Approximating Sample Correlation and Covariance Matrices in R

A Financial Application

Kristian Zefi

A Dissertation presented for the Degree:
Masters of Statistics and Finance

University of Kent
United Kingdom
28/08/2020

Contents

Acknowledgements

I would like to thank my supervisor Dr. Alfred Kume for all of the support and guidance he has provided me throughout my project.

1 Summary

This Dissertation is inspired by the methodology Higham proposed in his 2002 paper 'Computing the Nearest Correlation Matrix '. We explore the methodology and see it applied to several examples including a financial data set taken from 2019. We outline techniques that can be used to generate approximations for a matrix with missing data. These matrices may not be positive semi-definite so Dykstra's Alternating Projections Algorithm is used to find the nearest positive semi-definite correlation matrix to allow for subsequent financial analysis. We highlight some errors that may occur computationally due to machine imprecision and also use Lanczos' Iteration to efficiently reduce the algorithms computational cost by only using the eigenvalues and corresponding eigenvectors of interest. Two methods are explored to find an approximate sample covariance matrix, one of which based on using Dykstra's Alternating Projections Algorithm along with the algebraic relationship between correlation and covariance matrices and the other by possibly projecting a covariance matrix that is not positive semi-definite onto the positive semi-definite set. Furthermore we will apply these results to find the Global Minimum Variance Portfolio and see the practical implications of each method when applied to a dataset of returns taken from 5 large companies with a market cap of at least 100 billion dollars.

2 Introduction

Investors and Financial Institutions do not spend money without rationalising their positions. One way they can do this is through analysing large matrices of historical data based upon stock returns, these can inform an investor how risky a stock is and the expected returns it can generate. More often than not, there are days which the returns cannot be reported (over the weekends for example) so we are left with several missing data points. To tackle this problem we can use imputation techniques to approximate the missing data from the days price data is available or even remove those missing data points if certain conditions are met and continue with the statistical analysis. This can generate a symmetric matrix which looks like a correlation matrix but may not necessarily be positive semi-definite. Therefore further methods are required to generate a correlation matrix which is acceptable for subsequent use in finance, this is one area where the nearest correlation matrix problem arises.

Variance is essential due to its important representation as risk. If a stock has high levels of variance, risk averse investors may deem it to be an unfavourable investment. Covariance shows the directional relationship between stocks. These obviously rely on an a covariance matrix which is accurate and reliable, a similar approximation problem occurs when a covariance matrix is approximated from inconsistent data. The resulting matrix may end up singular which does not allow for further financial analysis such as in portfolio optimization.

3 Mathematical Theory

We will begin by highlighting some mathematical concepts and definitions that will be used throughout the project and refer back to them when required. Relevant readings and proofs for each topic will be cited for possible further study.

3.1 Eigenvalues, Eigenvectors and Eigendecomposition

Vector $\boldsymbol{v}$ is an eigenvector of $\boldsymbol{X}$ with a corresponding eigenvalue $\boldsymbol{\lambda}$ if they satisfy the linear equation $\boldsymbol{Xv} = \boldsymbol{\lambda v}$. Let $\boldsymbol{X} \in \boldsymbol{\Re^{n \times n}}$ be a square matrix with linearly independent eigenvectors, $\boldsymbol{X}$ can then be deconstructed via eigen-decomposition(or equivalently spectral decomposition) as

$$\boldsymbol{X = VQV^T} \tag{1}$$

such that $\boldsymbol{V}$ is an orthogonal matrix ($\boldsymbol{V^{-1} = V^T}$) with eigenvectors along the columns and $\boldsymbol{Q}$ is a diagonal matrix with the corresponding eigenvalues across the diagonal. Another equivalent more commonly seen form is $\boldsymbol{X = VQV^{-1}}$. Much has been studied on the eigen-decomposition of a matrix and its properties (Roger A. Horn et. al (1985) for example).

A useful known property is that the determinant of a matrix is equal to the product of its eigenvalues: i.e. for eigenvalues $\boldsymbol{\lambda_1..\lambda_n}$,

$$\boldsymbol{det(X) = \prod \lambda_1 ... \lambda_n.} \tag{2}$$

(Beauregard & Fraleigh 1973)

3.2 Singularity

Square matrix $\boldsymbol{A}$ is said to be singular if the matrix inverse of $\boldsymbol{A}$ does not exist. That is, a matrix is singular if and only if its determinant is equal to 0.
(Horn & Johnson 1985 p.14)

3.3 Sample Correlation and Standard Deviation

Let $\boldsymbol{X}$ be a real $\boldsymbol{p \times q}$ data matrix consisting of random variables in the columns with $\boldsymbol{n}$ observations each. The Sample correlation coefficient between the random variables $\boldsymbol{p}$ and $\boldsymbol{q}$ is

$$\boldsymbol{r_{pq}} = \frac{\sum_{i=1}^{n}(\boldsymbol{p_i} - \bar{\boldsymbol{p}})(\boldsymbol{q_i} - \bar{\boldsymbol{q}})}{\sqrt{\sum_{i=1}^{n}(\boldsymbol{p_i} - \bar{\boldsymbol{p}})^{\boldsymbol{2}}}\sqrt{\sum_{i=1}^{n}(\boldsymbol{q_i} - \bar{\boldsymbol{q}})^{\boldsymbol{2}}}} = \frac{\sum_{i=1}^{n}(\boldsymbol{p_i} - \bar{\boldsymbol{p}})(\boldsymbol{q_i} - \bar{\boldsymbol{q}})}{\boldsymbol{(n-1)\sigma_p\sigma_q}} \tag{3}$$

(George R Terrell, section 2.7)
such that $\boldsymbol{\sigma_q} = \sqrt{\frac{\boldsymbol{1}}{\boldsymbol{n-1}}\sum_{i=1}^{n}(\boldsymbol{q_i} - \bar{\boldsymbol{q}})^{\boldsymbol{2}}}$ is the sample standard deviation of $\boldsymbol{q}$ and the same form for $\boldsymbol{p}$. The corresponding sample correlation matrix $\boldsymbol{R} \in \boldsymbol{\Re^{n \times n}}$ is filled with correlation coefficients between respective random variables. These matrices have $\boldsymbol{(a)}$ unit diagonal with off diagonal values limited by $\boldsymbol{|r_{ij}| \leq 1}$ and $\boldsymbol{(b)}$ the matrix is positive semi-defnite in nature. The reasoning behind $\boldsymbol{(a)}$ is trivial and we will explain $\boldsymbol{(b)}$ in section **3.6** .

3.4 Sample Variance and Covariance

Sample covariance matrix $\boldsymbol{S}$ contains sample variances along the diagonal and covariances on the off diagonals. For data matrix $\boldsymbol{X}$, The sample covariance between columns $\boldsymbol{p}$ and $\boldsymbol{q}$ is given by

$$\boldsymbol{s_{pq}} = \frac{\mathbf{1}}{\boldsymbol{n-1}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} (\boldsymbol{p_i} - \boldsymbol{\bar{p}})(\boldsymbol{q_i} - \boldsymbol{\bar{q}}) \tag{4}$$

which is equal to the variance when $\boldsymbol{p} = \boldsymbol{q}$.

(George R Terrell, section 2.7)
Note that covariance and correlation matrices are related via the equation

$$\boldsymbol{S} = \boldsymbol{CRC} \tag{5}$$

(Ong $ Ransinghe, 2000) such that $\boldsymbol{C} = \boldsymbol{diag(\sigma)}$, a diagonal matrix containing standard deviations. If we consider a $\mathbf{3 \times 3}$ correlation matrix $\boldsymbol{R}$ then equation (5) can easily be seen below,

$$\begin{bmatrix} \boldsymbol{\sigma_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\sigma_2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\sigma_3} \end{bmatrix} \times \begin{bmatrix} \boldsymbol{r_{11}} & \boldsymbol{r_{12}} & \boldsymbol{r_{13}} \\ \boldsymbol{r_{21}} & \boldsymbol{r_{22}} & \boldsymbol{r_{23}} \\ \boldsymbol{r_{31}} & \boldsymbol{r_{32}} & \boldsymbol{r_{33}} \end{bmatrix} \times \begin{bmatrix} \boldsymbol{\sigma_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\sigma_2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\sigma_3} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\sigma_1^2} & \boldsymbol{\sigma_1 r_{12} \sigma_2} & \boldsymbol{\sigma_1 r_{13} \sigma_3} \\ \boldsymbol{\sigma_2 r_{21} \sigma_1} & \boldsymbol{\sigma_2^2} & \boldsymbol{\sigma_2 r_{23} \sigma_3} \\ \boldsymbol{\sigma_3 r_{31} \sigma_1} & \boldsymbol{\sigma_3 r_{32} \sigma_2} & \boldsymbol{\sigma_3^2} \end{bmatrix}.$$

as $\boldsymbol{r_{11}} = \boldsymbol{r_{22}} = \boldsymbol{r_{33}} = \mathbf{1}$. Recall (3) and (4), $\boldsymbol{r_{pq}}$ is simply equal to the covariance divided by the standard deviations i.e.

$$\boldsymbol{r_{pq}} = \frac{\boldsymbol{s_{pq}}}{\boldsymbol{\sigma_p \sigma_q}} \tag{6}$$

which can be rearranged into $\boldsymbol{s_{pq}} = \boldsymbol{\sigma_p r_{pq} \sigma_q}$. We can see the matrix formed in the example above is a covariance matrix which contains entries of $\boldsymbol{s_{pq}}$ with $\boldsymbol{\sigma_p^2} = \boldsymbol{s_{pp}}$, the variance.

3.5 Sylvester's Law of Inertia

The properties below along with the proofs can be found in Sylvester's original 1852 paper.

Matrix $\boldsymbol{A}$ is congruent to matrix $\boldsymbol{B}$ if they both satisfy $\boldsymbol{A} = \boldsymbol{DBD^T}$ such that $\boldsymbol{D}$ is non singular. The law then states that the number of positive, negative and 0 elements along the diagonal of $\boldsymbol{D}$, regardless of the $\boldsymbol{S}$ that is chosen and given that it's not singular, is always the same. A key consequence from this arises when performing the transformation $\boldsymbol{A} = \boldsymbol{DBD^T}$ regarding the corresponding eigenvalues of $\boldsymbol{A}$ and $\boldsymbol{B}$. If $\boldsymbol{A}$ and $\boldsymbol{B}$ are congruent, they have the same number of positive,negative and 0 eigenvalues.

3.6 Importance of Positive Semi-Definiteness

Matrix $\boldsymbol{A} \in \Re^{n \times n}$ is said to be positive semi-definite ($\boldsymbol{PSD}$) if

$$\forall \boldsymbol{x} \in \boldsymbol{R^n}, \quad \boldsymbol{x^T A x} \geq \boldsymbol{0}. \tag{7}$$

(positive definite if $\boldsymbol{x^T A x > 0}$). Another equivalent definition is that $\boldsymbol{A}$ has non-negative eigenvalues. Furthermore the entries $\boldsymbol{a_{ii}}$ $\forall \boldsymbol{i = 1..n}$ of $\boldsymbol{A}$ must be non-negative for the matrix to be positive semi-definite.

The final statement can easily be proven by contradiction. Suppose matrix $\boldsymbol{A}$ is positive semi-definite with some negative diagonal entry $\boldsymbol{a_{ii}}$. Let $\boldsymbol{x}$ correspond to a vector of all 0's apart from a 1 on element $\boldsymbol{x_{ii}}$. The resulting matrix $\boldsymbol{x^T A x}$ then becomes

$$\begin{bmatrix} \boldsymbol{0} & \boldsymbol{\dots} & \boldsymbol{x_{ii}} & \boldsymbol{\dots} & \boldsymbol{0} \end{bmatrix} \times \begin{bmatrix} \boldsymbol{0} & & & & \\ & \boldsymbol{\ddots} & & \boldsymbol{0} & \\ & & \boldsymbol{a_{ii}} & & \\ & \boldsymbol{0} & & \boldsymbol{\ddots} & \\ & & & & \boldsymbol{0} \end{bmatrix} \times \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{\vdots} \\ \boldsymbol{x_{ii}} \\ \boldsymbol{\vdots} \\ \boldsymbol{0} \end{bmatrix} = \boldsymbol{a_{ii} < 0}$$

which is a contradiction due to definition (7). This property ensures that the diagonal entries of a covariance matrix are non-negative, the diagonal entries contain the variances of each random variable so they need to be non negative by definition as negative variance is not possible. Also property (b) of (3) holds due to (5) preserving inertia, if $\boldsymbol{R}$ is not positive semi-definite then $\boldsymbol{S}$ will not be positive semi-definite which will also result in negative variance.

3.7 Nearest Correlation Matrix Problem

This section is inspired primarily on Higham's 2002 paper: Computing the Nearest Correlation Matrix. Higham found that to interpret this problem mathematically, conditions are set based upon the attributes of a correlation matrix and theoretical results found in convex analysis and Euclidean distance problems. The aim is to find a correlation matrix that is optimally close to the one we estimate for reliable stock analysis.
For some real symmetric matrix $\boldsymbol{Z} \in \boldsymbol{R^{nxn}}$, find matrix $\boldsymbol{B}$ such that it minimizes

$$||\boldsymbol{Z - B}|| \tag{8}$$

with respect to a weighted Frobenius Norm. The Norm condition is set because it minimizes the difference between the sum of all the entries of $\boldsymbol{Z}$ and $\boldsymbol{B}$.
The Frobenius Norm for $\boldsymbol{Z} \in \boldsymbol{R^{nxn}}$ is denoted as

$$||\boldsymbol{Z}||_{\boldsymbol{F}} = \sqrt{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \sum_{\boldsymbol{j=1}}^{\boldsymbol{n}} (\boldsymbol{z_{ij}})^{\boldsymbol{2}}} \tag{9}$$

Weighted versions of the Frobenius Norm will be used as they allow for us to express our confidence in each correlation coefficient approximation in $\boldsymbol{Z \in R^{nxn}}$. The Norm's that will be used in this paper will be The W Norm and the H Norm:

$$||\boldsymbol{Z}||_{\boldsymbol{W}} = ||\boldsymbol{W^{1/2} Z W^{1/2}}||_{\boldsymbol{F}} \tag{10}$$

$$||\boldsymbol{Z}||_{\boldsymbol{H}} = ||\boldsymbol{H} \circ \boldsymbol{Z}||_{\boldsymbol{F}} \tag{11}$$

for some symmetric positive definite matrix $\boldsymbol{W}$.

"$\boldsymbol{H} \circ \boldsymbol{Z}$" denotes the Hadamard product between a symmetric matrix $\boldsymbol{H}$ consisting of weights and correlation matrix $\boldsymbol{Z}$. We take this product via entry wise multiplication so $\boldsymbol{H} \circ \boldsymbol{Z} = \boldsymbol{h_{ij} z_{ij}}$.

This is effective because for each $\boldsymbol{z_{ij}}$, the corresponding weight $\boldsymbol{h_{ij}}$ can be assigned in accordance to the confidence of the correlation coefficient. If it is believed that $\boldsymbol{z_{ij}}$ is accurate, make $\boldsymbol{h_{ij}}$ to be large. This forces $\boldsymbol{z_{ij}}$ to be close to $\boldsymbol{b_{ij}}$ and thus more of an accurate approximation.

The key aspects that define a correlation matrix are positive semi-definiteness and unit diagonal; $\boldsymbol{x_{ij} = x_{ji}}$ $\forall \boldsymbol{i, j}$, the matrix consists of 1's along the diagonal entries $\boldsymbol{x_{ii}}$ and it has non-negative eigenvalues. Using these facts we can construct two sets which narrow down our problem. These will be:

$$\boldsymbol{S = \{Y = Y^T \in R^{nxn} : \lambda_n \geq 0\}} \tag{12}$$

such that $\boldsymbol{\lambda_n}$ denotes the $\boldsymbol{n^{th}}$ eigenvalue. (for n = 1,2..)

$$\boldsymbol{D = \{Y = Y^T \in R^{nxn} : y_{ii} = 1 | i = 1..n\}} \tag{13}$$

4 Convex Analysis

It is best to first imagine convex sets geometrically. If we were to take any two points within a set and connect them with a line segment, the set is said to be a convex set if the line segment lies completely within the set. The mathematical definition is that set $\boldsymbol{A}$ is convex if

$$\boldsymbol{\theta a_1 + (1-\theta)a_2 \in A \ \forall\, a_1, a_2 \in A, \forall \theta \in [0,1].} \tag{14}$$

(Rockafellar 1970 page 10).

Sets $\boldsymbol{S}$ and $\boldsymbol{D}$ are both closed convex sets.

Proofs **:** for set $\boldsymbol{D}$, $\boldsymbol{d_1, d_2 \in D}$, $\boldsymbol{\theta \in [0,1]}$ for arbitrary $\boldsymbol{d_1, d_2}$ and $\boldsymbol{\theta}$.
Matrices in set $\boldsymbol{D}$ are symmetric so $\boldsymbol{\beta_{ij} = \beta_{ji}}$ and $\boldsymbol{\alpha_{ij} = \alpha_{ji}}$. Then

$$\boldsymbol{d_1} = \begin{bmatrix} \mathbf{1} & \boldsymbol{\beta_{12}} & \dots & \boldsymbol{\beta_{1n}} \\ \boldsymbol{\beta_{12}} & \mathbf{1} & \dots & \boldsymbol{\beta_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\beta_{1n}} & \boldsymbol{\beta_{2n}} & \dots & \mathbf{1} \end{bmatrix}, \boldsymbol{d_2} = \begin{bmatrix} \mathbf{1} & \boldsymbol{\alpha_{12}} & \dots & \boldsymbol{\alpha_{1n}} \\ \boldsymbol{\alpha_{12}} & \mathbf{1} & \dots & \boldsymbol{\alpha_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\alpha_{1n}} & \boldsymbol{\alpha_{2n}} & \dots & \mathbf{1} \end{bmatrix}.$$

applying definition (14) to $\boldsymbol{d_1}$ and $\boldsymbol{d_2}$ we get :

$$\boldsymbol{\theta} \begin{bmatrix} \mathbf{1} & \boldsymbol{\beta_{12}} & \dots & \boldsymbol{\beta_{1n}} \\ \boldsymbol{\beta_{12}} & \mathbf{1} & \dots & \boldsymbol{\beta_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\beta_{1n}} & \boldsymbol{\beta_{2n}} & \dots & \mathbf{1} \end{bmatrix} + \boldsymbol{(1-\theta)} \begin{bmatrix} \mathbf{1} & \boldsymbol{\alpha_{12}} & \dots & \boldsymbol{\alpha_{1n}} \\ \boldsymbol{\alpha_{12}} & \mathbf{1} & \dots & \boldsymbol{\alpha_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\alpha_{1n}} & \boldsymbol{\alpha_{2n}} & \dots & \mathbf{1} \end{bmatrix} =$$

$$\begin{bmatrix} \boldsymbol{\theta + 1 - \theta} & \boldsymbol{\theta\beta_{12} + \alpha_{12} - \theta\alpha_{12}} & \dots & \boldsymbol{\theta\beta_{1n} + \alpha_{1n} - \theta\alpha_{1n}} \\ \boldsymbol{\theta\beta_{12} + \alpha_{12} - \theta\alpha_{12}} & \boldsymbol{\theta + 1 - \theta} & \dots & \boldsymbol{\theta\beta_{2n} + \alpha_{2n} - \theta\alpha_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\theta\beta_{1n} + \alpha_{1n} - \theta\alpha_{1n}} & \boldsymbol{\theta\beta_{2n} + \alpha_{2n} - \theta\alpha_{2n}} & \dots & \boldsymbol{\theta + 1 - \theta} \end{bmatrix} =$$

$$\begin{bmatrix} \mathbf{1} & \boldsymbol{\theta\beta_{12} + (1-\theta)\alpha_{12}} & \dots & \boldsymbol{\theta\beta_{1n} + (1-\theta)\alpha_{1n}} \\ \boldsymbol{\theta\beta_{12} + (1-\theta)\alpha_{12}} & \mathbf{1} & \dots & \boldsymbol{\theta\beta_{2n} + (1-\theta)\alpha_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\theta\beta_{1n} + (1-\theta)\alpha_{1n}} & \boldsymbol{\theta\beta_{2n} + (1-\theta)\alpha_{2n}} & \dots & \mathbf{1} \end{bmatrix} \in \boldsymbol{D}.$$

Matrix $\boldsymbol{A}$ is said to be positive semi-definite if $\forall \boldsymbol{x} \in \boldsymbol{R^n}$,$\boldsymbol{x^T A x \geq 0}$. If we take some arbitrary a,b $\in \boldsymbol{S}$ and Apply definition (14) to set $\boldsymbol{S}$ with $\boldsymbol{\theta \in [0,1]}$,

$$\boldsymbol{x^T(\theta a + (1-\theta)b)x = \theta x^T a x + (1-\theta)x^T b x \geq 0.} \tag{15}$$

Proving that the sets are closed is relatively simple but we need to highlight some basic topological facts. A set of points $\boldsymbol{X}$ in $\boldsymbol{\Re^n}$ is called open if for every point $\boldsymbol{x} \in \boldsymbol{X}$ and some sufficiently small positive number $\boldsymbol{\epsilon}$, each point $\boldsymbol{p}$ such that $\boldsymbol{d(x,p) < \epsilon}$ is contained in $\boldsymbol{X}$(Shiga et.al 2003). The neighbourhood $\boldsymbol{V}$ of some point $\boldsymbol{x} \subseteq \boldsymbol{X}$ is an open set of all the points that lie within some distance $\boldsymbol{r}$ of $\boldsymbol{x}(\boldsymbol{r > 0})$.

A subset $\boldsymbol{A}$ of metric space $\boldsymbol{M}$ is closed if the set $\boldsymbol{M - A}$ is open. Also, set $\boldsymbol{A}$ is a closed set if it contains all of its limit points. take some point $\boldsymbol{a} \subseteq \boldsymbol{A}$, $\boldsymbol{a}$ is a limit point of $\boldsymbol{A}$ if every neighbourhood of $\boldsymbol{a}$ contains at least some point other than $\boldsymbol{a}$. In simpler terms this means that $\boldsymbol{a}$ is a limit point if it can be approached from the inside of $\boldsymbol{A}$ and the outside of $\boldsymbol{A}$. These definitions and further properties of open and closed sets where found in Munkres(2000) and Grinberg(2017). The logic may seem circular with what we have defined but it is all we need for a proof by contradiction.

Consider set $\boldsymbol{D} \in \boldsymbol{\Re^{nxn}}$, for every $\boldsymbol{d} \in \boldsymbol{D}$, $\boldsymbol{d}$ will have unit diagonal. This also means that all of the limit points of $\boldsymbol{D}$, say $\boldsymbol{d_{Li}}$ will have a unit diagonal. Now suppose that set $\boldsymbol{D}$ is not closed, there then exists some limit point $\boldsymbol{d_{Li}}$ such that $\boldsymbol{d_{Li} \notin D}$. If $\boldsymbol{d_L \notin D}$ then $\boldsymbol{d_{Li} \in \bar{D}}$ such that $\boldsymbol{\Re^{n\times n} \backslash D = \bar{D}}$ i.e. the set of matrices that do not have unit diagonal. This is a contradiction as $\boldsymbol{d_{Li}}$ has a unit diagonal as stated prior so set $\boldsymbol{D}$ must be closed. A similar proof can be used for set $\boldsymbol{S} \in \boldsymbol{\Re^{n\times n}}$. For every $\boldsymbol{s} \in \boldsymbol{S}$ we take, matrix $\boldsymbol{s}$ will be positive semi-definite. This also means that all of the limit points $\boldsymbol{s_{Li}}$ of $\boldsymbol{S}$ will be positive semi-definite. Now suppose that set $\boldsymbol{S}$ is not closed then there exists some limit point $\boldsymbol{s_{Li}}$ such that $\boldsymbol{s_{Li} \notin S}$. It follows that $\boldsymbol{s_{Li} \in \bar{S}}$ such that $\boldsymbol{\Re^{n\times n} \backslash S = \bar{S}}$ i.e. the set of matrices that are negative definite. This is a contradiction as $\boldsymbol{s_{Li}}$ is positive definite as stated prior so set $\boldsymbol{S}$ must be closed.

In this section we will use the $\boldsymbol{W}$ norm and define the inner product that induces the $\boldsymbol{W}$ norm as :

$$\langle \boldsymbol{Z}, \boldsymbol{\gamma} \rangle = \boldsymbol{trace}(\boldsymbol{Z}^{\boldsymbol{T}} \boldsymbol{W} \boldsymbol{\gamma} \boldsymbol{W}). \tag{16}$$

Let $\boldsymbol{\Gamma} \in \boldsymbol{R}^{\boldsymbol{nxn}}$ be a convex set and $\boldsymbol{\gamma} \in \boldsymbol{\Gamma}$. The normal cone to the convex set $\boldsymbol{\Gamma}$ at $\boldsymbol{\gamma}$ is equal to $\boldsymbol{\epsilon} = \boldsymbol{\epsilon}^{\boldsymbol{T}}$ such that for all $\boldsymbol{\alpha} \in \boldsymbol{\Gamma}$, $\langle \boldsymbol{\alpha} - \boldsymbol{\gamma}, \boldsymbol{\epsilon} \rangle \leq \mathbf{0}$.

That is, $\boldsymbol{\epsilon}$ belongs to the normal cone of convex set $\boldsymbol{\Gamma}$ at $\boldsymbol{\gamma}$ if and only if when you take a random point $\boldsymbol{\alpha}$ in $\boldsymbol{\Gamma}$ and form the vector $\boldsymbol{\alpha} - \boldsymbol{\gamma}$, the inner product between $\boldsymbol{\epsilon}$ and $\boldsymbol{\alpha} - \boldsymbol{\gamma}$ is not positive i.e. its angle is larger than $\mathbf{90°}$. We will denote the normal cone to the convex set $\boldsymbol{\Gamma}$ at $\boldsymbol{\gamma}$ as $\boldsymbol{\Gamma^*(\gamma)}$. This can be formally defined as

$$\boldsymbol{\Gamma^*(\gamma) = \{\epsilon = \epsilon^T \in R^{nxn} : \langle \alpha - \gamma, \epsilon \rangle \leq 0 \forall \alpha \in \Gamma\}.} \tag{17}$$

We have proven sets $\boldsymbol{S}$ and $\boldsymbol{D}$ to be closed convex sets so we can approach the problem via optimization theory (Glunt et. al 1990). All correlation matrices have unit diagonal and are positive semi-definite so our solution $\boldsymbol{B}$ lies within the intersection between sets $\boldsymbol{S}$ and $\boldsymbol{D}$. Given that sets $\boldsymbol{S}$ and $\boldsymbol{D}$ are convex with a common interior point $\boldsymbol{B}$ then

$$\boldsymbol{(S \cap D)^*(B) = S^*(B) + D^*(B)} \tag{18}$$

(Rockafellar 1970 Corollary 23.8.1).

A key characterization to the problem has been established by Luenburger 1969:

$$\langle \boldsymbol{\alpha - B, Z - B} \rangle \leq \mathbf{0} \qquad \boldsymbol{\forall \alpha \in S \cap D} \tag{19}$$

which can be understood as $\boldsymbol{Z - B \in (S \cap D)^*(B)}$ (Higham 2002). Moreover the solution $\boldsymbol{B}$ is charactarized by

$$\boldsymbol{Z - B \in S^*(B) + D^*(B).} \tag{20}$$

Higham(2002).

To proceed we then need to find the individual normal cones of our convex sets at $\boldsymbol{Z}$.

$$\boldsymbol{D^*(Z) = \{W^{-1} diag(\theta_i) W^{-1} : arbitrary \;\; \theta_i\}} \tag{21}$$

$$\boldsymbol{S^*(Z) = \{Y = Y^T : \langle Y, Z \rangle = 0, Y \leq 0\}} \tag{22}$$

Higham(2002).

4.1 Projection

To begin our computations, we need to turn our symmetric matrix into a correlation matrix. This is done by individually projecting onto sets S and D with respect to our weighted Frobenius Norm. The projections force the symmetric matrix to have unit diagonal(Projection onto D) and then force it to become positive seimi-definite(Projection onto set S). $\boldsymbol{P_D(Z)}$ will denote the projection onto set $\boldsymbol{D}$ at $\boldsymbol{Z}$,

$$\boldsymbol{P_D(Z) = Z - W^{-1} diag(\theta_i) W^{-1}} \tag{23}$$

such that $\boldsymbol{[\theta_1 ...., \theta_n]^T = diag(A - I)(W^{-1} \circ W^{-1})^{-1}}$. For computational simplicity, when $\boldsymbol{W}$ is a diagonal matrix this is simply replacing all of the diagonal entries $\boldsymbol{Z_{ii}}$ by 1.

For the projection onto set S, let $\boldsymbol{Z} \in \boldsymbol{\Re^{n \times n}}$ have the eigendecomposition stated in (1) then

$$\boldsymbol{Z_+ = V} \begin{bmatrix} \boldsymbol{max(\lambda_1, 0)} & \boldsymbol{\dots} & \boldsymbol{0} \\ \boldsymbol{\vdots} & \boldsymbol{\ddots} & \boldsymbol{\vdots} \\ \boldsymbol{0} & & \boldsymbol{max(\lambda_n, 0)} \end{bmatrix} \boldsymbol{V^T},$$

such that $\boldsymbol{\lambda_n}$ denotes the $\boldsymbol{n^{th}}$ eigenvalue of $\boldsymbol{Z}$. The Projection onto set $\boldsymbol{S}$ is therefore

$$\boldsymbol{P_S(Z) = W^{-\frac{1}{2}}((W^{-\frac{1}{2}} Z W^{-\frac{1}{2}})_+) W^{-\frac{1}{2}}} \tag{24}$$

These projections along with proofs are found in Higham(2002).

4.2 Weighted Matrix W

$\boldsymbol{W}$ is chosen to force $\boldsymbol{B}$ to be as close to $\boldsymbol{Z}$ as possible. Note that the W norm is easier to work with than the H norm as $\boldsymbol{Z \rightarrow W^{\frac{1}{2}} Z W^{\frac{1}{2}}}$ is a congruence so it preserves inertia while $\boldsymbol{Z \rightarrow W \circ Z}$ only preserves symmetry(Higham 2002) as it is clear that if we multiply a matrix by a weighted matrix through a Hadamard product, the eigenvalues will also be affected.

Recall Sylvester's Law of Inertia (3.5). It is clear that the most natural choice of $\boldsymbol{W}$ is a diagonal matrix which will preserve the congruence mapping and make the problem computationally simpler. Throughout this paper the weighting $\boldsymbol{W = I}$ will be used such that W has the same dimensions as $\boldsymbol{Z}$ to preserve the congruence relation however it does not necessarily need to be equal to the identity.

5 Alternating Projections Algorithm (APA)

One way to tackle the problem is to iteratively project the approximated correlation matrix onto sets $\boldsymbol{S}$ and $\boldsymbol{D}$ until convergence. This can be done using the Alternating Projections Algorithm which was first developed by Von Neumann for Hilbert spaces, however Han (1988) found that when applied to convex sets, the algorithm converges to some theoretical intersection point which may not be optimally close while we require the nearest spot to our external estimated matrix $\boldsymbol{Z}$ with respect to a given Norm. The APA was intended for Hilbert spaces so we are unable to use it directly for $\boldsymbol{S}$ and $\boldsymbol{D}$, fortunately an alteration made by Dykstra(1983) which implements a small correction allows the use of the Alternating Projections Algorithm for closed convex sets. Dykstra proposed to add dummy variables to the traditional Alternating Projections Algorithm(Von Neumann) which iteratively track the residuals between each of projections, these gradually shrink until the algorithm converges.

5.1 Dykstra's Alternating Projections Algorithm

Given some starting point $\boldsymbol{Z} \in \boldsymbol{R^{nxn}}$, we repeat the operation

$$\boldsymbol{Z} \leftarrow \boldsymbol{P_D}(\boldsymbol{P_S}(\boldsymbol{Z})) \tag{25}$$

until it converges to a satisfied tolerance level.

By setting the initial conditions $\boldsymbol{dS_0} = \begin{bmatrix} \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \ddots & \\ \mathbf{0} & & \mathbf{0} \end{bmatrix} \in \Re^{n \times n}, \quad \boldsymbol{Y_0} = \boldsymbol{Z},$

the following algorithm finds the nearest correlation matrix with respect to the W Norm and Dykstra's correction:

for $\boldsymbol{k} = \mathbf{1, 2, ..}$
$\boldsymbol{R_k} = \boldsymbol{Y_{k-1}} - \boldsymbol{dS_{k-1}}$
$\boldsymbol{X_k} = \boldsymbol{P_S}(\boldsymbol{R_k}))$
$\boldsymbol{dS_k} = \boldsymbol{X_k} - \boldsymbol{R_k}$
$\boldsymbol{Y_k} = \boldsymbol{P_D}(\boldsymbol{X_k})$
and break once convergence is met i.e. a tolerance level is achieved. The algorithm converges when it satisfies

$$\frac{||\boldsymbol{Y_k} - \boldsymbol{X_k}||_\infty}{||\boldsymbol{Y_k}||_\infty} \leq \boldsymbol{\Omega} \tag{26}$$

(Higham 2002)

such that $\boldsymbol{\Omega}$ is a chosen level of tolerance to decide the point at which the algorithm converges. Both $\boldsymbol{X_k}$ and $\boldsymbol{Y_k}$ converge to our desired value $\boldsymbol{B}$ as $\boldsymbol{k} \rightarrow \boldsymbol{\infty}$(Boyle & Dykstra 1986) which minimizes $||\boldsymbol{Z} - \boldsymbol{B}||$. We will let $\boldsymbol{Y}$ be our output to remain consistent throughout. The DAPA code is found in the appendix on listing 29 and 30.

Computationally we iterate through a while loop until condition (26) is achieved. This condition minimizes the distance between the $\boldsymbol{Y}$ matrix and the $\boldsymbol{X}$ matrix in regards to a given tolerance level which will be set to $\boldsymbol{1e-07}$ and can be adjusted depending on how close you wish the matrix to be. There will also be a tolerance level set on the negative eigenvalues of $\boldsymbol{Z}$ to ensure $\boldsymbol{B}$ has strictly non negative eigenvalues. This is mainly due to computer imprecision which will be explained further in section 6 and the eigenvalue tolerance level will generally be set at $\boldsymbol{1e-07}$ throughout the paper. This means that the eigenvalues for $\boldsymbol{B}$ corresponding to the negative eigenvalues for $\boldsymbol{Z}$ will not be displayed as 0, rather they will be close enough to 0 that we can treat them as such.

If $\boldsymbol{Z = Z^T}$ has diagonal elements $\boldsymbol{z_{ii} \geq 1}$ with $\boldsymbol{t}$ non-positive eigenvalues and $\boldsymbol{W}$ is a diagonal matrix then $\boldsymbol{X_k}$ has at least $\boldsymbol{t}$ 0 eigenvalues for all k(or eigenvalues close enough to 0 that we can treat them as such). (Higham 2002 Corollary 3.5) This leads to his conclusion that for large $\boldsymbol{t}$, There is no need to compute the whole eigensystem when projecting onto set $\boldsymbol{S}$, it suffices to compute the largest $\boldsymbol{n-t \leq n}$ eigenvalues $\boldsymbol{\lambda_p}$ and corresponding orthonormal eigenvectors $\boldsymbol{q_p}$. This therefore significantly reduces the computational cost, making the algorithm converge at a faster rate.
Instead of computing $\boldsymbol{(W^{-\frac{1}{2}}ZW^{-\frac{1}{2}})_+}$ we can simply replace (24) with

$$\boldsymbol{P_S(Z) = W^{-\frac{1}{2}} \sum_{\lambda_i \geq 0} (\lambda_i q_i q_i^T) W^{-\frac{1}{2}}} \tag{27}$$

Which can be done efficiently by Lanczos' Iteration.

5.2 Lanczos Iteration

This method dates back to the Hungarian mathematician C.Lanczos in the 1950s. The idea is that finding the eigendecomposition for $\boldsymbol{A \in \Re^{n \times n}}$ where $\boldsymbol{n}$ is relatively large can be extremely costly, by using the $\boldsymbol{n-t \leq n}$ most useful eigenvalues and eigenvectors we can make the computation significantly more efficient. The most useful eigenvalues refer to the ones tending torwards an extremum point so for the nearest correlation matrix problem, setting $\boldsymbol{n-t}$ to be the non-negative eigenvalues will allow us to use Lanczos' Iteration to compute (27).

The Algorithm goes like this, for matrix $\boldsymbol{A}$ as shown above with corresponding eigenvalues $\boldsymbol{\lambda_1 \leq ... \leq \lambda_n}$ and eigenvectors $\boldsymbol{q_1, ..., q_n}$, we define the subspaces
$\boldsymbol{V_k(x) = span\{x, Ax, ..., A^{k-1}x\}}$. For some starting value $\boldsymbol{x \neq 0}$, we can construct an orthonormal basis $\boldsymbol{v_1 ..., v_k}$ of $\boldsymbol{V_k(x)}$ via the iterative algorithm:

for $\boldsymbol{k = 1, 2...}$

$\boldsymbol{v_0 = 0}$, $\boldsymbol{v_1 = \frac{x}{||x||_2}}$

$\boldsymbol{\alpha_k = \langle v_k, Av_k \rangle}$

$\boldsymbol{w_{k+1} = Av_k - \alpha_k v_k - \beta_k v_{k-1}}$

$\boldsymbol{\beta_{k-1} = ||w_{k-1}||_2}$

$\boldsymbol{v_{k-1} = \frac{w_{k-1}}{\beta_{k-1}}}$.

This results in a Tridiagonal matrix $\boldsymbol{T_k}$ that has the same eigenvalues and eigenvectors corresponding to the $\boldsymbol{n - t}$ eigenvalues of interest. That is, a columned orthanormal matrix $\boldsymbol{Q_k}$ =$\boldsymbol{[v_1 ... v_k]}$ is formed such that

$$\boldsymbol{Q_k^T A Q_K} = \begin{bmatrix} \boldsymbol{\alpha_1} & \boldsymbol{\beta_2} & & & & & \\ \boldsymbol{\beta_2} & \boldsymbol{\alpha_2} & \boldsymbol{\beta_3} & & & & \\ & \boldsymbol{\beta_3} & \boldsymbol{\alpha_3} & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \boldsymbol{\beta_{k-1}} & \\ & & & & \boldsymbol{\beta_{k-1}} & \boldsymbol{\alpha_{k-1}} & \boldsymbol{\beta_k} \\ & & & & & \boldsymbol{\beta_k} & \boldsymbol{\alpha_k} \end{bmatrix} = \boldsymbol{T_K}.$$

where $\boldsymbol{||.||_2}$ denotes the 2 norm and $\langle\rangle$ the standard inner product. Further Information Regarding Lanczos' Algorithm can be found in Deuflhard & Hohmann(2003) along with the origonal Lanczos(1950) paper. The DAPA code implementing Lanczos' Iteration can be found in the appendix section on listing 31 and 32.

Possible alternatives include using Householder Transformations which find the tridiagonal form by reflecting a matrix about the plane orthogonal to the corresponding unit normal. Details can be found in Householder(1958).

We will see how Corollary 3.5 in Higham(2002) works in practice with the example below.

Listing 1: Matrix

```
> A
     [,1] [,2] [,3] [,4]
[1,] 1.0 0.5 1.0 1.0
[2,] 0.5 1.0 0.5 0.5
[3,] 1.0 0.5 1.0 0.0
[4,] 1.0 0.5 0.0 1.0

> eigen(A)
eigen() decomposition
$values
[1] 2.8167126 1.0000000 0.6157299 -0.4324426
```

We will label Dykstra's Algorithm with with Lanczos' iteration as DAPA2. The example in listing 1 has one negative eigenvalue so from Higham's Corollary 3.5 our output should contain at least one 0 eigenvalue. Using $\boldsymbol{Z}_+$ as shown in section 6, the resulting nearest correlation matrix is as follows.

Listing 2: DAPA2 of Listing 1 Matrix

```
> DAPA2(A)
          [,1] [,2] [,3] [,4]
[1,] 1.0000000 0.5407239 0.7606601 0.7606601
[2,] 0.5407239 1.0000000 0.4715081 0.4715081
[3,] 0.7606601 0.4715081 1.0000000 0.1674507
[4,] 0.7606601 0.4715081 0.1674507 1.0000000
$Frobenius_Norm
[1] 0.5401647
$Rank
[1] 3
$iterations
[1] 25
$eigenvalues
[1] 2.616931e+00 8.325493e-01 5.505196e-01 2.616933e-10
```

As we can see the negative eigenvalue is now positive and computationally close enough to 0. We will explain more as to why they are greater than 0 rather then equal to 0 in section 6.

6 Computational Inaccuracy in R

Listing 3: R inaccuracy example

```
> 0.2+0.2
[1] 0.4
```

Solving simple inconspicuous arithmetic problems like the one above can result in cumbersome errors when dealing with R. Most coding programmes deal with the problem using floating point arithmetic rather than the standard strategy that is more commonly known. For all of the numbers with a denominator that is not a power of two, R will round these numbers to a certain level of accuracy(usually 53 digits in R).

Listing 4: R inaccuracy example

```
> options(digits=20)
> 0.2+0.2
[1] 0.40000000000000002
```

Floating point numbers have some base $\boldsymbol{\beta}$ which is always even and a precision parameter $\boldsymbol{p}$, this can then be represented as

$$\pm \boldsymbol{d_0.d_1d_2...d_{p-1}} \times \boldsymbol{\beta}^{-e} \tag{28}$$

such that $\pm \boldsymbol{d_0.d_1d_2...d_{p-1}}$ is classified as the $\boldsymbol{significand^2}$ (Goldberg 1991). Looking back to the example in listing 3, the sum computed was actually $\boldsymbol{(0.2+\epsilon_1)+(0.2+\epsilon_1)=(0.4+\epsilon_2)}$ such that $\boldsymbol{\epsilon_1}$ denotes the distance between 0.2 and the floating point representation of $\boldsymbol{0.2}$ (the same for $\boldsymbol{\epsilon_2}$ and 0.4).

Fortunately R does not output what is shown in listing 4. Rather, it simplifies the output displayed for the benefit of the user. When implementing theoretical concepts in a programme such as R, it is important to take into account computational inaccuracies due to machine imprecision. This is especially true for approximation problems like the nearest correlation matrix due to all of the algebra involved and the fact that we are working right on the edge of floating point numbers when dealing with very small eigenvalues.

Recall $\boldsymbol{Z_+}$ in section **4.1**, we are required to compare each eigenvalue with 0 which deals with the negative eigenvalues. Let $\boldsymbol{\delta V}$ represent an arbitrarily small number that has been added to $\boldsymbol{V}$, $\boldsymbol{\delta D}$ some small number added to $\boldsymbol{D}$ and let

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{max(\lambda_1, 0)} & \boldsymbol{\dots} & \boldsymbol{0} \\ \boldsymbol{\vdots} & \boldsymbol{\ddots} & \boldsymbol{\vdots} \\ \boldsymbol{0} & & \boldsymbol{max(\lambda_n, 0)} \end{bmatrix}.$$

The solution R gives us is not $\boldsymbol{Z_+ = VDV^T}$, rather it has been programmed to compute

$$\boldsymbol{\bar{Z}_+ = \bar{V}\bar{D}\bar{V}^T = (V + \delta V)(D + \delta D)(V + \delta V)^T.} \tag{29}$$

A problem arises computationally when comparing each eigenvalue with 0, R may replace the negative eigenvalues with a value that is arbitrarily close to 0 but not exactly 0. This can produce negative eigenvalues for our resulting nearest matrix and other computational issues so by replacing 0 with some number $\boldsymbol{\epsilon}$ that is arbitrarily small but $\boldsymbol{\epsilon > 0}$, the algorithm will no longer output negative eigenvalues. This however clearly impacts the nearness of our approximation but is a requirement when implementing the code.
$\boldsymbol{Z_+}$ then should become

$$\boldsymbol{Z_+ = V} \begin{bmatrix} \boldsymbol{max(\lambda_1, \epsilon)} & \boldsymbol{\dots} & \boldsymbol{0} \\ \boldsymbol{\vdots} & \boldsymbol{\ddots} & \boldsymbol{\vdots} \\ \boldsymbol{0} & & \boldsymbol{max(\lambda_n, \epsilon)} \end{bmatrix} \boldsymbol{V^T}.$$

6.1 Computationally Singular Matrices

Matrix Algebra is used in several areas from theoretical applications such as in (23)& (24) to financial reasons like computing the Global Minimum Variance Portfolio which is further discussed in section 8.5. Recall the well known definition of a matrix inverse, $\boldsymbol{A^{-1}}$ is the matrix inverse of $\boldsymbol{A}$ if

$$\boldsymbol{A * A^{-1} = I.} \tag{30}$$

When a matrix is singular it does not have a inverse. This is because to compute the inverse we need to divide by the determinant which is not possible with an eigenvalue of 0 due to (2). R may treat the determinant as computationally close enough to 0 that the inverse is unable to be computed so a different approach may be needed to calculate the matrix inverse on these occasions.

For example consider the covariance matrix in listing 5 of some log returns data. We will attempt to take the inverse of this matrix.

Listing 5: Covariance matrix of Log returns

```
          [,1] [,2] [,3] [,4]
[1,] 0.0008510490 0.0006636930 0.0009005647 0.0008058894
[2,] 0.0006636930 0.0008306829 0.0013251178 0.0007259356
[3,] 0.0009005647 0.0013251178 0.0027498929 0.0012597573
[4,] 0.0008058894 0.0007259356 0.0012597573 0.0008846773
[5,] 0.0003438972 0.0003242190 0.0001894144 0.0002577797
            [,5]
[1,] 0.0003438972
[2,] 0.0003242190
[3,] 0.0001894144
[4,] 0.0002577797
[5,] 0.0003535677
```

Listing 6: Eigenvalues of Listing 5 & Matrix Inverse

```
eigen(A)
eigen() decomposition
$values
[1] 4.628474e-03 7.731378e-04 2.355349e-04 3.272333e-05
2.161359e-19
matrix.inverse(C)
Error in solve.default(x) :
  system is computationally singular: reciprocal condition number
 = 2.17366e-17
```

The matrix is positive definite as all of the eigenvalues are strictly above 0 so the matrix is non-singular. The Inverse should therefore exist however as we can see an error appears, R treats this matrix computationally singular due to the extremely small eigenvalue of 2.16e-19.

One approach to deal with this problem is by taking the Moore-Penrose Inverse(Pseudoinverse) rather than the Matrix Inverse which guarantees a solution to exist.

An extensive overview of Moore's work in the 1900's can be found in Israel(2002).

Moore studied matrix inverses during the early 1900's and proposed a reciprocal for singular matrices. The matrix can be found after computing the Singular Value Decomposition(SVD) of a given matrix.

For $\boldsymbol{A}$, the singular value decomposition of $\boldsymbol{A}$ is

$$\boldsymbol{A} = \boldsymbol{BDC^T} \tag{31}$$

such that $\boldsymbol{B}$ and $\boldsymbol{C}$ are unitary orthogonal matrices and and $\boldsymbol{D}$ is a diagonal matrix with non negative real entries.

The Pseudo-inverse $\boldsymbol{A^+}$ of $\boldsymbol{A}$ can then be represented as

$$\boldsymbol{A^+} = \boldsymbol{CD'B^T} \tag{32}$$

such that $\boldsymbol{D'}$ is obtained by replacing the diagonal entries by their reciprocals.

These representations along with further properties can be found in Golub & Kahan(1965). Matrix $\boldsymbol{A's}$ Singular Value Decomposition can easily be computed with $\boldsymbol{R's}$ base function 'svd'

Listing 7: SVD

```
svd(A)
$d
[1] 4.628474e-03 7.731378e-04 2.355349e-04 3.272333e-05 2.215661e-19

$u
          [,1] [,2] [,3] [,4] [,5]
[1,] -0.3435143 0.5902805 -0.35876597 0.3977883 0.4966068
[2,] -0.4040307 0.1191170 0.47788536 0.5628563 -0.5266770
[3,] -0.7344939 -0.5701706 0.06049547 -0.1476342 0.3316152
[4,] -0.4073841 0.2742009 -0.46476977 -0.4793827 -0.5594939
[5,] -0.1153866 0.4869285 0.65057046 -0.5228323 0.2301977

$v
          [,1] [,2] [,3] [,4] [,5]
[1,] -0.3435143 0.5902805 -0.35876597 0.3977883 -0.4966068
[2,] -0.4040307 0.1191170 0.47788536 0.5628563 0.5266770
[3,] -0.7344939 -0.5701706 0.06049547 -0.1476342 -0.3316152
[4,] -0.4073841 0.2742009 -0.46476977 -0.4793827 0.5594939
[5,] -0.1153866 0.4869285 0.65057046 -0.5228323 -0.2301977
```

By taking the Singular Value Decomposition output, we can use the function below to directly find the inverse of a computationally singular matrix.

Listing 8: Pseudoinverse Code

```
PseudInv<-function(B){
  eig<-eigen(B)
  val<-eig$values
  svdb<-svd(B)
  U<-svdb$u
  D<-diag(1/svdb$d)
  D[,which(val<1e-05)]<-1e-05
  v<-svdb$v
  a<-v%*%D
  a%*%t(U)
}
```

The inverse of listing 5 now becomes

Listing 9: Covariance matrix of Log returns

```
> pseudoinverse(A)
         [,1] [,2] [,3] [,4] [,5]
[1,] 5858.194 6235.159 -2267.609 -4879.907 -6966.225
[2,] 6235.159 10704.607 -2440.366 -9110.783 -7587.897
[3,] -2267.609 -2440.366 1218.647 1905.836 2185.112
[4,] -4879.907 -9110.783 1905.836 8072.962 6558.379
[5,] -6966.225 -7587.897 2185.112 6558.379 10459.966
```

A practical example where this may occur in finance is (38), the covariance matrix needs to be non-singular for the Global Minimum Variance Portfolio to be found. If the covariance matrix is computationally singular then change the formula of the Minimum Variance Portfolio on listing 40 in the appendix by replacing the matrix inverse for the Pseudoinverse.

7 Dealing with Missing Data

All of the following topics and definitions regarding missing data can be found in Ruben (1976).

In the financial application to this problem, there may be days where some stock prices are not available to be recorded. It is important to know why the stock market data has not been recorded, is there an underlying pattern within the missing values or are they completely at random? Data can be missing for a variety of reasons from bank holidays to regulatory conditions of a given market. Knowing the reason behind the missing data points can direct us to appropriate methods of Imputation or approximation techniques. Imputation is the statistical process of substituting approximated values in for missing data. The observed data is said to be observed at random if for each of the missing values, the conditional probability of the observed pattern given the missing data and observed data is the same for all of the possible values of the observed data. This means each observed value has the same chance of being observed given a pattern of missing data. An extensive overview of strategies that can be used to tackle missing data can be found in C.Acock (2005).

7.1 MAR & MCAR

Incomplete data is said to be Missing Completely at Random(MCAR) if there is no relationship between the pattern at which the missing data is found, the observed values and the unobserved values. This would imply that the cause of the missing data is unrelated to the data that is missing and the probability that one data point is missing is the same across all data points.

Data is Missing at Random(MAR) when there is a clear underlying relationship behind the frequency of the missing values and the observed data but not the missing data itself. Imagine you are a statistician looking to find out how much money each household in London makes so you send out questionnaires to gather the data. For all of the questionnaires that end up lost in the mail, the missing data is said to be missing completely at random and the observed data is observed at random. Now after receiving the questionnaires, you notice that some households did not choose to tell you their income. You also notice that a lot of young people below the age of 25 did not tell you their income. If the probability of completion can be inferred from the participants age and not the level of income, then the missing data is said to be Missing at Random.

In practice, it's difficult to tell if the data is missing at random or missing completely at random as you need to know the underlying probability that the data goes missing. Tests can be conducted to help you infer what type of missing data you are dealing with, this can be found in Rodrick(1988) for example. Throughout this paper we will not focus on how prove if the data is MAR or MCAR, rather we will highlight a few strategies that can be used to deal with the missing data at question.

7.2 Classical Methods to Deal with Missing Data

The methods we will focus on to deal with missing values will be pairwise deletion, listwise deletion and mean Imputation. Listwise deletion removes the rows corresponding to the missing values and then proceeds with the given statistical analysis.C.Acock(2005) states that Listwise deletion can lead to bias if the data is not MCAR. Furthermore it will lead to a reduction in the statistical power of the findings if the sample size is not large, there is also a risk of increasing the type 2 error rate. This can be ineffective for finance because the matrices we are dealing with are of high rank so large amounts of data will also be unaccounted for however if one can prove that the data is missing completely at random, it could be a good approach. Pairwise deletion is a method that drops the elements that are unaccounted for and then uses the other elements that are accounted for on that day for subsequent statistical analysis. Mean imputation is simply replacing the missing values with the respective column means. This can clearly be problematic as if the data follows a normal distribution then most of the observations will be close to the mean therefore this may result in a bad approximation when the data has high levels of skewness. C.Acock(2005) further stated that the method is problematic when a large amount of data is missing.

7.3 Sample Covariance with Missing Values Example

This trivial example highlights the paiwise deletion method well. Take a real $\boldsymbol{m \times n}$ matrix $\boldsymbol{G}$ with NA representing missing values and $\boldsymbol{n_{,1}}$ representing the first column. We are able to individually extract the columns that contain missing values and impute the data using Pairwise deletion.

$$\boldsymbol{G} = \begin{bmatrix} \mathbf{4} & \mathbf{4} & \mathbf{6} & \mathbf{8} & \mathbf{3} \\ \mathbf{3} & \mathbf{1} & \mathbf{5} & \boldsymbol{NA} & \mathbf{3} \\ \mathbf{3} & \boldsymbol{NA} & \mathbf{7} & \mathbf{6} & \mathbf{4} \\ \mathbf{7} & \mathbf{2} & \mathbf{3} & \mathbf{6} & \mathbf{8} \\ \mathbf{2} & \mathbf{2} & \mathbf{9} & \mathbf{12} & \mathbf{13} \\ \mathbf{5} & \mathbf{4} & \mathbf{3} & \mathbf{9} & \mathbf{4} \end{bmatrix} \qquad \boldsymbol{n_{,2}} = \begin{bmatrix} \mathbf{4} \\ \mathbf{1} \\ \boldsymbol{NA} \\ \mathbf{2} \\ \mathbf{2} \\ \mathbf{4} \end{bmatrix} \qquad \boldsymbol{n_{,4}} = \begin{bmatrix} \mathbf{8} \\ \boldsymbol{NA} \\ \mathbf{6} \\ \mathbf{6} \\ \mathbf{12} \\ \mathbf{9} \end{bmatrix}$$

To compute the variance between random variable 2 and 4, only the data points that are available on both days will be used. recall (4), the resulting covariance between random variables 2 and 4 will be

$$\bar{\boldsymbol{s}}_{\mathbf{2,4}} = \tfrac{1}{4-1} \begin{bmatrix} \mathbf{8} - \bar{\boldsymbol{n}}_{,4} & \mathbf{6} - \bar{\boldsymbol{n}}_{,4} & \mathbf{12} - \bar{\boldsymbol{n}}_{,4} & \mathbf{9} - \bar{\boldsymbol{n}}_{,4} \end{bmatrix} \begin{bmatrix} \mathbf{4} - \bar{\boldsymbol{n}}_{,\mathbf{2}} \\ \mathbf{2} - \bar{\boldsymbol{n}}_{,\mathbf{2}} \\ \mathbf{2} - \bar{\boldsymbol{n}}_{,\mathbf{2}} \\ \mathbf{4} - \bar{\boldsymbol{n}}_{,\mathbf{2}} \end{bmatrix}$$

such that $\bar{\boldsymbol{n}}_{,\mathbf{2}} = \frac{1}{4}(\mathbf{4} + \mathbf{2} + \mathbf{2} + \mathbf{4})$, $\bar{\boldsymbol{n}}_{,4} = \frac{1}{4}(\mathbf{8} + \mathbf{6} + \mathbf{12} + \mathbf{9})$ and $\bar{\boldsymbol{s}}_{\mathbf{2,4}}$ denotes the approximated sample covariance between random variables 2 and 4.

The sample correlation matrix can be computed in a similar way as above using definition (3). The approximated sample correlation and covariance matrices $\bar{\boldsymbol{S}}$ and $\bar{\boldsymbol{R}}$ are respectively filled with these approximated variance, covariance and correlation coefficients. None of the methods highlighted above guarantee the matrix to be positive semi-definite so subsequent statistical analysis is required prior to the financial application(DAPA for example). Mean Imputation would simply involve substituting $\bar{\boldsymbol{n}}_{,\mathbf{2}}$ and $\bar{\boldsymbol{n}}_{,\mathbf{4}}$ in for the respective missing values in column 2 and column 4. These are simple classical approaches to deal with missing data, other more modern approaches include maximum likelihood to which estimates the missing data using the information that is available and multiple imputation which generates several data sets to be imputed and then pools together each of the results and compares the standard errors between estimates. These methods along with others can be found in C.Acock(2005).

8 Finance

8.1 Returns

Sometimes it is beneficial in finance to deal with asset returns rather than the underlying asset price. We can denote the price at time t as $\boldsymbol{P_t}$ and the simple return over time period t as $\boldsymbol{R_t}$, simple returns are then denoted as

$$\boldsymbol{R_t} = \frac{\boldsymbol{P_t} - \boldsymbol{P_{t-1}}}{\boldsymbol{P_{t-1}}}. \tag{33}$$

The Log return over time period $\boldsymbol{t}$ highlights the logarithmic price changes over the period that the asset is held. This is seen as

$$\boldsymbol{r_t} = \boldsymbol{log(P_t) - log(P_t - 1).} \tag{34}$$

Technical analysis along with empirical data comparing the effectiveness of simple returns to Log returns can be found in Hudson & Gregoriou (2010). Set a portfolio to have $\boldsymbol{n}$ assets, denoted $\boldsymbol{A_i}$ for $\boldsymbol{i = 1..n}$, with expected return of $\boldsymbol{R_i}$ for each asset and the corresponding weights $\boldsymbol{W_i}$. The expected return of the portfolio is simply the sum of the expected asset returns multiplied by the corresponding weights where the weights are the proportion of the investment that is held in one asset therefore the portfolio expected return is

$$\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{R_i W_i} \tag{35}$$

such that $\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \boldsymbol{W_i} = \mathbf{1}$.

8.2 Short and Long

The two main positions an investor can take in relation to investing is going short or going long(and a multitude of combinations between the two are possible). Going long is buying a stock with the goal of selling it at a later date once the price has gone up to make subsequent profit. Going short is a bit more complicated, if an investor believes that a stock will eventually drop in price then they are able to temporarily borrow a stock from a broker and sell it at todays date with the obligation to give it back to the broker at some point. Once the stock has dropped in price, the investor can buy it from the open market and give it back to the broker keeping the difference made between the initial sale and the final sale as profit. Clearly if the price of the stock increases, the investor is obliged to purchase the stock at a higher price than he sold it for which results in a loss. Portfolios tend to consist primarily of long trades but can also be comprised of short positions if there is a combination of portfolio weights with a greater return at a given level of risk when short positions are taken into account.

8.3 Modern Portfolio Theory

A financial concept that tends to be seen as trivial is the risk return trade off. Suppose you had to choose between two assets with the same level of risk but asset A had higher returns than asset B, logically A would be preferable. A portfolio is the span of assets an investor holds. Optimal Portfolio Theory became highly popular in finance due to Harry Markowtiz' research in 1952. For Modern portfolio theory to be assumed accurate, several assumptions need to be highlighted:

- People invest solely based on Mean and Variance
- Investors are rational and invest with the aim of maximizing expected return and minimizing risk
- Transaction costs are not taken into account while constructing portfolios

Markowitz suggested that portfolios can be chosen based on levels of risk and return. This theory is an important analytical tool to help an investor either find a portfolio which maximizes returns at a given risk level or minimizes risk at a given return level. Furthermore, every possible portfolio of a given number of stocks can be plotted on a graph against the level of risk and return, this plot reveals the set of portfolios which are deemed optimal to an investor and the portfolio with the lowest overall risk can even be found.

8.4 Portfolio Theory with Matrix Notation

When dealing with larger portfolios, it is often beneficial to represent the data you are working with in vector and matrix form as it helps with practical computation. Suppose there are N assets in a given market, Let $\boldsymbol{R}$ be a vector of returns for assets $\boldsymbol{1, 2..., N}$ then the expected returns of these assets can then be denoted as

$$\boldsymbol{E(R) = E}\left(\begin{bmatrix}\boldsymbol{R_1}\\ \vdots \\ \boldsymbol{R_N}\end{bmatrix}\right) = \left(\begin{bmatrix}\boldsymbol{E(R_1)}\\ \vdots \\ \boldsymbol{E(R_N)}\end{bmatrix}\right) = \left(\begin{bmatrix}\boldsymbol{\mu_1}\\ \vdots \\ \boldsymbol{\mu_N}\end{bmatrix}\right) = \boldsymbol{\mu} \tag{36}$$

such that $\boldsymbol{\mu_N}$ is the mean return for the $\boldsymbol{N^{th}}$ asset. The weights assigned to each stock can be seen as a vector $\boldsymbol{W}$ such that $\boldsymbol{WV_1 = 1}$ where $\boldsymbol{V_1}$ is a $\boldsymbol{N \times 1}$ vector of 1's. the variance covariance matrix for the returns is simply $\sum = \boldsymbol{cov(R)}$ where $\sum \in \boldsymbol{\Re^{N \times N}}$.

8.5 Global minimum Variance portfolio

Investors want to invest in stocks with price movements they can more accurately predict. If a stock has high levels of variance, this means that it varies greatly from the expected value it was predicted to take which makes the stock a relatively risky investment. The Global Minimum Variance Portfolio can be interpreted as the Portfolio which carries the least risk attached to it. This can simply be interpreted as a minimization problem. The investor wants to find $\boldsymbol{W}$ such that it minimizes the variance of the portfolio.

The problem is to solve

$$\boldsymbol{W_{mv} = argmin(W^T \sum W)} \tag{37}$$

with respect to $\boldsymbol{W^T V_1 = 1}$. The Weights for the Minimum variance portfolio become

$$\boldsymbol{W_{mv}} = \frac{\boldsymbol{\sum^{-1} V_1}}{\boldsymbol{V_1^T \sum^{-1} V_1}} \tag{38}$$

The code for the solution above is found in the appendix on listing 40. The expected portfolio return can be found by

$$\boldsymbol{\mu_{mv}} = \frac{\boldsymbol{\mu^T \sum^{-1} V_1}}{\boldsymbol{V_1^T \sum^{-1} V_1}}. \tag{39}$$

These solutions are subject to short positions being allowed in the portfolio. Extensive analytical proofs can be found in Mertons 1972 paper.

9 Nearest Covariance Matrix Problem

We have established the methodology behind finding the closest correlation matrix to one which may not positive semi-definite. Using the nearest correlation matrix along with relation (5), we will explore strategies to find an approximation for the corresponding sample covariance matrix which is used more often in practical financial analysis. Let the nearest correlation matrix to some non positive semi-definite matrix $\boldsymbol{R}$ be $\bar{\boldsymbol{R}}$, if we are able to find an approximation $\bar{\boldsymbol{C}}$ for the diagonal matrix of standard deviations then we can theoretically form

$$\bar{\boldsymbol{V}} = \bar{\boldsymbol{C}}\bar{\boldsymbol{R}}\bar{\boldsymbol{C}} \tag{40}$$

which is a possible approximation for our covariance matrix $\boldsymbol{V}$. This clearly depends on the accuracy of approximations for $\boldsymbol{\sigma}$ and how near our $\bar{\boldsymbol{R}}$ is. Recall the law of Inertia in 3.5, now as $\bar{\boldsymbol{C}}$ is a diagonal matrix then $\bar{\boldsymbol{C}} = \bar{\boldsymbol{C}}^{\boldsymbol{T}}$ which means our transformation $\bar{\boldsymbol{V}} = \bar{\boldsymbol{C}}\bar{\boldsymbol{R}}\bar{\boldsymbol{C}}$ preserves Inertia. The resulting covariance matrix $\bar{\boldsymbol{V}}$ will thus have strictly non negative eigenvalues after performing Dykstras Alternating Projections Algorithm on $\bar{\boldsymbol{R}}$.

We have explained that positive semi-definiteness is a requirement for covariance matrices due to the possibility of negative variance. In Finance, covariance has many applications such as (38) and (39) so if a covariance matrix is estimated, banks and investors would want to use the nearest covariance matrix to a given estimated matrix which may not necessarily be positive semi-definite for subsequent calculations. Recall the nearest correlation matrix problem (8), for some symmetric $\boldsymbol{C} \in \mathfrak{R}^{\boldsymbol{n}\times\boldsymbol{n}}$ we can then aim to find $\boldsymbol{V}$ that will minimize

$$||\boldsymbol{C} - \boldsymbol{V}|| \tag{41}$$

such that $\boldsymbol{V}$ is a covariance matrix in weighted Frobenius Norm. This is simply the nearest positive semi-definite matrix problem in theory.

An interesting problem may arise when calculating the Global Minimum Variance Portfolio if we look at its definition. An underlying property is assumed that is $\sum$ must be a non-singular matrix. Recall (38), the matrix inverse of $\sum$ is required to be solvable for the subsequent minimum variance portfolio to be found. As we have stated prior, a covariance matrix must be positive semi-definite and the set of positive semi-definite matrices contain the set of positive definite matrices. If we were to take some arbitrary symmetric singular matrix $\boldsymbol{A}$ Which is strictly positive semi-definite and not positive definite i.e. it has an eigenvalue of 0, (38) would be impossible to solve. Recall property (2), the determinant is equal to the product of the eigenvalues so if matrix $\boldsymbol{A}$ has an eigenvalue of $\mathbf{0}$, the solution to the global minimum variance portfolio weights in (38) would be invalid. Now as all positive definite matrices are non-singular by definition due to strictly positive eigenvalues, it may be more beneficial to find the nearest positive definite matrix instead.

If we take a look at problem (8) again, we can always find some symmetric matrix $\boldsymbol{B}$ that is the closest in weighted Frobenius Norm with unit diagonal by finding its eigendecomposition and turning its negative eigenvalues non-negative as explained throughout this paper. Consider an $\boldsymbol{n} \times \boldsymbol{n}$ matrix $\boldsymbol{A}$ with eigenvalues $\boldsymbol{\lambda_1 ... \lambda_n}$ and at least one eigenvalue is negative. As explained in 4.1, by replacing the negative eigenvalue with 0 when solving for $\boldsymbol{Z_+}$, this forces the resulting nearest matrix to have non negative eigenvalues. However if we want to find the nearest positive definite matrix $\boldsymbol{V}$ using the same methodology, for every $\boldsymbol{\epsilon > 0}$ we propose to replace the non-positive eigenvalue of $\boldsymbol{A}$ with, there will always be some $\boldsymbol{V'}$ with a corresponding eigenvalue of $\boldsymbol{\epsilon - \delta > 0}$ $\boldsymbol{(0 \leq \delta < \epsilon)}$ that will be nearer with respect to the Frobenius Norm. Therefore the nearest positive definite matrix is impossible to solve in theory.

By replacing 0 with some arbitrarily small $\boldsymbol{\epsilon}$ we avoid computational error. Moreover this forces the matrix to be positive definite but does not guarentee it to be the nearest in Frobenius Norm. We will explore the results of finding an approximated covariance matrix by projecting it onto the positive semi-definite set in section 10.1.

10 Tests

In this section we will test the effectiveness of Dykstra's Alternating Projections Algorithm on several data sets including a financial data set taken from 2019 and several randomly generated sample correlation matrices with numbers taken from a random normal distribution. Furthermore We will compare the effectiveness of Lanczos' Iteration in reducing computational time at different dimensions. The number of missing values will be kept proportional to the dimensions of the matrix.

$$Z = \begin{array}{c} \\ 02/01 \\ 03/01 \\ 04/01 \\ 07/01 \\ 08/01 \end{array} \begin{array}{c} \begin{array}{ccccc} \textit{FB} & \textit{AMZN} & \textit{NVDA} & \textit{MSFT} & \textit{PEPSI} \end{array} \\ \left[\begin{array}{ccccc} \mathbf{135.68} & \mathbf{1539.13} & \mathbf{136.22} & \mathbf{101.12} & \mathbf{109.28} \\ \mathbf{131.74} & \mathbf{1500.28} & \mathbf{127.99} & \mathbf{97.40} & \mathbf{108.26} \\ \mathbf{137.95} & \mathbf{1575.39} & \mathbf{136.19} & \boldsymbol{NA} & \mathbf{110.48} \\ \mathbf{138.05} & \mathbf{1629.51} & \mathbf{143.40} & \mathbf{102.06} & \mathbf{109.53} \\ \mathbf{142.53} & \boldsymbol{NA} & \mathbf{139.83} & \mathbf{102.80} & \mathbf{110.58} \end{array}\right] \end{array}$$

$$R(Z) = \begin{array}{c} \\ \textit{FB} \\ \textit{AMZN} \\ \textit{NVDA} \\ \textit{MSFT} \\ \textit{PEPSI} \end{array} \begin{array}{c} \begin{array}{ccccc} \textit{FB} & \textit{AMZN} & \textit{NVDA} & \textit{MSFT} & \textit{PEPSI} \end{array} \\ \left[\begin{array}{ccccc} \mathbf{1} & \mathbf{0.8914397} & \mathbf{0.7809127} & \mathbf{0.9195515} & \mathbf{0.9092267} \\ \mathbf{0.8914397} & \mathbf{1} & \mathbf{0.9543933} & \mathbf{0.8483285} & \mathbf{0.6400660} \\ \mathbf{0.7809127} & \mathbf{0.9543933} & \mathbf{1} & \mathbf{0.9297665} & \mathbf{0.6402122} \\ \mathbf{0.9195515} & \mathbf{0.8483285} & \mathbf{0.9297665} & \mathbf{1} & \mathbf{0.9281296} \\ \mathbf{0.9092267} & \mathbf{0.6400660} & \mathbf{0.6402122} & \mathbf{0.9281296} & \mathbf{1} \end{array}\right] \end{array}$$

The first example above is an approximated $\mathbf{5 \times 5}$ correlation matrix $\boldsymbol{R(Z)}$ taken from the closing price data between the $\mathbf{2^{nd}}$ and the $\mathbf{8^{th}}$ of January 2019. The sample correlation matrix was estimated using formula (3) and the methodology in section 7.3. As stated previously, $\boldsymbol{R(Z)}$ has turned out to not be positive semi-definite.

Listing 10: Eigenvalues of R(Z)

```
eigen() decomposition
$values
[1] 3.935163717 1.334165448 0.196367411 0.008249804 -0.473946381
```

The second example is taken from a $\mathbf{50 \times 50}$ randomly generated matrix that had 20 random values replaced with $\boldsymbol{NA}$ and then a sample correlation matrix is computed using (3) and the technique described in section 7.3. This matrix had 22 negative eigenvalues ranging from $\mathbf{-2.539e-17}$ to $\mathbf{-2.069e-02}$ and positive eigenvalues ranging from $\mathbf{8.299e-18}$ to $\mathbf{1.659e+01}$.

Example 3 is a $\mathbf{500 \times 500}$ matrix with 200 missing values. The matrix has 251 negative eigenvalues ranging from $\mathbf{-2.999e-03}$ to $\mathbf{-1.379e-23}$ and positive eigenvalues ranging from $\mathbf{3.293e-24}$ to $\mathbf{2.748e+02}$. A convergence tolerance level of $\mathbf{1e-07}$ was set for example 1 and 2 and $\mathbf{1e-04}$ for example 3.

Dimensions	Neg Eig	Time(no lanc)	Time(Lanc)	Iterations	F Norm	missing values
5	1	0.2158	0.2157	27	0.58135	2
50	22	6.581	5.754	84	0.04267	20
500	251	24.478	23.849	1	0.04559	200

We can see that using Lanczos' Algorithm reduces the computational time taken in finding the nearest correlation matrix for each of our experiments. The smallest difference of 0.0001 was found to be for our $\mathbf{5 \times 5}$ matrix. This makes sense as Lanczos' Algorithm only had to deal with one negative eigenvalue of -0.47 as seen in listing 10. The largest reduction in time was found to be when computing example 2. The example converged in 84 iterations which could suggest that for instances where a significant amount of iterations are needed, Lanczos' Algorithm may be more effective. A further test found that after increasing the convergence tolerance of example 3 to $\mathbf{1e-06}$, the time difference between not using lanczos' Algorithm and using lanczos' Algorithm increased from 0.827 seconds to seconds to 1.25 seconds. The improvement shown may seem insignificant but would presumably increase when dealing with correlation matrices that have columns of data in the thousands however this would still need to be tested. The results can also be found in the appendix from listing 34 to 39.

10.1 The Financial Data Set

This section outlines the steps taken from dealing with an inconsistent data set up until a financial application. We will first detail the process and then comment on the appropriateness of the methods used. To deal with missing values we need to first know the underlying reason as to why the values went missing. For this example suppose that an investor is looking at the historical prices of 5 different stocks in 2019 on some website online. For some reason there is a technical error when displaying some of the stock prices, the probability that a stock price is not displayed is known to be $\boldsymbol{\theta}$. This means that the missing values are missing Completely at Random. There is no systematic relationship between the pattern of the missing data and the observed values and the missing data does not effect the observed data as it was taken from the past. This assumption is unrealistic in practice because MCAR is difficult to prove and the investor would just look elsewhere for the stock prices.

This example however shows us how we can use Dykstra's Alternating Projections Algorithm in practice. To deal with the missing data we will be using pairwise deletion to find an approximated correlation matrix which ends up not being positive semi-definite. Following this we will use Dykstra's Alternating Projections Algorithm to find the closest correlation matrix which fixes the issue of negative eigenvalues. Furthermore we will convert the approximated sample correlation matrix to an approximated sample covariance matrix using the statistical relation highlighted in section 3 and listwise deletion to find the approximated sample standard deviations. The financial Application we will be focusing on is finding the Global minimum Variance Portfolio using the matrix solution (38) and interpreting the findings. We will also explore a possible method for directly finding the closest covariance matrix by projection and comparing the results.

Given an $\boldsymbol{m \times n}$ matrix $\boldsymbol{Q}$ consisting of daily closing prices and recalling (34), the corresponding matrix of log returns $\boldsymbol{LQ}$ is found using

$$\boldsymbol{lq_{m,n} = log(q_{m,n}) - log(q_{m-1,n}).} \tag{42}$$

A problem clearly arises for when $\boldsymbol{q_{m,n}}$ or $\boldsymbol{q_{m-1,n}}$ is missing. For every missing element $\boldsymbol{q_{m,n}}$, there will be 2 missing elements $\boldsymbol{lq_{m,n}}$ and $\boldsymbol{lq_{m-1,n}}$ in $\boldsymbol{LQ}$(1 missing element for entries on the first and last rows).

The dataset that will be used in this section will be the same data set used for example 1 in the beginning of section 10 but an extra day of closing prices has been added. This is to highlight the fact that sample covariance between log returns is impossible to compute using pairwise deletion if there is only one observation present on both days.

Recall $\boldsymbol{Z}$ in the beginning of section 10, the log returns of Amazon and Microsoft are then

$$\boldsymbol{LA} = \begin{bmatrix} \mathbf{-0.02} \\ \mathbf{0.04} \\ \mathbf{0.03} \\ \boldsymbol{abs} \end{bmatrix} \qquad \boldsymbol{LM} = \begin{bmatrix} \mathbf{-0.037} \\ \boldsymbol{abs} \\ \boldsymbol{abs} \\ \mathbf{0.007} \end{bmatrix}$$

such that $\boldsymbol{abs}$ denotes a log return where $\boldsymbol{lq_{m,n}}$ or $\boldsymbol{lq_{m-1,n}}$ is not observed. Recall (4), it is clear that it is not possible to use pairwise deletion for estimating the sample covariance between MSFT and AMZN as $\boldsymbol{N-1=0}$ in this instance. This will not occur often in finance because the vectors of Log returns would have dimensions in the thousands.

The data is taken from the first 6 observed stock prices of 5 different companies with a market cap of at least $100 billion in 2019, data on the 5th and 6th is absent due stock prices not being observed over the weekend.

$$\boldsymbol{X} = \begin{array}{c} \\ \mathbf{02/01} \\ \mathbf{03/01} \\ \mathbf{04/01} \\ \mathbf{07/01} \\ \mathbf{08/01} \\ \mathbf{09/01} \end{array} \begin{array}{c} \begin{array}{ccccc} \boldsymbol{FB} & \boldsymbol{AMZN} & \boldsymbol{NVDA} & \boldsymbol{MSFT} & \boldsymbol{PEPSI} \end{array} \\ \begin{bmatrix} \mathbf{135.68} & \mathbf{1539.13} & \mathbf{136.22} & \mathbf{101.12} & \mathbf{109.28} \\ \mathbf{131.74} & \mathbf{1500.28} & \mathbf{127.99} & \mathbf{97.40} & \mathbf{108.26} \\ \mathbf{137.95} & \mathbf{1575.39} & \mathbf{136.19} & \mathbf{101.93} & \mathbf{110.48} \\ \mathbf{138.05} & \mathbf{1629.51} & \mathbf{143.40} & \mathbf{102.06} & \mathbf{109.53} \\ \mathbf{142.53} & \mathbf{1656.58} & \mathbf{139.83} & \mathbf{102.80} & \mathbf{110.58} \\ \mathbf{144.23} & \mathbf{1659.42} & \mathbf{142.58} & \mathbf{104.27} & \mathbf{107.49} \end{bmatrix} \end{array}$$

The two elements are removed and replaced with "NA" to simulate a missing dataset $\boldsymbol{X'}$,

$$\boldsymbol{X'} = \begin{array}{c} \\ \mathbf{02/01} \\ \mathbf{03/01} \\ \mathbf{04/01} \\ \mathbf{07/01} \\ \mathbf{08/01} \\ \mathbf{09/01} \end{array} \begin{array}{c} \begin{array}{ccccc} \boldsymbol{FB} & \boldsymbol{AMZN} & \boldsymbol{NVDA} & \boldsymbol{MSFT} & \boldsymbol{PEPSI} \end{array} \\ \begin{bmatrix} \mathbf{135.68} & \mathbf{1539.13} & \mathbf{136.22} & \mathbf{101.12} & \mathbf{109.28} \\ \mathbf{131.74} & \mathbf{1500.28} & \mathbf{127.99} & \mathbf{97.40} & \mathbf{108.26} \\ \mathbf{137.95} & \mathbf{1575.39} & \mathbf{136.19} & \boldsymbol{NA} & \mathbf{110.48} \\ \mathbf{138.05} & \mathbf{1629.51} & \mathbf{143.40} & \mathbf{102.06} & \mathbf{109.53} \\ \mathbf{142.53} & \mathbf{1656.58} & \mathbf{139.83} & \mathbf{102.80} & \mathbf{110.58} \\ \mathbf{144.23} & \boldsymbol{NA} & \mathbf{142.58} & \mathbf{104.27} & \mathbf{107.49} \end{bmatrix} \end{array}$$

.

The log returns are then computed which leaves us with 3 missing data entries.

Listing 11: Log Returns

```
       [,1] [,2] [,3] [,4] [,5]
[1,] -0.02946 -0.02556 -0.06231 -0.03748 -0.00937
[2,] 0.04606 0.04885 0.06209 abs 0.02029
[3,] 0.00072 0.03377 0.05158 abs -0.00863
[4,] 0.03193 0.01647 -0.02521 0.00722 0.00954
[5,] 0.01185 abs 0.01947 0.01419 -0.02834
```

Notice that one line of returns has been removed. This is because for the log returns on the $\mathbf{2^{nd}}$ of January, we do not have access to the returns on the $\mathbf{1^{st}}$ so Listing 11 denotes returns from the $\mathbf{3^{rd}}$ of January to the $\mathbf{9^{th}}$. The columns correspond to the columns seen in $\boldsymbol{X'}$.

Using (3) and the method highlighted in 7.3, we then compute the approximated sample correlation matrix .

Listing 12: Estimated Correlation matrix using Pairwise Deletion

```
      [,1] [,2] [,3] [,4] [,5]
[1,] 1.00000 0.81532 0.58868 0.89997 0.62692
[2,] 0.81532 1.00000 0.94295 1.00000 0.62895
[3,] 0.58868 0.94295 1.00000 0.89942 0.19210
[4,] 0.89997 1.00000 0.89942 1.00000 -0.12503
[5,] 0.62692 0.62895 0.19210 -0.12503 1.00000
```

Notice that in this sample, the log returns of Amazon and Microsoft have a perfect correlation of 1. Both Microsoft and Amazon are the stocks with missing data entries so the actual sample correlation between Amazon and Microsoft is most likely not 1. This rarely happens in practice due to the thousands of observations that are being used which all effect the correlation coefficient however there have been such instances where companies in the s&p 500 had correlation levels as high as 0.95.

Listing 13: Eigenvalues of Listing 12 Matrix

```
$values
[1] 3.743917360 1.203880638 0.380563954
0.003119779 -0.331481731
```

Now clearly we cannot proceed with further analysis as our correlation matrix is not positive semi-definite with a negative eigenvalue of -0.331. Dykstras Alternating Projections Algorithm will be used to find the closest Positive semi-definite matrix so we can continue.

Listing 14: Nearest Correlation Matrix

```
        [,1] [,2] [,3] [,4] [,5]
[1,] 1.00000 0.86580 0.61947 0.77615 0.55769
[2,] 0.86580 1.00000 0.90742 0.84575 0.51969
[3,] 0.61947 0.90742 1.00000 0.87384 0.18717
[4,] 0.77615 0.84575 0.87384 1.00000 0.02326
[5,] 0.55769 0.51969 0.18717 0.02326 1.00000
```

Listing 15: DAPA2 on listing 12 matrix output

```
start.time <- Sys.time()
> NCLX'<-DAPA(CLX')
> NCLX'
$Frobenius_Norm
[1]0.408133

$Rank
[1]5

$iterations
[1]22

$eigenvalues
[1]3.610174e+00,1.088780e+00,3.010465e-01
,3.610174e-07,3.610174e-07

>end.time<-Sys.time()
>time.taken<-end.time-start.time
>time.taken
Time_difference_of_0.1678419_secs
```

As we can see the eigenvalues are now strictly positive and the algorithm converged in less than 0.2 seconds. Now recall (5), to convert the nearest sample correlation matrix into a sample approximated covariance matrix we now need to compute the diagonal matrix of approximated sample standard deviations. To do this we will use listwise deletion for MSFT and AMZN to account for 'abs'. R can efficiently account for this with its 'sd' function along with 'na.rm=TRUE'.

Listing 16: Diagonal matrix D of standard deviations

```
n<-nrow(NCLX')
D<-matrix(0,nrow=n,ncol=n)
for(i␣in␣1:n){
f<-sd(NCLX'[,i],na.rm = TRUE)
D[i,i]<-f}
diag(D)
[1] 0.02917274 0.03214746 0.05243942
0.02804195 0.01880339
```

Finally, using matrix $\boldsymbol{D}$ in listing 16 and the correlation matrix in listing 14, a proposed approximated sample covariance matrix can be generated.

Listing 17: Estimated Covariance matrix

```
       [,1] [,2] [,3] [,4] [,5]
[1,] 0.00085 0.00081 0.00095 0.00063 0.00031
[2,] 0.00081 0.00103 0.00153 0.00076 0.00031
[3,] 0.00095 0.00153 0.00275 0.00128 0.00018
[4,] 0.00063 0.00076 0.00128 0.00079 0.00001
[5,] 0.00031 0.00031 0.00018 0.00001 0.00035
```

Listing 18: Estimated Covariance Matrix Eigenvalues

```
eigen() decomposition
$values
[1] 4.803976e-03 6.691159e-04 3.012278e-04 2.905852e-13 1.286987e-13
```

The covariance matrix we have found has strictly positive eigenvalues which means it is computationally positive definite so it can be used in a range of financial applications due to the non-singularity. The application we will be focusing on is computing the Global Minimum Variance Portfolio using the estimated Covariance matrix in Listing 17. This can be solved by solely Using matrix solution (38). This is one example where finding the nearest Correlation matrix may be directly applicable in finance.

The Global Minimum Variance Portfolio code used to find the weights below can be found in the appendix on listing 40.

Listing 19: Weights for the Global Minimum Variance Portfolio

```
[1,] -0.6948917
[2,] -0.4813277
[3,] -0.1557765
[4,] 1.2656917
[5,] 1.0663042
```

At first, the solution may seem strange due to the negative weight's assigned to Facebook, Amazon and Nvidia. Recall section 8.2, the negative weights represent taking a short position and the positive ones taking a long position(notice the sum of weights still equal to 1). This can be better understood as when you go long, you own an asset and when you go short you owe the asset hence the respective positive and negative weightings. The weightings above $|\mathbf{1}|$ represent leverage(purchasing a stock on borrowed money), the solution tells us that the Minimum Variance Portfolio consists of shorting 69% of your money on Facebook, 48% of your money on Amazon and 15% of your money on Nvidia.The approximated Global Minimum Variance Portfolio for $\boldsymbol{V}$ would contain going short on FB, AMZN and NVDA while going long on MSFT and PEPSI with the corresponding proportions above.

This may seem like nonsense in practice due to the need of leverage and the fact that some investors prefer not to stay in short positions as theoretically the risk of loss is infinite compared to a finite risk for going Long. If you go long on Amazon when the stock costs \$1500, the maximum amount you can lose is \$1500 when the stock hits \$0. However if you short Amazon at \$1500, the stock can theoretically infinitely grow leaving you with large losses.

One way of getting more interpretable results is to limit the weights such that $\forall \boldsymbol{i} \in [\mathbf{1}, \boldsymbol{n}], \mathbf{0} \leq \boldsymbol{W_i} \leq \mathbf{1}$ however solution (38) will not hold as shorting positions are now not allowed due to all of the weights being positive.

The "IntroCompFinR" package made by Eric Zivot for the University of Washington contains a "GlobalMin.Portfolio" function which uses expected returns and a covariance matrix to find the Global Minimum Variance Portfolio. How the package is initialized can be seen on listing 41 in the appendix.

The expected returns are simply the estimated means for each stock, these will be computed using listwise deletion.

Listing 20: Expected Returns

```
u<-colMeans(LX',na.rm=TRUE)
u
[1]0.012222015,0.018384479,0.009126404
-0.005352969,-0.003303115
```

Applying the globalmin.portfolio function to our expected returns and our estimated sample covariance Matrix gives us

Listing 21: Weights with no Shorts

```
Portfolio weights:
  [1] 0.000 0.000 0.000 0.306 0.694
```

Listing 21 tells us to invest 69% of our money into Pepsi and the remaining 31% into Microsoft. Notice that the largest weights in the previous portfolio with shorting was also Pepsi and Microsoft. This has given us one possible solution to find the Global Minimum Variance Portfolio from a matrix of stock returns with missing data.

We will now approach this problem by Projecting an estimated covariance matrix onto the positive semi-definite set which makes it computationally positive definite. The corresponding projection code can be found in the Appendix section on listing 42 and 43. Using Pairwise deletion to deal with the missing values, we begin with approximating the covariance matrix.

Listing 22: Approximate Covariance Matrix using Pairwise Deletion

```
       [,1] [,2] [,3] [,4] [,5]
[1,] 0.00085 0.00088 0.00090 0.00079 0.00034
[2,] 0.00088 0.00103 0.00182 0.00094 0.00029
[3,] 0.00090 0.00182 0.00275 0.00103 0.00019
[4,] 0.00079 0.00094 0.00103 0.00079 -0.00007
[5,] 0.00034 0.00029 0.00019 -0.00007 0.00035
```

Listing 23: Eigenvalues of Listing 22 Covariance Matrix

```
eigen() decomposition
$values
[1] 0.00497 0.00074 0.00048 -0.00012 -0.00029
```

The matrix is however not positive semi-definite with two negative eigenvalues of -0.00012 and -0.00029. Our next step is to compute the nearest positive semi-definite matrix with respect to a weighted Frobenius norm.

Listing 24: Nearest Covariance Matrix using Near_Var

```
       [,1] [,2] [,3] [,4] [,5]
[1,] 0.00090 0.00086 0.00092 0.00075 0.00031
[2,] 0.00086 0.00123 0.00174 0.00086 0.00024
[3,] 0.00092 0.00174 0.00279 0.00106 0.00021
[4,] 0.00075 0.00086 0.00106 0.00087 0.00000
[5,] 0.00031 0.00024 0.00021 0.00000 0.00040
```

Listing 25: Near_Var output

```
$eigenvalues
[1] 4.969300e-03 7.410639e-04 4.785939e-04 4.969300e-11 4.969300e-11
$normF
[1] 0.0003175401
$iterations
[1] 2
$converged
[1] TRUE
```

The resulting matrix now has eigenvalues which are strictly positive hence the matrix is non singular. Now that an estimated covariance matrix has been found which is non singular, it can be used directly to find the estimated Global Minimum Portfolio Weights.

Listing 26: Global Minimum Variance Portfolio Weights using Listing 24

```
[1,] -1.26794734
[2,] -0.30400137
[3,] -0.02433171
[4,] 1.42955874
[5,] 1.16672167
```

Once we restrict the weights to be strictly non negative, we get

Listing 27: Global Min Portfolio Weights using listing 24 no shorts

```
Portfolio weights:
  [1] 0.0000 0.0000 0.0000 0.3187 0.6813
```

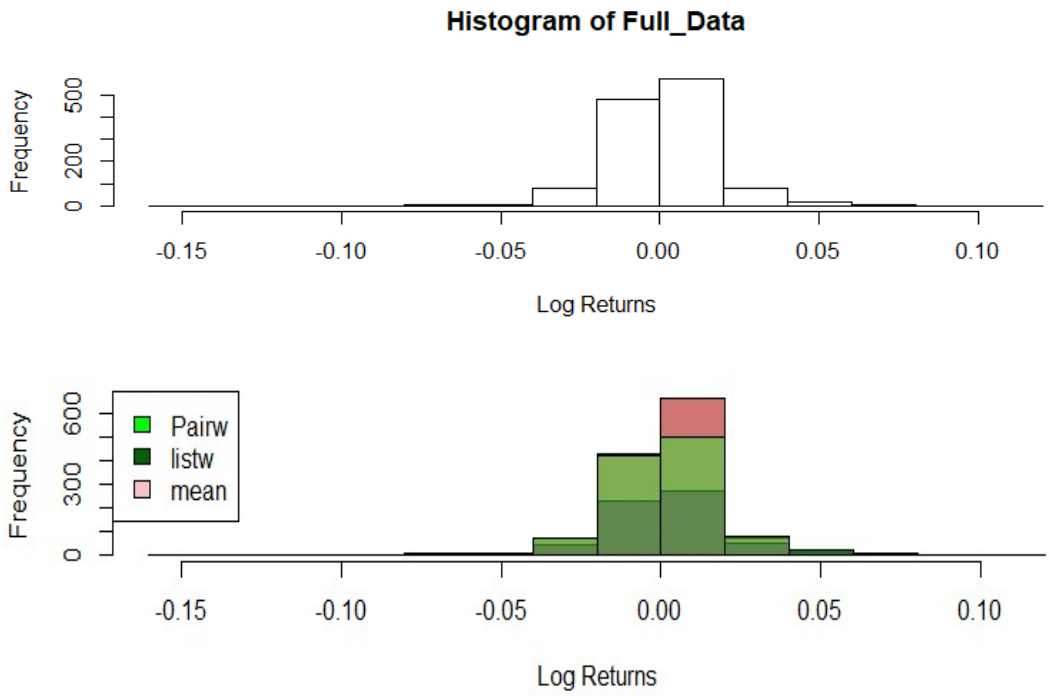
All of the weights are better highlighted below.

	FB	AMZN	MSFT	NVDA	PEPSI
Cor to Cov shorts	-0.695	-0.481	-0.156	1.266	1.066
Cor to Cov no shorts	0	0	0	0.306	0.694
Project Cov shorts	-1.268	-0.304	-0.024	1.430	1.167
Project Cov no shorts	0	0	0	0.319	0.681

We can see that the overall patterns of each method coincide with one another. Each of the portfolios consist of going long on NVDA and PEPSI and smaller weights are assigned to Facebook, Amazon and Microsoft. The biggest clear difference between the weights are in Facebook for the Cor to Cov method compared to the projection Method however Facebook has the most shorted weight attached to it in both of the methods. Pepsi Has the most weight attached to it going Long in all of the methods as well.

10.2 Distributional Effects of Imputation

Throughout this paper we have highlighted how imputation techniques can be used to deal with missing data. When deciding what technique to use, we need to take into account the effect each method has on the overall distribution of the data. To better highlight this we will take all 252 closing prices of the stocks highlighted in $\boldsymbol{X}$. 2 out of 30 stock prices where missing in $\boldsymbol{X}$ so we will keep it proportional and remove $\frac{\mathbf{2}}{\mathbf{30}} \times \mathbf{(252 \times 5) = 84}$ entries in our new matrix. The replicable code can be found in the Appendix on listing 44.



The largest distribution change came from Liswise deletion. This is understandable as at least 84 rows of stock returns have been removed which has shrunk the distribution completely. Using Mean imputation narrowed the distribution with more elements centred around the mean. Using pairwise deletion and dropping the missing values was found to have the smallest distributional change amongst the classical imputation methods, this is probably because many more entries are kept compared to listwise deletion however each variable(stock) would result in different sample sizes which could be problematic for further analysis.

10.3 Conclusion

If you take into account the lack of optimal nearness, the methods highlighted in 10.1 are two possible strategies to deal with a covariance matrix that is not positive definite. The accuracy of these results is obviously questionable due to the small sample size which suggests high type two error. Also the data being Completely Missing at Random has been assumed and not proven so this could further increase the level of type two error. Other data imputation techniques can possibly be used on the estimated Standard deviations to improve the first method. Listwise deletion has been found to distort the distribution of the data the most amongst the classical imputation methods so it is not a recommended strategy when dealing with stock return data. Simply dropping the missing values rather than the whole row had less of an effect on the distribution. If you require for the distribution to remain fairly similar throughout the statistical analysis then removing missing data could prove to be costly. A sensible approach would be to explore other techniques such as Maximum Likelihood and Multiple Imputation which can be found in C.Acock(2005), these methods reduce the overall bias of the missing data compared to Pairwise and Listwise Deletion. Other strategies

have emerged due to the extensive research that has been published on Machine Learning. Relevant literature comparing the use Neural Networks and Expectation Maximization for data imputation can be found in Mohamed & Marwala(2007).

Several factors have also long been contested over Markowitz' portfolio Theory. The theory suggests that people invest primarily based on the mean(returns) and variance(risk). A natural assumption to then make is that the returns are normally distributed as the normal distribution is defined by its mean and variance. This sometimes does not hold in practice as the normal distribution does not account for extreme events such as the COVID-19 pandemic.

In reality returns can be heavily fat tailed so other models may be preferable such as Behavioural Portfolio Theory (Shefrin & Statman 2000) which suggests that investors make their decisions based on several factors other than risk and return and Black Litterman Models(Black et.al 1992) which build from Markowitz' Portfolio Theory and solve some of its problems. A generally sensible approach for an investor would be to consider a multitude of models when constructing a portfolio. The first step is to try and minimize any missing data before looking at imputation techniques so analysing asset prices rather than asset returns could be more beneficial as you deal with less uncertainty. If you are confident in your imputed data but it ends up having negative eigenvalues then the methodology highlighted in 10.1 can solve this.

11 Final Thoughts & Possible Further Study

The steps taken in this paper to generate a portfolio highlight how Dykstra's Alternating Projections Algorithm can be used in a financial setting. Lanczos' Iteration effectively reduced the computational time across all of the tests that where performed, the biggest drawback of the algorithm has been found to be a linear convergence rate (Higham 2002) so further possible improvements can be made to make it more efficient. A possible improvement in computational time could arise from iterating across the diagonal entries of the weighted matrix $\boldsymbol{W}$ until a nearer matrix has been found. Higham suggested that adding possible rank constraints to the solution could further improve the computational time. Adding some tolerance level when comparing values to 0 in $\boldsymbol{R}$ to ensure strict positivity reduced the number of errors that appeared throughout the tests.

Unfortunately an excessive amount of time was taken constructing the Nearest correlation matrix code so I was not able to pursue many other endeavours. There is still much room for improving the nearness of the approximated positive definite covariance matrix. I would of liked to explore the effect different imputation techniques have on the closeness of the covariance matrix using both of the methods highlighted in section 10. Other ways have also been explored to estimate inconsistent covariance matrices such as using maximum likelihood in Deng & Yuan(2009) and Nearest Neighbour leaning to estimate the missing values which is found in Wang & E. Raftery(2002).

12 Bibliography

Nicholas J. Higham(2001) *Computing the nearest Correlation matrix- a problem in Finance,IMA Journal of Numerical Analysis, Volume 22*, 329–343.

Charles R. Johnson & Roger A. Horn(1985), *Matrix Analysis*, Cambridge University Press, second edition

G.B. Price(1947), *Some Identities in the Theory of Determinants, The American Mathematical Monthly*,volume 54:2, 75-90

Beauregard, Raymond A & Fraleigh, John B.(1973), *A First Course In Linear Algebra: with Optional Introduction to Groups, Rings, and Fields*, Chapter 39

George R. Terrell(1999), *Mathematical Statistics, Springer Texts in Statistics*, section 2.7

Rockafellar. Tyrrell .R(1970), *Convex Analysis, Princeton New Jersey, Princeton University Press*, 10–23

W. Glunt; T.L. Hayden; S. Hong & J.Wells(1990), *An alternating projection algorithm for computing the nearest Euclidean distance matrix. SIAM K, Matrix anal. Appl.., 11(4)*, 589-600

John Von Neumann(1949),*On Rings of Operators. Reduction Theory, Annals of Mathematics, Apr, Second Series, Vol.50, No.2*, 401-485

S.Han(1988), *A Successive Projection Method. Mathematical Programming 40*, 1-14

R. Dykstra(1983), *A Method for Finding Projections onto the Intersection of Convex Sets in Hilbert Spaces,Lecture Notes in statistics,Volume 37*, 28-46

Peter Deuflhard & Andres Hohmann(2003), *Numerical Analysis in Modern Scientific Computing,An introduction, Second Edition,Texts in Applied Mathematics, Series 43*, section 8.5

Alston Householder(1958), *Unitary Triangularization of a Nonsymmetric Matrix. Journal of the ACM(JACM), Association for Computing Machinery,Vol 5, no.4*, 339-342

Robert S Hudson & Andros Gregoriou(2010), *Calculating and Comparing Security Returns is harder than you think: A comparison between Logarithmic and Simple returns*

Harry Markowitz(1952), *The Journal of Finance, Mar, Vol. 7, no.1, Wiley for the American Finance Association*, 77-91

Merton, Robert. (1972). *An Analytic Derivation of the Efficient Portfolio Frontier. Journal of Financial and Quantitative Analysis. 7. 10.2307/2329621.*

David Goldberg(1991), *What Every Computer Scientist Should Know About Floating-Point Arithmetic , Association for Computing Machinery, New York, Vol. 23 no.1.*

Donald B. Rubin(1976), *Inference and Missing Data,Biometrika, Volume 63,Issue 3*, 581-592

Cornelius Lanczos(1950), *An iteration Method for the Solution of the Eigenvalue Problem of LInear Differential and Integral Operators, Journal of Research of the National Bureau of Standards, Vol 45, No 4*, 255-282

Pettengill, Glenn N.(2003) *"A Survey of the Monday Effect Literature." Quarterly Journal of Business and Economics, vol. 42, no. 3/4*, 3-27

Boyle J.P, Dykstra R.L.,(1986) *A Method for Finding Projections onto the Intersetion of Convex Sets in Hilbrert Spaces, Advances in Order Restricted Statistical Inference. Lecture Notes in Statistics, Vol 37, Springer New York*

Richard L. Dykstra(1983), *An Algorithm for Restricted Least Squares Regression, Journal Of the American Statistical Association, Vol 78, No. 384*, 837-842

Kenji Ueno, Koji Shiga, Toshikazu Sunada, Shigeyuki Morita(2003)*A Mathematical Gift, III: The Interplay Between Topology, Functions, Geometry, and Algebra, Mathematical World, Volume 23, American Mathematical Society,* Lecture 2

Grinberg, Raffi(2017), *The Real Analysis Lifesaver: All the Tools you Need to Understand Proofs. Princeton University Press*, Chapter 10

James Munkres(2000), *Topology, 2nd edition,Pearson New International Edition*, Chapter 2

Shefrin,Hersh and Meir Statman, *Behavioral Portfolio Theory, The Journal Of Financial and Quantitative Analysis, Vol 35, No.2*, 127-151

James Joseph Sylvester(1852), *A demonstration of the theorem that every homogeneous quadratic polynomial is reducible by real orthogonal substitutions to the form of a sum of positive and negative squares, Philosophical Magazine IV*, 138-142

Alan C. Acock(2005), *Working with Missing Values,A journal of Marriage and Family, vol. 67, no.4*, 1012-1028

Black, Fischer & LItterman(1992), *A Global Portfolio Optimization. A Financial Analysts Journal, Vol 48, no.5*, 28-43

Ben-Israel, Adi(2002), *The Moore of the Moore-Penrose Inverse, The Electric Journal of Linear Algebra, Volume 9*, 150-157

Golub & Kahan (1965), *Calculating the Singular Values and Pseudo-Inverse of a Matrix, Journal of the Society for Industrial and Applied Mathematics, Series B, Numerical Analysis, Vol.2, No.2*, 205-224

Xinwei Deng & Ming Yuan(2002)*Journal of COmputational and Graphical Statistics, Vol. 18, No.3*, 640-657

Naisyin Wang & Adrian E. Raftery(2002),*Journal of the American Statistical Association, Vol.97, No.460*, 994-1006

Nelwamondo, F, Mohamed & Marwala, T.(2007)*Missing data: A comparison of neural Network and Expectation Maximization techniques, Current Science, 93(11)*, 1514-1521

Ong, Seow-Eng, Malik Ranasinghe(2000)*Portfolio Variance and Correlation Matrices. The Journal of Real Estate Portfolio Management, 6(1)*, 1-6

13 Appendix

Part 1 and Part 2 in this section should all be pasted as one section of code.

Listing 28: How to implement code

```
#part 1
add_formula<-function(a,b){
#part 2
print(a+b)}

#Implement part 1 and part 2 in R terminal as

add_formula<-function(a,b){
print(a+b)}
```

Listing 29: DAPA with no Lanczos' iteration part 1

```
library(expm)
library(matrixcalc)
library(Matrix)
DAPA1<-function (a,Convergence_Tolerance = 1e-07,Iteration_Limit=300,
Positive_eigen_Tolerance = 1e-07)
{W <- diag(1, nrow(a),ncol(a))
Projection_Onto_Set_D<-function(a){
diag(a)<-1
  return(a)
}
Projection_Onto_Set_S<-function(a){
  Wsqrt <- sqrtm(W)
  Winvsqrt <- matrix.inverse(Wsqrt)
  brac<-Wsqrt %*% a %*% Wsqrt
  Eigen_Brac<-eigen(a)
  QBrac<-Eigen_Brac$vectors
  QValBrac<-Eigen_Brac$values
  D<-diag(QValBrac)
  D[D<Positive_eigen_Tolerance]<- Positive_eigen_Tolerance
  Spec_Brac<-QBrac %*% D %*% t(QBrac)
  PS<-Winvsqrt %*% Spec_Brac %*% Winvsqrt
  return(PS)
}
n <- nrow(a)
DS <- matrix(0,nrow(a),ncol(a))
Y<-a
j <- 0
converged <- FALSE
while (!converged ) {
  R <- Y - DS
  X <- Projection_Onto_Set_S(R)
  DS<-X-R
  Y <- Projection_Onto_Set_D(X)
  Convergence_Test <- norm(Y - X, "I")/norm(Y, "I")
  j <- j + 1
  converged <- (Convergence_Test<= Convergence_Tolerance)
  if(j> Iteration_Limit){
    break }}
```

Listing 30: DAPA with no Lanczos' iteration part 2

```
if (!converged)
  warning("DAPA␣did␣not␣converge")
eigen_Y <- eigen(Y, symmetric = TRUE)
eigenY_values <- eigen_Y$values
Positive_eigen_Tolerance <- Positive_eigen_Tolerance *abs(eigenY_values[1])
if (eigenY_values[n] < Positive_eigen_Tolerance) {
  eigenY_values[eigenY_values < Positive_eigen_Tolerance] <- Positive_eigen_Tolerance
EigenY_Vectors <- eigen_Y$vectors
Y <- EigenY_Vectors %*% (eigenY_values * t(EigenY_Vectors))}
f<-eigen(Y)
EigenValues<-f$values
diag(Y) =1
Rank<-qr(Y)$rank
structure(list(as.matrix(Y),
Frobenius_Norm = norm(a - Y, "F"),Rank = Rank,iterations = j,
eigenvalues= EigenValues))}
```

Listing 31: DAPA with Lanczos' iteration part 1

```
library(expm)
library(matrix)
library(matrixcalc)
library(mgcv)
DAPA2<-function (a,Convergence_Tolerance = 1e-07,
Iteration_Limit=100,Positive_eigen_Tolerance = 1e-07)
{W <- diag(1, nrow(a),ncol(a))
Projection_Onto_Set_D<-function(a){
  diag(a)<-1
  return(a)}
Projection_Onto_Set_S<-function(a){
  Wsqrt <- sqrtm(W)
  Winvsqrt <- matrix.inverse(Wsqrt)
  eigena<-eigen(a)
  eigenval<-eigena$values
  numbeig<-eigenval
  numbeig<-numbeig[numbeig>Positive_eigen_Tolerance]
  g<-length(numbeig)
  F<-slanczos(a,g,-1)
  brac<- F$vectors
  Eigen_Brac<-eigen(a)
  QBrac<-Eigen_Brac$vectors
  QValBrac<-Eigen_Brac$values
  D<-diag(QValBrac)
  D[D<Positive_eigen_Tolerance]<- Positive_eigen_Tolerance
  Spec_Brac<-QBrac %*% D %*% t(QBrac)
  PS<-Winvsqrt %*% Spec_Brac %*% Winvsqrt
  return(PS) }
n <- nrow(a)
DS <- matrix(0,nrow(a),ncol(a))
Y<-a
j <- 0
converged <- FALSE
while (!converged ) {
  R <- Y - DS
  X <- Projection_Onto_Set_S(R)
  DS<-X-R
  Y <- Projection_Onto_Set_D(X)
  Convergence_Test <- norm(Y - X, "I")/norm(Y, "I")
  j <- j + 1
  converged <- (Convergence_Test<= Convergence_Tolerance)
  if(j> Iteration_Limit){ break }}
```

Listing 32: DAPA with Lanczos' iteration part 2

```
if (!converged)
  warning("DAPA␣did␣not␣converge")
eigen_Y <- eigen(Y, symmetric = TRUE)
eigenY_values <- eigen_Y$values
Positive_eigen_Tolerance <- Positive_eigen_Tolerance *abs(eigenY_values[1])
if (eigenY_values[n] < Positive_eigen_Tolerance) {
  eigenY_values[eigenY_values < Positive_eigen_Tolerance]<- Positive_eigen_Tolerance
EigenY_Vectors <- eigen_Y$vectors
Y <- EigenY_Vectors %*% (eigenY_values * t(EigenY_Vectors))}
f<-eigen(Y)
EigenValues<-f$values
diag(Y) =1
Rank<-qr(Y)$rank
structure(list(as.matrix(Y), Frobenius_Norm =
norm(a - Y, "F"),Rank = Rank,iterations = j,eigenvalues=
EigenValues))}
```

Listing 33: Test example 1

```
C<-matrix(c(135.68,131.74,137.95,138.05
,142.53,1539.13 ,1500.28,1575.39 ,1629.51
,NA,136.22 ,127.99 ,136.19 ,143.40 ,139.83
,101.12 ,97.40,NA ,104.27,109.28,108.26,110.48
,109.53,110.58,1),5,5)
D<-cor(C,use="pairwise.complete.obs")
D
D

          [,1] [,2] [,3] [,4] [,5]
[1,] 1.0000000 0.8914397 0.7809127 0.9982986 -0.7610282
[2,] 0.8914397 1.0000000 0.9543933 0.9626585 0.2361299
[3,] 0.7809127 0.9543933 1.0000000 0.7812869 -0.3031823
[4,] 0.9982986 0.9626585 0.7812869 1.0000000 -0.8297377
[5,] -0.7610282 0.2361299 -0.3031823 -0.8297377 1.0000000
```

Listing 34: Test eg1 DAPA with no Slanczos' Iteration

```
> start.time <- Sys.time()
> DAPA1(D)
[[1]]
          [,1] [,2] [,3] [,4] [,5]
[1,] 1.0000000 0.754334492 0.8108775 0.9997737 -0.657510981
[2,] 0.7543345 1.000000000 0.8949435 0.7513935 -0.007122293
[3,] 0.8108775 0.894943508 1.0000000 0.8171191 -0.257204826
[4,] 0.9997737 0.751393451 0.8171191 1.0000000 -0.662897490
[5,] -0.6575110 -0.007122293 -0.2572048 -0.6628975 1.000000000

$Frobenius_Norm
[1] 0.5813537

$Rank
[1] 4

$iterations
[1] 26

$eigenvalues
[1] 3.757332e+00 1.123748e+00 1.189197e-01 3.757332e-08
3.757332e-08

> end.time <- Sys.time()
> time.taken <- end.time - start.time
> time.taken
Time difference of 0.215899 secs
```

Listing 35: Test eg1 DAPA with Slanczos' Iteration

```
 start.time <- Sys.time()
> DAPA2(D)
$matrix
          [,1] [,2] [,3] [,4] [,5]
[1,] 1.0000000 0.754334346 0.8108775 0.9997735 -0.657510858
[2,] 0.7543343 1.000000000 0.8949434 0.7513932 -0.007122383
[3,] 0.8108775 0.894943421 1.0000000 0.8171192 -0.257204907
[4,] 0.9997735 0.751393184 0.8171192 1.0000000 -0.662897472
[5,] -0.6575109 -0.007122383 -0.2572049 -0.6628975 1.000000000

$Frobenius_Norm
[1] 0.5813541

$Rank
[1] 4

$iterations
[1] 27

$eigenvalues
[1] 3.757332e+00 1.123748e+00 1.189197e-01 3.757332e-07
 3.757332e-07

> end.time <- Sys.time()
> time.taken <- end.time - start.time
> time.taken
Time difference of 0.2157941 secs
>
```

Listing 36: Test eg 2 DAPA with no Slanczos'

```
set.seed(1)
c_50x50<-matrix(rnorm(200,5), ncol=50,nrow=50)
c_50x50[sample(1:length(c_50x50), 20, replace = FALSE)] <- NA
f_50x50<-cor(c_50x50,use="pairwise.complete.obs")
$Frobenius_Norm
[1] 0.04266866
$Rank
[1] 31
$iterations
[1] 84
$eigenvalues
 [1] 1.658760e+01 1.302287e+01 1.262202e+01 7.765477e+00 1.518997e-03
 [6] 5.436186e-04 3.850084e-06 1.658760e-06 1.658760e-06 1.658760e-06
[11] 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06
[16] 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06
[21] 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06
[26] 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06
[31] 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06
[36] 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06
[41] 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06
[46] 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06
> end.time <- Sys.time()
> time.taken <- end.time - start.time
> time.taken
Time difference of 6.580724 secs
```

Listing 37: Test eg 2 DAPA with Slanczos'

```
start.time <- Sys.time()
 DAPA2(f_50x50)
 end.time <- Sys.time()
 time.taken <- end.time - start.time
 time.taken
#output
$Frobenius_Norm
[1] 0.04266866
$Rank
[1] 31
$iterations
[1] 84
$eigenvalues
 [1] 1.658760e+01 1.302287e+01 1.262202e+01 7.765477e+00 1.518997e-03
 [6] 5.436186e-04 3.850084e-06 1.658760e-06 1.658760e-06 1.658760e-06
[11] 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06
[16] 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06
[21] 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06
[26] 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06
[31] 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06
[36] 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06
[41] 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06
[46] 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06 1.658760e-06

> end.time <- Sys.time()
> time.taken <- end.time - start.time
> time.taken
Time difference of 5.753589 secs
```

Listing 38: Test eg 3 DAPA with no Slanczos'

```
set.seed(1)
c_500x500<-matrix(rnorm(200,5), ncol=500,nrow=500)
c_500x500[sample(1:length(c_500x500), 200, replace = FALSE)] <- NA
f_500x500<-cor(c_500x500,use="pairwise.complete.obs")

start.time <- Sys.time()
DAPA1(f_500x500)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
#output
$Frobenius_Norm
[1] 0.04558927
$Rank
[1] 499

$iterations
[1] 1

> end.time <- Sys.time()
> time.taken <- end.time - start.time
> time.taken
Time difference of 24.47763 secs
```

Listing 39: Test eg 3 DAPA with Slanczos'

```
start.time <- Sys.time()
DAPA2(f_500x500)
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
$Frobenius_Norm
[1] 0.04558927
$Rank
[1] 499
$iterations
[1] 1
> end.time <- Sys.time()
> time.taken <- end.time - start.time
> time.taken
Time difference of 23.84923 secs
```

Listing 40: Global Minimum Variance Portfolio weights code

```
MinVarPortfolio<-function(estvar){
  s<-diag(estvar)
  n<-nrow(estvar)
  c<-ncol(estvar)
  if(c!=n){stop("not␣a␣square␣Covariance␣Matrix")}
  if(any(s<0))
  {stop("Covariance␣matrix␣seems␣to␣be␣negative␣semi-definite")}
  else {
    vect1<-as.matrix(rep(1,each=n))
    Numerator<-matrix.inverse(as.matrix(estvar))%*%vect1
    denom1<-t(vect1) %*%matrix.inverse(as.matrix(estvar))
    denom2<-denom1 %*%vect1
    f<-Numerator/denom2[1,1]}
  print(f)
}
```

Listing 41: No shorts Global min portfolio using near cov matrix and estimated means

```
install.packages("IntroCompFinR", repos="http:/
/R-Forge.R-project.org")
library(IntroCompFinR)
EstMean<-colMeans(LogReturns,na.rm = TRUE)
 globalMin.portfolio(EstMean,Covaprox1,shorts
 = FALSE)
Call:
  globalMin.portfolio(er = EstMean, cov.mat =
 Covaprox1, shorts = FALSE)

Portfolio expected return: -0.003930358
Portfolio standard deviation: 0.01578388
Portfolio weights:
  [1] 0.000 0.000 0.000 0.306 0.694
```

Listing 42: Nearest positive definite matrix code part 1

```
library(expm)
library(Matrix)
library(matrixcalc)
library(mgcv)
Near_Var<-function (a,Convergence_Tolerance = 1e-10,
Iteration_Limit=300,Positive_eigen_Tolerance = 1e-10)
{W <- diag(1, nrow(a),ncol(a))
Projection_Onto_Set_S<-function(a){
  Wsqrt <- sqrtm(W)
  Winvsqrt <- matrix.inverse(Wsqrt)
  eigena<-eigen(a)
  eigenval<-eigena$values
  numbeig<-eigenval
  numbeig<-numbeig[numbeig>Positive_eigen_Tolerance]
  g<-length(numbeig)
  F<-slanczos(a,g,-1)
  brac<- F$vectors
  Eigen_Brac<-eigen(a)
  QBrac<-Eigen_Brac$vectors
  QValBrac<-Eigen_Brac$values
  D<-diag(QValBrac)
  D[D<Positive_eigen_Tolerance]<- Positive_eigen_Tolerance
  Spec_Brac<-QBrac %*% D %*% t(QBrac)
  PS<-Winvsqrt %*% Spec_Brac %*% Winvsqrt
  return(PS)
}
n <- nrow(a)
DS <- matrix(0,nrow(a),ncol(a))
Y<-a
j <- 0
converged <- FALSE
while (!converged ) {
  X <- Projection_Onto_Set_S(a)
  Y <- X
  Convergence_Test <- norm(Y - X, "I")/norm(Y, "I")
  j <- j + 1
  converged <- (Convergence_Test<= Convergence_Tolerance)
  if(j> Iteration_Limit){
    break }}
```

Listing 43: Nearest positive definite matrix code part 2

```
if (!converged)
  warning("DAPA␣did␣not␣converge")
eigen_Y <- eigen(Y, symmetric = TRUE)#Only using the
#positive Eiganvalues
eigenY_values <- eigen_Y$values
Positive_eigen_Tolerance <- Positive_eigen_Tolerance *abs(eigenY_values[1])
if (eigenY_values[n] < Positive_eigen_Tolerance) {
  eigenY_values[eigenY_values < Positive_eigen_Tolerance]<-Positive_eigen_Tolerance
  EigenY_Vectors <- eigen_Y$vectors
  Y <- EigenY_Vectors %*% (eigenY_values * t(EigenY_Vectors))
}
f<-eigen(Y)
EigenValues<-f$values
Rank<-qr(Y)$rank
structure(list(matrix=as.matrix(Y), Frobenius_Norm =
               norm(a - Y, "F"),Rank = Rank,iterations =
 j,eigenvalues=EigenValues))}
```

Listing 44: Histogram of each classical Imputation Method

```
fb<-FB[,5]
amzn<-AMZN[,5]
msft<-MSFT[,5]
nvda<-NVDA[,5]
pep<-PEP[,5]

close<-matrix(c(fb,amzn,nvda,msft,pep),nrow=252,ncol=5)
close
difFul<-diff(log(close))

set.seed(1)
close[sample(1:length(close), 84, replace = FALSE)] <- NA
dif<-diff(log(close))
listdif<-dif[complete.cases(dif),]
mean_dif<-dif
for(i in 1:ncol(mean_dif)){
  mean_dif[is.na(mean_dif[,i]), i] <- mean(mean_dif[,i],
 na.rm = TRUE)
}
mean_dif
hist(dif)


p1 <- hist(difFul) #full data
p2 <- hist(dif)#pairwise deletion
p3<-hist(listdif)#listwise deletion
p4<-hist(mean_dif)#mean imputation
```