Portfolio Construction and Risk Management

Anton Vorobets

Latest sneak peek available at: https://ssrn.com/abstract=4807200.

Crowdfunding page: https://igg.me/at/pcrm-book.

This book should currently only be accessible to crowdfunding campaign contributors at: https://igg.me/at/pcrm-book. If you have gained access to it in another way, you should know that your access is unlicensed and in violation of the authors All Rights Reserved copyright. In that case, you are encouraged to contribute to the crowdfunding campaign and get licensed access to the book and accompanying code.

Preface

This book is written based on the need to bridge quantitative analysis and investment management in practice, particularly focusing on portfolio construction and risk management. The book's approach can be characterized as being "quantamental", i.e., a mix of quantitative analysis and human and machine interaction. It is based on the philosophy that some things are really hard for machines and statistical models to do, because these usually require a significant amount of stationary data to work well, while humans are able to quickly develop an understanding based on little or imperfect information. Hence, most people likely have the highest probability of being successful using a quantamental approach to investment management, while all the methods can be used for fully systematic investing.

The book attempts to keep the mathematics practically rigorous while referring to proofs of very technical results instead of replicating them. It does not attempt to theorize or quantify human behavior like finance and economics academics do with utility theory. Conventional utility theory has poor empirical support and limits the analysis to methods that have very little to do with real-world market behavior. In relation to investor preferences, our hypothesis is that investors characterize investment risk as large losses, not squared deviations from the mean. It is very hard to reject this hypothesis in practice. We will in particular focus on the conditional Value-at-Risk (CVaR) investment risk measure, because it has many nice properties and is easy to interpret.

The goal is to keep the book concise and to the point. It does not spend time on introducing general mathematical concepts as these are readily available in other great books. Hence, it is a prerequisite that you understand linear algebra, calculus including convex constrained optimization with Lagrange multipliers, probability theory, multivariate time series, and machine learning / artificial intelligence / econometrics / statistics. Finally, we will use Python and in particular the fortitudo.tech[1] package for the code examples.

This book is cohesive in the sense that it gives you a complete investment risk and analysis framework. The approach is scientific as it tries to address many of the complex nuances of real-world investment markets. It is based on my experiences managing multi-asset portfolios for institutional clients and my regular dialogues with institutional asset managers about the problems they experience in practice. The book will naturally draw on my articles[2], while presenting the framework in a more coherent way to make it perfectly clear how it should be used from start to finish including its subtle nuances.

[1] Available at https://github.com/fortitudo-tech/fortitudo.tech
[2] Available at https://ssrn.com/author=2738420

I omit aspects related to signal extraction and alpha generation. You should rather see the framework from this book as the general principles for a good investment calculator that allows you to get the most out of your investment risk budget and, hence, generate "portfolio construction alpha". Alpha related to security selection and market timing is so elusive that I would trade it myself and not tell anyone how I did it.

This book is very different from mainstream academic finance and economics books that continue to spend time on theories and methods like utility theory, CAPM, mean-variance, and Black-Litterman. I think these methods do more harm than good in practice due to their highly unrealistic market assumptions. However, they are still being taught due to academic and commercial vested interests. We will occasionally use mean-variance to build intuition in the idealized case, but it is never recommended to actually use mean-variance to manage portfolios in practice. The elliptical distribution assumption is too oversimplifying to represent the behavior of real-world markets well. All methods presented in this book operate directly on fully general Monte Carlo distributions with associated joint probability vectors.

The book is written for investment practitioners who employ a scientific approach to investment management that potentially combines data with human discretion. By scientific, I mean constantly testing the theories and methods against real-world market behavior. Readers are generally encouraged to critically examine the content of this book and suggest improvements that will make the framework even better for navigating investment markets in practice. Hypothetical issues or introduction of constraints due to theories with poor empirical support will not be considered.

The book and supporting material may eventually be provided free of charge online, while crowdfunding makes it possible to write it. During writing, all campaign contributors beyond a threshold amount will have access to a private GitHub repository where the latest version of this book will be available in addition to the accompanying Python code. I will also answer questions in the forum of this repository. Top 10 contributors by the time this book is made publicly available will have their name written in this preface (if they wish) in addition to getting three months access to an institutional-grade implementation of the framework described in this book. Hence, backing this project gives you access to a community that is actively learning the content simultaneously with you. It is a great opportunity to master the theory and methods before most others.

The book will occasionally use proprietary software for some of the case studies and examples. I decided to do this in order to show readers some of the more advanced analysis that they can perform using the framework, although the accompanied code cannot be provided. In most situations, the code will be available, and it is an integral part of studying the content of this book. There is always full transparency in relation to the problem being solved, while readers should not expect the code to of production quality. The accompanying code is given without any warranty. The author cannot be made liable for any potential damages stemming from using the code.

To get early access to the book and the accompanying Python code, you can support the crowdfunding campaign here: https://igg.me/at/pcrm-book.

It doesn't matter how beautiful your theory is. If it disagrees with experiment, it's wrong.

- Richard Feynman

Acknowledgments

The creation of this book is made possible through crowdfunding contributions at:

https://igg.me/at/pcrm-book

Contributors have often also shared their feedback on the content, making it better and easier to understand for all readers. All contributions are highly appreciated, while some people have contributed significant amounts and allowed their names to be shared in this section. While the book is being written, the below lists will be updated and expand. When the book is finished around the new year 2025, the final top 10 list will be revealed. Until then, the top 10 contributors section will include the necessary contribution amount to enter the top 10 as of the date listed in the section. Top 10 contributors will get access to exploring an institutional-grade implementation of the framework and really test out their newly acquired knowledge in practice. Potential ties among top 10 contributors are settled on a first come first served basis, giving early supporters an advantage.

Top 10 contributors

As of June 29, 2024, your contribution must be above 20 euro to enter the current top 10.

Platinum contributors

Gold contributors

Silver contributors

Bronze contributors

Matteo Nobile, Roger McIntosh, Mark Brezina.

Contents

Chapter 1

Introduction and overview

1.1 Book and framework overview

The framework in this book is centered around a Monte Carlo simulation given by the matrix $R \in \mathbb{R}^{S \times I}$ and associated joint scenario probability vector $p \in \mathbb{R}^S$ with $\sum_{s=1}^{S} p_s = 1$ and $p_s \in (0, 1]$. This market representation allows us to work with fully general distributions that we can sample from. If we want to introduce some (subjective) market views or perform stress-testing, we generally do it by adjusting the prior probability vector p into the posterior probability vector $q \in \mathbb{R}^S$.

The columns of R represent samples from the marginal distribution of price, return, or factor i, $i = 1, 2, \ldots, I$. The rows of R represent samples from the joint distribution of the prices, returns, and factors. Hence, the core elements of the framework are

$$R = \begin{pmatrix} R_{1,1} & R_{1,2} & \cdots & R_{1,I} \\ R_{2,1} & R_{2,2} & \cdots & R_{2,I} \\ \vdots & \vdots & \ddots & \vdots \\ R_{S,1} & R_{S,2} & \cdots & R_{S,I} \end{pmatrix}, \quad p = \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_S \end{pmatrix}, \quad \text{and} \quad q = \begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_S \end{pmatrix}. \tag{1.1.1}$$

Our first task is to generate realistic prior simulations of the market R based observed historical market data $D \in \mathbb{R}^{T \times N}$. Realistic simulations are characterized by their ability to capture the stylized market facts presented in Chapter 2 and ideally additional subtle nuances of real-world investment markets. Another important point is that analysis and optimization methods must be able to handle the fully general simulations R and associated probability vectors p and q. Mean-variance analysis clearly fails in this regard, because it reduces the market to an elliptical distribution and only allows for linear and constant dependencies through the covariance matrix.

Note that the market representation in (1.1.1) and most of the analysis in this book are one-period in nature, while we do not make any assumptions about living in a one-period world. Hence, there is an implicit horizon in the matrix R. Sometimes we will be explicit and write R_h, $h = 1, 2, \ldots, H$, as in Chapter 3 about market simulation, while maintaining the more concise notation throughout most of the book. For dynamic investment strategies, it is important to simulate entire paths, while their cumulative returns can still be analyzed in a meaningful way for some horizon $h \in \{1, 2, \ldots, H\}$.

When generating Monte Carlo simulations of the market, the focus will be on generating simulations of factors and then translating their simulations into instrument price and return simulations. We will distinguish between two kind of factors: risk factors and market factors. A risk factor is defined as something that goes directly into the pricing function of an instrument, for example, US government bond zero-coupon interest rates are risk factors for US government bonds. A market factor is something that might have a statistical effect on the price or return of an instrument, for example, Purchasing Managers' Index (PMI) numbers, but does not directly enter in the pricing function of an instrument. We will explore risk factors much more carefully in Chapter 4 about instrument pricing.

Once we have good simulations of prices, returns, and factors, we probably want to incorporate adjustments based on information that might not be available in the historical data. We call this (subjective) market views. We might also want to examine how our simulations behave in adverse market scenarios, which we call stress-testing. No matter which of the two cases it is, the fundamental method we use is called Entropy Pooling (EP), introduced by Meucci (2008a). EP minimizes the relative entropy (Kullback–Leibler divergence) between the prior probability vector p and the posterior probability vector q subject to linear constraints on the scenario probabilities, i.e.,

$$q = \underset{x}{\operatorname{argmin}} \left\{ x^T (\ln x - \ln p) \right\}$$

subject to

$$Gx \leq h \quad \text{and} \quad Ax = b.$$

Note that we use $\ln x$ to mean the logarithm of each element of x, and that $\leq$ and $=$ are used to denote element-wise (in)equalities. Entropy Pooling can be used in many sophisticated ways, see Vorobets (2021) and Vorobets (2023), that we will carefully explore in Chapter 5.

After implementing the market views and stress-testing, we likely want to optimize our portfolios subject to various investment constraints and include portfolio optimization parameter uncertainty, see Kristensen and Vorobets (2024) and Vorobets (2024). Portfolio optimization is the topic of Chapter 6, where we will focus on CVaR optimization with risk targets and risk budgets in addition to the ability to seamlessly handle derivatives instruments by separating relative market values $v \in \mathbb{R}^I$ from relative exposures $e \in \mathbb{R}^I$, see Vorobets (2022a) and Vorobets (2022b).

Chapter 7 is about general risk and return analysis, with a particular focus on the nuances of tail risk hedging and management. The general idea is that the methods in the first six chapters will allow us to build well-diversified portfolios, which we can additionally equip with tail risk hedges if we wish. Outright hedging often comes at a significant negative (volatility) risk premium. Hence, tail risk hedges should rather be designed with the objective of giving a positive return when key diversification assumptions fail, while outright tail risk hedging can be considered on a tactical basis.

Chapter 8 contains a summary of the investment risk and analysis framework introduced in this book in addition to a comparison with old methods such as mean-variance and Black-Litterman (BL). In Chapter 8, we will see that in the best case the old framework is a simple subset of the new, while BL contains so many questionable aspects that it does not even produce a correct updating of the CAPM prior to a posterior distribution when returns are normally distributed. For this reason, it is never recommended to use the old framework in practice, and especially not the BL model.

The rest of this chapter presents core principles for reasoning about market states, structural breaks, and time-conditioning in addition to the essence of investment and risk management. These principles will be important to keep in mind throughout the book and should apply to the majority of investment markets and strategies. Very niche investment markets or strategies might have characteristics that are so unique and different from the majority that they fall outside the scope of a general portfolio construction and risk management book like this one.

1.2 Market states, structural breaks, and time-conditioning

We start this section with the dreaded coin flipping analysis. Not because we want to study the probability theory behind coin flips and the associated Bernoulli and Binomial distributions. Also not because coin flipping is a good representation of investment markets, but because it allows us to introduce the important concepts of market states and structural breaks in a way that is easy to understand. These concepts will be important for thinking about real-world investment and risk management, where the sheer complexity might overshadow the essence of the points made in this section.

As an investment professional, you might have experienced a situation where someone you know asked you for investment advice along the lines of "what about company X? Should I buy it now, will it go up?". Many people seem to think that what investment managers do is to predict and time the realizations of investment markets. Some investment managers trading in very special markets or on very special information might, but the vast majority do not, simply because it is impossible.

Many people find the answer about market realizations not being predictable unsatisfactory, while they fully understand that no one can predict the realization of the next coin flip or draw of lottery numbers. In fact, if someone claimed to be able to do that, most people will immediately distrust that claim. It is clear to most that the best we can do when it comes to coin flips and lotteries is to make probabilistic predictions. Yet the additional complexity of investment markets fools many people into believing that they have discovered a pattern for the next period's realization. As an inexperienced investor, you might even think that you have this ability yourself, but with time you will realize that consistently calling the realization of investment markets is not possible.

So, is careful investment analysis and risk management a complete waste of time? Not quite, it is just important to understand that we can only add value in a probabilistic sense, which we will carefully explore in the next section. In this section, we will continue to focus on coin flips to introduce the important concepts of market states, structural breaks, and time-conditioning.

Without going into too much mathematical formality, let us consider a coin C that can come out heads with probability $p \in [0, 1]$ and tails with probability $1 - p$. We can flip this coin over a discrete time-period $t = 1, 2, \ldots, T$ and denote its realization at each step $c_t \in \{\text{heads}, \text{tails}\}$. Let us assume that you have \$100 that you can invest in various strategies involving the coin, giving you the opportunity of increasing the value of your initial \$100. If it is a "fair" coin with $p = 0.5$, the expected value of engaging in this activity is \$0. You might still want to participate in the coin flipping just for the thrill of it and design strategies that fix the probability of losing all your money during the time-period to some sufficiently low number.

Let us now imagine that the coin is biased such that $p = 0.9$, making it obvious that you can expect to earn money by systematically betting on heads. If you are not careful, you can still lose all your money as there is no guarantee that the coin will come out heads. Hence, it is still impossible for you to call the realization of the coin c_t at any time t. How you decide to invest in this coin will to a large extent be a function of your risk willingness. The highest expected return will be achieved by betting all your money on heads at each point in time $t = 1, 2, \ldots, T$, while it also gives you the highest probability of losing all your money. A reasonable suggestion would be to find a strategy where the trade-off between the expected return and the probability of losing all your money is the best. While we are not going to spend time on that in this book, the interested reader can study this further.

Continuing with the coin flipping, let us imagine a situation where $p = 0.1$. In this case, betting on tails will obviously be associated with a positive expected return. However, let us say that you are constrained to only betting on heads, i.e., you are "long-only" heads, while you can adjust the size of your bets before each coin flip at time t. If the coin is in a constant state with $p = 0.1$, while you are only allowed to bet on heads, it only make sense for you to participate if you are a real thrill seeker. However, if the coin switches between $p \in \{0.1, 0.5, 0.9\}$, you still have the opportunity of generating a positive expected return if you are able to infer the value of p and size your investments accordingly.

To formalize the above a bit more, let us introduce a state variable $S_t \in \{s_{t,1}, s_{t,2}, s_{t,3}\}$, $t = 0, 1, \ldots, T$, and imagine that states changes according to the following transition probability matrix

$$\mathcal{T} = \begin{pmatrix} 0.9 & 0.1 & 0.0 \\ 0.1 & 0.8 & 0.1 \\ 0.0 & 0.1 & 0.9 \end{pmatrix}.$$

Each row in the transition matrix represents the probability $\mathcal{T}_{ij}$ for transitioning from a state i to a state j. This evolution of states is known as a Markov chain, because the next state S_{t+1} only depends on the current state S_t.

A stationary probability vector π can be computed for the transition matrix $\mathcal{T}$, i.e., a row vector τ so that $\tau\mathcal{T} = \tau$. In our particular example, the transition matrix is called doubly stochastic, because both its rows and columns sum to 1. For doubly stochastic matrices, the stationary distribution will be uniform, i.e., $\pi = (1/3, 1/3, 1/3)$ in our case. Readers can verify that this is indeed the case by performing the matrix multiplication $\pi\mathcal{T}$. Here, by the stationary distribution we mean what the Markov chain will produce over the long run. Hence, if the time period T is sufficiently long, we expected the three different states to occur equally frequently regardless of the initial state.

Readers who are interested in exploring the characteristics of the above Markov chain further are encouraged to examine the accompanying code for this section. The code contains a function for simulating this Markov chain with some elementary analysis illustrating that it behaves as expected. The important realization one has to make is that there is stochasticity in the state, which determines the probability of the coin coming up heads p, and finally there is stochasticity in the outcome of the coin. This is an important conceptual separation. In practice, we would only observe outcomes of the coin and have to estimate both the transition matrix $\mathcal{T}$ and the associated heads probabilities p for each state. If we are only good at estimating one of them and very bad at estimating the other, our strategy is unlikely to be successful.

In the previous paragraphs, a lot of new terminology has been vaguely introduced. To maintain our focus, we will not delve deeper into Markov chains and simply refer the interest reader to Norris (1998). This might be unsatisfactory for some readers, but we do not want the mathematical formality to overshadow the main points. The objective of the above presentation was to introduce the concept of market states, if we for a second imagine that the coin flipping is a market that someone would allow us to invest in. It was also to introduce the concept of short-term and long-term behavior, by showing that the current state helps us to infer the likely next state, while it does very little to help us infer the state many time-steps into the future. You can see this in practice in the accompanying code to this section.

The analysis so far has been very nice and simple, with a very well-behaved coin having constant states and state transition probabilities $\mathcal{T}$. Now let us imagine that a sudden appearance of new state $p \in [0, 1] \notin \{0.1, 0.5, 0.9\}$ can happen at a random point in time, causing a fundamental change in the transition matrix $\mathcal{T}$. Or maybe one of the existing states stops appearing all together. No matter which case it is, we will refer to any changes to the transition matrix $\mathcal{T}$ as a structural break.

Structural breaks pose additional challenges because we cannot simply lean back and enjoy the profits of our strategy that might have been very successful historically at inferring the state S_t and sizing investment into the heads outcome accordingly. We must constantly monitor the performance of our investment strategy and adapt it to the new reality. If we insist on using fully systematic strategies, we have to rely on having sufficient data showing us that a structural break has occurred. This will probably lead to a period of lower expected returns and a higher probability of losing money, with the investment strategy being suboptimal.

If we are allowed to adjust our strategy based on other information, for example, someone reliable communicating to us that a structural break has occurred, we have the opportunity to adapt before we see realizations of the structural break in the data. In real-world markets, such announcements could, for example, be a central bank communicating a new forward guidance on their monetary policy. The ability to mix historical data with forward-looking adjustments based on qualitative inputs is what makes the quantamental investing approach appealing.

For real-world investment markets, prices, returns, and factors have much more complex distributions than the Bernoulli distribution of the coin, and the states are much more abstract than the probability of heads. It is probably also very hard to find true stationarity in the data, because markets are constantly evolving. But the concepts of market states and structural breaks will still be valuable for us when thinking about investment markets, and you are surely going to hear investment practitioners talk about these things. At the very least, we are going to use these concepts in Section 3.2.1 about resampling methods for market simulation. Note that practitioners sometimes refer to states as regimes.

Since real-world investment markets are governed by much more complex states, we are unlikely to be able to capture all necessary state-dependent information at any given point in time by relying solely on interpretable state variables. For example, we might use the VIX index and the slope of the interest rate curve as state variables for real-world markets, but there is likely still much more that we need to condition on to fully capture the market state. Hence, we probably need a residual state variable that we can condition on.

Time-conditioning has historically been a frequently used way to capture market state. Popular examples are the exponentially weighted moving average (EWMA) and generalized autoregressive conditional heteroskedasticity (GARCH) models. The hypothesis is that recent data tells us more about the immediate future than old data. While it is probably true that markets tomorrow are more similar to markets today than 30 years ago, it is a strong assumption that time-conditioning alone will capture all the necessary information. In this book, we will view time-conditioning as a way to capture residual information that our other state variables are potentially not able to capture.

Here comes another real-world blow. All the perspectives related to market state, structural breaks, and time-conditioning might be completely wrong for how investment markets actually behave. While we can have full control over the coin flipping experiment, we have no control over the distributions of investment markets. These perspectives are simply the ones that we currently have the mathematical machinery to handle, so our analysis will naturally be biased by them. Hence, when we analyze the stylized market facts in Chapter 2, it will be through this lens. The same goes for our simulation approaches in Chapter 3. We are, however, not stipulating that market distributions can be fully characterized by their state, or that market states behave according to a Markov chain. The hypothesis is rather that something along these lines is approximately correct, where we are at least not introducing constraints that obviously violate the characteristics of real-world markets.

1.3 The essence of investment and risk management

In this section, we move a bit closer to investment markets by starting to talk about equities and bonds instead of a coin. We will still remain in a highly hypothetical world by assuming that the return distributions of equities and bonds are jointly normal and therefore fully characterized by the first two moments, allowing us to focus on just the mean vector μ and the covariance matrix Σ. As we will see in Chapter 2, this is quite far from reality, while it is the setting for the original analysis of investment risk and return introduced by Markowitz (1952).

Similarly to the previous section, let us assume that the return distributions for the next period have various states, which we call risk off, base case, and risk on. Our first goal is to infer which state we believe is most likely and decide on a portfolio for the next period. As with the coin's outcomes, investment outcomes are only predictable in a probabilistic sense, and the states are stochastic. We will not write out a transition matrix for the states in this section, but you can imagine that something along those lines determines the market state.

We will examine how the optimal portfolio is in all of the states while noting that we must eventually select just one portfolio. Using the Entropy Pooling method presented in Chapter 5 and Section 5.3 will allow us to weight the different cases according to their probability of occurrence to arrive at a single distribution that we can optimize over. We note that the naive approach of simply weighting the individual portfolios according to the probability of state occurrence does not guarantee mean-risk optimality over the final weighted distribution. From a portfolio optimization with parameter uncertainty perspective, this weighting can still make sense to reduce the variance of the optimal exposure estimates, see Kristensen and Vorobets (2024), Vorobets (2024), and Chapter 6 on portfolio optimization.

Let us assume that we characterize the states by the following assumptions on the mean vector and covariance matrix:

$$\mu_{off} = \begin{pmatrix} 0.05 \\ 0.06 \end{pmatrix} \quad \text{and} \quad \Sigma_{off} = \begin{pmatrix} 0.08^2 & -0.2 \cdot 0.08 \cdot 0.3 \\ \bullet & 0.3^2 \end{pmatrix},$$
$$\mu = \begin{pmatrix} 0.03 \\ 0.1 \end{pmatrix} \quad \text{and} \quad \Sigma = \begin{pmatrix} 0.1^2 & 0 \\ \bullet & 0.25^2 \end{pmatrix},$$
$$\mu_{on} = \begin{pmatrix} 0.02 \\ 0.125 \end{pmatrix} \quad \text{and} \quad \Sigma_{on} = \begin{pmatrix} 0.1^2 & 0.2 \cdot 0.1 \cdot 0.2 \\ \bullet & 0.2^2 \end{pmatrix},$$

where we use $\bullet$ to indicate that the covariance matrix is symmetric.

Let us say that our investment mandate requires us to maintain a volatility level of 12.5%, while our portfolio must be long-only and allow us to invest only in cash instruments, i.e., $e \geq 0$ and $\sum_{i=1}^{2} e_i = 1$. Note that we use the derivatives framework notation from Vorobets (2022a) and Section 6.1 throughout this book. In this case, the relative market values v are equal to a vector of ones, so exposures will correspond to portfolio weights, while this will not always be the case.

The portfolio optimization problem can be solved easily in this case using second-order cone programming, see Boyd and Vandenberghe (2004) and the accompanying Python code to this section. The optimal portfolios for the three different states are given in Table 1.1.

	Risk off	Base	Risk on
Bonds	58.10%	55.12%	46.43%
Equities	41.90%	44.88%	53.57%

Table 1.1: Optimal bond / equity portfolios with a 12.5% volatility target.

While we can optimize portfolios for each state, we must decide on one portfolio to invest in. It is, however, still interesting to examine what happens to the portfolio in each of the different cases that we have imagined. For example, what happens to the base case portfolio in the risk off case. For risk management purposes, we would then ask if it is a risk that we are comfortable with, or if we should attempt to reduce or eliminate it through an adjustment of the portfolio or with tail risk hedging, which will be presented more carefully in Section 7.2.

Figure 1.3.1 shows the return distributions of the optimal portfolios from Table 1.1 in the risk off state. It is clear that if we decide to invest in the optimal portfolio for the base case, it will be suboptimal in the risk off case. Even worse is the optimal portfolio for the risk on state, which performs a lot worse than the risk off optimal portfolio. Considerations like these are what we will define as risk management as opposed to general diversification.

One could argue that the risk-adjusted performance loss in the risk off case is not substantial for the base case optimal portfolio, while the risk on portfolio exposes us to significant losses if end up in the risk off case. Hence, unless we strongly believe that the risk on or risk off case is very likely, we should probably choose the base case portfolio. An important consideration is also how large the risk overshoot will be for the risk on portfolio if we are in a risk off state, something that we will examine much more carefully in Section 6.3.

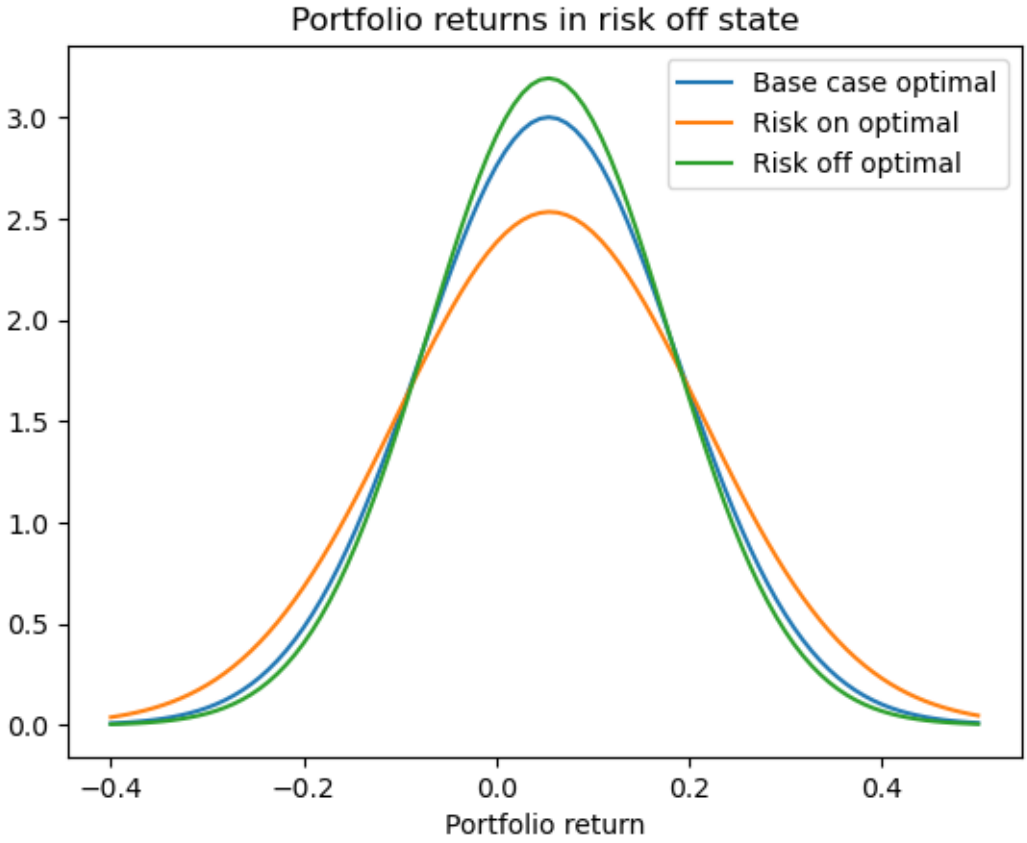

Figure 1.3.1: Portfolio return distributions in the risk off state.

Diversification, e.g., building a portfolio of both bonds and equities can of course also be seen as a form of risk management, but we will define risk management more specifically as a careful analysis of adverse scenarios and assessment of how to deal with them in this book. Diversification can be defined in many different ways, while we will loosely think about it as the ratio between the sum of the standalone risks of the individual exposures and actual portfolio risk, i.e.,

$$d = \frac{\sum_{i=1}^{I} \mathcal{R}(e_i)}{\mathcal{R}(e)},$$

where $\mathcal{R}(e_i)$ is the risk of the individual exposures, while $\mathcal{R}(e)$ is the risk of the portfolio.

The risk measure $\mathcal{R}$ can for example be CVaR, variance, or VaR. For coherent risk measures, see Artzner et al. (1999),

$$\mathcal{R}(e) \leq \sum_{i=1}^{I} \mathcal{R}(e_i), \tag{1.3.1}$$

which implies that $d \geq 1$ when the portfolio risk $\mathcal{R}(e)$ is positive, which will usually be the case. This property is formally referred to subadditivity and sometimes called the diversification principle. VaR is not a coherent risk measure, because it does not respect this diversification principle. VaR also has

several other issues, which is why CVaR is becoming the preferred investment tail risk measure for both market makers and investment managers.

Readers are encouraged to examine the accompanying Python code to this section to see how the computations have been performed for the optimal portfolios from Table 1.1 and the portfolio return plot from Figure 1.3.1. In addition, there are joint return plots for the three different states, while Figure 1.3.2 shows the joint return plots for the risk on and risk off states.

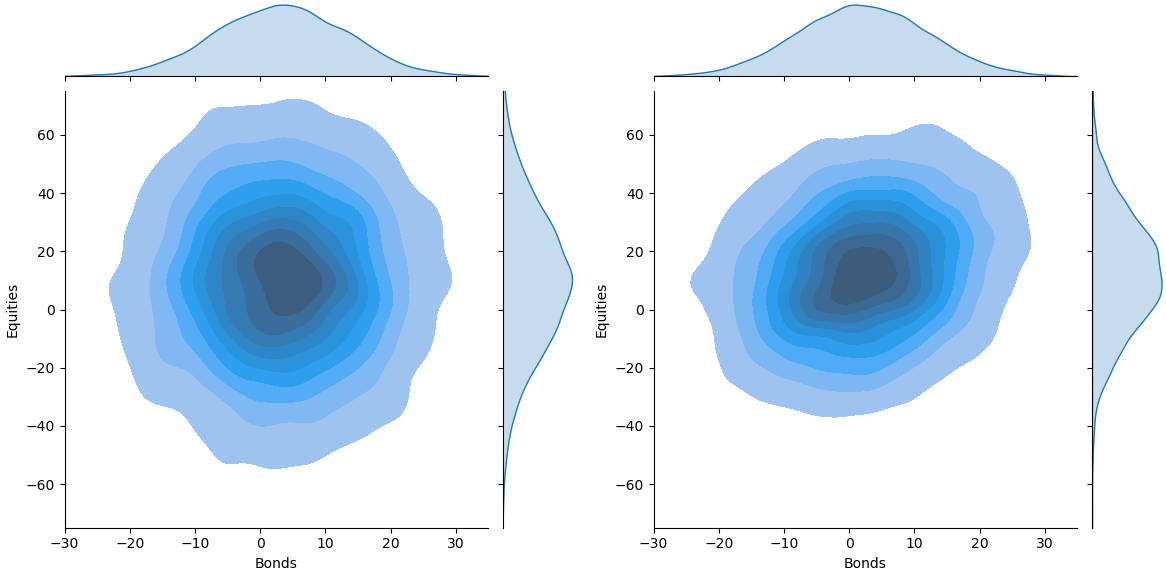

Figure 1.3.2: Joint return plots for bonds and equities in the base and risk on state.

Readers are also encouraged to spend some time thinking about what the graphs in Figure 1.3.2 show as we will be looking at graphs similar to these throughout the book. Think, for example, about which of the two graphs show a positive correlation between bonds and equities, in which graph equities have the highest variance, and what it means that the graph is darker and lighter in different areas. In Figure 1.3.2, we have purposefully simulated the distributions to generate the Monte Carlo market simulation matrix R from (1.1.1). The central graph is a contour plot of the joint distribution, while the two axes show the marginal distributions.

Finally, take a moment to reflect on the content in this chapter. Although the perspectives have been introduced in highly oversimplified cases, the concepts of market state, structural breaks, time-conditioning, and the essence of investment and risk management will carry over to more complex cases and analysis. These concepts have simply been presented in the most basic cases to make them perfectly clear in this chapter. As we increase the complexity of the market simulation and analysis, they might become less apparent, but they will be fundamentally the same.

In the rest of this book, we will start focusing more on tail risk and risk budgeting in addition to risk contribution analysis. In some instances, we will also analyze the market simulations directly either through joint plots similar to 1.3.2 or other conditional perspectives. All this will show that once we abandon the focus on the covariance matrix, we can perform much deeper and more meaningful analysis of investment markets.

Chapter 2

Stylized market facts

2.1 Risk clustering and the volatility risk premium

2.2 Risk return trade-off

2.3 Skewness and kurtosis

Chapter 3

Market simulation

3.1 From market factors to stationary transformations

3.2 Projection of stationary transformations

3.2.1 Time- and state-dependent resampling

3.2.2 Generative machine learning

3.3 From projected stationary transformations to simulated factors

Chapter 4

Instrument pricing

4.1 Demystifying derivatives

Chapter 5

Market views and stress-testing

This chapter gives a comprehensive presentation of the Entropy Pooling (EP) method, introduced by Meucci (2008a) and refined by Vorobets (2021), as well as its applications in causal and predictive market views and stress-testing as introduced by Vorobets (2023). While all this content is in principle publicly available already, it seems to be challenging for most people to understand. Hence, this chapter fills in the gaps in one place with a careful treatment and unified notation.

Entropy Pooling is a way to update the prior probability vector p, associated with the market Monte Carlo distribution R, by inputting information about the desired update. Some people like to think about Entropy Pooling as a generalization of the Black-Litterman (BL) model without all the oversimplifying normal distribution and CAPM assumptions, additionally avoiding the questionable engineering with the τ parameter. I think EP is so much more than that, so it is not doing it justice to make this comparison, which you will hopefully also discover after reading this chapter.

Section 5.1 presents the basic version of Entropy Pooling, which at its core is a relative entropy (Kullback–Leibler divergence) minimization between the prior probability vector p and the posterior probability vector q subject to linear constraints on the posterior probabilities. The section also presents some of the common views specifications, ranking views, and the nuances of VaR and CVaR views.

Section 5.2 presents the sequential Entropy Pooling refinement introduced by Vorobets (2021), which improves on many of the limitations imposed by the requirement that views are specified as linear constraints on the posterior probabilities and usually gives much better results. The sequential approach can also solve practically interesting problems that the original approach simply cannot.

Section 5.3 presents aspects related to view confidences and multiple users or states, which is presented in the appendix of Meucci (2008a) but is still a largely unexplored area of the EP method. View confidences give additional nuances to the method, and user weights or state probabilities allow us to input potentially conflicting views that are aggregated into one posterior distribution.

Section 5.4 uses the multiple states results and presents the Causal and Predictive Market Views and Stress-Testing framework from Vorobets (2023). This framework combines Entropy Pooling with a causal Bayesian network layer on top to generate joint causal views and their associated joint probabilities, which allow us to compute a single posterior distribution that incorporates the causality assumptions from the Bayesian network.

5.1 Entropy Pooling

As mentioned in the introductory section 1.1, Entropy Pooling (EP) solves the relative entropy (Kullback–Leibler divergence) minimization problem subject to linear constraints on the posterior probabilities

$$q = \underset{x}{\operatorname{argmin}} \left\{ x^T \left(\ln x - \ln p \right) \right\} \tag{5.1.1}$$

subject to

$$Gx \leq h \quad \text{and} \quad Ax = b.$$

The first natural question that arises is: why minimize the relative entropy? The short answer is because it has good properties for our updating problem, similar to the mean squared error having good properties for linear regression. However, this explanation is probably insufficient for most people from an intuitive perspective, which is arguably important for the adoption of a new method when nontechnical people ask them to explain it.

To start building Entropy Pooling intuition, we must first be clear about what we intuitively will not be able to explain. We will not be able to intuitively explain the actual value of the relative entropy, but we can transform it to the effective number of scenarios given in (5.1.2) below, which has a nice intuitive interpretation as the probability mass concentration over the S joint Monte Carlo samples in R.

The relative entropy represents a statistical distance between two distributions p and q, while it is not a mathematical metric because it is asymmetric and does not satisfy the triangle inequality. It can be interpreted as the expected excess surprise from using the distribution q instead of p. We are now approaching the essence of what we are doing when we are minimizing the relative entropy. We are minimizing the spuriosity while updating our prior distribution p to the posterior distribution q.

In relation to the spuriosity, we probably want to avoid assigning all probability to one scenario s and get a degenerate posterior market distribution q. Unless we have it as an actual view or stress-test, we probably also want to avoid introducing dependencies that are completely opposite of what we have in our prior simulation. For example, if we have two equity indices that are highly dependent, we usually want a stress-test on one of them to affect the other, see the example with S&P 500 and STOXX 50 in Section 5.2 below.

Besides operating on fully general Monte Carlo distributions R, taking the potentially very complex dependencies into account is where the true power of Entropy Pooling comes from. For example, it is just as easy to stress-test a portfolio containing only S&P 500 as a portfolio containing S&P 500 and thousands of derivatives on S&P 500. Entropy Pooling essentially makes a prediction on how other instruments and factors are expected to behave under the posterior distribution q. As we will see in Section 6.2.1 below, there is no need to reprice European put and call options when we perform Entropy Pooling stress-testing. Their posterior P&L distribution is automatically given to us.

With all of the above in mind, it hopefully becomes clear that Entropy Pooling is so much more than the BL model, which relies on the oversimplifying and empirically rejected normal distribution and CAPM assumptions. It sounds nice that "equilibrium expected returns" can be extracted using the BL model, but since these have very little to do with reality, they probably do not add any actual investment value and might even be detrimental.

In summary, Entropy Pooling is a theoretically sound method for implementing market views and stress-testing fully general Monte Carlo distributions R. It helps us predict what will happen to all instruments that we invest in and all factors that our portfolios are exposed to. It will ensure logical consistency in our derivatives P&L. As we will see in Section 5.3 below, Entropy Pooling also handles view confidences and state probabilities in a much more natural, probabilistic way than the BL model with its τ parameter, see Meucci (2008b) for an explanation of the issues and paradoxes. Hence, there is no reason to continue using BL when fast and stable Entropy Pooling implementations are freely available. If you for some reason want to use the CAPM prior, you can still do that to simulate R while getting all the Entropy Pooling benefits when implementing views and stress-tests.

To explore Entropy Pooling further from a mathematical perspective, we start by noticing that if all elements of the prior probability vector p are equal to $\frac{1}{S}$, the second term of (5.1.1) reduces to $\ln S$, which does not affect the solution, i.e., in the uniform prior probability case we can reduce the problem to

$$q = \underset{x}{\operatorname{argmin}} \left\{ x^T \ln x \right\} = \underset{x}{\operatorname{argmax}} \left\{ -x^T \ln x \right\} = \underset{x_1, x_2, \ldots, x_S}{\operatorname{argmax}} - \sum_{s=1}^{S} x_s \ln x_s.$$

The expression $-\sum_{s=1}^{S} x_s \ln x_s$ is know as the entropy, which we are trying to maximize when the prior probability vector p is uniform.

In most practical cases, the prior probability vector p will be uniform, while we have the opportunity to specify any valid prior probability vector with strictly positive elements that sum to one. Hence, Entropy Pooling will in most practical cases correspond to entropy maximization subject to linear constraints on the posterior distribution. Many more technical details are given about the properties of relative entropy minimization by Caticha and Giffin (2006), who also show that it corresponds to a generalization of Bayesian updating in the sense that information about the posterior distribution is given by constraints instead of the usual data. Caticha and Giffin (2006) also explain that entropy does not require any interpretation in this situation. It just has good properties for updating the distribution when additional information is given about the moments of the posterior distribution.

There is generally a lot of information about the relative entropy (Kullback–Leibler divergence), which is a quantity used in many different fields of mathematics and statistics. Readers who are interested in exploring this further must be warned that the content can be confusing due to the order of the distributions in (5.1.1). Sometimes, the objective is formulated equivalently as $-x^T (\ln p - \ln x)$, while in other cases the formulation remains the same but the prior and posterior order is switched for computational convenience. Finally, there are situations where the prior is implicitly assumed to be uniform, in which case we have seen that relative entropy minimization corresponds to entropy maximization.

A useful way of assessing how much the views and stress-tests deviate from the prior is called the effective number of scenarios, introduced by Meucci (2012a) and given by the exponential of the posterior probability entropy

$$\hat{S} = \exp\left\{ -\sum_{s=1}^{S} q_s \ln q_s \right\}. \tag{5.1.2}$$

The idea is that the effective number of scenarios is $\hat{S} = 1$ if all probability mass is given to one scenario, while it is $\hat{S} = S$ when scenario probabilities are uniform and equal to $\frac{1}{S}$. Note that we use

the convention that $q_s \ln q_s = 0$ for $q_s = 0$. It is often convenient to compute the relative effective number of scenarios $\hat{s} = \frac{\hat{S}}{S} \in (0, 1]$. We note that the relative effective number of scenarios is just the exponential of the negative relative entropy when prior probabilities are uniform. Hence, in a uniform prior case, a lower relative entropy gives a higher effective number of scenarios, which makes intuitive sense as this indicates that the posterior distribution is close to the uniform prior distribution.

The second natural Entropy Pooling questions is: what are the matrices and vectors G, h, A, and b? The short answer is that G and A contain functions of the Monte Carlo simulation R from (1.1.1), while h and b contain constraint values for these functions. From an intuitive investment perspective, this might not help you much, so we improve on that with examples below.

Consider for a moment how you would implement a constraint on the posterior expected value of some price, return, or factor $i = 1, 2, \ldots, I$. If we let R_i denote column i from the matrix R containing the market Monte Carlo simulation, it should be easy to convince yourself that the prior expected value is given by $R_i^T p = \mu_i$. If we want this value to change for the posterior distribution to $\tilde{\mu}_i$, we must implement the constraint that $R_i^T x = \tilde{\mu}_i$, which can be done through the matrix A and vector b. More generally, we can write

$$R_i^T x \gtreqless \tilde{\mu}_i,$$

using $\gtreqless$ to indicate that the view can be an equality or inequality in one of the two directions, with inequality views being implemented through G and h.

Questions related to Entropy Pooling's limitations are more subtle. For example, we cannot implement views that are not feasible for the Monte Carlo market simulation in R, i.e., if we want to implement a view that the expected value of some return should be 10% while all our simulations are below 10%, we cannot do that. It is hard to definitely say if this is a bug or a feature, because if R does not contain all the possible scenarios that we believe can occur in reality, it is a prior problem rather than a posterior problem. The prior problem should be fixed by better quality simulations using, for example, the approaches from Chapter 3. Limiting ourselves to the scenarios in R allows us to avoid the potentially computationally expensive repricing of derivatives and other instruments after implementing risk factor views, so in that sense it is a feature.

A significant limitation of the EP method is, however, that views must be specified as linear constraints on the posterior probabilities. Ideally, we would want to be able to solve the problem with fully general constraints $G(x) \leq h$ and $A(x) = b$. Although we can specify nonlinear parameter views with linear constraints on the posterior probabilities, we cannot specify them in full generality because we must fix some parameters to be able to specify specific view types. For example, consider the variance view

$$R_i^T \odot R_i^T x - \left(R_i^T x\right)^2 \gtreqless \tilde{\sigma}_i^2,$$

which is clearly nonlinear in the posterior probabilities x, where we use $\odot$ to denote the element-wise Hadamard product.

Note that we in the specification of views will use a concise programming broadcast notation, where we subtract scalars from vectors. We use this notation to replicate the way that these views would be implemented in most programming languages. From a strictly mathematical perspective, readers can imagine that there is a conforming S-dimensional vector of ones multiplied with scalar values such as

$\left(R_i^T x\right)^2$ that we do not want to constantly replicate.

To specify a variance view with linear constraints on the posterior probabilities, we must fix the second term to some value $\tilde{\mu}_i$ and specify the variance view through the two constraints

$$R_i^T x = \tilde{\mu}_i \quad \text{and} \quad R_i^T \odot R_i^T x - \tilde{\mu}_i^2 \gtreqless \tilde{\sigma}_i^2.$$

There is naturally the philosophical question of whether one can have a variance view without having a mean view. While we will not delve into this, we note that it is a fact that we would be able to implement the variance view with nonlinear functions on the posterior probabilities without specifying a mean view. Hence, as we will clearly see later in Section 5.2, the linear constraints pose a significant limitation. The original suggestion by Meucci (2008a) is to fix the mean to the prior, i.e., $\tilde{\mu}_i = R_i^T p = \mu_i$, while we note that this imposes an implicit view about the mean staying the same.

5.1.1 Solving the problem

The Entropy Pooling problem formulation (5.1.1) is a convex problem with linear constraints. Hence, the problem has one unique solution that is optimal, see Boyd and Vandenberghe (2004). The problem's Lagrangian function is

$$\mathcal{L}(x, \lambda, \nu) = x^T (\ln x - \ln p) + \lambda^T (Gx - h) + \nu^T (Ax - b),$$

where λ and ν are Lagrange multipliers. Meucci (2008a) shows that the solution is given by

$$x(\lambda, \nu) = \exp\left\{\ln p - \iota - G^T \lambda - A^T \nu\right\}, \tag{5.1.3}$$

with ι being an S-dimensional vector of ones. The solution (5.1.3) illustrates that positivity constraints on scenario probabilities $x \geq 0$ are automatically satisfied and can therefore be omitted.

The original / primal problem formulation is potentially very high-dimensional as it is a function of the number of scenarios S. On the other hand, the Lagrange dual function

$$\mathcal{G}(\lambda, \nu) = \mathcal{L}(x(\lambda, \nu), \lambda, \nu)$$

is only a function of the Lagrange multipliers λ and ν and therefore has dimension equal to the number of views in addition to a Lagrange multiplier for the requirement that posterior probabilities sum to one. Meucci (2008a) therefore proposes to solve the dual problem given by

$$(\lambda^\star, \nu^\star) = \underset{\lambda \geq 0, \nu}{\operatorname{argmax}}\ \mathcal{G}(\lambda, \nu)$$

and subsequently recover the solution to the original / primal problem (5.1.1) by computing

$$q = x(\lambda^\star, \nu^\star).$$

A fast and stable Python implementation for solving the Entropy Pooling problem is freely available in the open-source packages entropy-pooling and fortitudo.tech.

5.1.2 Common views specifications and ranking views

This section presents views specifications that are commonly used in practice. In particular, mean, volatility, skewness, kurtosis, and correlation views. The convenient feature of these views is also that it is easy to implement ranking views once we understand how views on these parameters are specified directly. The list of views in this chapter is by no means exhaustive. It is simply the views that I have seen being used the most in practice. Once you understand how to specify these views and the limitations the linear constraints impose, you are encouraged to experiment with other types of views.

Before we start writing out the views specifications, we start by defining classes of views. This will probably seem abstract to you at first, but it is actually quite simple once you grasp it. Understanding this view classification will be essential for understanding the sequential Entropy Pooling refinement in Section 5.2, which solves many of the issues imposed by the linear constraints requirement and usually gives significantly better results.

We have already seen that some views can be naturally specified through linear constraints without loss of generality, for example, views on the mean. However, views on variances require us to fix the mean in order for us to be able to implement them as linear constraints on the posterior probabilities. With this in mind, let $\mathcal{C}_i$, $i \in \{0, 1, 2, ...\}$, denote the class of parameters that require i other parameters from some or all of the classes $\mathcal{C}_j$, $j = 0, 1, .., i-1$, to be fixed in order to be formulated as linear constraints on the posterior probabilities.

Finally, let $\bar{\mathcal{C}}$ denote the class of parameters that can be formulated as linear constraints on the posterior probabilities but do not belong to any $\mathcal{C}_i$. Hence, $\mathcal{C} = \left\{\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, ..., \bar{\mathcal{C}}\right\}$ is the class of all parameters that can be formulated as linear constraints on the posterior probabilities, with or without fixing other parameters.

Many commonly interesting parameters can be characterized by the $\mathcal{C}_i$ classes. For example, as we have already seen, means belong to $\mathcal{C}_0$, and variances belong to $\mathcal{C}_1$ (mean fixed). Below, we will see that skewness and kurtosis belong to $\mathcal{C}_2$ (mean and variance fixed), and correlations belong to $\mathcal{C}_4$ (two means and two variances fixed), while CVaR views belong to the residual class $\bar{\mathcal{C}}$.

Looking at the linear constraints $Gx \leq h$ and $Ax = b$, we see that a particular Entropy Pooling parameter view is formulated as

$$f(R)\, x \gtreqless c,$$

where $c \in \mathbb{R}$ is a constant. The elements of the vector $x \in \mathbb{R}^S$ represents scenario probabilities. Hence, we must always include the constraint

$$\sum_{s=1}^{S} x_s = \iota^T x = 1,$$

which we implement through the equality constraints matrix A and vector b. As we saw from the solution to EP problem (5.1.3), the positivity requirement for scenario probabilities $x \geq 0$ will automatically be satisfied, so we do not need to include it in our constraints.

As we have already seen, a view on the expected value of price, return, or factor i is the easiest to implement as

$$R_i^T x \gtreqless \tilde{\mu}_i,$$

Note that most other views require multiple constraints in order for them to be formulated as linear constraints in a logically consistent way. For example, a view on the variance requires an equality view on the mean in addition to a view on the variance using the mean view value, i.e.,

$$R_i^T x = \tilde{\mu}_i \quad \text{and} \quad R_i^T \odot R_i^T x - \tilde{\mu}_i^2 \gtreqless \tilde{\sigma}_i^2.$$

If we do not fix the mean with a view on the expected value i, there is no guarantee that we are subtracting the right constant in the variance view. The interested reader is encouraged to try this out and calculate the posterior variance to verify that it usually becomes incorrect without the mean view. It is generally good practice to compute posterior statistics using the posterior probability vector q to verify that views have been implemented correctly.

With a good understanding of mean and variance views, we proceed to skewness, kurtosis, and correlation views. These views require us to fix means and variances, so we will assume that these have been fixed through the constraints

$$R_i^T x = \tilde{\mu}_i \quad \text{and} \quad R_i^T \odot R_i^T x - \tilde{\mu}_i^2 = \tilde{\sigma}_i^2.$$

The skewness view is then simply specified as

$$\left(\frac{R_i^T - \tilde{\mu}_i}{\tilde{\sigma}_i}\right)^3 x \gtreqless \tilde{\gamma}_i.$$

The kurtosis view is similarly specified as

$$\left(\frac{R_i^T - \tilde{\mu}_i}{\tilde{\sigma}_i}\right)^4 x \gtreqless \tilde{\kappa}_i.$$

Finally, correlation views are specified as

$$\frac{\left(R_i^T - \tilde{\mu}_i\right) \odot \left(R_j^T - \tilde{\mu}_j\right)}{\tilde{\sigma}_i \tilde{\sigma}_j} x \gtreqless \tilde{\rho}_{ij}.$$

The astute reader might have recognized that views on parameters are simply specified as their definition for discrete distributions, which the Monte Carlo market simulation R represents. We note that we make no assumptions on the actual market distribution being discrete, but any samples R from this distribution will be. It is up to the reader what they believe is a sufficiently good approximation, but sample sizes of $S = 10,000$ seem to be sufficient for most practical purposes, while keeping the computation time unnoticeable. Readers who are interested in seeing a practical example of all these view types being implemented are encouraged to see the case study of Vorobets (2021).

So how would we implement a ranking view? Simply by subtracting two views specifications from each other, for example, for views on the expected value

$$R_i^T x - R_j^T x = \left(R_i - R_j\right)^T x \gtreqless 0.$$

It can often be convenient to multiple one of the terms by a scalar a and specify views such as

$$(R_i - aR_j)^T x \leq 0.$$

This allows us to implement views such as $\mu_i \leq 2\mu_j$, i.e., that the expected value of i is at most $a = 2$ times larger than the expected value of j, assuming that $\mu_j > 0$.

We will not write out the ranking views for variances, skewness, kurtosis, and correlations, but these are also simply subtracting one view specification from another potentially scaled view specification on the same parameter type. It is a good exercise for readers to try this out on their own and compute the posterior values to see whether they have understood this concept or not. Ranking views can of course also be specified across different parameters, for example, that the expected value of i is half the variance of j. While this is a technical possibility, I have not seen it being done in practice yet.

With some practice, this section will hopefully give readers the understanding they need to explore Entropy Pooling views on their own. Readers are generally encouraged to share their experience and views ideas. Practically interesting use cases might be added to this section over time.

5.1.3 VaR and CVaR views

Having an understanding of the basics of Entropy Pooling views specifications, we are now ready to delve into VaR and CVaR views. While these views are also nothing more than linear constraints on the posterior probabilities, they have some characteristics that make them special. For example, it is not possible to easily implement ranking views on these parameters.

For general CVaR views, there is the additional complexity that we a priori do not know the number of scenarios below the VaR. If we have a VaR view, that is not an issue, while in the general case we must develop an algorithm for finding an optimal number of scenarios below a yet undetermined VaR value, which makes general CVaR views quite complicated to implement in a fast and stable way.

Meucci, Ardia, and Keel (2011) were the first to analyze VaR and CVaR Entropy Pooling views, while they end up proposing a solution that introduces a determistic grid, because their algorithm for CVaR views with fully general Monte Carlo distributions R does not perform well in practice. We will present VaR and CVaR views in this section without going into details about algorithms that solve the general CVaR view problem, because it is a highly specialized, complex, and proprietary implementation. However, you will be presented for the challenges and of course have access to the solution proposed by Meucci, Ardia, and Keel (2011).

VaR views are fairly straightforward to implement. Let us say that we have a α-VaR view given by $\tilde{v}$ for return i. Next, we identify the scenarios $s \in \{1, 2, \dots S\}$ where the return simulations R_i are below this α-VaR value and define the row vector $a_{i,\alpha} = (a_1, a_2, \dots, a_S)$ with elements

$$a_s = \begin{cases} 0 & \text{if} \quad R_{s,i} > -\tilde{v} \\ 1 & \text{if} \quad R_{s,i} \leq -\tilde{v}. \end{cases} \tag{5.1.4}$$

We can then simply add $a_{i,\alpha}$ to the matrix G or A as well as $1 - \alpha$ to the vectors h and b, depending on whether the view is an inequality or equality view. Hence, for VaR views we simply identify the

scenarios where the losses are greater than the VaR view value $\tilde{v}$ and assign them a total probability of probability $1-\alpha$. Since VaR views do not require any parameters to be fixed, VaR views belong to the class $\mathcal{C}_0$. Note that we use the convention that the VaR view value $\tilde{v}$ is specified as a loss, meaning that an α-VaR value of 10% assigns $1-\alpha$ of the probability mass to the scenarios where the elements of R_i are below -10%. We will use the same convention for CVaR views below.

CVaR views introduce an additional complexity when we do not have a VaR view, because we do not a priori know the optimal number of scenarios below the VaR, which we can adjust the probabilities of to achieve the desired CVaR view value $\tilde{cv}$. By the optimal number of scenarios below the still undetermined VaR value $\bar{v}$, we naturally mean the number of scenarios that gives us the lowest relative entropy while satisfying the CVaR view.

Once we have a fast and numerically stable algorithm to find the optimal number of scenarios below the VaR, CVaR views become straightforward to implement simultaneously with an equality VaR view implemented through the constraints $a_{i,\alpha}$ in A and $1-\alpha$ in b, replacing $\tilde{v}$ with $\bar{v}$ when computing a_s using (5.1.4). In the presentation of CVaR views below, we will assume that this has been done, so that the VaR is fixed to some value $\bar{v}$, which we did not have a view on.

Assuming that we know the number of scenarios below the $\alpha-$VaR value $\bar{v}$, α-CVaR views on return i with a view value $\tilde{cv}$ can be formulated as

$$a_{i,\alpha}x = (1-\alpha) \quad \text{and} \quad R_i^T \odot a_{i,\alpha}x \gtreqless -(1-\alpha)\,\tilde{cv}.$$

To understand why this view combination gives us the desired $\alpha-$CVaR value, let us define $\mathcal{CV} \subseteq \{1,2,\ldots,S\}$ as the set of indices below the $\alpha-$VaR value $\bar{v}$. Hence, the sample CVaR is then given by

$$\mathbb{E}\left[R_i | R_i \leq -\bar{v}\right] = \frac{\sum_{s\in\mathcal{CV}} R_{s,i}x_s}{\sum_{s\in\mathcal{CV}} x_s} = \frac{R_i^T \odot a_{i,\alpha}x}{a_{i,\alpha}x} \gtreqless \frac{-(1-\alpha)\,\tilde{cv}}{(1-\alpha)} = -\tilde{cv}.$$

We note again that we define the $\alpha-$CVaR value as a loss, which is why we have a minus in front of the view value $\tilde{cv}$.

As we do not know the number of scenarios below the VaR value a priori, CVaR views belong to the residual class $\bar{\mathcal{C}}$. If we have a VaR view, CVaR could in principle be considered a $\mathcal{C}_1$ view, but as we will see in Section 5.2 this would impose unnecessary implicit constraints, so it is never recommended to classify CVaR views as belonging to $\mathcal{C}_1$.

With CVaR views being classified as $\bar{\mathcal{C}}$, we can shed some more light on what intuitively happens when we compute the optimal number of scenarios below the VaR. Imagine a situation where we have implemented all views except the CVaR view for i, and that we do not have a VaR view for i. The brute-force solution is to loop through all possible configurations of scenarios below the VaR value, which there are in principle S of, and then select the solution with the lowest relative entropy that also satisfies the CVaR view. This is obviously a quite slow way of finding the optimal solution, so we must build algorithms that make good initial guesses and quickly identify the optimal number of scenarios without getting stuck or being too sensitive to the numerical issues that are inherent to practical implementations.

To further understand the issues with developing algorithms for finding the optimal number of scenarios below the VaR value, see Meucci, Ardia, and Keel (2011). Going into details with these

algorithms is beyond the scope of this book and would derail our focus. Interested readers can study the original article by Meucci, Ardia, and Keel (2011) to see a mathematical formalization of how such algorithms can be designed, while we note that their proposed algorithm for fully general Monte Carlo simulations R is not stable enough in practice.

Although we do not go further into detail with general algorithms for implementing the CVaR views, there is an example with a view implementing a 50% increase in the 90%−VaR and 90%−CVaR for daily S&P 500 returns. This gives readers a practical example of how VaR and CVaR views are eventually implemented. We also assess which effect the combined VaR and CVaR view for S&P 500 has on the STOXX 50 daily return distribution. Figure 5.1.1 shows the results. Readers are encouraged to examine the accompanying to this section for more details and see how the VaR and CVaR views are implemented and validated to verify that they really understand the constraints above.

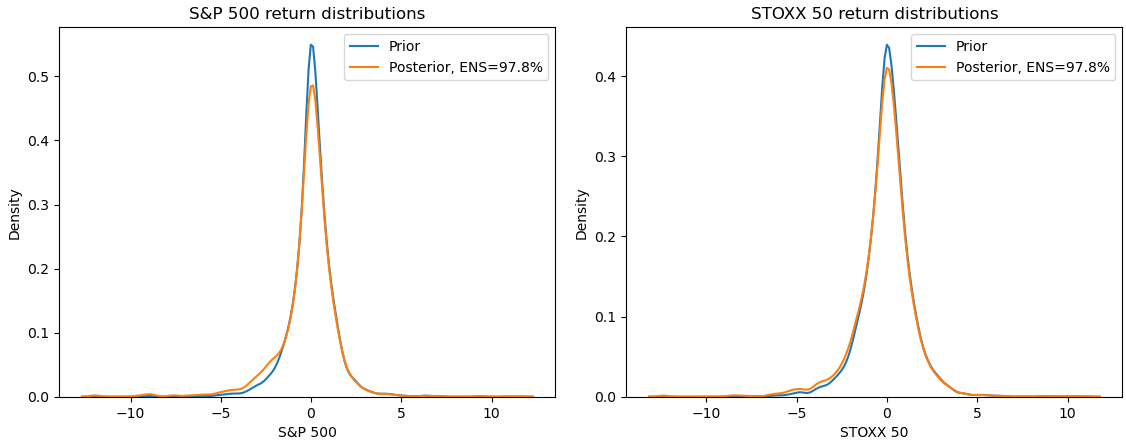

Figure 5.1.1: Daily return distributions including 90%−VaR and 90%-CVaR views for S&P 500.

As with almost all aspects of CVaR analysis for fully general distributions, it is significantly harder to implement CVaR Entropy Pooling views. However, the analysis of fully general distributions and their tail risks also gives us insights that are simply not possible to get using conventional methods based on the mean-variance oversimplification of the market. When we stress-test and analyze scenarios of historical or market simulations, we are really starting to look into the complex nuances of investment markets. Our imagination truly becomes the most limiting factor. Hence, although this section has given you some examples of practical use cases and views specifications, you should not be limited by these. Once you understand the Entropy Pooling technology, you might get ideas that go well beyond what is shown in this and the following sections.

A final interesting use case of CVaR views is to implement them directly on a portfolio's return and analyze how the marginal risk contributions change, both due to the higher standalone CVaR values but also due to changes in diversification properties. See Figure 5.1.2 for the results of such an analysis with a simple log-normal prior return simulation, where we have implemented a 50% increase in the 90%-CVaR of a portfolio containing 25% equities, 20% alternatives (including private equity), and the rest in government and corporate bonds. This example is generated using a proprietary implementation.

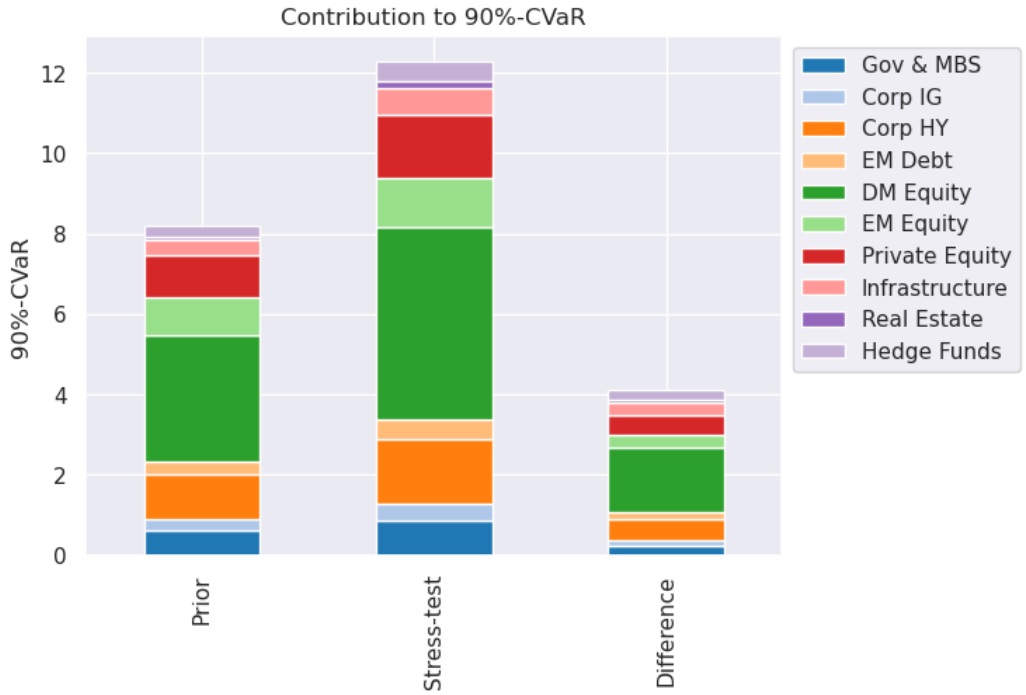

Figure 5.1.2: Portfolio CVaR view.

5.2 Sequential Entropy Pooling

As mentioned in Section 5.1 above, the requirement that views are specified as linear constraints on the posterior probabilities imposes significant limitations on the EP method. However, this requirement is necessary for us to be able to solve the problem quickly and reliably for high-dimensional market simulations R and their associated prior probability vector p. If we could solve the problem in a fast and stable way using general constraints on the posterior probabilities $G(x) \leq h$ and $A(x) = b$, we would do that. Unfortunately, we are not able to do that with the current technology.

The biggest issue stemming from the linear constraints requirement is that we must fix some parameter from the class $\mathcal{C}_i$ to be able to specify a view on a parameter from the class $\mathcal{C}_j$, with $i < j$. The issue stems from the fact that a view for the class $\mathcal{C}_i$ parameter might significantly affect the value of other parameters from the class $\mathcal{C}_i$. For example, if we have a view on the mean of STOXX 50, this view might significantly affect the mean of S&P 500. However, if we also want to specify a variance view for S&P 500, we must fix the mean of S&P 500. The original suggestion of always using the prior mean can therefore impose a potentially strong implicit view.

To overcome these practical limitations and solve additional practically interesting problems, Vorobets (2021) introduces heuristic sequential Entropy Pooling algorithms that sequentially process views according to their class characterizations $\mathcal{C}_i$. The fundamental hypothesis is that views that belong to the class $\mathcal{C}_i$ affect other parameters that belong to the class $\mathcal{C}_i$ more than parameters that belong to the class C_j, with $i \neq j$. For example, that views on expected value of i affect the expected value of j more than views on the variance of i affect the expected value of j.

This fundamental hypothesis is also what practitioners generally experience, while there is no guarantee that it is always true. It is perhaps also possible to generate simulations that specifically exploit the design of the sequential heuristics, but for simulations that resemble real-world market behavior and views on commonly interesting parameters, the assumption seem to hold most of the time.

We now proceed to define the sequential heuristic algorithms, closely following some sections from Vorobets (2021). A particular set of views $\mathcal{V}$ can be partitioned in a similar way to the view class $\mathcal{C}$, i.e., $\mathcal{V} = \left\{\mathcal{V}_0, \mathcal{V}_1, ..., \mathcal{V}_I, \bar{\mathcal{V}}\right\}$ with each $\mathcal{V}_i$ being the set of views on parameters that belong to $\mathcal{C}_i$, and $\bar{\mathcal{V}}$ being the set of views on parameters that belong to $\bar{\mathcal{C}}$. The main idea of the sequential heuristics is to process views according to this partition, carry forward the updated parameters θ_i, $i = 0, 1, ..., I$, and use them to set fixed values when specifying the views in $\mathcal{V}_j$, $j = i+1, i+2, ..., I$, and $\bar{\mathcal{V}}$. More specifically, EP is sequentially applied to the sequence of views with increasing cardinality given by $\mathcal{V}^0 = \{\mathcal{V}_0\}, \mathcal{V}^1 = \{\mathcal{V}_0, \mathcal{V}_1\}, ..., \mathcal{V} = \left\{\mathcal{V}_0, \mathcal{V}_1, ..., \mathcal{V}_I, \bar{\mathcal{V}}\right\}$. Note that the final set in this sequence contains all views $\mathcal{V}$, so the final posterior probabilities are guaranteed to satisfy all views, assuming that the views are feasible for the scenarios in R of course.

With the partitioning of the views established, the remaining question is which prior probability to use in the sequential processing. There are two natural choices in this regard. One is to use the original prior probability vector p, while the other is to use the updated posterior probabilities $q_0, q_1, ..., q_I$ associated with the updated parameters $\theta_0, \theta_1, ..., \theta_I$. This choice is exactly the difference between the two heuristics. Algorithm 1 (H1) uses the original probability vector p in all iterations, while Algorithm 2 (H2) uses the updated posterior probabilities $q_0, q_1, ..., q_I$, except the first iteration where p is used.

The two heuristics usually lead to similar final posterior probabilities q, but H1 is slightly better when measured by the relative entropy, as each relative entropy minimization step is against the original probability vector p, while H2 is usually slightly faster. Hence, H2 can be used if computation time is a crucial factor, while H1 is recommended for all other purposes.

Before presenting the sequential heuristics, some additional definitions must be established. For convenience, we define $q_{-1} = p$ and $\theta_{-1} = \theta_{prior}$. By $EP\left(\mathcal{V}^i, \theta, r\right)$ we mean that the EP method presented in Section 5.1 is applied to the set of views $\mathcal{V}^i$ using the parameter values in θ as fixed values when necessary and $r \in \mathbb{R}^S$ as the prior probability vector. Finally, $f\left(R, r\right)$ denotes the function that computes updated parameter values.

The two sequential heuristics are given by Algorithm 1 (H1) and Algorithm 2 (H2) below.

Algorithm 5.1 (H1)

for $i \in \{0, 1, ..., I\}$
  if $\mathcal{V}_i \neq \emptyset$, compute $q_i = EP\left(\mathcal{V}^i, \theta_{i-1}, p\right)$ and $\theta_i = f\left(R, q_i\right)$
  else $q_i = q_{i-1}$ and $\theta_i = \theta_{i-1}$
if $\bar{\mathcal{V}} \neq \emptyset$, compute $q = EP\left(\mathcal{V}, \theta_I, p\right)$
else $q = q_I$
return q

Algorithm 5.2 (H2)

for $i \in \{0, 1, ..., I\}$
  if $\mathcal{V}_i \neq \emptyset$, compute $q_i = EP\left(\mathcal{V}^i, \theta_{i-1}, q_{i-1}\right)$ and $\theta_i = f\left(R, q_i\right)$
  else $q_i = q_{i-1}$ and $\theta_i = \theta_{i-1}$
if $\bar{\mathcal{V}} \neq \emptyset$, compute $q = EP\left(\mathcal{V}, \theta_I, q_I\right)$
else $q = q_I$
return q

In the last five years, where I have worked with the sequential Entropy Pooling heuristics quite extensively, there has only been one extreme instance where the original heuristic of always using the prior value has given a better result. In all other cases, the sequential heuristics have given significantly better results. Readers are encouraged to test this out in practice on their own data and compare the performance gains through the relative entropy or effective number of scenarios as well as a visual assessment of the view / stress-test.

Below is a very simple case study using the H1 heuristic on daily S&P 500 and STOXX 50 data. The prior statistics are given in Table 5.1, while the posterior statistics for, respectively, the original Entropy Pooling heuristic and H1 are given in Table 5.2 and Table 5.3. We implement a view on STOXX 50 expected return and the volatility of S&P 500. The relative effective number of scenarios is 85% for the original heuristic and 89.4% for H1. For a more advanced case study involving skewness, kurtosis, and correlations views, see Vorobets (2021).

	Mean	Volatility	Skewness	Kurtosis
S&P 500	0.0373%	1.2650%	-0.2522	15.0594
STOXX 50	0.0133%	1.3990%	-0.0936	10.6756

Table 5.1: Prior statistics for S&P 500 and STOXX 50 daily returns.

	Mean	Volatility	Skewness	Kurtosis
S&P 500	0.0373%	**1.5180%**	0.2786	14.0392
STOXX 50	**-0.6862%**	1.8115%	-1.8402	9.4778

Table 5.2: Posterior statistics for S&P 500 and STOXX 50 daily returns with original EP heuristic.

	Mean	Volatility	Skewness	Kurtosis
S&P 500	-0.3630%	**1.5180%**	-2.1073	13.5475
STOXX 50	**-0.6862%**	1.8322%	-2.2120	12.2508

Table 5.3: Posterior statistics for S&P 500 and STOXX 50 daily returns with H1 EP heuristic.

From the tables above, we clearly see the constraint that the original Entropy Pooling heuristic imposes by fixing the mean of S&P 500, which is marked with an underscore in Table 5.2. Views are marked with bold. Figure 5.2.1 shows the prior and posterior distributions for both S&P 500 and STOXX 50. A visual inspection reveals that the sequential heuristic H1 also gives results that look nicer and more realistic, without sudden kinks in unexpected areas of the distribution.

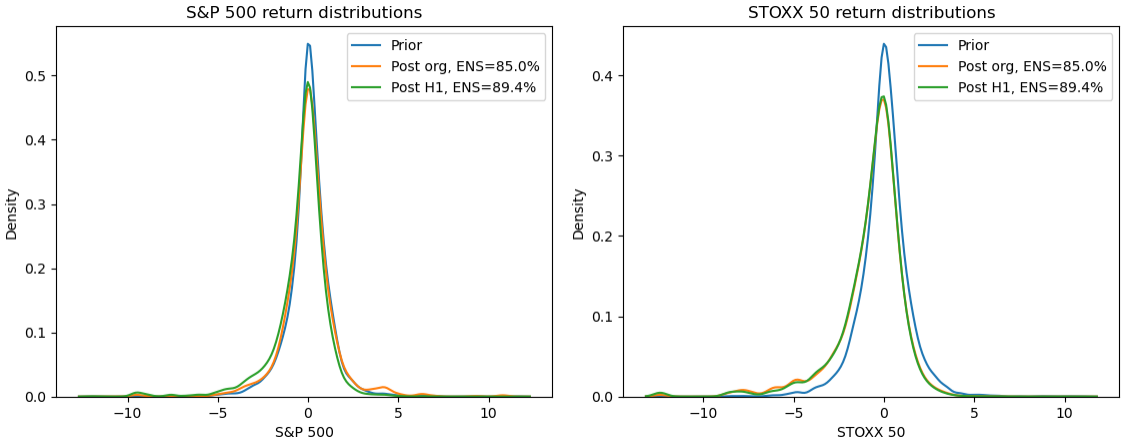

Figure 5.2.1: Prior and posterior distributions for S&P 500 and STOXX 50.

To understand the reasoning behind the sequential algorithms, the idea for both is to only fix parameters when we absolutely have to according to the class definitions $\mathcal{C} = \{\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, ..., \bar{\mathcal{C}}\}$. Hence, we allow views for the classes $\mathcal{C}_i$ to affect the parameters of the classes $\mathcal{C}_j$, $i < j$, for as long as we can. The difference between H1 and H2 is then whether we anchor the sequential updates in the latest intermediate posterior probability q_i or the prior p. You can think of H1 as looking into the future and then taking the information back to the prior to look further into the future, while H2 takes a step in the right direction and continues from there. As each iteration of H1 is against the prior probability, it is natural that it tends to give the best results, while the author currently cannot provide a proof that it will always be the case. Readers are encouraged to explore the methods and share their experiences.

With the above principles in mind, it hopefully becomes clearer why we characterized CVaR as a parameter belonging to $\bar{\mathcal{C}}$. There is simply no need for us to process it as a $\mathcal{C}_1$ parameter. In the algorithms that we use to find the optimal number of scenarios below the VaR, we quite clearly are likely to get a lower relative entropy if we allow views before $\bar{\mathcal{C}}$ to affect the final value of VaR before implementing the CVaR view.

The sequential algorithms do not only give us better results with lower relative entropies and higher effective number of scenarios. They also give us distributions that look more realistic. See for example the accompanying code to this section and another example in Figure 5.2.2 below. The example in Figure 5.2.2 implements several multi-asset views and compares the result of the original Entropy Pooling heuristic, which always fixes parameters to their prior values, and the H1 heuristic. These computations are performed using a proprietary implementation, so the accompanying code is not provided, but the views are exactly the same in the two cases. We simply see that the H1 heuristic gives significantly better results and more realistic-looking distributions.

There are some additional important benefits to the sequential heuristics. For example, they allow us to solve problems where we have a ranking view that must increase the expected value of i to make it higher than the expected value of j, and then implement views on higher moments for one or both of the assets. This is not possible with the original approach. It is also convenient that means are updated automatically, for example, in the case where we have views on two assets' returns and a view on the variance of a basket of the two. See Vorobets (2021) for more perspectives.

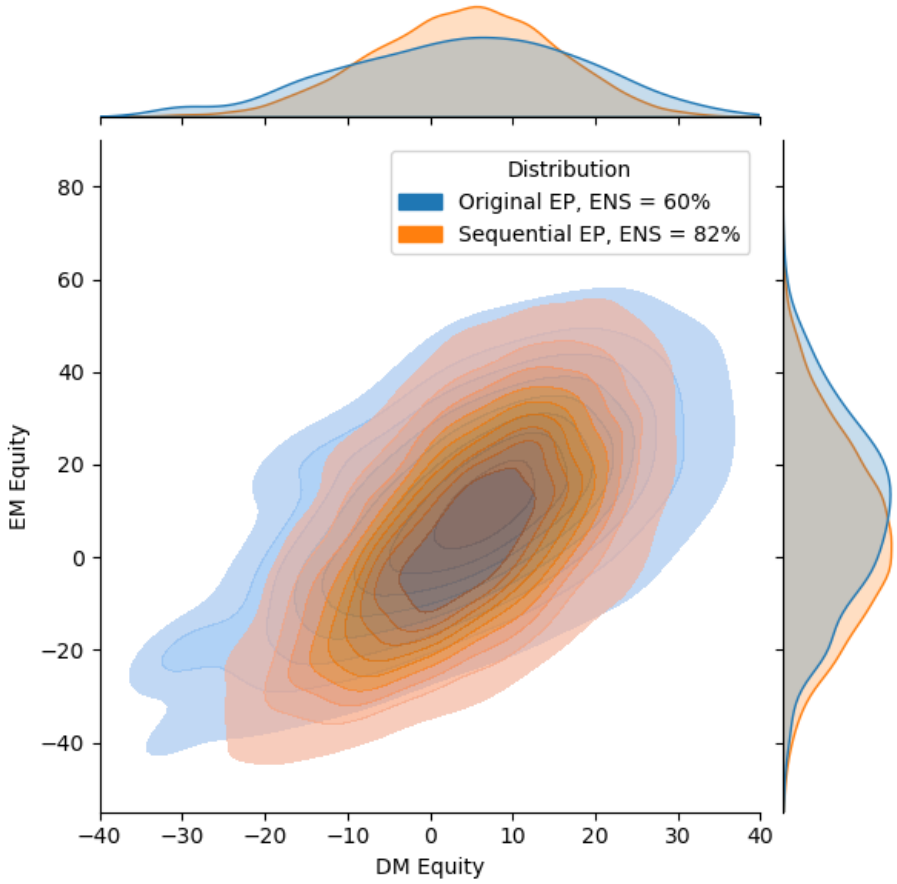

Figure 5.2.2: DM and EM equity posterior distributions.

5.3 View confidences and multiple users or states

Until now, we have assumed that you have full confidence in your Entropy Pooling views and stress-tests. While this sounds like a strong assumption, it is actually how Entropy Pooling is used most of the time in practice. People simply want to see what will happen to the market given their views, with an understanding that it is all uncertain including the prior Monte Carlo distribution R. View confidences, however, add an additional nuance to the Entropy Pooling method, and they are treated in a natural probabilistic sense. The general framework is presented in the appendix of Meucci (2008a), while we will present it in a slightly more simple way as well as include additional perspectives.

While view confidences are not very frequently used in practice, probabilities of different parameter values or market states are used quite frequently. This perspective will be foundational for the Causal and Predictive Market Views and Stress-Testing framework in Section 5.4 below. We will make a separation between these two perspectives. To be precise, view confidences $c \in (0, 1]$ are specified over a logically consistent set of views. Weights assigned to individual users and probabilities of different states $u \in [0, 1]$ can be specified over potentially conflicting set of views. It is probably still confusing to you what the difference is, so there are illustrative examples below.

In the simplest case, you have a set of views $\mathcal{V}$ and some common confidence c in these views. Hence, it is natural to allocate this confidence to the posterior probability vector $q_{\mathcal{V}}$ and the rest to the prior, so the final posterior becomes

$$q = cq_{\mathcal{V}} + (1 - c)\, p.$$

Note that we require that the total confidence sums to 1. We cannot logically be more than 100% confident in our views.

What happens when we have multiple views with multiple confidences? For example, that the yearly expected return of S&P 500 should be 10% with 70% confidence, and that the yearly expected return of STOXX 50 should be 12% with 90% confidence. What does that mean? Obviously, we cannot just allocate 70% confidence to one posterior probability vector q_{V_1} and 90% to another posterior probability vector q_{V_2}. If we think more carefully about it, it means that with 70% confidence you believe that the expected return of S&P 500 should be 10% and the expected return of STOXX 50 should be 12%. With additionally $90\% - 70\% = 20\%$ confidence, you believe that the expected return on STOXX 50 should be 12%. And finally, you allocate the last $1 - 90\% = 10\%$ to the prior. Ordering and partitioning views in this way is what we will define as view confidence.

Note that we use the notation V_i instead of $\mathcal{V}_i$ to distinguish between partitioning of views according to the sequential Entropy Pooling heuristics and view sets in general. To define how view confidence is handled in general, imagine that we have a set of views $\mathcal{V} = \{V_1, V_2, ..., V_I\}$ and their associated confidences $c = \{c_1, c_2, \dots c_I\}$, ordered from lowest confidence to highest confidence, i.e., $c_i \leq c_j$ for $i < j$. Let us then define $\bar{\mathcal{V}}_i = \{V_j | j \geq i\}$, e.g., $\bar{\mathcal{V}}_1 = \mathcal{V} = \{V_1, V_2, ..., V_I\}$ and $\bar{\mathcal{V}}_2 = \{V_2, V_3, ..., V_I\}$. Hence, the posterior probability vector with view confidences become

$$q = \sum_{i=1}^{I} (c_i - c_{i-1})\, q_i + (1 - c_I)\, p, \tag{5.3.1}$$

where $q_i = EP\left(\bar{\mathcal{V}}_i, p\right)$ and $c_0 = 0$ for convenience. Note that $EP\left(\bar{\mathcal{V}}_i, p\right)$ can be Entropy Pooling with the original heuristic or one of the sequential heuristics H1 or H2 from Section 5.2. The general formula (5.3.1) might seem complicated at first, but it is the same principle as describe in the simple case above with $c_1 = 70\%$ and $c_2 = 90\%$. It is a good exercise to verify that you can see this.

To conclude view confidences, we again underline that it must be a set of logically consistent views. For example, we cannot be 50% confident that a parameter is equal to 10 and 50% confident that it is equal to 12. We can, however, believe that there is a 50% probability that the parameter is equal to 10 and 50% probability that the parameter is equal to 12. This is assigning probabilities to states or weights to different users with conflicting views, which Entropy Pooling also can handle.

We can think of view confidences as the probabilities we assign within one user's views or one state, while the other probabilities and weights are assigned across states or users. This distinguishing will be important in Section 5.4 below, where we can define views with multiple confidences for each state and have conflicting view values across states. Handling of probabilities assigned to users and states is quite simple. We just need to make sure that they are positive and sum to one. How we determine these probabilities and view confidences is however more complex, but we have full flexibility.

5.4 Causal and predictive market views and stress-testing

This section presents the Causal and Predictive Market Views and Stress-Testing framework from Vorobets (2023). Contrary to the article, you now have a better understanding of Entropy Pooling view confidences and the state probability weighting of posterior probability vectors. The causal and predictive framework is essentially a combination of Bayesian networks (BNs) and Entropy Pooling, where the Bayesian network acts as a causal joint view generator that additionally produces the state probabilities for each posterior probability vector through the joint view probabilities. EP is then used in the usual way to project each of the joint views over the market simulation represented by the matrix R. Hence, the framework naturally combines and leverages the strengths of the two methods.

The idea of using Bayesian networks for causal market analysis gained traction after the introduction by Rebonato and Denev (2011), while the idea of combining Bayesian networks with Entropy Pooling was first introduced by Meucci (2012b). Contrary to the framework introduced by Meucci (2012a), which discretizes the Monte Carlo simulation R into bins and applies EP to multivariate histograms, the framework in this book does not impose such limitations. Instead, we work with the fully general market representation (1.1.1).

Rebonato and Denev (2014) give a careful treatment of the thoughts behind building Bayesian networks for investment analysis, which have inspired the basic use cases of the framework in this book. However, the book's framework can be used in many creative ways that go well beyond the original perspectives, see Vorobets (2023) for several case studies. The presentation of the framework will naturally draw quite heavily on Vorobets (2023), while hopefully being easier to understand given the deeper understand of Entropy Pooling, view confidences, and state probability weighting.

5.4.1 An introduction to Bayesian networks

We will start with a high-level introduction to Bayesian networks that will be sufficient for our purposes, but it might be beneficial to find some introductory literature for readers who are completely unfamiliar with graphs and BNs. A BN is a directed acyclic graph (DAG) that represents a set of variables and their conditional dependencies. Each node represents a variable and each directed edge (arrow) represents a conditional dependency. If two nodes are not connected by a directed path, they are said to be conditionally independent.

Figure 5.4.1 shows a simple BN. In this BN, we say that A, B, and F are root nodes, while E, D, and F are leaf nodes. We call a node root if it has no incoming edges, while we call a node leaf if it has no outgoing edges. Note that a node can be both a root and a leaf node, which is the case for node F in this example. It follows immediately that any leaf node is conditionally independent of any other leaf node, because leaf nodes only have incoming edges. Additionally, we say that a node X is a parent of the child node Y if X has an outgoing edge to Y. For example, in Figure 5.4.1 B and A are parent nodes for C. Similarly, A is a parent node for the child node D, and C is a parent node for child node E.

The conditional dependence between two variables in a BN does not necessarily represent a causal relationship, but that is the usual assumption, and we will make it in this framework as well. An important realization is that causality is almost always an assumption in investment applications and

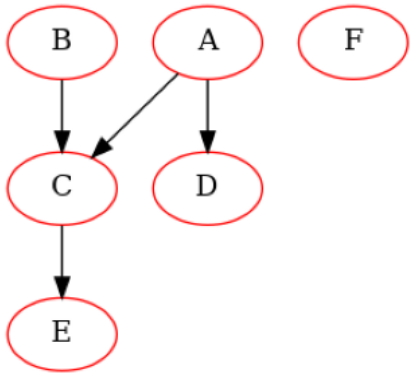

Figure 5.4.1: Simple Bayesian network illustration.

social sciences. It is usually non-trivial to prove causality, and it is also the hardest part of estimating BNs. Once the conditional dependencies are specified, estimating the probabilities of a discrete BN is straightforward. In this framework, we suggest that you specify both the conditional dependencies and probabilities for the BN. This gives a high level of flexibility, allowing you to hypothesize about and impose future causal relationships that are not necessarily strongly present in historical data. However, a BN where both the causal structure and probabilities are estimated based on historical data will work just as well in this framework.

Specifying the probability tables of a discrete BN might seem as a daunting task at first, but it is usually easier than one expects if the task is approached in a structured way. The following four step procedure works well in practice:

1. Specify the relevant variables (nodes) of the BN,
2. Specify the causal relationships (edges / arrows) between the variables,
3. Infer the size of the conditional probability tables,
4. Populate the conditional probability tables one at a time.

In all of these steps, it of course helps to have an elegant implementation of the BN technology to keep track of and manage all the information. Especially in step 3, it is convenient to let the tables be auto-generated. For step 4, it is also convenient to be able to specify probability ranges or leaving probabilities unspecified with some method helping you to estimate a probability based on the defined BN structure. A suggestion is to use maximum entropy for inferring missing values as described by Corani and Campos (2015).

With a high-level understanding of BNs, the question remains how to use them for investment analysis. The idea is well-described in Rebonato and Denev (2014), so it is only briefly summarized here: we want the relevant assets, returns, or factors to correspond to the leaf nodes of the BN. For example, for an asset allocation investor, the leaf nodes E, D, and F in Figure 5.4.1 could be variables like real rate, inflation, and risk premium, while the other nodes are variables that causally affect the distribution of the real rate, inflation, and risk premium, see the case study in Section 5.4.4 below.

Letting $X = (X_1, X_2, \ldots, X_N) \in \mathbb{N}^N$ denote the N-dimensional vector of random variables / nodes in a discrete BN, the joint probability can be computed using the well-known chain rule factorization

$$\mathbb{P}(X_1, X_2, \ldots, X_N) = \prod_{i=1}^{N} \mathbb{P}(X_i | pa(X_i)),$$

where $\mathbb{P}(X_i | pa(X_i))$ denotes the probability of X_i conditional on its parents. Since root nodes do not have any parents, $\mathbb{P}(X_i | pa(X_i)) = \mathbb{P}(X_i)$.

Letting $\mathcal{LN} \subseteq \{1, 2, ..., N\}$ denote the leaf node indices, the framework in this article mostly focuses on the joint distribution of the leaf nodes X_i, $i \in \mathcal{LN}$, given by

$$\mathbb{P}(\{X_i | i \in \mathcal{LN}\}) = \prod_{i \in \mathcal{LN}} \mathbb{P}(X_i | pa(X_i)). \tag{5.4.1}$$

The total number of joint leaf node probabilities is given by

$$J = \prod_{i \in \mathcal{LN}} S_i,$$

where S_i is the number of states for node $i \in \mathcal{LN}$.

5.4.2 Integrating Entropy Pooling

The presentation so far has brought us to a point where we understand how to specify a discrete Bayesian network. These networks can be interesting to analyze on their own, but they first become really interesting when we are able to project the discrete joint states of the leaf nodes over general Monte Carlo simulations R. This is what we use Entropy Pooling for.

It is strongly recommended to use the sequential Entropy Pooling refinements as presented in Section 5.2 above. However, it is important to underline that the framework also works with the original EP heuristic of always using prior parameter values when necessary, i.e., there is no reliance on any aspects of the sequential refinements from Section 5.2. They just usually give significantly better results.

We will denote the joint leaf node probabilities from (5.4.1) by p_j, $j = 1, 2, \ldots, J$, and use them as weights for the associated EP posterior probability vectors. These joint leaf node probabilities are natural weight candidates for posterior EP probability vectors because $p_j \in [0, 1]$ and $\sum_{j=1}^{J} p_j = 1$, which implies that the sum of the elements in $q = \sum_{j=1}^{J} p_j q_j \in \mathbb{R}^S$ is one for any set of valid EP posterior probability vectors $q_j \in \mathbb{R}^S$. The EP posterior probability vectors q_j are computed using EP for each of the joint leaf node events. Formally, each of the leaf node states $x_i = \{x_{i,1}, x_{i,2}, ..., x_{i,S_i}\}$ is mapped into EP views $v_i = \{v_{i,1}, v_{i,2}, ..., v_{i,S_i}\}$, $i \in \mathcal{LN}$, and combined using a Cartesian product over all v_i into J joint leaf node EP views.

The above weighting is an implementation of the state probability weighting approach. Specifically, the joint leaf node probabilities correspond to the u's from Section 5.3. Hence, the BN effectively helps us to generate these weights. View confidences c are directly specified for each leaf node state and therefore reflected in the posterior probability vectors q_j, $1, 2, \ldots, J$.

5.4.3 Additional perspectives

The framework is introduced with one variable for each leaf node, but it is actually much more flexible than that because we can use fully general EP views for each of the leaf node states. The only limitation is that a leaf node cannot contain views that contradict the views from another leaf node. For example, you cannot have a state in one leaf node with a view that some variable should be equal to 10, while you have a view in another leaf node that this variable should be equal to 20. You must implement these views using the same node with two different states to ensure logical consistency in the joint views. This is the only restriction.

It can be tempting to interpret the framework in the following way: the causality comes from the BN, while the predictiveness comes from EP. This is a reasonable way to think about the framework, but it is strictly speaking not correct. It is clear that the BN defines causal relationships, this is one of its main features. However, there is also an element of predictiveness when one conditions on realizations of the variables, making some events more and less likely. Similarly with EP, where the predictiveness aspect is clear, while there can also be causal elements in the market simulation, for example, interest rates causing changes in bond prices.

It is important to realize that assuming the causal relationships in the BN introduces an EP view, even when we do not condition on realization of relevant variables. How much this view deviates from the prior can be assessed by the relative entropy between the prior and unconditional posterior. If the prior is uniform in scenario probabilities, the (relative) effective number of scenarios can also be used to assess how concentrated scenario probabilities are.

Although the BN defines a causal structure, it can also be used to answer non-causal questions. For example, if one has defined a BN where a central bank's decision depends on inflation and employment, one can condition on the central bank being hawkish and compute the probability of high inflation or low unemployment, i.e., answering the question of how the distribution of these variables should be in order for the central bank to be hawkish. These results can then be compared to what is implied by the market and used for taking positions.

A full implementation of the framework does not only have a good interface for BNs combined with maximum entropy for estimating partially specified probabilities, but also an integration of sequential EP presented in Section 5.2 with support for rank views, view confidences, and CVaR views. Building such an implementation is a very daunting task that should probably only be attempted by the most determined. However, once successful the framework allows investment and risk managers to perform sophisticated market views and stress-testing analysis at a level that is way beyond current standards.

For case studies using the framework presented in this section, see Vorobets (2023). There is a simple Bayesian network example available as open-source code using the fortitudo.tech Python package. There is even a video walkthrough of the article and the accompanying code. This book will purposefully not replicate more from the article than what has already been done, except present some perspectives on asset allocation cases in Section 5.4.4 below, which are interesting and easy to relate to for most people.

We conclude this section by stating that the framework is incredibly flexible and powerful in the hands of skilled quantamental investment practitioners, who are usually very excited about its possibilities.

5.4.4 Asset allocation case study

A popular case that most investment managers can relate to is an analysis of the macroeconomy and translation into portfolio P&L. Specifically, consider the Bayesian network given in Figure 5.4.2. This is a plain vanilla application of the framework, where each leaf node consist of a key macro risk factor. Many asset allocation investors think about the risk in their portfolios from a perspective similar to this one.

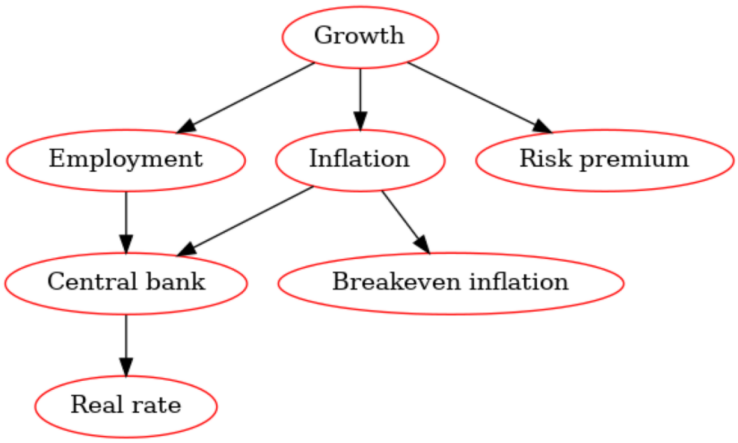

Figure 5.4.2: Asset allocation Bayesian network.

Figure 5.4.3 shows the result of a stagflation and rate hike stress-test using this Bayesian network for a low risk portfolio and its tail risk hedge. The case uses a proprietary implementation and is therefore not available in the accompanying code.

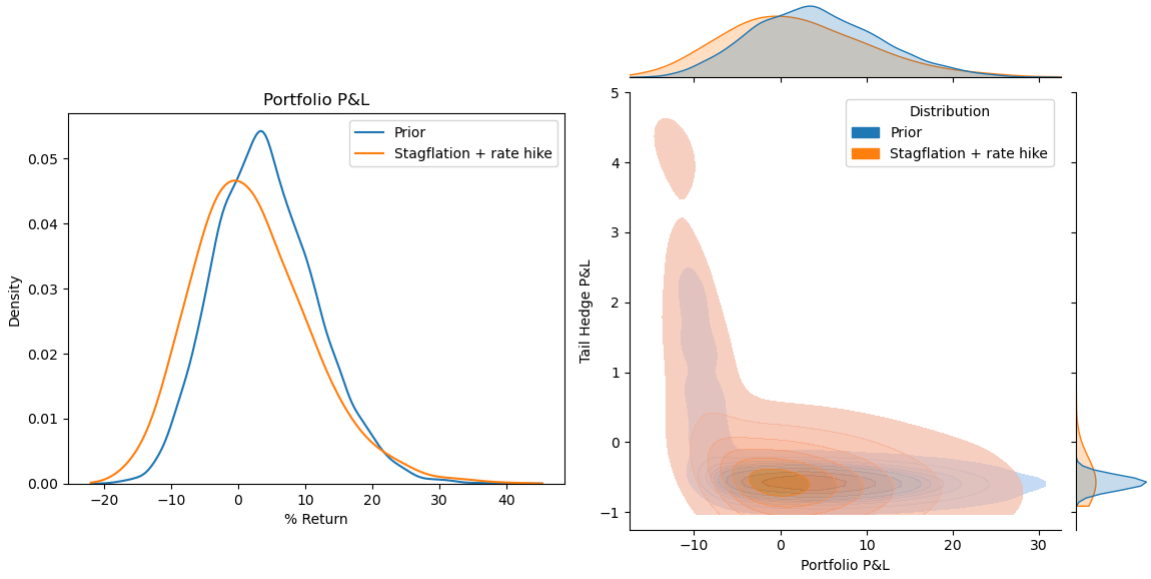

Figure 5.4.3: Portfolio and tail hedge prior and posterior P&L.

Chapter 6

Portfolio optimization

This chapter focuses on portfolio optimization, and in particular CVaR portfolio optimization for fully general Monte Carlo simulations R and associated probability vectors p and q, i.e., the starting point presented in (1.1.1). An elegant aspect of CVaR optimization in practice is that it operates directly on the market simulations R and the associated probability vectors p and q. Hence, it does not matter how complex our simulations or market views and stress-tests are, CVaR will give us meaningful results.

Section 6.1 presents the portfolio management framework that we work with throughout the book, first documented by Vorobets (2022a). By making a separation between relative exposures e and relative market values v, we can handle derivative instruments as easily as plain vanilla cash instruments such as stocks and bonds.

Section 6.2 compares CVaR optimization to the traditional variance optimization, showing you how CVaR optimization problems can be solved with linear programming, and giving you an understanding of why CVaR optimization is much harder to implement in practice than variance optimization. However, fast and stable algorithms that make CVaR optimization practically feasible exist.

Section 6.3 presents the portfolio optimization problems with multiple risk targets, in particular a risk target for overall portfolio risk and a risk target for deviations from a benchmark portfolio. The joint optimization of these two risk targets introduce important trade-offs between the risk of the benchmark and the risk of the deviations from the benchmark. This analysis is initially presented for variance optimization, because it makes it easy for us to build intuition by representing diversification with the correlation, while we eventually generalize the results to CVaR.

Section 6.4 introduces parameter uncertainty into the portfolio optimization problem, which is necessary to incorporate in most practical cases. We focus in particular on resampled optimization with the perspectives and methods presented by Kristensen and Vorobets (2024). While derivatives are easy to handle in general portfolio management, they introduce significant complexities when it comes to portfolio optimization with parameter uncertainty. Conveniently, the Entropy Pooling method, which is thoroughly presented in Chapter 5, helps us solve these problems, see Vorobets (2024).

We end the chapter with Section 6.5 that introduces a new method for intelligent portfolio rebalancing, which is presented for the first time in this book. Portfolio rebalancing is such an essential part of portfolio management, but it is usually handled in an ad hoc manner without any framework for thinking about the rebalancing problem. This book suggests a framework for a structured analysis.

6.1 Exposures and relative market values

The most well-known portfolio in investment management is perhaps the long-only portfolio characterized by the constraints $\sum_{i=1}^{I} w_i = 1$ and $w_i \geq 0$, with w_i representing the weight of the total portfolio value invested in asset i. There are no issues with this characterization for simple portfolios consisting of only cash instruments and in fact being long-only. However, modern portfolios are increasingly utilizing derivatives, making the traditional framework insufficient.

Not much attention has been given to generalize the traditional framework to portfolios that invest in derivative instruments. As a consequence, practitioners usually try to treat derivative portfolios in the same way as portfolios investing in cash instruments only. A key quantity in this attempt is the offsetting cash position, introduced to make everything sum to one by offsetting the leverage introduced by the derivative instruments.

This book argues that although the offsetting cash approach might seem like a natural first guess by trying to somehow replicate the derivative exposure, it is in fact unnecessarily complicated. It is much easier and fairly straightforward to treat the derivative positions as they are by properly separating exposure / notional and market value / price. The exposures are then used for all aspects of portfolio management, while prices are used for elementary bookkeeping, see Vorobets (2022a) for more perspectives.

The key realization you have to make is that a derivative instrument is characterized by its exposure / notional and market value / price. To understand the need for separating these two quantities, consider a futures contract. The futures contract has initial market value of zero, but a positive or negative exposure depending on the direction of the position. It follows immediately that it is meaningless to use the market value of the futures contract as a measure of exposure, because it will appear as if the contract has no effect on the portfolio's future P&L distribution. Hence, it is important to realize that it is the exposure that determines the portfolio's future P&L distribution, while the market value is used for elementary bookkeeping.

The usual self-financing constraint found in, e.g., portfolio optimization problems reads something like

$$\sum_{i=1}^{I} w_i = \iota^T w = 1, \tag{6.1.1}$$

with ι being an I-dimensional vector of ones. With the above reasoning in mind, this book argues that the self-financing constraint should be more accurately expressed as

$$\sum_{i=1}^{I} v_i e_i = v^T e = 1, \tag{6.1.2}$$

with v being an I-dimensional vector of relative market values (the value of the derivative instrument with exposure / notional set to one), and e being an I-dimensional vector of exposures relative to portfolio value.

For plain vanilla cash instruments like stocks and bonds, market value and exposure are the same. Hence, $v = \iota$ and $e = w$. This illustrates how (6.1.1) is in fact a special case of (6.1.2). Now imagine that we are allowed to invest in at-the-money forward European put and call options. The price of

these options with one year to expiry, zero interest rate, zero dividends, implied volatility of 15%, and a notional of $E_i = 100$ is approximately $V_i = 5.98$, see the accompanying code to this section. So what is v_i in this case? That is easily calculated as $v_i = \frac{V_i}{E_i} = \frac{5.98}{100} = 0.0598$.

The vector of relative market values v makes it easy for us to calculate the value V_{pf} of any portfolio characterized by the exposures vector $E \in \mathbb{R}^I$, i.e.,

$$v^T E = V_{pf}. \tag{6.1.3}$$

To compare (6.1.3) to (6.1.2), we multiply (6.1.2) by V_{pf} to get

$$v^T e V_{pf} = V_{pf}. \tag{6.1.4}$$

From (6.1.3) and (6.1.4), it follows that $eV_{pf} = E \Leftrightarrow e = \frac{E}{V_{pf}}$. This illustrates precisely how e is the vector of exposures relative to portfolio value V_{pf}.

So far, exposure has not been explicitly defined. The recommendation is to use the notional for both derivative and cash instruments. If the notional is not directly given in the term sheet of the derivative instrument, the necessary information (e.g. contract unit) for computing the notional is. For plain vanilla cash instruments, notional is simply equal to the market value / price. Hence, there is no need for complicated computations when exposure is measured by the notional.

Another benefit of using the notional is that it will often correspond to the cash equivalent of the position. For example, an equity futures contract with a notional exposure of $100 gives the same exposure to price movements as a $100 cash position in the underlying, making it easy to work in conceptually similar ways for these instruments. Conclusively, notional is often the best conceptual approximation to the conventional portfolio weight, and it is easy to compute.

The most important nuance introduced by this portfolio management framework is the clear separation of portfolio weights given by $v_i e_i$ and portfolio exposures given by e_i. Hence, the sum of the weights is still required to be one as in the conventional framework (6.1.1), while there are no a priori restrictions on the sum of exposures. This underlines the main point that weights are used only for elementary bookkeeping, while exposures are relevant for the future portfolio P&L distribution.

Since we are usually interested in optimizing the portfolio's return and decompose its risk, it is interesting to examine how the return equation looks for this case. Relative P&L over the next period is naturally defined as $\Delta v_i = \frac{V_i - V_{i,0}}{E_{i,0}}$, with $V_{i,0}$ being the initial value and $E_{i,0}$ the initial exposure of position i. Summing over the products of relative P&L Δv_i and relative exposures $e_i = \frac{E_{i,0}}{V_{pf}}$, it follows that the portfolio percentage return is given by

$$r_{PF} = \sum_{i=1}^{I} \Delta v_i e_i = \sum_{i=1}^{I} \frac{V_i - V_{i,0}}{E_{i,0}} \frac{E_{i,0}}{V_{pf}} = \frac{1}{V_{pf}} \sum_{i=1}^{I} V_i - V_{i,0}. \tag{6.1.5}$$

Note that (6.1.5) is exactly how portfolio return is conventionally computed for cash portfolios. Hence, instead of percentage returns and portfolio weights, we can use the relative P&L Δv_i and exposures e_i relative to portfolio value V_{pf} for portfolio optimization and risk decomposition.

For realistic optimization and rebalancing applications, the self-financing constraint (6.1.2) should

include transaction costs that are payable at the time of the trade. In that case, (6.1.2) becomes

$$v^T e + TC\,(e - e_0) = 1,$$

where $TC\,(e - e_0)$ is the transaction cost function (implicitly normalized by portfolio value V_{pf}) for the turnover $e - e_0$, with $e_0 \in \mathbb{R}^I$ being the vector of initial exposures relative to portfolio value.

Using the traditional cash-based framework in (6.1.1), long-short portfolios are usually characterized by the constraint

$$\sum_{i=1}^{I} w_i = 0.$$

However, it is rarely the case (if ever) in practice that one is allowed to take margin free risk, e.g., borrow a stock to sell it in the market and use all of the proceeds to take a long position. Usually, the portfolio must be collateralized in some way. The value of the collateral / margin is thus the value of the portfolio, and the long-short constraint is therefore more correctly implemented by the requirement that

$$\sum_{i \in LS} e_i = 0,$$

where $LS \subsetneq \{1, 2, ..., I\}$ is the subset of instruments that can be used for the long-short exposures.

6.2 CVaR vs variance optimization

Investors are rarely worried about their returns being too high or their portfolio's return distribution being skewed too much to the upside. Both of these properties should indeed be desirable under normal circumstances where there are no strange incentives that affect investor preferences for higher risk-adjusted returns. The best examples of positive skewness being a desirable property are perhaps put and call options, where investors have historically been willing to pay a volatility risk premium above the fair value for a long position in these instruments, see Section 2.1. However, mean-variance optimization treats upside and downside deviations as equally undesirable, clearly going against this important principle.

Rockafellar and Uryasev (2000) were the first to introduce a practical method for solving CVaR portfolio optimization problems. They show that solutions can be found using linear programming for fully general Monte Carlo distributions like (1.1.1). While linear programming sounds nice and easy, we will see in Section 6.2.1 below that it is nontrivial to get good performance, and that the original formulation is probably too slow and unstable for practically relevant CVaR optimization that includes resampled parameter uncertainty as presented in Section 6.4 below. However, fast and stable algorithms that exploit the structure of the problem exist.

Proposition 1 of Rockafellar and Uryasev (2000) allows us to determine when CVaR and variance optimization coincide. If the return distribution is elliptical (for example normal or t-distribution), results will always coincide when the expected return constraint is binding. Mean-CVaR optimization results always coincide with mean-variance optimization when demeaned portfolio returns are used and instrument returns are jointly normal, see Vorobets (2022b) for more information and a case study

verifying this result. Hence, you lose nothing from using CVaR in the oversimplified textbook case, while you gain a lot when it comes to real-world investment analysis.

If we use demeaned returns when computing CVaR, it becomes more comparable to other deviation measures, e.g., variance and (lower semi-)absolute deviation. Since most portfolios in practice have a positive expected return, it will also give us more conservative risk estimates, which in may practical cases is arguably better than underestimating the risk. Readers can freely decide what they believe is best for their purposes, while a formal treatment of the differences between these two choices is given by Rockafellar, Uryasev, and Zabarankin (2006).

Besides giving meaningful results for fully general Monte Carlo distributions (1.1.1) and focusing on investment tail risk, what are some other benefits of using CVaR? First of all, contrary to VaR, CVaR is a coherent risk measure, respecting the diversification principle in (1.3.1), which is arguably a desirable feature for an investment risk measure, see Artzner et al. (1999). From a less technical perspective, it becomes harder to hide significant risks below the VaR value, as the mean is sensitive to outliers. Hence, CVaR is steadily overtaking VaR to become the preferred investment tail risk measure among both market makers and investment managers.

While all of the above should be sufficient to stop using variance and start using CVaR for investment practitioners, there is an additional important aspect to CVaR. It is much easier to interpret than variance for investment clients and other nontechnical stakeholders. For example, if you ask a regular person which risk they can tolerate as the expected loss in the worst year out of 10 (90%-CVaR), it is probably much easier for them to answer this question than which level of expected squared deviations from the mean they can tolerate.

As an investment practitioner, you might have developed some sense of what 5% and 10% yearly volatility looks like, but regular people do not have this sense. The ease of interpretability is very important for nontechnical people on boards and asset management clients. Finally, if you tell a regular person that your portfolio optimization method focuses as much on minimizing positive deviations from the mean as negative deviations, you probably will see someone who looks very strangely at you and tells you not to do that.

It is mainly finance and economics academics, or people who otherwise feel reputationally invested in the mean-variance system they have used for the past many years, who continue to defend mean-variance analysis despite its obvious and severe deficiencies. One of the seemingly scientific arguments that these people sometimes present is that "CVaR implies risk neutrality below the VaR value". This argument is based on a utility theory definition of risk aversion, which in itself is known to be a very poor representation of how people actually behave.

That utility theory is a good representation of reality seems to be a rumor among finance and economics academics, which they take for granted without being able to share any studies that actually show this. For a careful treatment of the issues with utility theory, see Friedman et al. (2014). Interestingly, Friedman et al. (2014) conclude that people seem to behave more according to a linear utility function subject to constraints, which fully agrees with practical CVaR optimization and analysis.

If utility theory was a good representation of how people actually behave, almost anyone who was presented with the CVaR risk measure should feel a deep sense of unease by the "risk neutrality below the VaR". However, people's reaction tend to be quite the opposite, where they feel that it

represents their preferences for avoiding losses quite well. If you tell them that minimizing the upside is a byproduct of your optimization method, you will probably see them become very uneasy. Hence, the reality is that mean-variance optimization does not represent what people actually want to do. Interestingly, Markowitz (1952) already acknowledged this and argued that the focus should be on the downside, but this was practically unthinkable with the technology that was available at the time.

Perhaps even more intriguing, Harry Markowitz told in an interview that he did not use mean-variance to manage his own portfolio. He used the $1/N$ heuristic. While we will rely on heuristics when it comes to resampled portfolio optimization in Section 6.4, the methods and framework recommended in this book are actually used to manage the author's own money.

We will not delve more into utility theory or the attempts to justify the continued use of mean-variance analysis. The author's hypothesis is that it is mainly driven by reputational investments in this theory and the ease of implementing mean-variance optimization compared to mean-CVaR optimization. We will not rely on any aspects related to utility theory and simply note that CVaR seems to be a good representation of the investment risk that people want to avoid. For a more detailed comparison between variance and CVaR as investment risk measures, see Vorobets (2022b).

6.2.1 Solving CVaR problems

A practical way to solve CVaR problems was first introduced by Rockafellar and Uryasev (2000), while equivalent formulations of the risk-adjusted return objectives and general, scenario probability weighted Monte Carlo distributions (1.1.1) are analyzed by Krokhmal, Palmquist, and Uryasev (2002). The CVaR theory and the justification for solving the problem using linear programming through a discretization and linearization are quite mathematically advanced. It is intentionally not replicated here. We simply focus on the problem formulation using the notation from this book.

Focusing initially on the case where we want to minimize $\alpha-$CVaR, potentially subject to an expected return constraint, the linear programming problem is given by

$$e^\star = \underset{VaR,e}{\operatorname{argmin}}\ VaR + \frac{1}{1-\alpha}p^T y$$

subject to

$$\begin{aligned} y_s &\geq -R_s e - VaR, \\ y_s &\geq 0, \\ \mu_{target} &\geq \mu^T e, \\ v^T e &= 1, \\ e &\in \mathcal{E}. \end{aligned}$$

In the above, $y = (y_1, y_2, \ldots, y_S)^T$ is a vector of auxiliary variables, while R_s represents row s from the Monte Carlo simulation R, and p is the associated joint probability vector from (1.1.1). Finally, $\mu = R^T p$ is the vector of expected relative P&L's, while v is the vector of relative market values introduced in Section 6.1. We use $\mathcal{E}$ to represent the set of linear (in)equality constraints on the portfolio exposures e, forming a convex polyhedron.

To focus on the essence of the problem, we do not include transaction costs, which require an additional introduction of auxiliary variables $e^+ \geq 0$ and $e^- \geq 0$, representing the buys and sells in addition to the constraints $e = e_0 + e^+ - e^-$, $e^+ e^- = 0$, and an extension of the self-financing constraint to $v^T e + TC\,(e - e_0) = 1$ as explained in Section 6.1. We would also need to subtract the transaction costs from the expected return as explained by Krokhmal, Palmquist, and Uryasev (2002).

From the CVaR optimization problem, we immediately notice that it requires an introduction of S auxiliary variables y_s, for $s = 1, 2, \ldots, S$, in addition to $2S$ extra constraints. Hence, even though it can be formulated as a linear programming problem, it is potentially very high-dimensional and introduces a trade-off between computation time and approximation quality due to the discretization and linearization of the objective function. In practice, the original formulation of the CVaR optimization problem is significantly slower and less stable than the traditional mean-variance optimization. This fact is especially inconvenient as portfolio optimization in practice almost always needs to include parameter uncertainty as presented in Section 6.4 below.

Using the original formulation, solving CVaR problems subject to CVaR constraints is as straightforward as solving problems with expected return constraints. Krokhmal, Palmquist, and Uryasev (2002) show that these problems simply require us to change the optimization objective to

$$e^\star = \underset{VaR,e}{\operatorname{argmin}} \; -\mu^T e = \underset{VaR,e}{\operatorname{argmax}} \; \mu^T e$$

and add the CVaR formulation to the constraints

$$VaR + \frac{1}{1-\alpha} p^T y \leq CVaR_{target},$$

while of course removing the expected return constraint from our initial formulation above.

Krokhmal, Palmquist, and Uryasev (2002) show that we solve equivalent problems in both cases, so it does not matter which method we use for the same combination of risk and return. This is a feature that we will use in Section 6.3.1 below. While solving the problem with a single CVaR target is straightforward, albeit slow and potentially unstable, solving the problem with multiple CVaR targets become much more complicated. Imagine for example if we had two CVaR targets that would require $2S$ auxiliary variables and $4S$ constraints to solve the problem. Whatever issues we have with speed and stability from just one CVaR target are guaranteed to be amplified in this case.

The most sophisticated investment managers usually want to solve CVaR optimization problems with multiple risk constraints as presented in Section 6.3 below. Fortunately, there exist fast and stable algorithms to solve CVaR problems. These are, however, hard to discover and very complex to implement. Due to their proprietary nature, they cannot be shared in this book, while a fast and semi-stable version of an algorithm solving the problem with a return target can be found in the fortitudo.tech package.

It is left as an exercise for readers to really test out whether they understand the CVaR problem formulation and solve a problem for a portfolio with a 5% return target and 25% individual upper bounds on cash instruments as well as -50% to 50% bounds for derivative instruments, i.e., solving the prior and posterior optimization problems from Vorobets (2022a). See the accompanying code to this section, which gives you results for these problems using the fortitudo.tech package.

6.3 Risk budgeting and tracking error constraints

The easiest way to generate a higher return is just to take more risk. For example, instead of a futures position with a notional of \$100, you increase the exposure to \$200. That does not require skillful investment management, only a bit more margin. It is especially easy to take more "market beta" risk, so it is not something that would be worth paying an asset manager for. However, building portfolios with a good balance between tail risk and return requires a lot of skill and is worth paying for. Throughout this book, we assume that you as an investment manager are determine to maximize the risk-adjusted return and not simply maximize the return as many bonus schemes unfortunately incentivize. Hence, good portfolio construction is focused on maximizing the expected return subject to risk constraints.

Following Vorobets (2022b), we will start with an analysis of the portfolio variance, because it is easy to understand and implement, while going into more details than the article and having an Entropy Pooling case studies. The starting point of our analysis is a decomposition of the portfolio return into a return on a benchmark / strategic / long-term portfolio and a tracking error / tactical / short-term portfolio

$$r_{PF} = r_{BM} + r_{TE}. \tag{6.3.1}$$

This is usually how most investment managers think about the risk and returns of their portfolios, while it can of course be partitioned in many other ways. To make it easier for us to analyze the main ideas, we stick to the usual decomposition (6.3.1).

The return decomposition above implies that the portfolio variance can be expressed as

$$\sigma_{PF}^2 = \sigma_{BM}^2 + \sigma_{TE}^2 + 2\rho_{BM,TE}\sigma_{BM}\sigma_{TE}. \tag{6.3.2}$$

A meaningful risk budgeting workflow is to initially set the benchmark / long-term portfolio (in many cases it is externally given to the investment manager) and then risk budget using σ_{PF} and σ_{TE}. An important but perhaps trivial observation is that once the benchmark risk σ_{BM} is fixed, there are only two degrees of freedom among the remaining parameters σ_{PF}, σ_{TE}, and $\rho_{BM,TE}$.

Several interesting insights can be extracted from analyzing (6.3.2). For example:

1. The risk contribution from the tracking error portfolio is amplified by $\rho_{BM,TE}$ since $\frac{\partial \sigma_{PF}^2}{\partial \sigma_{TE}} = 2\sigma_{TE} + 2\rho_{BM,TE}\sigma_{BM}$,
2. A tracking error portfolio reduces the risk of the overall portfolio σ_{PF} when $\sigma_{TE}^2 + 2\rho_{BM,TE}\sigma_{BM}\sigma_{TE} < 0 \Leftrightarrow \frac{\sigma_{TE}}{\sigma_{BM}} < -2\rho_{BM,TE}$, which can only happen when $\rho_{BM,TE} < 0$.

The practical implication of 1. is that tracking error underestimation affects portfolio risk σ_{PF} more adversely the higher the correlation $\rho_{BM,TE}$ is, i.e., you are more exposed to risk overshoots due to estimation error related to σ_{TE}, e.g., if $\sigma_{TE} = 3\%$ while you estimate $\hat{\sigma}_{TE} = 2\%$. Figure 6.3.1 below illustrates this for various correlations $\rho_{BM,TE}$ and tracking errors σ_{TE}. This figure clearly illustrates that the risk overshoots increase with the correlation. See the accompanying code to this section for the details of how these computations have been performed, and think about how the graph will look for $\rho_{BM,TE} = 1$. You can potentially use the code to validate your intuition.

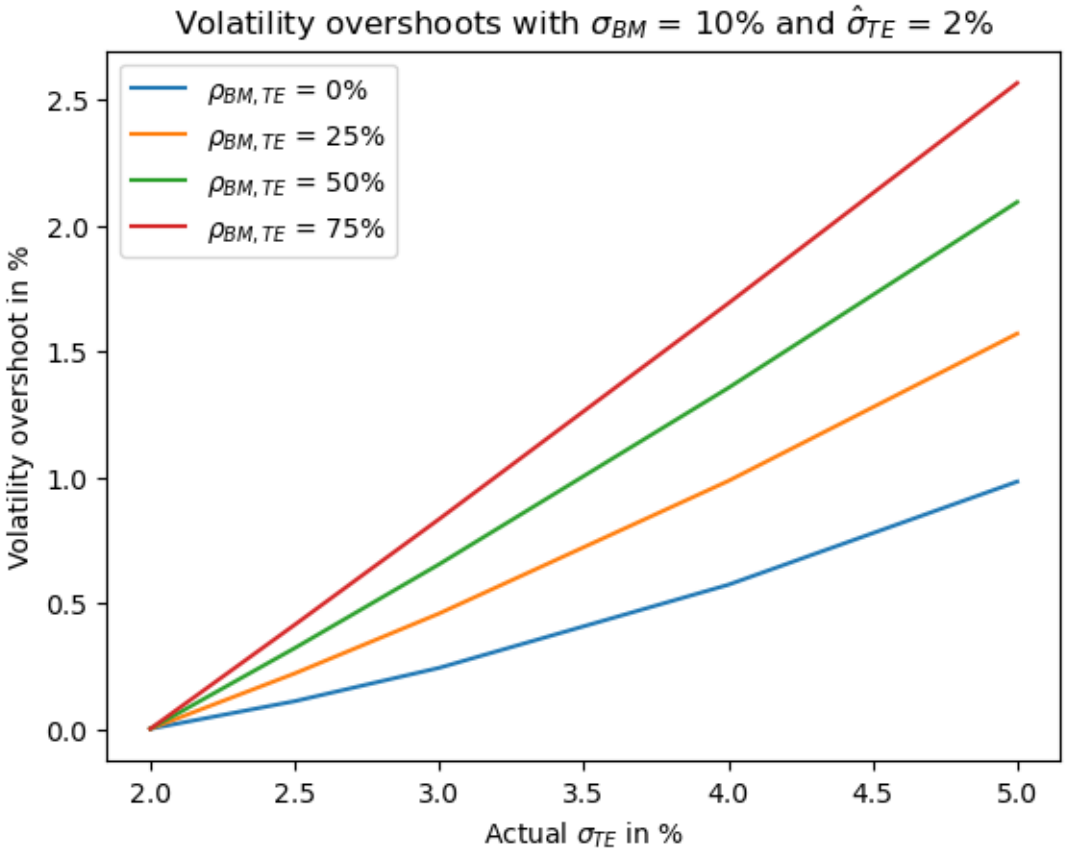

Figure 6.3.1: Risk overshoot for $\sigma_{BM} = 10\%$, $\hat{\sigma}_{TE} = 2\%$, and various $\rho_{BM,TE}\sigma_{BM}$.

There is a slightly more subtle second-order effect from the fact that you are probably also more likely to underestimate the benchmark risk σ_{BM} simultaneously with underestimating the tracking error risk σ_{TE} if the two portfolios have a high correlation $\rho_{BM,TE}$. Figure 6.3.2 below repeats the analysis from Figure 6.3.1 using Entropy Pooling to stress-test the tracking error σ_{TE} for various correlation levels. We clearly see that the risk overshoot becomes significantly higher not only due to the ceteris paribus effect from a higher tracking error than we have estimated, but also due to a higher benchmark risk. Hence, maintaining a low correlation $\rho_{BM,TE}$ between the benchmark and tracking error portfolios is a very important part of skillful portfolio construction in practice.

The practical implication of 2. is that there is a trade-off between standalone tracking error risk σ_{TE} and the correlation $\rho_{BM,TE}$, i.e., a lower correlation $\rho_{BM,TE}$ allows for a higher standalone tracking error σ_{TE} without increasing the overall portfolios risk σ_{PF}. Hence, diversification can happen within and between the benchmark and tracking error portfolios, with $\rho_{BM,TE}$ being a proxy for the degree of diversification between the two portfolios. Note also that the trade-off between σ_{TE} and $\rho_{BM,TE}$ depends on the ratio $\frac{\sigma_{TE}}{\sigma_{BM}}$. This observation explains why the impact of FX hedging on overall portfolio risk is more significant for a low risk portfolio of short-term investment grade bonds than a high risk portfolio of equities.

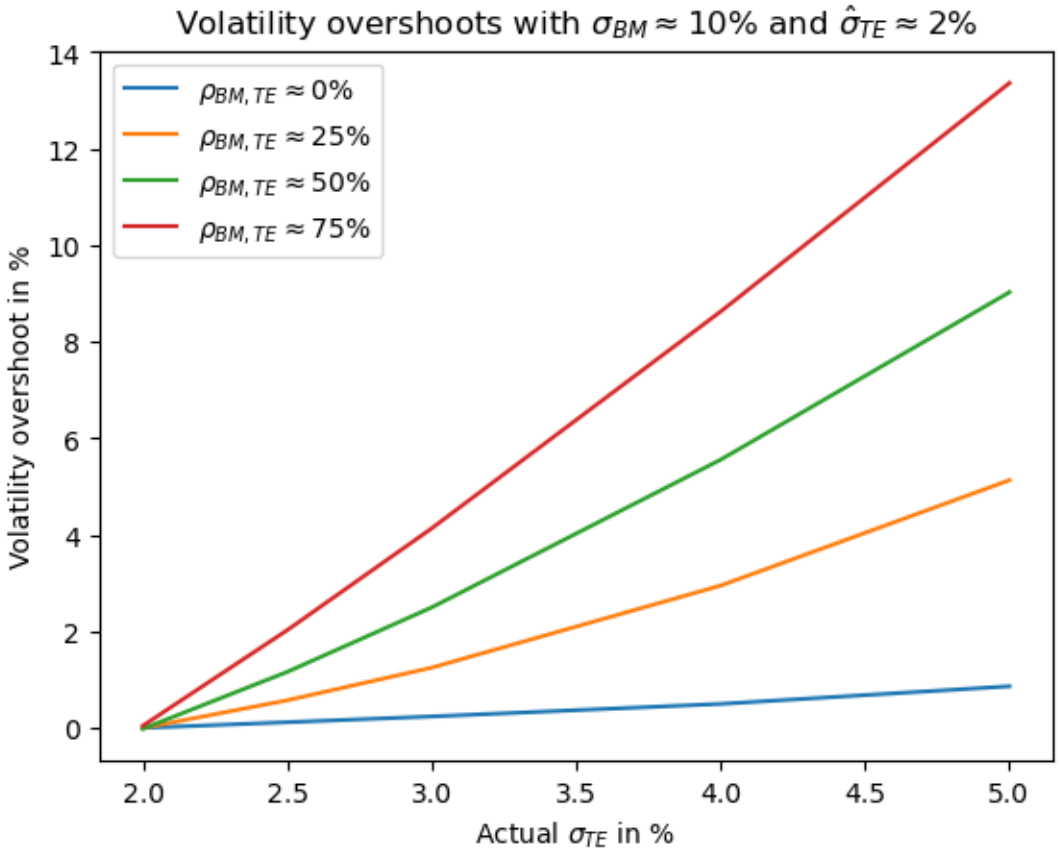

Figure 6.3.2: Risk overshoot with EP for $\sigma_{BM} \approx 10\%$, $\hat{\sigma}_{TE} \approx 2\%$, and various $\rho_{BM,TE}\sigma_{BM}$.

To understand the FX hedging argument, let σ_{BM} represent the risk of US equities or investment grade bonds. Let us assume that $\sigma_{BM} = 20\%$ for equities and $\sigma_{BM} = 5\%$ for bonds. As a foreign EUR investor, you have the opportunity to hedge the USD risk of these bonds or not. In that case, the tracking error portfolio consists of EURUSD. Let us assume that its risk is $\sigma_{TE} = 10\%$, which is the same no matter if we invest in equities or bonds. From the relation $\sigma_{TE}^2 + 2\rho_{BM,TE}\sigma_{BM}\sigma_{TE} \leq 0 \Leftrightarrow \frac{\sigma_{TE}}{\sigma_{BM}} \leq -2\rho_{BM,TE}$, we can see that for bonds we must require that $\rho_{BM,TE} = -1$ if the USD risk is not to increase the risk of the portfolio, while we only require $\rho_{BM,TE} = -0.25$ for equities. It is quite obvious that we are more likely to have a correlation of $\rho_{BM,TE} = -0.25$ or lower than $\rho_{BM,TE} = -1$. Hence, FX affects the risk of low risk bond portfolios more than higher risk equity portfolios.

CVaR is a more complex measure of risk that in the general case does not follow nice relationships like (6.3.2). For CVaR, the best we can do to replicate the form of (6.3.2) is

$$CVaR\left(PF\right) = CVaR\left(BM\right) + CVaR\left(TE\right) + f\left(BM, TE\right), \tag{6.3.3}$$

where $f\left(BM, TE\right) \leq 0$ is the diversification term. It follows that $f\left(BM, TE\right) \leq 0$ from the subadditivity property of CVaR, i.e., $CVaR\left(X + Y\right) \leq CVaR\left(X\right) + CVaR\left(Y\right)$, see Artzner et al. (1999) and Rockafellar, Uryasev, and Zabarankin (2006).

Note that the analysis of the volatility tracking error σ_{TE} and the correlation between the tracking error portfolio and benchmark $\rho_{BM,TE}$ can be generalized to the CVaR risk measure. With the portfolio and benchmark CVaR risks $CVaR(PF)$ and $CVaR(BM)$ fixed, there is again a trade-off between the diversification between the benchmark and tracking error portfolio $f(BM, TE)$ and the standalone tracking error portfolio risk $CVaR(TE)$. In summary, a better diversification represented by a lower $f(BM, TE)$ allows for a higher standalone risk $CVaR(TE)$ for the tracking error portfolio. For CVaR, the two terms are just more abstract and flexible than standard deviations and correlations.

Although it is only variance that we can decompose nicely into σ_{BM}, σ_{TE}, and $\rho_{BM,TE}$, the general principles are likely to hold for more complex risk measures such as CVaR, because the decomposition (6.3.3) will hold for any coherent risk measure $\mathcal{R}$. Hence, we must not be limited by the convenience of variance, because we pay a high price in terms of building our portfolios based on highly unrealistic market assumptions in that case.

Let us generally define portfolio optimization problems with risk targets and tracking error / tactical / short-term risk budgets as

$$e^{\star} = \underset{e}{\operatorname{argmax}}\ \mu^T e$$

subject to

$$\begin{aligned} \mathcal{R}(e) &\leq \mathcal{R}_{target}, \\ \mathcal{R}(e - e_{BM}) &\leq \mathcal{R}_{TE}, \\ v^T e &= 1, \\ e &\in \mathcal{E}. \end{aligned}$$

For some investment risk measure $\mathcal{R}$, e.g., CVaR or variance. As we have seen in the analysis above, this formulation introduces an implicit constraint on the diversification term $f(BM, TE)$.

The main takeaway from this section is that the two risk constraints introduce important trade-offs between the benchmark risk and the tracking error risk. It is important that we constantly take these dependencies into account and avoid potentially significantly underestimating the risk in our portfolio as shown in Figure 6.3.2. While it will be shown in Section 6.3.1 below how you can solve the risk budget optimization problem for variance, the same analysis as well as the subsequent Entropy Pooling stress-testing of the tracking error risk can be performed for CVaR. Readers are encouraged to test their understanding using the CVaR formulation from Section 6.2.1 and the CVaR Entropy Pooling views from Section 5.1.3 to perform this analysis.

A final comment is that we can of course also underestimate the risk of the benchmark. In the typical case where the benchmark represents broad "market beta" exposure, a high correlation between the benchmark and tracking error portfolios implies that the tracking error portfolio is simply more "market beta". It is almost equivalent to increasing the exposure of the futures position introduced in the first paragraph of this section. If the tracking error portfolio is simply characterized by systematically having more benchmark exposure, it should arguably become a part of the benchmark.

6.3.1 Validating portfolio optimization solvers

Having established that it is the problem with an overall risk target $\mathcal{R}_{target}$ and a risk budget for short-term deviations from a benchmark portfolio $\mathcal{R}_{TE}$ that is interesting from a portfolio construction perspective, we want to be able to validate that we are solving these problems correctly. This section and the accompanying code will give you the general principles for how you validate that and show you how to solve the problem for the variance risk measure using second-order cone programming, which is fairly straightforward compared to CVaR optimization.

Let us imagine that we have a solver that can solve the portfolio optimization problem for a return target μ_{target}. Perhaps it is a straightforward implementation like quadratic programming for mean-variance optimization, which we have potentially also validated against other implementations and believe is correct with a high probability. For this particular section and its accompanying code, we will use the mean-variance implementation from the fortitudo.tech package to compute the optimal long-only portfolio for a $\mu_{target} = 5\%$ return target and 25% individual upper bounds for the multi-asset instruments that follow with the package.

So how do we use the optimal exposure results $e^\star$ of the target return optimization to validate a target risk optimization? We simply compute the portfolio volatility $\sigma^\star$ for the optimal portfolio exposures $e^\star$ and use that as a target for the problem formulation that maximizes the expected return. These are equivalent problem formulations, so we can compare their results for the optimal exposures $e^\star$. Furthermore, we can compute the optimal expected return $\mu^\star$ and validate that it is indeed equal to $\mu_{target} = 5\%$. Hence, the validation of the target risk solver is straightforward to understand and hopefully implement.

Validating solvers that have both a risk target $\mathcal{R}_{target}$ and a tracking error target $\mathcal{R}_{TE}$ is a bit more challenging, while still conceptually fairly easy to understand. First, we must fix some benchmark exposure e_{BM} and introduce auxiliary variables e_{TE} through the constraint $e_{TE} = e - e_{BM}$. We can then implement risk target constraints directly on e_{TE} in the same way that we implement risk constraints on the overall portfolio risk. To validate that the dual risk solver gives us the correct results, we can compute the tracking error risk $\sigma^\star_{TE} = \mathcal{R}\left(e^\star - e_{BM}\right)$ using the optimal exposures $e^\star$ and use this as the target tracking error risk $\mathcal{R}_{TE}$ in our dual risk optimization.

The requirement that you have access to a portfolio optimization solver with a target return is in practice not that strict. This is the usual formulation, so for most risk measures you should be able to find an implementation that you can compare your own against, or simply be very careful with your own implementation and validate it in other ways. For the CVaR risk measure, you can use the fortitudo.tech package for target return and efficient frontier optimizations.

The accompanying code to this section illustrates the above procedure for variance using a second-order cone solver. You are encouraged to repeat the exercise for fully general Monte Carlo distributions and CVaR optimization after finalizing the CVaR optimization exercise from Section 6.2.1. Table 6.1 shows the results of the exercise for the variance risk measure. You can use the structure of the accompanying code to repeat this for CVaR, starting with a 5% return target. Make sure that you validate the CVaR solvers both for uniform and fully general scenario probabilities. The results for both cases are given in the accompanying code to Section 6.2.1. Note that this is still a very basic use case, while solving the problem with advanced constraints and transaction costs is more challenging.

	Target return	Target risk	Target risk and tracking error	Benchmark
Gov & MBS	0.00	0.00	0.00	10.00
Corp IG	0.00	0.00	0.00	10.00
Corp HY	0.00	0.00	0.00	10.00
EM Debt	18.39	18.39	18.39	10.00
DM Equity	0.00	0.00	0.00	10.00
EM Equity	0.00	0.00	0.00	10.00
Private Equity	6.61	6.61	6.61	10.00
Infrastructure	25.00	25.00	25.00	10.00
Real Estate	25.00	25.00	25.00	10.00
Hedge Funds	25.00	25.00	25.00	10.00
Return target	**5.00**			
Volatility		**6.69**	**6.69**	
Tracking error			**3.18**	

Table 6.1: Mean-variance optimized portfolios using different objectives.

The numbers in bold indicate which constraints were used to optimize the portfolio. Hence, we started by a target return constraint, then a target risk constraint, and finally a target risk and tracking error constraint. You can find all the details in the accompanying code to this section.

Note that in Table 6.1 above, we optimized a portfolio with a return target first and then took the risk and tracking error targets as given. In practice, we would usually specify risk targets and tracking errors and then take the optimal expected return as given. Tracking error constraints can in practice contribute to alleviating the inherent portfolio optimization issues related to concentrated portfolios, especially when combined with transactions costs, but they do not solve all the issues. Therefore, portfolio optimization in practice must almost always take parameter uncertainty into account, which we do in the next section.

6.4 Parameter uncertainty and Exposure Stacking

This section introduces resampled parameter uncertainty into the portfolio optimization problem using the perspectives from Kristensen and Vorobets (2024). While the fundament is the same including a recap of the Exposure Stacking method, this section introduces new ways of using the Exposure Stacking objective and includes a case study that shows you how you can use the method to combine CVaR optimizations with parameter uncertainty and different α levels in Section 6.4.3 below.

As we have seen in the previous sections, portfolio optimization problems are highly sensitive to parameter estimates or more generally the market model input. Some practitioners are so skeptical that they do not use portfolio optimization methods at all, at least explicitly. Most people still attempt to build mean-risk optimal portfolios, albeit in implicit or heuristic ways.

Resampled portfolio optimization is a heuristic first introduced by Michaud and Michaud (1998), where B market model or parameter samples are generated. For each of these samples, we optimize portfolios with the same constraints and similar risk levels to compute a sample of optimal exposures $e_b^\star \in \mathbb{R}^I$, $b \in \{1, 2, \ldots, B\}$. Afterwards, we compute an average of the optimal exposures to get the

final resampled portfolio

$$e_w^\star = \sum_{b=1}^{B} w_b e_b^\star, \tag{6.4.1}$$

with $w_b \geq 0$ and $\sum_{b=1}^{B} w_b = 1$.

The original suggestion by Michaud and Michaud (1998) is to set $w_b = \frac{1}{B}$ and align the portfolio risk through the portfolio's index on the efficient frontier. A desirable aspect of the resampled approach is that it is highly flexible, while it initially did not have much justification. Fundamental perspectives justifying the resampled approach are presented by Kristensen and Vorobets (2024), while also introducing other ways of aligning the portfolio risk with a new method for determining the sample weights w_b.

To understand the new stacking methods, it is important to view the portfolio optimization problem from the right perspective. We consider a risk-adjusted return objective f, which is a function of the market model (R, p) with $R \in \mathbb{R}^{S \times I}$ being a matrix of joint market scenarios and $p \in \mathbb{R}^S$ being an associated scenario probability vector from (1.1.1). Additionally, f is a function of the optimization constraints $\mathcal{E}$ and the portfolio exposures $e \in \mathbb{R}^I$. First, consider the case where f determines the expected return, while constraints on the risk are included in $\mathcal{E}$. The optimal portfolio exposures $e^\star$ are then given by

$$e^\star = \underset{e}{\operatorname{argmax}}\ f\left(R, p, \mathcal{E}, e\right).$$

It is also possible to work from the perspective of minimizing risk with a return target. In such cases, the optimal portfolio exposures are given by

$$e^\star = \underset{e}{\operatorname{argmin}}\ g\left(R, p, \mathcal{E}, e\right),$$

where g is a function that determines the risk, while the return target is included in the constraints $\mathcal{E}$. As we noted in Section 6.3.1 above, the two perspectives solve equivalent risk-adjusted return problems.

Kristensen and Vorobets (2024) argue that the parameter uncertainty issue can be analyzed from the same perspective as statistical learning models using the bias-variance trade-off. Specifically, treating the market model or parameters as data and the optimal portfolio exposures as estimates, with the particular mean-risk optimization method being an estimator. From this perspective, the traditional mean-risk portfolio optimization estimators have no bias but a high variance due to their sensitivity to the market model and parameter values. On the other hand, resampled versions of these estimators have some bias but a lower variance.

To explore resampled portfolio optimization from a bias-variance perspective, we define the optimal exposure for sample $b \in \{1, 2, \ldots, B\}$ by

$$e_b^\star = \underset{e}{\operatorname{argmax}}\ f\left(R_b, p_b, \mathcal{E}, e\right) = \underset{e}{\operatorname{argmin}}\ g\left(R_b, p_b, \mathcal{E}, e\right).$$

Note that in this definition, we are optimizing over samples (R_b, p_b) of the market model. For the mean-variance objective, this is simply estimates of mean vectors μ_b and covariance matrices Σ_b. For the mean-CVaR objective, this can be entirely new joint market scenarios R_b and associated probability vectors p_b, with p_b possibly generated using sequential Entropy Pooling as described in Chapter 5. In

their case study, Kristensen and Vorobets (2024) show that it is usually the mean vectors that affect the efficient exposures $e_b^\star$ the most.

How do we analyze the risk-adjusted objective from a bias-variance perspective? We must first understand in which sense the traditional portfolio optimization methods such as mean-variance and mean-CVaR are unbiased. We call a mean-risk estimator unbiased if it correctly estimates $e^\star$, i.e., if it correctly estimates the optimal portfolio exposures given the market model (R, p). Clearly, the resampled estimator (6.4.1) is not unbiased in the general case.

Hence, when we perform resampled optimization, we are purposefully introducing bias to reduce the variance of the final optimal exposures estimate. If we weigh the sample estimates equally as originally suggested, we are just hoping that the increase in bias will reduce the variance more. Although this is also what people experience in practice, we do not have any guarantees that the bias-variance trade-off is minimized by an equally weighted average, so it is worth exploring whether we can do better.

Direct minimization of the vector mean squared error

$$MSE\left(e_w^\star\right) = \mathbb{E}\left[\left\|\sum_{b=1}^{B} w_b e_b^\star - e^\star\right\|_2^2\right]$$

of the resampled estimator (6.4.1) is not interesting given that the solution will simply try to find a linear combination of the vectors $e_b^\star$ which is as close as possible to $e^\star$ with respect to the Euclidean norm $\|\cdot\|_2$. Instead, we focus on multivariate stacking objectives having a form similar to the vector mean squared error.

Stacked generalization, see Wolpert (1992) and Breiman (1996), has received increased attention in recent years due to its versatility and potential for improving out-of-sample performance. Hence, we analyze the resampled portfolio optimization problem from this perspective. A natural, albeit slightly naive, starting point for the objective function is to ensure that the Euclidean norm of the difference between $e_w^\star$ and each $e_b^\star$ is, on average, as small as possible, i.e., solve the problem

$$w^\star = \underset{w}{\operatorname{argmin}} \ \frac{1}{B}\sum_{k=1}^{B}\left\|e_k^\star - \sum_{b=1}^{B} w_b e_b^\star\right\|_2^2. \tag{6.4.2}$$

However, as showm in Appendix A of Kristensen and Vorobets (2024), the optimal solution to (6.4.2) always yields a vector of equal sample weights $w_b^\star = \frac{1}{B}$ for all $b \in \{1, 2, \ldots, B\}$, corresponding to the traditional resampling method.

Instead of (6.4.2), Kristensen and Vorobets (2024) define an objective function based on the ideas of stacked regression and cross-validation. Let $\mathcal{B} = \{1, 2, \ldots, B\}$ be the set of sample indices and suppose that we partition $\mathcal{B}$ into L nonempty sets $\mathcal{K}_1, \mathcal{K}_2, \ldots, \mathcal{K}_L$ for some $L \in \{2, 3, \ldots, B\}$. For any choice of $l \in \{1, 2, \ldots, L\}$, we consider the set of exposure vectors $\{e_k \mid k \in \mathcal{K}_l\}$ as a validation set, find sample weights w_b for the remaining $B - |\mathcal{K}_l|$ exposure vectors, and calculate the average difference ε_l between the weighted exposure and the validation set exposures, i.e., we compute

$$\varepsilon_l = \frac{1}{|\mathcal{K}_l|}\sum_{k\in\mathcal{K}_l}\left\|e_k^\star - \sum_{b\notin\mathcal{K}_l} w_b e_b^\star\right\|_2^2.$$

If we repeat the analysis using each of the sets $\mathcal{K}_1, \mathcal{K}_2, \ldots, \mathcal{K}_L$ as our validation set indices, we can compute the L-fold cross-validation estimate as the average $\varepsilon = \frac{1}{L}\sum_{l=1}^{L} \varepsilon_l$, see James et al. (2023). Thus, instead of using the naive objective function (6.4.2), we wish to find the vector of sample weights $w^\star$ that minimizes the cross-validation estimate ε, i.e., solve the problem

$$w^\star = \underset{w}{\operatorname{argmin}} \frac{1}{L}\sum_{l=1}^{L}\left(\frac{1}{|\mathcal{K}_l|}\sum_{k\in\mathcal{K}_l}\left\| e_k^\star - \sum_{b\notin\mathcal{K}_l} w_b e_b^\star \right\|_2^2\right). \tag{6.4.3}$$

Kristensen and Vorobets (2024) show in Appendix B that (6.4.3) can be formulated as a quadratic programming problem and therefore solved in a fast and stable way. Note that (6.4.3) allows us to stack exposures both across different market model samples (R_b, p_b) to incorporate parameter uncertainty as well as different efficient portfolio estimators. For example, one could combine mean-variance, mean-CVaR, or even mean-CVaR with different CVaR levels such as 90%, 95%, and 97.5%, see the case study in Section 6.4.3.

The original suggestion to stack based on portfolio exposures $e_b^\star$ is based on the risk alignment considerations mentioned in Section 2.1 of Kristensen and Vorobets (2024). The case study in the accompanying code to this section confirms that the portfolios stacked on the portfolio mean or volatility indeed drift quite significantly in relation to risk and return compared to the original resampled frontier. Table 6.2 shows the results for a repeated analysis from Kristensen and Vorobets (2024) with a different seed and including portfolio mean and volatility stacked resampled portfolios.

	Resampled	2-Fold Exposure Stacking	Mean stacking	Vol stacking	Frontier
Gov & MBS	0.12	0.00	0.00	0.00	0.00
Corp IG	0.00	0.00	0.00	0.00	0.00
Corp HY	0.00	0.00	0.00	0.00	0.00
EM Debt	4.88	0.00	4.38	16.70	0.00
DM Equity	0.48	0.00	0.00	0.00	0.00
EM Equity	0.07	0.00	0.00	0.00	0.00
Private Equity	18.07	13.92	30.20	34.12	20.41
Infrastructure	34.35	46.49	38.50	18.48	40.49
Real Estate	16.36	10.49	5.48	7.05	11.06
Hedge Funds	25.66	29.11	21.44	23.64	28.04
Mean	6.34	6.25	7.43	7.29	6.71
Vol	8.96	8.78	11.09	11.27	9.61

Table 6.2: Mean-variance optimal exposures as well as expected return and volatility.

6.4.1 Return and Risk Stacking

6.4.2 Derivatives and risk factor parameter uncertainty

6.4.3 Multiple CVaR levels case study

6.5 Portfolio rebalancing

Chapter 7

Risk and return analysis

7.1 Marginal risk and return contributions

7.2 Tail risk hedging

Chapter 8

Summary and comparison with the old approach

Bibliography

Artzner, P. et al. (1999). "Coherent Measures of Risk". *Mathematical Finance* 9.3, pp. 203–228.

Boyd, Stephen and Lieven Vandenberghe (2004). *Convex Optimization*. Cambridge University Press.

Breiman, Leo (1996). "Stacked Regressions". *Machine Learning* 24, pp. 49–64.

Caticha, A. and A. Giffin (2006). "Updating Probabilities". *Bayesian Inference and Maximum Entropy Methods in Science and Engineering Conference*.

Corani, G. and C. de Campos (2015). "A Maximum Entropy Approach to Learn Bayesian Networks from Incomplete Data". *Interdisciplinary Bayesian Statistics. Springer Proceedings in Mathematics & Statistics*. URL: https://doi.org/10.1007/978-3-319-12454-4_6.

Friedman, D. et al. (2014). *Risky Curves: On the Empirical Failure of Expected Utility*.

James, Gareth et al. (2023). *An Introdution to Statistical Learning with Applications in Python*. Springer.

Kristensen, L. and A. Vorobets (2024). "Portfolio Optimization and Parameter Uncertainty". *SSRN*. URL: https://ssrn.com/abstract=4709317.

Krokhmal, P., J. Palmquist, and S. Uryasev (2002). "Portfolio Optimization with Conditional Value-at-Risk Objective and Constraints". *Journal of Risk* 4, pp. 43–68.

Markowitz, H. (1952). "Portfolio Selection". *The Journal of Finance* 7.1, pp. 77–91.

Meucci, A. (2008a). "Fully Flexible Views: Theory and Practice". *Risk* 21.10, pp. 97–102. URL: https://ssrn.com/abstract=1213325.

Meucci, A. (2008b). "The Black-Litterman Approach: Original Model and Extensions". URL: https://ssrn.com/abstract=1117574.

Meucci, A. (2012a). "Effective Number of Scenarios in Fully Flexible Probabilities". *GARP Risk Professional, February 2011*, pp. 32–35. URL: https://ssrn.com/abstract=1971808.

Meucci, A. (2012b). "Stress-Testing with Fully Flexible Causal Inputs". *Risk*. URL: https://ssrn.com/abstract=1721302.

Meucci, A., D. Ardia, and S. Keel (2011). "Fully Flexible Extreme Views". *Journal of Risk* 14.2, pp. 39–49. URL: https://ssrn.com/abstract=1542083.

Michaud, R. and R. Michaud (1998). *Efficient Asset Management: A practical Guide to Stock Portfolio Optimization and Asset Allocation*. Oxford University Press.

Norris, J. R. (1998). *Markov Chains*. Cambridge University Press.

Rebonato, R. and A. Denev (2011). "Coherent Asset Allocation and Diversification in the Presence of Stress Events". *Journal of Investment Management*. URL: https://ssrn.com/abstract=1824207.

Rebonato, R. and A. Denev (2014). *Portfolio Management under Stress: A Bayesian-Net Approach to Coherent Asset Allocation*. Cambridge University Press.

Rockafellar, R. T. and S. Uryasev (2000). "Optimization of Conditional Value-at-Risk". *Journal of risk* 2, pp. 21–42.

Rockafellar, R. T., S. Uryasev, and M. Zabarankin (2006). "Generalized Deviations in Risk Analysis". *Finance and Stochastics* 10.1, pp. 51–74.

Vorobets, A. (2021). "Sequential Entropy Pooling Heuristics". *SSRN*. URL: https://ssrn.com/abstract=3936392.

Vorobets, A. (2022a). "Portfolio Management Framework for Derivative Instruments". *SSRN*. URL: https://ssrn.com/abstract=4217884.

Vorobets, A. (2022b). "Variance for Intuition, Cvar for Optimization". *SSRN*. URL: https://ssrn.com/abstract=4034316.

Vorobets, A. (2023). "Causal and Predictive Market Views and Stress-Testing". *SSRN*. URL: https://ssrn.com/abstract=4444291.

Vorobets, A. (2024). "Derivatives Portfolio Optimization and Parameter Uncertainty". *SSRN*. URL: https://ssrn.com/abstract=4825945.

Wolpert, David H. (1992). "Stacked Generalization". *Neural Networks* 5, pp. 241–259.