterman Model

The Black-Litterman model [BL91] takes a Bayesian approach to asset allocation. Bayesianism introduces the concept of a priori probability, which summarizes prior knowledge together with data into a probability distribution. In the case of the Black-Litterman model, knowledge has the form of investor's views on the expected returns. The views can either be absolute (Asset A will rise by 15%) or relative (Asset C will outperform Asset B by 20%). The views must be specified in the k x 1 vector Q and are mapped to the basket of available assets via the k x n matrix P. The prior, according to Black and Litterman, is the market's estimate of returns:

$$\Pi = \psi \Sigma w_M \tag{6.3}$$

Here we calculate the total amount of risk contributed of each asset by weighting Σ with w_M, the relative market capitalization. Then by multiplying it with the market's risk premium, we obtain Π the market-implied return vector. The market-implied risk premium is quantified by the market's excess return divided by its variance:

$$\psi = \frac{r_M - r_f}{\sigma_M^2} \tag{6.4}$$

The variances of each view, that is the amount of confidence the investor expresses in his or her views, are written in a k x k diagonal covariance matrix called the confidence matrix Ω. Following Idzorek's formulation [Idz19], the Black-Litterman formula for the posterior estimate of the return is expressed as:

$$E(r)_{BL} = [\Sigma^{-1} + P^T\Omega^{-1}P]^{-1}[\Sigma^{-1}\Pi + P^T\Omega^{-1}Q] \tag{6.5}$$

with Σ the n x n covariance matrix of asset returns. And as follows for the posterior estimate of the covariance matrix:

$$\Sigma_{BL} = \Sigma + [\Sigma^{-1} + P^T\Omega^{-1}P]^{-1} \tag{6.6}$$

The estimated return is the Bayesian weighted-average of the prior and the views. Whereas the weighting is achieved according to the given confidence. This process can be intuitively understood from Figure 6.3.

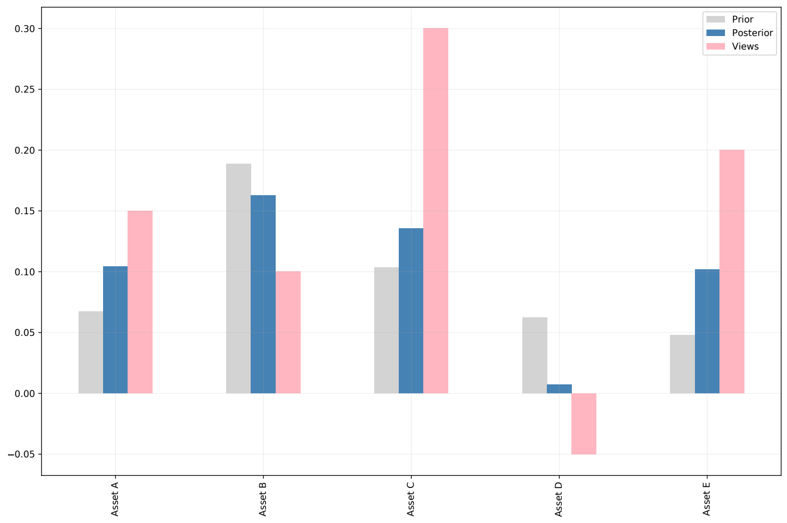


Figure 6.3: Arbitrary illustration of input and output of the Black-Litterman model.

The Bayesian estimates derived from the Black-Litterman model can then be used as expected means and covariance for the convex optimization problem. Alternatively, as Meucci [Meu05, p. 432] suggests, the Black-Litterman formulas can be incorporated into the optimization model voiding the two-step process. The advantage of the Black-Litterman approach is that it can rely solely on prior information, but it can also be based on historical information. When the confidence in the investor's views increases, the distribution shifts away from the market-implied model. In this way the Black-Litterman model addresses some of the difficulties in estimating sample mean and sample covariance and tries to solve them. However, Black and Litterman did not detail how they obtained the formula or what the different parameters mean. Thus, the model is left with a few unintuitive parameters that are difficult to use. For this reason, the described model does not suit the purpose of this thesis.

6.2.2 Shrinkage

Another method to reduce estimation error, especially from the sample covariance matrix, is called shrinkage. The general method of shrinkage in statistics was introduced by Stein [EM77] as early as 1955 and was not related to portfolio theory. In this context however it is used for covariance estimation. The shrinkage constant provides a weighted average of the sample covariance matrix and a highly structured estimator whose diagonal elements are the sample variances.

Since the sample covariance is unbiased and relatively easy to calculate, it has become an integral part of the mean-variance framework. Nevertheless, it is also well known that it contains estimation error, especially when the number of observations is comparable to the number of considered assets. On the other hand, a highly structured estimator needs to be chosen carefully and is susceptible to bias. Therefore, ideally a compromise between the two has to be found. Ledoit and Wolf [LW03a] approach this challenge as follows: They consider the sample covariance matrix S and a highly structured estimator, denoted by F. The authors then attempt to find a compromise between the two by computing the following convex linear combination:

$$\delta F + (1\delta)S \tag{6.7}$$

Where δ is a number between 0 and 1, referred to as the shrinkage constant. Regarding the shrinkage target Ledoit and Wolf propose either a single-index matrix [LW03a] or the constant correlation model [LW03b] which proves to be more accurate for optimization problems with larger (N$\geq$ 225) number of assets. In the first case, the shrinkage target is obtained from Sharpe's single-index model [Sha64] in the second case, by assuming that each pair of stocks has the same correlation.

The greater challenge lies in finding the optimal shrinkage constant δ^* that minimizes the expected distance between the shrinkage estimator and the true covariance matrix. The parameter depends on the mean and the sphericity of the eigenvalues of the covariance matrix. The mathematical derivation of the formula proposed by Ledoit and Wolf [LW03b, p. 12] for estimating δ^* can be found in the appendix of the corresponding paper. This thesis does not go into further detail on this task. The calculation of the optimal shrinkage constant following the method proposed by the authors is part of the *scikit-learn* [Ped+11] Python library. Since the library only supports the one shrinkage target, namely the diagonal covariance matrix, the other shrinkage targets had to be set up additionally. A Python adaptation of the Matlab code [LW14a] [LW14b], made available by Ledoit and Wolf, was found in different open source projects [Str11][tM19] available on Github.

Table 6.2: Optimization results obtained by different covariance matrices for the out-of-sample period from Jan 2019 until Dec 2019

	Sample Cov.	**Single Index**	**Const. Corr.**
Annual Return	20.68%	20.95%	21.22%
Annual Volatility	5.77%	5.63%	5.71%

The differently shrunk covariance matrices were applied to the data-set used throughout this thesis. A minimum variance optimization problem such as (4.3) was set up, whereas the constraint on the required return was slightly altered. The return was not set to be equal to the index return ($w^T M = r_P$) but at least high as such ($w^T M \geq r_P$). Therefore, allowing the effects of the different covariance matrices to materialize.

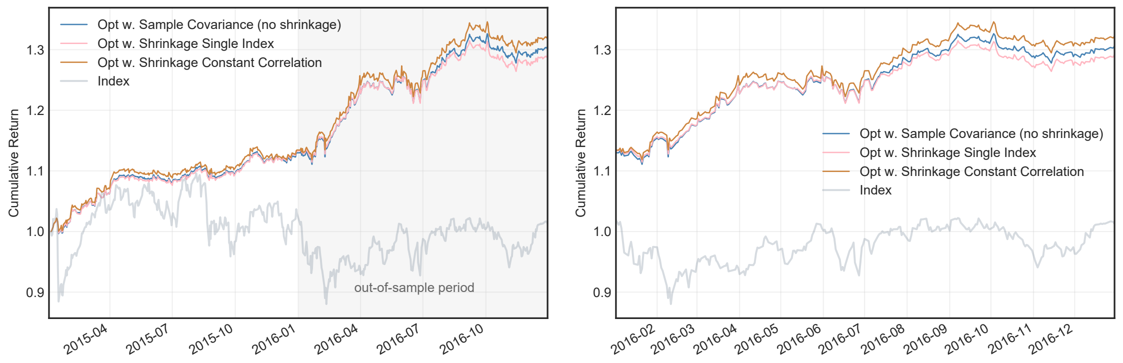


Figure 6.4: Cumulative returns of the portfolios obtained by different covariance matrices.

The results in Table 6.2 confirm the assumption that shrinking the sample covariance matrix can increase the realized out-of-sample return compared to using the sample covariance matrix as input to the optimization problem.

6.3 The Challenge of Convex Constraints

Formulating constraints in a way that ensures convexity is one of the challenges in convex optimization. This can be seen in the formulation of a convex minimum holding size constraint. Following Cornuejols et al. [CPT18, p. 167] we threat w as a semi-continuous variable, meaning that is must take a value between its minimum and maximum or zero and reformulate our constraint. By employing the (binary) boolean variable d we can implement the constraint in the form of $d \times min\_arr \leq w \leq d \times max\_arr$ where min_arr and max_arr are arrays containing the lower and upper bounds respectively.

```
constraints = [ ret == get_mu(df_index),
                cp.sum(w) == 1,
                w >= 0,
                w >= cp.multiply(d, min_arr),
                w <= cp.multiply(d, max_arr)
              ]
```

Listing 6.1: Cvxpy implementation of the minimum holding size constraint.

Now the problem becomes a Mixed-Integer Quadratic Programming problem. Within cvxpy, only ECOS [DJ14] which is an embedded conic solver, was able to solve the problem successfully, although with long solving times. It is to be assumed that commercial solvers such as CPLEX or MOSEK would tackle this problem more efficiently. Nevertheless, the usage of a commercial solver for this thesis is foregone.

6.4 Multi Period Models

The classic Markowitz mean-variance model only considers single-period optimization problems. The basic assumption is that an investor calculates the optimal weights for each asset, invests his or her wealth in fractions according to the weights, and then holds this portfolio for an undefined time. This assumption does not seem to be realistic for most investment cases. Many investors are obliged to have some portfolio rebalancing or want to do justice to changing market dynamics over time.

A first approach to creating a more dynamic model is to construct a multi-period optimization model by cascading multiple single periods. In practice this means that if we previously considered one year for our single-period model, we would now take the same model and supply it with the updated information, that is, the asset returns from the past year, to calculate the weights for the coming year. In this way, the model gets more dynamic because, for each period, it can adapt to the information collected in the previous period.

To test the validity of this approach, we took the return optimization problem (4.4) and employed the following backtesting: At the end of each calendar year, the past returns and distributions were used to calculate the optimal weights. These weights were then used to rebalance the portfolio for the following year. In this way, the optimized out-of-sample returns shown in Figure 6.5 were obtained. Even though we were able to achieve out-of-sample excess returns for various optimization goals and with different constraints, this is still a very primitive approach. Two significant factors that are not being considered in this approach are transaction costs and time-varying forecasts. These parameters are better dealt with in an actual multi-period setting, such as the approach proposed by Boyd

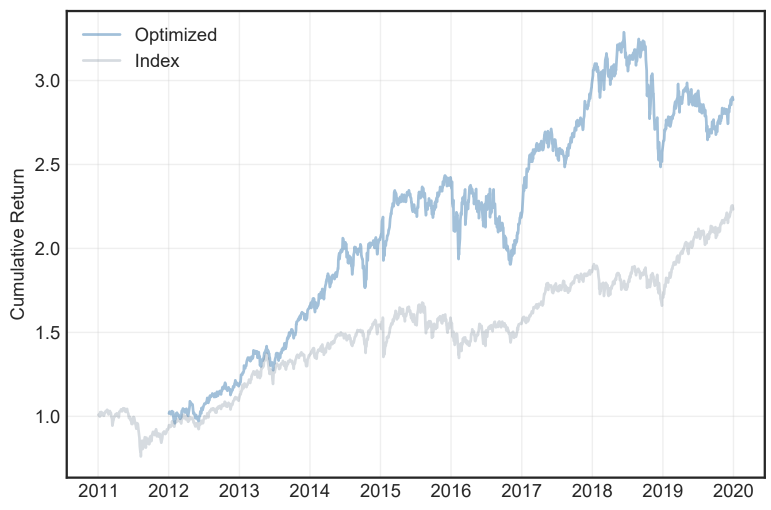


Figure 6.5: Cumulative returns form single period optimization for consecutive annual periods.

et al. [Boy+17]. The authors optimize the post-trade portfolio (h_t^+), which depends on the dollar values of the trades (u_t) and the total value of the portfolio from the previous period (h_t). In this way they can incorporate transaction and holding costs directly into the optimization model.

7 Conclusion

The main focus of this thesis was to engineer practical tools for asset allocation and implement them with Python. We approached this by first outlining a general understanding of the basic mean-variance model. Further, a discussion on developments in asset allocation with analytical and numerical examples has been provided. We took the two primary goals of keeping the routines simple but still obtaining precise results into account at all stages of the process.

The historical data was processed and projected to the investment horizon in a way that maintained the distribution of the underlying invariants. This thesis provides a convex formulation of the most common optimization objectives: Reducing volatility, enhancing returns, and improving Sharpe ratio; we also optimized for different objectives while accounting for risk aversion and Expected Shortfall. The problems implemented in this thesis include various equality and inequality constraints. Estimation error was successfully reduced for many problems by shrinking the sample covariance matrix toward a structured estimate. The proposed and implemented routines work reliably and solve with short computation times using open-source solvers for linear programming and convex optimization. Several custom-built functions support the comprehensibility of inputs and results through visualizations that can be plotted right from the code. In particular, we proposed to optimize for Expected Shortfall rather than variance. We were able to verify this approach analytically and numerically. Combining many of the findings and tools, we successfully constructed a portfolio of Swiss stocks which beat its benchmark index over eight consecutive years, regarding Conditional Value at Risk, in out-of-sample back-testing.

We addressed the problem of lacking openly accessible, efficient routines by creating a set of Python functions around the embedded modeling language cvxpy. In the next step, a general user interface could be built on top of these functions to ensure usability for an even broader audience. Moreover, the models could be further refined by evaluating results from larger, more heterogeneous data sets or by including other asset classes. From this, a multi-period model that takes transaction costs into account could be built.

This thesis has shown two strong points: First, most optimization problems can be expressed in convex form. They can be implemented and solved efficiently using different

Python modules to create portfolios from real-world data. Secondly, that even in an environment with high correlation within and among asset classes, it is still possible to achieve a competitive return with a lower expected shortfall and lower excess risk than the market over multiple periods.

Bibliography

[Mar52] Harry Markowitz. “Portfolio Selection”. In: *The Journal of Finance* 7.1 (Mar. 1952), p. 77. issn: 00221082. doi: 10.2307/2975974. url: https://www.jstor.org/stable/2975974?origin=crossref (visited on 02/27/2021).

[Tob58] J. Tobin. “Liquidity Preference as Behavior Towards Risk”. en. In: *The Review of Economic Studies* 25.2 (Feb. 1958), p. 65. issn: 00346527. doi: 10.2307/2296205. url: https://academic.oup.com/restud/article-lookup/doi/10.2307/2296205 (visited on 04/28/2021).

[Sha64] William F. Sharpe. “Capital Asset Prices: A Theory of Market Equilibrium under Conditions of Risk”. en. In: *The Journal of Finance* 19.3 (Sept. 1964), pp. 425–442. issn: 00221082. doi: 10.1111/j.1540-6261.1964.tb02865.x. url: http://doi.wiley.com/10.1111/j.1540-6261.1964.tb02865.x (visited on 03/13/2021).

[HG15] Robert S. Hudson and Andros Gregoriou. “Calculating and comparing security returns is harder than you think: A comparison between logarithmic and simple returns”. en. In: *International Review of Financial Analysis* 38 (Mar. 2015), pp. 151–162. issn: 10575219. doi: 10.1016/j.irfa.2014.10.008. url: https://linkinghub.elsevier.com/retrieve/pii/S1057521914001380 (visited on 05/20/2021).

[BS73] Fischer Black and Myron Scholes. “The Pricing of Options and Corporate Liabilities”. en. In: *Journal of Political Economy* 81.3 (May 1973), pp. 637–654. issn: 0022-3808, 1537-534X. doi: 10.1086/260062. url: https://www.journals.uchicago.edu/doi/10.1086/260062 (visited on 03/13/2021).

[NCT13] Paul Newbold, William L. Carlson, and Betty M. Thorne. *Statistics for business and economics*. eng. 8. ed., global ed. Always learning. OCLC: 796196139. Boston, Mass.: Pearson, 2013. isbn: 978-0-273-76706-0.

[Pei94] Amado Peiró. “The distribution of stock returns: international evidence”. en. In: *Applied Financial Economics* 4.6 (Dec. 1994), pp. 431–439. issn: 0960-3107, 1466-4305. doi: 10.1080/758518675. url: http://www.tandfonline.com/doi/abs/10.1080/758518675 (visited on 03/09/2021).

[BGP20] Eric Benhamou, Beatrice Guez, and Nicolas Paris. “Three remarkable properties of the Normal distribution”. en. In: *arXiv:1810.01768 [math]* (July 2020).

arXiv: 1810.01768. url: `http://arxiv.org/abs/1810.01768` (visited on 03/13/2021).

[Meu05] Attilio Meucci. *Risk and asset allocation*. Springer finance. OCLC: ocm57166378. Berlin ; New York: Springer, 2005. isbn: 978-3-540-22213-2.

[Mer72] Robert C. Merton. “An Analytic Derivation of the Efficient Portfolio Frontier”. en. In: *The Journal of Financial and Quantitative Analysis* 7.4 (Sept. 1972), p. 1851. issn: 00221090. doi: `10.2307/2329621`. url: `https://www.jstor.org/stable/2329621?origin=crossref` (visited on 03/29/2021).

[VM44] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior*. Princeton classic editions. OCLC: ocm78989126. Princeton, N.J. ; Woodstock: Princeton University Press, 1944. isbn: 978-0-691-13061-3.

[KT79] Daniel Kahneman and Amos Tversky. “Prospect Theory: An Analysis of Decision under Risk”. In: *Econometrica* 47.2 (Mar. 1979), p. 263. issn: 00129682. doi: `10.2307/1914185`. url: `https://www.jstor.org/stable/1914185?origin=crossref` (visited on 04/02/2021).

[Arr71] Kenneth J. Arrow. *Essays in the theory of risk-bearing*. Markham economics series. Chicago: Markham Pub. Co, 1971. isbn: 978-0-8410-2001-6.

[Pra64] John W. Pratt. “Risk Aversion in the Small and in the Large”. In: *Econometrica* 32.1/2 (Jan. 1964), p. 122. issn: 00129682. doi: `10.2307/1913738`. url: `https://www.jstor.org/stable/1913738?origin=crossref` (visited on 04/03/2021).

[BV04] Stephen P. Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge, UK ; New York: Cambridge University Press, 2004. isbn: 978-0-521-83378-3.

[Wri04] Margaret H. Wright. “The interior-point revolution in optimization: History, recent developments, and lasting consequences”. en. In: *Bulletin of the American Mathematical Society* 42.01 (Sept. 2004), pp. 39–57. issn: 0273-0979. doi: `10.1090/S0273-0979-04-01040-7`. url: `http://www.ams.org/journal-getitem?pii=S0273-0979-04-01040-7` (visited on 04/17/2021).

[Mar70] Harry Markowitz. *Portfolio selection: efficient diversification of investments*. eng. Second printing. Monograph / Cowles Foundation for Research in Economics at Yale University 16. OCLC: 214279493. New Haven London: Yale University Press, 1970. isbn: 978-0-300-01372-6.

[Str91] Gilbert Strang. *Calculus*. eng. OCLC: 831865786. Wellesley, Mass: Wellesley-Cambridge Press, 1991. isbn: 978-0-9614088-2-4.

[CPT18] Gerard Cornuejols, Javier Francisco Peña, and Reha Tütüncü. *Optimization methods in finance*. Second edition. Cambridge, United Kingdom ; New York, NY: Cambridge University Press, 2018. isbn: 978-1-107-05674-9.

[SP94] Frank A. Sortino and Lee N. Price. "Performance Measurement in a Downside Risk Framework". en. In: *The Journal of Investing* 3.3 (Aug. 1994), pp. 59–64. issn: 1068-0896, 2168-8613. doi: `10.3905/joi.3.3.59`. url: `http://joi.pm-research.com/lookup/doi/10.3905/joi.3.3.59` (visited on 04/28/2021).

[Est07] Javier Estrada. "Mean-Semivariance Optimization: A Heuristic Approach". en. In: *SSRN Electronic Journal* (2007). issn: 1556-5068. doi: `10.2139/ssrn.1028206`. url: `http://www.ssrn.com/abstract=1028206` (visited on 04/27/2021).

[Che09] Sergei Vasilievich Cheremushkin. "Why D-CAPM is a Big Mistake? The Incorrectness of the Cosemivariance Statistics". en. In: *SSRN Electronic Journal* (2009). issn: 1556-5068. doi: `10.2139/ssrn.1336169`. url: `http://www.ssrn.com/abstract=1336169` (visited on 04/27/2021).

[JPM96] J.P.Morgan/Reuters. "RiskMetrics Technical Manual". en. In: Fourth Edition (Dec. 1996), p. 296. url: `https://www.msci.com/documents/10199/5915b101-4206-4ba0-aee2-3449d5c7e95a` (visited on 05/18/2021).

[Art+99] Philippe Artzner et al. "Coherent Measures of Risk". en. In: *Mathematical Finance* 9.3 (July 1999), pp. 203–228. issn: 0960-1627, 1467-9965. doi: `10.1111/1467-9965.00068`. url: `http://doi.wiley.com/10.1111/1467-9965.00068` (visited on 05/18/2021).

[RU00] R. Tyrrell Rockafellar and Stanislav Uryasev. "Optimization of conditional value-at-risk". en. In: *The Journal of Risk* 2.3 (2000), pp. 21–41. issn: 14651211. doi: `10.21314/JOR.2000.038`. url: `http://www.risk.net/journal-of-risk/technical-paper/2161159/optimization-conditional-value-risk` (visited on 05/17/2021).

[Hil19] Yves J. Hilpisch. *Python for finance: mastering data-driven finance*. Second edition. Sebastopol, CA: O'Reilly Media, 2019. isbn: 978-1-4920-2433-0.

[Duh09] Charles Duhigg. "Stock Traders Find Speed Pays, in Milliseconds". In: *The New York Times* (June 2009). url: `https://www.nytimes.com/2009/07/24/business/24trading.html`.

[Har+20] Charles R. Harris et al. "Array programming with NumPy". en. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. issn: 0028-0836, 1476-4687. doi: `10.1038/s41586-020-2649-2`. url: `http://www.nature.com/articles/s41586-020-2649-2` (visited on 05/03/2021).

[DS16] Steven Diamond and Stephen Boyd. “CVXPY: A Python-Embedded Modeling Language for Convex Optimization”. en. In: *Journal of Machine Learning Research* (Apr. 2016), p. 5. url: `https://web.stanford.edu/~boyd/papers/pdf/cvxpy_paper.pdf`.

[Agr+18] Akshay Agrawal et al. “A rewriting system for convex optimization problems”. en. In: *Journal of Control and Decision* 5.1 (Jan. 2018), pp. 42–60. issn: 2330-7706, 2330-7714. doi: `10.1080/23307706.2017.1397554`. url: `https://www.tandfonline.com/doi/full/10.1080/23307706.2017.1397554` (visited on 05/03/2021).

[Ste+20] Bartolomeo Stellato et al. “OSQP: an operator splitting solver for quadratic programs”. en. In: *Mathematical Programming Computation* 12.4 (Dec. 2020), pp. 637–672. issn: 1867-2949, 1867-2957. doi: `10.1007/s12532-020-00179-2`. url: `http://link.springer.com/10.1007/s12532-020-00179-2` (visited on 05/03/2021).

[Ban+17] Goran Banjac et al. “Embedded code generation using the OSQP solver”. en. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. Melbourne, VIC: IEEE, Dec. 2017, pp. 1906–1911. isbn: 978-1-5090-2873-3. doi: `10.1109/CDC.2017.8263928`. url: `http://ieeexplore.ieee.org/document/8263928/` (visited on 05/03/2021).

[EA68] John L. Evans and Stephen H. Archer. “DIVERSIFICATION AND THE REDUCTION OF DISPERSION: AN EMPIRICAL ANALYSIS*”. en. In: *The Journal of Finance* 23.5 (Dec. 1968), pp. 761–767. issn: 00221082. doi: `10.1111/j.1540-6261.1968.tb00315.x`. url: `http://doi.wiley.com/10.1111/j.1540-6261.1968.tb00315.x` (visited on 05/10/2021).

[Sta87] Meir Statman. “How Many Stocks Make a Diversified Portfolio?” In: *The Journal of Financial and Quantitative Analysis* 22.3 (Sept. 1987), p. 353. issn: 00221090. doi: `10.2307/2330969`. url: `https://www.jstor.org/stable/2330969?origin=crossref` (visited on 05/10/2021).

[CLM] John Y Campbell, Martin Lettau, and Burton G Malkiel. “Have Individual Stocks Become More Volatile? An Empirical Exploration of Idiosyncratic Risk”. en. In: *The Journal of Finance* (), p. 44.

[DLR06] Dale L. Domian, David A. Louton, and Marie D. Racine. “Diversification in Portfolios of Individual Stocks: 100 Stocks are Not Enough”. en. In: *SSRN Electronic Journal* (2006). issn: 1556-5068. doi: `10.2139/ssrn.906686`. url: `http://www.ssrn.com/abstract=906686` (visited on 05/10/2021).

[Meu11] Attilio Meucci. “‘The Prayer’ Ten-Step Checklist for Advanced Risk and Portfolio Management”. en. In: *SSRN Electronic Journal* (May 2011). issn:

1556-5068. doi: `10.2139/ssrn.1753788`. url: `http://www.ssrn.com/abstract=1753788` (visited on 05/17/2021).

[CL16] Fei Mei Chan and Craig J. Lazzara. "The Dispersion-Correlation Map". en. In: *S&P Global Research* (2016), p. 12. url: `https://www.spglobal.com/spdji/en/documents/research/research-the-dispersion-correlation-map.pdf` (visited on 05/27/2021).

[Meu10] Attilio Meucci. "Linear vs. Compounded Returns". en. In: (May 2010), p. 5. url: `http://ssrn.com/abstract=1586656`.

[Boy+17] Stephen Boyd et al. "Multi-Period Trading via Convex Optimization". In: *arXiv:1705.00109 [math, q-fin]* (Apr. 2017). arXiv: 1705.00109. url: `http://arxiv.org/abs/1705.00109` (visited on 11/01/2020).

[BL91] Fischer Black and Robert B Litterman. "Asset Allocation: Combining Investor Views with Market Equilibrium". en. In: *The Journal of Fixed Income* 1.2 (Sept. 1991), pp. 7–18. issn: 1059-8596, 2168-8648. doi: `10.3905/jfi.1991.408013`. url: `http://jfi.pm-research.com/lookup/doi/10.3905/jfi.1991.408013` (visited on 05/20/2021).

[Idz19] Thomas Idzorek. "A Step-By-Step Guide to the Black-Litterman Model Incorporating User-specified Confidence Levels". en. In: *SSRN Electronic Journal* (2019). issn: 1556-5068. doi: `10.2139/ssrn.3479867`. url: `https://www.ssrn.com/abstract=3479867` (visited on 05/20/2021).

[EM77] Bradley Efron and Carl Morris. "Stein's Paradox in Statistics". In: *Scientific American* 236.5 (May 1977), pp. 119–127. issn: 0036-8733. doi: `10.1038/scientificamerican0577-119`. url: `https://www.scientificamerican.com/article/steins-paradox-in-statistics` (visited on 05/24/2021).

[LW03a] Olivier Ledoit and Michael Wolf. "Improved estimation of the covariance matrix of stock returns with an application to portfolio selection". en. In: *Journal of Empirical Finance* 10.5 (Dec. 2003), pp. 603–621. issn: 09275398. doi: `10.1016/S0927-5398(03)00007-0`. url: `https://linkinghub.elsevier.com/retrieve/pii/S0927539803000070` (visited on 05/22/2021).

[LW03b] Olivier Ledoit and Michael NMI Wolf. "Honey, I Shrunk the Sample Covariance Matrix". en. In: *SSRN Electronic Journal* (2003). issn: 1556-5068. doi: `10.2139/ssrn.433840`. url: `http://www.ssrn.com/abstract=433840` (visited on 05/22/2021).

[Ped+11] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[LW14a] Olivier Ledoit and Michael Wolf. *covMarket.m*. Apr. 2014. url: `https://www.econ.uzh.ch/dam/jcr:ffffffff-935a-b0d6-0000-0000648dfc98/covMarket.m.zip`.

[LW14b] Olivier Ledoit and Michael Wolf. *covCor.m*. Apr. 2014. url: `https://www.econ.uzh.ch/dam/jcr:ffffffff-935a-b0d6-ffff-ffffde5e2d4e/covCor.m.zip`.

[Str11] Jason Strimpel. *covshrink.py*. Nov. 2011. url: `https://git.io/JsNSY`.

[tM19] tuantp7 and Robert Martin. *risk_models.py*. Mar. 2019. url: `https://git.io/JZIZS`.

[DJ14] Alexander Domahidi and Juan Jerez. *FORCES Professional*. Published: Embotech AG, url=https://embotech.com/FORCES-Pro. 2014.

[Sha66] William F. Sharpe. "Mutual Fund Performance". en. In: *The Journal of Business* 39.S1 (Jan. 1966), p. 119. issn: 0021-9398, 1537-5374. doi: `10.1086/294846`. url: `https://www.jstor.org/stable/2351741` (visited on 04/02/2021).

A Mathematical Formulation

A.1 Maximizing Sharpe Ratio

The Sharpe [Sha66] ratio of a given portfolio w is the ratio of its expected return to its volatility (standard deviation) of returns. Following the formulation thought in the *Columbia University IEOR E4500 Course* we reformulate the natural formulation of the Sharpe Ratio as a standard convex quadratic program.

The natural formulation for this problem is the following:

$$\begin{aligned} \text{maximize} \quad & \frac{w^T M}{\sqrt{w^T \Sigma w}} \\ \text{subject to} \quad & w^T \mathbb{1}_m = 1 \\ & w_i \geq 0 \end{aligned} \tag{A.1}$$

Notice that in this case unless the risk-free rate is 0 M is not the vector of expected absolute returns but vector of expected relative returns.

The assumption we make is: $w^T M > 0$, satisfying all constraints.

Since $\sum_{i=1}^{N} w_i = 1$,

$$f(w) = \frac{w^T M}{\sqrt{w^T \Sigma w}} = \frac{w^T M \sum_{i=1}^{N} w_i}{\sqrt{w^T \Sigma w}} \tag{A.2}$$

We observe that for any vector w with $\sum_{i=1}^{N} w_i = 1$, and any scalar $\lambda > 0, f(\lambda w) = f(w)$. Further the assumption that $w^T M > 0$ for some feasible w implies that we can choose $\kappa > 0$ such that $z^T M = 1$. Using this together with first second assumption, we consider the problem:

$$\begin{aligned}
\text{maximize} \quad & \frac{1}{\sqrt{z^T \Sigma z}} \\
\text{subject to} \quad & z^T M = 1 \\
& \sum_{i=1}^{N} z_i = \kappa \\
& \kappa \geq 0 \\
& w_i = \frac{z_i}{\kappa}
\end{aligned} \tag{A.3}$$

which is equivalent to this standard quadratic program.:

$$\begin{aligned}
\text{minimize} \quad & z^T \Sigma z \\
\text{subject to} \quad & z^T M = 1 \\
& \sum_{i=1}^{N} z_i = \kappa \\
& \kappa \geq 0 \\
& w_i = \frac{z_i}{\kappa}
\end{aligned} \tag{A.4}$$

B Python Code

This appendix provides a list of the most important user-defined functions and optimization routines created for this thesis. For convenience the helper functions listed in B.2 are located in a separate module which is imported for every optimization routine.

B.1 Datareader

```python
def eikon_datareader_TR(RICs, Start, End, Frequency):
"""
Extracts Total Return Data from the Thomson Reuters API for
a List of RICs. Data for individual assets is stored in
separate columns and returned as DataFrame.
RICs : List of Thomson Reuters Instrument Code
Start: Last day of requested time series yyy-mm-dd
End: First day of requested time series yyy-mm-dd
Frequency: D, W, M, Q, Y
"""

    df, err = ek.get_data([RICs[0]],fields=[
        "TR.TotalReturn(SDate={},EDate={},Frq={}).date"
        .format(Start, End, Frequency)])
    df_store = pd.DataFrame(df["Date"].str[:10])

    for i in range(len(RICs)):
        df, err = ek.get_data([RICs[i]],fields=[
            "TR.TotalReturn(SDate={},EDate={},Frq={}).date"
            .format(Start, End, Frequency),
            "TR.TotalReturn(SDate={},EDate={},Frq={})"
            .format(Start, End, Frequency)])

        df_returns_temp = pd.DataFrame(df["Total Return"])
        df_returns_temp.columns = [df.iloc[1][0]]
        df_store = pd.concat([df_store, df_returns_temp], axis=1)
        i += 1

    return df_store
```

B.2 Helper Functions

```python
import numpy as np
import pandas as pd
import cvxpy as cp
import cvxopt
import matplotlib.pyplot as plt
import copy

def process_csv(path,log=False):
    """
    Loads CSV into DataFrame and formats with Date as Index
    path: Path of csv file with simple net returns;
    First Column are dates; Index are RIC
    """
    df = pd.read_csv (path)
    df["Date"] = pd.to_datetime(df["Date"], format="\%Y-\%m-\%d")
    #Fotmat Dates as Datetime
    df = df.set_index("Date") #Set Dates Column as DataFrame Index
    if log:
        df = np.log1p(df) #Simple Net Returns to Log total Return
    df = df.sort_values(by='Date')
    df = df.dropna(axis=0, how="any")

    return df

def get_mu(timeseries):
    """
    Calculates annualized mean according to Meucci's proposal for
    the projection of invariants (https://ssrn.com/abstract=1586656).
    Returns the annualized mean of the input
    timeseries: pandas DataFrame containing daily returns
    """
    return np.exp(np.log1p(timeseries).mean()*252)-1

def _mu(timeseries):
    """
    Local performance function; equivalent to get_mu.
    Returns the annualized mean of the input
    timeseries: pandas dataframe containing daily returns
    """
    return np.exp(np.log1p(timeseries).mean()*252)-1

def get_cov(timeseries, annualized = True):
    """
    Calculates annualized covariance (Matrix).
    timeseries: pandas DataFrame containing daily returns
    annualized: defaults to 252 (number of trading days per year)
    """
    S = timeseries.cov()
    if annualized:
        S = S * 252

    return S

def get_std(timeseries, annualized = True):
    """
    Calculates annualized standard deviation.
    timeseries: pandas DataFrame containing daily returns
    annualized: defaults to 252 (number of trading days per year)
    """
    std = timeseries.std()
    if annualized:
        std = std * np.sqrt(252)

    return std

def get_var(timeseries, annualized = True):
    """
    Calculates annualized variance.
    timeseries: pandas DataFrame containing daily returns
    annualized: defaults to 252 (number of trading days per year)
    """
    var = timeseries.var()
    if annualized:
        var = var * 252

    return var

def get_hist_VaR(timeseries, alpha=5):
    """
    Calculates (negative) historic Value at Risk
    timeseries: pandas DataFrame containing daily returns
    alpha: Confidence
    """

    return -np.percentile(timeseries, alpha)

def get_hist_CVaR(timeseries, alpha=5):
    """
```

```python
    Calculates (negative) historic Conditional Value at Risk.
    timeseries: pandas DataFrame containing daily returns
    alpha: Confidence
    """
    belowVaR = timeseries <= np.percentile(timeseries, alpha)
    return -timeseries[belowVaR].mean()

def get_weighted_returns(assets, weights):
    """
    Calculates returns relative to given weights.
    assets: pandas DataFrame containing daily returns
    weights: vector of weights; has to be same length as number of assets
    """
    portfolio_rets = assets@weights
    return portfolio_rets

def get_mvp_weights(timeseries):
    """
    Calculates weights of the (optimized) MVP.
    timeseries: pandas DataFrame containing daily returns
    """
    w = cp.Variable(len(timeseries.columns),)
    ret = w.T@_mu(timeseries)
    sigma_square = cp.quad_form(w, get_cov(timeseries))
    objective = cp.Minimize(sigma_square)
    constraints = [
                    cp.sum(w) == 1,
                    w >= 0]

    prob = cp.Problem(objective, constraints)
    prob.solve()
    w_mvp = w.value

    return w_mvp

def get_max_ret_weights(timeseries):
    """
    Returns a vector of weights with the same length as the timeseries,
    where all the weight is allocated to the asset with the maximum mean.
    timeseries: pandas DataFrame containing daily returns
    """
    w_max = np.zeros(len(timeseries.columns))
    w_max[_mu(timeseries).argmax()] = 1

    return w_max
```

```python

def get_summary(weights, timeseries, pct=True):
    """
    Returns selected summary statistics on the portfolio.
    weights: vector of (portfolio) weights
    timeseries: pandas DataFrame containing daily returns
    """
    w_opt = weights
    if pct:
        d = [
        ["### Portfolio Summary ###",""],
        ["","" ],
        ["Number of Assets:", "{:}"
        .format(np.count_nonzero(weights.round(6)+0.0))],
        ["Expected Annual Return:", "{:.3f}\%"
        .format((w_opt.T@_mu(timeseries))*100)],
        ["Annual Volatility:", "{:.3f}\%".format(np.sqrt(
        w_opt.T@get_cov(timeseries)@w_opt)*100)],
        ["Sharpe Ratio:", (w_opt.T@_mu(timeseries)/np.sqrt(
        w_opt.T@get_cov(timeseries)@w_opt)).round(3)],
        ["Portfolio Leverage:",
        get_port_leverage(weights).round(3)],
        ["1-Year 95\% VaR:", "{:.3f}\%".format(100*np.sqrt(252)*
        get_hist_VaR(get_weighted_returns(timeseries, weights)))],
        ["1-Year 95\% CVaR:", "{:.3f}\%".format(100*np.sqrt(252)*
        get_hist_CVaR(get_weighted_returns(timeseries, weights)))],
        ["1-Day 95\% VaR:", "{:.3f}\%".format(100*
        get_hist_VaR(get_weighted_returns(timeseries, weights)))],
        ["1-Day 95\% CVaR:", "{:.3f}\%".format(100*
        get_hist_CVaR(get_weighted_returns(timeseries, weights)))],
        ["","" ],
        ]

        for v in d:
            title, value = v
            print ("{:<25} {:<30}".format( title, value))

def get_top_weights(number, timeseries, weights):
    """
    Returns n(umber) of the assets with the biggest weights.
    number: Integer value defining the number of weights to be returned
    timeseries: pandas DataFrame containing daily returns
    weights: vector of (portfolio) weights
    """
    for _ in range(number):
        print(timeseries.iloc[:, (-weights).argsort()[_]]
        .name,weights[(-weights).argsort()[_]].round(5))

def get_bot_weights(number, timeseries, weights):
    """
    Returns n(umber) of the assets with the smallest weights.
    number: Integer value defining the number of weights to be returned
    timeseries: pandas DataFrame containing daily returns
    weights: vector of (portfolio) weights
    """
    number = number-1
    for _ in range(number+1):
        print(timeseries.iloc[:, (weights).argsort()[number-_]]
        .name,weights[(weights).argsort()[number-_]].round(5))

def get_port_leverage(weights):
    """
    Calculates leverage measured in excess of 1
    weights: vector of (portfolio) weights
    """
    return np.abs(weights).sum().round(5) -1
```

B.3 Generate Random Portfolios

```python
def gen_weights(n,long_only = True):
    """
    Returns a vector of n weights.
    n: Number of weights (portfolios).
    """
    if long_only:
        #rand for only positive values
        r = np.random.rand(n)
        return r / sum(r)
    else:
        #randn for positive and negative
        r = np.random.randn(n)
        return r / sum(r)

def random_portfolio(timeseries):
    """
    Returns mean, standard deviation and asset weights
    for a random portfolio.
    timeseries: pandas DataFrame containing daily returns
    """
    M = pd.DataFrame(timeseries.mean()) #Return Vector (3x1)
    w = pd.DataFrame(gen_weights(timeseries.shape[1],0),
    index=['A', 'B', 'C']) #Weight Vector (3x1)
    S = pd.DataFrame(timeseries.cov()) #CovMatrix

    mu = w.T@M
    sigma = np.sqrt(w.T@S@w)

    if sigma.iloc[0,0] > 5: #cleaning up outliers for plotting
        return random_portfolio(timeseries)
    return mu, sigma , w

def random_portfolio_long(timeseries):
    """
    Returns mean, standard deviation and asset weights
    for a random (long-only) portfolio.
    timeseries: pandas DataFrame containing daily returns for 3 assets
    """
    M = pd.DataFrame(timeseries.mean()) #Return Vector (3x1)
    w = pd.DataFrame(gen_weights(timeseries.shape[1],1),
    index=['A', 'B', 'C']) #Weight Vector (3x1)
    S = pd.DataFrame(timeseries.cov()) #CovMatrix

    mu = w.T@M
    sigma = np.sqrt(w.T@S@w)

    if sigma.iloc[0,0] > 5: #cleaning up outliers for plotting
        return random_portfolio(timeseries)
    return mu, sigma , w
```

B.4 Variance Optimization Problem

```python
# Minimize portfolio variance for given return (ret)
w = cp.Variable(len(df_assets.columns),)
ret = w.T@get_mu(df_assets)
sigma_square = cp.quad_form(w, get_cov(df_assets))
objective = cp.Minimize(sigma_square)
constraints = [
                ret == get_mu(df_index).mean(),
                cp.sum(w) == 1,
                w >= 0
                ]

```

```python
prob = cp.Problem(objective, constraints)
prob.solve(verbose = False)
get_summary(w.value,df_assets)
```

B.5 Return Optimization Problem

```python
# Maximize portfolio return for given variance limit (sigma_square)
w = cp.Variable(len(df_assets.columns),)
ret = w.T@get_mu(df_assets)
sigma_square = cp.quad_form(w, get_cov(df_assets))
objective = cp.Maximize(ret)
constraints = [
                sigma_square <= get_var(df_index),
                cp.sum(w) == 1,
                w >= 0
                ]

prob = cp.Problem(objective, constraints)
prob.solve(verbose = False)
get_summary(w.value,df_assets)
```

B.6 Sharpe Optimization Problem

```python
# Minimize portfolio Sharpe for given return (ret)
z = cp.Variable(len(df_assets.columns),) #transformed variable
k = cp.Variable() #scalar (variable)
ret = z.T@get_mu(df_assets)
sigma_square = cp.quad_form(z, get_cov(df_assets))
objective = cp.Minimize(sigma_square)
constraints = [
                ret == 1 #adjustable return target constraint
                cp.sum(z) == k,
                z >= 0,
                k >= 0
                ]

prob = cp.Problem(objective, constraints)

prob.solve(verbose = False)
w = (z.value / k.value)
get_summary(w,df_assets)
```

B.7 Minimum CVaR

```python
# Minimize absolute portfolio CVaR
w = cp.Variable(len(df_assets.columns),)
alpha = cp.Variable()
u = cp.Variable(len(df_assets)) #auxiliary var. (see Rockafellar & Uryasev)
beta = 0.95 #adjustable parameter (confidence)

objective = cp.Minimize(alpha + 1.0 / (len(df_assets) *
(1 - beta)) * cp.sum(u))
constraints = [
                cp.sum(w) == 1,
                u >= 0.0,
                w >= 0.0,
                df_assets.values@w + alpha + u >= 0.0,
                ]

prob = cp.Problem(objective, constraints)
prob.solve(verbose = False)
get_summary(w.value,df_assets)
```

B.8 Minimum CVaR for Return Target

```python
# Minimize portfolio CVaR for given return limit (ret)
w = cp.Variable(len(df_assets.columns),)
alpha = cp.Variable()
u = cp.Variable(len(df_assets)) #auxiliary var. (see Rockafellar & Uryasev)
beta = 0.95 #adjustable parameter (confidence)

return_target = get_mu(df_index)
ret = w.T@get_mu(df_assets)

objective = cp.Minimize(alpha + 1.0 / (len(df_assets) *
(1 - beta)) * cp.sum(u))
constraints = [
                cp.sum(w) == 1,
                u >= 0.0,
                w >= 0.0,
                df_assets.values@w + alpha + u >= 0.0,
                ret >= return_target
                ]

prob = cp.Problem(objective, constraints)
```

```python
prob.solve(verbose = False)
get_summary(w.value,df_assets)
```

B.9 Maximum Return for CVaR Target

```python
# Maximize portfolio Return for given CVaR limit (cvar_target)
w = cp.Variable(len(df_assets.columns),)
alpha = cp.Variable()
u = cp.Variable(len(df_assets)) #auxiliary var. (see Rockafellar & Uryasev)
beta = 0.95 #adjustable parameter (confidence)
cvar = alpha + 1.0 / (len(df_assets) * (1 - beta)) * cp.sum(u)
cvar_target = 0.008 #adjustable parameter

objective = cp.Maximize(w.T@get_mu(df_assets))
constraints = [
                cp.sum(w) == 1,
                u >= 0.0,
                w >= 0.0,
                df_assets.values@w + alpha + u >= 0.0,
                cvar <= cvar_target
                ]

prob = cp.Problem(objective, constraints)
prob.solve(verbose = False)
get_summary(w.value,df_assets)
```

B.10 Risk Aversion Optimization

```python
# Minimize portfolio risk adjusted return.
w = cp.Variable(len(df_assets.columns),)
ret = w.T@get_mu(df_assets)
sigma_square = cp.quad_form(w, get_cov(df_assets))
gamma = cp.Parameter(nonneg=True) #cvxpy parameter; modified later
objective = cp.Maximize(ret - gamma*sigma_square)
constraints = [
                cp.sum(w) == 1,
                w >= 0
                ]

prob = cp.Problem(objective, constraints)

```

```python
prob.solve(verbose = False)
get_summary(w.value,df_assets)
```

B.11 Efficient Frontier

```python
# Calculates efficient frontier from any optimization problem.
gamma.value = 0
w_opt = np.ones(len(df_assets.columns))
mu_front = []
sigma_front = []
gamma_front = []
weight_front = pd.DataFrame()

# Iterate as long as variance is more than 5*10^-4 away from MVP
while (np.sqrt((w_opt.T@get_cov(df_assets)@w_opt)) -
np.sqrt((get_mvp_weights(df_assets).T@get_cov(df_assets)
@get_mvp_weights(df_assets)))) > .0005:
    prob.solve()
    #cvxpy problem defined outside
    w_opt = w.value
    gamma_front.append(round(gamma.value,1))
    #save current gamma value
    mu_front.append(w.value.T@get_mu(df_assets))
    #save mu for current gamma value
    sigma_front.append(np.sqrt(w.value.T@get_cov(df_assets)@w.value))
    #save sigma for current gamma value
    weight_front  = pd.concat([weight_front,
    pd.DataFrame(w_opt.reshape((len(w_opt), 1)),
    columns=[round(gamma.value,1)])], axis=1)
    #save weights for current gamma value
    gamma.value += .1
```

B.12 Rolling Single Period Optimization

```python
# Cascading SPO simulating MPO (Max. return for 0.01
# 1-day CVaR); Last years opt weights (in-sample) are used
# to calculate returns for the "current" year (out-of-sample)
w_optP = {}
start_dict = {}
end_dict = {}
end_value = {}
```

```python
#Setup dictionaries to store opt values
df_combined = pd.DataFrame()

d_start = df_assets.index[0]+pd.offsets.YearBegin(-1)
#Set first Date of dataframe as start of the opt period
start_dict[0] = d_start
d_end = d_start+pd.offsets.YearEnd(0)
#Set end of opt period 1 year from start
end_dict[0] = d_start
df_assetsP = df_assets.loc[d_start:d_end]
df_indexP = df_index.loc[d_start:d_end]
#Create dataframe for current period

w = cp.Variable(len(df_assetsP.columns),)
alpha = cp.Variable()
u = cp.Variable(len(df_assetsP)) #auxiliary var. from CVaR opt
beta = 0.95
cvar = alpha + 1.0 / (len(df_assetsP) * (1 - beta)) * cp.sum(u)
cvar_target = 0.01

objective = cp.Maximize(w.T@get_mu(df_assetsP))
constraints = [
                cp.sum(w) == 1,
                u >= 0.0,
                w >= 0.0,
                w <= 0.10,
                df_assetsP.values@w + alpha + u >= 0.0,
                cvar <= cvar_target
              ]

prob = cp.Problem(objective, constraints)
prob.solve()
#Standard CVaR problem; see max. return for CVaR
w_optP[0] = copy.copy(w.value)
i = 0

#Iterations over all 1-year periods of the given timeseries
while i < len(pd.date_range(df_assets.index[0],
df_assets.index[-1], freq='1Y')):
    d_start = df_assets.index[0]+pd.offsets.YearBegin(i+1)
    start_dict[i+1] = d_start
    d_end = d_start+pd.offsets.YearEnd(0)
    end_dict[i+1] = d_end
    df_assetsP = df_assets.loc[d_start:d_end]
    df_indexP = df_index.loc[d_start:d_end]

    w = cp.Variable(len(df_assetsP.columns),)
    alpha = cp.Variable()
    u = cp.Variable(len(df_assetsP))
    beta = 0.95 #confidence; if adjusted here, adjust below
    cvar = alpha + 1.0 / (len(df_assetsP) * (1 - beta)) * cp.sum(u)
    cvar_target = 0.01 #if adjusted here, adjust below

    objective = cp.Maximize(w.T@get_mu(df_assetsP))
    constraints = [
                cp.sum(w) == 1,
                u >= 0.0,
                w >= 0.0,
                w <= 0.10,
                df_assetsP.values@w + alpha + u >= 0.0,
                cvar <= cvar_target
              ]

    prob = cp.Problem(objective, constraints)
    prob.solve()
    w_optP[i+1] = copy.copy(w.value)
    _temp_rets = (df_assets.loc[start_dict[i+1]:end_dict[i+1]])@w_optP[i]
    #Calculate weighted returns for current period
    df_temp_rets = pd.DataFrame(_temp_rets)
    df_combined = df_combined.append(df_temp_rets)
    #Append current weighted returns to DataFrame with previous returns
    i += 1
```

B.13 Plotting Returns

```python
def plt_retuns(returns,weights,benchmark,save=False):
    """
    Plots time-series of returns in 2-dimensional
    plot with dates on the x-axis. Visualizing return variance.
    returns: pandas DataFrame containing daily returns
    weights: vector of weights; has to be same length as number of assets
    benchmark: pandas DataFrame containing daily returns of a benchmark
    """
    fig = plt.figure()
    ax = fig.add_subplot(111)
    plt.plot(get_weighted_returns(returns,weights),
    label='Optimized', alpha=0.5, color="steelblue")
    plt.plot(benchmark, label='Index',alpha=0.3, color="lightslategray")
    ax.legend()

    plt.ylabel('Return')
    ax.legend()
    plt.tight_layout() #more space for x and y labels
    plt.grid(alpha=0.3)

    if save:
        plt.savefig('/Users/../_vola.pdf', dpi=500, format='pdf')

    plt.show()
```

B.14 Plotting Cumulative Returns

```
def plt_cum_retuns(returns,weights,benchmark,save=False):
    """
    Plots time-series of cumulative returns in 2-dimensional
    plot with dates on the x-axis.
    returns: pandas DataFrame containing daily returns
    weights: vector of weights; has to be same length as number of assets
    benchmark: pandas DataFrame containing daily returns of a benchmark
    """
    fig = plt.figure()
    ax = fig.add_subplot(111)
    plt.plot((returns@weights).add(1).cumprod(),
    label='Optimized', alpha=0.5, color="steelblue")
    plt.plot(benchmark.add(1).cumprod(),
    label='Index',alpha=0.3, color="lightslategray")

    ax.legend()
    plt.ylabel('Cumulative Return')
    plt.tight_layout() #more space for x and y labels
    plt.grid(alpha=0.3)

    if save:
        plt.savefig('/Users/../_opt_return.pdf', dpi=500, format='pdf')

    plt.show()
```

B.15 Plotting Efficient Frontier

```python
def plt_frontier(mu_front, sigma_front, sharpe_front ,save=False):
        """
    Calculates and plots efficient frontier.
    mu_front: array of efficient means
    sigma_front: array of efficient standard deviations
    sharpe_front: array of Sharpe ratios
    """
    fig = plt.figure()
    ax = fig.add_subplot(111)

    plt.plot(sigma_front, mu_front, color='steelblue',
    linewidth=2,alpha=0.5, label='Efficient frontier') # Plot frontier
    plt.plot(sigma_front[np.argmax(sharpe_front)],
    mu_front[np.argmax(sharpe_front)], marker = '*',
    color='steelblue', label="Maximum Sharpe portfolio",
    markersize=12) #Highlight max Sharpe portfolio

    v = [.1,1,2,3,5,10,20,30,50]
    #levels of risk aversion to be highlighted on the frontier

    for _ in range(len(v)):
        plt.plot(sigma_front[gamma_front.index(v[_])],
        mu_front[gamma_front.index(v[_])], marker = 'o', color='steelblue')
        ax.annotate(r"$\gamma = \%.2f$" \% (v[_]),
        xy=(sigma_front[gamma_front.index(v[_])]+.002,
        mu_front[gamma_front.index(v[_])]-0.004))
        #Annotate gamma value


    ax.legend(loc = 4)
    plt.grid(alpha = 0.3)
    plt.xlabel('Standard Deviation (Volatility)')
    plt.ylabel('Expected Annual Return')

    plt.tight_layout()
    if save:
        plt.savefig('/Users/../_frontier.pdf', dpi=500, format='pdf')
    plt.show()
```

B.16 Plotting Efficient Frontier with Assets Scatter

```python
def plt_frontier_scatter(mu_front, sigma_front, sharpe_front,
                         assets ,save=False):
    """
    Calculates and plots efficient frontier relative
    to all assets of the portfolio.
    mu_front: array of efficient means
    sigma_front: array of efficient standard deviations
    sharpe_front: array of Sharpe ratios
    assets: pandas DataFrame containing daily returns
    """
    fig = plt.figure()
    ax = fig.add_subplot(111)

    plt.plot(sigma_front, mu_front, color='steelblue',
    linewidth=2,alpha=0.5, label='Efficient frontier')
    # Plot frontier
    plt.plot(sigma_front[np.argmax(sharpe_front)],
    mu_front[np.argmax(sharpe_front)], marker = '*',
    color='steelblue', label="Maximum Sharpe portfolio",
    markersize=10) #Higlight max sharpe portfolio
    plt.scatter(get_std(assets),get_mu(assets), s=8 , label='Assets')

    ax.legend(loc = 3)
    plt.grid(alpha = 0.3)
    plt.xlabel('Standard Deviation (Volatility)')
    plt.ylabel('Expected Annual Return')

    plt.tight_layout()

    if save:
        plt.savefig('/Users/../_scatter_front.pdf', dpi=500, format='pdf')
    plt.show()
```

B.17 Plotting Correlation Matrix

```python
def plot_corr(timeseries):
    """
    Plots the correlation matrix with a weighted color scheme.
    timeseries: pandas DataFrame containing daily returns
    """
    return timeseries.corr().style.background_gradient
    (cmap='coolwarm', axis=None).set_precision(4)
```

B.18 Plotting Maximum Drawdown

```python
def maximum_drawdown(timeseries, plot=False):
    """
    Calculates and plots the MDD over a given period.
    timeseries: pandas DataFrame containing daily returns
    """
    if plot:
        plt.plot(np.exp((w.value@np.log(timeseries+1).T).cumsum()))
    prev_peak = np.exp((w.value@np.log(timeseries+1).T).cumsum()).cummax()
    if plot:
        plt.plot(prev_peak)
    drawdown = (np.exp((w.value@np.log(timeseries+1).T)
    .cumsum())-prev_peak)/prev_peak
    if plot:
        plt.plot(drawdown)
    print("Maximum Drawdown: {:.3f}\%".format(drawdown.min()
    *100),"on",drawdown.idxmin().strftime('\%d/\%m/\%Y'))

    return None
```