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

M. Auer, *Hands-On Value-at-Risk and Expected Shortfall*, Management for Professionals, https://doi.org/10.1007/978-3-319-72320-4_9

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.