MSBD 5003: Big Data Computing Project Report Statistical Factor Analysis

WANG Jin 20550441
RAJPUT Kabir 20560795
LEUNG Terence 20581189
DE LAVERGNE Cyril 20109216

The Hong University of Science and Technology

May 2019

Abstract

Statistical factors are essential to risk managers in the finance industry to mitigate risk assigned to each asset returns in one portfolio. We compute statistical factors on a universe of more than 400 cryptocurrencies available on the trading platform named Binance on one minute bar data over the past year. The size of this dataset cannot usually easily manipulated on a regular computer due to its size and the complexity of operations to be carried out. Hence, we design an algorithm in PySpark aiming to solve risk managers' problems.
Keywords: Statistical Factor, Financial Risk Management, Cryptocurrency Market, Financial Time Series Analysis, Data Mining, Machine Learning

1 Previous Work

The capital asset pricing model (Sharpe W., 1964) and five asset pricing model (Fama E. and French K., 2013) are popular in the financial industry to explain most of the risk and variance across US equities between 1693 and 2013. It follows the traditional Factor model:

$$R_{it} - Rf_t = \alpha_i + \sum \beta_i * F_t + \epsilon_{it}$$

Open source R CRAN project participated in Google Summer of Code (GSoC) 2016 and 2017 aiming to improve the package 'factorAnalytics' in R, the statistical programming language. One of the methods of the package is statistical factors. Statistical factors unlike time series and fundamental factors are not directly observable. In fact, it is extracted performing principal component analysis (Pearson, 1900) or asymptotic principal component analysis (Connor G. and Korajczyk R.A., 1986) on all assets in one universe. If the universe of assets is bigger than the period used to estimate the factors, it was demonstrated that asymptotic principal component analysis should be performed.
Statistical factors explain most of the variance among the components used in the analysis. This can be confirmed through the R squared standard linear regression of each asset over PCA or APCA first component in decreasing order of importance.
The solution we provide is to reliably estimate risk of some assets to explain most of the variance among assets of the universe implemented in PySpark.

2 Data Collection

Data was retrieved from the official API of the Binance platform, one of the most uprising brokers in the cryptocurrency industry. Cryptocurrency platforms are opened 24hours a day and 7days a week. Open High Low Close Volume (OHLCV) data is extracted from the broker REST API as far as possible back in time. For each of the 431 trading pairs on Binance, a Python script was used to query bid/ask quote data with one minute interval, starting from 2nd February 2019 and until the earliest quote available for each pair. From our findings, the API only provides 1 minute quotes for the last 1 year, but provides daily quotes since the inception of the trading pair on the exchange. Hence, dataset is ranging from the 2nd of February 2018 to the 2nd of February 2019 for minute bar data.

The API allows querying a maximum of 1000 data points for every request. While this was sufficient for querying all available daily quote data for a trading pair in a single request, a queuing approach (dramatik.io) was used to retrieve 1 minute quote data using approximately 400 API calls for each trading pair while staying within rate limits.

3 Algorithm Implementation

Stationarity conditions

Statistical analysis of asset prices involves the stationarity hypothesis that when time is shifted, the joint probability distribution does not change. The weak stationary assumption holds when asset prices first two moments are time-invariant. Asset prices indeed involve trend that need to be removed for time series to be stationary. To remove trend from time series, we need to differentiate it. We denote asset prices as C and returns are therefore obtained as such (Cont R., 2000):

$$r(t, \Delta t) = ln(C(t + \Delta t)) - ln(C(t))$$

t is defined as a amount of time between two snapshot of prices. In our paper, we consider it to be the close to close asset return. In the finance literature, it is common to assume that an asset return series is weakly stationary (Tsay R.S.,1999).

Principal Component Analysis

Let a matrix of time series of asset returns defined as:

$$R = \begin{pmatrix} R_1 \\ \vdots \\ R_n \end{pmatrix}$$

We can now compute the mean of observations for each crypto-currency:

$$\mu = \begin{pmatrix} \mu_1 = \frac{\sum_{i=1}^{N} R_1}{N} \\ \vdots \\ \mu_n = \frac{\sum_{i=1}^{N} R_n}{N} \end{pmatrix}$$

Note we attempted to apply a weighted decay to the factor returns, however it seems that Value-at-Risk defined in a later section is underestimated. We first de-mean the returns unless E(X) = 0 to avoid principal component analysis to be biased towards one crypto-currency:

$$R = \begin{pmatrix} R_1 - \mu_1 \\ \vdots \\ R_n - \mu_n \end{pmatrix}$$

We get the unbiased weighted covariance (GEORGE R. PRICE, 1972) as we consider the sample size to be N > T where N stands for the sample of cryptocurrencies and T, the sample of time series considered:

$$\Sigma = \frac{RR'}{T}$$

However, if the universe of assets is bigger than the period used to estimate the factors, it was demonstrated that asymptotic principal component analysis should be performed. Hence, when N <= T we should instead use the asymptotic principal component analysis (Bai, Ng., 2002) with the formula:

$$\Sigma == \frac{RR'}{N}$$

Our covariance matrix can then be represented as such

$$\Sigma = var(X) = \begin{pmatrix} \sigma_{11} & ... & \sigma_{1n} \\ \vdots & \vdots & \vdots \\ \sigma_{n1} & ... & \sigma_{nn} \end{pmatrix} \text{ where } \sigma_{kl} = cov(R_k, R_l)$$

By matrix singular value decomposition, we know

$$\Sigma = e\Lambda e'$$

where

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 & ... & 0 \\ 0 & \lambda_2 & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & \lambda_n \end{pmatrix} \text{ with } \lambda_1 \geq ... \geq \lambda_t > 0$$

and

$$e = \begin{pmatrix} e_1 & \vdots & \cdots & \vdots & e_n \end{pmatrix} = \begin{pmatrix} e_{11} & ... & e_{1n} \\ \vdots & \vdots & \vdots \\ e_{n1} & ... & e_{nn} \end{pmatrix} \text{is an orthogonal matrix}$$

We can now extract factor realizations over the time series sample as such:

$$F = R'e$$

As in the financial literature, β are called loadings L and the model becomes:

$$R_i = \sum L_{it} F_t + \epsilon_i \text{ with } i = 1, \ldots, n$$

Note that intercept becomes null as we have previously demeaned returns. The first statistical factors explain most of the variance among assets used in the analysis.

Factor rotation

To avoid the problem of jumpy loadings, we suggest the simple enhancement:

$$\text{if } median(F_1) < 0 \text{ then } F_1 = -F_1$$

where F_1 denotes the first principal component that explains the most variance in the universe.

Value at Risk

By Euler's theorem, the value-at-risk of asset i's return is:

$$VaR.fm_i = \sum_{k=1}^{K+1} cVaR_{i,k} = \sum_{k=1}^{K+1} B^*_{i,k} mVaR_{i,k}$$

The marginal contribution to VaR.fm is defined as the expectation of F.star, conditional on the loss being equal to VaR.fm. This is approximated as described in Epperlein and Smillie (2006) using a triangular smoothing kernel. This estimate VaR.fm non-parametrically using the sample quantile (default) or assuming a normal distribution.

4 Optimizations

Our implementation includes optimizations from three primary contexts:

DataFrames

DataFrames make it simple to work with large datasets and has low garbage collection overhead. For our project, we used DataFrames to optimize join ordering so that the steps needed to perform a query would be minimized.

Resilient Distributed Datasets

RDDs allow us to parallelize some of our operations. An example in our implementation is when converting factor formats. Mapping was used to reduce the overall amount of data we worked with so that we only used what we needed, and caching was used to store computations that take a longer time to compute. A caveat of caching is the tradeoff between time and space, which can be an issue especially with the size of our dataset. Fortunately, this was not an issue when we tested our implementation.

Code modularization

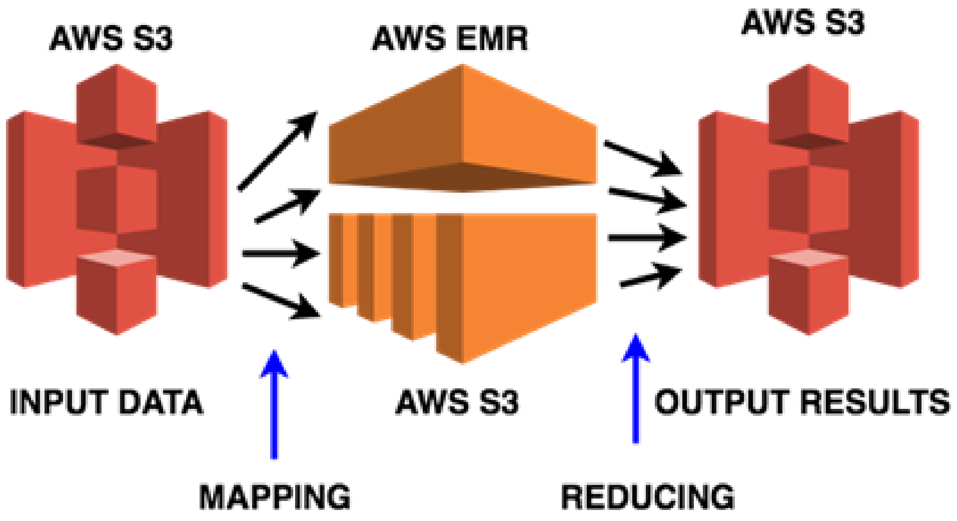
Modularization of code components mainly relates to clean code principles. As such, this optimization is moreso about improving the programming process rather than improving the program execution time. By splitting our code into functional components, it simplified the process of fine-tuning, and allowed for individual components to be tested and parallelized separately.

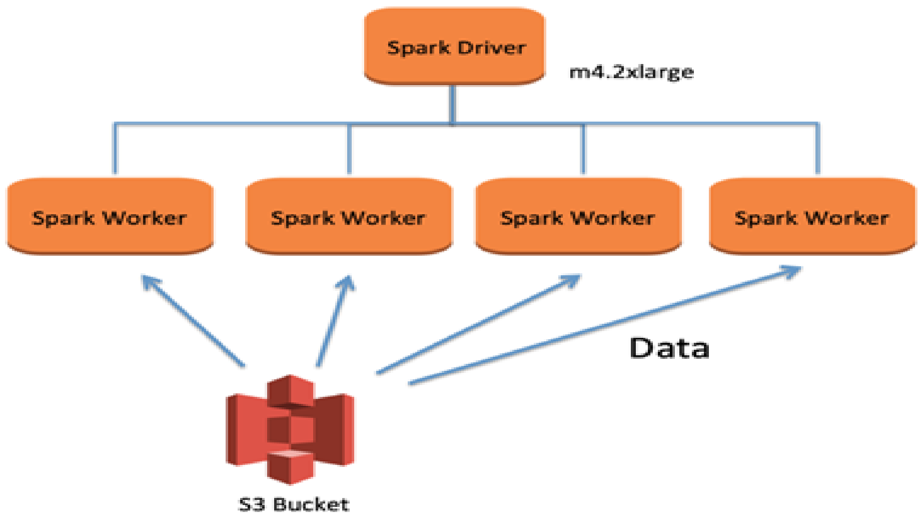
5 AWS Integration

AWS being de facto standard for cloud environment for most of the enterprises and their storage service S3 being very low cost, elastic and promising SLA(availability and durability) is one of the motivations to move away from a relatively expensive storage solutions like HDFS as it involves maintaining a cluster and its related admin costs and support.

Spark leverages Hadoop's "FileOutputCommitter" implementations to write data. Writing data again involves multiple steps and on a high level staging output files and then committing them. A spark job is divided into multiple stages and set of tasks and due to nature of distributed computing the tasks are prone to failure so there is also provision to re-launch same task due to system failure or speculative execution of slow running tasks and that leads to concepts of task commit and job commit functions.

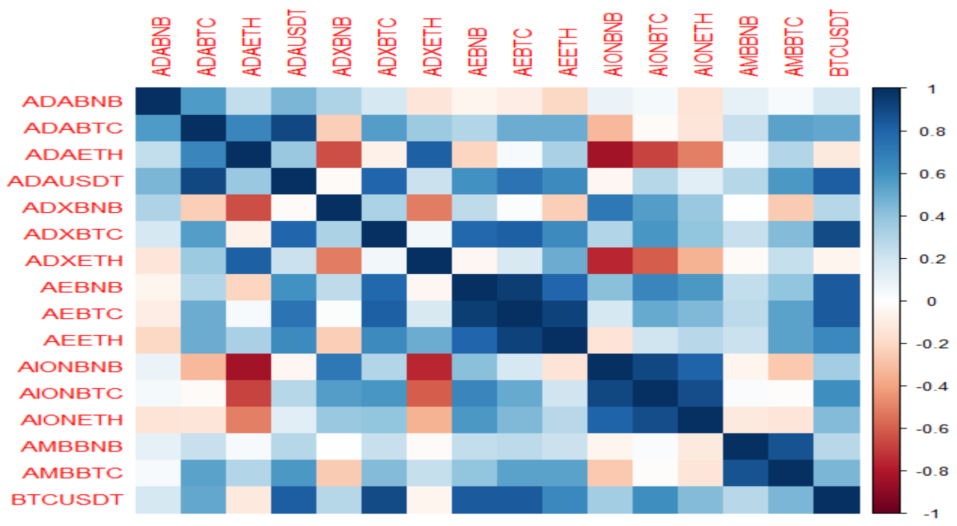
We used Amazon AWS S3 for our data storage which mapped into the AWS EMR and finally gave the output to AWS S3 in the final step. We used m4.2xlarge as the spark driver and connected to 4 spark worker nodes to perform the computation. The following flowchart shows the architecture layout-



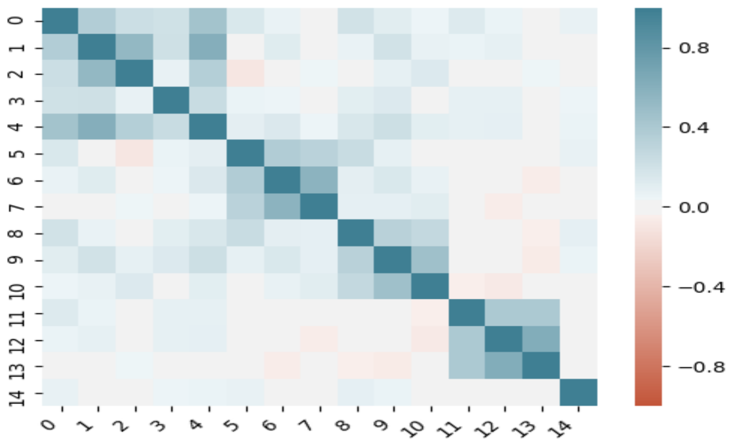


6 Results

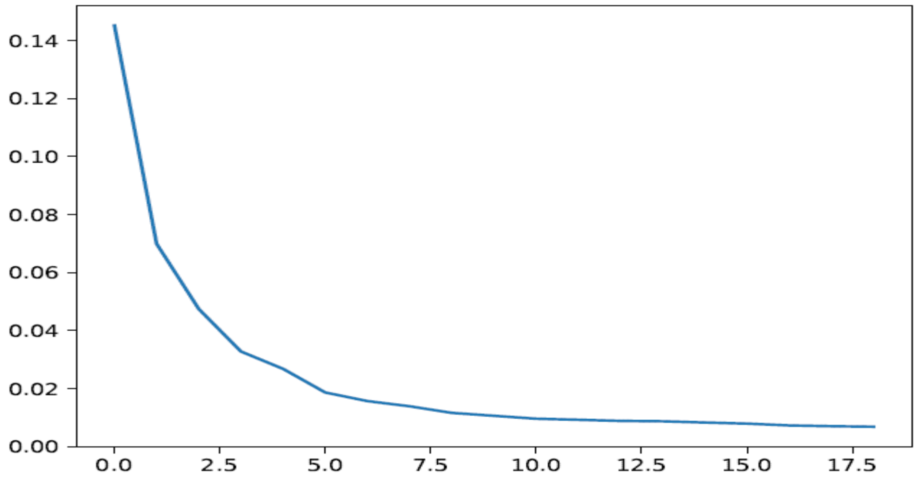
Correlation matrix of a few asset prices to demonstrate the purpose of the application of our algorithm in the cryptocurrency market:



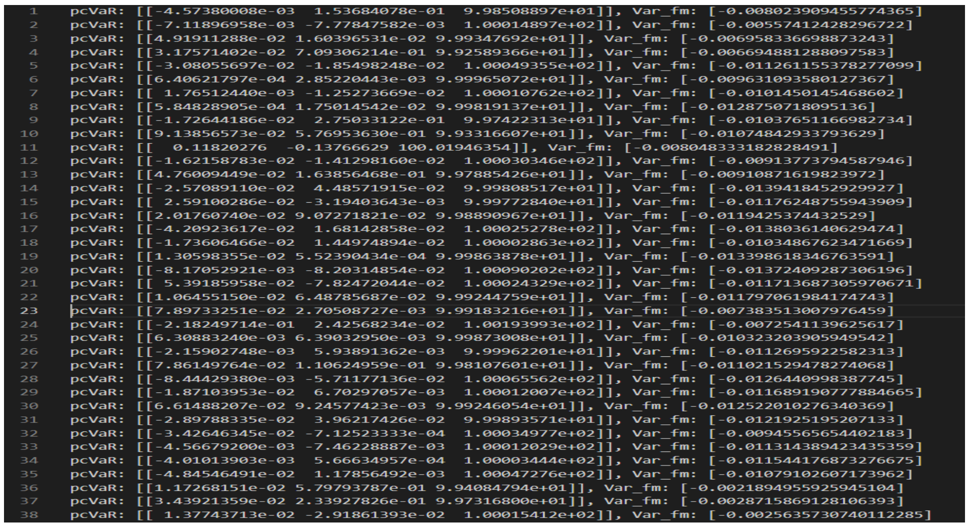
Note that asset prices are mainly positevely correlated. As compared to the correlation matrix of asset returns:



Scree plot explaining the variance of each component in decreasing number of importance:



The Value At Risk of each crypto-currency can then be decomposed between the first X principal components (2 in our case) and the residuals. THe output is as follows:



```
pcVaR: [[-4.57380008e-03  1.53684078e-01  9.98508897e+01]], Var_fm: [-0.008023909455774365]
pcVaR: [[-7.11896958e-03 -7.77847582e-03  1.00014897e+02]], Var_fm: [-0.005574124282967221]
pcVaR: [[4.91911288e-02 1.60396531e-02 9.99347692e+01]], Var_fm: [-0.006958336698873243]
pcVaR: [[3.17571402e-02 7.09306214e-01 9.92589366e+01]], Var_fm: [-0.006694881288097583]
pcVaR: [[-3.08055697e-02 -1.85498248e-02  1.00049355e+02]], Var_fm: [-0.011261155378277099]
pcVaR: [[6.40621797e-04 2.85220443e-03 9.99965072e+01]], Var_fm: [-0.009631093580127367]
pcVaR: [[ 1.76512440e-03 -1.25273669e-02  1.00010762e+02]], Var_fm: [-0.0101450145468602]
pcVaR: [[5.84828905e-04 1.75014542e-02 9.99819137e+01]], Var_fm: [-0.0128750718095136]
pcVaR: [[-1.72644186e-02  2.75033122e-01  9.97422313e+01]], Var_fm: [-0.01037651166982734]
pcVaR: [[9.13856573e-02 5.76953630e-01 9.93316607e+01]], Var_fm: [-0.01074842933793629]
pcVaR: [[  0.11820276  -0.13766629 100.01946354]], Var_fm: [-0.008048333182828491]
pcVaR: [[-1.62158783e-02 -1.41298160e-02  1.00030346e+02]], Var_fm: [-0.00913773794587946]
pcVaR: [[4.76009449e-02 1.63856468e-01 9.97885426e+01]], Var_fm: [-0.00910871619823972]
pcVaR: [[-2.57089110e-02  4.48571915e-02  9.99808517e+01]], Var_fm: [-0.0139418452929927]
pcVaR: [[ 2.59100286e-02 -3.19403643e-03  9.99772840e+01]], Var_fm: [-0.01176248755943909]
pcVaR: [[2.01760740e-02 9.07271821e-02 9.98890967e+01]], Var_fm: [-0.0119425374432529]
pcVaR: [[-4.20923617e-02  1.68142858e-02  1.00025278e+02]], Var_fm: [-0.0138036140629474]
pcVaR: [[-1.73606466e-02  1.44974894e-02  1.00002863e+02]], Var_fm: [-0.01034867623471669]
pcVaR: [[1.30598355e-02 5.52390434e-04 9.99863878e+01]], Var_fm: [-0.013398618346763591]
pcVaR: [[-8.17052921e-03 -8.20314854e-02  1.00090202e+02]], Var_fm: [-0.01372409287306196]
pcVaR: [[ 5.39185958e-02 -7.82472044e-02  1.00024329e+02]], Var_fm: [-0.011713687305970671]
pcVaR: [[1.06455150e-02 6.48785687e-02 9.99244759e+01]], Var_fm: [-0.011797061984174743]
pcVaR: [[7.89733251e-02 2.70508727e-03 9.99183216e+01]], Var_fm: [-0.007383513007976459]
pcVaR: [[-2.18249714e-01  2.42568234e-02  1.00193993e+02]], Var_fm: [-0.0072541139625617]
pcVaR: [[6.30883240e-03 6.39032950e-03 9.99873008e+01]], Var_fm: [-0.010323203905949542]
pcVaR: [[-2.15902748e-03  5.93891362e-03  9.99962201e+01]], Var_fm: [-0.0112695922582313]
pcVaR: [[7.86149764e-02 1.10624959e-01 9.98107601e+01]], Var_fm: [-0.011021529478274068]
pcVaR: [[-8.44429380e-03 -5.71177136e-02  1.00065562e+02]], Var_fm: [-0.0126440998387745]
pcVaR: [[-1.87103953e-02  6.70297057e-03  1.00012007e+02]], Var_fm: [-0.011689190777884665]
pcVaR: [[6.61488207e-02 9.24577423e-03 9.99246054e+01]], Var_fm: [-0.012522010276340369]
pcVaR: [[-2.89788335e-02  3.96217426e-02  9.99893571e+01]], Var_fm: [-0.0121925195207133]
pcVaR: [[-3.42646345e-02 -7.12523333e-04  1.00034977e+02]], Var_fm: [-0.00945565654402183]
pcVaR: [[-4.56679200e-03 -7.46228887e-03  1.00012029e+02]], Var_fm: [-0.011314389423435359]
pcVaR: [[-4.01013903e-03  5.66634957e-04  1.00003444e+02]], Var_fm: [-0.011544176873276675]
pcVaR: [[-4.84546491e-02  1.17856492e-03  1.00047276e+02]], Var_fm: [-0.01079102607173962]
pcVaR: [[1.17268151e-02 5.79793787e-01 9.94084794e+01]], Var_fm: [-0.0021894955925945104]
pcVaR: [[3.43921359e-02 2.33927826e-01 9.97316800e+01]], Var_fm: [-0.0028715869128106393]
pcVaR: [[ 1.37743713e-02 -2.91861393e-02  1.00015412e+02]], Var_fm: [-0.0025635730740112285]
```

7 Future Work and Potential Improvements

There are a few potential improvements to our implementation that can be done for future work.

One improvement is setting up the implementation on a high performance computing cluster using cloud services such as AWS. For our current project, we tested on a local cluster, and although we attempted to migrate our project onto a cloud computing cluster, cost and knowledge constraints that prevented its use. By making use of cloud computing, we would be able to scale our task more efficiently.

Another potential improvement is to use Spark Streaming to compute a sliding window of live, high-throughput data directly from the Binance API. This way, we would be able to aggregate tick data on a minute bar for a more accurate analysis as compared to the hourly data we are currently using.

8 References

Ait-Sahalia Y., Xiu D. (2016) Principal Component Analysis of High Frequency Data

Bai, J., Ng, S. (2002). Determining the number of factors in approximate factor models. Econometrica, 70(1), 191-221.

Connor, G., Korajczyk, R. A. (1988). Risk and return in an equilibrium APT: Application of a new test methodology. Journal of Financial Economics, 21(2), 255-289.

Connor, G., Korajczyk, R. A. (1993). A test for the number of factors in an approximate factor model. The Journal of Finance, 48(4), 1263-1291.

Cont R. (2000) Empirical properties of asset returns: stylized facts and statistical issues. Centre de Mathematiques Appliquees, Ecole Polytechnique.

Epperlein and Smillie (2006) ”Cracking VAR with Kernels,” Risk.

Fama E. F.and French K. R. (2013). A Five-Factor Asset Pricing Model

Hallerback (2003), ”Decomposing Portfolio Value-at-Risk: A General Analysis”, The Journal of Risk 5/2.

Münnix M.C., Schäfer R., Grothe O. (2011) Estimating correlation and covariance matrices by weighting of market similarity, y, Quantitative Finance, DOI: 10.1080/14697688.2011.605075

Price G. R. (1972) Extension of covariance selection mathematics. The Galton Laboratory, University College London.

Yamai and Yoshiba (2002).”Comparative Analyses of Expected Shortfall and Value-at-Risk: Their Estimation Error, Decomposition, and Optimization Bank of Japan.

Zivot E., Srinivasan S. and Chen Y.A. (2015). FactorAnalytics package. Extracted from https://github.com/AvinashAcharya/factorAnalytics.