A C++ scoring algorithm for ranking global currencies based on financial market sentiment

C++ Programming
Master 203 Financial Markets - Université Paris-Dauphine
December 2017 - January 2018

Suppose an investor wishes to rank global currencies based on available market information. It would be erroneous to rank those currencies based on their pairwise level.

[Explain briefly methodology using risk reversals]

[Provide results for real-world snapshot]

[Interpretation]

[Alternative methods]

Least squares implementation

Suppose we have an **antisymmetric matrix** $M \in \mathcal{M}_{3\times3}(\mathbb{R})$ where $M' = -M$:

$$M = \begin{pmatrix} 0 & M_{12} & M_{13} \\ M_{21} & 0 & M_{23} \\ M_{31} & M_{32} & 0 \end{pmatrix} = \begin{pmatrix} 0 & M_{12} & M_{13} \\ -M_{12} & 0 & M_{23} \\ -M_{13} & -M_{23} & 0 \end{pmatrix}$$

And an **outer-score difference matrix** $O \in \mathcal{M}_{3\times3}(\mathbb{R})$ of a score vector S, where $O_{i,j} = S_i - S_j$:

$$O = \begin{pmatrix} 0 & S_1 - S_2 & S_1 - S_3 \\ S_2 - S_1 & 0 & S_2 - S_3 \\ S_3 - S_1 & S_3 - S_2 & 0 \end{pmatrix}$$

By using the least squares criterion, we can use the properties of the Frobenius Norm to formulate the approximation of M by O as a least squares problem:

$$\|M - O\|_2^2 = \left\| \begin{pmatrix} 0 & M_{12} - (S_1 - S_2) & M_{13} - (S_1 - S_3) \\ M_{21} - (S_2 - S_1) & 0 & M_{23} - (S_2 - S_3) \\ M_{31} - (S_3 - S_1) & M_{32} - (S_3 - S_2) & 0 \end{pmatrix} \right\|_2^2$$

Since the **Frobenius Norm** for a given matrix A is equal to $\|A\|_2 = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$. Thus:

$$\|M - O\|_2^2 = [M_{12} - (S_1 - S_2)]^2 + [M_{13} - (S_1 - S_3)]^2 + [M_{21} - (S_2 - S_1)]^2 + [M_{21} - (S_2 - S_1)]^2 + [M_{23} - (S_2 - S_3)]^2 + [M_{31} - (S_3 - S_1)]^2 + [M_{32} - (S_3 - S_2)]^2$$

Which can be formulated as a **least-squares problem with an overdetermined system of linear equations**:

$$Y = X \cdot S$$

$$\begin{pmatrix} M_{12} \\ M_{13} \\ M_{21} \\ M_{23} \\ M_{31} \\ M_{32} \end{pmatrix} = \begin{pmatrix} S_1 - S_2 \\ S_1 - S_3 \\ S_2 - S_1 \\ S_2 - S_3 \\ S_3 - S_1 \\ S_3 - S_2 \end{pmatrix} = \begin{pmatrix} \mathbf{1} & -1 & 0 \\ \mathbf{1} & 0 & -1 \\ -1 & \mathbf{1} & 0 \\ 0 & \mathbf{1} & -1 \\ -1 & 0 & \mathbf{1} \\ 0 & -1 & \mathbf{1} \end{pmatrix} \cdot \begin{pmatrix} S_1 \\ S_2 \\ S_3 \end{pmatrix}$$

One way to find a solution for this overdetermined system is through least squares regression. The solution S^* is the solution such that the difference $Y - X \cdot S$ is very small. We choose to solve via **least squares singular value decomposition**, as it is the most accurate least squares method provided by the Eigen numerical linear algebra package for C++.