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.

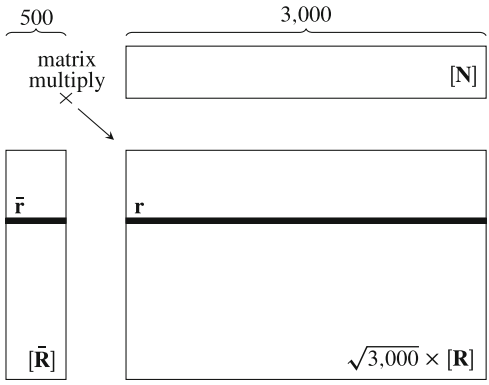

Fig. 10.1 Returns via random averaging

If we repeatedly recalculate the vector expression with new sets of standard normals, we can generate arbitrary many return vectors $\mathbf{R}$ and thus arbitrary many scenarios. Figure 10.1 illustrates how this can be expressed as a matrix-matrix multiplication. With $[\mathbf{N}]$ 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.