Management for Professionals

Martin Auer

Hands-On Value-at-Risk and Expected Shortfall

A Practical Primer

Springer

Martin Auer

Hands-On Value-at-Risk and Expected Shortfall

A Practical Primer

Springer

Martin Auer
Raiffeisen Bank International
Vienna, Austria

ISSN 2192-8096 ISSN 2192-810X (electronic)
Management for Professionals
ISBN 978-3-319-72319-8 ISBN 978-3-319-72320-4 (eBook)
https://doi.org/10.1007/978-3-319-72320-4

Library of Congress Control Number: 2017961124

Cover illustration: eStudio Calamar and Ivonne Barrera Villanueva, Berlin/Figueres

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Endorsements

A very useful guide to the theoretical and practical aspects of implementing and operating a risk-monitoring system for a mid-size financial institution. It sets a common body of knowledge to facilitate communication between risk managers, computer and investment specialists by bridging their diverse backgrounds.

Giovanni Barone-Adesi
Professor, Universitá della Svizzera italiana
Lugano, Switzerland

This unassuming and insightful book starts from the basics and plainly brings the reader up to speed on both theory and implementation.

Shane Hegarty
Director Trade Floor Risk Management
Scotiabank

Visit the book's website at www.value-at-risk.com.

For
Nahuel,
Mirjam,
Yuna,
Yannik,
and Julian

Preface

This book sets out to describe the simplest market risk model that is still practical. It outlines the model's underlying math, daily operation, and implementation, while stripping away as much statistical overhead as deemed fit. It does not advance some novel methods but attempts to pick and present those modeling approaches and methods that might conceivably do the job. We set up and operate a highly similar model in a mid-sized Austrian bank; it performed very well overall, right through some turbulent years in financial markets. Of the calls we made, we got a few right. Some we didn't at first, but we fixed them and hopefully learned from them. They all came to shape this account.

I wrote this book for the fun of it. But why should you read it? If you are a recent graduate on your first day in office or a time-starved manager ready to brush up on your market risk fundamentals, you hopefully get an easy-to-digest introduction to basic risk measures and their properties. If you are a programmer, you might learn about the mathematical underpinnings of your code, making your exchanges with the risk department just a bit smoother. If you are a quant, maybe you can gauge the types of support statistics most useful to daily risk operation. If you are a team leader allocating money and time, you possibly find ways to steer through the technical jargon and rein in the understandable compulsion of your team to use the latest tech and the fanciest math. If you operate the model on a daily basis, some of the analytical support methods given here might help you understand, explain, and defend the numbers. If you're anyone, really, in this motley crew, I hope this book will ease your communication with colleagues, clients, and controllers.

After my studies, I quickly figured out that college is far more fun than work, so I applied to several programs at New York universities. By chance, I got admitted to a financial mathematics one, where in my very first lecture I discovered this "bond" thing, an alien concept in my cash-strapped student life. That other people may likewise get exposed to market risk topics after varying journeys set up this book's angle. This book at times will state the obvious and thus often unmentioned—even apparently trivial calls are easier to make if their trail is known to be trodden. It will appear sloppy at times, as it proposes heuristics rooted in the nature of imperfect real-world data. While it tries to offer a consistent notation, it will gladly gloss over many technical details. Several opinions in it you will find judgmental; indices missing in a sacrifice to readability; and shortcuts taken to save on paper and rumination. (And those are just the infractions I committed on purpose.) The views given here are mine and do not represent those of my current or former employers.

I meant this book, in the end, to be about simplicity and about communication between team members of wildly differing backgrounds. I hope reading it is worth a bit of your while.

Vienna, Austria
2018

Martin Auer

Acknowledgments

I had the honor of being part of Raiffeisen Bank International's market risk team, and for this I am very grateful—thank you, Alexander Tsorlinis and Hannes Mösenbacher and Michael Glaser.

A big thanks to the market risk team: Brigitte Ackermann, Harald Dür, Maria Fizek, Edwin Glassner, Karl "Charly" Hackl, Marin Karaga, Sara Lambing, Stefan Mischer, Olga Muster, Emil Persson, Peter Prochazka, Jure Radmann, and Johannes Rhomberg—for your dedication and counsel, your can-do attitude, and your patient endurance of me cursing at my screen.

It was a true privilege to encounter and work with many others throughout the bank and over the years. I'd especially like to thank Nikolas Besemer, Boris Breidenbach, Ernst Galavics, Harald Gaudera, Markus Grabner, Elisabeth Gruber, Roland Hafner, Helmut Hauser, Martin Hödl, Olivera Janic, Gerhard Keplinger, Günther Kickinger, Stephan Kossmeier, Beatrix Lehner, Christian Mörtl, Sandra Ober, Sebastian Pramreiter, Martin Predota, Rainer Pullirsch, Matthias Rabl, Wolfgang Sakuler, Stefan Schäfer, Gerhard Schauer, Stefan Sigl-Worel, Maria Siopacha, the missed Alexandru Tanase, Josef Teichmann, Peter Trappl, Gerhard Unger, Alison Urbas, Julian Wergieluk, and Joachim Wieser. From an earlier stint I still owe thanks to Gernot Gebauer, Andreas Halwein, and Hannes Kavelar.

For invaluable help in revising various drafts of this text, I am especially grateful to Andrea Chang, Daniela Escobar, Edwin Glassner, Martin Hödl, Thomas Tschurtschenthaler, Jan Vecer, and Jean Wang; for her valiant fight against the semicolon, to Jennifer Granić. Thank you all!

Many thanks for their great guidance and support to the whole team at Springer: Silvia Schilgerius, Rocio Torregrosa, Abinay Subramaniam, and Uma Periasamy.

Thanks to Petra, for watching GNTM with me; thanks to Commodore for making the Amiga, the best computer there ever was; and thanks to Maria and Anton for buying me one.

Contents

List of Figures

1 Introduction

Financial markets let people trade promises of future payments. These payment promises are called financial *assets*. Prominent examples are bonds or company stocks. A bond is a way to borrow money, and it promises its buyer a future debt repayment with interest. Stock is used to raise capital and promises its holder future dividend payments. In addition to those, many other types of assets exist, but at their core they all are tradable contractual claims on future cash flows. Supply and demand determine the prices at which to buy or sell them—the prices at which to *enter positions*.

Now, asset prices and thus the values of the positions in them change over time. This can have fundamental causes, e.g., a company discovering a new drug or a country recovering from a recession, but it can also be due to the market activity itself, as witnessed in the hefty fluctuations of stock prices even on slow-news days. These latter, seemingly random market movements in asset prices constitute *market risk* to positions.

One way to assess this kind of risk is to determine the potential impact of specific market movements on positions' values. Such what-if inquiries often mimic or simulate market shocks and are then called *stress tests*. If the shocks are tiny and standardized, the results are called *sensitivities*; they serve to track and compare the positions' asset exposures.

In the same vein, you can try to estimate the plausibility and impact of future asset price changes from historically observed ones and condense the effect on the positions into some aggregate measure of risk. A prominent measure is the so-called *value-at-risk* or *VaR*, which is a hypothetical daily loss expected to be breached once every hundred days—in other words: *the probability that tomorrow there will be a loss larger than the VaR is* 1%. A sibling of the VaR is the *expected shortfall* or *ES*, which yields the estimated average loss over several worst-case scenarios.

These and similar measures are used to monitor the positions' risk profile, to signal critical market conditions, to limit exposures, and to otherwise meet requirements as prescribed by law and banking regulation authorities. They may

M. Auer, *Hands-On Value-at-Risk and Expected Shortfall*, Management for Professionals, https://doi.org/10.1007/978-3-319-72320-4_1

also determine a bank's crucial *capital requirements*, the amount of capital a bank has to hold given its exposure to the markets, and thus the cost of doing business. The risk numbers and their workings therefore matter to a large audience besides the risk manager and his IT guy: to the trader, the accountant, the compliance officer, the board member, and the regulator.

With the stakeholders many and varied, a risk model should not merely accurately capture risks but also be transparent and easy to explain, avoid recurring modifications and the resulting scrutiny, and operate reliably under real-world duress like imperfect data. We now set out to describe a system designed to achieve these goals.

Part I of this book outlines the basic risk measures and their relations, and it proposes a simple VaR approach (a *filtered historical* one).

Part II describes how to apply the risk measures to common questions about a risk profile's bearings, and it details risk measure properties, time series behavior, and model sanity-checks.

Part III illustrates a possible overall design of a risk system and presents ways to implement this system into software.

Finally, the appendix collects some mathematical foundations, a brief digression on risk-neutral pricing, and links to further reading material.

Before all that, however, the next chapter aims to give an intuitive introduction to how risk can be thought of and compressed into one single number.

2 Motivation

For some intuition on market risk, let's first take a look at a simple asset position. Assume throughout that our domestic currency is the dollar.

Consider a position of 10,000 units of a stock S whose price is quoted in euro. We are interested in the dollar value of our position, which is affected by two asset prices: the stock price $S_€$ in euro and the euro/dollar exchange rate $€_\$$, i.e., the price of one euro in dollar terms. The current value or price p of our position is

$$p = 10{,}000\, S_€\, €_\$.$$

As stock price and exchange rate change over time, so does the value of our position—it can net a profit or suffer a loss. We are interested in how large and how likely a loss this position might experience tomorrow. We turn to history as a guide. Figure 2.1 depicts the time series of stock prices over the course of 2 years or 500 business days.

The stock price was more than 30% lower 2 years ago, but the daily price changes we are interested in were more subdued, as shown by the relative returns between consecutive days given in Fig. 2.2. These returns seem to be a useful dimension to work in. If tomorrow the stock price and the exchange rate were to change by relative returns of x and y, then the change in value of our position would be

$$\begin{aligned} \Delta p &= 10{,}000\, S_€(1+x)\, €_\$(1+y) - p \\ &= p(1+x)(1+y) - p. \end{aligned}$$

This dollar quantity's expected daily fluctuation or *volatility* is of interest to us, and it depends on the underlying assets' behavior:

- If the underlying assets fluctuate a lot, i.e., if they are volatile as manifested by large returns in both directions, then the position's value will also be more

M. Auer, *Hands-On Value-at-Risk and Expected Shortfall*, Management for Professionals, https://doi.org/10.1007/978-3-319-72320-4_2

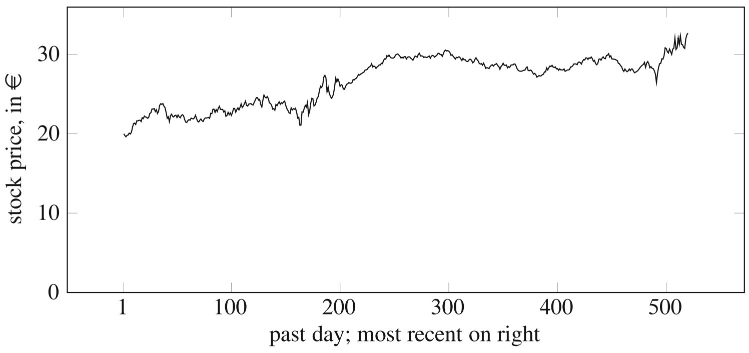

Fig. 2.1 History of stock price

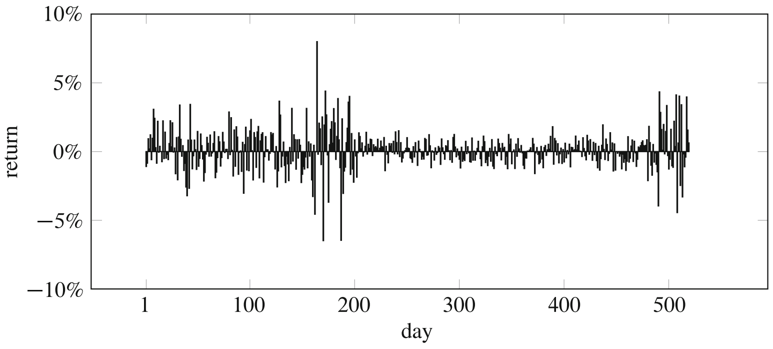

Fig. 2.2 History of stock prices changes, as relative returns

volatile. The standard statistical measures of volatility are the *variance* and its square root, the *standard deviation*.

- If the underlying assets tend to move in the same direction, the combined effect on the position will be larger, meaning higher volatility and risk; if they tend to move in opposite directions, their impact will partially cancel out and translate into lower position volatility. To measure this degree of co-movement, we use the *covariance* and its normalized offspring, the *correlation*.

A risk model must account for both effects.

We can, in a first attempt, approximate the position's price change as $\Delta p \approx px + py$ (neglecting the small term pxy). Its volatility or standard deviation can be derived from the standard deviations of x and y and from their correlation, all of which we can, in turn, estimate from historical time series. We then assume the underlying asset returns x and y to be *normally* distributed—a common, mathematically handy assumption. Conveniently, it turns out that our Δp, as a sum of normals, is then itself normally distributed, which gives us a homely handle on its risk characteristics. We will outline this *analytical* approach in Chap. 7.

Such an approach is flexible in the sense that it is easy to tweak. Consider again the time series of stock returns given above. Clearly, the recent returns on the right-hand side seem to be more volatile than most of the older, previously observed ones. This suggests that we should consider not the overall standard deviation but maybe just the one observed in the most recent past, e.g., during the last 20 days. By simply updating our volatility estimate for the stock this way, our model takes into account the recent, apparent increase in the stock's volatility and becomes more up-to-date or in tune with current market conditions.

This basic model involves few and malleable steps. Alas, it also has several drawbacks: it requires an admittedly modest amount of thought about standard deviations and correlations; it (sometimes falsely) assumes normally distributed asset returns; and it (often falsely) relies on well-behaved position pricing. (Good behavior here means, loosely, that twice-as-large a return has twice-as-large a price impact. Such *linearity* holds true in our example but does not have to in general.)

There is a straightforward way to avoid those drawbacks, at the expense of some additional computation. Instead of worrying about the statistics of the underlying assets, we ignore them and examine the whole set of hypothetical price changes under the observed historical returns x_i and y_i:

$$\{\Delta p_i = p(1 + x_i)(1 + y_i) - p\}.$$

In our example setup, 500 pairs of historical asset returns (x_i, y_i) would yield 500 position price changes Δp_i—all certainly, and equally, plausible, since each one has already been observed in the past. In fact, we can even do one better: if we add all mirrored pairs $(-x_i, -y_i)$ to our initial set, we can obtain 1000 plausible position price changes to work with—we simply hold that each uptick could have just as well been a downtick, and vice versa. Put roughly, this is possible because we do not substantially alter the core characteristics of the assets' standard deviations and correlations, and it is useful because we can operate on larger data sets with more expressive statistics.[1]

This enlarged set of 1000 Δp_i-values can now be used to derive arbitrary measures of volatility or risk: the standard deviation, the minimal value, the average, the median, etc. This is the core idea in the *historical* approach, which we will describe in Chap. 4.

Even at first glance, though, we see that the drawbacks of the analytical approach mentioned above do not apply here. First, in the historical approach we don't even bother to compute and shuffle around the assets' standard deviations and correlation. We also do not make potentially unwarranted assumptions about the underlying assets' distributions, in particular their normality. Finally, we can use the exact price of our position including, in our example, the small px_iy_i terms, so even such non-

[1]We will later encounter an alternative method to artificially increase the number of price changes to work with, also without using a larger historical observation period.

linear behavior would not be an issue. All-in-all, this is a rare specimen of a model—it combines simplicity and accuracy.

We'd almost be done, were it not for the tweak described above—the one where we update our asset volatility estimate and base it on the most recently observed returns to capture the current market mood. Not accounting for such recent market information could lead to risk estimates based on outdated information and could therefore potentially underestimate the current risk. The historical approach needs a fix for this.

Luckily, the standard workaround is as simple as it appears beastly—we just modify all historical returns: we stretch those that seem to appear in periods with a volatility smaller than the current one, and we squeeze those that reside in periods of larger volatility. We hereby obtain new *rescaled* asset returns and use those instead of the original ones to get our set of position price changes. This model variation is known as the *filtered* historical approach (we will usually omit this slightly obscure label for brevity).

Both the analytical and the historical approach thus give us means to describe the volatility of our position's value. The analytical approach expresses the gains and losses via a normal distribution with some standard deviation; the historical approach operates on a set of plausible, historically derived gains and losses. We are now ready to graph both approaches, show how they are aligned, and compress the full risk picture into one snappy measure.

Let's first depict, in the somewhat ugly Fig. 2.3, the price changes of our position obtained from the historical returns by simply plotting each change as an individual dot. We see that the price changes lie roughly between −30,000 and +30,000 dollars. Because they are many and huddled together, the dots overlap and the figure is murky. A much better view is given by a *histogram*, which counts occurrences of price changes in separate bins (see Fig. 2.4).

The analytical approach, on the other hand, uses a normal distribution with some standard deviation to express the volatility behavior of price changes. Its shape

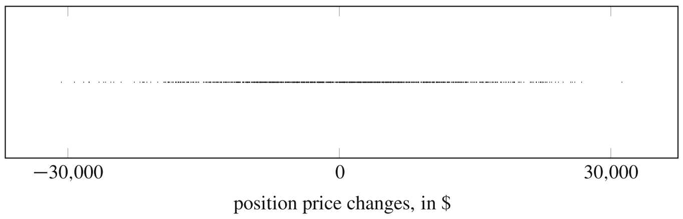

Fig. 2.3 Price changes, raw

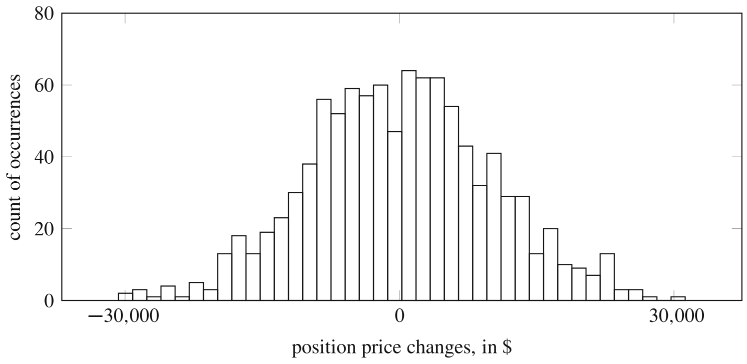

Fig. 2.4 Price changes, histogram

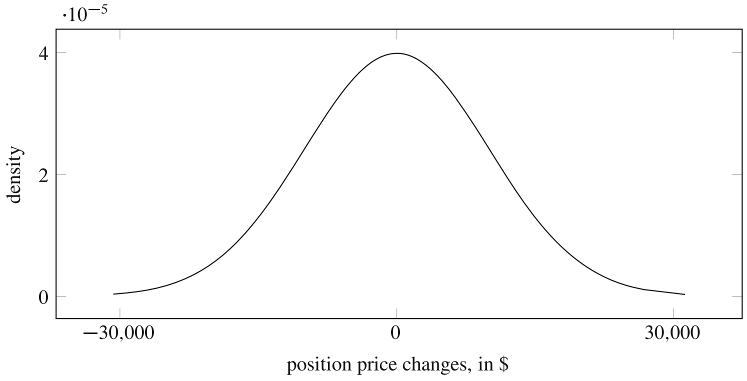

Fig. 2.5 Price changes, analytical

or *probability density function* is essentially an idealized version of the histogram above (see Fig. 2.5).[2]

Now, the graphs actually already represent the full risk profile along the two risk dimensions of probability and impact. Still, we want to consistently condense that wealth of information into a succinct, single number (just imagine having to relay the copious histogram shape over the phone!).

Why not just use the minimal, most negative price change, i.e., the most severe loss, as such a summary measure of risk? Well, this would not be elegant. Such a measure has no ready meaning in the analytical world of normal distributions because losses are unbounded there. In the historical setup, it might oscillate wildly over time, since it would be driven by a single observation, potentially an extreme

[2]Note the different y-axis scale of histogram and normal density. The latter is normalized to a total area of 1, which lets us interpret partial areas as probabilities. We could normalize the histogram, too, by dividing the bar heights by the total bar area.

outlier. Alternatively, the standard deviation is a well-defined and highly stable measure in both worlds; however, it may neglect substantial clusters of unusually large losses, which really should be accounted for.

Yet these considerations already hint at a viable, middle-ground solution. It is to split up the price changes into two sets, one containing the ten largest losses and one containing all the remaining price changes, and then to characterize the overall risk with the threshold value that separates those two sets. This is the *value-at-risk* or *VaR*. This measure is both stable over time and sensitive to clusters of large losses. We expect to see losses more severe than this VaR in 10 out of 1000 cases, or with a 1% probability.

In the historical approach, the VaR is simply the 10th most severe loss, as seen in Fig. 2.6. Hence, sorting the price changes is all that is needed to pick it out.

In the analytical approach, the price changes are represented by a normal distribution with some standard deviation. But here, too, the VaR separates the worst 1% of the outcomes from the rest, as visualized in Fig. 2.7. The VaR is again easy to compute because it turns out to be simply the normal distribution's standard deviation times a constant of −2.33. So in this analytical setup, the VaR doesn't actually give us any additional information that the standard deviation doesn't already imply. In the historical approach, however, the VaR does do that—it captures how gains and losses behave at the very fringe of plausible values.

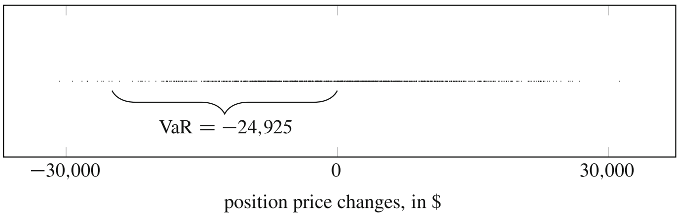

Fig. 2.6 Raw price changes and VaR

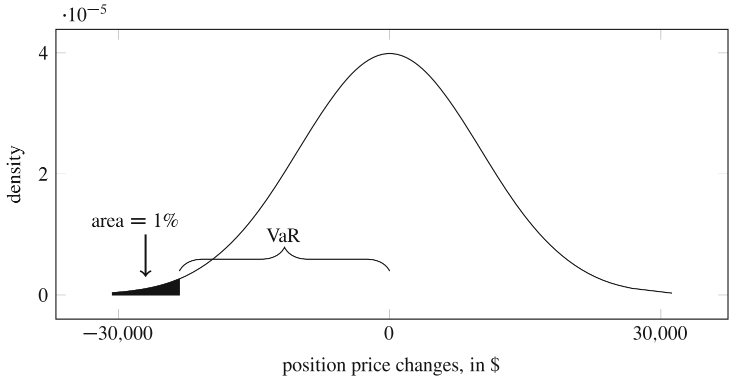

Fig. 2.7 Analytical price changes and VaR

We don't choose between these models but will rather use them both:

- The analytical approach is fast, which makes it well-suited for many support tasks. It yields reasonably accurate results if assets and positions behave well, and it thus serves as a good backup or sanity-check. It also provides, as side-products, useful hints for getting at the reasons of VaR changes.
- The historical approach is more precise because it embraces both non-normal and non-linear behavior. Nevertheless, it is simple to the point of being dumb, which is always a good feature that makes knowledge sharing easy and programming bugs rare.

A brief reminder before we start proper: you can find recaps of basic concepts like the standard deviation, the correlation, or the normal distribution in Appendix A. It is meant to be a bare, self-contained crash course in statistical topics most relevant to VaR modeling, or a slightly more formal reference. Otherwise, the following account will usually just intersperse the statistical results appropriate to the issues at hand.

Part I

Measures

Basic Terms and Notation 3

In this chapter, we give names to some basic concepts—assets, prices, returns, positions, portfolios, and profit-and-loss—and introduce a few related notational conventions.

Assets and Markets Financial *assets* are contracts that impose obligations to make and confer rights to receive future monetary payments. They are first set up or sold or *issued* mainly to raise money, as the rights come at a cost. This way, for example, a company going public sells *stock* or stakes in itself in exchange for the promise of handing out to the buyer some share of the profits to come. Assets are thus claims on future cash flows.

These assets (more specifically, the rights that come with them) can later be traded in financial markets like the stock market. These markets lower entry barriers and increase participation because asset buyers know that they can easily get rid of their investment. Markets thus help raise and allocate capital, borrow and lend money, and unwind or hedge such obligations. Market risk, now, is concerned not with the initial creation or issue of assets (that's the investment bankers' job) but with those markets' short-term behavior.

Apart from the prominent company stock, a wide range of asset types exists. A *bond* is a debt agreement; buying one guarantees a future repayment by the bond's issuer of the money "lent" to him in the initial bond sale, along with interest. A *call option* on a stock grants its buyer the right to a conditional future cash flow that depends on the stock's future performance. A *swap* comes with both rights and duties; here, two parties agree to periodically exchange payments of different types, for example, a fixed amount versus a variable one that depends on current market conditions. Even money itself—a government-issued IOU granting the right to settle public and private debts—can be considered a financial asset (alas, with guaranteed future payments of zero).

M. Auer, *Hands-On Value-at-Risk and Expected Shortfall*, Management for Professionals, https://doi.org/10.1007/978-3-319-72320-4_3

The perceived value of all such assets is driven by supply and demand in the markets; the recent trading transactions determine or, better, offer a glimpse on the assets' current prices. We can readily observe the price of General Motors stock, the price of a US government bond, and the price or *foreign exchange rate* of Swiss money.

Bond Basics Before we can proceed, we must take a quick digression and explore how these prices are best quoted. In theory, we could quote them all in monetary terms, i.e., as dollar prices—after all, money is what's actually changing hands in any given transaction. Yet for certain asset types, other ways to express the prices are more insightful, especially as they allow for better price comparisons.

Take a *fixed rate bond*, for example, which promises to pay out a certain dollar amount or *nominal* at some future date or *maturity*, and to also make several smaller, intermediate payments called *coupons*, e.g., of 2% of the bond's nominal each year. The relation between those future payments and the bond's current price allows for a consistent way to describe different bonds and to make them comparable. This is best illustrated with a *zero coupon bond* or *zero bond*, which is a special fixed rate bond promising only a final repayment of some nominal while making no coupon payments in between. Take, for example, a 1000$, 2-year zero bond worth 950$. We can determine how much money this bond "earns" by calculating, via compounded annual percentage increases, its *rate of interest* r from

$$950\,(1 + r)^2 = 1000.$$

It turns out that this bond pays $r = 2.6\%$ interest, which could now be easily compared to the rate achieved from a different zero bond, say, a 2000$, 7-year one currently trading at 1671$. (By the way, these two bonds earn the same interest.) The idea is easy to extend to bonds in general, since individual coupon payments can be viewed as small zero bonds.

So, basically, interest rates are best regarded as a convenient way of quoting (and thus comparing) actual bond prices. Of course, if you knew a bond's interest rate, maturity, and nominal, you could compute its current price; this is routinely done and called cash flow *discounting*. Still, it is helpful to keep in mind the underlying real dependency: interest rates are derived from actual bond transactions and prices; they do not determine those prices but express and replicate them.

Here are a few bond traits to keep in mind:

- The interest rate depends mainly on two factors: the creditworthiness of the bond issuer and the bond's time to maturity, i.e., how far off the repayment will take place. High creditworthiness means lower interest rates; a long time to maturity usually means higher interest rates. For each issuer, interest rates are expressed as functions of time to maturity, or $r(t)$.
- In place of the annual compounding, other compounding frequencies, like a semi-annual one, can be used just as well. The most convenient one is the limiting case

of *continuous* compounding,[1] which relates bond price p, nominal N, interest rate $r(t)$, and maturity t via

$$p\,e^{r(t)t} = N.$$

The reverse operation, discounting, in this case becomes

$$p = Ne^{-r(t)t}.$$

- To actually determine such interest rate functions $r(t)$, a procedure called *bootstrapping* is used. A simple example helps describe it. Assume that there are two bonds out there for some issuer, a 1-year zero bond B and a 2-year fixed rate bond B' with annual coupon payments. The first bond's price will immediately yield the 1-year interest rate $r(1)$ from

  $$p = Ne^{-r(1)\times 1}.$$

  The 2-year bond's first coupon payment, also due in one year, can then be discounted with that rate; the remaining 2-year payout of nominal plus final coupon then locks in the 2-year interest rate $r(2)$ via

  $$p' = c'e^{-r(1)\times 1} + (c' + N')e^{-r(2)\times 2}.$$

  This procedure can then be naturally extended to further, longer bonds. Alternatively, you can treat this as an optimization problem and try to find the $r(t)$ function that best fits the available bond prices. This means searching, e.g., with Excel's solver, for values x and y that minimize the following sum of squared pricing errors:

  $$(p - Ne^{-x\times 1})^2 + (p' - c'e^{-x\times 1} - (c' + N')e^{-y\times 2})^2.$$

- It is often convenient to quote some rates in terms of *spreads*. For this, you first determine a base interest rate, e.g., the US interest rate $r(t)$ derived from government debt. The interest $r'(t)$ some company's bonds pay can then be given as spread $s(t)$ over that base rate:

  $$s(t) = r'(t) - r(t).$$

[1]Interest rates and their compounding conventions are inseparable. Annual compounding with r, over a one year horizon, means we have to compound with $(1 + r)$. Semi-annual compounding with r, again over one year, means we must compound with $(1 + \frac{r}{2})^2$. Because $(1 + \frac{x}{n})^n$ converges to e^x for large n, compounding with e^r also seems to make sense as the limiting case of infinitely small, continuous compounding steps.

This allows us to elegantly separate generic from issuer-specific interest rate risk.

- Interest rates are usually positive, and prices of zero bonds thus smaller than their nominals. There are exceptions, e.g., in flight-to-safety circumstances, where bond buyers actually pay the borrower more than the nominal they'll get in return to safely store their wealth.
- Finally, a common unit encountered in dealing with interest rates is the so-called *basis point*: it denotes one one-hundredth of 1%, or 0.0001.

Scenarios The market's price information can be expressed, at any point in time, in a snapshot of asset prices we call *scenario*. Typical scenarios are the current market scenario, past or historical market scenarios, and hypothetical scenarios.

Say, we want to track General Motors (GM) stock, the currencies € and £, US and UK government bonds, as well as GM debt. The current market scenario $\mathbf{S}_0$ might look like this:

$$\mathbf{S}_0 = \begin{pmatrix} 35\$ \\ 1.10\$ \\ 1.54\$ \\ 0.52\% \\ 0.72\% \\ 1.40\% \\ 1.60\% \\ 2.10\% \\ 2.30\% \end{pmatrix} \leftrightarrow \begin{array}{c} \text{GM stock price} \\ \text{€/\$ exchange rate} \\ \text{£/\$ exchange rate} \\ \text{US interest rate, 1yr} \\ \text{US interest rate, 3yr} \\ \text{UK interest rate, 1yr} \\ \text{UK interest rate, 3yr} \\ \text{GM spread over US interest rate, 1yr} \\ \text{GM spread over US interest rate, 3yr} \end{array}$$

In practice, these scenarios are much larger vectors. We might want to track, say, 100 exchange rates; dozens of interest rate curves for major countries or currency regions, each ranging in maturities from 1 day/overnight, over 1 month, 3 months, 6 months, up to 50 or more years; spread curves for various issuers; stock prices; etc. For mnemonic purposes, let's assume we track 2200 assets.

The price history is, in our case, expressed in 520 historical scenarios $\mathbf{H}_1$ to $\mathbf{H}_{520}$, with the latest scenario $\mathbf{H}_{520}$ corresponding to the current market scenario $\mathbf{S}_0$. To compute VaR, sensitivities, and stress tests, we will soon use artificially created, hypothetical scenarios $\mathbf{S}_1$, $\mathbf{S}_2$, etc..

Risk Factors and Returns Each component or value v in a scenario is associated with an asset. Since those assets drive the market risk, we also refer to them as *risk factors*.

Changes in the values of risk factors, usually between consecutive days, are called *returns*. There are various ways to quote returns between two values v' and v, for example, absolute returns $v' - v$, relative returns $(v' - v)/v$, or logarithmic or log returns $\log(v'/v)$, which behave similarly to relative returns.[2] A risk factor's type suggests the appropriate return type to use. If a stock, say, is up by 2 dollars, it matters whether it was worth 4 or 400 dollars beforehand—we therefore use either relative or log returns for such prices. For interest rates, we use absolute returns, which can be neatly motivated by the close, proportional relation between the log return of bond prices $\log(e^{-r't}/e^{-rt})$ and the absolute return of the corresponding rates $r' - r$. You can find more details on return types in Chap. 9.

Finally, note that returns are dimensionless.

Positions and Portfolios We call our own asset exposures *positions*, and we denote them with Greek letters $(\alpha, \beta, \ldots)$. Examples are a position α of 10 units of GM stock ("10GM") or a three-year, 100\$ zero bond position β ("+100\$ @3yr"). Assume that we need to keep track of many positions, e.g., 10^6 different ones.

Typically, we are interested in a whole set or *portfolio* of positions:

$$\Omega = \alpha + \beta + \ldots$$

The mathematical operation + in such expressions denotes set operations, as you can't, e.g., really add a dollar and a unit of stock. Some basic arithmetic still makes sense, for example, $\alpha + \alpha = 2\alpha$ plainly means to double down on a position.

Negative signs denote that you *owe* something, which should come as no surprise to anyone who has ever seen a negative account balance like "−10\$." We can owe other assets as well, for example, 10 units of GM stock or "−10GM," in a position we might as well call $-\alpha$. To enter such a position, you borrow the stock and sell it—this is called to *short* the stock. It is used to bet on falling prices, for you might be able to buy the stock back on the cheap before returning it. (Being *long* something means to own it.)

Hedges If you borrow stock but decide to just keep it for a while, you enter two opposing positions at the same time: you owe $-\alpha = -10\text{GM}$ and also hold the borrowed $\alpha = 10\text{GM}$. Your portfolio is

$$\Omega = -\alpha + \alpha.$$

Even though the individual positions may change in value and are thus risky, this combined portfolio carries no risk: regardless of how the stock performs, you can obviously always just return it without losing money.

Another take on this is that such opposing positions effectively cancel each other out—whatever one position gains, the other one loses. For example, if you both own and owe euro, a falling euro will decrease the dollar value of your euro holdings but

[2]Careful with log returns in Excel: the natural logarithm is required, so use the `LN` function instead of the (tempting but base-10) `LOG` one.

also reduce your (euro) debt burden in dollar terms. Two positions that behave this way are said to *hedge* each other. They don't necessarily have to be strictly opposite positions as in our example of a perfect hedge. Two long stock positions, say, in a company selling swords and in one selling ploughshares, may also act as hedges if their stock price movements mirror each other closely.

Prices and Profit-and-Loss Determining the value of a position is called *pricing* the position. Conceptually, it means looking up the position-related asset prices in a given scenario. The position price is thus a function of the scenario. Here are some pricing examples under the scenario $\mathbf{S}_0$:

- A position α of "10GM" means we own some GM stock; its current dollar value or price is

$$p^{\alpha}(\mathbf{S}_0) = 350.$$

- A corresponding short position $-\alpha$ would be valued at

$$p^{-\alpha}(\mathbf{S}_0) = -350.$$

- A bond position β paying 100\$ in three years' time can be expressed as "+100\$ @3yr". We infer its current dollar price from

$$p^{\beta}(\mathbf{S}_0) = 100\, e^{-0.72\% \times 3}.$$

- Another bond position γ pays 70£ in two years' time ("+70£ @2yr"). Prices of UK bonds due in two years are not quoted in our scenario, but it seems reasonable to assume an interest rate that's somewhere between the 1- and 3-year ones given—we use their average of 1.50%.[3] We also, of course, need to convert pounds into dollars. The bond's price is then

$$p^{\gamma}(\mathbf{S}_0) = 70\, e^{-1.50\% \times 2} \times 1.54.$$

As asset prices move, the values of positions change. This change in value is called *profit-and-loss* or *PnL*, and it can be expressed as the difference of prices under two scenarios $\mathbf{S}_0$ and $\mathbf{S}_1$:

$$\Delta p^{\alpha} = p^{\alpha}(\mathbf{S}_1) - p^{\alpha}(\mathbf{S}_0).$$

[3]This price approximation is simple, but there are others far more complex. They are the subject of *quantitative finance*, which interpolates new asset prices from known ones.

Like prices, PnLs are dollar values. If GM stock is only worth 33$ in scenario $\mathbf{S}_1$, our stock positions α and $-\alpha$ change by

$$\begin{aligned}\Delta p^{\alpha} &= p^{\alpha}(\mathbf{S}_1) - p^{\alpha}(\mathbf{S}_0) = 330 - 350 = -20,\\ \Delta p^{-\alpha} &= p^{-\alpha}(\mathbf{S}_1) - p^{-\alpha}(\mathbf{S}_0) = -330 - (-350) = +20.\end{aligned}$$

Of course, prices and PnLs are additive, so the price and the PnL of a portfolio of positions (under the same scenario) are

$$\begin{aligned}p^{\Omega} &= p^{\alpha} + p^{\beta} + \dots,\\ \Delta p^{\Omega} &= \Delta p^{\alpha} + \Delta p^{\beta} + \dots\end{aligned}$$

For opposite positions, we have

$$\begin{aligned}p^{\alpha} &= -p^{-\alpha},\\ \Delta p^{\alpha} &= -\Delta p^{-\alpha}.\end{aligned}$$

When dealing with several hypothetical scenarios $\mathbf{S}_i$, we are often interested in whole vectors of price changes—PnL vectors—relative to a base scenario $\mathbf{S}_0$. We write

$$\Delta \mathbf{p}^{\alpha} = \left(p^{\alpha}(\mathbf{S}_1) - p^{\alpha}(\mathbf{S}_0), p^{\alpha}(\mathbf{S}_2) - p^{\alpha}(\mathbf{S}_0), \dots\right).$$

Please note: we will denote column vectors like scenarios with bold, uppercase letters $\mathbf{X}$; row vectors like PnL vectors with bold, lowercase letters $\mathbf{x}$; and matrices with brackets $[\mathbf{X}]$. We want the fundamental data entity of a single position's PnLs to be a row vector; this settles the orientation of the remaining vector types.

We try to keep the notation light and avoid indices as much as possible, and we use real-world ranges (like 2200 risk factors, a history of size 520, or our 10^6 positions) to more intuitively keep track of the dimensionalities involved.

4 Historical Value-at-Risk

One main concern of market risk management is to guess the plausible future behavior of a portfolio's value. There are two main parts to this:

1. Estimate asset price movements: *compute returns* and *generate scenarios*.
2. Determine the impact of those movements on the positions' values and distill portfolio risk measures: *price* positions, *aggregate* results, and *summarize*.

There are several ways to implement this, and they differ in accuracy, underlying assumptions, and computational effort. We propose to mainly rely on a variant of a historical approach and to back it up by a second, approximate one based on normal distribution assumptions. Both approaches are conceptually simple. The historical one is more accurate and makes few assumptions on the underlying market data, but it is computationally expensive. The approximate approach is less accurate and makes more assumptions on market data and pricing behavior, but it is computationally cheap.

This chapter describes the main historical approach. It assumes that risk factors will roughly behave like they did in the recent past, and that their observed returns exhibit volatilities that change over time. It computes historical risk factor returns; it rescales them to be responsive and mirrors them to increase the sample size; it creates hypothetical future scenarios; it prices each position under each scenario; it aggregates the prices into portfolio PnLs; and it reports summary stats derived from those PnLs.

We now outline these individual steps in more detail.

Step 1: Calculate Returns We base our analysis on the recent 2 years of market price information (1 year roughly contains 250 business days), more specifically, on 520 historical scenarios $\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_{520}$. Each historical scenario corresponds to the market scenario of a past day. We want to track 2200 risk factors, so each

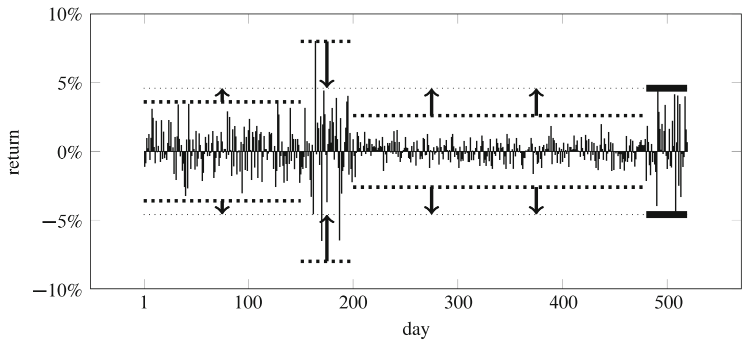

Fig. 4.1 Daily stock returns, raw

historical scenario is a column vector with 2200 entries. Aligned next to each other, these columns form a 2200×520 matrix $[\mathbf{H}]$ with 2200 rows and 520 columns.

Any row $\mathbf{h} = \left(h_1, h_2, \ldots, h_{520}\right)$ in this matrix corresponds to one risk factor and represents the time series of its values. For each such row, we calculate raw historical returns $\widetilde{r}_i$ between consecutive days. Note that different risk factors might use different return types and that we need to keep track of those types. For risk factors like stock prices and foreign exchange rates, we use log returns $\widetilde{r}_i = \log(h_{i+1}) - \log(h_i) = \log(h_{i+1}/h_i)$; for interest rate or spread risk factors, we use absolute returns $\widetilde{r}_i = h_{i+1} - h_i$.[1]

Doing this for each day in a row and for each row yields a 2200×519 matrix of *raw* returns $[\widetilde{\mathbf{R}}]$. The wobbly tilde should denote that those returns potentially exhibit a history of varying volatility.

Step 2: Rescale Returns As hinted at in Chap. 2, we now rescale the historical returns of each risk factor to its current volatility. Let $\widetilde{\mathbf{r}}$ denote any row of $[\widetilde{\mathbf{R}}]$, containing 519 historical returns associated with a risk factor. We assume that the most recent 20 returns $\widetilde{r}_{500}$ to $\widetilde{r}_{519}$ give us the best estimate of the current market volatility of that risk factor, and we use their standard deviation as *target volatility* T. We want to stretch, or increase in magnitude, past returns that took place in periods of low volatility; we want to squeeze, or decrease in magnitude, returns of past high volatility periods. We picture our goal in Fig. 4.1.

To accomplish this, we first estimate the *local volatility* L_i of *each* past return $\widetilde{r}_i$ as the standard deviation of the 20 returns $\widetilde{r}_{i-19}, \widetilde{r}_{i-18}, \ldots, \widetilde{r}_i$, i.e., over a window of size 20 that runs up to and includes $\widetilde{r}_i$. We then simply determine the *rescaled* return $\bar{r}_i$ as

$$\bar{r}_i = \frac{\widetilde{r}_i}{L_i} T.$$

[1]Other return types for interest rates can commonly be encountered (see Chap. 9).

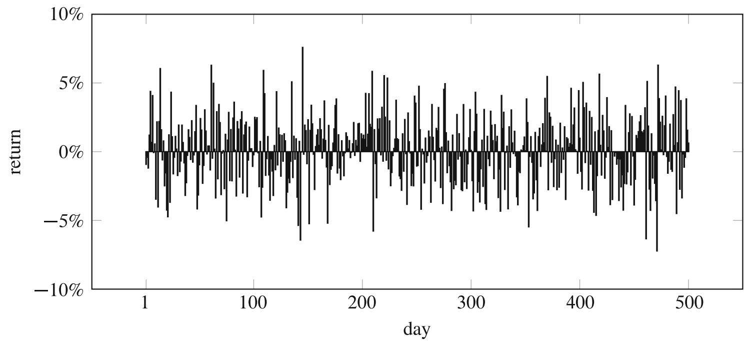

Fig. 4.2 Daily stock returns, rescaled

We do this for each return $\widetilde{r}_i$—except for the first 19 ones, as they don't have enough preceding values for that local vola. We discard them and are left, by a happy coincidence, with an even number of 500 rescaled returns we rename to $\bar{r}_1$ to $\bar{r}_{500}$. Figure 4.2 shows how the series of rescaled returns looks for our example.

Doing this for every row yields a 2200×500 matrix of rescaled returns $[\bar{\mathbf{R}}]$. The bar should denote the "even keel" volatility behavior of the rescaled returns.[2]

Step 3: Mirror Returns For our historical approach, we rely on past data. We need a decent number of days to do useful statistics, but we want to avoid time series much longer than 2 years, since we hold ancient information to be less relevant today. Still, the sample size of 500 returns seems to be a bit small, especially when compared to the large number of risk factors. A neat trick can help: if we assume that past returns were equally likely to go up or down, we can simply negate all of our returns and append those mirrored returns to our original sample, doubling its size. This does not unduly alter volatilities or correlations. The resulting return matrix is then the 2200×1000 matrix $[\bar{\mathbf{R}}, -\bar{\mathbf{R}}]$, or simply $[\mathbf{R}]$.

Step 4: Create Scenarios We are now ready to predict tomorrow's scenarios, based on the current market scenario $\mathbf{S}_0$.[3] Each entry in (the column vector) $\mathbf{S}_0$ corresponds to a value of a market risk factor; let one such value be s_0. If we take the corresponding return vector $\mathbf{r}$ (a row in our return matrix), we can generate a row of scenario values for that risk factor. Together with its current value as first entry, we obtain a scenario vector of size 1001.

[2] A simpler rescaling procedure would be to just rescale all returns with the same constant to achieve the desired target volatility. The drawback here is that—potentially brief—high-vola regimes would dominate the resulting correlation disposition.

[3] Note that $\mathbf{S}_0$ coincides with the latest, most recent historical scenario $\mathbf{H}_{520}$.

We just need to be mindful of the return type appropriate for any given risk factor:

- Absolute return scenarios are

$$\mathbf{s} = \left(s_0,\, s_0 + r_1,\, s_0 + r_2,\, \ldots,\, s_0 + r_{1000}\right).$$

- Log return scenarios are

$$\mathbf{s} = \left(s_0,\, s_0\, e^{r_1},\, s_0\, e^{r_2},\, \ldots,\, s_0\, e^{r_{1000}}\right).$$

Repeating this process for each risk factor yields the 2200×1001 scenario matrix $[\mathbf{S}]$. Its first column is $\mathbf{S}_0$, the current market scenario. The remaining 1000 columns $\mathbf{S}_1$ to $\mathbf{S}_{1000}$ are our risk factors' hypothetical VaR scenarios.

Step 5: Price Positions and Determine PnLs For any given position, we can determine prices under each scenario:

$$\mathbf{p} = \left(p(\mathbf{S}_0),\, p(\mathbf{S}_1),\, p(\mathbf{S}_2),\, \ldots,\, p(\mathbf{S}_{1000})\right).$$

Subtracting the current market price from the other scenario prices yields the PnL vector of size 1000:

$$\Delta\mathbf{p} = \left(p(\mathbf{S}_1) - p(\mathbf{S}_0),\, p(\mathbf{S}_2) - p(\mathbf{S}_0),\, \ldots,\, p(\mathbf{S}_{1000}) - p(\mathbf{S}_0)\right).$$

We repeat this step for each of our 10^6 positions.

Step 6: Aggregate Positions We are usually interested in the PnL behavior of a portfolio of positions $\Omega = \alpha + \beta + \ldots$; to obtain the portfolio PnLs, we simply add the positions' PnL vectors:

$$\Delta\mathbf{p}^{\Omega} = \Delta\mathbf{p}^{\alpha} + \Delta\mathbf{p}^{\beta} + \ldots$$

Step 7: Determine Risk Measures For any PnL vector—of a position or a portfolio—we can then obtain various risk measures. For the 1%-VaR, we sort the vector and (because our PnL vectors are of size 1000) take its 10th most negative entry. We denote this by the bracketed index "(10)":

$$\mathrm{VaR}[\Omega] = \Delta\mathbf{p}^{\Omega}_{(10)}.$$

Figure 4.3 outlines these steps and traces the various matrix dimensions.

And that's all there is, really, to our VaR model. Yet since it can be easy to lose sight of the wood for the trees, we must re-emphasize the one important and slightly non-trivial step—the rescaling of the historical returns. The proposed way to do it is to apply a simple variant of a feature used by the family of filtered historical simulations. (Originally, such simulations use so-called GARCH

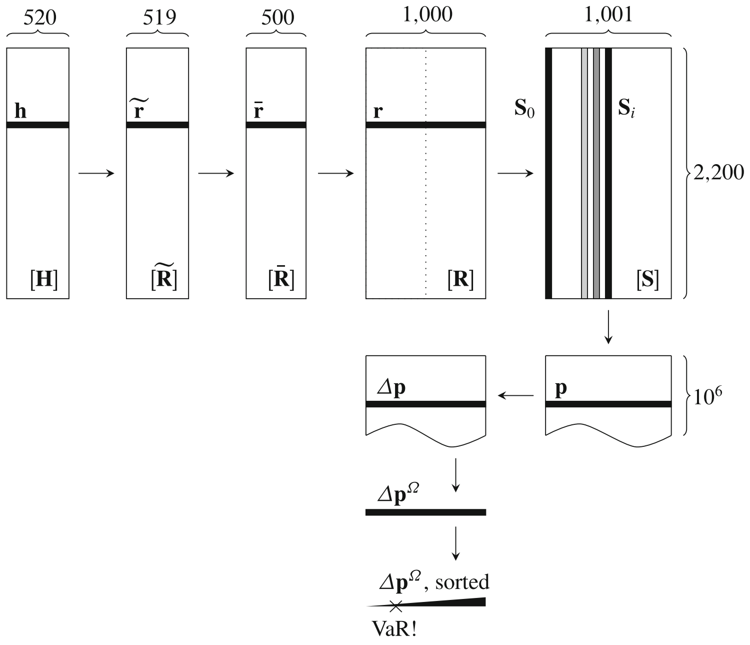

Fig. 4.3 Returns, scenarios, prices, PnLs, and VaR

estimates to determine local vola levels, but otherwise they operate equivalently.) The rescaling of returns is sometimes referred to as *volatility declustering*; we will mostly stick to calling it *volatility rescaling*.

This rescaling is important because it allows the use of the full set of historical returns and their correlation information while still accounting for the most recent vola levels seen in the market—the model is not beholden to potentially unrealistic, long-gone vola behavior. The model is able to react quickly to changes in the markets, so if volatilities increase, it will very soon yield more sizeable VaR estimates and indicate a larger risk. For more on this model reactivity, please refer to Chap. 14.

Vola rescaling, even though it operates on time series, should not be viewed through the lens of time series modeling. It is largely parameter-free, does not minimize error functions, and estimates no regression coefficients. It is probably best interpreted as *operator*, as a vector-to-vector function $f(\cdot)$,[4] which takes raw historical returns as input and yields rescaled returns as output. It has only two goals: (1) to make the rescaled returns as volatile as desired and (2) to preserve the co-movements and thereby correlations of historical return pairs (note how rescaling never changes a return's direction).

[4]The various operators in fields like signal or image processing are often called *filters*.

The operator does not aim to do anything else. Specifically, it does not try to extract or preserve further statistical properties of the original time series. We implicitly assume that many such characteristics change over time and that it is neither very useful nor feasible to try to grasp them. (See Sect. 17.6 for a more detailed discussion of a specific statistical property the operator ignores.)

Now, whether such an operator is "correct" or "wrong" is not really a valid question—it can certainly be technically computed, and it clearly achieves its, very restricted, ambitions. But whether it is *sensible* in light of the overall goal of having a realistic VaR estimate, i.e., whether it is possible to discard much of the historical data's properties, only end-to-end model tests like those given in Chaps. 15 and 16 can reveal.

Some final remarks and links to follow-up topics:

- One model parameter we encountered is the size of the windows used to determine local vola levels, in our case 20. This could be regarded as being a *meta parameter*, i.e., one with no mathematical optimality to it, set at the model users' discretion. Defending the choice of parameters whose meta aspect is questioned can be challenging; Sect. 17.4 delves into this topic.
- The local vola a return is rescaled with is affected by the return itself. This might lead to questions from model reviewers, as some alternative approaches constrain the vola estimate to rely on strictly preceding returns only. In short, both options work. The advocated one can be considered to operate plainly in the vein of, for instance, a moving average operator, which also "self-impacts" and does so without scrutiny. The case for our choice is one of numerical stability in rare cases; Chap. 9 has more on this topic of the local vola window's location.
- Another variant of the historical simulation is the BRW approach,[5] which does not use volatility rescaling. It instead computes portfolio PnLs from raw historical returns and assigns weights (summing up to 1) to them, with recent PnLs weighted more prominently. The PnLs are sorted *along* with their weights; a 1% cumulative weight of the largest losses then signals the PnL to be used as VaR. This is a perfectly fine and workable method. In my view, however, the approach proposed in the present chapter is slightly more elegant: bare-bone BRW (without mirroring) dismisses positive PnLs in trying to react to market vola changes; it requires some extra steps, however simple (weight handling and, typically, interpolation), which makes manual test computations or comparisons a tiny bit more involved; the decay parameter determining the weights is, in my view, less tangible than plain vola windows; the correlations that end up getting used are also weighted towards the recent past and potentially less stable; and adding a Monte Carlo layer would be faintly more cumbersome. Still, expect its results to strongly resemble those in our approach.

  (You can find further references, for example, for the filtered or the BRW approach, in Appendix C.)

[5]So named after Boudoukh, Richardson, and Whitelaw.

Sensitivities 5

The price of a position depends on the underlying assets or risk factors, and we express this price as the function $p(\mathbf{S})$ of a scenario. A natural question to ask is how this price reacts to specific scenario changes. The particular price change resulting from a small change in *only one* of the underlying risk factors is called the *sensitivity* of the position with regard to that risk factor.

To calculate a specific sensitivity, we can change one risk factor value of a given scenario and reprice the position with the new scenario—the difference to the original price is the sensitivity. By convention, we typically use a relative or absolute change of 1 basis point.

Take, for example, a zero bond that pays 1,000,000€ in 2 years' time, and assume a 2-year interest rate (IR-EUR-Y02) of 3% as well as a €/$ exchange rate (FX-EUR) of 1.11. The current dollar price of this bond is

$$1{,}000{,}000 \times e^{-3\% \times 2} \times 1.11 = 1{,}045{,}358.63.$$

Its sensitivity with regard to the interest rate risk factor is

$$1{,}000{,}000 \times e^{-3.01\% \times 2} \times 1.11 - 1{,}045{,}358.63 = -209.05. \tag{5.1}$$

So this position *loses* 209$ in value if the 2-year interest rate increases slightly.

Its sensitivity with regard to the exchange rate risk factor is

$$1{,}000{,}000 \times e^{-3\% \times 2} \times 1.110111 - 1{,}045{,}358.63 = +104.54. \tag{5.2}$$

This position therefore *gains* 104$ in value if the exchange rate increases slightly, i.e., if the euro appreciates against the dollar.

(Notice the absolute 1-basis point increase in the interest rate from $3\% = 0.03$ to $3.01\% = 0.0301$, and the logarithmic 1-basis point increase in the exchange rate from 1.11 to $1.110111 = 1.11 \times e^{0.0001}$.)

The sensitivities with regard to all other 2198 risk factors are zero—those risk factors do not influence this position's price.

Sensitivities let us compare positions by pointing out which risk factors they are exposed to, which direction this exposure takes, and which positions are exposed most to any specific risk factor. They also allow us to compare risk factors and to determine the most influential ones.

Sensitivities and market risk measures like the VaR are related. Typically, a portfolio with zero sensitivity to a specific risk factor will not show market risk with regard to that risk factor (except in special non-linear setups). A large positive or negative sensitivity, however, hints at major risk drivers. A risk factor, for example, with a sensitivity of 1000 (and that per just 1 basis point!) can apparently noticeably affect prices and thus our risk as it changes. (Since those changes can go either way, the sign of the sensitivity does not tell us all that much here.) Two risk factors with a similar sensitivity may affect the overall risk differently, though: a risk factor that typically jumps between $+5$ and -5 basis points has less impact than a risk factor of the same sensitivity but with a $+10$ to -10 jump range. We will explore the relation between sensitivities, volatilities, and risk in Chap. 7.

Conventions The bond in our example above has a maturity of exactly 2 years. What if it had one of 18 months, i.e., 1.5 years? Different sensitivity conventions can be used in this case. (Assume that, like before, our 1-year interest rate is 2%, that our 2-year interest rate is 3%, and that such a bond is priced with the interpolated, 1.5-year interest rate of 2.5%.)

- Theoretically, we could just define the 1-year sensitivity to mean the price change from a 1-basis point shift of the interest rate in the whole interval [1 year, 2 year), i.e., via a partial parallel shift of the interest rate curve (see Fig. 5.1). The bond would be repriced under $r'(1.5) = 2.51\%$.

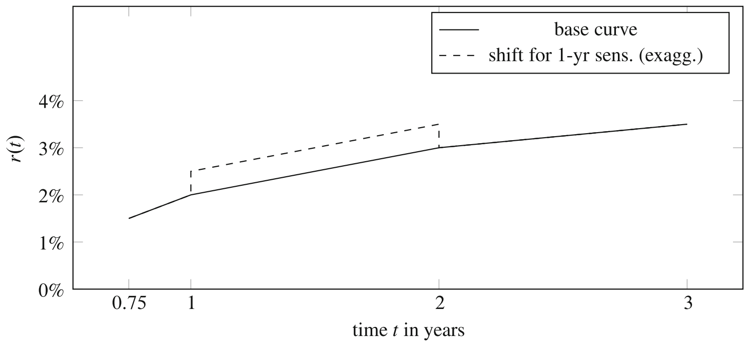

Fig. 5.1 Partial parallel shift of interest rate curve

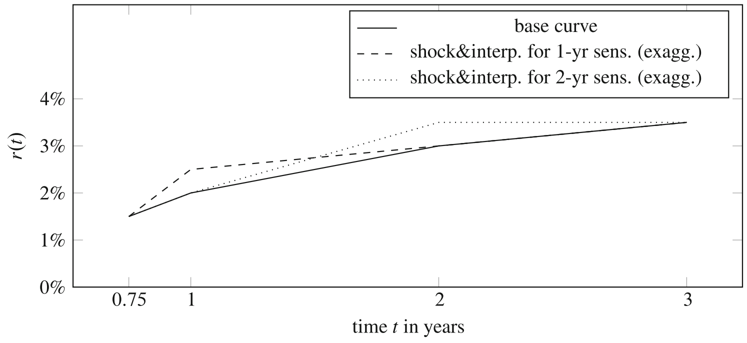

Fig. 5.2 Bending the interest rate curve like a bowstring

- Alternatively, we could apply a 1-basis point shock to the rate at one specific maturity only and then interpolate towards the neighboring maturities' unchanged rates—much like bending a bowstring. Figure 5.2 illustrates this.

  In this case, our bond will, due to the interpolations, only experience an effective 0.5-basis point impact when calculating its 1-year sensitivity. However, it will *also* experience a 0.5-basis point impact from the 2-year sensitivity calculation. This effectively smears out the bond's sensitivities across two risk factors or interest rate maturities. (Their sum, though, will be very close to the parallel shift sensitivity above.)

The bowstring approach is preferable. It is consistent with how we would interpolate in the VaR calculation (this lets us use the same framework without any sensitivity-specific adaptions). The most important point in its favor, though, is its numerical stability. In the parallel shift approach, a bond with a maturity of 2 years and 1 day will only affect the 2-year sensitivity; 2 days later, its maturity will be 2 years minus 1 day, and thus fully impact the 1-year sensitivity. This sudden jump is a bit jarring.

It gets even worse for the parallel approach when it comes to hedges. That is because real-world hedges might easily be a few days off with regard to their maturities, i.e., not match perfectly. If both positions in such a hedge fall into the same time interval that is parallel-shifted, their sensitivities correctly cancel each other out to zero. Yet close to the interval borders, the hedges might happen to fall into distinct intervals—we'd observe two opposing sensitivity spikes where ideally there should be none at all. This is unwarranted and distracting noise. The bowstring approach, now, is able to neatly avoid this noise: as time passes, it just makes sensitivities flow very gradually from maturity to maturity. There are no worries about interval boundaries, no sudden sensitivity jumps, and no hedges split asunder.

Framework To actually compute the sensitivities, we can use the same framework used in the VaR setup encountered before. Starting off with our current market scenario $\mathbf{S}_0$, we create a new scenario $\mathbf{S}_1$ that is identical to $\mathbf{S}_0$ except for a 1-basis point return applied to its 1st entry. We then copy $\mathbf{S}_0$ to a new $\mathbf{S}_2$, applying a return only to its second entry, and so on. In the end, each of the 2200 new scenarios differs from $\mathbf{S}_0$ in only one entry.[1] All the $\mathbf{S}_i$ together form a 2200×2201 scenario matrix.

For any position, these scenarios then yield 2201 prices and 2200 price changes or PnLs, each a sensitivity associated to one risk factor:

$$\mathbf{s}^\alpha = \left(p^\alpha(\mathbf{S}_1) - p^\alpha(\mathbf{S}_0),\, p^\alpha(\mathbf{S}_2) - p^\alpha(\mathbf{S}_0), \ldots, p^\alpha(\mathbf{S}_{2200}) - p^\alpha(\mathbf{S}_0)\right).$$

Such a sensitivity vector describes the individual exposures of a position to all the risk factors; it is usually non-zero for only a few of them. Sensitivity vectors are, like other vectors of price changes, additive, and the portfolio sensitivities become

$$\mathbf{s}^\Omega = \mathbf{s}^\alpha + \mathbf{s}^\beta + \ldots$$

Another way sensitivities may be added is *within* one position or portfolio. We can, for example, take a sensitivity vector and add just its IR-EUR entries of various maturities:

$$\mathbf{s}^\alpha_{\text{IR-EUR-M09}} + \mathbf{s}^\alpha_{\text{IR-EUR-Y01}} + \mathbf{s}^\alpha_{\text{IR-EUR-Y02}} + \mathbf{s}^\alpha_{\text{IR-EUR-Y03}}.$$

This yields the price impact of a full, 1-basis point parallel shift of the interest rate curve,[2] a commonly used measure.

Not all sensitivity entries of different risk factors, however, can be meaningfully added—adding two different exchange rate sensitivities apparently makes little sense. In special cases, however, we might want to add the absolute values of different sensitivities, e.g., if we want to limit some overall measure of sensitivity exposure to whole classes of risk factors.

If we know a risk factor's sensitivity s, we know the price impact of a 1-basis point change in that risk factor. The approximate price impact of a 100-basis point change is then, presumably, $100\,\text{s}$. This extrapolation works reasonably well for small ranges and for positions whose pricing behavior doesn't deviate much from linearity, and it is a handy back-of-the-envelope shortcut to estimate PnL impacts. For an exact result, a full repricing is required (see also Chap. 6).

One more remark about the vector framework used above. As most entries in a position's sensitivity vector will be zero, a simple optimization will avoid potentially costly repricings under all but the few relevant sensitivity scenarios (see Sect. 20.7).

[1]This is equivalent of using a 2200×2200 return matrix with only diagonal entries of 10^{-4}.

[2]Convince yourself of this by mentally adding the bowstring approach's overlapping triangles.

Relation to Derivatives Conceptually, sensitivities are partial derivatives quoted in a more convenient scale. To see how sensitivities and partial derivatives are aligned, consider this example of some position's price expressed as a two-dimensional function of asset returns:

$$f(x_1, x_2) = 1{,}000{,}000 \times e^{-(3\%+x_1)\times 2} \times 1.11e^{0+x_2}.$$

The current price $p(\mathbf{S}_0)$ corresponds to $f(0,0)$, i.e., the "no returns" case.

We could now calculate the partial derivatives d_1 and d_2 of f analytically, or just lazily approximate them numerically via

$$\begin{aligned} d_1 &= \frac{f(0.0001, 0) - f(0,0)}{0.0001}, \\ d_2 &= \frac{f(0, 0.0001) - f(0,0)}{0.0001}. \end{aligned}$$

These two expressions correspond to Eqs. (5.1) and (5.2), save for the usual normalization to step size 1 in the denominator here. We realize that

$$\left(d_1, d_2\right) = \mathbf{d} = \frac{\mathbf{s}}{0.0001} = 10^4\,\mathbf{s}.$$

6 Stress Tests

The sensitivity measure we encountered in the previous chapter gives us price impacts of individual, small risk factor changes; it mainly provides comparability of exposures across risk factors. To gauge the impact of simultaneous and large changes to several risk factors at once, we reprice our positions under custom-made scenarios—this is called *stress testing*.

The stress tests often mimic certain past events, for example, 9/11 or the Lehman collapse in 2008. To create a stress scenario $\mathbf{S}_i$, we simply apply the desired risk factor returns—either historically observed or custom-designed ones like the popular "200-basis point shift in all interest rates"—to our current market scenario. It is not uncommon to create hundreds of such stress scenarios, many of which may be prescribed by the regulator. Like with the VaR and sensitivities, once such a stress scenario matrix is set up, prices and PnLs can be handled in our customary pricing framework.

A tiny detail might warrant your attention. We know that for small shocks, some return types are basically interchangeable (a relative return of one basis point is very close to a logarithmic one of the same size, for example). But stress tests usually apply rather large shocks, which makes return types deviate substantially. So make sure to ascertain the appropriate return type to use if tasked, for example, with a regulatory stress exercise of a -50% shock to all stocks. A relative return of that magnitude would halve a stock's value from 100$ to 50$; a logarithmic one of the same magnitude yields what might not immediately spring to mind: 60.65$. Also note that a -120% shock to a stock price is valid in logarithmic terms (it wipes off about 70 of 100$) while making much less sense in relative terms.

Relation to Sensitivities Sensitivities can be used to approximate stress test results; after all, they can be considered atomic or "mini" stress tests.

To estimate the PnL impact of a joint $(+20, +10)$-basis point change in two risk factors, we could scale the two respective (1-basis point) sensitivities by 20 and 10, and add the results. The first approximation error can be spotted in the example position outlined in Chap. 2, where we neglected the interaction term *pxy* that only a full repricing takes into account.

The second and usually more grave issue concerns the scaling of sensitivities itself. If a position's pricing function does not behave linearly with respect to the underlying returns, this scaling—especially if large—leads to another approximation error. Now, even basic pricing approaches like discounting exhibit slightly non-linear behavior, but it can often be neglected.[1] Let's examine an asset type where this effect is more pronounced.

Example for Non-linearity We dropped this asset's name in Chap. 3, and it is now time to revisit it: a *call option* on a stock S pays out, at some future date or *expiry*, either zero if the stock price at expiry is below some level or *strike* K, or it pays out $S - K$ if the stock price is above.

Clearly, such an option always has a positive value or price, regardless of the current stock price or strike—after all, we can't lose any money. The option is also worth more the higher the current stock price is, as a higher stock price makes a payout both more likely and potentially larger. In other words: if the stock goes up, so does the value of the call option.

Consider now a current stock price of 30$, a strike of 50$, and an expiry in 1 year's time. The option would be worth very little, as it's not all that likely that the stock could increase by at least 20$ during that time. If the stock was to suddenly increase to 31$, the option's value would be only minimally higher (the stock is still unlikely to make up the now-missing 19$), so the call option's PnL resulting from this 1$ jump will be some very small Δp_{S+1}, in this example, around 0.01$. Yet if the stock price suddenly jumped up to 65$, we would expect the option to soon yield approximately 65$ − 50$ = 15$. It will thus be immediately worth roundabout that amount, and Δp_{S+35} will be very roughly about 15$.

We can immediately see that this pricing behavior is non-linear—just scaling up the PnL from the 1$ jump by a factor of 35 will not do:

$$35\,\Delta p_{S+1} \approx 0.35\$ \neq 15\$ \approx \Delta p_{S+35}.$$

So for non-linear assets, we obviously can't simply extrapolate PnLs for large risk factor changes from the PnLs of small ones, and we must resort to a full repricing. (The difference or error made in such an approximation attempt, however, can be used as a measure for the exposure to non-linearity.)

[1]Compare, e.g., the PnL impact of a 2% interest rate change to twice the PnL impact of a 1% change—they differ:

$$e^{-(r+2\%)} - e^{-r} \neq 2(e^{-(r+1\%)} - e^{-r}).$$

7 Analytical Value-at-Risk

A second approach, already mentioned at the beginning, to calculate the VaR is an analytical one. It is only approximate, as its assumptions don't always hold in practice, but it involves fewer computational steps because it relies on sensitivities and avoids the 1000×10^6 full position pricings. Often it is very close to the VaR obtained in the historical simulation, which makes it a useful sanity-check. It also clearly exposes the relation between the VaR and the sensitivities, volatilities, and correlations. Even more importantly, it provides some helpful analysis tools in dealing with the questions we're most interested in: How does the VaR react if we change our positions? What risk factors contribute most to the VaR? What is the reason for a particular VaR change?

This small chapter is mathematically the most involved. At first reading, you could also just skip it to avoid getting bogged down. Maybe return to it later if you find, e.g., the VaR-contribution helpful in dealing with the model's daily operation, as described in Chap. 17 entitled "Nine to Five." (If you tackle it, you can find some background on multiple randoms, the covariance matrix, and normal quantiles in Appendix A.)

7.1 Approximate VaR

We start off with our familiar 2200×1000 matrix of returns $[\mathbf{R}]$ (recall it from Fig. 4.3). It represents, after some rescaling and mirroring, our view of the history of asset or risk factor returns. Each row is associated to a risk factor and represents a historical sample of its returns.

We can also view each risk factor as a normal random variable x_i instead and denote all risk factors as

$$\mathbf{X} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{2200} \end{pmatrix}.$$

These random variables have certain properties that can be estimated from their corresponding historical samples. Each x_i has a variance $\mathbb{V}\text{ar}[x_i]$, and each pair of x_i, x_j has a covariance $\mathbb{C}\text{ov}[x_i, x_j]$. Their estimates form the symmetric covariance matrix $[\mathbf{C}]$.[1]

Now assume that the current price of a position or portfolio is p_0 and that its price under various returns is a function of $\mathbf{X}$, denoted as $p(\mathbf{X})$. We are, as ever, interested in price *changes* or PnLs, which we can express as

$$\Delta(\mathbf{X}) = p(\mathbf{X}) - p_0.$$

This is a function of the random variables $\mathbf{X}$, and we'd like to get a handle on its volatility, i.e., standard deviation. For this, we first simplify Δ in a first-order Taylor expansion (we'll soon simply drop the anxious "$\approx$" and let the linearity assumption take over):

$$\Delta(\mathbf{X}) \approx p_0 + x_1 \frac{\partial p}{\partial x_1}(0) + x_2 \frac{\partial p}{\partial x_2}(0) + \cdots - p_0.$$

If we denote the partial derivatives here with d_i, we may write:

$$\Delta(\mathbf{X}) = d_1 x_1 + d_2 x_2 + \ldots$$

This Δ is a weighted sum of random variables. With $\mathbf{d} = \big(d_1, d_2, \ldots\big)$, its variance and standard deviation become

$$\begin{aligned} \mathbb{V}\text{ar}[\Delta] &= \mathbf{d}[\mathbf{C}]\mathbf{d}^\top, \\ \text{std}[\Delta] &= \sqrt{\mathbb{V}\text{ar}[\Delta]} = \sqrt{\mathbf{d}[\mathbf{C}]\mathbf{d}^\top}. \end{aligned}$$

[1] Hence this method's alternative name of *variance-covariance* approach.

Because Δ, as sum of normals, is normal itself, the 1%-quantile or VaR[2] can now be directly derived from its standard deviation:

$$\mathrm{VaR}(\mathbf{d}) = \sqrt{\mathbf{d}[\mathbf{C}]\mathbf{d}^\top} \times (-2.33..).$$

We don't want to compute the partial derivatives. Luckily, for any position or portfolio we usually have ready access to a close relative of their derivatives d_i—their sensitivities s_i, collected in the vector $\mathbf{s}$. We have already seen that $\mathbf{d} = 10^4\mathbf{s}$, which gets us the following useful expression for the VaR as a function of our sensitivities:

$$\begin{aligned}\mathrm{VaR}(\mathbf{s}) &= \sqrt{(10^4\mathbf{s})[\mathbf{C}](10^4\mathbf{s}^\top)} \times (-2.33..)\\ &= -2.33.. \times 10^4 \times \sqrt{\mathbf{s}[\mathbf{C}]\mathbf{s}^\top}\\ &= c\sqrt{\mathbf{s}[\mathbf{C}]\mathbf{s}^\top}.\end{aligned}$$

We hereby have a fast way to estimate the VaR for each position or portfolio whose sensitivity vector we know.

(Note: the behavior of this VaR estimate depends on the covariance matrix. If we use rescaled returns to estimate it, this analytical VaR will also be aligned to the most recently observed market volatilities.)

7.2 VaR-Sensitivity

Our VaR function here is expressed as a single-valued function of all the s_i. A natural question to ask is then: how does this VaR change if the underlying sensitivities change (i.e., if our exposures or, effectively, our positions change)? To answer this, we need to look at the partial derivatives of the VaR function itself, called *VaR-sensitivities*. There is one for each risk factor.

To compute these VaR-sensitivities, let us first denote the vector of partial derivatives of any single-valued vector function $f(\mathbf{x})$ as follows:

$$\frac{\partial f}{\partial \mathbf{x}} = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots\right).$$

We recall that for any quadratic form $g(\mathbf{x}) = \mathbf{x}[\mathbf{A}]\mathbf{x}^\top$ we have

$$\frac{\partial g}{\partial \mathbf{x}} = \mathbf{x}([\mathbf{A}] + [\mathbf{A}]^\top).$$

[2]The variance "$\mathbb{V}$ar" is not the value-at-risk "VaR."

Mindful that $[\mathbf{C}] = [\mathbf{C}]^\top$, we proceed to partially differentiate our VaR function by its sensitivities s_i:

$$\begin{aligned}\mathbf{v}(\mathbf{s}) := \frac{\partial \operatorname{VaR}}{\partial \mathbf{s}} &= \frac{\partial c\sqrt{\mathbf{s}[\mathbf{C}]\mathbf{s}^\top}}{\partial \mathbf{s}} \\ &= \frac{1}{2}c\frac{1}{\sqrt{\mathbf{s}[\mathbf{C}]\mathbf{s}^\top}}\frac{\partial \mathbf{s}[\mathbf{C}]\mathbf{s}^\top}{\partial \mathbf{s}} \\ &= \frac{1}{2}c\frac{1}{\sqrt{\mathbf{s}[\mathbf{C}]\mathbf{s}^\top}}\mathbf{s}([\mathbf{C}] + [\mathbf{C}]^\top) \\ &= c\frac{\mathbf{s}[\mathbf{C}]}{\sqrt{\mathbf{s}[\mathbf{C}]\mathbf{s}^\top}}.\end{aligned}$$

Each entry v_i in the vector of VaR-sensitivities $\mathbf{v}$ describes by how much the VaR changes if the corresponding sensitivity increases by 1$.

Example We would like to increase our exposure to asset i. Its sensitivity $s_i = -12{,}000$. The corresponding VaR-sensitivity $v_i = -8$. Changing the sensitivity by, say, $-1000$$ should then approximately affect the VaR by $-1000 \times -8 = +8000$, so we'd expect the (negative) VaR to change by $+8000$, decreasing in magnitude.

VaR-sensitivities are handy for quickly assessing the sign and magnitude of the VaR impact due to a prospective change in sensitivities, i.e., in positions. For relatively small sensitivity changes, it is also fairly accurate, but we can just as easily employ an exact, full repricing. The most useful application of VaR-sensitivities, however, is described next.

7.3 VaR-Contribution

Our expression for the VaR turns out to be a so-called homogeneous function of order 1, since for any scaling factor a we have

$$\operatorname{VaR}(a\mathbf{s}) = a\operatorname{VaR}(\mathbf{s}).$$

This allows us to write, per Euler's homogeneous function theorem,

$$\operatorname{VaR}(\mathbf{s}) = s_1v_1 + s_2v_2 + \ldots$$

The terms s_iv_i sum up to the VaR-value; we call such a term the *VaR-contribution* $c_i = s_iv_i$ of its risk factor. These VaR-contributions can be positive or negative. Large negative VaR-contributions indicate the risk factors that dominate or drive the VaR's magnitude. An example for a portfolio whose VaR is driven by the euro/dollar exchange rate is given in Table 17.1.

8 Expected Shortfall

The VaR ignores quite a bit of seemingly important information—those losses that are even larger than the VaR. To take large losses into account, we could measure, e.g., the average of the 2.5% largest losses. This is called *expected shortfall* or *ES*. In our case of 1000 PnL values, it is given by

$$\mathrm{ES}[\Omega] = \frac{1}{25}(\Delta \mathbf{p}^{\Omega}_{(1)} + \Delta \mathbf{p}^{\Omega}_{(2)} + \cdots + \Delta \mathbf{p}^{\Omega}_{(25)}). \tag{8.1}$$

In other words, and similarly to the way we determine the VaR, we just sort the vector of PnLs and average the worst 25 ones.

Why specifically use 25 of the 1000 PnL values? This is because of how VaR and ES are related in a normal distribution: the 1%-VaR there is very close to the 2.5%-ES. We have already encountered the multiplier 2.32635 that relates our analytical VaR to a normal standard deviation; the respective multiplier for the ES is 2.33780. This basically allows us to replace the 1%-VaR on the fly with the 2.5%-ES at minimal disruption, e.g., when these measures are used to determine capital requirements.

The analytical VaR approach in Chap. 7 would directly translate to an equivalent ES one (up to the almost identical constant mentioned); we would gain nothing by switching measures in this case. In the historical VaR calculation, however, the VaR and ES can differ much more, depending on how the largest losses behave.

ES has become more popular recently. It is supposed to account for some VaR weaknesses, and its application is usually required by regulation. We will outline its properties in Chap. 13. Without giving too much away, it is not necessarily a much better measure than the VaR, as it comes with its own share of drawbacks.

It does, however, offer a transparent and stable attribution of risk to individual positions that is rather convenient. Consider a portfolio $\Omega = \alpha + \beta + \ldots$. We can compute ES[$\Omega$] and the individual positions' ES[α], ES[β], etc., as described above. The ES of individual positions behaves fairly similarly to their individual VaRs and won't be of much help (more on this in Chap. 11). But a modified version of it, the *conditional ES* or *cES* for individual positions with respect to their portfolio, will be. It is calculated as follows:

- First, sort the portfolio PnL vector $\Delta\mathbf{p}^{\Omega}$ and remember the scenario indices of the 25 largest losses. For example, the 3rd largest loss "$(3 \sim \Omega)$" might reside at index 714 of Ω's PnL vector.
- Then, for a position α, calculate its cES as the average of exactly those entries in its $\Delta\mathbf{p}^{\alpha}$. In other words, we average exactly those position PnLs that contribute to the 25 largest losses for the portfolio. The cES of the position α with respect to the portfolio Ω is

$$\mathrm{cES}[\alpha|\Omega] = \frac{1}{25}(\Delta\mathbf{p}^{\alpha}_{(1\sim\Omega)} + \Delta\mathbf{p}^{\alpha}_{(2\sim\Omega)} + \cdots + \Delta\mathbf{p}^{\alpha}_{(25\sim\Omega)}).$$

We will use the cES in much the same way as the VaR-contribution described in Chap. 7, as the cES values of a portfolio's positions sum up to the portfolio's (unconditional!) ES:

$$\mathrm{ES}[\Omega] = \mathrm{cES}[\alpha|\Omega] + \mathrm{cES}[\beta|\Omega] + \ldots$$

Again, negative cES values indicate risk-increasing positions; positive values risk-decreasing ones. Chapter 13 will put these cES features to good use.

Note: the VaR-contribution decomposes the VaR additively to *risk factors*, while the cES decomposes the ES additively to *positions*. What about a partner swap? Well, a hypothetical analytical "ES-contribution" of risk factors would provide no additional insight, as mentioned above. A "conditional VaR" for positions, on the other hand, is easy to contrive but suffers from instability (this is discussed at the beginning of Chap. 11).

Model Choices 9

Without much ado, Chap. 4 outlined a relatively straightforward historical VaR model. In the bank I worked at, such a model proved to work reliably right through the 2008 financial crisis and its aftermath. Many a model aspect, however, could be tuned or tweaked or altered, and this chapter zooms in on some of those model choices. But how to weigh these features, how to choose between model options? Let me give you my personal take on this.

By far the most important aspect of a VaR model is its input data. Thousands of risk factors must be diligently tracked over years from different and evolving market data sources. Market data may contain outliers and invalid entries, missing values and whole gaps, or stale and constant periods; market data sources may break down, change quoting conventions, or become inconsistent with other sources. And eventually, all of this is going to happen. Some of those glitches can be avoided by careful and continuous data screening, by manual overrides, by automatic checks and warnings, etc.—in other words, by patience and diligence, also known as time and money. But illiquid markets will continue to provide data that is inherently spotty and unreliable. Unaligned snapshots will still make price moves decorrelate. And holidays in all those countries won't ever stop disrupting the time series.

Some data issues are more fundamental still. The sample size is relatively small (often only 2 years of data are used, and sometimes fewer if mandated). But there is often no choice, really: ancient market data is likely to be of less relevance today, and newly introduced asset types can't possibly boast an ample history. Replicating the most recent vola levels in the markets reduces the effective sample size even more. Consequently, the sample is unlikely to contain reliable tail information, and it sure won't be able to even hint at the unknown unknowns—the flood, the law, the war. This sample, then, is no match for the numerous risk factors and their fearsome dimensionality curse.

Nor is market data the result of some neat, reproducible process—it is the product of this most complicated of non-linear feedback systems: man. The market data's statistical properties we try to grasp change in ways we don't comprehend. The very magnitude of market risk itself hints at some reservoir of irrationality that calls into question overly elaborate conclusions.

Market data quality, its size and dimensionality, and its intractability thus seem to suggest a humble approach: to acknowledge and accept the data imperfections and to keep unsupported, extravagant math at bay. A new model feature or enhancement should be warranted by the data.

The second most important aspect is running the model, i.e., building, operating, and maintaining an IT system that must reliably produce risk measures every day. The model choices we make have a direct impact on such a system and its costs.

A complex model feature makes an IT system more time-consuming to build, i.e., to program and test and document. It makes the computational steps more difficult to replicate for others and the results thus harder to accept, while impeding the use of common standard software that might not support the more arcane model choices.

A brittle model feature affects operation. It has more ways to break down, impairing reliability; it might be numerically less efficient, slowing down recalculations or requiring costly hardware upgrades; and it obstructs custom, ad-hoc analyses for a wide range of model end users.

Worst of all, an elaborate model feature is hard to maintain. Because its code is larger in size and thus more difficult to understand and modify, it is usually less flexible in accommodating new or changing requirements. Its black box nature furthermore fosters secluded islands of knowledge and risky dependencies on fewer and costly experts.

In the end, of course, all these considerations essentially just come down to money—they compel us to eschew complexity wherever possible and instead to aim for a system that is easy to code and reenact and trust, reliable to run, and simple to grasp and adapt. Model features should not thwart this.

Third, the model must usually be approved by the regulator. Any particularly inventive or unique model choices naturally—and rightfully—lead to disproportionate efforts in explaining and defending them. Non-standard approaches must be extensively plausibilized; model parameters setups without ready reference to canonical settings can and will be scrutinized; and even seemingly unsuspicious changes may lead to additional tests and expensive recalculations.

The main factor here is time. Any model change requires a lengthy and procedural back-and-forth; large changes may require years to get approved. As regulatory mills will grind diligently and slowly, the cast of characters will change. Original model developers might be gone and their intentions become all but forgotten. New personnel at the regulator will want to focus anew on the same, already expounded issues and demand clarifications whose new angle, very naturally and understandably, originates in the involved individuals' personal views on modeling issues. The same topic—any feature or choice or parameter—may raise its head time and again.

Simplicity, again, goes a long way in minimizing discussions, dodging follow-up efforts, and increasing model acceptance—in saving time. The path to delayed model approvals is paved with fancy mathematical intentions; we should opt for model features that take a more direct and transparent route instead.

Mathematical elegance or optimization should, in my opinion, rank last when weighing in on a particular model choice. Tune the kurtosis? Fine. Do a maximum likelihood fit of a distribution to the PnL estimates and use the fit's slightly more stable quantile? By all means. But only do it if (1) it is warranted in light of the data's limitations, (2) it does not overly affect system complexity and operations, and (3) it is transparent and defensible to a wider audience.

For each minor tweak has consequences: you will have to analyze it, discuss it, estimate its implementation costs, program it, test it, change-track it, deploy it on your various systems, run it in parallel for some time, document and explain it, present it in PowerPoint slides to an skeptical audience, defend it, debug it, maintain it, and potentially discard it if anything goes haywire. And that's before someone formally asks you to "validate" it.

Some examples seem in order:

- The vola rescaling feature makes very little assumptions on the data and is certainly warranted to them, whereas trying to get a handle on, for instance, co-kurtosis is most likely futile.
- Several intricate interpolation methods are widely recommended and thus easy to defend, and they can't be dismissed from a data limitation stance. But because they can substantially complicate handling and replication, we may often go for a simpler, linear interpolation.
- In a world of positive interest rates, should we opt for absolute or square root return types? Both methods roughly match (or fail to match exactly) the data; both have the same system complexity. But because square root returns are widely used in literature, they might just be slightly easier to sell.
- This picture changes in a world of negative interest rates. Square root returns now require a shift, which makes this choice both more complex to handle and more onerous to justify and defend. Absolute returns become the expedient choice.

In light of these considerations, let's revisit some steps and choices of the historical approach outlined previously.

Return Types We use different ways to calculate returns from a historical time series of values, depending on an asset's type. Keeping in line with conventions here does make some sense mathematically, but, more importantly, it facilitates the defense of a model, as its reviewers appreciate familiarity in such basic model choices. For strictly positive asset prices like foreign exchange rates or stock prices, we will usually choose the commonly used logarithmic or log returns $\log(v'/v)$. Their use is so common that it is rarely challenged. Their benign numerical properties (they are tractable and naturally shy of negative values) should not belie, though, that reality log-normal is not.

For interest rates we can use absolute returns $v' - v$. Consistency is one point in favor of this: log returns of bond *prices* neatly mimic absolute returns of *rates*.

An alternative is to use square root returns for time series and scenario generation, i.e., $r = \sqrt{v'} - \sqrt{v}$ and $s = (\sqrt{s_0} + r)^2$. In order to handle negative interest rates, however, they require positive shifts of the time series (and reverse ones to the scenarios):

$$\begin{aligned} r' &= \sqrt{v' + c} - \sqrt{v + c}, \\ s' &= (\sqrt{s_0 + c} + r')^2, \\ s &= s' - c. \end{aligned}$$

The shift c also requires a modicum of caution: the expression $\sqrt{s_0 + c} + r'$ should not be negative lest we introduce a distorting repel-from-zero behavior. (Shifted log returns avoid this issue naturally.)

The end results differ little from those created via absolute returns, but the shift parameter involved can lead to follow-up questions in a model review—How is the shift determined? What effects do different shifts have? Are strictly positive time series shifted as well? Absolute returns avoid the potentially time-consuming exercise of proving the immateriality of these concerns.

If, say, for reasons of continuity towards a legacy implementation, a shift can not be avoided, it is good to be aware that such shifts are not that drastic—asymptotically, large shifts will simply begin to mimic the behavior of absolute returns. A specific way to set the shift is given in Sect. 20.1.

Target Volatility We use the 20 latest historical returns, i.e., the most recent local volatility, to obtain the target volatility T of a risk factor. Do this by all means if you are very confident in your time series' data quality. However, consider a time series that becomes stale and essentially starts to stay constant (maybe due to a market data source failure). Over the course of 20 days, the target vola will gradually approach zero, possibly without raising much suspicion. The risk factor stops being measured by the VaR and effectively disappears. To avoid this and to account gracefully for at least some staleness, we can redefine the target vola as the maximum of the original, most current local vola and, say, 30% of the overall or *long-term* vola of the raw returns $\tilde{\mathbf{r}}$, effectively flooring the target volatility.

The one main drawback of this approach is that we introduce a rather arbitrary new model parameter (the 30%). Parameters like this are often in the drill-down focus of model reviewers. Still, this safety valve is probably worth its likely explanation effort. For more on such parameters and their defense, see Sect. 17.4.

Local Volatility via Decaying Weights The target volatility is fully or, if floored, mainly driven by the latest 20 returns. A single new, large return can greatly increase it and cause a jump in the VaR. After 20 days, this very return exits the

20-day window, and the target vola and VaR fall back. This is sometimes considered unwanted behavior. A workaround often proposed is to modify the way the local vola is calculated, e.g., via exponentially decaying weights, which make an extreme return fade away more gradually. (One such approach is the so-called *exponentially weighted moving average*, or *EMWA*.)

I am not in favor of such an approach. It makes this step marginally more complicated—it becomes, e.g., more tedious to exactly reenact manually in Excel, which is often very useful for model end users. It merely sugarcoats the results; I find the transparent raw results more informative. But if truly abhorred, such VaR jumps are best dealt with ex post, for example, by using the moving average of VaR values when determining capital requirements. Furthermore, the decay factor used can, as a model parameter, again solicit unwanted attention during model reviews.

Detrending A commonly used procedure we omitted is *detrending*—removing the sample average from the raw returns by subtracting it. Now, the return average is typically much smaller than the volatility and seldom requires dedicated actions. Also, our mirroring of the returns already achieves a similar result of ensuring a certain symmetry in return distributions, so we may often take a pass on a practically redundant intermediate step.

Good intentions, oversight, or modeling by the numbers, however, could well lead to its accidental implementation. This can, in certain cases, lead to unintended behavior. Consider a time series with one large positive jump and, hence, one large positive return (this might occur if you have to paste your time series together from two market data source systems). Subtracting the overall return average (positive because of our outlier) from all returns might drive the non-outliers down and off their near-zero mean; they might even all become, in our example, quite a bit negative. The possibly erroneous outlier thus unduly influences all other returns and their distribution; subsequent mirroring then effectively blows up the vola even further by adding, for good measure, reversely skewed return versions as well. Of course it's best to avoid the outlier in the first place; still, a model that is unfazed by erroneous input data, that is stable, is usually preferable.

Operations like this one—an overall normalization or "master override"—are often not innocuous. They casually and globally affect the whole input (and can thus do quite a bit of harm). They are also often suspiciously easy to implement, to test against a well-behaved data set, to OK, and to then forget. Because in technical or syntactic terms this operation cannot fail, it all too easily becomes invisible while its semantics go unchecked and remain dangerously concealed. Should the problem mentioned above then ever materialize, you most likely will spend precious time tracking the various process steps to identify the culprit.[1]

[1] Such overrides are not uncommon. If, for example, separate models yield probabilities that should sum up to 1, the probabilities can be normalized to enforce an exact match or identity to 1. If one model fails and yields, say, 230%, such a step, if thoughtlessly implemented, may cover up and obfuscate a breakdown more apparent otherwise.

Location of Local Vola Window The time series of returns exhibit varying vola levels, also called *heteroscedasticity*. A vola rescaling operator does not leverage but actively tries to destroy this property. To do this, it must estimate the local volatility of the region a return resides in.

The textbook approach for this is to use so-called GARCH (short for, you guessed it, generalized autoregressive conditional heteroscedasticity) models. Such models are mathematically sound and widely used in literature. In fact, the original filtered historical VaR simulations rely on GARCH, which makes its application easy to defend.[2] But GARCH models must be fit first—in our case, 2200 times or once for each risk factor, and repeatedly. This means an increased implementation effort and plenty of things that can go wrong. It also makes it difficult or impossible to precisely reenact model results for anyone who does not command a ready time series analysis kit. For these reasons, we propose to use a poor man's version of a vola estimate, the plain standard deviation.

When using the standard deviation, it is tempting to stick with as many GARCH conventions as possible to minimize any perception of deviation from an ingrained, established method. Since GARCH is essentially a regression-based approach, it relies on returns up to, but not including, the day for which a vola is estimated. In that line, the standard deviation of those same, preceding returns could be used for estimating the local vola of a return about to be rescaled. This is not necessary, however, as the standard deviation is simply not bound by regression limitations. In fact, if you were tasked to come up with a risk factor's volatility for some January 15, you would most likely take the standard deviation of that month's returns, never even considering to drop the return of the 15th itself from that estimate.

If we associate, as we have to, our target volatility for tomorrow with the most recent local vola, which clearly precedes tomorrow, we could become inclined to align past vola estimation windows and the corresponding raw return day likewise, in a strictly preceding manner. This might appear to be conceptually more elegant or consistent.

Alas, such elegance would come at a cost—the vola estimator's numerical behavior after periods of stale data. Consider a constant series of historical price quotes and a corresponding period of zero returns. This may arise due to a problem with a market data source system, but this can happen naturally as well, e.g., when certain markets are closed during bank holidays over a fortuitous regimen of feast days. When rescaling, the first non-zero return after such a stale period would have to be divided by an essentially arbitrary small local vola value if a preceding window is used, causing the rescaled return to basically explode. Of course, one could floor the local vola somehow or impose a cap on the rescaled return, yet this would introduce a new and undesired ad-hoc parameter.

If, however, a return influences its own local vola estimate (like in the proposed approach), that vola is ensured to be at least somewhat positive and to thus have

[2]Just as nobody got ever fired for buying IBM, no vola model was ever rejected for relying on GARCH.

a much lower blow-up potential. To be sure, both approaches are of course off with regard to the new, correct vola level—we know that only after a few days of non-stale data. And after roughly 20 days, both will converge to the newly established vola level. It's just that the proposed approach does so in a less disruptive manner and overestimates the rescaling factor required by less (it is still conservative in many conceivable real-world setups of such a departure from staleness). The numerical stability in face of bad data outweighs the very slight inelegance in design. In fact, vola rescaling would work just as well with centered windows (in that case, you'd just have to handle the edges of your time series, which is possible in any number of ways).

Technical Floor for Local Vola A special case still necessitates mention and treatment: the rescaling of zero returns within stale periods of zero local vola. The rescaled return, for lack of any usable information, will still have to be zero, but the raw computation would break down. Flooring the local vola at an unreachably low level (say, 10^{-12}) at least avoids the dreaded division-by-zero error.[3]

Exact Rescaling Once the volatility rescaling is done, the rescaled returns will have a vola T' that is similar, but not exactly identical, to the desired target vola T. We can easily rescale them all again by a single constant T/T' to make them precisely match the target vola.

Such a final rescaling is, like detrending above, a master override; if applied without support checks, it may hide some ugly data. Consider a long period of stale data and zero returns. Since zero returns scale to zero, the usual rescaling will achieve a lower volatility than the target vola; if monitored, this becomes noticeable and can be fixed. If, instead, the final exact rescaling is blindly applied on top of the standard one, the non-zero returns will be increased more aggressively (as the zero ones will remain unaltered), and while the target vola is reached by construction, those returns alone will carry the correlation information. This is quite subtle a model distortion, which might thus go unnoticed. So exact rescaling does not rid us from checking the basic vola rescaling's results.

Nevertheless, enforcing the exact target vola is most likely still worth the small effort (separate checks for staleness are of course strongly advised). It simply does away, in regular circumstances, with those random cases where rescaled returns have a slightly lower vola than expected, a risk-underestimating characteristic that would, despite its usually very small deviation, welcome the regulator's comprehensible scrutiny. To prove, at the almost inevitable request, that the impact of the (original version's) target vola mismatch is insignificant would require implementing exact

[3]The lowest naturally occurring local volas "in the wild" remain unaffected by this floor. The standard deviation of 19 zero returns combined with a return of one one-hundredth of one basis point is—with about 10^{-7}—already much larger than this purely technical floor.

rescaling for comparison purposes anyway. We might as well nip this issue in the bud.

Artificial Kurtosis Once the rescaled returns $\mathbf{r}$ are created, we could further tweak them. One such adjustment is the artificial increase of their *kurtosis*—a measure of how extreme the values at the fringes or *tails* are when compared to extremeness levels expected from normal distributions.

To make the tails "heavier," i.e., to increase the kurtosis, we can scale up some of the returns, say, 10% of them, by a constant factor $s > 1$. If we scale the same entries in each risk factor's return vector, i.e., if we scale whole return column vectors $\mathbf{R}$, we don't affect the correlation structure too much. Larger values of s lead to larger kurtosis and to more extreme VaR values.

This feature is possibly best used if the returns are obtained with the Monte Carlo modification (see the upcoming Chap. 10), as those returns exhibit a normal distribution and thus no kurtosis. While crude and best thought of as a safety valve, it is a simple way to ensure some non-normality, which might be a regulatory requirement.

The main problems lie in defending the chosen target kurtosis (see Sect. 17.4) and—worse—in determining one in the first place (see Sect. 17.6).

Scenario Drift When not using absolute returns, two scenarios s^+ and s^- based on a scenario s and mirrored returns r and $-r$ do not result in exactly symmetric scenarios: $s \neq (s^+ + s^-)/2$ (try it, for example, with log returns). This makes the average of the generated scenarios deviate or *drift* from the base scenario value, which is often considered undesirable as it breaks the symmetry of profits and losses. One could correct for this effect; however, over small time periods like in our case, its impact is negligible. For example, in a Monte Carlo setup (which we will cover soon), tens of thousands of scenarios would be required to even notice this effect, especially as the Monte Carlo error involved is more pronounced for the scenario VaR than for the scenario average. (For time horizons much longer than 1 day, though, certain market-implied target scenario averages different from zero may be considered.)

Simplifications The proposed steps are a typical minimal setup for 2 years of data and the 1%-VaR. If you are comfortable with using a larger market data window (or you have to use it via regulatory requirements), or if you only need, say, the 10%-VaR, you might omit the mirroring step. For certain tasks, VaR models might be required with equally weighted returns; sometimes, daily backtesting or a fast model reaction to increasing vola levels is not desired—in such cases the rescaling of returns is superfluous. Each such simplification has immediate benefits for the implementation, testing, documentation, etc.

An alternative to mirroring for generating scenario returns $\mathbf{r}$ from the rescaled ones $\bar{\mathbf{r}}$ warrants its own, upcoming chapter.

A Monte Carlo Modification 10

Our VaR model typically uses 2 years of data or 500 returns, and it generates, via mirroring, twice that number of scenario returns. Yet we might be confronted with smaller input samples, be required to generate more scenarios, or both:

- For newly introduced asset types, the usual 2 years of market data might simply not be available.
- The regulator might prescribe the use of a smaller time window, e.g., 1 year or 250 days.
- Sometimes we'd like to analyze multi-day returns instead of daily ones; if we want them to be non-overlapping, the effective number of available returns shrinks fast.
- For various reasons, more extreme VaR levels could be desirable (e.g., a 0.1%-VaR), or we might otherwise require a larger number of scenarios to increase the density of their event coverage.

Mirroring, which only doubles the number of effective scenarios, might not be sufficient in such cases. An alternative approach is one that corresponds to random averaging, and it is called *Monte Carlo* approach. It generates arbitrary many (normally distributed) scenario returns $\mathbf{r}$ from a given sample while preserving its correlation structure in a numerically stable way.

To illustrate how it works, we start off from a single row vector of rescaled returns $\bar{\mathbf{r}}$, which we will, to be more in line with the terminology in Appendix A, call $\mathbf{x} = (x_1, x_2, \ldots, x_{500})$. If we wanted to create one new individual normal return with this sample's properties, we would usually just estimate the mean and standard deviation from the sample and generate a corresponding random normal.

M. Auer, *Hands-On Value-at-Risk and Expected Shortfall*, Management for Professionals, https://doi.org/10.1007/978-3-319-72320-4_10

Alternatively, we could choose the more laborious route of drawing samples from 500 (independent) standard normals N_i and computing

$$X' = \frac{1}{\sqrt{500}} \sum_{i=1}^{500} x_i N_i.$$

(Note: we will here assume a mean of zero to avoid having to juggle with too many terms; this is also largely justified numerically in this setup.)

As a sum of normals, this expression is also normally distributed. Since each normal has zero expectation, its expected value is zero. Its variance, by construction, is the one implied by the sample:

$$\mathbb{V}\mathrm{ar}[X'] = \frac{1}{500} \sum_{i=1}^{500} x_i^2 \, \mathbb{V}\mathrm{ar}[N_i] = \frac{1}{500} \sum_{i=1}^{500} x_i^2 \times 1.$$

If we recalculate this expression for a full new set of standard normals, we can generate a second return, and so on. But why bother with creating 500 normals for just one single new return? The reason is that this scales well to additional dimensions. If we throw a second vector of rescaled returns $\mathbf{y}$ in the mix, we can calculate:

$$\begin{pmatrix} X' \\ Y' \end{pmatrix} = \frac{1}{\sqrt{500}} \sum_{i=1}^{500} \begin{pmatrix} x_i \\ y_i \end{pmatrix} N_i.$$

The components X' and Y' still preserve the mean and variance of the samples $\mathbf{x}$ and $\mathbf{y}$, respectively. But beyond that, they also preserve the sample pair's covariance:

$$\begin{aligned} \mathbb{C}\mathrm{ov}[X', Y'] &= \mathbb{E}[X'Y'] \\ &= \mathbb{E}\left[\left(\frac{1}{\sqrt{500}} \sum x_i N_i\right)\left(\frac{1}{\sqrt{500}} \sum y_i N_i\right)\right] \\ &= \frac{1}{500}\, \mathbb{E}\left[\sum_{i=j} x_i y_i N_i^2 + \sum_{i \neq j} x_i y_j N_i N_j\right] \\ &= \frac{1}{500} \sum_i x_i y_i. \end{aligned}$$

(The final step uses $\mathbb{E}[N^2] = 1$ and $\mathbb{E}[N_1 N_2] = 0$ for uncorrelated standard normals. All steps should be traceable via the statistics crash course in Appendix A.)

We can straightaway extend this from 2 dimensions to the 2200 dimensions of our risk factors. Using 500 normals, we thus generate one column vector $\mathbf{R}$ of 2200 returns, in a way that preserves the (co)variances.

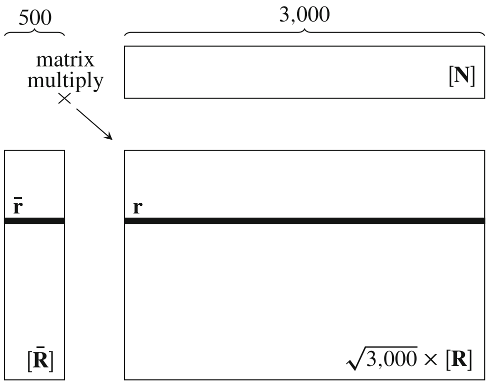

Fig. 10.1 Returns via random averaging

If we repeatedly recalculate the vector expression with new sets of standard normals, we can generate arbitrary many return vectors **R** and thus arbitrary many scenarios. Figure 10.1 illustrates how this can be expressed as a matrix-matrix multiplication. With [**N**] a 500 × 3000 matrix of standard normals, the graph shows how 3000 return vectors are generated from the 500 rescaled ones—in short: $[\mathbf{R}] = \frac{1}{\sqrt{3000}}[\bar{\mathbf{R}}]\,[\mathbf{N}]$.

The computational costs are negligible. The main drawbacks of this approach are twofold: it enforces the generated returns to be normally distributed, and it introduces an additional Monte Carlo error on top of the time series-driven noise. To deal with the imposed normality, this approach can be combined with the artificial kurtosis feature described in Chap. 9. Benign negligence (recommended), increasing the number of return scenarios that are generated, or using standard variance reduction techniques like antithetic variates can all help reduce the Monte Carlo noise.

By the way, the textbook approach for generating random normals with a given correlation structure is to *Cholesky-factorize* a covariance matrix and to use the resulting triangular matrix to generate the desired normals. Yet numerical errors in the floating-point calculations or additional, too restrictive assumptions on the matrix properties—as in NumPy's `cholesky` function—can cause standard algorithms to fail, which would have to be addressed with non-trivial workarounds. These issues are especially pronounced if the number of dimensions is larger than the sample size, like in our case. We therefore avoid that approach altogether.

Using this Monte Carlo approach omits the mirroring step. Mirroring has, besides increasing the sample size, the useful feature of making any return also appear as its negative sibling—so a large, PnL-increasing return that would itself not influence the VaR much is also bound to appear, mirrored, in a PnL-decreasing variant. This is prudent, as we assume return directions to be essentially random and don't want to miss out on large ones just because our positions' direction happens to be a fortunate one. In addition, mirroring also forces the returns to have zero mean, which can be desirable (even if the original mean is likely to be so close to zero as to practically vanish anyway).

Luckily, random averaging already has an implicit mirroring effect built in: an individual rescaled return $\bar{r}$, as part of the random average of several returns, is scaled by a different random number in each scenario, maybe $+1.01$ in one scenario, -1.03 in the next, etc. Overall, this achieves a near-perfect mirroring of the input returns as well.

A final note on the names of the presented VaR methods. We used the terms historical, analytical, and—here—Monte Carlo approach. Each method, however, clearly relies mostly on (recent) historical data. The conceptual difference between them can be considered smaller than their names imply.

11 Support Measures

We have selectively presented a few risk measures in the preceding chapters that, in our experience, cover many relevant aspects and tasks in a real-world market risk setup. We propose to mainly use the volatility-rescaled historical VaR$[\Omega]$ for daily risk management. It is especially well-suited to capturing "tomorrow's PnL," as it reacts fast to changes in volatility levels. The concurrent use of the sensitivity-based analytical VaR$(\mathbf{s}^{\Omega})$ serves as a sanity check and provides an additive decomposition to VaR-contributions of the risk factors, which is a handy analysis tool because it appropriately weighs risk factors by both their sensitivity and volatility. Finally, the most helpful measure we take away from the expected shortfall world is the position-wise conditional expected shortfall cES$[\alpha|\Omega]$, which provides a useful complementary breakdown of risk to positions.

The main difficulty in analyzing VaR figures and thus the need for additional support measures arise because the VaR is generally not additive:

$$\mathrm{VaR}[\alpha + \beta] \neq \mathrm{VaR}[\alpha] + \mathrm{VaR}[\beta].$$

More background on this and actual use cases of our measures will be given in Part II. Before that, the current chapter will mention additional helper measures you should be able to reference. Some of them are useful, others less so; either way, certain ones might be mandated by the regulator.

First, let's address an apparent gap in our measures presented earlier. As mentioned in Chap. 8, the analytical VaR approach immediately translates to an analytical ES approach (we only need to tweak the final multiplier of the standard deviation ever so slightly). But can, reversely, the concept of the conditional expected shortfall be translated to the VaR as well? Can we find an additive decomposition of the VaR to positions?

The answer is we sure can, and easily too, but we should nevertheless steer well clear of it. First, how would we do it? Well, in analogy to the cES, we could determine the index of the portfolio PnL vector's VaR scenario; the corresponding

© Springer International Publishing AG 2018
M. Auer, *Hands-On Value-at-Risk and Expected Shortfall*, Management for Professionals, https://doi.org/10.1007/978-3-319-72320-4_11

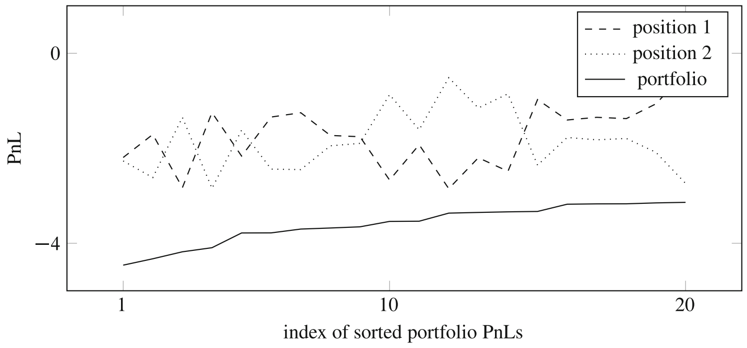

Fig. 11.1 Individual and combined PnLs near the VaR scenario

entries in the positions' PnL vectors could then be considered the individual "cVaR" values for each position (of course summing up to the portfolio PnL in that scenario, i.e., the VaR). We did the exact same for the cES—we just remembered all the original indices of the 25 most negative portfolio PnLs.

Why should we avoid doing this? The short answer is that this measure would be far too unstable, essentially yielding almost random numbers (the averaging involved in the cES calculation, on the other hand, provides for stability). One way to illustrate this unreliability of a cVaR is to artificially create, e.g., in Excel, 1000 pairs of (random normal) returns/PnLs of uncorrelated positions, along with their sums, the hypothetical portfolio PnLs. Figure 11.1 depicts—for a random example—the subset of those 20 return pairs with the largest portfolio losses. The tenth return pair from the left (call it pair A) can lay claim to represent our VaR and proclaim two cVaR values.

A first hint at the fickleness of all this is that the return pairs neighboring the VaR one (e.g., pair B, the 11th from the left) exhibit quite different cVaR candidates. Now assume the return sets to change slightly (after all, as time progresses, new returns materialize and old ones disappear). Say, the fifth most negative PnL above is dropped because its corresponding return pair vanishes. Suddenly, pair B sits at the tenth location and provides apparently very different cVaR values.[1] Workarounds have been proposed to modify such a cVaR and to make it more stable, but they essentially perform some sort of averaging over several entries around the portfolio's VaR scenario index and therefore basically converge to the cES behavior, with the added baggage of custom heuristics.

Keep in mind that the desire for an additive decomposition of the VaR to positions is comprehensible. It would allow us, for example, to cleanly assign parts of the risk as described by the overall portfolio VaR to positions and sub-portfolios. Along with this, the risk costs arising from the capital requirements could be allocated to units

[1]This issue gets even more pronounced for negatively correlated positions, where more return pairs yield close VaR contenders.

and departments and desks and individuals. A cVaR is too fickle, as we have seen, but even the cES is not ideally suited: while additive and stable, its values can be both positive or negative, offering no obvious "weight" interpretation akin to, e.g., some always non-negative probabilities. Such same-sign additive decompositions are, alas, not available.[2]

Let's now examine some other helper measures.

Individual VaR We are mostly interested in the VaR of a portfolio of positions, i.e., some VaR$[\Omega]$. Of course we could compute each individual position's VaR as well, as in VaR$[\alpha]$, maybe with the aim to detect outlier positions or track down suspicious changes or jumps in the overall VaR at the position level.

In practice, such an individual VaR is not that useful. A position with a conspicuously extreme individual VaR might, in the end, not affect the portfolio VaR by much (it depends on how the position is correlated to the remaining portfolio). As another example, two positions that hedge each other could signal two extreme individual VaR values but actually have, combined, no effect on the portfolio VaR at all. If such hedges were to involve non-linear positions, we could potentially get one extreme and one moderate individual VaR value, muddying the analysis waters further (those positions' combined influence on the VaR is, again, zero).

In some instances, sub-portfolio VaRs can of course be helpful, as they help restrict the search space to position subsets when VaR changes need to be pinned down. Calculating and storing each position's VaR, however, can usually be avoided.

Incremental VaR When adding a new position α' to an existing portfolio $\Omega = \alpha + \beta + \dots$, the portfolio VaR changes. By how much mainly depends on the new position's size and its correlation to the portfolio's PnL behavior. The resulting new portfolio VaR can range from 0 (if the deal mirrors the portfolio exactly) to arbitrarily negative values (if the deal is dominant). Naturally, we'd like to know in advance how the VaR (and thus our costs, i.e., capital requirements) would change if we entered a new position. Computing such an impact is often referred to as performing a *pre-deal inquiry*.

Plainly, we just compute the new VaR and relate it to the current one. The new position's impact is called its *incremental* VaR:

$$\mathrm{iVaR}[\alpha'|\Omega] = \mathrm{VaR}[\Omega + \alpha'] - \mathrm{VaR}[\Omega].$$

[2]The concepts involved here are more deeply connected, as one could consider each position to represent a separate asset. While we tried to delineate those views on risk decomposition, they really represent two sides of the same coin. The terms marginal VaR and component VaR are commonly used in this context. The names used here I chose by sympathy and memorability; marginal VaR often also refers to what we called incremental VaR, while component VaR sounds a bit like poet laureate or Astronomer Royal.

We can do the same with a portfolio's current positions. For a deal α already contained in our portfolio, we can determine the VaR impact of its *removal* from the portfolio[3]:

$$\mathrm{VaR}[\Omega - \alpha] - \mathrm{VaR}[\Omega].$$

This corresponds to the incremental VaR of the deal's hedge, $-\alpha$, which compensates for or cancels the original position's impact:

$$\mathrm{iVaR}[-\alpha|\Omega] = \mathrm{VaR}[\Omega - \alpha] - \mathrm{VaR}[\Omega].$$

This expression should help drive home one particular point. When calculating the incremental VaR, we usually have already calculated the portfolio VaR and thus have at our disposal the PnL vectors of all positions and of the portfolio. Determining the VaR impact is then cheap:

- When adding a new position, we only have to compute its PnL vector $\Delta\mathbf{p}^{\alpha'}$ and *add* it to the known portfolio PnL vector $\Delta\mathbf{p}^{\Omega}$ before the subsequent sort and lookup steps.
- When removing an existing position, we *subtract* the known $\Delta\mathbf{p}^{\alpha}$ from the known $\Delta\mathbf{p}^{\Omega}$.

For existing positions, this measure has similar drawbacks as the individual, position-level VaR.

Partial VaR We usually generate scenarios on all risk factors. We can reduce this scope to subsets of risk-factors, e.g., to only foreign exchange (FX) or to only interest rate (IR) risk factors, which yields *partial* VaRs denoted as $\mathrm{VaR}^{\mathrm{FX}}$ or $\mathrm{VaR}^{\mathrm{IR}}$. This is done by creating new, distinct scenario sets where all but those risk factors we're interested in are kept constant.

Partial VaR figures facilitate locating possible sources of overall VaR changes. A VaR jump in only one of the tracked partial VaRs expeditiously narrows down the set of positions or risk factors we have to examine further. All major risk factor classes should therefore be tracked this way. Theoretically, we could break this down to individual risk factors, but this might become computationally too expensive.

Note that partial VaRs also do not add up to the overall portfolio VaR, the same way individual VaRs don't. Chapter 12 will illustrate this further.

There is a shortcut for actually implementing partial VaRs that avoids creating separate scenario sets with (somewhat redundant) constant rows. The pricing step can simply rely on the original scenarios and, before performing the computation, force the appropriate scenarios to be constant on the fly (see Sect. 19.5).

[3]We must refrain from dubbing this excremental VaR.

Synthetic Marginals Computing the partial VaR of a single risk factor (e.g., $\text{VaR}^{\text{IR-USD-Y10}}$) would allow us to track the model performance with respect to just that one risk factor and to its individual or *marginal* distribution. This would require performing a whole VaR calculation over a million positions 2200 times over, which is often unfeasible. A quicker alternative is to create artificial or *synthetic* test positions that are only sensitive to individual risk factors or small sets thereof.

To keep the set of tracked risk factors small and manageable, we can use the risk factors' VaR-contributions to determine the main and thus most interesting risk drivers and only set up and track synthetic positions for them.

Analyzing the behavior of those synthetic portfolios can then either show that that model performs well with respect to major risk factors or help expose problematic risk factors that might otherwise remain hidden in the joint distribution.

Stressed VaR Our VaR setup relates the recent two-year period of market activity to tomorrow's PnL behavior. Alternatively, one might ask which historic precedent of a previously observed market period would, now and for our current positions, indicate a high degree of risk—after all, such past periods might conceivably occur in similar form again.

To answer this, we need to find the historical period[4] of returns that projects the most extreme VaR for our current positions. So for each past day, we take its corresponding return window, create new scenarios based on it and on today's market scenario $\mathbf{S}_0$, and compute a VaR. One such return window will yield the most extreme or *stressed* VaR.

Finding such a worst-case past period is computationally very expensive, as it entails running a full VaR evaluation for each day in our history of thousands of days. This step is therefore typically only performed once a year in a separate calibration exercise.[5] It should also be explicitly triggered whenever the characteristics of our portfolio composition change drastically (e.g., when a new trading strategy is put in place or when different risk factors start to dominate the portfolio's risk). The hereby settled stress period returns can then be used each day anew to compute the current portfolio's stressed VaR, off of the current market scenario. (Note that for the stressed VaR the volatility rescaling step is omitted, for it would effectively mean selecting the worst-case 20-day return period.)

A main reason for using this measure is simply that you might have to, as regulators increasingly rely on it. As a mathematical instrument, though, it is not very elegant. The stressed VaR has no direct relation to PnLs that are actually observed, and it is thus nigh impossible to plausibilize (except that it should exceed the VaR in all but rare instances). And its warning-signaling power may

[4]The regulator prescribes the window size to be used for this purpose; one-year periods are typically used for the stressed VaR.

[5]The fast analytical approach can provide valuable support for speeding up this procedure. It can, for example, run a first tentative selection, thus limiting the number of full VaR evaluations required.

be overstated: benign market conditions, like happy families, are all alike; every market crisis is probably messed up in its own way.

The individual, incremental, partial, and stressed expected shortfall are computed along the very same lines as their VaR cousins.

Part II

Operations

12 Properties of VaR

The VaR is an altogether relatively intuitive abbreviation of the two-dimensional concept of risk, but one characteristic in particular might not be self-evident at first sight. Before we turn to that, let's address a few of the basic properties first:

- The VaR is usually negative except in rare instances; we can omit those special cases in the following discussion. If you encounter positions with a positive VaR, either check and fix your calculation or start investing in them right now instead of reading on.
- Doubling down on a position will double its risk as reported by the VaR:

  $$\mathrm{VaR}[\alpha + \alpha] = \mathrm{VaR}[2\alpha] = 2\,\mathrm{VaR}[\alpha].$$

  You can verify this relation in both the historical and the analytical VaR setup. In the former, we add a PnL vector to itself and hence also double the PnL in the sum vector's VaR entry; in the latter, the new position will have twice the original sensitivities. (The relation also holds true for any positive constant other than 2, of course.)
- A position α and its hedge $-\alpha$ have opposite PnL values. As mentioned already,

  $$\Delta\mathbf{p}^{-\alpha} = -\Delta\mathbf{p}^{\alpha}.$$

  Adding $\Delta\mathbf{p}^{\alpha}$ and $\Delta\mathbf{p}^{-\alpha}$ thus yields a zero vector $\Delta\mathbf{p}^{\alpha-\alpha} = (0, 0, \ldots, 0)$, and we obtain what we intuitively expect:

  $$\mathrm{VaR}[\alpha + (-\alpha)] = \mathrm{VaR}[\alpha - \alpha] = 0.$$

M. Auer, *Hands-On Value-at-Risk and Expected Shortfall*, Management for Professionals, https://doi.org/10.1007/978-3-319-72320-4_12

- Because both α and $-\alpha$ do have a (negative) VaR, we can immediately infer that, except in special cases, the VaR is not additive:

$$\text{VaR}[\alpha + \beta] \neq \text{VaR}[\alpha] + \text{VaR}[\beta].$$

- What about the specific VaR$[-\alpha]$? You might encounter simple linear hedges and observe that, for those, apparently VaR$[\alpha] \approx$ VaR$[-\alpha]$. For non-linear ones like the call options described in Chap. 6, this is not the case in general. Just imagine a position α with some slightly negative PnLs and some massively positive ones; its hedge $-\alpha$ necessarily has, in turn, some slightly positive PnLs and some massively negative ones. Their VaRs are driven by the negative PnLs in both cases, and their level can thus differ arbitrarily:

$$\text{VaR}[\alpha] \neq \text{VaR}[-\alpha].$$

(Lastly, convince yourself that VaR$[-\alpha] \neq -$ VaR$[\alpha]$.)

How do combinations of different positions behave under our measure of risk? We start off with a common case: two positions with PnLs that are normally distributed and behave independently of each other.

The normality ensures that our VaR is simply a scaled standard deviation, so we have VaR$[\alpha] = -2.33.. \times$ std$[\alpha]$ and VaR$[\beta] = -2.33.. \times$ std$[\beta]$. Their portfolio VaR will of course be VaR$[\alpha + \beta] = -2.33.. \times$ std$[\alpha + \beta]$.

We recall that in such a setup the involved standard deviations are related:

$$\text{std}[\alpha + \beta] = \sqrt{\text{std}[\alpha]^2 + \text{std}[\beta]^2}.$$

It becomes clear, via Pythagoras, that

$$\text{std}[\alpha + \beta] \leqslant \text{std}[\alpha] + \text{std}[\beta],$$

and consequently, aware of the negative sign of our constant,

$$|\,\text{VaR}[\alpha + \beta]| \leqslant |\,\text{VaR}[\alpha]| + |\,\text{VaR}[\beta]|. \tag{12.1}$$

This reflects the concept that the risk of two combined positions is lower than the sum of the individual risks. This is called *diversification* or *portfolio effect*, also paraphrased as “not putting all your eggs in one basket.”[1]

Before we proceed, we briefly square the initial examples with the simplest setups that are not independent. When we combine two identical positions (which

[1] An even simpler example might be of use. Assume to own 2 units of some stock that can go up or down with the same probability; you will lose money 50% of the time. Investing instead in 1 unit of two different, independent stocks each will have you lose money only 25% of the time.

clearly are not independent), the combined standard deviation is, as we would expect,

$$\begin{aligned}
\operatorname{std}[\alpha+\alpha] &= \sqrt{\operatorname{std}[\alpha]^2+\operatorname{std}[\alpha]^2+2\,\mathbb{C}\mathrm{ov}[\alpha,\alpha]} \\
&= \sqrt{\operatorname{std}[\alpha]^2+\operatorname{std}[\alpha]^2+2\,\mathbb{V}\mathrm{ar}[\alpha]} \\
&= \sqrt{\operatorname{std}[\alpha]^2+\operatorname{std}[\alpha]^2+2\operatorname{std}[\alpha]^2} \\
&= 2\operatorname{std}[\alpha].
\end{aligned}$$

When we combine a position with its hedge (also not independent), we get

$$\begin{aligned}
\operatorname{std}[\alpha-\alpha] &= \sqrt{\operatorname{std}[\alpha]^2+\operatorname{std}[-\alpha]^2+2\,\mathbb{C}\mathrm{ov}[\alpha,-\alpha]} \\
&= \sqrt{\operatorname{std}[\alpha]^2+\operatorname{std}[\alpha]^2-2\,\mathbb{C}\mathrm{ov}[\alpha,\alpha]} \\
&= \sqrt{\operatorname{std}[\alpha]^2+\operatorname{std}[\alpha]^2-2\operatorname{std}[\alpha]^2} \\
&= 0.
\end{aligned}$$

In both cases, the portfolio behaves as promised above. If our positions move in tandem, the risk is additive. With opposing positions, the risk not only becomes smaller but in fact zero. Most often, positions diversify the risk and thus act between those two extremes. Either way, the relation (12.1) seems to hold.

Until, of course, it doesn't. To find an example of how the VaR might break relation (12.1), we must leave the benign world of normally distributed PnLs. The historical approach, where the VaR is the 10th most negative PnL, yields an example where the combined risk appears to be larger than the sum of the individual risks—where the VaR is said to be *not sub-additive*:

$$|\operatorname{VaR}[\alpha+\beta]| > |\operatorname{VaR}[\alpha]| + |\operatorname{VaR}[\beta]|.$$

Consider a PnL vector $\Delta\mathbf{p}^{\alpha}$. It will contain negative and positive PnLs scattered about randomly. Somewhere we'll encounter the tenth most negative entry, the $\operatorname{VaR}[\alpha]$, say, at position 714. The vector will also contain a most negative value somewhere, say, at position 339, a second most negative value, maybe at position 107,.. well, you get the picture. Now imagine the nine most negative values to be *very* negative, make it $-\infty$. In fact, no matter how negative you imagine them to be and how risky a position you thus concoct in your mind, the $\operatorname{VaR}[\alpha]$ remains unconcerned and keeps signaling the same risk.

Now on to another position β. With the same thought experiment, we can assume its PnL vector $\Delta\mathbf{p}^{\beta}$ to contain a reasonable $\operatorname{VaR}[\beta]$ somewhere, as well as nine further super-negative entries at positions 216, 17, 903, etc.—again without impacting the position's own individual VaR.

To obtain the portfolio's VaR[$\alpha+\beta$], we need to add the individual PnL vectors as in $\Delta\mathbf{p}^{\Omega} = \Delta\mathbf{p}^{\alpha} + \Delta\mathbf{p}^{\beta}$ and get this sum's tenth most negative entry. Our searched-for example now readily looms in plain sight: up to 18 entries will be arbitrarily negative ($-\infty$), and thus the tenth most negative one, the portfolio VaR, will be too. The combined positions' risk appears to be larger than the sum on the individual risks. The VaR, here, suggests we actually put all our eggs in one basket rather than in two.[2]

This is considered, by many, to be a highly undesirable property of a risk measure, and alternative measures—guaranteed to be sub-additive—are therefore often proposed instead. Yet first, instances where the VaR behaves this way are very rare in the real world (I personally encountered them twice over the course of 8 years). And while being sub-additive is desirable, alternative measures may exhibit unwanted traits of their own, as we will soon see.

So it is good to be aware of this issue, for in instances of violated sub-additivity, you might have to do some explaining lest your model results be discredited. For the overwhelming part, you will find assumption (12.1) to hold up well.

[2]The same effect can crop up with partial VaRs.

13 Properties of ES

The expected shortfall or ES shares the basic properties of the VaR given in the previous chapter: it is negative; it scales for positive multiples of positions; it vanishes for hedges; it is not additive; etc. For normally distributed PnLs, it is again a mere multiple of the standard deviation, and the scaling factor of the 2.5%-ES is, with 2.34.., very close to the 2.33.. of the 1%-VaR.

One nice-to-have characteristic of the ES is that it is always sub-additive, unlike the VaR. Consider two positions' PnL vectors, whose 25 worst losses determine, via their average, the positions' ES. It turns out that the portfolio ES can't be "worse" than the sum of the individual ES values:

$$|\,\mathrm{ES}[\alpha + \beta]| \leqslant |\,\mathrm{ES}[\alpha]| + |\,\mathrm{ES}[\beta]|.$$

There are various ways to formally prove this.[1] We persuade ourselves of this statement's validity with a thought experiment. Now, to create a counterexample and thereby *breach* the inequality above, the portfolio's $\mathrm{ES}[\alpha+\beta]$ would have to be very negative—we can ask ourselves: how would the two positions' PnL vectors have to be aligned in order to create a maximally negative portfolio ES? (The individual ES values are not influenced by any such alignment). Well, for the portfolio ES to become very negative, it would have to pick up as many very negative PnLs from α and β as possible; ideally, it would pick up all their most negative PnLs. This can only be the case if the worst α-scenarios coincide with, i.e., have the same vector position as, the worst β-scenarios. Yet we see that even in this special case, the portfolio's ES could at best be identical to the sum of the positions' ES-values. In all other cases, it can't even reach that equality.

[1]See, for example, "Seven Proofs for the Subadditivity of Expected Shortfall" at https://people.math.ethz.ch/~embrecht/ftp/Seven_Proofs.pdf. Also note that the idea here remains true even in case of freak positive ES values; we would just have to phrase it more awkwardly.

Another seemingly good feature of the ES is that it takes into account the PnL distribution's tail by factoring in all the most extreme negative PnL values, while the VaR de facto ignores those potentially risk-revealing, most severe losses.

Mostly because of these two reasons, the ES has gained prominence in recent years. In practice, the choice of measure might not be up to you, as it can be prescribed by the regulator or by precedent. In any case it is good to have both measures at the ready, if for nothing else than as a sanity check.

Yet one should be very careful not to write off the VaR and not to overly exalt the ES:

- The VaR indeed fails to account for the outer fringe of the PnLs' tail, unlike the ES. However, the data itself does not necessarily allow for a full or reliable description of tail behavior—the sample size is small, and the historical time window might not encompass many revealing extreme or tail cases.

  Now, while the VaR does not even pretend to capture that tail behavior, the ES might effectively often just capture an incomplete or distorted one, possibly providing a false sense of risk coverage.

  Just as reporting a "4 out of 7" survey result as "57.14%" could be considered slightly obfuscating and misleading, using ES implies confidence in a knowledge about a model's tail behavior that more often than not is unwarranted. The VaR, on the other hand, is modest in this regard.
- Many model quality issues arise from data quality or, better, from missing data quality. Individual outliers in the input data (like an erroneous return) or bugs in pricing functions may end up producing erroneous extreme PnLs at times. The VaR is benign in such situations because the most extreme PnLs barely affect it. The same cannot be said for the ES, which can be arbitrarily distorted by individual outliers.

  It goes without saying that the ES is hardly to blame for bad input data. Of course, data quality should be carefully and continuously monitored and pricing implementations diligently tested in order to avoid such freak instances. Yet *if* an issue still sneaks through, which might not be entirely avoidable, the ES might turn out to be a bit less forgiving than the VaR.
- The VaR is straightforward in one sense especially: we can easily test this measure against the actual PnLs. We expect, on average, to observe a PnL breaching the previous day's VaR estimate about once every hundred days, and testing this essentially just boils down to counting (more on this procedure in Chap. 15).

  It's trickier to test the ES. Intuitively, only the small number of days where the VaR is breached can be analyzed further with respect to the breach's (average) size. There are various approaches and workarounds addressing this limitation, but they lack the transparency of the VaR-related tests.
- Finally, the VaR is indeed not sub-additive. In practice, however, this quirk manifests itself relatively rarely. The main drawback here is the effort you have to put in if model result consumers stumble upon such occurrences and start questioning the numbers.

So while the ES is indeed smartly sub-additive, it is more pretentious in its assertions, resentful with respect to outliers, and more elusive to validate. In a bit more earnestness, the ES is a fine measure with no agenda of its own. We should probably just try not to infuse it with an expressiveness unsupported by the underlying data.

Of way more interest to us is the conditional expected shortfall or cES. (Recall that the positions' cES values add up to the portfolio ES and can be either negative or positive.)

Unlike the individual ES and the individual VaR, the cES has a useful property with regard to hedges. As it is simple to verify, a position's hedge has the negative cES of its counterpart:

$$\mathrm{cES}[-\alpha|\Omega] = -\,\mathrm{cES}[\alpha|\Omega].$$

This can be used to visualize a portfolio's hedging disposition. The scatter plot in Fig. 13.1, for instance, gives an overview of each position's impact on the ES (and thus, by proxy, on the closely related VaR). At one glance, we can identify distinct clusters of positions:

- Positions on the right-hand side have a large PnL standard deviation; they are plausible candidates for being influential.
- Positions far below the x-axis have a large negative cES; they drive the overall portfolio ES.
- Position pairs that appear mirrored along the x-axis represent hedges; they cancel each other out and, together, do not impact the ES.
- Positions on or close to the x-axis also hardly impact the ES, even if they appear on the far right; they are correlated with the rest of the portfolio in a way so as to minimize the overall ES impact.
- Finally, positions in the lower-left part, i.e., with a small standard deviation but a relatively massive negative cES, possibly feature somewhat extreme PnL values in the tails. How come? Well, in general, the worst-case cES is the individual ES.[2] Thus, the cES of normally distributed PnLs (i.e., without tails) is restrained by the scaled standard deviation—breaching this would therefore clearly indicate non-normal, heavy tails. Under portfolio diversification, the effect is often less flagrant and an outright breach unlikely. Still, large cES values versus small standard deviations hint at non-normality.

Especially if we have no explicit meta-information about which positions hedge each other, or if hedges just slightly deviate from their perfectly complementary PnL

[2] Verify, along a similar line of thought as given at the beginning of this chapter, that

$$|\,\mathrm{cES}[\alpha|\Omega]| \leqslant |\,\mathrm{ES}[\alpha]|.$$

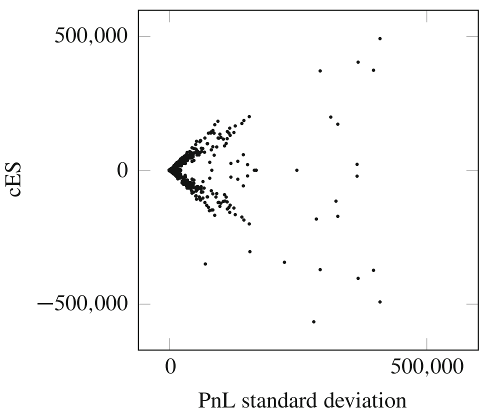

Fig. 13.1 Standard deviation of positions' PnL vectors vs. their cES

behavior, this view can help us weed out distracting positions with large individual ES/VaR values but zero risk influence. The remaining positions become candidates for further introspection in any analysis exercise we might have to perform.

VaR Noise 14

Having examined the static properties of the VaR, we now look into its dynamic behavior over time. As new positions are entered or old ones closed, and as the volatilities of the assets involved change, the VaR, recalculated every day, will change as well. Often, such VaR changes and their reasons are of more interest in risk management than the level of the VaR itself.

In order to appraise VaR changes, it is useful to first look into the behavior of our VaR measure in the special case of a constant portfolio and benign markets. It turns out that even in such a stable environment the VaR will fluctuate to some extent. This baseline of natural noise is good to keep in mind when analyzing a particular VaR change, when comparing different VaR models, or when assessing the usefulness of certain optimizations. Effects below that baseline might be inconsequential really.

To sketch this baseline, we create a pseudo-history of random normal returns (say, in Excel) for a hypothetical asset. Since we can disregard units here, these returns can directly be viewed as PnLs.

The history consists of 3 years or 750 days. For the first 2.5 years, we create standard normal returns, i.e., with standard deviation 1. For the remaining half a year, we create returns with twice that standard deviation. We proceed to take a look at the VaR behavior in the third, last year (the first 2 years only provide a full, valid input of "historical" data for that final year of interest). Since we know the underlying distribution, we know what the VaR ought to be on each day, and we can compare it to the VaR estimates of various model flavors.

We first examine the simplest one—computing the VaR from mirrored but non-rescaled returns (see Fig. 14.1). During the first 6 months, we notice how the VaR estimate from the raw returns skips between constant levels of similar magnitude. Such a skip happens whenever the return associated to the VaR scenario for a given day falls out of the historical 500-day window used for the following day.

Then, after 6 months, the VaR estimate starts to change and to converge to the newly established vola level—yet as can be seen, very slowly. It takes time for the

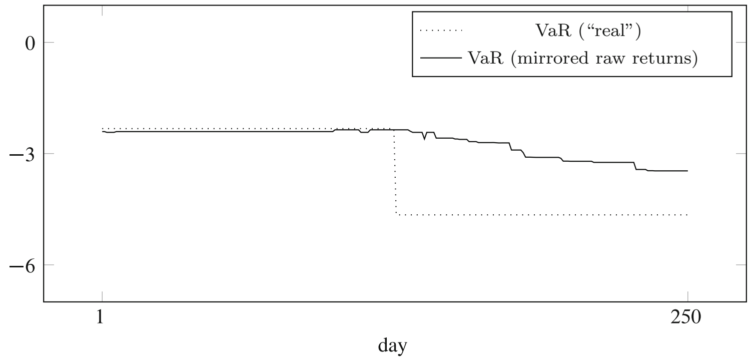

Fig. 14.1 VaR estimate from artificial raw returns; 1 asset

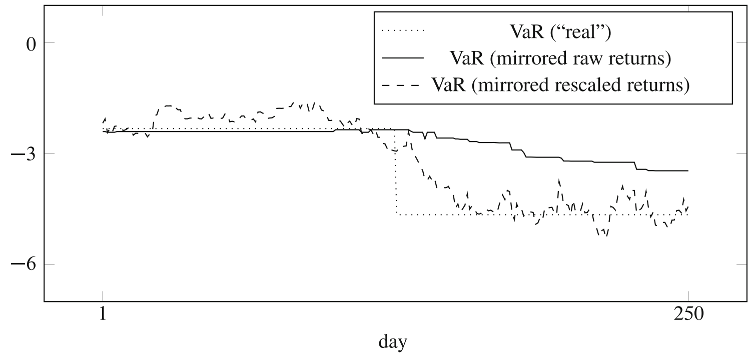

Fig. 14.2 VaR estimate from artificial rescaled returns; 1 asset

new, larger returns to influence sets of 500 returns that still mostly contain old, low-vola ones.

Let's see how our historical approach—based, again, on mirrored but now also rescaled returns—performs under the same circumstances. Figure 14.2 immediately shows that the rescaling has a major impact on the VaR estimate—it is much more volatile and overshadows any level skips (which happen underneath anyway). The reason for this VaR volatility is that each day's local vola estimate is based on only 20 returns. This small sample size hence causes the crucial target vola estimate to fluctuate more and to randomly deviate further from any "real" underlying standard deviation. The more important aspect, however, is that this VaR estimate, by design, almost immediately reacts to the mid-year vola level change and quickly converges to the newly established regime.

Now, the magnitude of our VaR's volatility is still somewhat striking. There are two ways to cope with this. The first is to notice that the deviation behavior is only an "error" because we made it so—in real life, we never know the underlying

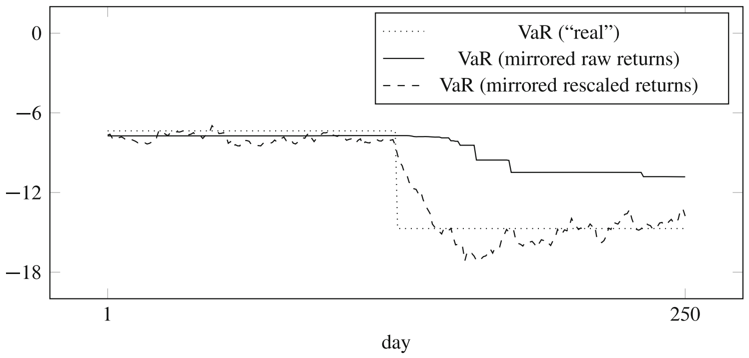

Fig. 14.3 VaR estimate from artificial rescaled returns; 10 assets

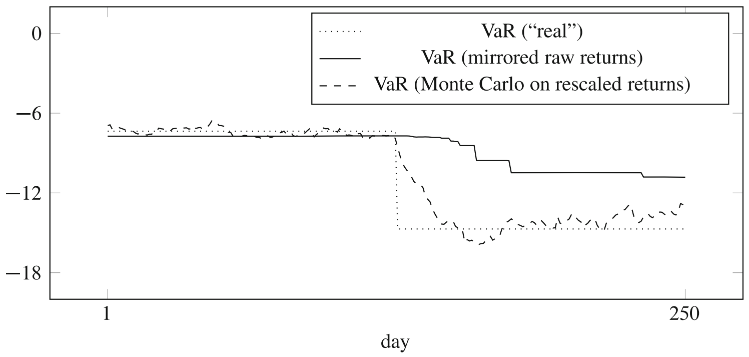

Fig. 14.4 VaR estimate from Monte Carlo on rescaled returns; 10 assets

distribution and might as well tacitly assume our estimate to be perfect (so just remove, in your mind, the dotted line of the real VaR).

Second, the effect becomes attenuated if a portfolio does not depend solely on one single asset. For a portfolio of 10 uncorrelated assets, the behavior seems less suspicious, as can be seen in Fig. 14.3. Some risk factors' target volas are overestimated, some are underestimated; combined, these errors tend to partially offset each other.

We get a very similar picture with the Monte Carlo approach on top of rescaled returns (see Fig. 14.4), again for 10 uncorrelated assets. There is one important issue to keep in mind: Monte Carlo introduces an additional random deviation or Monte Carlo error. This can be made arbitrarily small, e.g., by using a very large number of normals or by appropriately mirroring those normals as well. However, even when using an absurd 10^{10} scenarios and thus basically eliminating any Monte Carlo error, your Monte Carlo VaR estimate will only ever converge to the (dashed) line driven by the target volas (the figure, in fact, depicts this limit) and *not* to the (dotted) real VaR. So you can't Monte Carlo yourself towards the truth—it is just too elusive through our short-term vola spectacles.[1]

The area between the line of a VaR estimate and that of the real VaR is an indication of how systematically or how long an estimate is off. If it takes too long for an estimate to adjust, whole series of VaR breaches or backtesting violations may ensue. Rescaling clearly makes this area smaller, but because nothing is free, we buy this model reactivity by sacrificing some day-to-day stability of our risk measure. The proposed model mimics a hummingbird instead of a sloth.

Finally, we can take a more narrow look at the daily VaR fluctuations driven by a short-window target volatility. For one individual asset and a time series of standard normals, we can compute each day's local/target vola estimate L_i and the relative changes $L_{i+1}/L_i - 1$ over time. It turns out that the standard deviation of these changes is about 5%, par for par the standard deviation of relative VaR changes in any rescaled setup.

Like above, we can do the same exercise for 10 uncorrelated assets (we use the square-rooted sum of the local *variance* estimates for this). Here, the daily VaR changes clock in at a standard deviation of about 1.7%. Now, with 2200 risk factors we are tracking many more than just 10, but we should be aware that often just a small subset of them drives the risk, especially in sub-portfolios. Furthermore, risk factors may at times be highly correlated. This clumps them together and makes them act as if they were fewer in number, with less noise offsetting or relief.

[1]The Monte Carlo error depends mainly on the number of Monte Carlo scenarios used. You could analytically determine how far the Monte Caro estimate is likely to be off the limiting case of infinite scenarios, or you can simply try out sets of different random numbers to get an idea of this error range. You may experience, e.g., the Monte Carlo VaR with 5000 scenarios in a real-world portfolio to randomly deviate from the dashed (not dotted!) line by between ±3% and ±5% in relative terms.

15 Backtesting

The VaR model estimates tomorrow's PnL behavior by projecting plausible asset returns from these assets' recent history. The next day, specific assets returns will materialize, as will a corresponding actual PnL. To compute it, we merely have to price our positions under two scenarios—the market scenario used for the VaR calculation and that of the following day.

Let's use some exemplary dates to illustrate this. On the evening of some January 10, our VaR calculation uses historical market snapshots (the most recent one being $\mathbf{S}_0$ of January 10) to estimate the PnL behavior of the 11th. One day later, on the 11th, a new market snapshot $\mathbf{S}_0'$ becomes available and will in turn contribute to the VaR prediction for the 12th. But at the same time, this new scenario allows us to calculate the actual PnL experienced between the 10th and 11th[1]:

$$\Delta p^{\Omega} = p^{\Omega}(\mathbf{S}_0') - p^{\Omega}(\mathbf{S}_0).$$

As the non-bold typeface suggests, this PnL value is a single number and not a vector. It is what actually happened.

Comparing the actual PnL and the VaR over time gives a good first indication of our VaR model behavior (see Fig. 15.1 for an example). On most days, the PnLs should be above the VaR. After all, we expect the VaR to be breached with only a 1% probability—in other words, in a time series of 200 days, we expect roughly 2 PnLs to be below the VaR, i.e., to violate it. Also, whenever the PnL fluctuation starts to increase, we'd expect the VaR to widen soon in turn because, by design, it will take those new, more volatile asset returns relatively quickly into account with its short, 20-day tackle on the target vola.

[1]Note: we calculate this PnL one day after the corresponding VaR and should therefore use the portfolio of positions as of the tenth to align VaR and actual PnL. The next day's VaR calculation might already operate on an evolved portfolio.

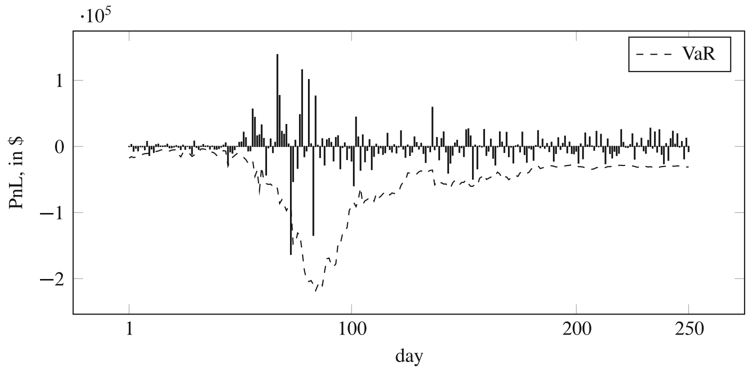

Fig. 15.1 Actual PnL versus VaR

This can be quantified a bit more rigorously with what is called *backtesting*. For this, we look at the number of VaR breaches or backtesting violations over a certain period of time, usually one year or 250 business days. Over this period we expect, on average, 2.5 such violations, but of course we will actually experience 0 or 1 or 3 or 6. To assess how many violations are acceptable, we simply use the probability of witnessing certain numbers of violations under a perfect model assumption and check whether this probability seems unrealistically low. For a perfect model, the number of observed violations X follows a binomial distribution, and the probability of experiencing exactly k violations in 250 days is:

$$p_{X=k} = \binom{250}{k} 1\%^k (1-1\%)^{250-k}.$$

Now, we could use this probability of observing a specific number of backtesting violations to grade our model, but the probabilities of individual outcomes are tricky to compare across different time horizons—observing exactly two violations in 200 days has a much higher probability than observing exactly 2000 violations in 200,000 days.

It is better to standardize our metric and to rely on the probability of observing k *or more* violations:

$$p_{X \geqslant k} = \sum_{i=k}^{n} p_{X=i} = 1 - \sum_{i=0}^{k-1} p_{X=i}.$$

Statistical tests generally result in such probabilities of observing an outcome under a given model assumption; these probabilities are referred to as *p-values*.

Table 15.1 Probability of observing k or more violations over 250 days

k	$p_{X \geqslant k}$ (%)
0	100.0
1	91.9
2	71.4
3	45.7
4	24.2
5	10.8
6	4.1
7	1.4
8	0.4
9	0.1
10	0.0

In Excel, you can use the formula `1-BINOM.DIST(k-1;250;0.01;TRUE)` to obtain this value for our $n = 250$ setup (for $k = 0$, we have $p_{X \geqslant 0} = 1 = 100\%$).

If this probability is small for some observed number of violations k, we should question the model quality. Table 15.1 lists the probabilities for witnessing $k+$ violations over 250 days.

Witnessing, for example, 8 or more violations over 250 days is highly unlikely at a probability of 0.4%—we should check our model implementation, our data feeds, our pricing engine, etc. While on average we expect 2.5 violations, witnessing 4 or more violations is not that unusual at a 24.2% probability—this is still acceptable. We probably want to start doubting our model at around 6 violations, where the probability becomes smaller than 5%, an often-used threshold in statistical tests of this sort. However, it is still necessary to put such a result into context: at a 4% probability, we actually expect 6 violations to occur—even in a perfect model—about once every 25 years. So once in a generation, we might as well let such a result slide (it helps if it doesn't happen in a model's first year of operation). Even more pronouncedly, if we monitor 25 independent portfolios, one of them is likely—again, under a perfect model—to exhibit 6 violations purely by chance.

Finally, in practice, there is another issue to consider. In addition to backtesting violations we expect from a merely statistical point of view, there are some violations that fall outside the scope of the VaR model entirely. The VaR can only predict distribution characteristics visible in the market data, and it cannot possibly account for risk factors changes not represented there. Out-of-left-field events must unsurprisingly surprise the VaR. Most often, such events are political in nature, e.g., a central bank unpegging a currency after years of artificially low volatility, such as when the Swiss central bank in 2015 allowed a sudden overnight increase in the Swiss franc's value. Such events should be managed via stress tests, and any related VaR violations should ideally be discarded from backtesting considerations (although the regulator, of course, might beg to differ).

Backtesting and its visualization vindicate our particular choice of VaR model. A bare-bone historical simulation, i.e., one without volatility rescaling, will obviously run into issues like clusters of backtesting violations during rising-vola regimes, to which such a model would not react quickly enough.

In our setup with vola rescaling, however, backtesting basically *has* to work by construction—as long as tomorrow behaves similarly to the recent past. Here, backtesting is less of a statistical test of the underlying model assumptions, for there are very few of them in the first place; instead, it mainly checks for a bug-free model implementation. A secondary fruitful application of backtesting is assessing a model's dependence on parameter choices, its *parameter sensitivity* (more on this in Sect. 17.4).

Now, while backtesting is a simple, quantitative way of plausibilizing a model's behavior, we seem to disregard much of what is going on above the 1% quantile, which, consequently, also forces us to use relatively large time series in such backtesting setups. An alternative test—the *distribution test*—is more expressive, and it is coming up next.

16 Distribution Tests

We can relate the VaR model's prediction (tomorrow's PnL distribution) with actual, later outcomes (the realized and experienced PnL) in a way that is more expressive than the backtesting with its focus on relatively rare VaR violations. We can view the prediction—a set of 1000 equally likely PnL values—as a skeleton or proxy for the infinitely many possible PnL outcomes. More specifically, the *sorted* PnL predictions $\Delta p_{(i)}$ delimit 1001 adjacent intervals into one of which the actual PnL must later perforce fall:

$$(-\infty, \Delta p_{(1)}], (\Delta p_{(1)}, \Delta p_{(2)}], \ldots, (\Delta p_{(1000)}, \infty).$$

To depict these interval boundaries, we simply reprise a graph we've encountered at the beginning of the book in Fig. 16.1.

Note how the left- and right-most intervals tend to be larger than the central ones, as the predicted PnL values or interval boundaries are typically spaced further apart at the fringes. In a leap of faith, we declare that the probabilities for an actual PnL to fall in any of those intervals—if the model is correct—should be just about the same (in this case, 1/1001). Intuitively, dense regions represent areas where we predict the actual PnL to materialize more often—but the intervals in those regions are also narrower, which keeps their hit probability at bay. Conversely, sparse regions indicate areas where we generally expect few PnLs to happen—but if a stray PnL should ever appear nearby, the larger intervals there can scoop it up more easily. In both cases, hit frequencies and interval sizes offset each other, yielding uniform probabilities for the various intervals.

We can have a go at this from another direction: the VaR, i.e., the 1%-quantile, delimits the 10 most negative PnLs, and the probability of hitting that area is 1%. The 2%-quantile fences in the 20 most negative PnLs; the probability of hitting that area is 2%. Consequently, the probability of falling *between* those two

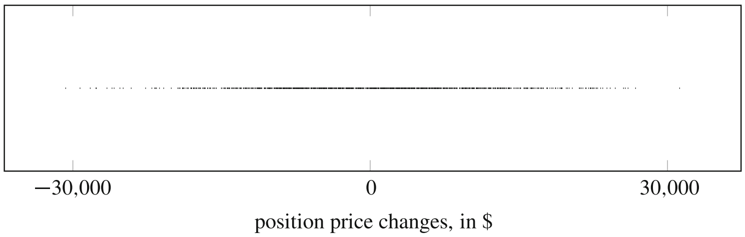

Fig. 16.1 Price changes as interval boundaries

quantiles—the 10th and 20th most negative PnLs—must again be 1%. The reasoning above just breaks this down to a more granular level.[1]

We can proceed to a fairly straightforward test setup. Each day, an actual PnL will fall into one of the 1001 predicted, hypothetical PnL intervals, and we can assign that interval's index to that PnL. If we furthermore divide that index by 1001, we get, for each day, a value u between 0 and 1. We can repeat that for a certain number of days—typically 40—and obtain as many u-values. As each index is equally likely, the normalized u values are—or at least should be in case of a correct model—uniformly distributed. And that's something we can test.

The following steps should help build up some intuition; you can easily retrace them using Excel's `RAND` formula, which generates a [0, 1]-uniform. An example of 40 such artificially generated uniforms is given in Fig. 16.2.

These values sure look random, but are they uniform? If they are, there should be some small, some medium, and some large u-values. To visually check for that, we sort them by size and graph them again in Fig. 16.3.

Clearly, sorted uniforms should form a diagonal, as they do in this example. Now, by pure chance, such uniforms might also deviate from the diagonal (the smaller the sample size, the larger the potential deviation). To give you an impression of that deviation range in our case of a 40-sized sample, and sticking with this setup of artificially created uniforms, Fig. 16.4 depicts ten such sorted uniform samples.

It turns out that this range can be described with so-called *confidence intervals* based on the beta distribution. The kth largest uniform $u_{(k)}$ of a size-n sample is

[1]The probability of a random number falling between two quantiles $q_d < q_u$ is $u - d$ (see Sect. A.7). The PnL predictions (our interval bounds) correspond to $k/1001$-quantiles; the probability of a random number falling between two adjacent ones, e.g., $d = 17/1001$ and $u = 18/1001$, is $u - d = 1/1001$.

You will notice that this treatment of quantiles is slightly different from our previous one, where we defined the 10th most negative PnL value to correspond to the 1%-quantile instead of the $10/1001 = 0.99\%$-quantile. This is largely inconsequential here due to the large number of PnL values involved, and our respective definition choices are solely motivated by convenience and concise, short form notation. In any case, empirical quantiles like these are often handled slightly differently according to a given problem at hand, for example, when it comes to dealing with the very first and last intervals or with questions of quantile interpolation.

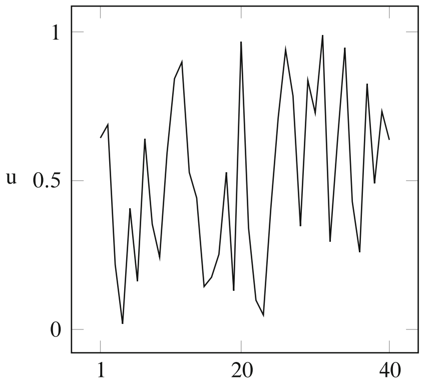

Fig. 16.2 A sample of 40 uniforms

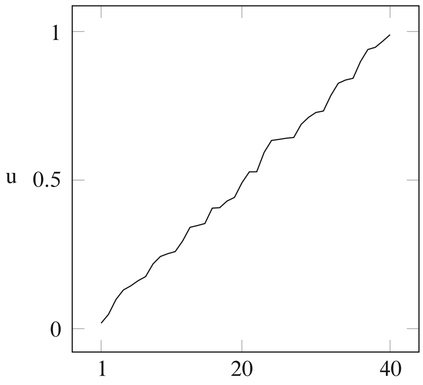

Fig. 16.3 The same sample of 40 uniforms, sorted

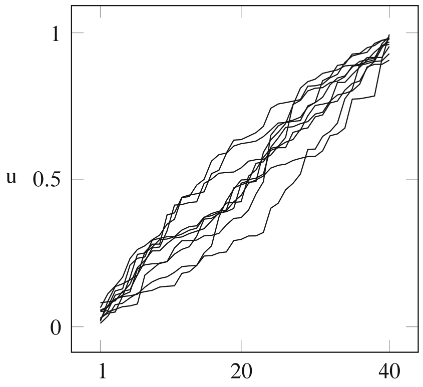

Fig. 16.4 Ten samples of 40 uniforms, sorted

Beta$(k, n + 1 - k)$-distributed, which allows us to gauge plausible ranges for those uniforms.[2]

[2]The confidence interval delimits the outcome of each u between an upper and lower bound (it is thus two-sided); it uses the 5% and the 95% quantiles of the Beta distribution for this, spanning the 90% of outcomes in between (hence it is a 90% confidence interval). You can use Excel's

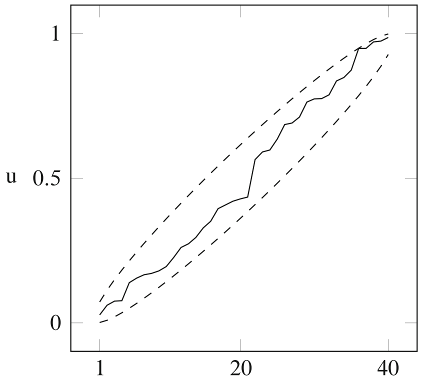

Fig. 16.5 Good distribution test

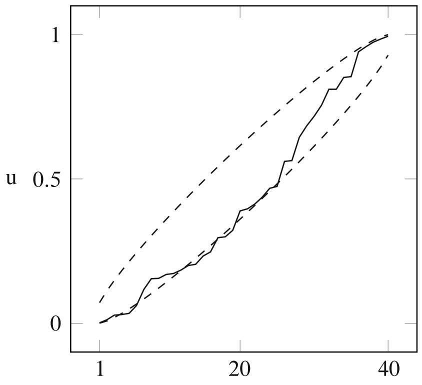

Fig. 16.6 Bad distribution test

We are now ready to look at series of normalized PnL interval indices, derived from actual PnLs versus their respective PnL distribution predictions. Here are several real-world examples, along with two-sided 90%-confidence intervals. The first example, given in Fig. 16.5, shows a portfolio for which the model works well and makes predictions within the confidence intervals.

A second example, given in Fig. 16.6, shows a model that does not seem to work that well—it seems to tend to underestimate the actual risk. Why is that so? The lower left part of the graph is the region of interest, for it roughly corresponds to negative PnL values, i.e., losses. That's because PnLs close to zero should usually hover around interval index 500 (or $u \approx 0.5$), positive PnLs or gains lie in the upper intervals above it, and negative ones or losses are below. Like for all PnLs, we expect the losses (represented by the observed u-values) to lie on the main diagonal. If they are below, it means that realized PnLs happened to fall in lower-than-expected intervals—the losses are more severe than predicted. The model thus underestimates these losses' magnitude and hereby the risk.

`BETA.INV(0.95;k;n+1-k)` and `BETA.INV(0.05;k;n+1-k)` functions to obtain each day's upper and lower bounds.

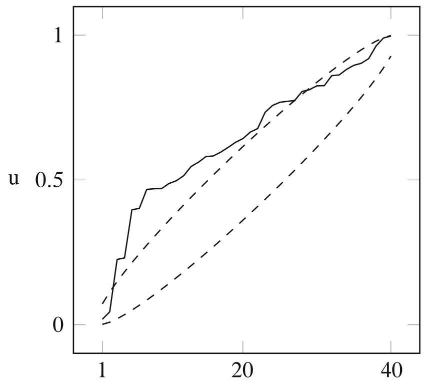

Fig. 16.7 Ugly distribution test

Finally, another example of a model that doesn't work well, albeit with less dire consequences, is given if Fig. 16.7. In this case, the model overestimates the risk and the magnitude of losses. Mathematically, this is just as poor a model as the previous one. In practice, though, this model errs on the conservative side, which for the regulator usually proves fine. Bearing the increased costs, i.e., higher capital requirements, of slightly risk-averse models may be preferable to constantly having the validity of a model hovering around the lower confidence band questioned.

Using such distribution test graphs for various portfolios is a quick and comprehensive way to check a model's quality. I personally prefer such graphical, expressive tests to the blind use of statistical tests and their resulting—mono-dimensional and highly condensed—p-values (like the ones we encountered in the backtesting chapter, supplementing the VaR/PnL time series graphs). Some don't deem such tests to be strictly quantitative—I disagree: they represent a wealth of quantitative data and do so in an interpretable context. Nonetheless, some stakeholders will insist on p-values, and they can easily be created. We can, for instance, check for the uniformity of our u-values with standard tests like Kolmogorov-Smirnow or Anderson-Darling, which readily yield those comforting numbers.[3] In my view, p-values best serve to unceremoniously seal an argument you are ready to win with a more transparent reasoning anyway, much like tossing in a PhD title into an email signature.

Distribution tests are more powerful than backtesting, in the sense that they use more information than mere binary and therefore abridged "violation yes/no" flags. They require fewer days to become meaningful, and we use 40 to 60 days as opposed to the 250 days common in backtesting. As the number of days increases,

[3]While we generally prefer Python and NumPy when doing statistics, the software suite R offers some good support for such tests. The Kolmogorov-Smirnov test is built-in (`ks.test(u, "punif")`), while Anderson-Darling is available as a separate library (called `ADGofTest`). Please note that some tests do not allow for identical u-values, which in our case can happen because of the discrete intervals. You can simply add/subtract tiny and different offsets to each u-value to disentangle them, e.g., $10^{-8}k(2\mathbb{1}_{u_{(k)}<1/2} - 1)/n$.

the confidence bands will get narrower, and as is the case with all statistical tests, too large a sample size will eventually guarantee a *p*-value violation. Using 2–3 months of business days seems to be a reasonable compromise: the time series is long enough to avoid too large a leeway in terms of confidence bands, and it is short enough to not artificially cause spurious rejections due to an oversized sample.

Like some other modeling choices, the number of days to use in distribution tests is a matter of preference rather than optimality, which can lead to inquiries with remarkable follow-up explanation efforts. Defending such choices will be the topic of Sect. 17.4.

17 Nine to Five

We now try to examine our risk measures and their properties in the practical context of daily risk management. We structure this chapter by the most common questions you might face, in reverse order of urgency.

17.1 Why Did My VaR Increase?

This is the most common question you will encounter. It will be raised by the CEO, desk managers, traders, and just about anyone in the risk audience.[1] A VaR blowup means increasing capital requirements, may cause breaches of VaR limits imposed on desks or traders, indicate problems with a hedge, signal a pick-up in market activity, etc. The opposite question—"Why did my VaR decrease?"—is much rarer, albeit just as warranted.

The sheer number of positions and risk factors, i.e., the vast amount of raw data available, combined with the (desirable but difficult-to-segregate) portfolio effect can make it tricky to immediately assess specific reasons for VaR movements. To answer this question and to localize any root causes of a VaR change, we basically try to narrow down this large search space.

But before we delve into our data, we should check the calculation for profane mishaps. Maybe a position source system has failed to deliver its data, and instead of the usual 1,000,000 positions, we have calculated only half of them? Maybe some fallback mechanism for such missing feeds (usually just copying the previous day's corresponding data over) has failed? Maybe a market data system has delivered a clearly wrong market value that escaped the data scrubbing team, causing a huge spike and an increase in the target volatility of a dominant risk factor? Maybe all positions are there and priced alright, but the definition of which positions belongs

[1]We like to treat the VaR as negative value to preserve consistency with support measures like the VaR-contribution. We quickly translate "increase" into its opposite and keep mum about it.

Table 17.1 VaR-contribution

Risk factor	Sensitivity	VaR-sensitivity	VaR-contribution
FX-EUR	−12,228	176.45	−2,157,701
FX-RUB	−661	152.43	−100,727
FX-GBP	537	−104.10	−55,889
FX-CNY	−119	182.50	−21,641
IR-EUR-Y30	21,668	−0.61	−13,119

to which portfolio was messed up due to a small typo during yesterday's otherwise minor change to the portfolio definitions? Robust automatic check procedures, detailed and human-readable logs, and warning or error alerts will greatly help account for such adverse technical headwinds.

Once such mundane reasons are ruled out, we can proceed to see whether market changes or changes in our positions cause our suspicious figures. A quick cross-check is to calculate yesterday's positions with today's scenarios, and vice-versa. This is a good indicator of which investigative path to go down next.

On the market or risk factor side, the partial VaRs like VaR^{FX} or VaR^{IR} (and changes or non-changes in some of them) help restrict the search scope to types or subsets of risk factors. We can then determine the most important risk factors there (via the portfolio's absolute sensitivities) and verify whether the corresponding target volatilities have changed.

A more elegant and comprehensive way is to check the risk factors with the most negative VaR-contributions—this measure helpfully already intermingles sensitivities and volatilities in a consistent way. Table 17.1 shows an example report of the risk factors with the most negative VaR-contributions for some portfolio. Comparing such a report with, e.g., the previous day's one quickly reveals the comparative relative impact of the various risk factors and sniffs out the combined effect of changes in volas and/or sensitivities. Or to put it more prosaically: in isolation, a seemingly tiny IR vola increase from 0.0003 to 0.0005 might escape the quick human glance amid apparently larger vola movements; the VaR-contribution gives such changes the appropriate sensitivity-weights and is thus able to denounce a change as influential or less so.

On the position side, we should first be aware of this:

- Adding a position to a portfolio, while usually amplifying the VaR, can also dampen it, e.g., when adding a position that acts as a partial hedge to the remaining portfolio. In the extreme case of adding the portfolio $-\Omega$, we can even reduce the VaR to zero.
- Removing a position from a portfolio, while usually curbing the VaR, can also magnify it, e.g., if we remove one of two deals that hedge each other.[2]
- Even positions that idly remain in a portfolio can affect the VaR in both directions, for their contract terms might trigger changes in their PnL characteristics.

[2]As we want to avoid the ambiguous "increase" and "decrease" for the negative VaR, the thesaurus is having a field day.

So we need to go about our analysis accordingly and cannot limit our focus to new positions alone. With that in mind, how to best drill down to a position that moved the VaR?

It is tempting to first examine the positions' very tangible sensitivities. But while feasible, this is often quite tedious. There are, plainly, lots of sensitivities on the position level (10^6 times the average number of risk factors the positions are sensitive to). Many of them offset each other and blur the picture. Unsuspiciously small sensitivities might stem from non-linear positions and just camouflage their severe tail losses. And ostensibly large sensitivities might have a low corresponding volatility and thus a negligible impact.

Individual VaRs for each position often don't offer very clear signals either, as portfolio effects are not accounted for. Hedges, if not filtered out beforehand and dismissed for their zero risk contribution, will show up twice: linear hedges with two identical (potentially large and alarming) individual VaR values, and non-linear ones, even worse, with different individual VaRs. The former obfuscate the picture somewhat more openly (two positions with the initially troubling but exact same VaR of −217,244.63 are, after all, almost certainly hedges to be dismissed in a quick exploratory analysis). The latter can be more baffling, as their individual VaRs can differ arbitrarily. These very same issues also haunt the incremental VaR of positions.

A more useful hint comes from the positions' cES values. Positions with large negative cES values drive the VaR. And as the cES values of (both linear and non-linear) hedges have opposite signs and the same magnitude, hedges can often be reasonably identified and "guessed away," leaving the remaining positions with dominant cES values as analysis candidates. Still, if there are overly many hedges or imperfect ones that do not close each other completely, they will continue to fog any quick and mindless drill-down analysis of a large set of positions.

How to get around this issue of hedges? In an ideal world, positions that hedge each other are tagged, and you can thus filter them out explicitly. If not, you can try to automatically detect hedges by pair-wise comparison of the positions' basic attributes: if you detect two bonds with the same maturities and coupons but with opposite nominals N and $-N$, they act as hedges and can be removed from consideration. Finally, you could attempt to identify hedges by the positions' sensitivities. If a deal's sensitivities $\mathbf{s}^\alpha$ complement another one's $\mathbf{s}^\beta$ via $\mathbf{s}^\alpha \approx -\mathbf{s}^\beta$ suspiciously closely, those two deals likely hedge each other and can often plausibly be ignored in the search for influential positions. But even in a best-case scenario with all hedges explicitly known and accounted for, positions will often, by chance, at least partially act like hedges to parts or all of the remaining portfolio, so the deficiencies of our support measures with regard to offsetting behavior will linger.

A fast impact analysis can be performed via the visualization presented in Chap. 13: a scatter plot of all deals' PnL standard deviation versus their cES. This allows you to detect the largest and most influential positions as asymmetries in the various characteristic parts of the plot, with the fuzzy human eye readily and helpfully filtering out hedges both perfect and imperfect.

17.2 How Will the VaR Change?

When a new position is about to be added to a portfolio, the effect on the VaR is clearly of interest—after all, it might increase the capital requirements or cause limit breaches. For relatively small positions, a fast shortcut to approximate the VaR impact is to use the portfolio's VaR-sensitivities (see Sect. 7.2). For this, we multiply the new position's sensitivities with the portfolio's corresponding VaR-sensitivities and simply add the resulting dollar terms to the portfolio VaR. This approximation is quick and often close enough. (Note: the signs matter, so if a deal is, e.g., sensitive to the 10-year EUR interest rate to the tune of -1000 and the respective VaR-sensitivity is -4, than the approximate impact on the VaR is $+4000$. This is added to the negative VaR—so $|\,\text{VaR}\,|$ decreases.)

A more tedious yet truly precise method is to simply recalculate the VaR with the new position included in the portfolio. As we usually already know the existing portfolio's PnL vector, it is sufficient to compute the new position's one and to then add those two vectors before extracting the resulting new VaR. This of course corresponds to the incremental VaR or pre-deal inquiry mentioned in Chap. 11.

A closely related question is the following: what size should a new position have in order to achieve a certain target VaR? This target is often some optimal or capital-minimizing VaR. To obtain it, we just recalculate the VaR repeatedly, each time adding a different multiple $a\alpha$ of a deal α to the portfolio. (We can even calculate the position's PnL vector only once and then just scale it by a before combining it with the portfolio's vector.)

How will the resulting $\text{VaR}[\Omega + a\alpha]$ look under various position sizes or scaling factors a? As we increase a arbitrarily towards ∞, the position $a\alpha$ will at some point become the dominant deal in the portfolio, and the VaR will get ever more negative. The same happens if we decrease a towards $-\infty$, as the deal once again will become dominant at some point. So we expect to see some sort of upside-down parabola whose maximum or least negative value corresponds to the "minimally risky" VaR. Figure 17.1 depicts an example where a weight of about $a = 0.8$ would lead to minimal capital requirements.

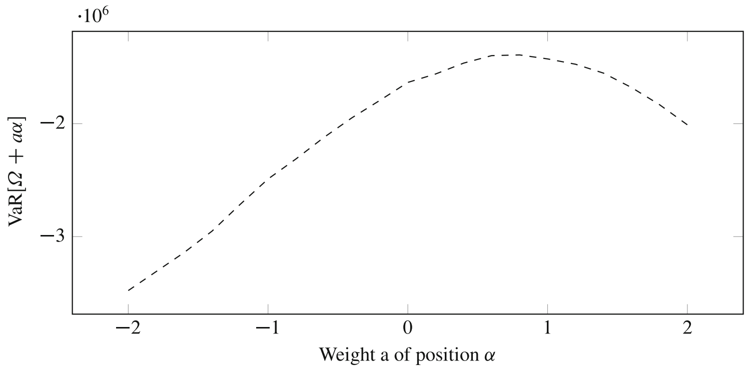

Fig. 17.1 Portfolio VaR under a variable new position

17.3 How Good Is the VaR Model Anyway?

The VaR is—like any summary statistic—a stark abbreviation, in this case of the wealth of information contained in a full PnL vector expressing the two-dimensional concept of risk. As such, the VaR may potentially hide much. It does capture one specific slice or aspect of risk but is probably most useful in a comparative sense (over time or between portfolios) rather than via its raw absolute level. The VaR is not the full picture of risk.

The data used to calculate it is real-world data, prone to errors, missing values, outliers, gaps, etc. The sample size (usually 2 years of data) is small with respect to the large number of risk factors or dimensions. The VaR model's quality much depends on the data fed into it; that data is not always pretty.

On top of that, to make the VaR model react quickly to volatility level changes, it is heavily driven by the last 20 days and their observed returns. This naturally involves a sample error, which is more pronounced for portfolios depending on few risk factors only. If the Monte Carlo modification is applied, an additional noise affects the VaR figure (see Chap. 14). Actual VaR results are fickle.

Nor can we simply math our way out of such issues. Alternative risk measures like the expected shortfall do have some desirable properties, but are still limited by the same basic restrictions. If anything, elaborate models may be too impressive for their own good, lulling the audience into a false sense of certitude and complacency. (More on this in Chap. 13 and the first part of Chap. 9.)

While it is a natural mental crutch to believe that the VaR or a VaR-like measure predicts the future, a better and more prosaic view is that it merely extrapolates some plausible asset behavior from that of the recent past. It assumes that tomorrow will behave like the last few weeks. If this assumption fails, the model's results become void. (For example, a central bank's decision to suddenly devalue a currency might cause a backtesting violation—yet the model couldn't possibly have accounted for it.) This must remain the realm of the underappreciated stress tests (see Chap. 6). The VaR can't foresee the future; it disregards the unrecorded risk.

These reasons should give anyone pause before overestimating the importance or reliability of the VaR. On the other extreme of the pendulum's swing, the VaR should not have to bear the blame for financial calamities—as some would have it in light of financial market crises. Instead, the man relying blindly or solely on it is the culprit, abetted by him who burdens this humble measure with elaborate but difficult-to-vindicate risk quantization duties. The overconfidence at the heart of it is not rarely fueled by mathematically impressive but less-than-transparent model choices.[3]

[3]Take, for example, VaR models that must estimate not overnight or 1-day PnLs, but 1- or 3-month PnLs. They may rely on correspondingly large historical returns, but this would make the returns either fewer or older or overlapping. Alternatively, they may try to use daily returns and project them farther into the future, which raises questions of reversion to the mean, among others. The frowned-upon traditional shortcut is to simply scale up the 1-day VaR by the square root of time (in days). Strictly speaking, this is somewhat off or incorrect, but it seems to usually do the trick. A

By construction, the VaR model will behave—within its limitations and its central assumption of "the world stays the same"—as expected and by and large correctly guess tomorrow's market volatility. Still, the model must pass at least some quality checks, and be it only to ensure the absence of programming mishaps. For this purpose, the plain backtesting or more elaborate distribution tests (Chaps. 15 and 16) serve well. Partial VaRs and synthetic marginals (Chap. 11) help plausibilize the model behavior on a more granular level, as the overall, aggregate numbers might hide erratic effects that cancel each other out. Going much further in discussion or deed often seems to be unwarranted—a simple model first and foremost calls for simple workability tests.

17.4 Hmm, How Did the Parameters Materialize?

An ideal model is one devoid of free parameters, one that "just works" without tinkering. John von Neumann once stated: "With four parameters I can fit an elephant"—aptly hinting at the problem of too much modeling leeway.

Now, our VaR model comes pretty close to that ideal. It has few parameters, and all have an immediate, transparent interpretation. We typically use 500 historical returns and rescale them to the recent 20 days' volatility (maybe flooring the target vola at 20% of the overall historical volatility). When using Monte Carlo or injecting kurtosis, a few more setscrews are introduced. If using log-normal interest rate returns, a certain shift is required to be able to account for negative returns. We won't lose much sleep over these settings.

But while these choices are readily made, they can be much more tricky to justify and defend. Especially those parameters without a natural optimality to them, i.e., those not resulting from an accepted calibration procedure, raise doubts. Rightfully probing questions from the regulator ("Why use 20 and not 30 days for local volas?") have no obvious retort. The answer of first resort—"expert opinion"—seems terribly smug; the contrasting alternative—"common sense"—amounts to an insult. Alas, there is often no very satisfactory answer.

There are, however, some avenues in responding that may at least minimize the back-and-forth required to allay such parametrization concerns. One can try, first, to kill the question in the crib and openly shame the parameters from the start as being arbitrarily set: "These parameters have wiggle room—there is no obviously correct or optimal setting. We discussed them and our consensus fell on this particular tuning; we feel that it is reasonable." Your audience, hopefully impressed that you didn't try to bury the issue but to highlight it, might feel less inclined to dig deeper; the hole is already there and plain to see.

A bit more suave it is to rechristen the parameters and declare them to be *meta parameters*—not really within the model scope but somewhat residing ethereally

proper long-horizon estimate, on the other hand, has ample leeway in terms of modeling and thus allows for a wide range of results, which doesn't necessarily invite much greater confidence in it.

outside of it. A good example is the choice of the 20-day window for target and local volas. Longer windows cause the VaR to be more stable and to react slower to market changes, while shorter ones do the opposite. This setting is due to the model users' preferences, not to some parameter optimality.[4] In any case, don't ever casually observe that you "tested" various setups (in an feeble attempt to make your point stronger). That will trigger the immediate, checklist-ticking "Can you please send us your test results?" Now you'd actually have to run those tests.

If the "questicide" fails, an answer must reluctantly be produced. Luckily, people since Hammurabi have a fond credulity for all things written. The "I told you so" gets more heft when framed as "This is already written in stone." Try hard to find a reference to a book or paper or institution that has already put forward what you are trying to do.[5] The best reference is NASA, but even an—as of yet illusive—Journal of Chart Analysis might handily absolve you from having to admit that no one ever has come up with your exact same ideas.

Unable to unearth a precedent, you can set up some committee responsible to agree on parameter values, along with periodic meetings, minutes, and stamp-wielding due process. Parameter values conjured up in such a setting are coated in paper-trailed consecration and less likely to be doubted.

If all fails, you are left to prove that other, similar parameter choices just don't make much of a difference. You could, for example, run a parallel calculation of the same model using 30-day instead of the usual 20-day windows. If the VaR time series obviously look very similar, then the model's *parameter sensitivity* is small, and the choice thus inconsequential and unworthy of investigative zeal. This is relatively costly in terms of setup and calculation efforts, but it might well be necessary to quell a topic.

All of the above is better, by the way, than trying futilely to prove some tuning parameter's "optimality"—any utility criterion you maximize will draw attention to its own shaky justification. Instead, as very last resort, you must rely on the gravitas of some PhD-adorned team member, preferably sporting a beard gray and flowing.

Needless to say, this section exists because we got stung. Be prepared to spend time on your parameter defense and reluctant of any model improvement entailing new free parameters (e.g., supposedly superior exponentially weighted volatilities with some—new!—decay parameter). We spent more time explaining parameters we considered obvious (and blatantly were not by our counterparts) than on actually programming the model. It was us who did not press them effectively from the start. The questions on model parameters are comprehensible, warranted, and indeed necessary. Our answers, however, were at times hesitant, piecemeal, or winding. You better best that.

[4]Other out-of-scope parameters are *fudge parameters*; they are introduced to circumvent numerical issues. An example is the shift used to make square root processes operate on negative interest rates.

[5]A good example is the RiskMetrics decay parameter setting of 94%. It often goes unscrutinized because it is so commonly used and ingrained in many model rehashes.

17.5 Can You Validate Your Model?

Answering this innocuous question may become formidably time-consuming fast. It is usually posed formally by entities like the regulator or audit, and it stubbornly clings to life in imposed conditions and follow-up requests. The stakeholders' incentives, their familiarity with a model setup, and their available time to dig into the matter vary—causing expectations to diverge:

- It starts with the poor choice of wording: the rather generic "validation" is bound to be interpreted differently by everyone involved. This is compounded by the fact that the actual people representing, for example, the regulator, will change over time, contributing novel and sometimes different views. Regulations or laws seldom offer explicit guidelines or unambiguous criteria.

  Worse still, you can technically *invalidate* a model and prove it to be deficient, but never really completely prove that it works—you might always be just one further test away from uncovering a major weakness.
- Charged with meaning in applied statistics, the word "validation" leads some to immediately focus and hone in on issues like hypothesis testing and distribution assumptions, possibly neglecting that, fundamentally, there is not a lot of statistics going on here. It is sure fun to ponder the use of a battery of statistical tests—more fun probably than stepping back in humility before the challenges in data or operations. Not infrequently have I encountered people that will gladly spend time on the most peripheral statistical aspects while not even contemplating looking at the 2200 risk factor time series—the actual core of it all. That would (obviously) be time-consuming and boring, but there is more to it than that. It is also just not on many people's radar, primed by the perfect data sandbox exercises prevailing in education and academia.
- Vague terms also naturally inspire a longing for formal frameworks and crutches to somehow pin it all down. Expect requests for "validation concepts" (essentially meta documents), for separate "initial model validations" and "continuous model validations," for "actionable" traffic-light score cards, and for explicit triggers and contingency plans (even to hitherto unknown events). While impressive at face value, this whole superstructure is built on the same unchanged, humble ground and more likely to cause sweat than make sense.

  Such a conceptual overhead tends to provide little relief in terms of requirement consistency. Over the years, we have been variously tasked to either provide *more* tests for increased coverage, or *fewer* ones for better readability and accessibility; to perform the *same* tests year in and year out for consistency, or to do *varied* ones to address different model aspects as markets change. These requests—all perfectly worthy and sensible on their own—are hard to reconcile.

What is there to be done? Well, first, avoid the word "validation." Rephrase it as test or, though unsexy and a mouthful, as plausibilization. No harm ever came from this (except to your tongue).

Second, make sure to impart that a VaR model is simple at heart; that the statistical assumptions are trivial; that market volatilities are captured by construction; that political events are out of scope; and that data and its limitations are front and center. You bolster your case if you avoid proprietary model customizations and opaque language or statisticalese. So curb you quants; smite them abacuses—stick to standard methods and familiar terms instead.

With all parties on the same page about these basics, *plausibilize* (see?) your model with the basic backtesting procedures and parameter justifications outlined in Sects. 17.3 and 17.4. Mention the VaR measure's inherent noise (Chap. 14) if you sense concern over minor model result deviations; address worries about distribution tails as needed (Sect. 17.6). Have your p-values ready, but provide them with context: a test might fail for risk-averseness, which is tolerable; and out of many tests, some are actually expected to fail statistically. Above all, visualize the main results—a graph grasped equals trust earned.

As for the overall scope and shape of the results, it would be fantastic to get away with providing a succinct, small subset of varying custom tests as a readable, crisp model quality characterization. Alas, I fear that this is not workable in practice. The regulator is very rightfully bound to distrust a bank's motives—what is to prevent one from running ten tests and reporting only the three best results? Test types should probably remain largely unchanged. As for sheer test counts, using fewer tests may be just as expressive and much more lightweight and accessible—but only as long as everyone involved agrees to this virtue of brevity. This is untenable, as different and changing personnel will require just another type of test, one more metric, or a higher granularity of sub-portfolios. Stemming this tide of requirement creep is hard, and it is probably better to instead just swim along with it by providing a large set of automatically performed tests. Such cover-our-ass results, however unwieldy and ponderous, are difficult to argue against. The world of VaR model plausibilization is one of prose, not lyric.

Now, anyone charged with evaluating something and reporting about it must also reveal at least some issues, lest it look suspicious. If handed the ten commandments under such a duty, we'd all find this or that commandment lacking in purpose or in need of clarification; in our best temper, we might limit our recommendation to reshuffling them. Something like this is to be expected in model validation as well—it will require some effort under the best of circumstances. The additional overhead will then vary greatly.

17.6 What About the Kurtosis?

It is likely that asset returns are not perfectly normally distributed, and it is generally assumed that returns exhibit tails heavier than normality would imply. A common way to measure this is the *kurtosis*. A normal distribution (of any standard deviation) has a kurtosis of 3. Larger kurtosis values denote heavier tails.

It is fairly easy to impose some kurtosis in a VaR model; one possible approach, that of scaling some of the joint returns, is given in Chap. 9. However, this would first require having an idea about a desired or target kurtosis, and it is here that it gets a bit murky—for kurtosis is quite fickle a measure, and it may, worse, also overzealously indicate tails where there really are none.

Let's look at the volatility of this measure first, which can easily be observed by doing a small experiment in Excel. Just create 40 random normals and determine their kurtosis with Excel's `KURT` function repeatedly; the sample's kurtosis will fluctuate between roughly 2 and 4, even though the sample stems from a normal distribution with a kurtosis of 3. This variability is caused by the influential 4th power term used in computing the kurtosis. For the same reason, outliers in a sample may potentially massively increase the kurtosis. It is indeed not unusual to observe market data time series with a kurtosis of over 100. Now, comparing such potentially large and highly fluctuating kurtosis values is possibly deceptive—two kurtoses of 60 and 80 seem to differ a lot, for example, but can be caused by data sets that are quite similar in nature.

Can we obviate this by relying on larger samples? Unfortunately, in our setup of historical returns observed in markets whose volatility regimes seem to change over time, this can lead to an undesired signaling of tails—the kurtosis cries wolf. Why so? Consider a sample of 500 artificial normal returns (with a standard deviation of 1), whose sample kurtosis will hover around 3. Now replace the last 25% of those returns with new normal returns with a standard deviation of 2. Those new returns are also normally distributed, and their kurtosis will fluctuate around 3 as well. The full 500 artificial returns should resemble an asset whose price fluctuations have doubled in the last 6 months, as illustrated by the raw returns and their local volatilities in Fig. 17.2.

If we blindly compute the overall kurtosis, however, we obtain a value different from 3—say, 4.83. This is purely caused by mixing two different, innocent normals. But blessed with the knowledge of how this time series came about, we can dismiss this warning of non-normal, heavy tails: this is a time series of normals whose vola level changed but whose tail characteristics effectively remain normal. Such a time series simply would not call for worrying about or injecting kurtosis into our model.

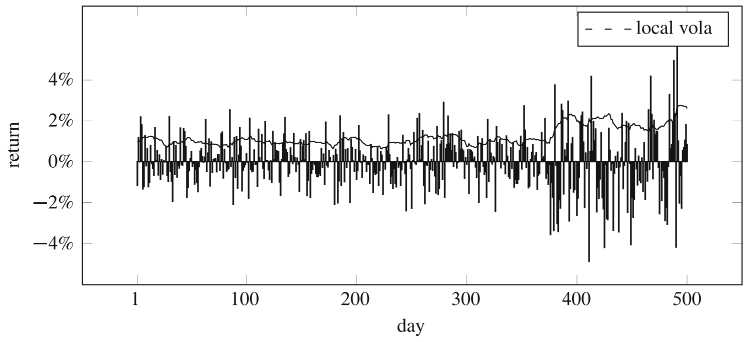

Fig. 17.2 Artificial returns with recently increased volatility

We can quantify this from a different angle: Fig. 17.3 depicts *local* kurtosis values over 40-day windows for the same time series. By their nature, the local kurtosis values fluctuate, but their median of 2.96 indicates no particular reason to believe in the large overall kurtosis.

We have seen that combining two sets of returns from different random normals will exhibit kurtosis. (Such combinations arise from *mixed* distributions, where a random variable can follow, at each realization, one of several distinct distributions). This is analytically tractable, but it is both simple and instructive to construct or simulate this effect. The border cases—taking all returns from the first normal distribution or taking all returns from the second one with a larger standard deviation—both result in a kurtosis of 3. Figure 17.4 shows how throwing an increasing percentage of large returns into a mix of normals affects the combined kurtosis. The volatile dotted lines are derived from 500 returns, where standard normal returns are replaced with more and more returns of standard deviation 2 (or 4). The smoother lines depict the same experiment done on a set of 100,000 normals.

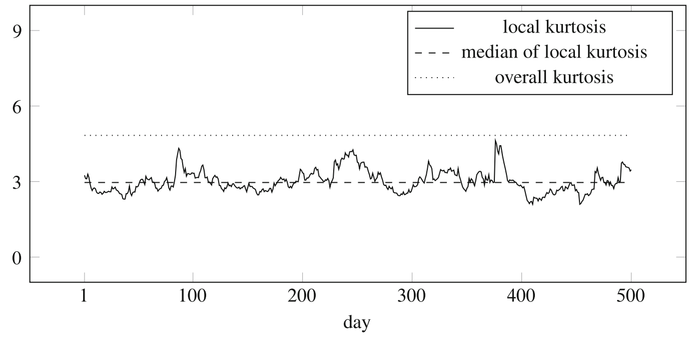

Fig. 17.3 Local kurtosis over time vs. overall one

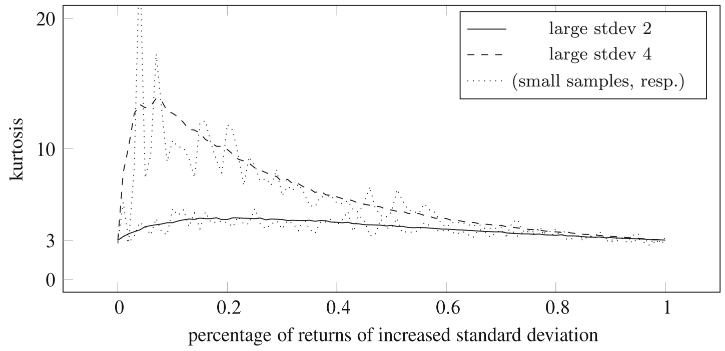

Fig. 17.4 Kurtosis of mixed normals

We first see that samples of size 500 are quite unstable with regard to kurtosis. More importantly, the kurtosis we experience—as exemplified in our artificial simulation—must be interpreted. If it is merely the artifact of vola regime changes, its tail indication might well be dismissed.

How would "real" kurtosis manifest itself? Let's revisit the same returns as in Figs. 17.2 and 17.3, but change their arrangement. Instead of putting all large returns at the end of the series (representing a vola regime change), we could arbitrarily mix them into the overall series, e.g., we could take every 4th overall return from the set of larger ones. First, obviously, the overall kurtosis isn't affected by the reshuffling and remains at 4.83. Yet this time series now is different—we changed its meaning. Now it represents (by our design) no longer a time series with recently increased vola; now it resembles one that frequently, intermittently exhibits shocks that are larger than expected. We can regard this as an actually heavy-tailed distribution as opposed to a mere sequence of two different normal ones. Under this interpretation of our series, and based on its new and known behavior over time, the overall kurtosis becomes meaningful (even though it has the same value).

Figure 17.5 shows what happens to our time series in this reshuffling exercise. Its overall kurtosis, as mentioned, is unchanged. But the behavior of the local kurtosis is very different now: all 40-day windows, and no longer only the right-most ones, now contain some large returns. Note how the median local kurtosis is no longer close to 3, as in the previous Fig. 17.3, but larger at 3.76.

Unfortunately and obviously, the picture is less clear-cut with real-world data. Figures 17.6 and 17.7 show returns and local kurtosis behavior for the Norwegian krone versus euro exchange rate. The local kurtosis is above the normal one in the first year; afterwards, it hovers closer to 3, indicating normality—except for two periods where return spikes (outliers?) cause brief periods of large local kurtosis. Even if we don't dismiss those two return instances as data flukes or political events, the overall kurtosis seems to be too large. Nevertheless, without the luxury of replications available in the artificial setup, where we can dismiss or at least corral in the kurtosis, agreeing on a target kurtosis for our one-off sample feels like an exercise with dubious success prospects.

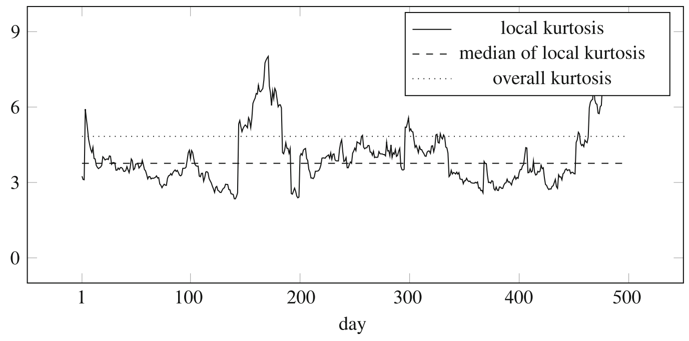

Fig. 17.5 Local kurtosis over time vs. overall one, with shuffled returns

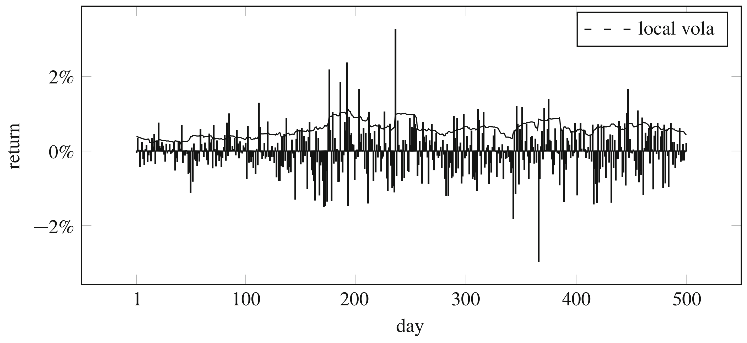

Fig. 17.6 Real returns

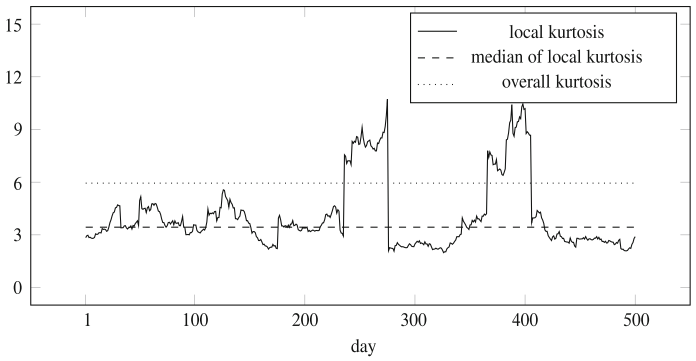

Fig. 17.7 Real local kurtosis over time vs. overall one

Distribution tails are certainly of interest, and an established measure for it is readily at hand with the kurtosis. The very interventionist operation of volatility rescaling may well reduce the overall historical kurtosis, as seen in our first example, and thus ostensibly suppress tails; a Monte Carlo approach applied on top of it would, by imposing normality, annihilate the kurtosis outright. This makes a discussion about it in the context of a VaR model almost inevitable.

What are the approaches that can and should be taken with regard to kurtosis, and how to defend them? By far the worst choice, in my opinion, is to tackle this issue in a conventional manner and try to model the kurtosis in detail, e.g., for each risk factor. Trying to pin down an actual kurtosis is a highly speculative exercise, due to spurious effects arising from vola level changes. In essence, we do not know the kurtosis. But even if one decided to ignore that and just blindly and technically computed the measure and imposed corresponding tails, the practical obstacles would be considerable: separate analyses on 2200 time series would be required; they would have to be repeated periodically in a dynamic VaR setup; and any kurtosis-related tweaks or optimizations would be unsteady as functions of a relatively fickle measure.

Possibly less elegant from a position of mathematical purity but far more practical is it to test the model via the standard backtesting and distribution tests. If the results are unconvincing and can be traced back to the modeling of tails, apply some simple heuristic to compensate for the shortfall; you could, for example, scale up the VaR by some constant factor or inject some fixed, artificial kurtosis as described in Chap. 9. This deliberately forgoes explicit criteria of optimality or kurtosis fits. It avoids lengthy validation discussions, prolonged by the underlying ignorance of any "real" kurtosis. It refuses to pretend to know the unknowable and avoids giving a false sense of confidence. It essentially puts the kurtosis topic outside the model's core realm.

On the other hand, if backtesting and distribution tests are able to plausibilize a model without explicit kurtosis handling, than ignoring kurtosis aspects is by far the preferable choice of action. In fact, one could view such end-to-end tests as strong indications that much of the technically signaled kurtosis is an artifact. We expect many real-world portfolios to perform just fine in a kurtosis-oblivious VaR model, and not only because often the main risk drivers happen to be liquid and quasi-normal assets like domestic interest rates and major foreign currencies. If the tests succeed, the benefits of further kurtosis tweaks surely appear to be marginal, especially in light of disproportionate efforts. In other words: if the distribution tests fit, you must acquit.

To sum up, we propose to neglect kurtosis issues whenever possible and justified by standard test procedures and to defend this decision on the case of equivocal kurtosis values in case of mixed distributions. If not viable, apply some simple catch-all heuristic to ballast the tails. Squash the topic or keep it at arm's length.

Part III

Setup

Context 18

Finance is no stranger to grandiose monikers (my personal favorite has to be the financial crisis's "Master Liquidity Enhancement Conduit"). The same holds true for software engineering and IT, with many a "business service framework" or "disciplined agile delivery" being thrown around. Each IT area continuously breeds forth new languages, customized frameworks, abstraction layers, and paradigms; some are useful, some promising, others opaque, and quite a few short-lived.

Luckily, we can mostly avoid weeding through the various software flavors of the month, since our VaR model is relatively trivial. From an input of market prices and position data, it computes position prices and some aggregate statistical measures. We can also easily break up the calculation into independent parts by, for example, calculating half of the position prices on one computer and half on another; such software is called "embarrassingly parallel." Together, this allows us to use low-complexity processes, cheap and standard open-source software, off-the-shelf support tools, and an unspectacular programming style.

Still, there are a few factors we should be mindful of when designing and implementing our system, as our VaR model does differ from conventional enterprise software like customer relationship databases or ticketing systems:

- The VaR system heavily relies on real-world market data. Such data is nothing like the one used in educational exercises. It is dirty in the sense that it may and will contain outliers and missing values. Data feeds from market data systems are expensive to set up and operate, and data formats are inconsistent across various sources and need to be aligned. A good deal of work thus will be spent on market data and its daily handling.
- The VaR model has more stakeholders than a lot of conventional business software with a perhaps narrower scope or audience. It affects capital requirements, which makes even senior management take a look at its results. Due to legal ramifications, internal audit and external regulators take an interest in its validity and in any changes to it. Model end users—for example traders working under

VaR limits—question its results as it directly affects their work. Accounting wants to reconcile its results and reports, which often originate by wholly other means and via different, historically grown systems, with VaR model results.

The main consequence of this is that changes to an existing VaR model are difficult to put through. You usually need to first obtain the regulator's OK, which will take time. Unexpected, even if correct, changes in the numbers will then trigger a multitude of questions and meetings. New model features, e.g., new methods or risk factors, must be consistently fed through the whole chain of calculation and reporting tools, requiring training efforts and documentation updates. Therefore, changes must be planned well in advance and are often best introduced by first running parallel "shadow" test series in order to obtain the required validation credentials and establish confidence in the upgraded model version's result.

Another effect of this multi-faceted impact of model changes is that the data, at its various stages, often needs to be exchanged with others, e.g., for comparing or plausibilizing intermediate results at various levels of granularity. This strongly hints at using common standard formats that are easy to exchange and process instead of optimized but proprietary and thus cordoned-off ones.

- Due to legal requirements, the system must reliably produce daily results—it can't just take a hiatus for a period of time. This requires backups and fallback strategies in case subsystems fail or data feeds break down. The system also needs to be able to scale: inevitably, the number of positions and risk factors increases; simulation types are added; custom recalculations are requested; and new regulatory requirements are introduced. If, say, a database struggles with the initial system setup's storage requirements, it will crack and collapse not too far down the line.
- Market risk management could be seen as a cost-*only* center, as it does not add to the bottom line (at best it avoids penalty payments). This might make expenses in infrastructure, software maintenance and development, and personnel trickier to justify—especially if the system seems to be running smoothly. But this natural human tendency towards complacency vis-à-vis something not obviously broken must be countered: in an evolving regulatory landscape, the system actually ages as some features become obsolete and new ones are added on top. Cleaning up the system environment from time to time by trimming or re-engineering geriatric parts keeps long-term maintenance and operational costs in check.
- Market risk involves mathematics to a certain (moderate) degree. This can make building the related software unsuited to the standard IT development processes in place. Such processes often rely on business departments specifying their requirements in writing, which is then interpreted and implemented by separate, dedicated IT personnel. This (sometimes enforced) separation is not productive and too inflexible in this context. Instead, quants and risk managers should be allowed to develop directly, since math ill-translates across too many intermediaries. Alternatively, consider embedding programmers into the risk department and let them share home and hearth—desk and definitions—with it.

As a corollary, standard IT guidelines or conventions need to be mendable. Math-affine risk systems might call for at least some tools and software hitherto unfamiliar to the IT department and thus not white-listed—the usage of such components should not be precluded on noble bureaucratic grounds of avoiding tool proliferation.

- It is common practice, and rightly so, to carefully manage and restrict access rights to various system parts containing sensitive data. In the context of a VaR system involving many parties that design or develop models and analyze data at various levels, too restrictive access limitations, however, carry substantial operative costs and operational risks. A more lenient handling of such rights, e.g., allowing developers to work on large and real production data sets instead of small and artificial dummy data sets, facilitates unhampered system interactions and avoids Chinese whispers in unending email threads. Like with organizational borders mentioned before, adapt such technical firewalls to how people work best and not the other way round. Each however well-intentioned separation, be it organizational or technical, ultimately sets up information fault lines that can be just as harmful as they appear to be judicious.

Scope and Workflow 19

In a practical market risk system, we need to calculate several risk measures on a daily basis. Let's briefly describe this target scope first. We, of course, want to determine the VaR for our main portfolios and to provide corresponding backtesting results. As support, we also calculate several partial VaRs (each with their own backtesting). In addition, we calculate the stressed VaR (without backtesting) as well as sensitivities and stress tests.

In practice, you might want to run it all several times, once in a (stable) setup approved by the regulator and once or more often with different model settings. This allows you to flexibly adapt model parameters for internal use or to run test or experimental systems in parallel.

Note that all the above computations usually run thrice: on a technical test system, on a production-affine business test system, and on the actual production system. All in all, this can amount to quite a bit of number crunching.

19.1 VaR Flow

With this overall environment in mind, we can now outline the actual system components in terms of both data and tools. We start off with a basic example that will help us identify and name our components. We use a pseudo command line syntax to describe the various steps necessary.

Each day, to compute the VaR of our portfolios of interest, we need the current positions' PnL vectors as well as the definition of which positions each portfolio contains. The positions' PnL computation, in turn, is based on the scenarios and the positions' characteristics. The scenarios, of course, are derived from the most recent historical market data. We thus start by extracting history, position information, and portfolio definitions from our database:

```
extract_from_db  'hist'  'today'  '520'  >  hist.txt
extract_from_db  'pos'   'today'  >  pos.txt
extract_from_db  'port'  'today'  >  port.txt
```

The `extract_from_db` is our tool or command to access the database. The `'pos'`, `'today'`, etc., represent special instructions or *flags* or *command line parameters* to our tool. For example, `'pos'` tells the tool to export position data. The `'today'` instructs it to export the most recent data; by using explicit past dates instead, we could export older entries. The ">" writes the output to the subsequently named text file. Let's examine the three steps above:

- After the first extract, the newly created file `hist.txt` contains the time series (of values, not returns) of our 2200 risk factors up to and including the latest market snapshot. We require the last 520 days of data. The file `hist.txt` is thus a 2200×520 matrix.
- The file `pos.txt` defines each of our 10^6 positions. Assume, for now, that it contains one line for each position, e.g.,

  ```
  position_42|cashflow|1000000|USD|20301224
  ```

  —more complex positions of course require a more extensive description.
- Lastly, the file `port.txt` describes how the portfolios are made up. For simplicity, consider each line to define one portfolio and to contain that portfolio's name followed by its positions, e.g.,

  ```
  portfolio_1 position_18 position_42 position_39
  ```

  (Alternative, hierarchical definitions are of course possible as well.)

We can now generate scenarios from the historical market data:

```
calc_scen  'var'  hist.txt  >  scen_var.txt
```

The file `hist.txt` we created before now serves as an input to the scenario generator `calc_scen`. The produced output file `scen_var.txt` contains 2200 rows of 1001 scenario values. The last, right-most column in `hist.txt` is identical to the first, left-most column in `scen_var.txt`—it is the current market snapshot/scenario.

The scenario generator may be used to create different types of scenarios; the first flag, here set to `'var'`, determines the desired type.

We are now able to compute the positions' PnL vectors:

```
calc_pnl  pos.txt  scen_var.txt  >  pnl_var.txt
```

The `pnl_var.txt` is the largest file we produce: each of its 10^6 lines contains a 1000-sized PnL vector—one for each position. It stores one billion values.

Finally, we can aggregate the positions' PnL vectors to portfolio vectors and compute some summary statistics along the way:

```
aggregate  port.txt  pnl_var.txt  >  aggr_var.txt
```

This `aggr_var.txt` file is much smaller. There's one line for each of our portfolios, and each contains a portfolio PnL vector as well as summary stats like the VaR or the ES.

We finally store the results in our database, but, to save space, usually only the summary statistics (this behavior is controlled by the first flag, here set to a `'var'` storage mode):

```
store_to_db  'var'  'today'  aggr_var.txt
```

19.2 Backtesting Flow

We now perform the backtesting for yesterday's VaR estimate. For this, we must compute the PnL actually experienced, based on the last two market data snapshots in our history. Note how we don't take today's positions and portfolios, but yesterday's:

```
extract_from_db  'pos'   'yesterday'  >  pos_bt.txt
extract_from_db  'port'  'yesterday'  >  port_bt.txt
```

Using the last two values of each time series, i.e., the last two market snapshots, we create a backtesting scenario file (the `'today'` is correct here):

```
extract_from_db  'hist'  'today'  '2'  >  scen_bt.txt
```

Two scenarios will result in two prices and ultimately in one single PnL value for each position:

```
calc_pnl  pos_bt.txt  scen_bt.txt  >  pnl_bt.txt
```

We then aggregate this by yesterday's portfolio setup:

```
aggregate  port_bt.txt  pnl_bt.txt  >  aggr_bt.txt
```

This simply sums up the single PnL values of the positions in each portfolio.

The results are then once again stored in our database. The date identifier must correspond to yesterday, since the backtesting PnLs correspond to yesterday's VaR calculation. Here, and triggered by `'bt'`, one backtesting PnL value is stored for each portfolio:

```
store_to_db  'bt'  'yesterday'  aggr_bt.txt
```

19.3 Sensitivity Flow

To compute sensitivities, we have to generate the respective scenarios from the last market data snapshot:

```
extract_from_db  'hist'  'today'  '1'  >  hist_curr.txt
calc_scen  'sens'  hist_curr.txt  >  scen_sens.txt
```

This generates the 2200 × 2201 matrix of sensitivity scenarios. (Here, or in the backtesting above, we could have just used the already exported `hist.txt` and its last column(s) as input, but for clarity we want to restrict the data scope to strictly required values.)

The PnLs obtained via the following command are our position sensitivities:

```
calc_pnl  pos.txt  scen_sens.txt  >  pnl_sens.txt
```

We aggregate via

```
aggregate  port.txt  pnl_sens.txt  >  aggr_sens.txt
```

With VaR results we usually only store the summary stats for each portfolio. Here, we want to store all the sensitivities contained in our PnL vector (or at least all the non-zero ones, again to save space). We instruct our tool to do that with an appropriate flag:

```
store_to_db  'sens'  'today'  aggr_sens.txt
```

19.4 Stress Test and Stressed VaR Flow

In alignment with the previous workflows, we can generate custom stress test scenarios (`scen_stress.txt`) as well as scenarios for the stressed VaR (`scen_svar.txt`).

The latter involves using the returns of a pre-determined period of history and applying them to the current market prices to generate scenarios. The period to use is usually determined once a year; it is simply the past period with the largest VaR given the portfolio at the time of the stress period selection. Its location in time should remain fairly stable, though, as it tends to fall into the same crisis.

Given a pre-defined stress period, e.g., a 1-year period ending in June 2008, we need to use that period's returns along with the current market snapshot to generate stressed VaR scenarios. The whole process becomes:

```
extract_from_db  'hist'  '20080630'  '251'  >  hist_svar.txt
extract_from_db  'hist'  'today'     '1'    >  hist_curr.txt

extract_from_db  'pos'   'today'  >  pos.txt
extract_from_db  'port'  'today'  >  port.txt
```

```
calc_scen  'svar'  hist_svar.txt  hist_curr  >  scen_svar.txt

calc_pnl  pos.txt  scen_svar.txt  >  pnl_svar.txt

aggregate  port.txt  pnl_svar.txt  >  aggr_svar.txt

store_to_db  'svar'  'today'  aggr_svar.txt
```

As we usually avoid doing the volatility rescaling for the stressed VaR, 251 historical scenarios are sufficient to generate 1-year return time series of size 250. Also, if the number of scenarios generated by mirroring is deemed too small, the `'svar'`-type scenarios can also be generated via the Monte Carlo modification.

19.5 Partial VaR Flow

A special case are partial VaR calculations. We could, e.g., generate separate scenarios like `scen_var-fx.txt`, where all but the FX risk factors remain constant and identical to their current market values. It is more efficient instead to just modify `calc_pnl`: after reading in the scenario file, it simply overwrites some risk factors' scenario values with their current market values on the fly. This way we avoid separate, largely redundant scenario files:

```
calc_pnl  'rf-filter:FX'  pos.txt  scen_var.txt
          >  pnl_var-fx.txt
```

19.6 Naming

All the above results are usually derived under different model setups and both for regulatory and strictly internal use. They are also typically calculated on various independent test and production systems. The file names should reflect that.

As an example, here are the names that denote the most important results: the ones from the production environment (`p`), used for regulatory reporting (`reg`), and under the main model parametrization:

```
p_reg_main_scen_var.txt
p_reg_main_pnl_var.txt
p_reg_main_aggr_var.txt
```

The regulator might demand additional, auxiliary model parametrizations for cross-checks, e.g., one with larger 30-day windows in the vola rescaling:

```
p_reg_lv30d_scen_var.txt
p_reg_lv30d_pnl_var.txt
...
```

Internally (`int`), we might want to use a Monte Carlo setup with more scenarios. The resulting files are generated on some test system; here are, as examples, the names of some partial VaR files generated on test system `t2`:

```
t2_int_mc_scen_var.txt
t2_int_mc_pnl_var-ir.txt
t2_int_mc_aggr_var-ir.txt
```

If all this sounds rather dull, that's because it is. But keeping your namespace clean from the start will greatly help in the long run. Proper naming goes a long way towards a sound system design.

19.7 Encapsulation

All the files are produced daily. It proved useful to encapsulate each day's data into one directory, e.g., `p_reg_20170601`. This directory contains all regulatory files of a given day created on the production system. The date should be the latest market snapshot date (sometimes also referred to as position date or business date) and not the subsequent date for which the VaR is estimated—that's because all model end users will usually refer to the position date. (I know this because I stupidly opted to use the VaR prediction date for this directory name some 8 years ago. Names and conventions, alas, often die a slow death, so this misguided date tag is still in place and requires regular asterisks in our emails.)

It is useful to copy, into each day's directory, the current version of the tools used for the calculation. At first, this might seem redundant—we have both

```
p_20160108/tools/calc_pnl
```

and

```
p_20160109/tools/calc_pnl
```

—two separate tool copies which most of the time will be identical. Yet as the software evolves, it will go through various versions and updates. Those could be tracked separately, of course, in order to recall which day was computed under which version (and to be able to reenact the whole computation). Yet copying the tool and wrapping it up with the data it belongs to is just much less error prone. It helps that the tools' sizes are absolutely insignificant really in the avalanche of data produced daily.

Implementation 20

With the overall workflow outlined in the previous chapter in mind, we now zoom in to the individual system components. We first cover the involved tools and their implementation and later focus on data sets and their structure.

As for the tools, their functionality is best split up into separate and manageable components. The Linux operating system serves well as a stable foundation, as it explicitly supports such modularity. Linux also provides a host of standard functionality out of the box, which our tools can beneficially leverage. Newly developed tools together with operating system ones are best thought of as a Lego set, with autonomous bricks being combined into a system ensemble that is bigger than the sum of its parts.

20.1 Generating Scenarios

We start off with the scenario generator `calc_scen`. This is a slender component with small data input and output, and it requires little computational effort. We therefore implement it in Python and one of its most popular numeric libraries, NumPy. The whole scenario generator can be implemented in a handful of lines of code, as shown in this working code snippet (in this example, using log returns):

```
for v in HIST:                                      #v is a risk
                                                    factor's history
    s0 = v[-1]                                      #current asset
                                                    value

    r_raw = np.log(v[1:] / v[:-1])                  #519 raw log
                                                    returns

    lv = np.zeros(500)                              #500 local volas
    for i in range(500):
        lv[i] = np.std(r_raw[i:i + 20], ddof=1)
```

```
r_resc = r_raw[-500:] / lv * lv[-1]      #target = last
                                         local vola

r = np.append(r_resc, -r_resc)           #mirror

s = s0 * np.exp(r)                       #1000 scenarios

print("%f|" % s0  +  "|".join(str(x) for x in s))   #1001
                                                    outputs
```

A few additional lines of code won't hurt if we want to explicitly handle the following special cases:

- Historical time series may contain missing values. Fill such gaps with previous-day values, log each instance as a warning, and check the market data source for underlying problems.
- Automatically scrub for clearly nonsensical outliers, e.g., interest rates above 500%, possibly due to erroneous manual data input of 6 instead of 0.06. Don't simply cap/floor such values, but again replace them with previous-day values, log, and investigate.
- Stale time series can exhibit zero local volatilities. Floor local volas at 10^{-12} and track the issue down.
- Make sure your return types (e.g., logarithmic or square root ones) are valid operations on your time series; logarithmic returns, for example, need strictly positive values. If such returns are used for interest rates (which can become negative after all), shift the whole series up into valid territory, create scenarios, and then shift those scenarios down again. This operation is less dubious or dangerous than it might appear at first, mainly because very large shifts make the returns behave asymptotically like absolute ones.

  A shifting logic for interest rates that proved to be workable is to automatically shift up a whole time series if its minimum value is below 1%, by so much that after the shift, the new minimal value equals 1%. (This makes sure that times series with negative values are shifted up; that time series with large positive values remain unchanged; and that time series between those extremes, i.e., those that are just slightly positive, are treated consistently and avoid any unbecoming gap behavior.)
- When generating square root-based scenarios, make sure they are not floored at (or reflected from) 0; shift the time series further up to let the scenario generation operate in a valid range. As always, log and track.

Python, with its excellent helper data structures and concise matrix notation, allows us to implement this with minimal overhead and in few lines of code to write, debug, understand, and maintain. Type safety and code organization is of secondary importance due to the small overall size of this tool. (By the way, NumPy's vectorization support is also very efficient from a numerical point of view.)

20.2 Computing PnLs

Once the scenarios are generated, the next step is to price the positions under these scenarios and to compute the PnL vectors with the pricing tool `calc_pnl`. Computationally, this will be by far the most demanding step, as you need to compute prices for each of the 10^6 positions under each of the 1001 scenarios. Furthermore, you will have to repeat this for each type of VaR simulation you want to run, and again on each test or production system. Also, the code complexity here will be far higher than that of the scenario generator, especially if you have to price sophisticated types of financial products. (While the topic of pricing is outside the scope of this book, Appendix B tries to hint at some of its fundamental concepts.)

This makes C++ the language of choice for implementing sophisticated pricing functionality:

- A common adage is that "C++ is fast," i.e., that it allows us to create highly efficient tools. Yet as the VaR calculation usually takes place overnight, one might ask if this execution speed is required—after all, there are plenty of hours available, and a slower tool (written in another language) might just also be able to make it through, especially if executed on subsets of positions in parallel. However, this neglects a fundamental use case: the recalculation of whole VaR time series, for example, when testing a different model parametrization. If an overnight run requires a run time of 8 h on your infrastructure, a year's worth of recalculations will take 2000 h—hardly interactive and prone to discourage you from testing new model features on sensibly large time windows.
- C++ is sometimes considered an old, complex, and difficult-to-learn language, giving rise to frighteningly difficult-to-maintain source code. The language Java is preferred in many setups, more for its user-friendliness than its computational efficiency. Now, it is possible to write efficient Java code, but there's a catch: C++ will create fairly fast tools if you don't mess up; Java will do the same, but only if you don't mess up anywhere. Just neglecting one bottleneck, maybe a slow data structure or a time-consuming memory management feature, will slow your Java code down. The level of expertise required to avoid those Java fallacies is entirely comparable to that of the sound handling of C++ quirks. Finally, many "dangerous" C++ features can simply be avoided or forbidden, as do the crack C++ programmers working on the Firefox web browser.[1]
- Finally, C++ provides object orientation for organizing your code in modular components, static type checking for early detection of suspicious expressions, a host of numerical libraries, and very granular performance optimization tools.

[1]Mozilla, the Firefox developers, discourage using quite a few C++ features in https://developer.mozilla.org/en-US/docs/Mozilla/C++_Portability_Guide. I don't agree with all of them, especially those concerning the C++ standard library and exceptions, but would certainly want to avoid C++ templates.

Even if you have successfully implemented a fast pricer, you still want to parallelize the calculation. You can, for example, compute the VaR and the stressed VaR in parallel or, in a more fine-grained approach, split up your positions and compute various types of PnL vectors for each of those subsets concurrently. Luckily, the computation of PnLs can be considered to be what is called "embarrassingly parallel," as each subset of positions is independent and can be treated completely on its own.[2]

20.3 Aggregating

The PnL aggregation with `aggregate` is again a small, well-defined component. It can readily be implemented in Python, which supports fast vector operations and suitable data structures like dictionaries. Alternative setups are also possible, e.g., in-memory databases or vector databases (standard databases add too much overhead in dealing with vectors). Such extra layers, however, are not strictly necessary and can be avoided for the sake of simplicity.

Aggregation serves as a good example of how tools can be combined and leveraged. Recall the input and output of `aggregate`:

```
aggregate  port.txt  pnl_var.txt  >  aggr_var.txt
```

The small input file `port.txt` contains the portfolio setup or composition; the small output file `aggr_var.txt` contains the resulting PnL vectors and summary statistics for each portfolio. The second input, `pnl_var.txt`, is by far the largest file. A sensible idea is to zip it to save disk space. Now, one could explicitly unzip the file each time it is required or allow `aggregate` to read zip files directly. But the former would add a step, while the latter would require adding zip functionality to our tool. There is a third, more elegant way: make `aggregate` support the operating system's *pipe*, a way of routing data directly from one tool to another:

```
unzip -p pnl_var.zip | aggregate  port.txt  -  > aggr_var.txt
```

The standard command `unzip -p <filename>` unzips the file and outputs its content; the pipe operator "`|`" then routes this output to the following tool in this chain; finally, the "`-`" denotes which file input (the second one) should be superseded with the pipe's throughput.

We also immediately gain support for multi-file handling, which therefore doesn't have to be encoded explicitly into `aggregate`. The following command

[2]This calls for using simple *parallel processes* instead of the more involved technique of *multi-threading*.

uses the wild card "⋆" to unzip all zip files in the current directory and to sequentially pass their contents on for aggregation:

```
unzip -p *.zip | aggregate port.txt - > aggr_var.txt
```

20.4 Accessing the Database

The database access tools (`extract_from_db`, `store_to_db`) embed the VaR system in the overall IT landscape. They mainly retrieve reference input data and centrally store the relevant end results. Such tools are often implemented in Java due to its traditionally strong support for database-related operations. It is no coincidence that Oracle, a major database vendor, has acquired Java and actively promotes its development. Nothing to see here; move along.

20.5 Coding Guidelines

Coding guidelines or conventions can help promote some degree of consistency and thus readability and maintainability of the source code. It's reasonable to rely on (and slightly modify) standard guidelines like Google's.[3] Enforcing them too strictly or endlessly debating their relative merits is best avoided. On a general level,

- strive—first and foremost—to carefully name and design your data structures;
- separate data and functionality into de-coupled modules;
- only *then* worry about the algorithms, which are secondary;
- write your code to minimize the time it would take a graduate student to understand it, and then opt for solutions with fewer lines of code, but only if clarity is preserved;
- comprehensively check for operation validity, error return codes, and any kind of program exception—no error indication must be left behind;
- extensively log error occurrences, warning conditions, and all extraordinary events (e.g., if a local volatility is floored);
- avoid doing the same thing in two different ways or re-implementing stuff that already exists (e.g., zip functionality);
- let your tools fail fast and especially avoid overriding guesses (e.g., by misguidedly trying to gracefully interpret typos in command line parameters);
- avoid any optimization unless you can prove that the speed gained more than offsets the additional complexity and impaired code readability.

[3]https://google.github.io/styleguide/cppguide.html.

Feel free to skip the following minor, technical recommendations with regard to C++ in our specific and narrow pricing setup (and look up the jargon only once needed):

- Avoid or minimize the use of C++ templates. They are useful in many cases for avoiding code duplication, but they are often overkill. They are also prone to generate difficult-to-interpret compiler error messages.
- Extensively use the `const` modifier to denote unchanging arguments and side effect-free member functions.
- Do use `break`/`continue` statements if a loop's readability is improved. Also, do use the `goto` statement in such instances, e.g., at special loop exit junctures.
- Avoid complex constructor/destructor logic, especially in combination with C++ error exceptions.
- Avoid enums to ease debugging; rely on ints or, often even better, strings instead.
- Use expressive strings to signal errors or warnings instead of error codes. The error message "`E:permission_denied:test.txt`" is more expressive than "`13`". The whole mapping logic from error codes to their meaning can be avoided or streamlined, along with its maintenance and documentation.
- Avoid unsigned integer data types. They are often unsuited to expressing exception indications, inevitably lead to overflow or underflow errors, and cause headaches with integer promotion and function/operator overloading.
- Avoid the float data type; use double instead. Floats require you to fret about floating-point math, and their performance behavior is difficult to predict as it highly depends on system specifics (e.g., using float operations can potentially and unintuitively even cause slower run times).
- Avoid fat external libraries. Libraries like Boost are very tempting to rely on, but often only a very small subset of features, like command line parsing, is actually used, at the expense of burdening your project with a host of unneeded ballast and an additional and avoidable compile-time dependency.
- Use static instead of dynamic linking. This gives you neatly encapsulated tools without external run-time dependencies; such tools are therefore easy to deploy.

20.6 File Formats

We mainly use plain text files for most input and output artifacts. They are easy to introspect and check, to exchange with other stakeholders or third-party software, and to process in a Linux environment via pipes and standard support tools. The files represent neat boundaries between tools and facilitate a clear separation of responsibilities between developers—they essentially double duty as contracts. File access permissions provide a lightweight approach to manage user access rights.

Now, files *could* of course be replaced with higher-level artifacts like vector databases, XML or binary files, or in-memory data structures, yet those alternatives are typically more complex and less transparent while adding an additional software layer to set up and maintain.

Before we describe the various data formats in our VaR setup in greater detail, a few general remarks about file formats are in order:

- Avoid rare file encodings and stick to UTF-8/ASCII.
- Avoid spaces in file and directory names.
- Within files, avoid using delimiters that might be interpreted differently depending on an operating system's locale, i.e., country and language settings. (The comma, for example, is the German decimal separator.) Whitespace delimiters impede the spotting of missing values, especially at the lines' ends. Best always use the pipe character "|".
- Regardless of your country, always use the English encoding of decimal points; avoid thousands separators entirely. Ideally, only deploy English versions of operating systems and office productivity software.
- Always encode mathematical values properly, i.e., use 0.02 to denote 2%. Only in the final reporting steps (e.g., pdf files sent to the board) should you reformat the values to an appropriate representation.
- Encode dates as integers in the YYYYMMDD format, e.g., "`20181224`". This is unambiguous across locales and conveniently exhibits the appropriate sorting behavior. Also, Excel's auto-formatting is gagged.
- In intermediate and result files, there is no need to ever contemplate allowing missing values or not-a-number codes like "`nan`". Especially as we want to combine, Lego-like, different tools (which may react differently to such occurrences), handling such special cases consistently becomes soon highly error-prone. The initial database export of historical data might well contain missing values; they should be recorded in error logs and defaulted to previous-day values. But if later, for example, a position's pricing function fails to compute a price, simply record it in the error log and dismiss the whole position. This way, no missing or invalid values are ever propagated through the tool chain.
- Avoid the tilde character "~" in IDs or names; Excel's various functions handle it inconsistently (I am looking at you, `VLOOKUP` function).

Let's now have a look at the various files we require.

Historical Data The file `hist.txt` contains the risk factors' history:

```
#risk-factor|20150101|20150102|...
...
FX-EUR|1.11|1.12|...
...
IR-EUR-Y10|0.012234|0.012344|...
...
```

Scenarios The scenario files have a similar structure; their first comment line describes the columns.

The standard VaR scenarios are stored as follows:

```
#risk-factor|S_0|S_1|...|S_1000
...
FX-EUR|1.11|1.14|...|1.09
...
IR-EUR-Y10|0.012234|0.012359|...|0.012199
...
```

Risk factor names are used to label the columns of sensitivity scenarios:

```
#risk-factor|S_0|FX-AUD|FX-CAD|...|IR-ZAR-Y20|IR-ZAR-Y30
FX-AUD|1.34|1.340134|1.34|...|1.34|1.34
...
IR-ZAR-Y30|0.0820|0.0820|0.0820|...|0.0820|0.0821
```

Stress scenarios contain the stress tests' descriptions:

```
#risk-factor|S_0|NineEleven|...|Apocalypse
...
FX-EUR|1.11|1.24|...|201.09
...
IR-EUR-Y10|0.012234|0.022359|...|0.341199
...
```

Positions The positions are described in the file `pos.txt`. Here is a simple example of a future cash flow:

```
...
pos_id_17|subpos_id_0|cashflow|1000000|USD|20181224
...
```

The first entry is the position's unique identifier or ID. But why include an additional sub-position ID? This is useful for handling complex positions, which can often be broken down into simpler ones. Here is a 2-year fixed rate bond with annual payments, represented as several cash flows:

```
...
bond_21|coupon_0|cashflow|10000|USD|20171224
bond_21|coupon_1|cashflow|10000|USD|20181224
bond_21|repayment|cashflow|1000000|USD|20181224
...
```

This is considered one single position. The pricing tool `calc_pnl` can aggregate the result vectors of the three sub-positions into one result vector. This essentially optimizes output file sizes while still retaining a position-level granularity. For our typical 10^6 positions, our position file actually contains about 10^7 sub-positions.

We alternatively could have encoded the bond into a single line, with a more complex interface. The three new fields denote payment frequency, coupon rate, and day count convention[4]:

```
...
bond_21||bond|1000000|USD|20181224|annual|0.01|ACT/ACT
...
```

If you have the capability to explicitly denote which deals are meant to hedge each other, consider adding a field that labels deal sets pertaining to a hedge. Depending on your portfolio setup and the quality of the hedge, you may be able to speed up the overall computation by not pricing perfectly hedged deals at all. At the very least, you can beneficially filter out those deals' noisy results in any later analysis exercises.

Position identifiers can be put to good use. While some systems just use numbers as identifiers, a more expressive name can save separate and time-consuming lookups into a position's meta-information. Consider encoding some relevant information into an ID, for example:

```
bond_desk:gvnt_bond:austria:438
```

PnL Vectors The result of `calc_pnl` are lines containing the positions' PnL vectors. The price under the current scenario $\mathbf{S}_0$ is given in the second field[5]:

```
...
bond_21|980000|1343|-1734|...|2504
...
```

Given the current price and the scenario PnLs, we could of course always recreate scenario prices, but that is rarely of use. Besides being our main measure of interest, PnLs also have the nice property of using up less disk space (e.g., about 4 instead of 6 digits above).

Be careful with fractional digits: for VaR calculation results, you can ignore them and save disk space; for sensitivity PnLs, several decimal digits are appropriate for signaling non-zero values:

```
...
some_small_position_27|1200|0|0|...|0|-0.3563|0|...|0
...
```

The standard floating-point output behavior of C++'s `printf` function, which uses the fewest number of digits needed to encode a number, is adequate here. In Python, use the format specifier "`%g`".

[4]Day count conventions define how time periods are handled in light of unevenly-sized months, leap years, etc.; `ACT/ACT` stands, e.g., for actual/actual, a convention that uses the actual number of days to measure time-interval lengths.

[5]Note that the sub-position ID is no longer part of this output.

Aggregated PnL Vectors The aggregated portfolio PnL vectors also contain summary statistics:

```
...
portfolio_27|12580600|-41356|20654|...|36776|var:010|-48546|
                                                   es:025|-49657
...
```

The summary statistics are stored as tuples of type and value; this way you can flexibly add or discard statistics. Consider adding to the usual `var:010` and `es:025` at least `stdev`, `avg`, `median`, `kurt`, `var:050`, and `var:990`.

Notice a detail in the proposed format: the conditional expected shortfall does not appear. That's of course because it is computed for a position or sub-portfolio *with respect to* an overall reference portfolio. The aggregation tool should therefore provide an explicit, alternate mode for this task:

```
aggregate 'ces' pnl_var.txt > aggr_ces.txt
```

This computes the cES of every entry in the PnL input versus the sum of all entries (no portfolio info is therefore needed) and outputs a `ces:025:all`|N tuple for each input line.

(Another useful tool option could trigger the output of either PnL vectors, summary stats, or both, since for sensitivity and stress aggregations the summary stats are not needed. Alternatively, just compute but ignore them.)

20.7 Optimizations

Optimizations should be applied judiciously and only if the gained benefits outweigh the increased complexity and maintenance effort. We will take a look at some of the rather obvious ones.

Approximate Pricing The VaR is based on PnL values, i.e., changes in the positions' prices. This has a fortunate consequence for our pricing step, which, recall, should be speedy as it needs to be executed millions of times each day. We can often get away with using fast, approximate pricing functions instead of slower, more precise ones because a pricing error or bias ϵ often roughly cancels out in our PnL view, in the sense of $\Delta p_i = p_i + \epsilon - (p_0 + \epsilon)$.

Efficient Repricing The calculations of sensitivities, stress tests, and partial VaRs only change a subset of risk factor values in each scenario. If a position does not depend on those changed risk factors, it should not be explicitly repriced but immediately be given the price of scenario 0.

This can save valuable computation time at a minimal implementation overhead. The speedup can be massive, e.g., for a sensitivity calculation, 2 full pricings (in the market and in one sensitivity scenario) might be sufficient instead of the full 2201 ones of a careless implementation. The effect becomes especially noticeable for complex product types, which might require time-consuming methods like trinomial trees or separate Monte Carlo simulations.

Binary File Formats Most files used in our VaR setup are small; the only exception are the large PnL result files, whose size makes them an optimization target. One could store them, e.g., in a binary format using essentially a matrix of double values (of 8 bytes each).

Now, depending on how your storage system is connected to your processing units, access to binary files might be faster. This is less pronounced in the pricing stage, where the time spent on outputting the results is relatively small compared to the actual pricing. In the aggregation step, however, much time is wasted converting string input to double values—this step could profit the most.

Yet binary file formats have several drawbacks. You need custom-made tools for inspecting those files, and many end-users will likely need new know-how or help in dealing with those files. And then there is the issue of file size, which impacts your archival strategy and horizon; the file sizes of VaR pricing results of 10^6 positions are exemplified in Table 20.1.

The text files are smaller than their binary cousin because many values in them (even accounting for the additional delimiter) are likely to use up less than 8 bytes. PnLs, as mentioned previously, take up the least space.

What if we were to compress or zip the files? Zipping the binary file usually leads to few gains if the original floating point values are exported as-is—they behave much like random numbers or noise. Yet can we truncate the decimal digits even in this binary export—this makes the double values contain more zero bytes and their pattern more regular and thus susceptible to compression. The results are given in Table 20.2.

Using zipped PnL text files seems to be a reasonable sweet spot between small storage size and simplicity of use. No additional custom layer is required, since the zip layer is standard. And for relatively slow network connections, the additional unzip step doesn't even have to impair the overall run time—it might actually speed up the whole process.

Table 20.1 Files sizes of pricing results

Result file type	Size in GB
Prices or PnLs as binary	8.0
Prices as text	4.3
PnLs as text	2.6

Table 20.2 Files sizes of compressed/zipped pricing results

Result file type	Size in GB
PnLs as binary, original floating-point values, zipped	7.7
PnLs as binary, decimal places truncated, zipped	0.9
PnLs as text, zipped	0.7

There is one part of the VaR calculation that can benefit from binary formats, though. When pricing positions, the tool `calc_pnl` must read in scenario files; in a parallelized setup, this happens many times over. Providing the scenario files in binary format eliminates a substantial string-to-floating-point conversion overhead.

20.8 Metal

Finally, let's examine the question of the actual computer hardware the system should run on. Basically, you could run all computations in-house on your local system, typically several servers with multiple processors (called a *grid*), or you could choose to run them via some cloud provider like Amazon or Google.

The VaR model described above seems—technically—tailor-made for cloud computing. The input/output sizes are relatively small (positions, history, and aggregated results), so bandwidth should not be an issue. The calculations are highly parallelizable. Peak computation requirements, like recalculating whole VaR time series, can be met with flexible hardware allocation. The downtime risk with large-scale cloud providers is probably smaller than with your own system.

What might balance the scale towards hoarding local hardware? In theory, once your model is stable, time-consuming recalculations of whole time series should become less frequently needed. And if your hardware is scaled to your positions and simulations, it will be used to capacity most of the time, which narrows the cost gap to a cloud solution. Doing your calculation off-site would probably require some position obfuscation (e.g., normalizing all positions' nominals to 1, encoding less information into deal and portfolio names, etc.), adding an extra (however thin) software layer. Internal policies or your local regulator might discourage or outright prohibit putting confidential information off-site. And even large and entrenched companies like the current cloud players have been known to vanish or to cease operations in some jurisdictions.

Still, it seems a stretch to deny that number crunching is likely to become commoditized like electricity. Raw computations need not heat up your own server rooms. It should be feasible to offload major parts of the computation to a cloud provider while retaining some base capacity for core regulatory purposes, backup, archival, and custom analysis exercises. An adventurous IT department and resourceful compliance officers will concur.

Part IV

Wrap-Up

Conclusion 21

Here are a few of the glorious blunders I committed that will remain on my head. I first thought it wise to output sorted PnL aggregation vectors (like destroying information is a good thing). I also proposed to add an additional column of zero to the PnL results for some “future use” as special indicator flag—a use that never materialized and now requires constant nudging when we have to explain our file formats. I unthinkingly used the VaR prediction date as label for our daily results, which is just plain stupid—everybody will always refer to the previous, close-of-business date. Each email now requires a qualifier, and each new hire will at least once perform an analysis of the wrong data. (The latter two imbecilities are actually still in place, ingrained and phlegmatic like a glacier.) Hopefully, this account helps you avoid similar pitfalls.

I have tried my best to outline a minimal VaR system that works. I believe that the model can and should be kept simple, that data rules in all its imperfection, and that it takes a band of many with different backgrounds to nourish our handle on risk every day. Here’s then to clarity, humility, and concert.

Do not hesitate to contact me at `martin@value-at-risk.com` with your questions and feedback and my bugs and typos.

There’s nothing left for me but to wish you godspeed with *your* endeavors.

M. Auer, *Hands-On Value-at-Risk and Expected Shortfall*, Management for Professionals, https://doi.org/10.1007/978-3-319-72320-4_21

Statistics 101 A

Dealing with uncertainty—experienced as randomness—is fraught with limitations, yet we can get at least *some* formal handle on it.

We can, first, simply observe random outcomes repeatedly and essentially count what happens. On 1000 tosses of some coin, we might observe, e.g., 104 head and 896 tail outcomes. We may interpret the frequencies of occurrences (104/1000 and 896/1000) as indicators of the likelihood of the two outcomes.

We also conclude that this particular coin does not seem to be very fair—that's because we have an idea of how a coin ought to behave. We intuit the chance or *probability* of observing either head or tail as about 50%, in the sense of counts or frequencies we would expect to see. With that idea of a fair coin in mind, we would not have been surprised to see, say, 495 occurrences of head. Here we deduce: this coin is rigged.

These two approaches of (i) watching and counting and of (ii) thinking and inferring are referred to as statistics and probability theory, respectively. They most often work in tandem. In a poll, for example, we count the responses of a small set of people to estimate some overall opinion; by making some assumptions about the nature of the involved uncertainty, we can then try to infer the confidence we can put in our estimate.

A.1 Random Variables and Probabilities

To formalize our view on randomness, we start off with the concept of a so-called *random variable*. Think of it as an entity or device that produces one specific output: a single number. It simply selects one number out of many, by chance. How likely a given number is bound to occur is governed by probabilities.

The random variable X describing a die, for example, can result in one of six numbers $1, 2, 3, 4, 5$, or 6, each with a probability of $\frac{1}{6}$. This is an example of a *discrete* random variable.

We can also inscribe larger numbers on our die, e.g., $10, 20, 30, 40, 50$, and 60. This pimped die will help us think of how to label stuff. The first outcome, i.e., 10, is called x_1, and its probability of $\frac{1}{6}$ is variously called p_1 or $p_{X=x_1}$. Overall, x_i is the outcome $10 \times i$, with $p_i = \frac{1}{6}$. (The standard die, where x_i corresponds to the outcome i, unhelpfully blurs name/index and number/outcome.)

First and obviously, the probabilities involved must always sum up to 1:

$$p_1 + \cdots + p_6 = \sum_{i=1}^{6} p_i = \sum p_i = 6 \times \frac{1}{6} = 1 = 100\%.$$

The probability of observing an outcome larger than 25 is

$$p_{X=x_3} + p_{X=x_4} + p_{X=x_5} + p_{X=x_6} = p_3 + p_4 + p_5 + p_6 = \sum_{i=3}^{6} p_i = \frac{4}{6} = 66.6\%.$$

When throwing the die twice in a row, the probability of observing 10 followed by 50 is

$$p_{X=x_1} \times p_{X=x_5} = p_1 p_5 = \frac{1}{6} \times \frac{1}{6} = \frac{1}{36}$$

—unsurprisingly, as there are 36 possible combinations.

Next, consider a spinning top like the one in the movie Inception. After a spin, once it falls and stands still, its handle will point in an arbitrary, random direction—this angle X is a random variable between 0 and 2π. It has uncountably many outcomes (provided we can measure the angle arbitrarily precisely), which makes X an example of a *continuous* random variable. No outcome or angle x is more likely than any other. This immediately means that we cannot assign a positive probability to an individual outcome—if we used even the smallest such probability ϵ for this, the probabilities could never sum up to 1 (because, well, $\infty \times \epsilon > 1$).

We instead capture the involved probabilities via a function $p(x)$, with $p(x) = \frac{1}{2\pi}$ for x in $[0, 2\pi]$, and $p(x) = 0$ otherwise. Why so? Well, this makes sure that the whole area under $p(x)$, the rectangle $2\pi \times \frac{1}{2\pi}$, equals 1. We can now interpret slices or partial areas over outcome ranges as probabilities. For example, the probability of X falling between 0.12 and 0.25 is the corresponding (in this case, rectangular) area $(0.25 - 0.12)\frac{1}{2\pi}$.

The function $p(\cdot)$ is called *probability density* function. It need not be constant, just positive and covering an area of 1. Slice areas or probabilities are then generally expressed as integrals, and the probability of X falling between a and b is

$$\int_a^b p(x)\, dx.$$

We let this sink in using our example. The probability of *any* outcome occurring is, in our case,

$$\int p(x)\,dx = \int_{-\infty}^{\infty} p(x)\,dx = \int_0^{2\pi} \frac{1}{2\pi}\,dx = \frac{1}{2\pi}x|_0^{2\pi} = 1 = 100\%.$$

In a clockwise arrangement, the probability of a lower-right or south-east outcome, i.e., that X lies between east ($\pi/2$) and south (π), is the plausible

$$\int_{\pi/2}^{\pi} \frac{1}{2\pi}\,dx = 25\%.$$

So in the continuous case, we only ever really deal with outcome *ranges*. The probability of a specific outcome to occur, e.g., $X = a$, is $\int_a^a p(x)\,dx = 0$, as mentioned before.

Both the discrete and the continuous examples were rather boring, as all the outcomes were equally likely (such random variables are called *uniform*). Let's make the next example a bit more exciting. Let Z denote the time you have to wait in line at some supermarket (a precise stop watch makes this a continuous random variable). Now, you might have observed that you usually have to wait between 1.5 and 2.5 min, but rarely less than 1 or more than 3, and never longer than 4. So, first, this is clearly not uniform. Second, unlike in the examples above, we don't know the real probabilities involved—but based on our experience, we can simply invent or postulate some probabilities and try to express them via a $p(x)$. We want $p(x)$ to be 0 for $x < 0$ (we can't wait a negative amount of time), and we set $p(x) = 0$ for any $x > 4$. We want the slice areas, i.e., probabilities, around 2 min to be larger than the areas at the edges of our 4 min range in order to match our anecdotal observations. A simple way to achieve this is to shape the function like a triangle with its peak at 2 min set to $p(2) = \frac{1}{2}$, for the whole, now triangular area must again equal $1 = \frac{1}{2} \times 4 \times p(2)$. We thus have $p(x) = \frac{1}{4}x$ for x in $[0, 2]$ and $p(x) = -\frac{1}{4}x + 1$ for x in $[2, 4]$ (see Fig. A.1).

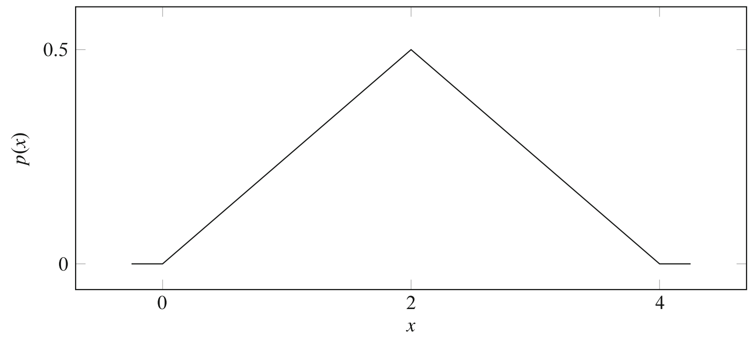

Fig. A.1 Triangular probability density

If we trust our hard-earned probabilistic model, we can now compute the probability of waiting between 17 and 25 s as the respective area under this function—the integral $\int_{17/60}^{25/60} \frac{1}{4} x \, dx$.

So outcomes and probabilities together describe and determine a random variable's behavior, its *distribution*.

A.2 Expected Value

As we have seen above, describing a random variable with all those outcomes and probabilities can be a wordy affair. We are looking for a way to get across some core characteristics of a random variable in a shorter, more succinct manner. On a hunch, we let us inspire by how we tend to average large sets of numbers (e.g., all the individual incomes of people living in Kansas) in order to compress the vast amount of information therein.

If a random variable X can have n discrete outcomes x_i, each with probability p_i, then we expect an "average," probability-weighted outcome—or *expected value*—of

$$\mathbb{E}[X] = \sum_{i=1}^{n} x_i p_i.$$

If each outcome is equally likely, we have $p_i = \frac{1}{n}$, and this expression becomes the familiar average.

In case of a continuous random variable, X can take on infinitely many values; their probabilities are described via a probability density function $p(x)$. By direct analogy with the discrete case we have[1]

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x p(x) \, dx.$$

(In both cases, or course, the probabilities themselves must always sum up to $1 = 100\%$, i.e., $\sum p_i = 1$ and $\int p(x) = 1$.)

Our examples fare as follows:

- The expected value of our pimped die is

$$\mathbb{E}[X] = \sum_{i=1}^{6} x_i p_i = \sum_{i=1}^{6} 10 i p_i = 35.$$

- The expected value of a standard die is $\sum_{i=1}^{6} i p_i = 3.5$.

[1]The discrete p_i corresponds to the infinitesimally small $p(x) \times dx$.

- The expected value of our spinning top is

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x\,p(x)\,dx = \int_{0}^{2\pi} x\frac{1}{2\pi}\,dx = \cdots = \pi.$$

- The expected value of the triangle probability density of waiting times we constructed at the beginning of this chapter is

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x\,p(x)\,dx = \int_{0}^{2} x\,(\frac{1}{4}x)\,dx + \int_{2}^{4} x\,(1 - \frac{1}{4}x)\,dx = \cdots = 2.$$

  (This corresponds to what we'd expect—the expected value here lies at our triangle's peak. As an exercise, try using a triangular probability density that is not isosceles and both guess and calculate its expected value.)

The expected value alone can't possibly give us the full picture of a random variable, yet it provides a first, brief glance at its behavior. If, for example, you hyper-pimped a die and only told me its new expected value of 3500, I might already get a fairly good impression of that die without knowing the details. (I might be wrong, because of course you could just have replaced a standard die's 6 with 20,985 to obtain that very same expected value.)

Sometimes the expected value already tells us all we need to know. Imagine a die game where you win a roll's outcome in dollars, e.g., a 4 nets you four bucks. Should you be willing to pay 3 dollars to take part in this game? We know that the die's expected value is 3.5, i.e., when playing repeatedly, you expect to receive 3.5 dollars and to thus earn 50 cents on average. Clearly, the 3 dollars investment would be worth it, but, alas, such games do not exist. If we reverse the setting, though, we obtain a game that does: would you be willing to *offer* the die game if someone paid 4 dollars to take part? Sure you would, and so do others; such games go by the name of lottery.

Moving on, the expected values intuitively extend to functions of randoms $f(X)$, i.e., to what we expect $f(x)$ to be on (probability-weighted) average, in both the discrete and the continuous case:

$$\mathbb{E}[f(X)] = \sum_{i=1}^{n} f(x_i)\,p_i,$$

$$\mathbb{E}[f(X)] = \int_{-\infty}^{\infty} f(x)\,p(x)\,dx.$$

For any constant c, we have

$$\mathbb{E}[cX] = c\,\mathbb{E}[X],$$

or, as special case (think: $X = x_1 = c$ with $p_1 = 100\%$),

$$\mathbb{E}[c] = c,$$

or, more generally,

$$\mathbb{E}[c_1 f_1(X) + c_2 f_2(X)] = c_1\,\mathbb{E}[f_1(X)] + c_2\,\mathbb{E}[f_2(X)].$$

A multiplicative separation is usually not possible. A discrete random variable X that is either 1 or 3 with a 50% chance has an expected value of $1 \times 50\% + 3 \times 50\% = 2$, while the expected value of X^2 is $1^2 \times 50\% + 3^2 \times 50\% = 5$. We see that, in general,

$$\mathbb{E}[X]^2 \neq \mathbb{E}[X^2].$$

A.3 Variance and Standard Deviation

The expected value expresses our average expectation of X. We'd also like to have a measure of a random's range of outcomes—its variability or volatility—around this expected value. For this, we examine $(X - \mathbb{E}[X])^2$—this expression becomes larger the more X tends to stray from its expected value. The average behavior of this expression is called *variance*, and it is defined as the following expected value:

$$\mathbb{V}\text{ar}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] = \int (x - \mathbb{E}[X])^2\, p(x)\, dx.$$

We can use the properties of the expected value mentioned above to find an alternative expression for the variance as exercise (the expected value of an expected value $\mathbb{E}[\mathbb{E}[X]]$ is the constant $\mathbb{E}[X]$ inside):

$$\begin{aligned}
\mathbb{V}\text{ar}[X] &= \mathbb{E}[(X - \mathbb{E}[X])^2] \\
&= \mathbb{E}[X^2 - 2X\,\mathbb{E}[X] + \mathbb{E}[X]^2] \\
&= \mathbb{E}[X^2] - 2\,\mathbb{E}[X]\,\mathbb{E}[X] + \mathbb{E}[X]^2 \\
&= \mathbb{E}[X^2] - \mathbb{E}[X]^2.
\end{aligned}$$

We can use either expression to derive the following general properties:

$$\begin{aligned}
\mathbb{V}\text{ar}[cX] &= c^2\,\mathbb{V}\text{ar}[X], \\
\mathbb{V}\text{ar}[c] &= 0.
\end{aligned}$$

The *standard deviation* is defined as the square root of the variance:

$$\text{std}[X] = \sqrt{\mathbb{V}\text{ar}[X]}.$$

It is often more useful than the variance because its scale or dimension is the same as that of X. If X values are in dollars, then the standard deviation lies on the same scale, while the variance has the unintuitive dimension of dollars-squared. (The variance's squaring approach merely helped make all the deviations from $\mathbb{E}[X]$ positively count toward our measure, and more gently so than the obnoxious absolute value.)

For our example of waiting times with its triangular probability density, we already know that

$$\mathbb{E}[X] = 2.$$

We compute

$$\mathbb{E}[X^2] = \int_{-\infty}^{\infty} x^2\, p(x)\, dx = \int_0^2 x^2\, (\frac{1}{4}x)\, dx + \int_2^4 x^2\, (1 - \frac{1}{4}x)\, dx = \cdots = \frac{14}{3}.$$

The variance becomes, via our shortcut,

$$\mathbb{V}\text{ar}[X] = \mathbb{E}[X^2] - \mathbb{E}[X]^2 = \frac{14}{3} - 2^2 = \frac{2}{3}.$$

Its square root yields the standard deviation of 0.82 min or 49 s.

A.4 Sample Estimates

Funnily, not many of the concepts mentioned above are of immediate use—we usually do not know the probabilities p_i or the shape of the probability density function $p(x)$, and we therefore cannot compute the expected value or the variance. What we can do is make some observations and *estimate* them.

Recall our example of the waiting-time random variable, where we postulated a triangular probability density function that allowed us to compute the random's expected value. Instead of making such a sweeping assumption, we might also observe and record some actual waiting times, for example, {1:45, 0:23, 2:35, 3:17, 1:33, 2:10, 1:52}.

We call such sample observations x_i.[2] Given n such observations, we use the *sample mean*

$$\bar{x} = \frac{1}{n} \sum x_i$$

[2]This is fine for continuous distributions where these names are nowhere to be seen. Just make sure not to confound them with the outcomes of a discrete distribution.

and the *sample standard deviation*

$$s = \sqrt{\frac{1}{n-1}\sum(x_i - \bar{x})^2}$$

to estimate the real but unknown expected value and standard deviation.

For our example of waiting times, the sample above yields an estimate for the expected value of 1:56 (as opposed to the calculated 2 min) and one for the standard deviation of 54 s (close enough to the 49 s obtained theoretically)—all without those pesky integrals.

The estimator for the standard deviation warrants a few words. Our original definition of variance would, in the discrete, equiprobable case, translate into $\frac{1}{n}\sum(x_i - E[X])^2$—a prime suspect for an estimator. Why then use the unintuitive $n-1$ in our sample standard deviation?

There are several ways to frame an answer. Dividing by $n-1$ can be shown to yield an *unbiased* estimate for the variance, i.e., it doesn't err systematically, which sure is a welcome feature.[3] Most of the time we can get away with just using this unbiased variant. It's also the default way Excel's `STDEV` function operates.

Also, a market risk setup typically involves different statistical software packages, programming languages, and the odd Excel analysis; outside parties like regulators or consumers of risk reports might try to reenact the figures on their own systems. This—most commonly used—unbiased estimator ensures the desired exact comparability of results.

And finally, as the sample size n gets larger, the correction by -1 becomes ever less significant. A professor of mine once quipped that whenever you worry about this denominator, you really should be worried about your sample size.

But what is the mathematical rationale behind all this? Omitting theory, we can give an intuitive mnemonic aid. The variance is all about squared deviations from the expected value. Unfortunately, we don't know this expected value and have to estimate it via the sample mean. Assume, for example, two samples of $\{-2, +1, +2, -1\}$ and $\{+2, +1, +2, +1\}$ (of the same distribution), and notice how the entries in the latter, by chance, all point in the same direction. The first sample has a mean of $\bar{x} = 0$, which would have the variance estimator add up terms the like of 1^2 and 2^2. The second, somewhat more compact sample has, smack in the middle of its value range, a mean of $\bar{x} = 1.5$, which would have the variance estimator add up the smaller 0.5^2 terms. In fact, the sample mean always minimizes the sum of squared differences to itself, and because the unknown underlying expected value is usually different, this sum inevitably tends to undershoot the real variance. Luckily, it can be shown that the humble tweak of averaging the sum of squares over the smaller $n-1$ instead of n can swimmingly correct for this tendency.

[3] A minor detail for your next Jeopardy session: the standard deviation estimate is still not unbiased, due to the square root operation.

In some cases, we do not have to estimate the expected value via the sample mean because we know it—for example, if we impose our own when creating random values artificially, or if we have some knowledge or strong intuition about the underlying random's behavior. In such cases, we don't have to compute an $\bar{x}$ from the sample but can directly apply the knowledge of $E[X]$ in estimating the variance. It turns out that the *uncorrected* estimator (note the n instead of $n-1$ and the $\mathbb{E}[X]$ instead of $\bar{x}$) is then the way to go:

$$s = \sqrt{\frac{1}{n}\sum(x_i - \mathbb{E}[X])^2}.$$

We use this variant, for example, when illustrating the Monte Carlo modification in our VaR setup.

Some standard software packages support this directly, e.g., Python's `statistics.pvariance(data,mu)` function, which accounts for a known expected value. Often, however, implementations only use n in the denominator but still implicitly estimate the mean.[4]

A.5 Kurtosis

We have primarily looked at the average behavior of distributions and at their range or volatility of outcomes. This can often already give us a pretty good idea about a distribution. Observing, for example, the height of males, we might obtain a mean of roughly 172 cm and a standard deviation of about 7 cm. We can relate to these numbers: we know quite a few average-height people, some that are shorter or taller, and a select few that are extremely short or tall. We also certainly know very few people that are, for example, ten times the standard deviation of 7 cm (or 70 cm) *taller* than the average. All in all, we are confident of having a good grasp on the height range and might well be inclined to call its distribution "normal."

Now consider the number of Barbie dolls in households. This might often be 0, or 1, or 7, and maybe have a mean of 3 and a standard deviation of, say, 2 dolls. It is easy to imagine, however, that one avid collector in Wichita will own maybe 250 dolls (many, but too few to meaningfully impact the standard deviation itself). This is more than 100 times that standard deviation of 2, or a full 200 dolls, *above* the mean of 3. We didn't observe such strange behavior with heights—there, the same multiplier of the standard deviation would describe a giant, 7 m taller than the

[4]Many software packages default to the unbiased estimate (Matlab; Octave; S-plus; R; SAS; Mathematica; SPSS; Python's `np.cov` for calculating a covariance matrix).

Several implementations, by default, divide by n without accounting for the potentially known mean (Boost's `variance` function; Python's `np.var` and `np.std` functions.)

Often, alternative estimator functions are provided and can be used to coordinate disparate implementations. (Excel's `STDEV.P` divides by n; Python's `np.var` and `np.std` use $n-1$ when setting the optional argument `ddof=1`.)

average. The doll distribution, now, seems to exhibit such extreme outliers that are many standard deviations away from the average.

Distributions with such behavior are said to feature *heavy* or *fat tails.* To measure them, we need to smoke out such very large deviations. We achieve this by examining $(X-\mathbb{E}[X])^4$, whose hefty fourth power should bring them to our attention. Additionally, we'd like a measure of "tailedness" to also be independent of the scales or dimensions involved; the "number of limbs of Barbie dolls," about four times the original random number, should have the same heavy tail indicator as the doll distribution itself.

The following measure does this, and it is called *kurtosis*:

$$\mathbb{K}\text{urt}[X] = \frac{\mathbb{E}[(X - \mathbb{E}[X])^4]}{\mathbb{V}\text{ar}[X]^2}.$$

Normalizing by the variance in the denominator ensures our desired invariance under scaling:

$$\mathbb{K}\text{urt}[cX] = \mathbb{K}\text{urt}[X].$$

For reasons we will tackle soon, a good reference value for the kurtosis is 3. It indicates a benign tail behavior and no undue or extreme outliers. Larger kurtosis values indicate heavier tails and outliers more extreme than conventionally expected, and it is not uncommon to observe kurtosis values of 10 or even 50 in the wild. As for our examples:

- A million and one households, one with 250 Barbie dolls, 500,000 with 1 doll, and 500,000 with 5 dolls, have a combined and unsuspicious mean of 3.0002, a standard deviation estimate of 2.0152, but a whopping kurtosis of about 224.
- Our example of waiting times above has a kurtosis of 4.16, or nothing much to worry about.

As with the variance, we usually don't compute this kurtosis but instead estimate it from a sample. Many burdensome tweaks are required to obtain an unbiased estimate, as a quick Google search for "kurtosis estimator" will reveal. For our purposes, it suffices to rely on your preferred software package's implementation.

Be mindful of one thing, though: some kurtosis functions, like Excel's `KURT` one, report the so-called *excess* kurtosis, which is the kurtosis minus 3.

A.6 Multiple Random Variables and Covariance

So far we have examined an individual random variable X and its properties. We now take a look at how multiple randoms play together. Consider, first, two discrete random variables X and Y, where X can take on $x_1 = 0$ or $x_2 = 1$ and where Y can take on $y_1 = 0$, $y_2 = 3$, or $y_3 = 9$. Combined, we can obtain 6 different outcomes:

$(0, 0)$, $(0, 3)$, $(0, 9)$, $(1, 0)$, $(1, 3)$, or $(1, 9)$. Correspondingly, we need 6 probabilities (adding, again, up to 1) to describe the joint distribution, best expressed in a two-dimensional matrix:

$$\begin{pmatrix} p_{X=0,Y=0} & p_{X=0,Y=3} & p_{X=0,Y=9} \\ p_{X=1,Y=0} & p_{X=1,Y=3} & p_{X=1,Y=9} \end{pmatrix}.$$

We can proceed to naturally define the expected value of, for example, the sum of the two random variables over all outcomes:

$$\mathbb{E}[X + Y] = \sum_{i=1}^{2} \sum_{j=1}^{3} (x_i + y_i)\, p_{X=x_i,Y=y_j}.$$

(If we assume identical probabilities of $\frac{1}{6}$ for each outcome, we obtain a result of 4.5 for this expression.)

The expected value of X alone would be, again involving 6 terms,

$$\mathbb{E}[X] = \sum_{i=1}^{2} \sum_{j=1}^{3} x_i\, p_{X=x_i,Y=y_i}.$$

The one-dimensional approach we encountered at the beginning of this chapter can also be used, if we appropriately collect the involved probabilities:

$$\begin{aligned} \mathbb{E}[X] = 0 \times (&p_{X=0,Y=0} + p_{X=0,Y=3} + p_{X=0,Y=9}) \\ &+1 \times (p_{X=1,Y=0} + p_{X=1,Y=3} + p_{X=1,Y=9}). \end{aligned}$$

The two probabilities involved, each a sum of three of the original ones, express the events $X = 0$ and $X = 1$, irrespective of Y. Such "collapsed" probabilities are called *marginal* probabilities.

By analogy, we use two-dimensional density functions $p(x, y)$ and double integrals in case of continuous distributions:

$$\mathbb{E}[f(X, Y)] = \int\!\!\int f(x, y)\, p(x, y)\, dx\, dy.$$

Luckily, all this comes down to a simple conclusion—shuffle the terms around and convince yourself that we usefully always have

$$\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y].$$

We can now begin to pose the first interesting question about multiple random variables: are they somehow related? For this, we examine whether they tend to move in the same direction. The expression $(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])$ is positive if both random variables are above their expected value, but it is also positive if both are below their expected value—this thus indicates co-movement in the same direction. The expression becomes negative, on the other hand, if they deviate in opposite directions from their expected values. Which of these types of co-movement dominates on average can be captured via the *covariance*:

$$\begin{aligned}\mathbb{C}\mathrm{ov}[X,Y] &= \mathbb{E}[(X-\mathbb{E}[X])(Y-\mathbb{E}[Y])]\\ &= \mathbb{E}[XY]-\mathbb{E}[X]\,\mathbb{E}[Y].\end{aligned}$$

If the covariance is positive, X and Y tend to move in the same direction on average; if it is negative, they tend to move in opposite directions. Either way, they two randoms are related.

We incidentally also note that

$$\mathbb{C}\mathrm{ov}[X,X] = \mathbb{V}\mathrm{ar}[X].$$

We shall mostly rely on estimators to guess the real but unknown covariance. In the special case where the individual expectations $\mathbb{E}[X]$ and $\mathbb{E}[Y]$ are known to be zero, the sample covariance estimate of n pairs of observations (x_1, y_1) to (x_n, y_n) simplifies to $\frac{1}{n}\sum x_i y_i$.

A convenient, normalized measure directly derived from this is the *correlation*, which yields values between $+1$ and -1, regardless of the volatilities underneath:

$$\mathrm{corr}[X,Y] = \frac{\mathbb{C}\mathrm{ov}[X,Y]}{\mathrm{std}[X]\,\mathrm{std}[Y]}.$$

(A quick intermediate sanity check: the variable X is surely strongly related to itself, as X always moves in the same direction as X (d'oh). If we actually evaluate the correlation of X to itself, we get $\mathrm{corr}[X,X] = 1$. Likewise, the correlation of X to its opposite $-X$ is $\mathrm{corr}[X,-X] = -1$.)

The covariance only approximates how random variables interact. For a deeper apprehension, we need the concept of *dependence*. It can be approached as follows: if knowing the outcome of one random variable does not give you any hint or additional information on how the other random variable will behave, then the two randoms are called *independent*.

It turns out that independent randoms always have a covariance or correlation of zero, and non-zero covariance or correlation thus signals dependence. Let's wrap our heads around this. If, for example, two randoms are positively correlated and we know that the first one went up, we'd expect the second random to tend to do the same. This is definitely tangible information, and it follows that the two randoms can't be independent.

When meeting a statistician at a bar, it is useful to keep in mind that zero correlation does not, in turn, guarantee independence. Witness the two randoms X (uniform in $[-1, 1]$) and $Y = X^2$. They are clearly dependent, for knowing the outcome of X will already foretell us the exact outcome of Y, but they have zero correlation. Yet such instances are rare in our context, and you'll find that zero correlation will many times correctly hint at independence.

We are now ready to tackle the issue of the variance of random sums. We already know that $\mathbb{V}\text{ar}[X - X] = 0$ and that $\mathbb{V}\text{ar}[X + X] = 4\,\mathbb{V}\text{ar}[X]$—a hint, maybe, that the relation between the random numbers might affect the variance of their sum. But let's plow through:

$$\begin{aligned}
\mathbb{V}\text{ar}[X+Y] &= \mathbb{E}[((X+Y) - \mathbb{E}[X+Y])^2] \\
&= \mathbb{E}[(X+Y)^2 - 2(X+Y)\,\mathbb{E}[X+Y] + \mathbb{E}[X+Y]^2] \\
&= \mathbb{E}[(X+Y)^2] - \mathbb{E}[X+Y]^2 \\
&= \mathbb{E}[X^2 + 2XY + Y^2] - (\mathbb{E}[X] + \mathbb{E}[Y])^2 \\
&= \mathbb{E}[X^2] + 2\,\mathbb{E}[XY] + \mathbb{E}[Y^2] - \mathbb{E}[X]^2 - 2\,\mathbb{E}[X]\,\mathbb{E}[Y] - \mathbb{E}[Y]^2.
\end{aligned}$$

If we look carefully at the last line's terms, we find that

$$\mathbb{V}\text{ar}[X+Y] = \mathbb{V}\text{ar}[X] + \mathbb{V}\text{ar}[Y] + 2\,\mathbb{C}\text{ov}[X,Y].$$

We get an even nicer expression for *independent* randoms, whose covariance, as mentioned above, is zero:

$$\mathbb{V}\text{ar}[X+Y] = \mathbb{V}\text{ar}[X] + \mathbb{V}\text{ar}[Y].$$

Going one step further, we can determine the variance of sums of more than two randoms. The number of terms becomes a bit unwieldy, but, fortunately, we can express the final result in a conveniently brief matrix notation. With a row vector of constants **c**, recalling that $\mathbb{V}\text{ar}[X] = \mathbb{C}\text{ov}[X,X]$, and defining a covariance matrix $[\mathbf{C}]$ with entries $\mathbb{C}\text{ov}[X_i, X_j]$, we conclude, with some patience, that

$$\mathbb{V}\text{ar}[c_1X_1 + c_2X_2 + c_3X_3] = \mathbf{c}[\mathbf{C}]\mathbf{c}^\top.$$

A.7 Distribution, Inverse, and Quantiles

The probability that $X < q$ is given by

$$\int_{-\infty}^{q} p(x)\,dx.$$

This recurring concept is best abbreviated via the *cumulative distribution function* $P(x)$:

$$P(x) = \int_{-\infty}^{x} p(t)\,dt.$$

The probability that $X < q$ is then $P(q)$.

The probability that X lies in a certain range $[q_d, q_u]$ is

$$\int_{-\infty}^{q_u} p(x)\,dx - \int_{-\infty}^{q_d} p(x)\,dx = P(q_u) - P(q_d).$$

The probability that $X > q$ is $1 - P(q)$.

We already know that the probability of $X = q$, i.e., of X ending up in the zero-length interval $[q, q]$, is $P(q) - P(q) = 0$. This odd property, once it has been shruggingly accepted, has the nice consequence that we need to worry less about open/closed intervals or the difference between“ $<$” and “$\leqslant$”—the infinitesimally small “border” outcomes make (for practical intents and purposes) no difference. This also makes many expressions for continuous distributions simpler, whereas, for discrete ones, we have to be much more careful about indices at boundaries.

Now let’s do the reverse: given a probability p, we can use the inverse $P^{-1}(\cdot)$ to find the corresponding value q such that $P(q) = p$:

$$q = P^{-1}(p).$$

Such q-values are called *quantiles*. The 1%-quantile $q_{1\%} = P^{-1}(1\%)$, for example, is our 1%-value-at-risk.

It makes sense to name or index the quantiles with their corresponding probabilities. The probability of a random number falling between $q_{3\%}$ and $q_{7\%}$, for instance, then becomes

$$P(q_{7\%}) - P(q_{3\%}) = 7\% - 3\% = 4\%.$$

What are the 5% shortest waiting times in our triangular waiting time distribution, i.e., what is its 5%-quantile? For quantiles q on the left side of the triangle, the integral in the cumulative is simply the triangular area $P(q) = \frac{1}{2}q\frac{q}{4} = \frac{1}{8}q^2$. For a probability $p = \frac{1}{8}q^2$, the inverse becomes $q = P^{-1}(p) = \sqrt{8p}$. Thus, for $p = 5\%$, the quantile $q_{5\%} = \sqrt{8 \times 5\%} = \sqrt{4/10}$. Verifying this, we see that indeed $P(q_{5\%}) = \frac{1}{8}q_{5\%}^2 = \frac{4}{80} = 5\%$. The 5% shortest waiting times lie between 0 and 38 s.

Quantiles, just like the expected value, scale and translate under linear transformations of the type $Y = aX + b$, with $a > 0$. You can formally prove this, or you can consider this to be simply a change of measurement units, like transforming Celsius to Fahrenheit, and sign off on it. The distribution’s core characteristics remain, and only the involved dimensions change. We have, e.g., for the 1%-quantile,

$$q_{1\%}^{Y} = a\,q_{1\%}^{X} + b.$$

A.8 Conditional Expectation

Sometimes only a subset of outcomes is of interest to us. The expected value, for instance, of a random variable under such a restricting condition is called *conditional expected value*. A typical example is the expected value of *X if X is smaller than a certain number c*. To obtain it, we simply sum/integrate up only to that number and normalize the result with the probability of our condition:

$$\mathbb{E}[X|X < c] = \frac{1}{P(c)} \int_{-\infty}^{c} x\, p(x)\, dx.$$

Our risk measure of the expected shortfall is such a conditional expectation. It deals with the 2.5% largest losses, so we have $c = q_{2.5\%}$ and $P(c) = P(q_{2.5\%}) = 2.5\%$. In our discrete case, we sum up 25 values of interest (the largest losses), each weighted with a probability $p_i = \frac{1}{1000}$. Dividing by the overall probability of 2.5% leaves us with the denominator 25 in Eq. (8.1).

A.9 The Normal Distribution

We finally get to meet an important and ubiquitous kind of distribution, one so common as to be called *normal* distribution. It arises in the context of sums of random variables, by which many phenomena can be characterized. A leaf falling through the air, for example, will undergo a series of tiny random nudges hither and thither before hitting the ground. A heap of leaves below a tree is then normally distributed. But onwards, from the bucolic to the more prosaic.

The normal distribution's probability density function $p(x)$ is driven by two parameters, μ and σ:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

This is denoted as an $N(\mu, \sigma^2)$-distribution. It is shaped like a bell, as can be seen in Fig. A.2, and also called a bell curve or a Gaussian.

This particular parameter setup is chosen to make the integral expressions for the base measures conveniently evaluate to

$$\begin{aligned}
\mathbb{E}[X] &= \int x\, p(x)\, dx = \cdots = \mu, \\
\mathbb{V}\mathrm{ar}[X] &= \int (x - \mathbb{E}[X])^2\, p(x)\, dx = \cdots = \sigma^2, \\
\mathrm{std}[X] &= \sigma.
\end{aligned}$$

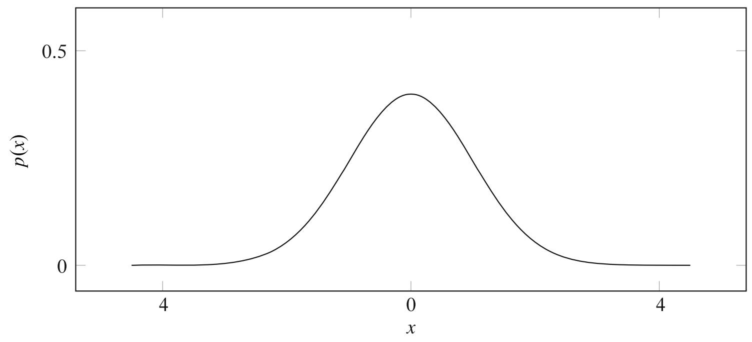

Fig. A.2 Normal probability density, for $\mu = 0$ and $\sigma = 1$

In theory, a normal random can take on any value—notice how the density is positive for all real numbers x, which allows for arbitrarily large positive or negative outcomes to occur. In practice, the probabilities become small so fast (due to the mighty e^{-x^2} term) that extreme events far away from μ are highly unlikely. The tails of a normal are thus not heavy but rather ordinary. The normal's kurtosis in fact came to signify unexciting and boring tail behavior. For any normal, regardless of its standard deviation, we have

$$\mathbb{K}\text{urt}[X] = 3.$$

That's where our ominous kurtosis value of 3 in Sect. A.5 originates.

A normal's cumulative and its inverse have no closed-form solution; we evaluate them by numerical approximation or by referring to tables with pre-computed values. For a normal distribution with $\mu = 0$ and $\sigma = 1$ (i.e., an $N(0, 1)$-distribution, also called *standard normal* distribution), the cumulative distribution is usually called Φ. Its 1%-quantile is approximately

$$q_{1\%} = \Phi^{-1}(1\%) = -2.32635.. \approx -2.33.$$

Quantiles scale with σ.[5] For an $N(0, \sigma^2)$-normal, we have

$$q_{1\%} = P^{-1}(1\%) = \sigma \times \Phi^{-1}(1\%) \approx -2.33\,\sigma.$$

To calculate the constant, use Excel's `NORM.INV(0.01;0;1)` or Wolfram Alpha's `InverseCDF[NormalDistribution[0,1], 0.01]`.

[5]That's because it can be proven that each $N(\mu, \sigma)$-normal can be expressed as a scaled $\sigma X + \mu$, with X standard normal. Since quantiles scale in general under such linear transformations, all normal quantiles can be retraced back to the standard ones, and we don't have to explicitly recalculate them for each different parameter combination.

The expected shortfall of a normal is the following conditional expectation:

$$\begin{aligned}\mathrm{ES}[X] &= \mathbb{E}[X|X < q_{2.5\%}]\\ &= \frac{1}{P(q_{2.5\%})}\int_{-\infty}^{q_{2.5\%}} x\,p(x)\,dx\\ &= \frac{1}{2.5\%}\int_{-\infty}^{q_{2.5\%}} x\,p(x)\,dx.\end{aligned}$$

For a standard normal with a density function φ, this evaluates to

$$\mathrm{ES}[X] = -\frac{1}{2.5\%}\varphi(\Phi^{-1}(2.5\%)),$$

and it is approximated numerically as 2.33780, e.g., using Excel's

```
-NORM.S.DIST(NORM.S.INV(0.025);FALSE)/0.025
```

It also scales with σ.

A.10 Sums of Randoms

We have already encountered random sums like $X + Y$. Still, we now get our hands dirty and try to become a bit more acquainted with them. Warning: your best friends in this section will be a pen and some sheets of paper.

Consider two independent random numbers with a uniform distribution, say, X over the interval $[0, 2]$ and Y over the interval $[0, 3]$. Their two-dimensional probability density function is

$$p(x, y) = \begin{cases} \frac{1}{6} & \text{for } x \text{ in } [0, 2] \text{ and for } y \text{ in } [0, 3],\\ 0 & \text{otherwise.}\end{cases}$$

Each random has, individually, its own (one-dimensional) probability density function. As its area must sum up to 1, we must have

$$p^X(x) = \begin{cases} \frac{1}{2} & \text{for } x \text{ in } [0, 2],\\ 0 & \text{otherwise;}\end{cases}$$

$$p^Y(y) = \begin{cases} \frac{1}{3} & \text{for } y \text{ in } [0, 3],\\ 0 & \text{otherwise.}\end{cases}$$

The relation between $p(x, y)$ and $p^X(x)$ or $p^Y(y)$ goes deeper. Just like we "collapsed" 6 discrete probabilities into 2 marginal ones in Sect. A.6, we may also determine $p^X(x)$ (for x in $[0, 2]$; it is zero otherwise) by integrating over y:

$$p^X(x) = \int_{-\infty}^{\infty} p(x, y)\, dy = \int_0^3 p(x, y)\, dy = \int_0^3 \frac{1}{6}\, dy = \frac{1}{2}.$$

Equivalently, integrating over the x-dimension yields the Y-marginal p^Y.

A very similar collapse from two dimensions to one will help us tackle the probabilities of random sums. If we define a new random variable $Z = X + Y$, with X and Y uniform as above, we might ask: What is Z's one-dimensional probability density? Is Z maybe also uniformly distributed?

To tackle this, we briefly digress to the discrete, two-dimensional setup of a uniformly random chess board with discrete axes X and Y between 1 and 8. The probabilities of the 64 outcomes $X = i$ and $Y = j$ are $p_{X=i,Y=j} = \frac{1}{64}$. What about a random $Z = X + Y$ and its (one-dimensional) probabilities $p_{Z=k}$?

First, $Z = 1$ can never happen, as it will always at least be 2.

There is only one way our Z can become 2: if both X and Y are 1 (all other setups create a larger Z). The corresponding probability is thus

$$p_{Z=2} = p_{X=1,Y=1}.$$

There are two ways to obtain $Z = 3$: via $X = 1$ and $Y = 2$, or via $X = 2$ and $Y = 1$. The corresponding probability is

$$p_{Z=3} = p_{X=1,Y=2} + p_{X=2,Y=1}.$$

There are three ways to obtain $Z = 4$: via $X = 1$ and $Y = 3$, via $X = 2$ and $Y = 2$, or via $X = 3$ and $Y = 1$:

$$p_{Z=4} = p_{X=1,Y=3} + p_{X=2,Y=2} + p_{X=3,Y=1}.$$

You get the idea—we basically sum up probabilities over diagonal segments of our board to obtain the $p_{Z=k}$. And it is also clear that these probabilities differ—that Z is not uniform. Its probabilities are as follows:

- The white main diagonal corresponds to $p_{Z=9}$:

$$p_{Z=9} = \sum_{t=1}^{8} p_{X=t,Y=9-t} = \frac{8}{64}.$$

- The lower left diagonals yield 7 probabilities for $Z = k$, with k between 2 and 8:

$$p_{Z=k} = \sum_{t=1}^{k-1} p_{X=t,Y=k-t} = \frac{k-1}{64}.$$

- The upper right diagonals also yield 7 probabilities for $Z = k$, with k between 10 and 16:

$$p_{Z=k} = \sum_{t=k-8}^{8} p_{X=t,Y=k-t} = \frac{17-k}{64}.$$

With a good hunch, we return to our continuous two-dimensional distribution. Because the original $[0,2] \times [0,3]$-uniform is a bit tedious with regard to integration bounds, we consider the simpler uniform on $[0,1] \times [0,1]$ with $p(x,y) = 1$ over that area. We confidently declare that the following density describes $Z = X + Y$:

$$\begin{aligned} p^Z(z) &= \int_{-\infty}^{\infty} p(t, z-t)\, dt \\ &= \begin{cases} \int_0^z p(t,z-t)\,dt = t|_0^z = z & \text{for } z \text{ in } [0,1], \\ \int_{z-1}^1 p(t,z-t)\,dt = t|_{z-1}^1 = 2-z & \text{for } z \text{ in } [1,2], \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

The probability density function of Z is thus a triangle. (As an exercise, you might want to try this for uniforms of unequal ranges.)

There are several reasons we went through this exercise. It should, first, underline the close correspondence of discrete and continuous setups. It also hopefully illustrates that boundary cases can often be managed more elegantly, and with less of an index mess, in a continuous setup. Mainly, however, is should stress that probability densities of randoms do not always translate trivially into the density of their sum. This should prepare the stage for what hopefully provides some relief now.

For it turns out that multiple *normal* random variables following the so-called *multi-variate* normal distribution behave much more benignly under summation. Each random variable is normally distributed, and, crucially, it can be shown that their sum is also normally distributed, which spares us laborious integrals. We can derive the characteristics (i.e., μ and σ) of the sum of normals directly from the individual distributions' μ, σ, and their correlation ρ:

$$\begin{aligned} \mu_{X+Y} &= \mu_X + \mu_Y, \\ \sigma_{X+Y} &= \sqrt{\sigma_X^2 + \sigma_Y^2 + 2\rho\sigma_X\sigma_Y}. \end{aligned}$$

The main takeaway, neglecting normals that follow degenerate "non-multi-variate" distributions, is that "the sum of normals is normal"—if for no other reason than because we often simply assume such a distribution off the bat.[6]

[6]Two more things to keep in mind: two independent normals also share a joint density, and the expressions above simplify further because their $\rho = 0$; and for multi-variate normals, independence and zero correlation are fully equivalent concepts.

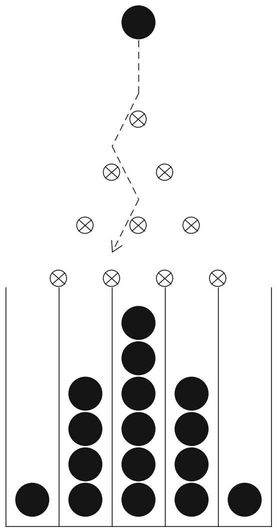
Fig. A.3 A Galton board

There are several ways to prove this, but they often only provide verification in a formal, technical sense. Surely such a fundamental property must be rooted understandably in the very setup of the normal distribution itself. So to instill some confidence, we look at the case of independent normals and their sum via the so-called Galton board depicted in Fig. A.3.

In this game, a ball is dropped over layers of offset nails. As it traverses downwards, it randomly goes left or right at each nail, before finally ending up in a bin below. The height of the ball stacks in the bins—the outcome of random sequences or sums of left/right movements—can be shown to resemble a normal density as we use more and more layers of nails. Assume that one such board corresponds to a normal X. Now, let's drill a hole in a bin below, attach a second board right below that hole, let the balls fall on, and collect them again further down; we repeat this for each bin. This is akin to adding a second (also normal) board Y to the first one. The whole procedure should result in the same final bin tally as when using one larger board with as many layers as X and Y combined. Because such a larger board is, like any board, also akin to a normal, the sum of the original boards better be as well.

We can use a similar trick with our sum of uniforms, whose probability density we already found to be triangular. This time the game is Tetris. We let a first random X in $[0, 4]$ determine the starting position of the coveted 1×4 brick. Each such brick we then interpret as the probability density of width 4 of a second uniform Y, right before we let it drop down. Once such a brick, starting off at position X, comes to rest, we consider it to stand for part of the density of $X + Y$. As each starting position X is equally likely, we might as well loop through them, obtaining the left-hand side in Fig. A.4.

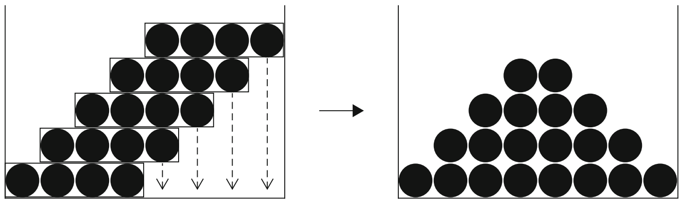

Fig. A.4 Tetris

If we then let the bricks unglue and the resulting 1×1 pieces drop to their final resting place, we obtain the $X + Y$ density—our familiar triangle.

A.11 Some Densities for the Road (to Independence)

We have seen that we were able to derive quite a few properties of randoms without much actual computation. Some distributions, most notably the normal one, provide additional shortcuts because of their very specific structure. Still, it is often instructive to perform a handful of raw calculations explicitly to whet our intuition, especially in the two-dimensional case. Here are some starting points.

We have already encountered the triangular distribution in our waiting time example. What would a two-dimensional probability density of two independent waiting times X and Y over $[0, 4] \times [0, 4]$ look like? To get an idea, go to your bedroom, grab your bed sheet right at the center of the mattress, and pull it up. The resulting structure resembles a wigwam. Let's try to construct a corresponding probability density $p(x, y)$.

The wigwam has its peak at $(x, y) = (2, 2)$. Let's look at the lower left part or quarter of this volume first (i.e., x in $[0, 2]$ and y in $[0, 2]$). If we define $p(x, y) = cxy$, we see that its height is 0 for $x = 0$ or $y = 0$, and it is $4c$ for $(x, y) = (2, 2)$. The volume of the lower left part is[7]

$$\int_0^2 \int_0^2 p(x, y)\, dx\, dy = \int_0^2 \int_0^2 cxy\, dx\, dy = 4c.$$

To also describe the remaining 3 quarters of the wigwam, we may use, for x and y both in $[0, 4]$,[8]

$$p(x, y) = c(2 - |2 - x|)(2 - |2 - y|).$$

[7]You can use Wolfram Alpha's website and type in
```
integrate c x y dx dy, x = 0 to 2, y = 0 to 2
```
to make sure.

[8]You can directly google `(2-abs(2-x))(2-abs(2-y))`, which should give you a nice 3D-plot where you just need to adjust the graph's display ranges.

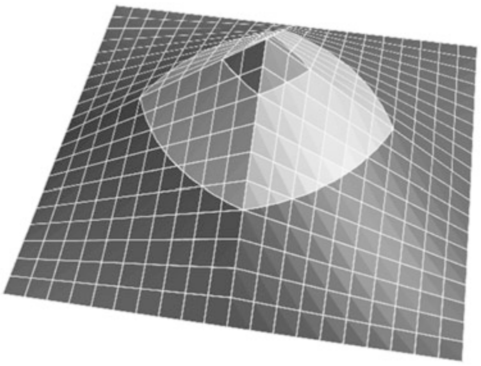

Fig. A.5 Wigwam (z-axis range tuned for clarity)

The total volume of the wigwam must be, because of its symmetry, $4 \times 4c = 16c$. For $c = 1/16$, we obtain a valid probability density (see Fig. A.5).

Let's check the marginal distribution of X, for x in $[0, 2]$ (larger x work similarly):

$$p^X(x) = \int_0^2 \frac{1}{16} xy\,dy + \int_2^4 \frac{1}{16} x(2 - (y - 2))\,dy = \cdots = \frac{1}{4} x.$$

This is the left side of our trustworthy triangle distribution. For the full range of x in $[0, 4]$, we get

$$p^X(x) = \frac{1}{4}(2 - |2 - x|).$$

We notice that in our wigwam case we have $p(x, y) = p^X(x)p^Y(y)$—so we could have avoided all the construction work and simply have multiplied the individual densities in the first place. Such a neat multiplicative separation of a two-dimensional probability density is not always possible. If it is, though, then this is very telling, as we will soon discover.

We are almost done with our wigwam but still want to check whether the X and Y described by it are independent. Their covariance is zero (as you can verify by doing the appropriate integration), but that's only a hint. Recall that independence essentially means that knowing X does not tell us anything about Y.

How can Y behave if we know that X equals some specific, say, x'? We intuit that it should loosely behave according to the one-dimensional function of y given by $p(x', y)$, i.e., a vertical slice through our wigwam. This slice is always a triangle here, as depicted in Fig. A.6.

This almost looks like a density already, except that its area does not have to be 1. We can easily remedy that by scaling the function and dividing it by its own area $\int p(x', y)\,dy$, which of course is simply the value given by $p^X(x')$. Doing this yields a valid *conditional* probability density with area 1:

$$p(y|X = x') = \frac{p(x', y)}{p^X(x')}.$$

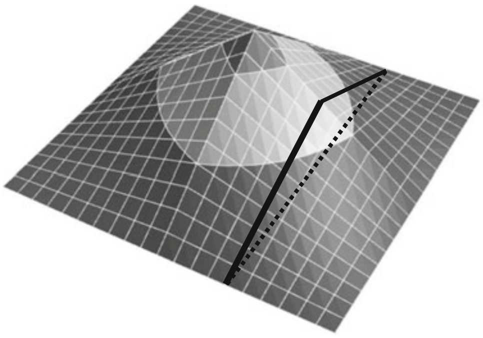
Fig. A.6 Wigwam with (yet unscaled) conditional density

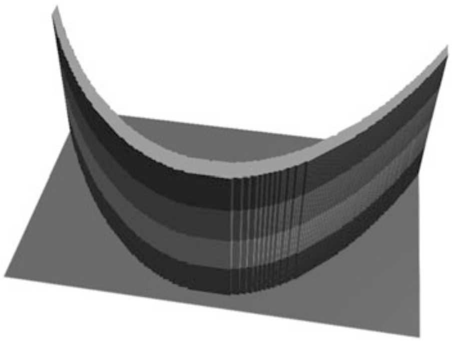
Fig. A.7 Plank

This density describes how Y behaves once X has settled on x'. The question of independence becomes: does this x' even influence Y? At first sight, yes (there are, after all, plenty of x' on the right-hand side of the equation). At second sight, we recall that for the wigwam it holds that $p(x, y) = p^X(x)p^Y(y)$, and therefore

$$p(y|X = x') = \frac{p(x', y)}{p^X(x')} = \frac{p^X(x')p^Y(y)}{p^X(x')} = p^Y(y).$$

So Y's conditional density is not affected by X at all—a realization of X does not tell us anything about how Y might behave. The wigwam must be independent. (It is also but a little mental stretch that makes us realize: independence and the multiplicative separation of a two-dimensional density mean one and the same.)

Now take two randoms, with X uniform in $[-1, 1]$ and Y uniform in $[x^2, x^2+0.01]$. The graph looks like a bended plank standing upright on its narrow side, 2 wide and 0.01 thick (see Fig. A.7). How long is it, i.e., how high is the graph? Well, the volume must be $1 = 0.01 \times 2 \times h$, so we have a height of 50.

The conditional density of Y given $X = x'$ is

$$p(y|X = x') = \begin{cases} \frac{50}{0.01\times 50} = 100 & \text{for } y \text{ in } [x'^2, x'^2 + 0.01], \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, we can't get rid of the x' here. Knowing x' is in fact vital and, in turn, also inevitably tells us a lot about how Y will behave. If X is close to 1 or -1, then

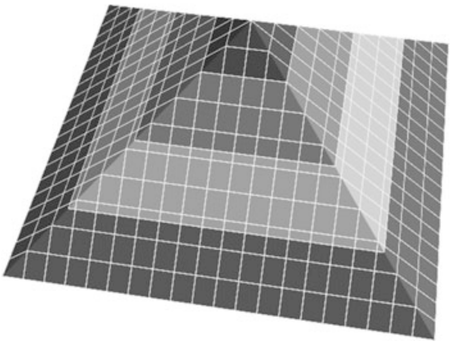
Fig. A.8 Pyramid

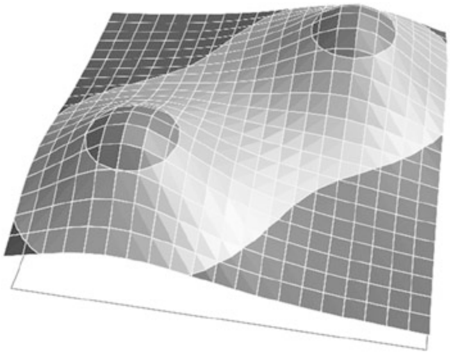
Fig. A.9 Camel

Y will be close to 1; if X is close to zero, so will be Y. These X and Y are therefore not independent.[9]

With this intuition, we can make short shrift of another density, the pyramid density of Fig. A.8 over the base $[0, 4] \times [0, 4]$, peaking at $(2, 2)$. Are the thusly described X and Y independent? Well, the conditional densities around values of x' close to 2 look like triangle distributions, whereas the ones around $x' = 1$ or $x' = 3$ look trapezoid. No amount of mere scaling can ever bring them in line—Y behaves differently for different x'. The pyramid is not independent.

To wrap up, lest we get the impression that there are only freak distributions out there, it helps to construct a plain one with a proper covariance. We might call this one a camel hump distribution. To obtain peaks at, say, $(1, 1)$ and $(3, 3)$, we start with the following guess[10]:

$$p^?(x, y) = \frac{1}{1 + (1 - x)^2 + (1 - y)^2} + \frac{1}{1 + (3 - x)^2 + (3 - y)^2}.$$

[9]Their covariance, though, is zero, as a calculation exercise reveals. Also note that the already encountered randoms X and $Y = X^2$ are the limiting case of ever-thinner planks.

[10]Google `1/(1+(1-x)^2+(1-y)^2) + 1/(1+(3-x)^2+(3-y)^2)` to confirm this function's "camelity," or refer to Fig. A.9.

Its volume on $[0, 4] \times [0, 4]$ is 9.4774,[11] which normalizes our guess into a proper density $p(x, y) = \frac{1}{9.4774} p^{?}(x, y)$. The resulting covariance is, finally, a full-fledged number:

$$\mathbb{C}\mathrm{ov}[X, Y] = \int\!\!\int (x - 2)(y - 2) p(x, y)\, dx\, dy = \cdots = 0.342146.$$

[11] Via Wolfram Alpha's
`integrate (1/(1+((1-x)^2+(1-y)^2))+1/(1+((3-x)^2+(3-y)^2))) dx dy, x=0 to 4, y=0 to 4.`

Pricing

B

Determining the value of your positions is easy for frequently traded, *liquid* assets because market quotes of prices are readily available. But what about completely new, never-before traded assets without market quotes, or assets that are very rarely traded, so-called *illiquid* assets? Determining their value cannot purely rely on direct lookups into a current market snapshot, as there are no or only sporadic records of actual trades.

The idea in pricing such positions is not spectacular and relies on one fundamental assumption: *asset prices should be consistent*. For example, bonds with similar maturities and from issuers of similar credit-worthiness should probably cost about the same. Likewise, currency exchange rates should be attuned to each other. If given two exchange rates $\$_{€}$ and $€_{£}$, then the direct exchange rate $\$_{£}$ should be in line with them. Otherwise there would be a cheap way to convert **$** into **£** (for example, by converting **$** into **€** and then **€** into **£**) and a more expensive one (the other route of directly converting **$** into **£**), and traders taking the cheap route would bid up the prices involved until both ways of conversion aligned again.

Following this train of thought, another way of expressing this idea of price consistency comes to mind: there should be no sure, risk-free profits. If you knew that a stock is currently worth 12**$** and someone offered it to you for 10**$**, you could buy it and immediately sell it on the market—pocketing a sweet, risk-free 2**$** in the process. Price consistency means that such discrepancies are assumed not to exist. Gains like these would be the financial world's equivalent of the physical impossibility of doing work without spending energy.

This reasoning can even be applied if it involves the fog of the future: you should not be able to schedule prospective asset exchanges in a way that guarantees you a risk-free profit. Assume, for example, that a stock **S** is currently worth 12**$**. Someone offers you a deal: if you pay him 10**$** now, he will deliver this stock in one month's time. It seems an OK deal, and you could accept it, pay, wait a month, and hope the stock will later be worth more than what you paid now—yet you still can both gain or lose, i.e., you take on some risk. However, you could transform this deal into one with guaranteed risk-free profit the following way: borrow the stock, sell it for its current price of 12$, pay 10$ for the original deal, wait a month, get the stock as promised, and return it to the stock lender. You are left with a sure profit of 2$, regardless of how the stock develops. Assuming no risk-free profits exist thus rules out such deal opportunities as well.

Extending this one step further, we even want to ban risk-free profits *on average*, for example, for repeated, risky deals. Say, someone offered to (repeatedly) cast a die and to give you the resulting number of stock units—how much would you be willing to pay to take part in this game? If he offered you this deal in exchange for three units of stock, you would certainly take it and just keep playing, for on average you'd get 3.5 units of stock by investing only 3. As above, it seems reasonable to rule out such deals as well.

In short, exchanging assets back and forth should not let you end up with more asset units than you started out with. Such a profit bonanza is called *arbitrage* opportunity (there is a precise mathematical definition of arbitrage, but we'll keep it shamelessly visceral). Finding consistent prices relies on the assumption that no such arbitrage exists, and it is called *no-arbitrage pricing*. Another angle on this is that prices of illiquid assets are *interpolated* from known prices of liquid ones, and that the discipline of pricing, known as *quantitative finance*, is actually a big interpolation framework.

But wait—didn't we already encounter an arbitrage opportunity? We could buy a zero bond for 0.9$ and thereby make sure to earn 0.1$ once the bond pays out its promised 1$. This seems to be a sure profit—but only on the face of it. Money, unfortunately, usually loses value over time, and comparing nominal units of money at different points in time is therefore misleading. A can of Coke, for example, cost 5 cents in the 1950s; it costs more now but is probably "worth" the same as back in the day.

So money is a special kind of asset, also called (negative) arbitrage asset for its holder. Like cars, money typically loses value over time with respect to other assets. It is a (positive) arbitrage asset for the issuer of the money, the government. This makes—ironically—monetary prices, i.e., prices with respect to money, somewhat ill-suited for consistent pricing. We shall now get to know a more elegant, money-eschewing approach to arbitrage-free pricing. We can only hint at its most basic ideas here. But after some simple examples, we should at least be able to price call options, which was worthy of a cool Nobel prize not too long ago.

Note: this brief motivational chapter heavily borrows both ideas and notations from Jan Vecer's "Stochastic Finance: A Numeraire Approach" (Vecer 2011).

B.1 Trades as Asset Exchanges

Financial assets are often described as being intangible, but I find that it sometimes helps to view them as existing, palpable, and immutable things. Like a dollar bill, we can consider a bond or a stock to be a piece of paper or a contract (and not so long ago, before online banking, those papers actually existed). In this chapter, we will denote assets in bold face—a dollar **\$**, a stock **S**, a zero bond **B**, etc., to demarcate assets from their prices. Assets do not change over time—a piece of paper remains a piece of paper. Their prices, however, do change.

Positions and portfolios are then quite naturally mere multiples and sums of such assets, e.g., a portfolio might consist of some stock, bonds, and debt:

$$80\mathbf{S} + 400\mathbf{B} - 1200\mathbf{\$}.$$

If we borrow one unit of a stock and sell it to bet on falling prices, i.e., if we short it, the resulting portfolio is

$$-\mathbf{S} + 12\mathbf{\$}.$$

This way of describing positions and portfolios is handy for keeping track of rights and obligations. It can also express asset trades and deal with time. We can, for example, describe buying a stock now (at time $t = 0$) with the following asset exchange relation:

$$\mathbf{S} \sim^0 12\mathbf{\$}.$$

This asset relation denotes that 12**\$** can (now) be traded for one unit of **S**, or vice-versa. The stock's price in **\$** is 12. We can also express promises this way, for example, that a zero bond **B** will pay 1**\$** at time T corresponds to the following future exchange:

$$\mathbf{B} \sim^T \mathbf{\$}.$$

A contract **F** that obliges you to buy a stock at a set dollar price k in the future is given by

$$\mathbf{F} \sim^T \mathbf{S} - k\mathbf{\$}.$$

Think of it as exchanging, at time T, a piece of paper called **F** for a piece of paper called **S** while parting with k precious pieces of paper called dollars.

The math of asset relations behaves intuitively. Adding or subtracting assets and grouping together assets of the same kind make sense; operations like multiplying an asset with another do not. Note that asset relations are valid only at a specific point in time, so even if it holds that $\mathbf{B} \sim^T \mathbf{\$}$, it does not follow that a bond can be exchanged for a dollar right now ($\mathbf{B} \not\sim^0 \mathbf{\$}$). A zero bond is typically cheaper than the future dollar it promises; we might, e.g., experience current exchange levels of $\mathbf{B} \sim^0 0.9\mathbf{\$}$. Expressed in terms of some continuously compounded interest rate r, we often equivalently express this as $\mathbf{B} \sim^0 e^{-rT}\mathbf{\$}$.

B.2 Prices as Ratios

Asset prices are exchange ratios that describe how many units of an asset can be exchanged for a unit of a different asset. Buying a stock **S** worth 12**$** can be expressed, as we have seen, in the following asset exchange relation:

$$\mathbf{S} \sim^0 12\mathbf{\$}.$$

Another way to describe this is to say "the price of **S** now, at time 0, in terms of **$** is 12" or, less chatty,

$$S_\$(0) = 12.$$

This is no longer an asset relation but a conventional mathematical equation of prices or ratios. The font face alerts us: $S_\$(0)$ is a number, while **S** is a thing.

The so-called reference asset used for pricing in the example above is the dollar **$**. But we can also express price ratios with respect to another asset, maybe a zero bond **B**. The price of **S** with respect to **B** is the number of units of **B** needed to buy one unit of **S**. How to get this new price? We know that we can exchange $\mathbf{S} \sim^0 12\mathbf{\$}$. If the current bond price is $B_\$(0) = 0.9$, we can exchange $\mathbf{B} \sim^0 0.9\mathbf{\$}$ or $\mathbf{\$} \sim^0 \frac{1}{0.9}\mathbf{B}$. So we can exchange

$$\mathbf{S} \sim^0 12\mathbf{\$} \sim^0 12 \times \frac{1}{0.9}\mathbf{B} \sim^0 13.33\mathbf{B}.$$

We thus obtain the current price of **S** with respect to **B**:

$$S_B(0) = 13.33.$$

Why would we ever want to use reference assets other than the **$**? The answer is that some pricing exercises become simpler. By sidestepping money as reference assets, we can often avoid having to compensate for its depreciation via discounting or its opposite, compounding. In more complex setups, we might be able to reduce the dimensionality of integrals. In short, it is simply more elegant.

So we mainly operate on prices with respect to no-arbitrage assets and in the end convert those prices to dollar ones via chained relations like

$$S_\$(0) = S_B(0)B_\$(0).$$

B.3 Prices of Future Delivery

The dollar prices of liquid assets are given by the current exchange ratios readily visible in the markets. But what happens if we want to exchange assets in the future? After all, asset prices fluctuate and the future is unknown.

This section deals with two such examples and involves, even though future prices are random, no probabilities. Both rely on the idea of no-arbitrage. The first answers how much you should be willing to pay *now* in order to get one unit of **S** in the future. The second is about how much you should agree to pay *in the future* for that same **S**.

Consider a contract **K** that promises to deliver, at time T, one unit of **S**:

$$\mathbf{K} \sim^T \mathbf{S}.$$

What is this contract worth now, i.e., what is its price $K_\$(0)$? Although we don't know what **S** will be worth at time T, we can determine this price. Consider the following two cases:

- If **K**'s current price were *higher* than the current stock price $S_\$(0)$, you could sell **K**, buy the stock **S** right now with only parts of the proceeds, hold it, and finally deliver it as promised. You could pocket the leftover money as immediate, risk-free gains.
- On the other hand, if the current price of **K** were *lower* than that of **S**, you could borrow the stock **S**, sell it at its current price, buy the contract **K** with only parts of the proceeds, and then wait unperturbed until **K** delivers you the **S** to be returned to its lender. Again, a profit at no risk.

The only price a buyer and seller can ever agree upon as fair is thus the current stock price:

$$K_\$(0) = S_\$(0) \quad \text{or} \quad K_S(0) = 1.$$

By the same reasoning, another contract $\mathbf{K}'$ delivering a bond **B** is priced as $K'_\$(0) = B_\$(0) = e^{-rT}$. Yet another contract $\mathbf{K}''$ delivering n units of **S** is of course priced as $K''_\$(0) = nS_\$(0)$.

These same simple relations do not hold for arbitrage assets like money. What would you be willing to pay now to get 1**\$** at time T? Certainly not 1**\$**. This is because money, as we mentioned, loses value with respect to other assets. Yet the workaround is simple, and we'll apply it in the following, second example.

With a *forward contract* **F**, you commit to buying **S** at time T for k**\$**:

$$\mathbf{F} \sim^T \mathbf{S} - k\mathbf{\$}.$$

What would be a fair future exchange ratio or price k that obviates any upfront money exchange, i.e., that makes $F_\$(0) = 0$?

First, **F** is clearly simply the sum $\mathbf{K}+\mathbf{J}$, with $\mathbf{K} \sim^T \mathbf{S}$ and $\mathbf{J} \sim^T -k\mathbf{\$}$. The current price of **K** is the same as the stock's (see our example immediately above):

$$K_\$(0) = S_\$(0).$$

But $\mathbf{J}$ is a monetary promise that ill-transcends time. Luckily, we know that a zero bond $\mathbf{B}$ can be exchanged for a $\mathbf{\$}$ at maturity T:

$$\mathbf{B} \sim^T \mathbf{\$}.$$

This lets us express $\mathbf{J}$'s promise in terms of a zero bond:

$$\mathbf{J} \sim^T -k\mathbf{\$} \sim^T -k\mathbf{B}.$$

We know that $\mathbf{J}$'s current price must thus be $-k$ times the current price of the bond $B_\$(0)$. We get:

$$J_\$(0) = -kB_\$(0) \quad \text{or} \quad J_\$(0) = -ke^{-rT}.$$

For the current price of the forward $\mathbf{F}$ to be zero, we must have

$$0 = F_\$(0) = K_\$(0) + J_\$(0) = S_\$(0) - kB_\$(0) = S_\$(0) - ke^{-rT}.$$

This finally yields the so-called *forward price* k:

$$k = S_\$(0)e^{rT}.$$

(This forward price k is not to be confused with the price of the forward $F_\$(\cdot)$ itself. The latter is, by agreement upon k, zero at the beginning. As the stock price then starts to fluctuate, $F_\$(\cdot)$ will stray from zero and fluctuate as well.)

So we have settled both our initial questions by purely relying on price consistency or no-arbitrage. We now go one step further and explore the random nature of prices over time.

B.4 Prices as Expectations

Consider the stock price in terms of bonds, $S_B(\cdot)$. We know its current value $S_B(0)$, but future prices $S_B(T)$ are random and can only be described in terms of probabilities. The future price of a stock in terms of bonds can be higher or lower than the current price. However, as we hinted at before, it makes sense to assume the following: at least *on average*, $S_B(T)$ should not be higher or lower than $S_B(0)$. For if $S_B(T)$ were usually higher than the current $S_B(0)$, we would surely exchange all our bonds for stock, wait, and convert the stock back into bonds, because we would expect to often end up with more units of the bond than we set out with. In fact, everybody would try to enter such trades and thus drive up the current stock price. So we simply rule out such gains in our pricing model.

We treat $S_B(T)$ as a continuous random variable and assume it behaves according to the commonly used *log-normal* probability density, with x shorthand for $S_B(T)$:

$$p(x) = \frac{1}{x\sigma\sqrt{2T\pi}} e^{-\frac{(\log(x)-\log(S_B(0))+\frac{1}{2}\sigma^2 T)^2}{2\sigma^2 T}}.$$

As discussed before, we want the average of our random variable to be identical to the current stock price. Expressed via the expected value, we want $\mathbb{E}[S_B(T)]$ to be identical to $S_B(0)$. That this is indeed the case can be verified by computing the integral $\mathbb{E}[S_B(T)] = \int x p(x)\, dx$, which actually yields $S_B(0)$. So this seems to be a reasonable probability density.

We can generally view current prices as the expected value of asset units delivered. Take a contract that promises a random number X of bonds:

$$\mathbf{K} \sim^T X\mathbf{B}.$$

The price of $\mathbf{K}$ with respect to the delivered asset $\mathbf{B}$ must then reasonably be $\mathbb{E}[X]$ if we exclude arbitrage, otherwise we could gain or lose assets on average. A contract whose price we already derived may underline this point:

$$\mathbf{K} \sim^T \mathbf{S}.$$

At time T, we could immediately exchange the stock for bonds and consider this equivalent contract:

$$\mathbf{K} \sim^T S_B(T)\mathbf{B}.$$

This is a promise of a random number of bond units, and we therefore expect that

$$K_B(0) = \mathbb{E}[S_B(T)].$$

As noted above, this expected value evaluates to $S_B(0)$. The resulting dollar price of $\mathbf{K}$ thus coincides with our previous price derivation because

$$K_\$(0) = K_B(0)B_\$(0) = \mathbb{E}[S_B(T)]B_\$(0) = S_B(0)B_\$(0) = S_\$(0).$$

This is all fairly gimmicky when only considering trivial assets. Yet when pricing so-called *derivative* assets, whose promises are conditional on the prices of basic assets, we can gainfully apply the same approach. The most prominent such derivative is coming up.

B.5 The Call Option

We are now prepared to take on the call option **C**. It grants you the right to buy, at some future time T, a stock at a pre-determined strike price k:

$$\mathbf{C} \sim^T (\mathbf{S} - k\mathbf{\$})^+.$$

The $+$ denotes that you will exercise your claim and enter the buying transaction on the right only if the stock's dollar price is larger than the strike at time T, i.e., if the resulting portfolio value is positive.[12] The option expires worthlessly otherwise.

Just like with the forward, we first replace the **\$** with a zero bond of maturity T:

$$\mathbf{C} \sim^T (\mathbf{S} - k\mathbf{B})^+.$$

We also replace the stock with a corresponding bond position:

$$\mathbf{C} \sim^T (S_B(T)\mathbf{B} - k\mathbf{B})^+.$$

As we are now only dealing with the bond asset on the right-hand side, we can factor it out:

$$\mathbf{C} \sim^T (S_B(T) - k)^+\mathbf{B}.$$

The coefficient of **B** is the random amount of units of **B** delivered by **C**, or $C_B(T)$. We are looking for the current price of the call, $C_B(0)$, which must equal

$$C_B(0) = \mathbb{E}[C_B(T)] = \mathbb{E}[(S_B(T) - k)^+].$$

We next have to actually calculate the corresponding integral, with $x = S_B(T)$:

$$\begin{aligned}
C_B(0) &= E[(x-k)^+] \\
&= \int_{-\infty}^{\infty} (x-k)^+ p(x)\,dx \\
&= \int_{k}^{\infty} (x-k) p(x)\,dx \\
&= \int_{k}^{\infty} (x-k) \frac{1}{x\sigma\sqrt{2T\pi}} e^{-\frac{(\log(x) - \log(S_B(0)) + \frac{1}{2}\sigma^2 T)^2}{2\sigma^2 T}}\,dx.
\end{aligned}$$

[12]The $(\cdot)^+$ is a valid mathematical operator on asset expressions because the *sign* of a portfolio's price does not depend on the reference asset used for pricing.

Depending on your mood, you can integrate this expression by hand or use integration software like Mathematica. The integral evaluates to:

$$\begin{aligned} C_B(0) = S_B(0)\ \Phi & \left[\frac{1}{\sigma\sqrt{T}}(\log S_B(0) - \log K + \frac{1}{2}\sigma^2 T)\right] \\ -K\ \Phi & \left[\frac{1}{\sigma\sqrt{T}}(\log S_B(0) - \log K - \frac{1}{2}\sigma^2 T)\right]. \end{aligned}$$

This almost looks like the formula you find in the books. To exactly match that classic formulation, which is given in dollar and not bond terms, two additional steps are required. First, we replace the stock price in bond terms with the equivalent dollar expression. We have

$$S_\$(0) = S_B(0)B_\$(0) = S_B(0)e^{-rT} \implies S_B(0) = S_\$(0)e^{rT},$$

and of course

$$\log S_B(0) = \log(S_\$(0)e^{rT}) = \log S_\$(0) + rT.$$

If we also translate the call price from bond to dollar terms via

$$C_\$(0) = C_B(0)B_\$(0) = C_B(0)e^{-rT},$$

we obtain the classic Black-Scholes formula:

$$\begin{aligned} C_\$(0) = S_\$(0)\ \Phi & \left[\frac{1}{\sigma\sqrt{T}}(\log S_\$(0) - \log K + rT + \frac{1}{2}\sigma^2 T)\right] \\ -Ke^{-rT}\ \Phi & \left[\frac{1}{\sigma\sqrt{T}}(\log S_\$(0) - \log K + rT - \frac{1}{2}\sigma^2 T)\right]. \end{aligned}$$

As much fun as this is, such formulas are rarely used for pricing. Options are traded, and their prices are determined by supply and demand. We can consider them a given like stock or bond prices. The main use we have for this framework is that we can, if you will, reverse it and determine the value of σ that yields the known option price—this σ is called the *implied volatility*. Just like interest rates in the context of bonds, it serves as a convenient way of quoting option prices.

B.6 Views on Probabilities

This is of course just a very brief glimpse into pricing. One additional facet worth hinting at, though, are the probabilities involved. To illustrate their behavior, we look at a simplified pricing model where the prices of a stock and a bond evolve into only two states (u for “stock up” and d for “stock down”) after some time T:

$$\begin{tikzcd}[row sep=tiny] & S_\$(T) = 20 \\ S_\$(0) = 10 \arrow[ur] \arrow[dr] & \\ & S_\$(T) = 5 \end{tikzcd} \qquad \begin{tikzcd}[row sep=tiny] & B_\$(T) = 1 \\ B_\$(0) = 0.9 \arrow[ur] \arrow[dr] & \\ & B_\$(T) = 1 \end{tikzcd}$$

We are mainly interested in prices with respect to no-arbitrage assets. Here are all asset prices with respect to the bond:

$$\begin{tikzcd}[row sep=tiny] & S_B(T) = 20 \\ S_B(0) = 11.11 \arrow[ur] \arrow[dr] & \\ & S_B(T) = 5 \end{tikzcd} \qquad \begin{tikzcd}[row sep=tiny] & B_B(T) = 1 \\ B_B(0) = 1 \arrow[ur] \arrow[dr] & \\ & B_B(T) = 1 \end{tikzcd}$$

This is the view we adopted in pricing the option above. But of course we can also express the asset prices with respect to the stock—unlike us, this is how Bill Gates might view the world:

$$\begin{tikzcd}[row sep=tiny] & S_S(T) = 1 \\ S_S(0) = 1 \arrow[ur] \arrow[dr] & \\ & S_S(T) = 1 \end{tikzcd} \qquad \begin{tikzcd}[row sep=tiny] & B_S(T) = 0.05 \\ B_S(0) = 0.09 \arrow[ur] \arrow[dr] & \\ & B_S(T) = 0.20 \end{tikzcd}$$

What probabilities p_u and $p_d = 1 - p_u$ should we—in our bond view—assign to the two outcomes? Ruling out arbitrage tells us:

$$S_B(0) = 11.11 = \mathbb{E}[S_B(T)] = p_u \times 20 + (1 - p_u) \times 5 \implies p_u = 0.407.$$

How about Bill Gates? He wants to assume the following:

$$B_S(0) = 0.09 = \mathbb{E}[B_S(T)] = p_u \times 0.05 + (1 - p_u)0.20 \implies p_u = 0.733.$$

Whoa—the probabilities differ! We see that depending on the reference asset used, the no-arbitrage condition entails different probabilities. We’d best rename those distinct probabilities for the “stock up” scenario to p_u^B for our bond-based view and to p_u^S for Bill’s stock-based one. We end up with two ways of computing the expectations involved:

$$\mathbb{E}^B[X] = p_u^B x_u + (1 - p_u^B) x_d,$$
$$\mathbb{E}^S[X] = p_u^S x_u + (1 - p_u^S) x_d.$$

We have, by construction,

$$\mathbb{E}^B[S_B(T)] = 11.11 = S_B(0),$$
$$\mathbb{E}^S[B_S(T)] = 0.09 = B_S(0),$$

as well as

$$\mathbb{E}^B[B_S(T)] = 0.14 \neq B_S(0),$$
$$\mathbb{E}^S[S_B(T)] = 16 \neq S_B(0).$$

How to price a contract **C** that pays out the stock **S** in the "stock up" scenario and nothing in the "stock down" one? Under the bond view, getting **S** is identical to getting $S_B(T)\mathbf{B}$, and the payoff (in bond terms) of this contract at time T is thus

$$C_B(T) = \begin{cases} S_B(T) = 20 & \text{in the "stock up" scenario,} \\ 0 & \text{otherwise.} \end{cases}$$

Its current price is

$$C_B(0) = \mathbb{E}^B[C_B(T)] = p_u^B \times 20 + (1 - p_u^B) \times 0 = 8.15.$$

Under the stock view, the contract payoff at time T is even simpler:

$$C_S(T) = \begin{cases} 1 & \text{in the "stock up" scenario,} \\ 0 & \text{otherwise.} \end{cases}$$

We have

$$C_S(0) = \mathbb{E}^S[C_S(T)] = p_u^S \times 1 + (1 - p_u^S) \times 0 = 0.73.$$

Yet both views agree on the dollar price:

$$C_\$(0) = C_B(0)B_\$(0) = 8.15 \times 0.9 = 7.33,$$
$$C_\$(0) = C_S(0)S_\$(0) = 0.73 \times 10 = 7.33.$$

The more natural way to price such a stock-affine payoff is Bill's stock view. Although multiplying by 20 in the bond view is certainly doable here, this step falls away for Bill. Hopefully, this lets you imagine that in the continuous case, where we have to evaluate integrals, a suitable problem formulation can bring about considerable simplifications.

We can use much the same reasoning with our call option. We have used a (bond-based) probability density that made sure that $S_B(0) = \mathbb{E}^B[S_B(T)]$. (Note: we usefully renamed the expectation just like above.) There is an alternative density for $S_B(\cdot)$ that allows us to evaluate expectations under the stock view as well, i.e., expressions of the form $\mathbb{E}^S[f(S_B(T))]$.[13]

[13]There are also two densities for $B_S(\cdot)$, corresponding to the two expectations. One of them neatly makes sure that $B_S(0) = \mathbb{E}^S[B_S(T)]$.

Expressing the call payoff via the indicator function as in

$$\mathbf{C} \sim^T \mathbb{1}_{S_B(T)\geqslant k}\mathbf{S} - k\mathbb{1}_{S_B(T)\geqslant k}\mathbf{B}$$

lets us then derive the call's current price from

$$\mathbf{C} \sim^0 \mathbb{E}^S[\mathbb{1}_{S_B(T)\geqslant k}]\mathbf{S} - k\,\mathbb{E}^B[\mathbb{1}_{S_B(T)\geqslant k}]\mathbf{B}.$$

Possibly even niftier: we can also selectively use $B_S(T)$ in this expression (recall that $B_S = 1/S_B$ and that the reciprocal of a log-normal distribution is also log-normal and helpfully preserves σ) and thereby use the "canonical" distributions under each expectation:

$$\mathbf{C} \sim^0 \mathbb{E}^S[\mathbb{1}_{B_S(T)\leqslant \frac{1}{k}}]\mathbf{S} - k\,\mathbb{E}^B[\mathbb{1}_{S_B(T)\geqslant k}]\mathbf{B}.$$

Computing these expectations also yields the Black-Scholes formula.

Further Reading C

An excellent book about the basis of it all—debt and money—is Graeber's "Debt: The First 5000 years" (Graeber 2014), which outlines how debt preceded and indeed paved the way for money and the subsequent financial products and markets. Some insight into why those markets may behave the way they do can be found in Akerlof and Shiller's "Animal Spirits" (Akerlof and Shiller 2010).

An extensive market risk classic is Jorion's "Value at Risk" (Jorion 2006), and many other general reference resources are available online.[14] An overview of the risk landscape and the particular role of market risk in it is given in Allen (2009). Many of the core concepts compiled in the book you are holding can be found in Ortega et al. (2009), a paper by my former work colleagues and creators of the initial version of our scenario generator. The historical VaR approach championed in this book belongs to the family of filtered historical simulations (Barone-Adesi et al. 1999, 2008). The BRW model is a commonly-encountered alternative (Boudoukh et al. 1998).

Artzner et al. (1999) shine some light on desirable properties of risk measures and introduce the influential concept of *coherent* measures. An in-depth treatment of risk measures' verificability can be found in Ziegel (2014). A workaround for the usually unstable additive decomposition of VaR to individual positions is presented in Epperlein and Smillie (2006). Anyone using p-values to make a point might find (Wasserstein and Lazar 2016) useful.

In the context of a VaR model, you'll inevitably encounter issues of pricing and arbitrage, topics we hinted at only very briefly. A great gateway into this world is Jan Vecer's "Stochastic Finance: A Numeraire Approach" (Vecer 2011). He neatly distinguishes between assets and their prices, concepts often intermingled in traditional notations. He also doesn't dwell on technical details and emphasizes explicit step-by-step calculations. Then either head down the math alley with Shreve's excellent books, especially (Shreve 2008), or get a comprehensive and less

[14] www.value-at-risk.net.

formal overview on pricing with Hull's standard reference "Options, Futures and Other Derivatives" (Hull 2011).

Books by practitioners can then greatly help you with more arcane products (Zhang 1996), tricky issues of calibration to market data (Rebonato 2002), and explicit algorithms (Brigo and Mercurio 2007). Supplement your modeling skills with the invaluable (Kutner et al. 2004). Finally, make sure to check out Glasserman's superb "Monte Carlo Methods in Financial Engineering" (Glasserman 2003). It is very accessible, and many of the presented methods, e.g., variance reduction techniques, can not only be used in pricing but also in our simple VaR model setup.

If you want to expose yourself to the wide and fast-paced IT-field, it can't hurt to understand its slang. Browse, for example, through the table of contents in Sommerville (2015), and try to zoom in on unfamiliar terms until your have a grasp of their meaning. Soon you should be able to roughly decipher the programmers' gobbledygook ("we have deployed unit testing to the grid"). For managing IT projects, consider looking into *agile software development* (Martin 2002).

Then learn about the Linux operating system (you can install one on a virtual machine[15] on your Windows desktop) and familiarize yourself with its command line interface (Powers et al. 2002). To actually learn how to program, start off with the programming language C, best with the concise and very elegant (Kernighan and Ritchie 1989). Once you master the concept of pointers, feel free to speed up your progress by learning Python (Gaddis 2014), possibly via some of the excellent online courses available.[16] Python also allows you to learn about object-oriented programming. Once you understand why a "square" class should not inherit from the "rectangle" one, you are ready for C++ (Stroustrup 2013), design patterns (Gamma et al. 1994), and UML (Fowler 2004). A tool for creating UML diagrams—high-level representations of object-oriented code—is UMLet.[17]

As for mathematical and statistical support tools, definitely check out NumPy[18] (a Python add-on) or R.[19] (NumPy, unlike R, uses 0-based indexing, which is better.[20]) Many of the examples in this book can be reenacted in Excel or via supporting Monte Carlo add-ins like MonteCarlito.[21] Finally, drop by at this book's www.value-at-risk.com.

[15]www.virtualbox.org.

[16]www.codecademy.com/learn/python.

[17]www.umlet.com (full disclosure: tool by author).

[18]www.numpy.org.

[19]www.r-project.org.

[20]www.cs.utexas.edu/users/EWD/transcriptions/EWD08xx/EWD831.html—or google "Edsger Dijkstra why numbering should start at zero" should this link prove unstable.

[21]www.montecarlito.com (tool by author).

References

Akerlof, G. A., & Shiller, R. J. (2010). *Animal spirits*. Princeton University Press: Princeton.

Allen, S. L. (2009). *Financial risk management*. Wiley: Hoboken.

Artzner, P., Delbaen, F., Eber, J. -M., & Heath, D. (1999). Coherent measures of risk. *Mathematical Finance*, *9*(3) , 203–228.

Barone-Adesi, G., Engle, R., & Mancini, L. (2008). A GARCH option pricing model with filtered historical simulation. *Review of Financial Studies*, *21*(3), 1223–1258.

Barone-Adesi, G., Giannopoulos, K., & Vosper, L. (1999). VaR without correlations for non-linear portfolios. *Futures Markets*, *19*, 583–602.

Boudoukh, J., Richardson, M., & Whitelaw, R. (1998). The best of both worlds. *Risk*, *11*, 64–67.

Brigo, D., & Mercurio, F. (2007). *Interest rate models*. Springer: Berlin

Epperlein, E., & Smillie, A. (2006). Cracking VAR with kernels. *Risk*, *19*(8), 70–74.

Fowler, M. (2004). *UML distilled: A brief guide to the standard object modeling language*. Addison-Wesley: Boston.

Gaddis, T. (2014). *Starting out with python*. Pearson: Boston.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994) *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley: Boston.

Glasserman, P. (2003) *Monte Carlo methods in financial engineering*. Springer: New York.

Graeber, D. (2014). *Debt: The first 5000 years*. Melville House: Brooklyn.

Hull, J. C. (2011) *Options, futures and other derivatives*. Prentice Hall: Upper Saddle River.

Jorion, P. (2006) *Value at risk*. McGraw-Hill: New York.

Kernighan, B. W., & Ritchie, D. (1989) *The C programming language*. Prentice Hall: Upper Saddle River.

Kutner, M., Nachtsheim, C., & Neter, J. (2004) *Applied linear regression models*. McGraw-Hill: New York.

Martin, R. C. (2002) *Agile software development: Principles, patterns, and practices*. Pearson: Upper Saddle River.

Ortega, J. -P., Pullirsch, R., Teichmann, J., & Wergieluk, J. (2009). A new approach for scenario generation in risk management. preprint arXiv:0904.0624.

Powers, S., Peek, J., O'Reily, T., & Loukides, M. (2002). *Unix power tools*. O'Reilly: Sebastopol.

Rebonato, R. (2002). *Modern pricing of interest-rate derivatives*. Princeton University Press: Princeton.

Shreve, S. E. (2008). *Stochastic calculus for finance II: Continuous-time models*. Springer: New York.

Sommerville, I. (2015). *Software engineering*. Pearson: Upper Saddle River.

Stroustrup, B. (2013). *The C++ programming language*. Addison-Wesley: Boston.

Vecer, J. (2011). *Stochastic finance: A numeraire approach*. CRC Press: Boca Raton.

Wasserstein, R. L., & Lazar, N. A. (2016). The ASA's statement on p-values: Context, process, and purpose. *The American Statistician* (preprint online).

Zhang, P. G. (1996). *Exotic options: A guide to second generation options*. World Scientific Publishing: Singapore.

Ziegel, J. F. (2014). Coherence and elicitability. preprint arXiv:1303.1690.

Index