A general approach to calculating VaR without volatilities and correlations

Peter Benson *

Peter Zangari
Morgan Guaranty Trust Company
Risk Management Research
(1-212) 648-8641
zangari_peter@jpmorgan.com

In the previous RiskMetrics Monitor[1], we described an alternative to the variance-covariance (VCV) method for portfolio risk analysis. We called this new method portfolio aggregation. Recall that portfolio aggregation involves reconstructing a time series of daily portfolio returns from a current set of portfolio positions and daily returns on individual securities. Value-at-Risk (VaR) estimates of such a portfolio are then obtained by computing the portfolio standard deviation directly from the portfolio return series instead of constructing individual volatilities and correlations.

In this note, we describe the data used in portfolio aggregation, and introduce more applications to risk analysis, including Monte Carlo simulation. We provide a general framework that end-users can use to produce estimates of VaR. As a specific example of this approach, we show how to employ Monte Carlo simulation without computing a covariance matrix. The rest of this article is organized as follows:

- In section 1 we demonstrate how to compute value-at-risk using without a covariance matrix. Specifically, we show how to calculate VaR directly from the underlying return series under equal and exponential weighting schemes. In the case where exponential weighting is applied, we show that this VaR calculation is identical to that used by RiskMetrics™ VCV methodology.
- In section 2 we explain how to compute Monte Carlo without first constructing a variance/covariance matrix.
- Section 3 briefly mentions other applications where covariance matrix is not required
- Section 4 presents conclusions and direction for future research

1. Using returns time series in place of volatilities and correlations

RiskMetrics provides volatilities and correlations for a set of benchmark securities. These securities are what we use to map actual cashflows. For example, suppose we need to compute the VaR of a cashflow denominated in US dollars that occurs in 6 years time. In order to compute VaR, we would map this cashflow to the two nearest RiskMetrics nodes which represent the 5 and 7 year US zero rates. The volatility of the log changes on the 5 and 7 year nodes as well as the correlation between the two log changes are then used to (1) find how much to allocated to the two nodes, and (2) compute VaR. Note that the 5 and 7 year nodes are what we refer to as the benchmark securities.

In these calculations, each benchmark has associated with it a time series of volatility adjusted returns, i.e., returns divided by their standard deviation. The volatilities and correlations are calculated from the set of all time series, and then these time series are discarded. Here, we suggest that rather than computing the volatilities and correlations and discarding the time series, we work directly with the time series of returns, and bypass the calculation of correlations.

This provides a number of advantages:

- Existing time series can be modified directly, and new time series added, without recalculating the correlations with other time series
- If returns are to be weighted (e.g, exponentially), the weighting scheme can be altered without replacing the dataset

* Peter Benson, previously with Risk Management Research at J.P. Morgan, is now with Greenwich Capital Markets Inc. in Connecticut.

[1] See the article, *Streamlining the market risk measurement process*.

- If there are a large number of benchmarks relative to the number of returns per benchmark, the dataset of benchmark returns is smaller than the correlation matrix
- Better numerical precision
- An intuitive interpretation of how Monte Carlo returns are generated
- Fast marginal risk analysis.

We now explain how users can work directly with the return time series (RTS). Define the RTS dataset as follows. For benchmark i, let r_i be the $T \times 1$ vector of returns (with mean 0), where T is the number of returns (historical observations) for each benchmark. Define $r_{i,t}$ to be the return in period t of benchmark i. Let $R = \langle r_1, r_2, ..., r_n \rangle$, where n is the number of benchmarks. We can write the T x n matrix of returns R as follows.

$$[1] \qquad R = \begin{bmatrix} r_{11} & \cdots & \cdots & \cdots & r_{1n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & r_{JJ} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ r_{T1} & \cdots & \cdots & \cdots & r_{Tn} \end{bmatrix}$$

Since R is a $T \times n$ matrix, the RTS dataset has Tn values. Compare this to the VCV dataset which has n standard deviations, and $n(n-1)/2$ correlations, for a total of $n(n+1)/2$ values. For situations where the number of benchmarks is more than twice the number of observations, RTS requires less storage, i.e., R requires less storage than its corresponding covariance matrix.

1.1 Relationship to the covariance matrix

The fact that we can use the returns directly to compute VaR is made obvious from the following observation: the covariance matrix under equally weighted observations is

$$[2] \qquad C = T^{-1} R^T R .$$

where R^T is the transpose of R. Hence, R provides a simple factoring of the covariance matrix. This will be useful later.

1.2 Application to VCV

If w is the vector of benchmark equivalents where each element of w, w_i, is the position value associated with one of the n benchmarks, VaR is given by the equation:

$$[3] \qquad VaR = 1.65\sqrt{w^T C w}$$

assuming that underlying returns are distributed according to the conditional multivariate normal distribution. Now, it follows from [1] that

$$[4] \qquad VaR = 1.65\sqrt{w^T C w} = 1.65\sqrt{w^T T^{-1} R^T R w} = 1.65 T^{-1} \| R w \| .$$

As we can see from the right-hand side of equation [3], the VaR calculation depends only on the benchmark weights, w, the underlying return matrix, R, and the number of historical observations T. Because the computational effort varies linearly with the number of benchmarks, using the RTS matrix is faster

than using the covariance matrix C (provided the number of benchmarks is more than twice the number of observations in each benchmark).

The preceding analysis demonstrates how to compute VaR by the VCV method when the data are equally weighted. However, we note that this methodology is general and applies equally well when the data are weighted exponentially.

1.3 Applying exponential weighting

We now show how similar results to those presented in section 1.2 are obtained when applying exponential weighting. When computing the covariance and correlation matrices, use, instead of the data matrix R, the augmented data matrix $\tilde{R}$ shown in equation [5].

$$[5] \qquad \tilde{R} = \begin{bmatrix} r_{11} & \cdots & \cdots & \cdots & r_{1n} \\ \sqrt{\lambda} r_{21} & \cdots & \cdots & \cdots & \sqrt{\lambda} r_{21} \\ \cdots & \cdots & \sqrt{\lambda^{J-1}} r_{JJ} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \sqrt{\lambda^{T-1}} r_{T1} & \cdots & \cdots & \cdots & \sqrt{\lambda^{T-1}} r_{Tn} \end{bmatrix}$$

Now, we can define the covariance matrix based on exponential weighting simply as

$$[6] \qquad \begin{aligned} \tilde{C} &= \left(\sum_{i=1}^{T} \lambda^{i-1} \right)^{-1} \cdot \tilde{R}^T \tilde{R} \\ &= \Lambda^{-1} \cdot \tilde{R}^T \tilde{R} \end{aligned}$$

where

$$[7] \qquad \Lambda = \sum_{i=1}^{T} \lambda^{i-1}$$

It follows immediately from the results presented in [4] that VaR in this case is

$$[8] \qquad VaR = 1.65 \Lambda^{-1} \left\| R^T w \right\|$$

Once a decay factor, λ, is selected in the VCV method, the correlation matrix is computed, and it is no longer possible to change the weight without recomputing the entire covariance matrix. However, since there are no assumed weights in the RTS dataset, we are free to choose different values without any additional computational burden.

2. Generating multivariate normal returns for Monte Carlo

To generate multivariate normal returns, one typically performs a Cholesky or Singular Value decomposition (SVD) on the correlation matrix[2]. The resulting matrix is then combined with a vector of random variates to produce multivariate returns.

There are drawbacks to this approach:

[2] See Appendix E of the RiskMetrics Technical document, 4th edition.

- The decomposed matrix does not easily provide an intuitive understanding of how the deviates are generated
- Changing a single return value of one benchmark requires a new decomposition
- Cholesky decomposition requires that the correlation matrix be PD (positive definite), and SVD requires PSD (positive semi-definite).[3]

Using the RTS dataset, we can take a more direct approach that suggests an intuitive interpretation. Consider a row of the R matrix. It represents one observation interval--a snapshot of all benchmark returns. Suppose we multiply each row by a normally distributed random variate, and add the rows. The result is a vector of Monte Carlo returns that have the correct variance and correlations. In other words, Monte Carlo returns are simply the sum of random multiples of the benchmark snapshots.

We now show how to perform Monte Carlo with the RTS that avoids any volatility and correlation calculations.
Let $\varepsilon = \{\varepsilon_1, \varepsilon_2, ..., \varepsilon_T\}$ be a $T \times 1$ vector of independent, N(0,1) random variates. Then we are interested in simulating a $1 \times n$ vector of correlated returns, $\varepsilon^T R$. Consider the ith component, $\hat{r}_i = \varepsilon^T r_i$. Note that $\varepsilon^T R$ is just a sum of independent, normally distributed random variables (all with mean zero, and standard deviation vector r_i). So the mean of the sum is the sum of the means which are zero. And the variance is the sum of the variances, which is the variance of r_i. So the mean and variance of $\hat{r}_i$ is the same as that of r_i. Unlike r_i, $\hat{r}_i$ is truly normally distributed.

What about the correlation of $\hat{r}_i$ with other simulated returns? Consider another simulated return, $\hat{r}_j$. The correlation is $\hat{\rho}_{ij} = cov\left(\left(\hat{r}_i, \hat{r}_j\right)/\sigma_i\sigma_j\right)$. Since $\hat{r}_i$ and $\hat{r}_j$ have mean zero,

$$[9] \qquad \begin{aligned} cov(\hat{r}_i, \hat{r}_j) &= E[\hat{r}_i \cdot \hat{r}_j] \\ &= E\left[\Sigma_{t=1}^{T} \varepsilon_t r_{i,t} \Sigma_{t=1}^{T} \varepsilon_t r_{j,t}\right] \\ &= E\left[\Sigma_{t=1}^{T} \varepsilon^2{}_t r_{i,t} r_{j,t} + 2\Sigma_{t=1}^{T} \Sigma_{s=t+1}^{m} (\varepsilon_t \varepsilon_s r_{i,t} r_{j,t})\right] \end{aligned}$$

Because the ε_i are independent, the cross terms have zero expected value. Since the expectation of the sum equals the sum of the expectation, the right-hand side becomes

$$[10] \qquad E\left[\Sigma_{t=1}^{T} \varepsilon^2{}_t r_{i,t} r_{j,t}\right] = \Sigma_{t=1}^{T} r_{i,t} r_{j,t} E\varepsilon_t^2 = \Sigma_{t=1}^{T} r_{i,t} r_{j,t} = cov(r_i, r_j)$$

Hence, covariance and correlations are also preserved. In general, we can show that the covariance matrix of the 1 x n vector of random variables $y=\varepsilon^T R$ is $C=R^T R$. This follows immediately from the definition of the variance of y where the mean of y is zero. Letting E(x) denote the mathematical expectation of x, we can write the variance of y as follows:

[3] The correlation matrix cannot be PD if there are more benchmarks (columns of R) than there are observations in each benchmark (rows of R). In fact, due to roundoff errors, even PSD is rarely satisfied. These problems can be dealt with by tweaking the correlation matrix to become PSD (which introduces other errors), or by increasing the number of observations in each benchmark (which bloats the dataset without adding significant information, and may require data that is unavailable, or outside the sample of interest). The RTS dataset allows analysis with fewer observations. Of course, one must still be careful that the chosen dataset is sufficiently representative of the relationships between benchmarks.

$$[11] \qquad \begin{aligned} Variance(y) &= E\left(y^T y\right) = E\left(R^T \varepsilon \varepsilon^T R\right) \\ &= R^T E\left(\varepsilon \varepsilon^T\right) R \\ &= C \end{aligned}$$

Now, certainly Monte Carlo with a covariance matrix represents a performance improvement in terms of pre-processing (i.e. in decomposing a correlation matrix), since we do not need any pre-processing. However, is it a fast way to generate deviates? Once the T independent normal deviates are generated, simulating each benchmark uses T multiples. This compares with k (where k is the rank of R) multiplications when using a Cholesky or SVD decomposition.

However, we can make return generation still faster. By randomly sampling the observation vectors (i.e. using only a subset), we can still preserve correlations and variances. In fact, we can generate our Monte Carlo returns with just one random normal deviate per trial, and one multiply per benchmark. Details are left for future discussion.

3. Other applications

The RTS dataset also allows more powerful tools for marginal analysis, and ad hoc manipulation of the benchmark returns. The Monte Carlo sampling technique discussed above, as well as marginal analysis techniques based on RTS, are employed by CreditManager™ (J.P. Morgan's credit risk calculator) for Monte Carlo simulation of credit portfolios. Details of these methods may be covered in a subsequent note.

4. Conclusions

As the number of benchmarks grows relative to the effective number of observations per benchmark, it becomes more efficient to use the RTS dataset. Moreover, using the RTS dataset provides a number of benefits over VCV-based approaches in terms of flexibility. In particular, we've shown that it provides an intuitively appealing technique for generating Monte Carlo samples.