High-Dimensional Covariance Estimation

Hector Kohler supervised by Prof. Jianxin Pan

January 2020

Abstract

In this undergraduate project we review and compare some modern covariance matrix estimators arising from the big data era. Indeed more and more data sets have high dimensions (more random variables than observations) especially in genomics. We begin this project by showing why classical estimators such as the Maximum Likelihood Estimator (MLE) fails for high-dimensional data. We then review and present major results on alternative estimators based on shrinkage and thresholding. We motivate and compare those new covariance estimators using computations in R code.

1 Introduction

1.1 Classical Covariance Estimation

Some Notation and background material: A random vector is a vector-valued random variable $\mathbf{X} = (X_1, \cdots, X_p)^T \sim F$. Its components $X_1, \cdots, X_p$ are univariate identically distributed random variables with distribtution F. The covariance of the random vector $\mathbf{X}$ is:

$$\mathbf{\Sigma} = \mathrm{E}\left[(x-\mu)(x-\mu)^T\right]$$

with the expectation :

$$\mathrm{E}(\boldsymbol{x}) = \boldsymbol{\mu} = \begin{pmatrix} \mathrm{E}(x_1) \\ \mathrm{E}(x_2) \\ \vdots \\ \mathrm{E}(x_p) \end{pmatrix} = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_p \end{pmatrix}$$

So the covariance of a random vector is a matrix:

$$\Sigma = (\sigma_{ij}) = \begin{pmatrix} \sigma_{11} & \cdots & \sigma_{1p} \\ \vdots & \ddots & \vdots \\ \sigma_{ip} & \cdots & \sigma_{pp} \end{pmatrix}$$

$\sigma_{ij} = \text{Cov}\,(x_i, x_j)$ and $\mathbf{\Sigma}$ has real entries, is symmetric, and is positive semidefinite [**6**]. When working with observations from a multivariate normal distribution:
$\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \overset{\text{iid}}{\sim} N_p(\mu, \Sigma)$, the random vector becomes a data matrix:

$$\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)^T = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1p} \\ x_{21} & x_{22} & \ldots & x_{2p} \\ \vdots & & & \\ x_{n1} & x_{n2} & \ldots & x_{np} \end{pmatrix}$$

Problem: Let $\mathbf{X} \in \mathbb{R}^{n \times p}$ be a data matrix with n observations and p random variables. The problem of covariance estimation is to find an estimator $\hat{\Sigma}(\mathbf{X})$ of the covariance Σ which minimizes $\|\hat{\Sigma}(\mathbf{X}) - \Sigma\|$ for some norm $\|.\|$ or maximizes the likelihood function. We also consider other aspects of the estimator such as computational implementation or positive definiteness (the most important one) so that the esimator is invertible and we can estimate the precision matrix Σ^{-1} as well.

Proposition 1.2.1: The sample covariance matrix defined as

$$\mathbf{S} = \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{x}_i - \overline{\boldsymbol{x}}) (\boldsymbol{x}_i - \overline{\boldsymbol{x}})^T$$

with $\overline{\boldsymbol{x}}$ is the sample mean vector (the average of the $\boldsymbol{x}_i$'s), is a good estimator for the population covariance as it is the maximum likelihood estimator (MLE) for the multivariate normal distribution [**2**].

Proof: It follows from the likelihood function:

$$\begin{aligned} f_{\mathbf{X}}(\boldsymbol{x}|\mu, \mathbf{\Sigma}) &= \prod_{i=1}^{n} \det(\mathbf{\Sigma})^{-\frac{1}{2}} \exp\left(-\frac{1}{2} (\boldsymbol{x}_i - \mu)^T \mathbf{\Sigma}^{-1} (\boldsymbol{x}_i - \mu) \right) \\ &= \det(\mathbf{\Sigma})^{-\frac{n}{2}} \exp\left(-\frac{1}{2} \sum_{i=1}^{n} (\boldsymbol{x}_i - \mu)^T \mathbf{\Sigma}^{-1} (\boldsymbol{x}_i - \mu) \right) \end{aligned}$$

We get the log-likelihood:

$$\begin{aligned} \log\left(f_{\mathbf{X}}(\boldsymbol{x}|\mu, \mathbf{\Sigma})\right) &= -\frac{n}{2} \log\det(\mathbf{\Sigma}) - \frac{1}{2} \sum_{i=1}^{n} (\boldsymbol{x}_i - \mu)^T \mathbf{\Sigma}^{-1} (\boldsymbol{x}_i - \mu) \\ &= l(\mu, \mathbf{\Sigma}|\boldsymbol{x}) \end{aligned}$$

We take the derivative with respect to μ. Because for $\mathbf{A}$ symmetric positive semi-definite, $\frac{\partial \mathbf{w}^T \mathbf{A}\mathbf{w}}{\partial \mathbf{w}} = 2\mathbf{A}\mathbf{w}$ we get:

$$\begin{aligned}\frac{\partial l}{\partial \mu} &= -\frac{1}{2}\sum_{i=1}^{n} \frac{\partial}{\partial \mu}(\boldsymbol{x}_i - \mu)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i - \mu) \\ &= -\frac{1}{2}\sum_{i=1}^{n} 2\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i - \mu)^T\end{aligned}$$

We set equal to 0 to get $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i$. So the MLE for the mean parameter is the sample mean $\overline{\mathbf{X}}$.

Now since for every i, $(\boldsymbol{x}_i - \mu)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_i - \mu)$ is a scalar, we can consider it as the trace of 1×1 matrix. We can use the property $\operatorname{tr}(\mathbf{AB}) = \operatorname{tr}(\mathbf{BA})$ to get:

$$\begin{aligned}f_{\mathbf{X}}(\boldsymbol{x}|\mu, \boldsymbol{\Sigma}) &= \det(\boldsymbol{\Sigma})^{-\frac{n}{2}} \exp\left(-\frac{1}{2}\sum_{i=1}^{n} \operatorname{tr}\left((\boldsymbol{x}_i - \mu)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_i - \mu)\right)\right) \\ &= \det(\boldsymbol{\Sigma})^{-\frac{n}{2}} \exp\left(-\frac{1}{2}\sum_{i=1}^{n} \operatorname{tr}\left((\boldsymbol{x}_i - \mu)(\boldsymbol{x}_i - \mu)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1}\right)\right)\end{aligned}$$

We get the log-likelihood:

$$\begin{aligned}\log\left(f_{\mathbf{X}}(\boldsymbol{x}|\mu, \boldsymbol{\Sigma})\right) &= -\frac{n}{2}\log\det(\boldsymbol{\Sigma}) - \frac{1}{2}\sum_{i=1}^{n} \operatorname{tr}\left((\boldsymbol{x}_i - \mu)(\boldsymbol{x}_i - \mu)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1}\right) \\ &= l(\mu, \boldsymbol{\Sigma}|\boldsymbol{x})\end{aligned}$$

We look at the derivative of $l(\mu, \boldsymbol{\Sigma}|\boldsymbol{x})$ with respect to $\boldsymbol{\Sigma}$. First note that $\frac{\partial}{\partial \mathbf{A}} \operatorname{tr}(\mathbf{BA}) = \mathbf{B}^T$ (proofs of some matrix analysis results in Appendix). So we get:

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\Sigma}} \operatorname{tr}\left((\boldsymbol{x}_i - \mu)(\boldsymbol{x}_i - \mu)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1}\right) &= \frac{\partial}{\partial \boldsymbol{\Sigma}} \operatorname{tr}\left((\boldsymbol{x}_i - \mu)(\boldsymbol{x}_i - \mu)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}\right) \\ &= (\boldsymbol{x}_i - \mu)(\boldsymbol{x}_i - \mu)^{\mathrm{T}} \boldsymbol{\Sigma}^{-T}\boldsymbol{\Sigma}^{-T}\end{aligned}$$

Furthermore $\frac{\partial}{\partial a_{ij}} \log\left(\det(\mathbf{A})\right) = \frac{1}{\det(\mathbf{A})} \frac{\partial}{\partial a_{ij}} \det(\mathbf{A})$. And $\frac{\partial}{\partial a_{ij}} \det(\mathbf{A}) = \operatorname{adj}(\mathbf{A})_{ij}^T$ So:

$$\begin{aligned}\frac{\partial}{\partial \sigma_{ij}^2} \log\det(\boldsymbol{\Sigma}) &= \frac{1}{\det(\boldsymbol{\Sigma})} \frac{\partial}{\partial \sigma_{ij}^2} \det(\boldsymbol{\Sigma}) \\ &= \frac{\operatorname{adj}(\boldsymbol{\Sigma})_{ij}^T}{\det(\boldsymbol{\Sigma})} \\ &= (\boldsymbol{\Sigma}_{ij}^{-1})^T = \boldsymbol{\Sigma}_{ij}^{-T} \text{(By definition of inverse matrix)}\end{aligned}$$

Finally:

$$\begin{aligned}\frac{\partial}{\partial \mathbf{\Sigma}} l(\mu, \mathbf{\Sigma}|\boldsymbol{x}) &= -\frac{n}{2}\frac{\partial}{\partial \mathbf{\Sigma}} \log\det(\mathbf{\Sigma}) - \frac{1}{2}\sum_{i=1}^{n}\frac{\partial}{\partial \mathbf{\Sigma}} \operatorname{tr}\left(\left(\boldsymbol{x}_i - \mu\right)\left(\boldsymbol{x}_i - \mu\right)^{\mathrm{T}} \mathbf{\Sigma}^{-1}\right) \\ &= -\frac{n}{2}\mathbf{\Sigma}^{-T} - \frac{1}{2}\sum_{i=1}^{n}\left(\boldsymbol{x}_i - \mu\right)\left(\boldsymbol{x}_i - \mu\right)^{\mathrm{T}} \mathbf{\Sigma}^{-T}\mathbf{\Sigma}^{-T} \\ &= -\frac{n}{2}\mathbf{\Sigma}^{-T} - \frac{1}{2}\mathbf{\Sigma}^{-T}\mathbf{\Sigma}^{-T}\sum_{i=1}^{n}\left(\boldsymbol{x}_i - \mu\right)\left(\boldsymbol{x}_i - \mu\right)^{\mathrm{T}} \\ &= \frac{1}{2}\mathbf{\Sigma}^{-T}\left(-n - \mathbf{\Sigma}^{-T}\sum_{i=1}^{n}\left(\boldsymbol{x}_i - \mu\right)\left(\boldsymbol{x}_i - \mu\right)^{\mathrm{T}}\right)\end{aligned}$$

We maximize to get $\frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{x}_i - \hat{\mu}\right)\left(\boldsymbol{x}_i - \hat{\mu}\right)^{\mathrm{T}} = \hat{\mathbf{\Sigma}} = \mathbf{S}$ $\square$

Note that the sample covariance matrix is a biased estimator of the true covariance matrix. What happens when $n < p$? Can we still use the sample covariance matrix to estimate the population covariance ?

1.2 Classical Covariance Estimation in High-Dimension

When dealing with high-dimensional data the number of features for each observation is greater than the number of observations i.e. $n < p$.

Proposition 1.3.1: For high-dimensional data, he $p \times p$ sample covariance matrix:

$$\mathbf{S} = \frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{x}_i - \overline{\boldsymbol{x}}\right)\left(\boldsymbol{x}_i - \overline{\boldsymbol{x}}\right)^{T}$$

is singular.

Proof: Without loss of generality $\overline{\boldsymbol{x}} = 0$ we thus get:

$$\mathbf{S} = \frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{x}_i - \overline{\boldsymbol{x}}\right)\left(\boldsymbol{x}_i - \overline{\boldsymbol{x}}\right)^{T} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i\boldsymbol{x}_i^{T}$$

We know that for any vector $\boldsymbol{x} \in \mathbb{R}^p$ with $\boldsymbol{x} \neq 0$, we have $\operatorname{rank}\left(\boldsymbol{x}\boldsymbol{x}^{T}\right) = 1$. So $\operatorname{rank}\left(\boldsymbol{x}_i\boldsymbol{x}_i^{T}\right) = 1$ for all $i = 1, 2, \cdots, n$. Furthermore for any two square matrices $\mathbf{A}$ and $\mathbf{B}$ we have that $\operatorname{rank}\left(\mathbf{A} + \mathbf{B}\right) \leq \operatorname{rank}\left(\mathbf{A}\right) + \operatorname{rank}\left(\mathbf{B}\right)$, So :

$$\begin{aligned}\operatorname{rank}\left(\boldsymbol{x}_1\boldsymbol{x}_1^{T} + \boldsymbol{x}_2\boldsymbol{x}_2^{T}\right) &\leq \operatorname{rank}\left(\boldsymbol{x}_1\boldsymbol{x}_1^{T}\right) + \operatorname{rank}\left(\boldsymbol{x}_2\boldsymbol{x}_2^{T}\right) = 1 + 1 = 2 \\ \operatorname{rank}\left(\left(\boldsymbol{x}_1\boldsymbol{x}_1^{T} + \boldsymbol{x}_2\boldsymbol{x}_2^{T}\right) + \boldsymbol{x}_3\boldsymbol{x}_3^{T}\right) &\leq \operatorname{rank}\left(\boldsymbol{x}_1\boldsymbol{x}_1^{T} + \boldsymbol{x}_2\boldsymbol{x}_2^{T}\right) + \operatorname{rank}\left(\boldsymbol{x}_3\boldsymbol{x}_3^{T}\right) = 2 + 1 = 3 \\ &\vdots \\ \operatorname{rank}\left(\left(\boldsymbol{x}_1\boldsymbol{x}_1^{T} + \boldsymbol{x}_2\boldsymbol{x}_2^{T} + \cdots + \boldsymbol{x}_{n-1}\boldsymbol{x}_{n-1}^{T}\right) + \boldsymbol{x}_n\boldsymbol{x}_n^{T}\right) &= \operatorname{rank}\left(\mathbf{S}\right) \leq n - 1 + 1 = n\end{aligned}$$

So $\text{rank}\,(\mathbf{S}) = n < p$. By the invertible matrix theorem a matrix $\mathbf{S} \in \mathbb{R}^{p\times p}$ has an inverse if and only if $\text{rank}\,(\mathbf{S}) = p$. $\square$

So $\mathbf{S}$ is not invertible i.e. singular thus we cannot compute inverse covariance necessary to compute the density function of the multivariate normal distribution or the precision matrix. Therefore there is a need for new covariance estimators in high-dimensional settings($n < p$).

2 Covariance Estimation with Shrinkage

2.1 Motivation

Take for example a random vector $\mathbf{X} = (X_1, X_2, \cdots, X_p) \sim N(0, I_p)$. We can simulate a sample of fixed size n=50 from such a multivariate normal using the `mvrnorm()` function in R. We can vary $p = 1, \cdots, 150$ to observe the eigenstructure of the sample covariance when $n > p, n = p$, and $n < p$. We do that and take a look at the mminimum eigenvalue. We expect to observe the minimum to get close to 0 (see Figure 1) as p grows since it is an equivalent property for singular matrices to rank $(\mathbf{S}) = n < p$ from the invertible matrix theorem.

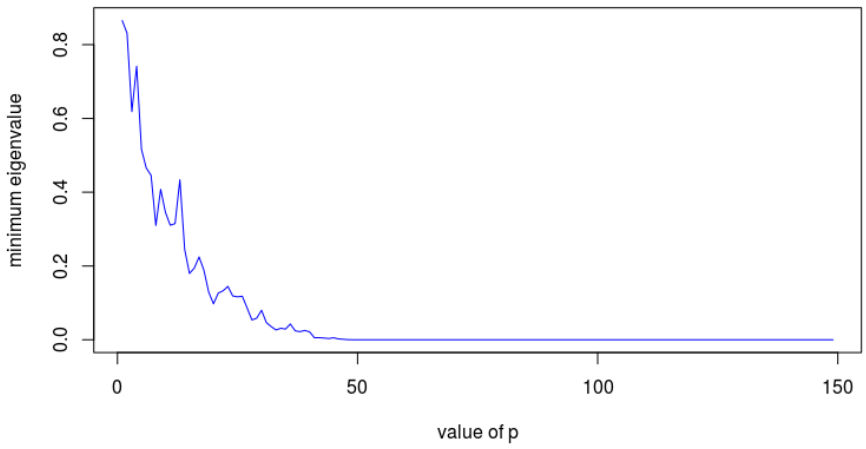

Figure 1: Minimum Eigenvalues of S the p by p sample covariance matrix

Furthermore the sample covariance matrix spread the eigenvalues compared to their true values. For decreasing ordered eigenvalues from $\mathbf{S}$ the first eigenvalues have an upward bias (they are bigger than their population counterpart), and the smallest eigenvalues have a downward bias (smaller than their population ocunterpart) as seen on Figure 2.

So there is a need for an estimator that preserves some of the nice properties of the MLE estimator and also preserves the eigenstructure at least partially. One way to do that is to take an estimator of the covariance based on the sample covariance and that shifts the eigenvalues to a central value so that no eigenvalue gets close to 0. We consider the estimator class of linear combinations of the identity matrix and sample covariance: $\mathbf{S}^* = \rho_1 \mathbf{I}_p + \rho_2 \mathbf{S}$. For $\rho_i \in \mathbb{R}^{>0}$ we

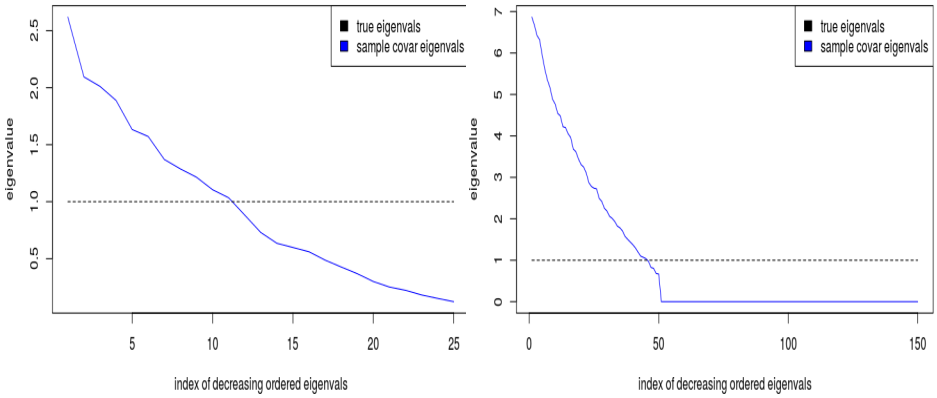

Figure 2: Spread of the ordered eigenvalues of the sample covariance around their true values for different feautres/samplesize ratios: on the left the ratio is $p/n = 0.5$ and 3 on the right.

get:

$$\mathbf{S}^* = \rho_1 \mathbf{I}_p + \rho_2 \mathbf{S}$$

Proposition 2.1.1: $\mathbf{S}^*$ is invertible i.e. none of its eigenvalues is 0.

Proof: By the Spectral Theorem for square normal matrices we get:

$$\mathbf{S}^* = \rho_1 \mathbf{U}\mathbf{I}_p\mathbf{U}^T + \rho_2 \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$$

$\mathbf{U}$ the unitary matrix of p linearly independent eigenvectors, $\mathbf{\Lambda}$ the diagonal matrix of non-negative eigenvalues of the positive semi-definite matrix $\mathbf{S}$.

$$\begin{aligned}
&= \mathbf{U}\left(\rho_1 \mathbf{I}_p \mathbf{U}^T + \rho_2 \mathbf{\Lambda}\mathbf{U}^T\right) \\
&= \mathbf{U}\left(\left(\rho_1 \mathbf{I}_p + \rho_2 \mathbf{\Lambda}\right)\mathbf{U}^T\right) \\
&= \mathbf{U}\left(\left(\rho_1 \mathbf{I}_p + \rho_2 \mathbf{\Lambda}\right)\mathbf{U}^T\right) \\
&= \mathbf{U}\,\mathrm{diag}(\rho_1 + \rho_2\lambda_1, \cdots, \rho_1 + \rho_2\lambda_p)\mathbf{U}^T
\end{aligned}$$

So we get:

$$\mathbf{S}^*\mathbf{u}_i = (\rho_1 + \rho_2\lambda_i)\mathbf{u}_i$$

Thus our new estimator $\mathbf{S}^* = \mathbf{U}\,\mathrm{diag}(\rho_1 + \rho_2\lambda_1 \cdots \rho_1 + \rho_2\lambda_p)\mathbf{U}^T$ is square and has non-zero eigenvalues thus it is invertible. □

In addition to being an invertible estimator for the covariance matrix we can choose optimal ρ_i such that, $\mathbf{S}^*$ has a better (smaller) Mean Squared Error (MSE) with the true covariance $\mathbf{\Sigma}$ than the sample covariance $\mathbf{S}$.

2.2 Ledoit-Wolf Shrinkage Estimator

The Ledoit-Wolf Shrinkage Estimator from their article of 2004 [**2**] is one that belongs to the class $\mathbf{S}^*$ and minimizes $\text{MSE}(\mathbf{S}^*) = \text{E}\left(\|\mathbf{S}^* - \mathbf{\Sigma}\|^2\right)$ where $\|A\| = \sqrt{\text{tr}(AA^T)/p}$, $< A, B >$ the Frobenius inner product and $\|A - \alpha B\| = \|A\|^2 + \alpha^2\|B\|^2 - 2\alpha < A, B >$. Let us look at the following optimization problem:

$$\min_{\rho_1,\rho_2} \text{E}\left(\|\mathbf{S}^* - \mathbf{\Sigma}\|^2\right)$$

subject to:

$$\begin{aligned}\mathbf{S}^* &= \rho_1\mathbf{I}_p + \rho_2\mathbf{S} \\ &= \rho v\mathbf{I}_p + (1-\rho)\mathbf{S} \text{ with the change of variable } \rho_1 = \rho v, \rho_2 = (1-\rho)\end{aligned}$$

Theorem 2.1: An optimal solution of the above problem is $\mathbf{\Sigma}^* = \frac{\beta^2}{\delta^2}\mu\mathbf{I}_p + \frac{\alpha^2}{\delta^2}\mathbf{S}$ and $\text{E}\left(\|\mathbf{\Sigma}^* - \mathbf{\Sigma}\|^2\right) = \frac{\alpha^2\beta^2}{\delta^2}$ with $\mu =< \mathbf{\Sigma}, \mathbf{I} >$, $\alpha^2 = \|\mathbf{\Sigma} - \mu\mathbf{I}\|^2$, $\beta^2 = \text{E}\left(\|\mathbf{S} - \mathbf{\Sigma}\|^2\right)$, and $\delta^2 = \text{E}\left(\|\mathbf{S} - \mu\mathbf{I}\|^2\right)$.

Proof: Using the variance bias decomposition of the MSE we get:

$$\begin{aligned}\text{E}\left(\|\mathbf{S}^* - \mathbf{\Sigma}\|^2\right) &= \text{E}\left(\|\mathbf{S}^* - \text{E}(\mathbf{S}^*)\|^2\right) + \|\text{E}(\mathbf{S}^*) - \mathbf{\Sigma}\|^2 \\ &= \text{E}\left(\|\mathbf{S}^* - \rho v\mathbf{I} - (1-\rho)\,\text{E}(\mathbf{S})\|^2\right) + \|\ \rho v\mathbf{I} + (1-\rho)\,\text{E}(\mathbf{S}) - \mathbf{\Sigma}\|^2\end{aligned}$$

Where we used a change of variable. We note $\text{E}(\mathbf{S}) = \mathbf{\Sigma}$.

$$\begin{aligned}&= \text{E}\left(\|(1-\rho)(\mathbf{S} - \mathbf{\Sigma})\|^2\right) + \|\rho v\mathbf{I} - \rho\mathbf{\Sigma}\|^2 \\ &= (1-\rho)^2\,\text{E}\left(\|\mathbf{S} - \mathbf{\Sigma}\|^2\right) + \rho^2\,\|\mathbf{\Sigma} - \mathbf{v}\mathbf{I}\|^2\end{aligned}$$

We can thus write our optimization problem as follows:

$$\min_{\rho,v}(1-\rho)^2\,\text{E}\left(\|\mathbf{S} - \mathbf{\Sigma}\|^2\right) + \rho^2\,\|\mathbf{\Sigma} - v\mathbf{I}\|^2$$

subject to:

$$\mathbf{S}^* = \rho v\mathbf{I} + (1-\rho)\mathbf{S}$$

We look for $\min_v \|\mathbf{\Sigma} - v\mathbf{I}\|^2 = \min_v \|\mathbf{\Sigma}\|^2 + v^2\|\mathbf{I}\|^2 - 2v < \mathbf{\Sigma}, \mathbf{I} >$. The first order condition yields $2v - 2 < \mathbf{\Sigma}, \mathbf{I} >= 2v - 2\mu = 0$. So optimal v is μ. We now look at $\min_\rho (1-\rho)^2 \, \mathrm{E}\left(\|\mathbf{S} - \mathbf{\Sigma}\|^2\right) + \rho^2 \|\mathbf{\Sigma} - \mu\mathbf{I}\|^2 = \min_\rho \rho^2\alpha^2 + (1-\rho)^2\beta^2$. The first order condition w.r.t. ρ yields $\rho^2\alpha^2 + (1-\rho)^2\beta^2 = 0$ so $\rho = \beta^2/(\alpha^2+\beta^2) = \beta^2/\delta^2$. And $(1-\rho) = \alpha^2/\delta^2$. Plugging those optimal values in the constraint we get $\mathbf{\Sigma}^* = \frac{\beta^2}{\delta^2} m\mathbf{I}_p + \frac{\alpha^2}{\delta^2}\mathbf{S}$. And plugging the optimal values in the objective function gives $(1-\rho)^2 \, \mathrm{E}\left(\|\mathbf{S} - \mathbf{\Sigma}\|^2\right) + \rho^2 \|\mathbf{\Sigma} - v\mathbf{I}\|^2 = \frac{\alpha^2\beta^2}{\delta^2} \square$

We can see that the parameters of the new estimator $\mathbf{\Sigma}^*$ depends on the true covariance $\mathbf{\Sigma}$. We thus have to find consistent estimators for those parameters i.e. estimators that preserve their structures in general asympsotics when $\frac{p}{n} \leq constant$ for $n, p \longrightarrow \infty$. Under certain assumptions [**2**], Ledoit and Wolf found that:

(1) $m =< \mathbf{S}, \mathbf{I} >$ is a consistent estimator for μ
(2) $d^2 = \|\mathbf{S} - m\mathbf{I}\|$ is a consistent estimator for δ^2
(3) $b^2 = \min(d^2, \bar{b}^2)$ is a consistent estimator for β^2 where $\bar{b}^2 = \frac{1}{n^2}\sum_{k=1}^{n} \left\|\mathbf{X}_k^T\mathbf{X}_k - \mathbf{S}\right\|$ and $\mathbf{X}_k$ the k-th row of the $n \times p$ data matrix $\mathbf{X}_k$
We can observe from R simulations that the ordered eigenvalues (in red) of the Ledoit-Wolf estimator are shrunk towards their true values as seen from Figure 3.

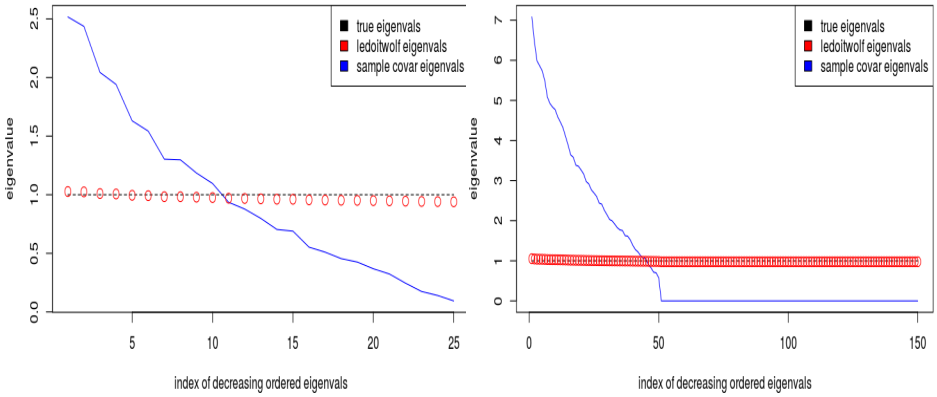

Figure 3: Spread of the ordered eigenvalues of the sample covariance and of the Ledoit-Wolf estimator around their true values for different feautres/samplesize ratios.

3 Estimation with Thresholding

3.1 Hard Thresholding

Unlike shrinkage estimators, thresholding is a method that acts on all the entries of the sample covariance matrix and thus also on its eigenvectors. This method is motivated by the sparsity $\mathbf{S}$ in high dimensional settings. Indeed, the more random the variables studied are (large p), the more likely it is to have small or zero covariances. Hard thresholding as introduced by P. J. Bickel and E. Levina in 2008 [**4**] relies on that additional sparsity information to map entries of $\mathbf{S}$ that are below the threshold to 0. For a sample covariance matrix $\mathbf{S} = (s_{ij})$ and a positive threshold t we define the thresholding operator by:

$$T_t(\mathbf{S}) = [s_{ij}\mathbf{1}(|s_{ij}| \geq t)]$$

We say that $\mathbf{S}$ has been thresholded at t. Something to note is that thresholding operator preserve symmetry (easy to see) but does not preserve positive definiteness (some eigenvalues might be zero). We can restrict this method to certain classes of covariance matrices to tackle the invertibility problem of the previous section. We now present some useful classes of covariance matrices.

Proposition 3.1.1: If $\|T_t(\mathbf{S}) - T_0(\mathbf{S})\| \leq \varepsilon \quad$ and $\quad \lambda_{\min}(\mathbf{S}) > \varepsilon$ the minimum eigenvalue of $\mathbf{S}$, then $T_t(\mathbf{S})$ is necessarily positive-definite.

P. J. Bickel and E. Levina defined a class of covariance matrices that is an implicit definition of sparsity and also guarantees that the thresholding operator converges uniformly to the real covariance in the operator norm if $\frac{\log p}{n} \longrightarrow 0$:

$$\mathcal{U}_\tau(q, s_0(p), C) = \left\{ \Sigma : \sigma_{ii} \leq C, \sum_{j=1}^{p} |\sigma_{ij}|^q \leq s_0(p), \text{ for all } i \right\}$$

for $0 \leq q \leq 1$, C a constant, and $s_0(p)$ the number of non-zero entries in Σ. When $q = 0$ we get:

$$\mathcal{U}_\tau(0, s_0(p), C) = \left\{ \Sigma : \sigma_{ii} \leq C, \sum_{j=1}^{p} 1(\sigma_{ij} \neq 0) \leq s_0(p) \text{ for all } i \right\}$$

$s_0(p)$ can be thought of a parameter depending on p and controlling the sparsity of the class. Indeed the summation $\sum_{j=1}^{p} 1(\sigma_{ij} \neq 0)$ is just counting the non-zero entries. We can also incorporate proposition 3.1.1 to have a class of invertible matrices with all the above properties:

$$\mathcal{U}_\tau(q, s_0(p), C, \varepsilon) = \{\Sigma : \Sigma \in \mathcal{U}_\tau(q, s_0(p), C) \quad \text{and } \lambda_{\min}(\Sigma) \geq \varepsilon > 0\}$$

P.J.Bickel and E.Levina's main result: Under certain conditions. For $\Sigma_1 \in \mathcal{U}_\tau\left(q, s_0(p), C\right)$, and $\hat{\Sigma_1}$ the sample covariance estimator,

$$\left\| T_{t_n}(\hat{\Sigma_1}) - \Sigma_1 \right\| \text{converges uniformly in probability.}$$

And for $\Sigma_2 \in \mathcal{U}_\tau\left(q, s_0(p), C, \varepsilon\right)$, and $\hat{\Sigma_2}$ the sample covariance estimator,

$$\left\| \left(T_{t_n}(\hat{\Sigma_2}) \right)^{-1} - \Sigma_2^{-1} \right\| \text{converges uniformly in probability.}$$

The main computational challenge is the choice of the threshold t.
Method 3.1.2: Split the sample randomly into two pieces of size n_1 and n_2 with $n_1 = n\left(1 - \frac{1}{\log n}\right), n_2 = \frac{n}{\log n}$ and repeat this N times. Let $\hat{\Sigma}_{1,\nu}, \hat{\Sigma}_{2,\nu}$ be the empirical covariance matrices based on the n_1 and n_2 observations, respectively, from the ν th split. Form

$$\hat{R}(t) = \frac{1}{N} \sum_{\nu=1}^{N} \left\| s_t\left(\hat{\Sigma}_{1,\nu}\right) - \hat{\Sigma}_{2,\nu} \right\|_F^2$$

And choose t that minimizes $\hat{R}(t)$. This method is called V-fold cross validation with $V = 2$. Hard Thresholding is a special case of Generalized Thresholding.

3.2 Generalized Thresholding

Rothman, Levina and Zhu generalized the thresholding [**5**] approach to covariance estimation to a whole class of estimators based on elementwise shrinkage and thresholding. For any $\lambda \geq 0$, the generalized thresholding operator is a function $s_\lambda : \mathbb{R} \to \mathbb{R}$ satisfying the following conditions for all $z \in \mathbb{R}$:

(i) $|s_\lambda(z)| \leq |z|$ (Shrinkage)
(ii) $s_\lambda(z) = 0$ for $|z| \leq \lambda$ (Thresholding)
(iii) $|s_\lambda(z) - z| \leq \lambda$ (Amount of shrinkage)

The generalized thresholded covariance estimator is obtained by applying the generalized thresholding operator $s_\lambda(\cdot)$ to each entry of the sample covariance matrix $\mathbf{S}$:

$$s_\lambda(\mathbf{S}) = \left(s_\lambda\left(s_{ij}\right)\right)$$

The generalized thresholding estimator has similar theoretical properties to the hard thresholding estimator in addition to mapping true 0 to 0 with probability tending to 1. The hard thresholding operator can be rewritten as a special case of generalized thresholding:

$$s_\lambda^H(z) = z1(|z| > \lambda)$$

It is easily seen that $s_\lambda^H(.)$ satisfies conditions (i) to (iii). Another estimator from this class that we will not study theoretically is soft thresholding:

$$s_\lambda^s(z) = \text{sign}(z)(|z| - \lambda)_+$$

The soft-thresholding operator s_λ^S obviously satisfies conditions (i) and (ii). To check (iii), note that $\left|s_\lambda^S(z) - z\right| = |z|$ when $|z| \leq \lambda$, and $\left|s_\lambda^S(z) - z\right| = \lambda$ when $|z| > \lambda$.

4 Performance of Estimators

4.1 Simulations

In this section we use the same methdology as in [**4**] and look at 3 different types of true covariance matrix and generate some performance metrics for the estimators studied so far: Sample Covariance, Ledoit-Wolf, Hard Threshold, Soft Threshold. For the thresholding estimators we got the threshold t using method 3.1.2 with 10 iterations (see the functions `getThresholdValue` , `objectiveFunctionRt` and `getThresholdValueSoft` , `objectiveFunctionRtforSoft` from the R code in Appendix). For each of the covariance matrices types we have looked at the estimators for fixed $n = 100$ and $p = 50, 100, 300$. We look at the average of the performance indicators generated 100 times for each p (see the function `BIGTABLE` and `someMetrics`):

(i) The absolute value of the cosine of the angle between the first PC of the true covariance and the first PC of the estimator
(ii) The Mean Square Error (MSE) between the estimator and the true covariance.
(iii) The Frobenius norm of the difference: $\|\hat{\Sigma}(\mathbf{X}) - \Sigma\|_F$
(iv) The 2-norm of the difference: $\|\hat{\Sigma}(\mathbf{X}) - \Sigma\|_2$
(v) The minimum eigenvalue λ_{min}

The 3 true covariance matrices studied are AR(1) where $\sigma_{ij} = \rho^{|i-j|}$, for $\rho = 0.7$. It is a covariance matrix for which the entries (the covariances) are getting smaller further away from the main diagonal. MA(1) where $\sigma_{ij} = \rho 1(|i - j| = 1) + 1(i = j)$, for $\rho = 0.3$. And finally the Identity matrix of size p.

Table 1: **p=50, true Covariance matrix is AR(1)**

	Sample Covariance	Ledoit-Wolf	Hard Thresholding	Soft Thresholding
Abs val of cos of the angle between 1st PCs	0.005019	0.005289	0.003697	0.004644
MSE	0.010428	0.008235	0.009086	0.006683
dif in Frobenius	5.096793	4.532187	4.736410	4.071148
dif in 2-norm	2.552717	2.232044	2.246731	2.221747
min eigenvalue	0.031138	0.247614	-0.258205	0.018553

Table 2: **p=100, true Covariance matrix is AR(1)**

	Sample Covariance	Ledoit-Wolf	Hard Thresholding	Soft Thresholding
Abs val of cos of the angle between 1st PCs	0.003334	0.003290	0.001594	0.003413
MSE	0.010343	0.006735	0.006267	0.005953
dif in Frobenius	10.164014	8.204516	7.856650	7.700488
dif in 2-norm	4.105875	3.079444	2.841175	4.039031
min eigenvalue	0.000020	0.350675	-0.337428	-0.078747

Table 3: **p=300, true Covariance matrix is AR(1)**

	Sample Covariance	Ledoit-Wolf	Hard Thresholding	Soft Thresholding
Abs val of cos of the angle between 1st PCs	0.000401	0.000473	0.000170	0.000358
MSE	0.010055	0.003906	0.003255	0.005307
dif in Frobenius	30.078306	18.748840	16.964563	21.848217
dif in 2-norm	8.785896	4.137653	3.702075	11.946580
min eigenvalue	-0.000000	0.604998	-0.442439	-0.365509

Table 4: **p=50, true Covariance matrix is MA(1)**

	Sample Covariance	Ledoit-Wolf	Hard Thresholding	Soft Thresholding
Abs val of cos of the angle between 1st PCs	0.001848	0.002281	0.001975	0.002314
MSE	0.010153	0.002639	0.003953	0.005493
dif in Frobenius	5.035188	2.568023	3.141641	3.700767
dif in 2-norm	1.864053	0.654037	0.812760	2.072114
min eigenvalue	0.074124	0.758666	0.535174	0.168914

Table 5: **p=100, true Covariance matrix is MA(1)**

	Sample Covariance	Ledoit-Wolf	Hard Thresholding	Soft Thresholding
Abs val of cos of the angle between 1st PCs	0.000600	0.000603	0.000804	0.000827
MSE	0.010134	0.001520	0.002062	0.005270
dif in Frobenius	10.065130	3.898526	4.537766	7.256831
dif in 2-norm	3.009317	0.661281	0.934344	4.070203
min eigenvalue	0.000047	0.842285	0.373802	0.009428

Table 6: **p=300, true Covariance matrix is MA1**

	Sample Covariance	Ledoit-Wolf	Hard Thresholding	Soft Thresholding
Abs val of cos of the angle between 1st PCs	0.000083	0.000136	0.000160	0.000101
MSE	0.010023	0.000566	0.000708	0.005076
dif in Frobenius	30.033322	7.137520	7.971239	21.372853
dif in 2-norm	6.584470	0.647186	1.038926	12.041471
min eigenvalue	-0.000000	0.934417	0.275305	-0.375642

Table 7: **p=50, true Covariance matrix is Identity**

	Sample Covariance	Ledoit-Wolf	Hard Thresholding	Soft Thresholding
Abs val of cos of the angle between 1st PCs	0.002120	0.002282	0.001365	0.002089
MSE	0.010181	0.000016	0.000451	0.005282
dif in Frobenius	5.043401	0.168057	1.042941	3.631523
dif in 2-norm	1.766997	0.042501	0.434759	2.083759
min eigenvalue	0.102616	0.983395	0.643057	0.307252

Table 8: **p=100, true Covariance matrix is Identity**

	Sample Covariance	Ledoit-Wolf	Hard Thresholding	Soft Thresholding
Abs val of cos of the angle between 1st PCs	0.000067	0.000484	0.000278	0.000292
MSE	0.010075	0.000005	0.000284	0.005128
dif in Frobenius	10.035988	0.183983	1.626239	7.159076
dif in 2-norm	2.834451	0.039160	0.579165	4.073625
min eigenvalue	0.000049	0.988855	0.481735	0.100138

Table 9: **p=300, true Covariance matrix is Identity**

	Sample Covariance	Ledoit-Wolf	Hard Thresholding	Soft Thresholding
Abs val of cos of the angle between 1st PCs	0.000000	0.000025	0.000033	0.000085
MSE	0.010015	0.000002	0.000112	0.005035
dif in Frobenius	30.021150	0.343032	3.034038	21.286871
dif in 2-norm	6.252391	0.063518	0.797053	12.046904
min eigenvalue	-0.000000	0.989064	0.242565	-0.345943

4.2 Observations

We can see from Table 1 that $\lambda_{min} < 0$ for the hard thresholding i.e. positive-definiteness is not guaranted. From Tables 3, 6, and 9, we observe that in high dimensions the sample covariance has indeed a minimum eigenvalue equal to 0. We can see from Table 6 and Table 9 that in high dimensions, the Ledoit-Wolf is indeed the estimator with smallest MSE (almost the case for AR1 as seen from Table 3). For Principal Component Analysis in high dimensions we observe that for MA1 and the identity matrix, the sample covariance performs better (at least for the 1st PC). But for AR1 it is better to use the hard thresholding estimator as seen from Table 3 (this result coincides with the one in [**4**]). We are happy to see that the simulations mostly agree with the theory. Those tables can serve as references to choose what estimator to use with respect to: the p/n ratio, the task to perform or the norm to minimize, and the type of true covariance matrix if known. We can of course use those estimators to study real life high-dimensional data sets. For example the result of an experiment on gene expression among prostate cancer subjects. The data set is $n = 102$ by $p = 6033$ (see [**9**]) . Computing the covariance matrix for this data set can help identify correlations between different gene that could potentially be responsible for prostate cancer

5 Discussion and Conclusion

In this project we have presented the problem of covariance matrix estimation: estimating covariances between variables from observed data. It is a problem of increasing importance as nowadays data is getting high-dimensional. We have identified shrinkage estimators as positive-definite estimators and found parameters minimizing the MSE which results in the Ledoit-Wolf estimator. We have studied thresholding methods, which are estimators acting on all the values of the sample covariance, and presented a method to pick a threshold t minimising the squared Frobenius norm using 2-fold cross-validation. We have identified hard thresholding as a special case of the wider class of generalized threshold estimators combining thresholding and shrinking and briefly introduced soft thresholding as an other special case. We computed those estimators from data generated from different types of Covariance matrices with different sizes ratios p/n. We had a first attempt at providing a general framework for covariance estimation in high dimensions. In the future it would be useful to review more estimators and continue the work of Section 4 to identify best performing estimator for each p/n ratio and for each true covariance type. I would personally like to further study the theoretical property of the thresholding estimators, and study other estimators from a very theoretical point of view.

As a learning experience, the project has taught me independent study, R programming (all the R code in the appendix has been written in pure R and is

my own work), some Matrix Analysis and a lot of Statistics. The most challenging aspects of this project were definitely to choose what topics to include and proving results that seem obvious for researchers but that are not to an undergrad like me. But I now feel confident explaining the topic of high-dimensional covariance estimation to a fellow undergrad. I have developed a genuine interest for High-Dimensional Estimation. And this has motivated me to study maths for as long as I can.

6 Appendix

6.1 Matrix Analysis Proofs

$$\frac{\partial}{\partial a_{ij}} \operatorname{tr}(\mathbf{AB}) = \frac{\partial}{\partial a_{ij}} \sum_k \sum_l a_{ij} b_{ji} = b_{ji}$$

So

$$\frac{\partial}{\partial \mathbf{A}} \operatorname{tr}(\mathbf{BA}) = \mathbf{B}^T$$

Note that $\frac{\partial}{\partial a_{ij}} \log(\det(\mathbf{A})) = \frac{1}{\det(\mathbf{A})} \frac{\partial}{\partial a_{ij}} \det(\mathbf{A})$. From Matrix Analysis we know that:

$$\det(\mathbf{A}) = \sum_{j=1}^{n} a_{ij} (-1)^{i+j} \det(\mathbf{A}_{ij})$$

where $(-1)^{i+j} \det(\mathbf{A}_{ij})$ is a cofactor of $\mathbf{A}$. So:

$$\begin{aligned} \frac{\partial}{\partial a_{ij}} \det(\mathbf{A}) &= (-1)^{i+j} \det(\mathbf{A}_{ij}) \\ &= \operatorname{adj}(\mathbf{A})_{ij}^T \end{aligned}$$

furthermore $\mathbf{A}_{ij}^{-1} = \frac{\operatorname{adj}(\mathbf{A})_{ij}}{\det(\mathbf{A})}$. So:

$$\begin{aligned} \frac{\partial}{\partial \sigma_{ij}^2} \log\det(\mathbf{\Sigma}) &= \frac{1}{\det(\mathbf{\Sigma})} \frac{\partial}{\partial \sigma_{ij}^2} \det(\mathbf{\Sigma}) \\ &= \frac{\operatorname{adj}(\mathbf{\Sigma})_{ij}^T}{\det(\mathbf{\Sigma})} \\ &= \mathbf{\Sigma}_{ij}^{-T} \end{aligned}$$

6.2 R code

```
# simple function to generate an AR1 matrix of size pxp with given correlation ##
AR1matrix=function(correlation,p){
  covariance=matrix(rep(0,p),ncol = p,nrow = p)
  for(i in 1:p){
    for (j in 1:p) {
      covariance[i,j]=correlation^(abs(i-j))
    }
  }
  return(covariance)
}
# simple function to generate an MA1 matrix of size pxp with given correlation#
MA1matrix=function(correlation,p){
  covariance=matrix(rep(0,p),ncol = p,nrow = p)
  for(i in 1:p){
    for (j in 1:p) {
      covariance[i,j]=correlation*as.integer(abs(i-j)==1)+as.integer(i==j)
    }
  }
  return(covariance)
}

# generate n observations from a multivariate normal ##
getObservations=function(n,covariance){
  p=dim(covariance)[1]
  mu=rep(0,p)
  sigma=covariance
  observations=mvrnorm(n,mu,sigma)
  return(observations)
}
## compute the hard thresholding covariance estimator with threshold t #
HardThresholdingCovar=function(SampleCovariance,t){
  thresholdedCovar=SampleCovariance*(abs(SampleCovariance)>=t)
  return(thresholdedCovar)
}
### compute the soft thresholding covariance estimator with threshold t ##
SoftThresholdingCovar=function(SampleCovariance,t){
  p=dim(SampleCovariance)[1]
  SoftThresholdedCovar=matrix(0,p,p)
  for (i in 1:p) {
    for (j in 1:p) {
      SoftThresholdedCovar[i,j]=sign(SampleCovariance[i,j])*
        (max(abs(SampleCovariance[i,j])-t,0))
    }
  }
  return(SoftThresholdedCovar)
}

# the objective function to estime the the hard threshold t from method 3.1.2#
```

```
objectiveFunctionRt=function(t,observationsFromMVN,N){
  n=length(observationsFromMVN[,1])
  p=length(observationsFromMVN[1,])
  Rt=0
  indexes=1:n
  n1=round(n-(n/log(n)))
  n2=n-n1
  for(v in 1:N){
    indextodelete=sample(indexes,n1,replace = TRUE)
    ##### get the training set and validation set ######
    observationsTRAIN=observationsFromMVN[indextodelete,]
    observationsTEST=observationsFromMVN[-indextodelete,]
    Sigma1v=SampleCovariance(observationsTRAIN,p,n1)
    Sigma2v=SampleCovariance(observationsTEST,p,n2)
    Rt=Rt+norm(HardThresholdingCovar(Sigma1v,t)-Sigma2v,"f")^2
  }
  return((1/N)*Rt)
}
# the objective function to estime the the soft threshold t from method 3.1.2#
objectiveFunctionRtforSoft=function(t,observationsFromMVN,N){
  n=length(observationsFromMVN[,1])
  p=length(observationsFromMVN[1,])
  Rt=0
  indexes=1:n
  n1=round(n-(n/log(n)))
  n2=n-n1
  for(v in 1:N){
    indextodelete=sample(indexes,n1,replace = TRUE)
    observationsTRAIN=observationsFromMVN[indextodelete,]
    observationsTEST=observationsFromMVN[-indextodelete,]
    Sigma1v=SampleCovariance(observationsTRAIN,p,n1)
    Sigma2v=SampleCovariance(observationsTEST,p,n2)
    Rt=Rt+norm(SoftThresholdingCovar(Sigma1v,t)-Sigma2v,"f")^2
  }
  return((1/N)*Rt)
}

#### function to perform the minimization of method 3.1.2 ####
getThresholdValue=function(observationsFromMVN,N,p){
  o=optimize(objectiveFunctionRt,observationsFromMVN,N,interval = c(0,1))
  t=o$minimum
  return(t)
}
getThresholdValueSoft=function(observationsFromMVN,N,p){
  o=optimize(objectiveFunctionRtforSoft,observationsFromMVN,N,interval = c(0,1))
  t=o$minimum
  return(t)
}
# compute a vector of metrics of an estimator with respect to a true covariance ##
someMetrics=function(estimatorcovariance,truecovar){
```

```
  AbsCosAngleBetweenfirstPC=abs(prcomp(estimatorcovariance)$x[1,]%*%
                                  prcomp(truecovar)$x[1,]/
                                  (Norm(truecovar,p=2)*
                                     Norm(estimatorcovariance,p=2)))
  MSE=MSE(estimatorcovariance,truecovar)
  froebdif=norm(estimatorcovariance-truecovar,"F")
  specdifference=norm(estimatorcovariance-truecovar,"2")
  minEV=minimumEigenValue(estimatorcovariance)
  return(c(AbsCosAngleBetweenfirstPC,MSE,froebdif,specdifference,minEV))
}

# generate a table of metrics with respect to 4 estimators and 1 true covariance #
BIGTABLE=function(truecovar){
  finallw=finalht=finalst=finalsc=rep(0,5)
  for(i in 1:100){
    observations=getObservations(100,truecovar)
    p=length(observations[1,])
    n=length(observations[,1])
    samplecov=SampleCovariance(observations,p,n)
    lw=ledoitwolf(samplecov,observations,n)
    ht=HardThresholdingCovar(samplecov,getThresholdValue(observations,10,p))
    st=SoftThresholdingCovar(samplecov,getThresholdValueSoft(observations,10,p))
    finalsc=finalsc+someMetrics(samplecov,truecovar)
    finallw=finallw+someMetrics(lw,truecovar)
    finalht=finalht+someMetrics(ht,truecovar)
    finalst=finalst+someMetrics(st,truecovar)
  }
  df=data.frame(0.01*finalsc,0.01*finallw,0.01*finalht,0.01*finalst)
  names(df)=c("Sample␣Covariance","Ledoit-Wolf","Hard␣Thresholding",
              "Soft␣Thresholding")
  return(df)
}



##### get the minimum eigenvalue of a matrix #####
minimumEigenValue=function(matrix){
  decomposition=eigen(matrix)
  eigenvalues=decomposition$values
  return(min(eigenvalues))}
##### compute the sample covariance given some observations #####
SampleCovariance=function(variables,p,n){
  covar = rep(0,p) %o% rep(0,p)
  for(j in 1:n){covar = covar + (variables[j,] %o% variables[j,])}
  return((1/(n))*covar)}
# compute the Ledoit-Wolf estimator given a sample covar and some observations#
ledoitwolf=function(samplecovariance,variables,n){
  p=dim(samplecovariance)[1]
  identity=diag(1,p,p)
  m=tr(t(samplecovariance))/p
```

```
  dsquared=tr((samplecovariance-m*identity)%*%t(samplecovariance-m*identity))/p
  bbarsquared=0
  for(k in 1:n){
    bbarsquared=bbarsquared+tr(((variables[k,]%o%variables[k,])-
                                  samplecovariance)%*%
                                 t(variables[k,]%o%variables[k,]-
                                     samplecovariance))/p
  }
  bbarsquared=1/(n^2)*bbarsquared
  bsquared=min(c(dsquared,bbarsquared))
  asquared=dsquared-bsquared
  LWestimator=((bsquared/dsquared)*m*identity)+(asquared/dsquared*samplecovariance)
  return(LWestimator)
}
##### compute some graphs ######
getgraphMinEV=function(samplesize,maxfeatures){
  n=samplesize
  q=maxfeatures
  x_=y_=rep(0,q)
  for(p in 2:q){
    mu=rep(0,p)
    sigma=diag(x=1,p,p)
    variables=mvrnorm(n,mu,sigma)
    samplecovar=SampleCovariance(variables,p,n)
    lw=ledoitwolf(samplecovar,variables,n)
    x_[p]=minimumEigenValue(lw)
    y_[p]=minimumEigenValue(samplecovar)
  }
  plot(y_[2:q],
       xlab="value␣of␣p",
       ylab="minimum␣eigenvalue",
       type="l",
       col="blue")
  lines(x_[2:q], col="red")
  legend("right",
         c("min␣eigenvals␣for␣ledoitwolf","min␣eigenvals␣for␣samplecov"),
         fill=c("red","blue")
  )
}
getgraphEVspread=function(samplesize,numberoffeatures){
  n=samplesize
  p=numberoffeatures
  mu=rep(0,p)
  sigma=diag(x=1,p,p)
  variables=mvrnorm(n,mu,sigma)
  samplecovar=SampleCovariance(variables,p,n)
  lw=ledoitwolf(samplecovar,variables,n)
  eigenvaluesLedoit=sort(eigen(lw)$values,decreasing = TRUE)
  eigenvaluesSampleCov=sort(eigen(samplecovar)$values,decreasing = TRUE)
  trueeigenvalues=sort(eigen(sigma)$values,decreasing = TRUE)
```

```
  plot(eigenvaluesSampleCov,xlab = "index␣of␣decreasing␣ordered␣eigenvals",
       ylab="eigenvalue",type="l",col="blue")
  lines(trueeigenvalues,lty=20)
  'points(eigenvaluesLedoit,col="red",cex=1.3)'
  legend("topright",c("true␣eigenvals","ledoit␣wolf␣eigenvals",
                      "sample␣covar␣eigenvals"),
         fill=c("black","red","blue"))
}
```

References

[1] Olivier Ledoit and Michael Wolf. *A well-conditioned estimator for large-dimensional covariance matrices*. Journal of Multivariate Analysis 88 365–411, 2004.

[2] Charles Stein. *Some problems in Multivariate Analysis Part 1*. Technical report number 6 for the Office of Naval Research, 1956

[3] Mohsen Pourahmadi. *High-Dimensional Covariance Estimation*. Wiley Series in Probability and Statistics, 2013.

[4] Peter J. Bickel and Elizaveta Levina. *Covariance Regularization By Thresholding*. The Annals of Statistics, 2008

[5] Adam J. Rothman, Elizaveta Levina, and Ji Zhu. *Generalized Thresholding of Large Covariance Matrix*. Journal of the American Statistical Association, 2009

[6] Korbinian Strimmer. *MATH38161 notes*. University of Manchester, 2019.

[7] Jianxin Pan. *MATH38001 notes*. University of Manchester, 2019.

[8] Yixin Fang, Binhuan Wang, Yang Feng *Tuning Parameter Selection in Regularized Estimations of Large Covariance Matrices*. Journal of Statistical Computation and Simulation, 2016

[9] Bradley Efron *Large-Scale Inference: Empirical Bayes Methods for Estimation, Testing and Prediction*. Stanford, 2010