Stony Brook University

Structure Breaks in PCA with Applications in Finance

A Dissertation presented

by

Yuzhou Song

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Applied Mathematics and Statistics

(Statistics)

Stony Brook University

May 2014

Stony Brook University

The Graduate School

Yuzhou Song

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation

Haipeng Xing - Dissertation Advisor
Associate Professor, Applied Mathematics and Statistics

Wei Zhu - Chairperson of Defense
Deputy Chair, Professor, Applied Mathematics and Statistics

Song Wu
Assistant Professor, Applied Mathematics and Statistics

Jinfeng Xu
Assistant Professor, Division of Biostatistics, Population Health,
School of Medicine, New York University

This dissertation is accepted by the Graduate School

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

Structure Breaks in PCA with Applications in Finance

by

Yuzhou Song

Doctor of Philosophy

in

Applied Mathematics and Statistics

(Statistics)

Stony Brook University

2014

Change-point stands for the times of discontinuities in a time series that can be induced from changes in distribution. Change-points widely exist in the time series data in the real world, such as the field of climate and finance. A bunch of models have been developed for change-point detection, however, not many of them focus on the covariance matrix. In this dissertation research, the goal is to find a model to detect change-points of covariance matrix. The difficulties originally come from the dimensionality. In multiple dimensional space, covariance matrix acts as the same role as that of variance in one dimensional space. However, situation gets more complicated as dimension getting higher. Much more noise exists in multiple dimensional space than that in one dimensional space. We expect the model is capable to filter the noise as much as possible, in the meanwhile, it reserves enough information for parameter change. Besides, we have to search a good tool to measure the change of a matrix, which is not as simple as that in one dimensional space. In this dissertation, motivated by the spirit of principal component analysis (PCA), we propose the eigen-structure to measure the change of matrix. PCA will be briefly reviewed in the first chapter. The model used for detecting change-point is proposed by Lai and

Xing (2011). The model is a Bayesian model relies on an assumption of exponential family resulting in a closed form for the final estimated parameter. We also present the derivation of explicit formulas for a special case: the data is normally distributed, for both one and multiple dimensional cases. The explicit formulas contributes to the simplicity of the Bayes model. For the purpose of improving calculation speed, BCMIX, an approximated algorithm, as well as several algorithms for eigen-decomposition are introduced in the beginning of the simulation part. The results of simulation will be shown afterward. At last, the model is applied to several real data sets with satisfactory results obtained.

To my dear parents and grandma, with all my love

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my advisor, Professor Haipeng Xing, for a very interesting topic suggested, for a professional way of doing research provided and for generous support on my life.

I would like to thank Professor Wei Zhu and Professor Song Wu, for their encouragements during my hardest time, for being my defense committee members and for great suggestions on my dissertation and research.

I would also like to thank all my friends, for their consistent help.

Chapter 1

Introduction

1.1 Motivation

In the field of finance, a very common phenomenon, is that time series data undergoes some sudden "jumps", which implies change of distribution. Specifically, for a series of time-ordered observations at some time points, the distribution of the observations changes. This change might be a change of type, such as from a Normal distribution to a Student t distribution. Another kind of change which we are more interested in, is the change of parameters while the type of distribution remaining the same.

Figures 1.1 and 1.2 are two simple examples for parameter changes. In figure 1.1, the observations from time 1 to 40 are simulated from a Normal distribution with mean 1 and variance 0.2. From time 41 to 80, the observations are drawn from a Normal distribution with mean 2 and variance 0.2. Obviously, at time 41, the mean of the distribution of observations undergoes a "jump" from 1 to 2. This change is well reflected from the sample in the figure. In figure 1.2, the observations from time 1 to 40 are simulated from a Normal distribution with mean 0 and variance 1, while the last 40 observations are drawn from Normal distribution with mean 0 and variance 3. It's also easy to find that the variation of sample changes significantly at time point 41. The change point simply

refers to the time point that parameter undergoes changes. In these two examples, the change point is time 41.

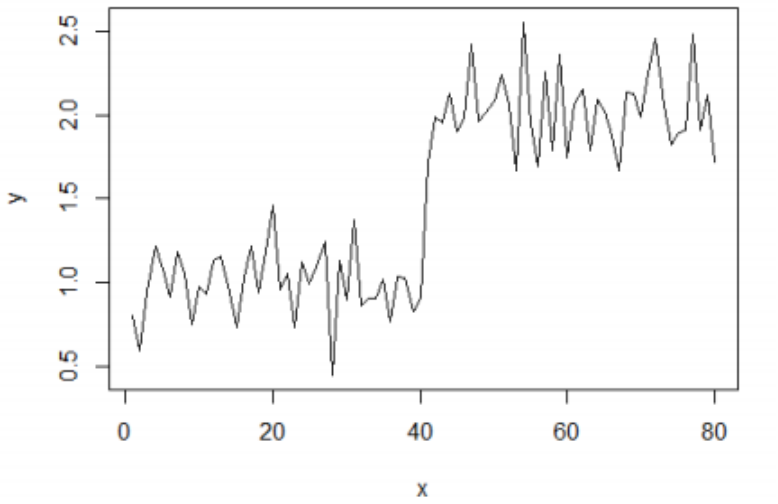

FIGURE 1.1: A Time Ordered Data with Mean Shifted

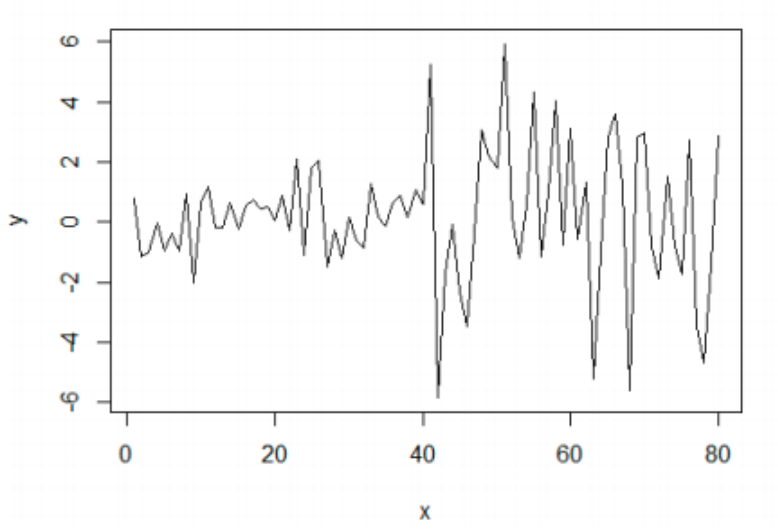

FIGURE 1.2: A Time Ordered Data with Variance Shifted

In these two examples, observations are in one dimensional space. However, in practice, the encountered observations are in the form of vectors coming from multiple dimensional space. The situation becomes quite different and complicated in multiple dimensional space. Taking the Normal distribution as an example, from one dimensional space to multiple dimensional space, the mean becomes a vector and the variance becomes

a covariance matrix. The first thing comes into our consideration is, how to measure the change of a vector or a matrix? It's not as simple as that of one dimensional space, in which the change can be measured by the absolute value of difference of two scalars.

In this dissertation, we concern about the change of covariance matrix, thus, at the very beginning, we need to figure out a proper tool to deal with the matrix in order to capture the change. In fact, motivated by principal component analysis (PCA), we propose the eigen-structure to measure the change of covariance matrix.

Invented by Karl Pearson in 1901, principal component analysis (PCA) has been one of the most famous statistical tool for dimensional reduction. Geometrically, PCA achieves dimensional reduction by conducting a rotation of axes. The new axes are in the same directions as those eigenvector of sample covariance matrix. The directions of top a few eigenvalues accounts most variation of the original data. Thus, we can project the data into the lower dimensional space constructed by the top few eigenvectors. In this situation, the dimension has been reduced while most variation (information) has been reserved.

Mathematically, PCA is equivalent to eigen-decomposition of sample covariance matrix, which implies that the eigen-structure of a matrix reveals the inner structure of a matrix.

Motivated by the spirit of PCA, it might be a proper way to perform a eigen-decomposition to the estimated covariance matrices. The eige-structure, including the eigenvalues and eigenvectors, describes the inner nature of covariance matrix by a series of orthogonal directions. The advantage is, it's possible for us to view the amount of variation (the eigenvalues) one by one from one dimensional spaces. In other words, an m dimensional covariance matrix represents the variation in the m dimensional space. We use m eigenvalues (eigenvectors) to measure it separately. Thus, any change of eigenvalues (eigenvectors) will imply the structure break of the covariance matrix.

Before reviewing the details of PCA, we would like to provide a simple example which implies potential existence of change-points in covariance matrix. The data is the U.S Treasury-LIBOR swap rates data. Each observation is an eight dimensional vector, with swap rate for different maturities (1 year, 2 years, 3 years, 4 years, 5 years, 7 years, 10 years and 30 years). Figure 1.3 is a snapshot of the original data. Since the case we care about is assumed with known mean zero, we difference the data in in advance: subtract the t^{th} day's swap rate by the $(t+1)^{th}$ day's swap rate. Thus, the differenced data actually reflects the daily change of swap rate. We call this data set the daily change of swap rate. Figure 1.4 is a snapshot of the differenced data.

swp1y	swp2y	swp3y	swp4y	swp5y	swp7y	swp10y	swp30y
7.10	7.16	7.17	7.17	7.17	7.20	7.24	7.24
7.03	7.06	7.07	7.07	7.08	7.11	7.14	7.16
7.07	7.13	7.14	7.15	7.16	7.19	7.21	7.21
7.01	7.04	7.06	7.06	7.07	7.10	7.14	7.14
7.04	7.09	7.11	7.13	7.14	7.17	7.20	7.19
7.04	7.10	7.11	7.13	7.14	7.18	7.22	7.20
7.06	7.12	7.14	7.15	7.17	7.20	7.23	7.19
7.04	7.09	7.10	7.12	7.13	7.16	7.19	7.13
7.08	7.14	7.16	7.17	7.19	7.21	7.23	7.17
7.12	7.21	7.23	7.25	7.28	7.31	7.35	7.28

FIGURE 1.3: Snapshot of Original Swap Rate Data

swp1y	swp2y	swp3y	swp4y	swp5y	swp7y	swp10y	swp30y
-0.07	-0.10	-0.10	-0.10	-0.09	-0.09	-0.10	-0.08
0.04	0.07	0.07	0.08	0.08	0.08	0.07	0.05
-0.06	-0.09	-0.08	-0.09	-0.09	-0.09	-0.07	-0.07
0.03	0.05	0.05	0.07	0.07	0.07	0.06	0.05
0.00	0.01	0.00	0.00	0.00	0.01	0.02	0.01
0.02	0.02	0.03	0.02	0.03	0.02	0.01	-0.01
-0.02	-0.03	-0.04	-0.03	-0.04	-0.04	-0.04	-0.06
0.04	0.05	0.06	0.05	0.06	0.05	0.04	0.04
0.04	0.07	0.07	0.08	0.09	0.10	0.12	0.11
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01

FIGURE 1.4: Snapshot of Differenced Swap Rate Data

Our motivation is, if all observations come from the same distribution (with mean

zero and the same covariance matrix), so will be any subset of the original data set. Therefore, if we choose a sequential subsets of the original data set and if any of their sample covariance matrices significantly differs from others, it implies that the observations might come from distributions with different covariance matrices. In other words, if we assume that all observations come from the same distribution (with mean zero and same covariance matrix), we expect the sample covariance matrices of those subsets are statistically the same.

A sequential subsets are chosen in this way. Let $k = 30$ denote the size of each subset. The first subset contains the observations from day 1 to day 30. The i^{th} subset contains the observations from day i to day $30 + i - 1$. There are $n = 1256$ days in the original data set, thus, finally we have 1227 subsets in a sequence. PCA is applied to each subset. The first eigenvalue and eigenvector are mostly concerned because they capture the most variance of the covariance matrix of each subset. If no significant change exists in the first eigenvalue or first eigenvector, it might imply that no change-point exists for the covariance matrix. Note that, it's also necessary to check other eigenvalues besides the first one since the change of other eigenvalues might also suggest a structure break.

Figures 1.5 shows the eight eigenvalues of the sample covariance matirx of the 1227 subsets. The first eigenvalue jumps up and down. It seems that the mean starts to go a little upward from time point 400. The figure 1.6 shows the corresponding eigenvector, which illustrates several "jumps" as well. As it's not very easy to figure out a clear change from the plots of first eigenvalue and eigenvector, we keep checking other eigenvalues. For the 2nd eigenvalue, a downward jump occurs close to the time point 400. The average value before time 400 is obviously higher than the average value after time 400. The plots of the 3rd and 5th eigenvalues suggest a great possibility of the existence of change-point. For the 3rd eigenvalue, a big change occurs around the time 500. The same thing occurs around time 300 for the 5th eigenvalue. For the rest plots of eigenvalues, the changes are not as clear as that of eigenvalue 3 and 5, but they doesn't keep stable as expected. Generally speaking, all eigenvalues don't keep stable along the time. The instability

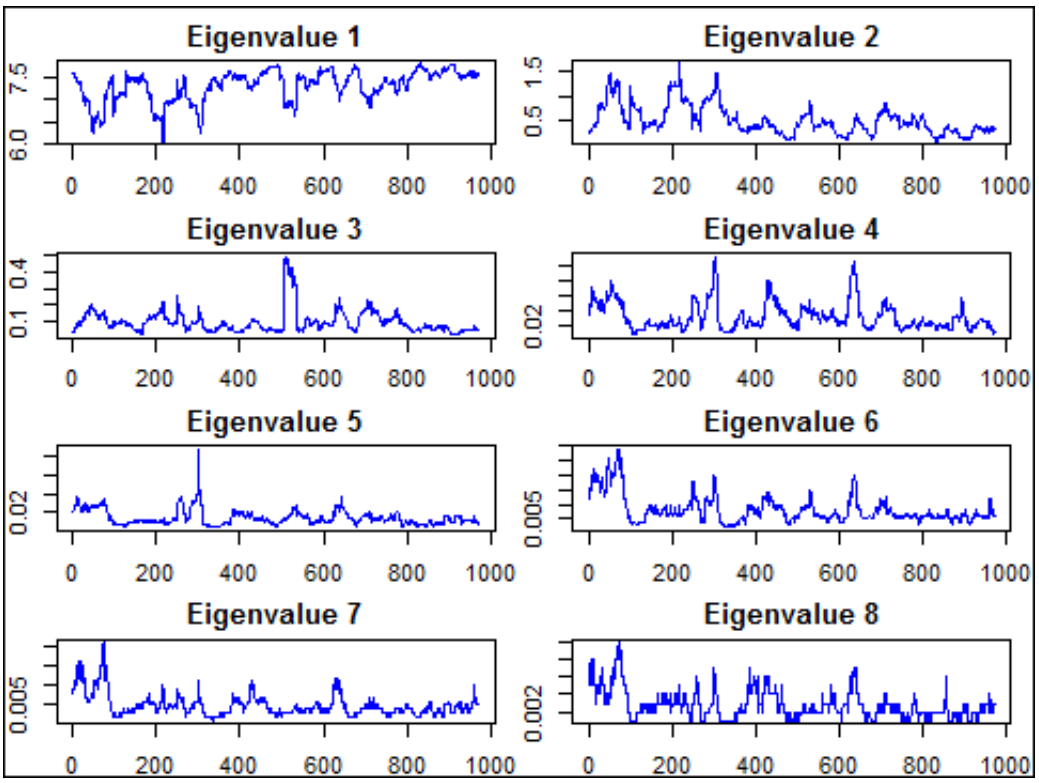

FIGURE 1.5: Eigenvalues of Sample Covariance Matrix of Each Subset

of eigenvalues (eigenvectors) of the sample covariance matrices of these subsets implies structure breaks in the covariance matrix of the true distribution.

1.2 Literature Review

In this section, we are going to briefly review the works on multiple change points in the literature and the mathematical description of principal component analysis.

1.2.1 Multiple Change Points

Parameter instability is a common phenomenon in time-series data. The time points at which parameters change are change-points. In previous section, figures 1.1 and 1.2

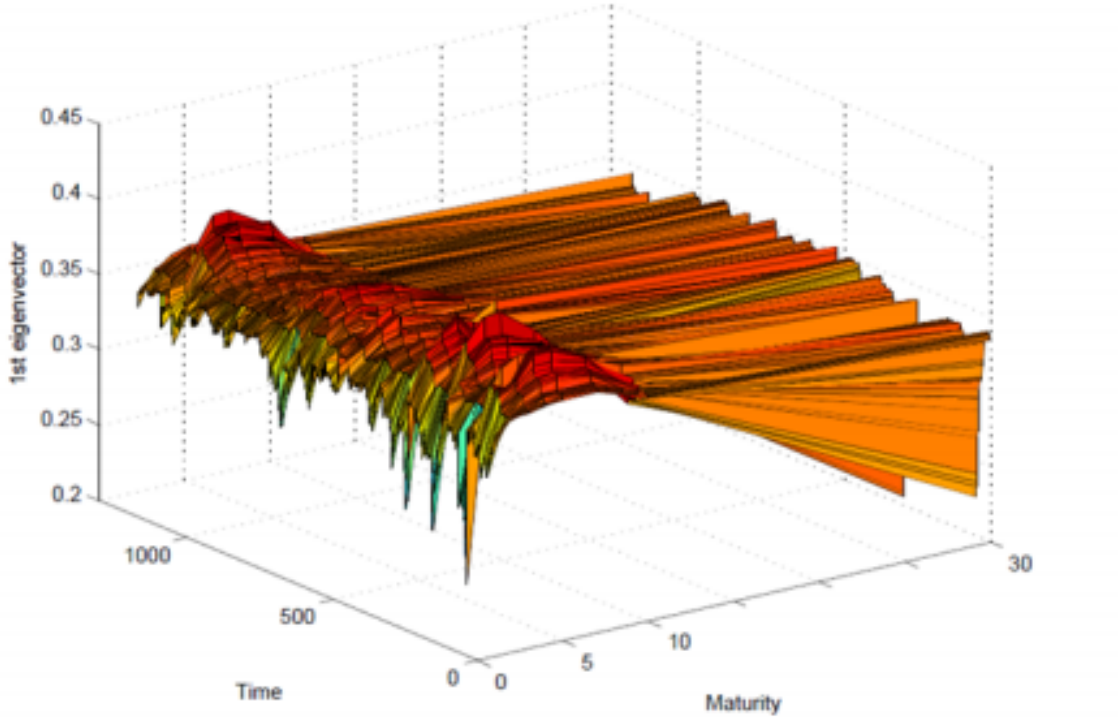

FIGURE 1.6: The First Eigenvector of Sample Covariance Matrix of Each Subset

are two simple examples illustrating samples from mean shift and variance shift in one dimensional case respectively.

The multiple change-points problem are considered here based on independent observations $x_1, ..., x_n$ such that x_i is a vector in m dimensional space. Instead of real time series, the sample is actually time-ordered observations, which means any two observations at different times are independent of each other. Its probability density function (pdf) is denoted as f_{θ_i}, in which the θ_i are unknown parameters that undergo some sudden "jumps". In the literature, works firstly has been done on assuming that at most one change-point exists. Like other traditional statistical inference, both frequentist and Bayesian methods are proposed. The frequentist approach can be considered as starting from Page's work in 1955, Quandt's work in 1958 and 1960, and Hinkley's work in 1970. In the meanwhile, a few Bayesian approaches can been found from Shiryaev's work in

1963, Carlin, Gelfand, and Smith's work in 1992. In their studies, they provide a hierarchical Bayesian model equipped with MCMC (Monte Carlo Markov Chain) sampling methods.

Right after the studies of single change-point problem, it's intuitive to extend the problem to multiple change-points problem which is more general. In the study of multiple change-points, the frequentist approach can be found in Bai's work in 1997, Bai and Perron's work in 1998, and Qu and Perron's work in 2007. They study the multiple change-points detection problem within a few regression models. In their study, the number of change-points is assumed bigger or equal to 2 and the regression coefficients are estimated by traditional least squares method with dynamic programming. However, the disadvantage of their method is the relatively high computational complexity. Vostrikova and Olshen suggest another method which is much more convenient, especially when the number of change-point is big. In order to choose a proper value of the number of change-point, the model selection criterion is used. In the study of Yao (1988), the famous Schwarzs Bayesian Information Criterion (BIC) is selected as the criterion for model selection. However, in the year of 2004 and 2006, Siegmund, Zhang and Siegmund found that for multiple change-points problems, the likelihood doesn't satisfy the regularity conditions which are required for deriving BIC. They have provided modifications to the criterion BIC in order to tackle this problem.

On the other hand, in the year of 1964, Cheronff and Zacks firstly apply the Bayesian approach to deal with multiple change-points problems. In their work, they consider the case of mean shift in Normal distribution. A few decades later, their work is extended to Gaussian auto-regressive models assuming potential changes in the level and error variance by McCulloch and Tsay in 1993. Almost at the same time, a product partition model is suggested by Barry and Hartigan. About two years later, the reversible jump MCMC (Markov chain Monte Carlo) method is introduced by Green. Besides, there are bunch of other Bayesian models, but they all rely on the assumption that the parameter

has a conjugate prior distribution. Moreover, the inference depends on numerical sampling method, such as Gibbs sampler. However, the drawback of all MCMC sampling methods is the unsatisfactory computational time.

The method used in this dissertation is proposed by Lai and Xing (2011), which is a Bayesian model for multiple change-points detection in an exponential family. This model has several advantages. The most attractive one might be that it has very good statistical and computational properties. In contrast with those Bayesian models which require MCMC for inference, explicit formulas for the estimates are available from our Bayesian model. The forward filter and backward filter are used to obtain the posterior densities of the parameters at each time given observations from time $1 \sim t$ and $(t+1) \sim n$ respectively. By combining these two, the final Bayesian estimate for the parameter at time t is a weighted average of posterior means. The weight can be calculated depending on the weights obtained from forward and backward filter. The explicit formulas will reduce the computational complexity significantly. The model is also equipped with an approximated algorithm called *bounded complexity mixture* (BCMIX) approximation. BCMIX bounds the amount of elements involving in the calculation so that it can speed up the original algorithm. The detail of this Bayesian model will be discussed in next chapter and BCMIX will be introduced at the beginning of simulation part.

1.2.2 Principal Component Analysis

Principal component analysis (PCA) is a well known statistical method used for dimensional reduction in multi-dimensional space. It transforms a group of observations with correlated features into a group of values of linearly uncorrelated features. These new features (variables) are referred as the principal components. Usually, we use less principal components (the new variables) than the number of original variables because principal

components are capable to account most variance of the data. Thus, principal component analysis achieves dimensional reduction and in the meanwhile it has reserved most information (variance) of the data in original space.

PCA has a close relationship with the eigenvalues and eigenvectors of a matrix. Let's first briefly review the definition of eigenvalue and eigenvector. We denote Q as a $m \times m$ matrix. We call λ the eigenvalue of matrix Q if there exists a non-zero vector v in m dimensional space such that $Qv = \lambda v$. Then the vector v is called the eigenvector of Q corresponding to the eigenvalue λ.

$Qv = \lambda v$ can be rewritten as $(Q - \lambda)v = 0$. Since a is a non-zero vector, this implies that λ is a solution to the equation $det(Q - \lambda I) = 0$, where det denotes the determinant of a matrix. It's easy to find that $det(Q - \lambda I)$ is a polynomial of degree m depending only on λ, thus, there must be m eigenvalues (in the field of complex number), while some of them can equal to each other. If Q is symmetric, then all its eigenvalues are real numbers and can be ordered as $\lambda_1 \geq ... \geq \lambda_m$. What's more, we have the followings:

$$det(V) = \lambda_1 ... \lambda_m, \quad tr(V) = \lambda_1 + ... + \lambda_m \tag{1.1}$$

The tr means the trace of a matrix. If v is an eigenvector of Q corresponding to the eigenvalue λ, then according to the definition of eigenvector, for any non-zero scalar c, cv is also an eigenvector of Q. Besides, with left-multiplying v^T to both sides of $\lambda a = Qv$ we will have

$$\lambda = v^T Qv / ||v||^2 \tag{1.2}$$

The main idea of principal component analysis is the following. We want to find a vector v with unit length $||v|| = 1$ such that the linear combination $v^T x$ has the largest variance over all such linear combinations. In order to maximize $v^T Qv (= Var(v^T x))$ over v subject to $||v|| = 1$, by introducing the Lagrange multiplier λ, this is equivalent to solve

the followings

$$\frac{\partial}{\partial v_i} v^T Q v + \lambda(1 - v^T v) = 0, \quad i = 1, ..., m \tag{1.3}$$

The m equations in (1.3) can be rewritten as the linear equation system $Qv = \lambda v$. For $v \neq 0$, this might imply that λ is an eigenvalue of matrix Q and v is the corresponding eigenvector, and with the relationship $\lambda = v^T Q v$ according to (1.2). Note that the denominator in (1.2) equals 1 since v is restricted as a unit vector.

Let $\lambda_1 = max_{v:||v||=1} v^T Q v$ and v_1 be the corresponding eigenvector with unit length. Next we consider the linear combination $v^T x$ that maximizes the $Var(v^T x) = v^T Q v$ subject to two restrictions: $v_1^T v = 0$ and $||v|| = 1$. Again, by introducing the Lagrange multiplier λ and η, it's equivalent to solve the following equations

$$\frac{\partial}{\partial v_i} v^T Q v + \lambda(1 - v^T v) + \eta v_1^T v = 0, \quad i = 1, ..., m \tag{1.4}$$

Similarly, the solution of the Lagrange multiplier λ is the second eigenvalue of matrix Q with corresponding unit eigenvector v_2. By continuing this process inductively, we can have all the eigenvalues $\lambda_1 \geqslant \lambda_2 ... \geqslant \lambda_m$ of the covariance matrix Q by solving the following optimization problem

$$\lambda_{k+1} = max_{v:||v||=1, v_j^T v = 0, 1 \leqslant j \leqslant k} v^T Q v \tag{1.5}$$

The v_{k+1} on the right hand side of (1.5) is the eigenvector corresponding to the $(k+1)th$ eigenvalue λ_{k+1}.

PCA can also be interpreted geometrically. We provide a simple example here to illustrate it. Suppose we have a sample with n observations and each observation is a two dimensional vector. Thus, each observation is a point in the two dimensional space. See figure 1.7.

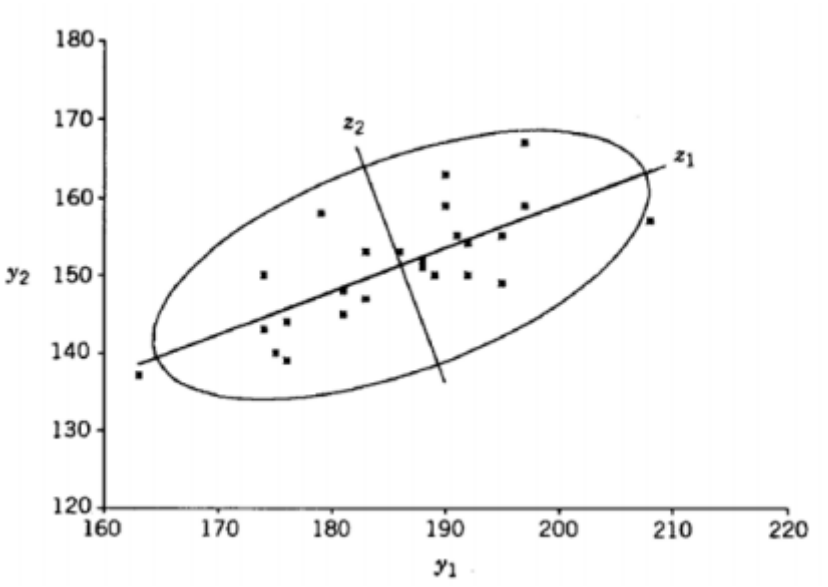

FIGURE 1.7: Geometrical Illustration of PCA

The two axes y_1 and y_2 are the original variables. z_1 and z_2 are in the same direction of the first and second eigenvector respectively. We can see from figure 1.7 that observations varies mostly in the direction of z_1. The variance equals to the first eigenvalue and it accounts the most variance of the total variance. z_2 is in the direction of the second eigenvector which is orthogonal to z_1. The variation of the data in the direction of z_2 is much less than that of z_1. From this figure, PCA can be explained as axes rotation as well: we rotate the old axes (y_1, y_2) to the new axes (z_1, z_2). The variance of the data is decomposed (mapped) into the two orthogonal directions of the new axes. If the top k ($k < p$, in this case, $k = 1$ and $p = 2$) directions account the most variance from the original p dimensional space, it's reasonable to map the data into a lower dimensional space without losing much information. Thus, dimensional reduction has been achieved.

Note that, in practice, we maximize $v^T\hat{Q}v$ instead of maximizing v^TQv, where the true covariance matrix Q is replaced by sample covariance matrix $\hat{Q}$, since the true covariance matrix Q is unknown. Therefore, the eigenvalues and eigenvectors all belong to sample covariance matrix. In fact, besides the sample covariance matrix, any estimation

for true covariance matrix can be plugged in. The corresponding eigen-structure belongs to the estimated covariance matrix. In this dissertation, we calculate the eigenvalues (eigenvectors) of the estimated covariance matrix at each time from the Bayes model.

With the powerful tool, PCA, we are able to decompose the variance from multiple dimensional space into several one dimensional space, in which the variation is visible and easy to be measured. Now, the last but the most important thing points to the basic problem in statistical field: estimate the parameter. However, our problem here is more than traditional estimation since we have to estimate parameters at each time. Actually it is a dynamical estimation. In next chapter, we are going to introduce the Bayes model which will tackle this problem.

Chapter 2

PCA with Multiple Change-Points

The model used to detect change-point is a Bayesian model. The final estimation of the parameter is a weighted average of a series of posterior means. The core of the model is how to calculate the weight. The model gives an closed form for the final estimation. However, for different cases of distribution, the formulas is different. Since our model relies on the Bayesian estimation, before introducing the details of the Bayesian model for change-point detection, we firstly give a brief review of Bayesian Inference.

The main idea of Bayesian inference is updating the prior distribution by the current data (information) in hand. In the view of frequentist, the unknown parameter, which we are interested in, is a fixed but unknown value. The disadvantage of frequentist approach is, it's likely to underestimate the variance of the predicted distribution. However, in the view of Bayesian statistician, the unknown parameter is also a variable which has a probability distribution. The distribution is known based on our prior knowledge on the parameter and so it is called the prior distribution. The prior distribution accounts uncertainty of unknown parameter and overcomes the drawbacks of traditional inference from frequentists.

Suppose x is an observation in the form of vector in m dimensional space and x is assumed from the distribution with parameter θ, i.e., $x \sim f(x|\theta)$. θ can be a scalar, a

vector or even a matrix. We denote the prior distribution of θ as $\pi(\theta|\eta)$, where η can be a vector and assumed known based on our prior knowledge. Let $\mathbf{X}$ be the set of n observations in the sample, i.e., $x_1, x_2, ..., x_n$. Bayesian method use the sample to update the prior distribution and obtain the posterior distribution. Firstly, we can calculate the marginal distribution of $\mathbf{X}$:

$$f(\mathbf{X}|\eta) = \int_\theta f(\mathbf{X}|\theta)\pi(\theta|\eta)d\theta,$$

Then, based on the Bayesian rule, the posterior distribution can be obtained:

$$\pi(\theta|\mathbf{X}, \eta) = \frac{f(\mathbf{X}, \theta|\eta)}{f(\mathbf{X}|\eta)} = \frac{f(\mathbf{X}|\theta)\pi(\theta|\eta)}{f(\mathbf{X}|\eta)}.$$

The Bayesian inference is based on the posterior distribution $\pi(\theta|\mathbf{X}, \eta)$. The estimator is called Bayes estimator. In decision theory, the Bayes estimator minimizes the posterior expected value of a loss function (i.e., the posterior expected loss). Let $\hat{\theta} = \hat{\theta}(x)$ be an estimator of θ, and let $L(\theta, \hat{\theta})$ be a loss function, such as squared error loss which might be the most popular one. The Bayes risk of $\hat{\theta}$ is defined as $E\{L(\theta, \hat{\theta})\}$, where the expectation is taken over the probability distribution of θ. Thus, the risk function actually is a function depending on the estimator $\hat{\theta}$. An estimator $\hat{\theta}$ is referred as a Bayes estimator if it minimizes the Bayes risk. Equivalently, for each x, if the estimator minimizes the expectation of posterior loss $E\{L(\theta, \hat{\theta}|x)\}$, it minimizes the Bayes risk. Thus it is a Bayes estimator as well.

The risk function we used here is the mean square error (MSE) which might be the most common one in the literature. The MSE is defined by

$$\text{MSE}(\hat{\theta}(x)) = E[(\hat{\theta}(x) - \theta)^2],$$

where the expectation is taken over the posterior distribution of θ given x. It's easy to proof that the posterior mean will minimize the MSE:

$$\begin{aligned}\text{MSE}(\hat{\theta}(x)) &= E[(\hat{\theta}(x) - \theta)^2] \\ &= E[(\hat{\theta}(x) - E(\theta|x) + E(\theta|x) - \theta)^2] \\ &= E[(\hat{\theta}(x) - E(\theta|x))^2] + E[(E(\theta|x) - \theta)^2] + 2E[(\hat{\theta}(x) - E(\theta|x))(E(\theta|x) - \theta)]\end{aligned}$$

Firstly let's look at the third part above, since both $\hat{\theta}$ and $E(\theta|x)$ are functions of x, so is $\hat{\theta} - E(\theta|x)$. Therefore, they are all constants under the expectation taken over the posterior distribution $\pi(\theta|x)$. Thus,

$$\begin{aligned}E[(\hat{\theta}(x) - E(\theta|x))(E(\theta|x) - \theta)] &= (\hat{\theta}(x) - E(\theta|x))E[E(\theta|x) - \theta] \\ &= (\hat{\theta}(x) - E(\theta|x))(E(\theta|x) - E(\theta|x)) \\ &= 0\end{aligned}$$

Then, the MSE can be reduced into

$$\text{MSE}(\hat{\theta}(x)) = E[(\hat{\theta}(x) - E(\theta|x))^2] + E[(E(\theta|x) - \theta)^2].$$

since the term $E[(E(\theta|x) - \theta)^2]$ is free of $\hat{\theta}(x)$, to minimize MSE is equivalent to minimize the non-negative part $E[(\hat{\theta}(x) - E(\theta|x))^2]$. Therefore, it's obvious that the minimizer is $\hat{\theta}(x) = E(\theta|x)$.

In practice, usually it's difficult to calculate the posterior mean of the posterior distribution directly. In other words, we can't do it analytically. Some posterior distributions even don't have closed form. In this case, sampling methods must be applied in order to get the posterior mean (or other statistical summaries). Markov Chain Monte Carlo (MCMC) provides a bunch of sampling methods (Metropolis–Hastings sampling, Gibbs sampling, etc.), which makes it's possible to get summaries numerically.

However, compare to the analytic form of posterior mean, the drawback of sampling

is the relatively high computational complexity. For MCMC method, it takes a large number of steps to get into the stable status (converge to the desired distribution). In our change-point detection model, the final estimation for a parameter at one time point is a weighted average of a series of posterior mean. The cost of calculating the weights is already a big burden. If we still have to calculate each posterior mean numerically by sampling methods, the total computational complexity will be unacceptable.

The assumption of distribution for observations is an exponential family, which will result in a closed form of posterior mean. Then, we have closed form of the final estimation and don't have to use sampling method to calculate posterior mean. If a distribution belongs to exponential family, its probability density function generally can be written as:

$$f(x|\theta) = u(x)v(\theta)exp(\eta(\theta)S(x)).$$

Exponential family has bunch of good properties. In Bayesian inference, if the data is observed from an exponential family, there often exists a conjugate prior that also belongs to exponential family, and it's easy to proof that the posterior distribution belongs to exponential family as well. Suppose the sample $\mathbf{X} = \{x_1, x_2, ..., x_n\}$ is from exponential family. We rewrite the density function of one observation parameterized with its natural parameter $\eta = \eta(\theta)$:

$$f(x|\eta) = u(x)v(\eta)exp(\eta S(x)),$$

Then, the joint density (likelihood) of the sample $\mathbf{X} = (x_1, x_2, ..., x_n)$ can be written as

$$f(\mathbf{X}|\eta) = (\prod_{i=1}^{n}(u(x_i))v(\eta)^n exp(\eta \sum_{i=1}^{n} S(x_i)).$$

Then, suppose the conjugate prior distribution is an exponential with density

$$\pi(\eta|\alpha, \beta) = w(\alpha, \beta)v(\eta)^\beta exp(\eta\alpha).$$

The parameter β stands for the effective number of observations that the prior distribution

contributes, while the parameter α can be regarded as the total amount that observations contribute to the sufficient statistic. $w(\alpha, \beta)$ is a normalization constant which guarantees the given function is a probability function (integration over domain equals 1). The posterior density is

$$\begin{aligned} f(\eta|\mathbf{X}, \alpha, \beta) &\propto f(\mathbf{X}|\eta)\pi(\eta|\alpha, \beta) \\ &= (\prod_{i=1}^{n} u(x_i))v(\eta)^n exp(\eta \sum_{i=1}^{n} S(x_i))w(\alpha, \beta)v(\eta)^\beta exp(\eta\alpha) \\ &\propto v(\eta)^{\beta+n} exp(\eta(\alpha + \sum_{i=1}^{n} S(x_i))), \end{aligned}$$

from which we can find that

$$f(\eta|\mathbf{X}, \alpha, \beta) = \pi(\eta|\alpha + \sum_{i=1}^{n} S(x_i), \beta + n)$$

which indicates that the posterior has the same form as the prior. The fantastic property of exponential family contributes to the closed form of Bayesian estimation from our change-point model.

2.1 A Bayesian Model for Multiple Change Points Detection

The main method used here to detect multiple change-points is a Bayesian model proposed by Lai and Xing (2011). We assume that the observations come from an exponential family of probability density function:

$$f_\theta(x) = exp\{\theta' x - \varphi(\theta)\} \tag{2.1}$$

Note that we have dropped the normalizing constant in the density function above. We denote the prior function of parameter θ as:

$$\pi(\theta; a_0, \mu_0) = c(a_0, \mu_0) exp\{a_0 \mu_0^{'} \theta - a_0 \varphi(\theta)\}, \tag{2.2}$$

a_0 and μ_0 are the parameters for the prior distribution and assumed known. $c(a_0, \mu_0)$ is a normalizing constant.

According to the conclusion of Diaconis and Ylvisaker, we are able to get the posterior density of θ given the observations $x_1, ..., x_n$

$$\pi(\theta; a_0 + n, \frac{a_0 \mu_0 + \sum_{i=1}^{n} x_i}{a_0 + n}); \tag{2.3}$$

Therefore, (2.2) is a conjugate prior density. Besides, it's obvious that

$$\int_{\Theta} f_\theta(x) \pi(\theta; a, \mu) d\theta = \frac{c(a, \mu)}{c(a + 1, \frac{a\mu + x}{a + 1})} \tag{2.4}$$

Now, we assume that the parameter vector θ_t may undergo some "jumps", instead of a constant along the time. For $t > 1$, we introduce the indicator variable

$$I_t := \begin{cases} 1, & \text{if } \theta_t \neq \theta_{t-1}; \\ 0, & \text{if } \theta_t = \theta_{t-1}. \end{cases} \tag{2.5}$$

which is independently distributed with $P(I_t = 1) = p$. Thus, it's a Bernoulli distribution and the parameter p is assumed known. If the parameter change occurs at time t (i.e., $I_t = 1$), the changed parameter θ_t is assumed to be sampled from its prior distribution $\pi(\theta | a_0, \mu_0)$. Then we are able to obtain the explicit formulas of $E(\theta_t | \mathcal{X}_t)$ and $E(\theta_t | \mathcal{X}_n)$, where $\mathcal{X}_t$ denotes $(x_1, ..., x_t)$. The property of exponential family contributes a lot in the explicit formulas of $E(\theta_t | \mathcal{X}_t)$ and $E(\theta_t | \mathcal{X}_n)$. We also use $\mathcal{X}_{i,j}$ to denote $(x_i, ..., x_j)$ for $1 \leq i \leq j \leq n$.

2.1.1 Forward Filter

We firstly define the most recent change-time C_t up to t as $C_t = max\{s \leq t : I_s = 1\}$. It plays an important role in the derivation of the explicit formulas of final estimation. By the conditional probability decomposition, we have

$$f(\theta_t|\mathcal{X}_t) = \sum_{i=1}^{t} P(C_t = i|\mathcal{X}_t) f(\theta_t|\mathcal{X}_t, C_t = i) \tag{2.6}$$

Let $p_{it} = P(C_t = i|\mathcal{X}_t)$ and note that $\{\mathcal{X}_t\} \cap \{C_t = i\} = \{\mathcal{X}_{i,t}\}$. This is because when given the most recent change up to time t occurs at time i, the information before time i is useless. Then we have

$$f(\theta_t|\mathcal{X}_t) = \sum_{i=1}^{t} p_{it} f(\theta_t|\mathcal{X}_{i,t}, C_t = i) \tag{2.7}$$

By the conclusion from (2.3), we can rewrite the posterior density as

$$f(\theta_t|\mathcal{X}_{i,t}, C_t = i) = \pi(\theta_t; a_0 + t - i + 1, \overline{X}_{i,t}) \tag{2.8}$$

where $\overline{X}_{i,t} = (a_0\mu_0 + \sum_{k=i}^{t} x_k)/(a_0 + t - i + 1)$ for $t > i$ and it is the posterior mean. We can find that the a_0 is updated by sample size $t - i + 1$ and the prior mean μ_0 is updated by the posterior mean. Combining (2.7) and (2.8) yields

$$f(\theta_t|\mathcal{X}_t) = \sum_{i=1}^{t} p_{it} \pi(\theta_t; a_0 + t - i + 1, \overline{X}_{i,t}) \tag{2.9}$$

For the purpose of calculating p_{it}, we provide the following recursive formulas. We have $\sum_{i=1}^{t} p_{it} = 1$ and

$$p_{it} \propto p_{it}^* := \begin{cases} pf(x_t|I_t = 1), & \text{if } i = t, \\ (1-p)p_{i,t-1} f(x_t|\mathcal{X}_{i,t-1}, C_t = i) & \text{if } i \leq t-1. \end{cases} \tag{2.10}$$

where

$$\begin{aligned} f(x_t|\mathcal{X}_{i,t-1}, C_t = i) &= \int f_{\theta_t}(x_t) f(\theta_t|\mathcal{X}_{i,t-1}, C_t = i) d\theta_t \\ &= \frac{c(a_0 + t - i, \overline{X}_{i,t-1})}{c(a_0 + t - i + 1, \overline{X}_{i,t})} \\ &= \frac{\pi_{i,t-1}}{\pi_{i,t}}. \end{aligned}$$

The p^*_{it} actually is the likelihood (given that the latest change occur at time i, for $i \leq t$) of x_t which can be decomposed by conditioning on whether the latest change occurring at time t. Thus, p_{it} (the probability that latest change occurs at time i for $1 \leq i \leq t$) is proportional to p^*_{it}. Then (2.10) can be simplified as

$$p^*_{it} = \begin{cases} \frac{p\pi_{0,0}}{\pi_{t,t}} & \text{if } i = t, \\ (1-p)p_{i,t-1}\frac{\pi_{i,t-1}}{\pi_{i,t}} & \text{if } i \leq t-1. \end{cases} \tag{2.11}$$

where $\pi_{0,0} = c(a_0, \mu_0)$ and $\pi_{i,j} = c(a_0 + j - i + 1, \overline{X}_{i,j})$.

We call the recursive formulas "forward filter" because we only use the information before time t. Particularly, $p_{it}(1 \leq i \leq t \leq n)$ can be calculated recursively as the following:

Starting from $t = 1$, p^*_{11} can be initialized as

$$p^*_{11} = \frac{\pi_{0,0}}{\pi_{1,1}}$$

Since p_{it} is proportion to p^*_{it}, p_{11} can be obtained by normalizing p^*_{11}

$$p_{11} = p^*_{11}/p^*_{11} = 1$$

Then, for $t = 2$,

$$p^*_{12} = (1-p)p_{11}\frac{\pi_{1,1}}{\pi_{1,2}}, \quad p^*_{22} = p\frac{\pi_{0,0}}{\pi_{2,2}}$$

Again, p_{12} and p_{22} can be calculated by normalization

$$p_{12} = \frac{p^*_{12}}{p^*_{12} + p^*_{22}}, \quad p_{22} = \frac{p^*_{22}}{p^*_{12} + p^*_{22}}$$

For $t = 3$,

$$p^*_{13} = (1-p)p_{12}\frac{\pi_{1,2}}{\pi_{1,3}}, \quad p^*_{23} = (1-p)p_{12}\frac{\pi_{2,2}}{\pi_{2,3}}, \quad p^*_{33} = p\frac{\pi_{0,0}}{\pi_{3,3}}$$

$$p_{13} = \frac{p^*_{13}}{p^*_{13} + p^*_{23} + p^*_{33}}, \quad p_{23} = \frac{p^*_{23}}{p^*_{13} + p^*_{23} + p^*_{33}}, \quad p_{33} = \frac{p^*_{33}}{p^*_{13} + p^*_{23} + p^*_{33}}$$

... ...

For $t = k$:

$$p^*_{ik} = (1-p)p_{i,k-1}\frac{\pi_{i,k-1}}{\pi_{i,k}}\,(i < k), \quad p^*_{kk} = p\frac{\pi_{0,0}}{\pi_{k,k}}$$

$$p_{ik} = \frac{p^*_{ik}}{\sum_{i=1}^{k} p^*_{ik}}\,(i \le k).$$

With the initialized value for p^*_{it} at $t = 1$, we recursively calculated p_{it} all the way down from $t = 1$ to $t = n$.

2.1.2 Backward Filter

The backward filter is very similar with the forward filter, obtained only by reversing time. We can define the indicator variable $\tilde{I}_t$ denoting if the change occurs from time t to time $t + 1$. Also we define $\tilde{C}_t = min\{s > t : \tilde{I}_s = 1\}$ as the closest change in the future of t. Then the posterior density given sample $\{x_{t+1}, ..., x_n\}$ can be expressed as

$$f(\theta_t|\mathcal{X}_{t+1,n}) = p\pi(\theta_t; a_0, \mu_0) + (1-p)\sum_{j=t+1}^{n} q_{j,t+1}\pi(\theta_t; a_0 + j - t, \overline{X}_{t+1,j}). \tag{2.12}$$

Since q_{jt} is proportion to the likelihood q^*_{jt}, we can also provide the following recursive formulas: $\sum_{j=t}^{n} q_{jt} = 1$ and

$$q^*_{j,t} = \begin{cases} \frac{p\pi_{0,0}}{\pi_{t,t}} & \text{if } j = t, \\ (1-p)q_{j,t+1}\frac{\pi_{t+1,j}}{\pi_{t,j}} & \text{if } j > t. \end{cases} \tag{2.13}$$

We call this "backward filter" because we only use the information in the future of time t. Particularly, $q_{j,t}$ can be calculated recursively as shown in the following:

Starting from $t = n$:

$$q^*_{n,n} = \frac{\pi_{0,0}}{\pi_{n,n}}$$

$$q_{n,n} = 1$$

For $t = n - 1$:

$$q^*_{n-1,n} = (1-p)q_{n,n}\frac{\pi_{n,n}}{\pi_{n-1,n}}, \quad q^*_{n-1,n-1} = p\frac{\pi_{0,0}}{\pi_{n-1,n-1}}$$

$$q_{n-1,n} = \frac{q^*_{n-1,n}}{q^*_{n-1,n} + q^*_{n-1,n-1}}, \quad q_{n-1,n-1} = \frac{q^*_{n-1,n-1}}{q^*_{n-1,n} + q^*_{n-1,n-1}}$$

For $t = n - 2$:

$$q^*_{n-2,n} = (1-p)q_{n-1,n}\frac{\pi_{n-1,n}}{\pi_{n-2,n}}, \quad q^*_{n-2,n-1} = (1-p)q_{n-1,n-1}\frac{\pi_{n-1,n-1}}{\pi_{n-2,n-1}}, \quad q^*_{n-2,n-2} = p\frac{\pi_{0,0}}{\pi_{n-2,n-2}}$$

$$q_{n-2,n} = \frac{q^*_{n-2,n}}{q^*_{n-2,n} + q^*_{n-2,n-1} + q^*_{n-2,n-2}}$$

$$q_{n-2,n-1} = \frac{q^*_{n-2,n-1}}{q^*_{n-2,n} + q^*_{n-2,n-1} + q^*_{n-2,n-2}}$$

$$q_{n-2,n-2} = \frac{q^*_{n-2,n-2}}{q^*_{n-2,n} + q^*_{n-2,n-1} + q^*_{n-2,n-2}}$$

... ...

For $t = n - k$:

$$q^*_{n-k,j} = (1-p)q_{n-k+1,j}\frac{\pi_{n-k+1,j}}{\pi_{n-k,j}}(j > n-k), \quad q^*_{n-k,n-k} = p\frac{\pi_{0,0}}{\pi_{n-k,n-k}}$$

$$q_{n-k,j} = \frac{q^*_{n-k,j}}{\sum_{j=n-k}^{n} q^*_{n-k,j}}$$

2.1.3 Explicit Expressions for $E(\theta_t|\mathcal{X}_n)$

From "forward filter" and "backward filter" we obtain the posterior density of $\theta_t|\mathcal{X}_t$ and $\theta_t|\mathcal{X}_{t+1,n}$. The posterior density of $\theta_t|\mathcal{X}_n$ can be derived by using Bayes's theorem combing these two. By Bayes's theorem,

$$\begin{aligned}
f(\theta_t|\mathcal{X}_n) &= \frac{f(\theta_t, \mathcal{X}_n)}{f(\mathcal{X}_n)} \\
&= \frac{1}{f(\mathcal{X}_n)} f(\theta_t, \mathcal{X}_t, \mathcal{X}_{t+1,n}) \\
&= \frac{1}{f(\mathcal{X}_n)} f(\mathcal{X}_t, \mathcal{X}_{t+1,n}|\theta_t)\pi(\theta_t; a_0, \mu_0) \\
&= \frac{1}{f(\mathcal{X}_n)} f(\mathcal{X}_t|\theta_t) f(\mathcal{X}_{t+1,n}|\theta_t)\pi(\theta_t; a_0, \mu_0) \\
&= \frac{1}{f(\mathcal{X}_n)} \frac{f(\theta_t|\mathcal{X}_t)}{\pi(\theta_t; a_0, \mu_0)} \frac{f(\theta_t|\mathcal{X}_{t+1,n})}{\pi(\theta_t; a_0, \mu_0)} \pi(\theta_t; a_0, \mu_0) \\
&= \frac{1}{f(\mathcal{X}_n)} \frac{f(\theta_t|\mathcal{X}_t) f(\theta_t|\mathcal{X}_{t+1,n})}{\pi(\theta_t; a_0, \mu_0)}
\end{aligned}$$

Note that $f(\mathcal{X}_n)$ is the marginal density which is free of θ_t, thus we have

$$f(\theta_t|\mathcal{X}_n) \propto \frac{f(\theta_t|\mathcal{X}_t) f(\theta_t|\mathcal{X}_{t+1,n})}{\pi(\theta; a_0, \mu_0)} \tag{2.14}$$

In order to write out (2.14), we firstly simplify an expression by:

$$\begin{aligned}
&\pi(\theta; a_0+t-i+1, \overline{X}_{i,t})\frac{\pi(\theta; a_0+j-t, \overline{X}_{t+1,j})}{\pi(\theta; a_0, \mu_0)} \\
&= \frac{c(a_0+t-i+1, \overline{X}_{i,t})c(a_0+j-t, \overline{X}_{t+1,j})}{c(a_0, \mu_0)} \\
&\quad * \frac{exp\{(a_0+t-i+1)(\overline{X}_{i,t}\theta - \varphi(\theta))\}exp\{(a_0+j-t)(\overline{X}_{t+1,j}\theta - \varphi(\theta))\}}{exp\{a_0\mu_0\theta - a_0\varphi(\theta)\}} \\
&= \frac{\pi_{i,t}\pi_{t+1,j}}{\pi_{0,0}} exp\{[(a_0+t-i+1)\overline{X}_{i,t} + (a_0+j-t)\overline{X}_{t+1,j} - a_0\mu_0]\theta - (j-i+1)\varphi(\theta)\}
\end{aligned}$$

Note that

$$\begin{aligned}
&(a_0+t-i+1)\overline{X}_{i,t} + (a_0+j-t)\overline{X}_{t+1,j} - a_0\mu_0 \\
&= a_0\mu_0 + \sum_{k=i}^{t} x_k + a_0\mu_0 + \sum_{k=t+1}^{j} x_k - a_0\mu_0 \\
&= a_0\mu_0 + \sum_{k=i}^{j} x_k \\
&= (a_0+j-i+1)\overline{X}_{i,j} \\
&= \frac{1}{\pi_{i,j}}\pi(a_0+j-i+1, \overline{X}_{i,j})
\end{aligned}$$

Thus we have

$$\pi(\theta; a_0+t-i+1, \overline{X}_{i,t})\frac{\pi(\theta; a_0+j-t, \overline{X}_{t+1,j})}{\pi(\theta; a_0, \mu_0)} = \frac{\pi_{i,t}\pi_{t+1,j}}{\pi_{00}\pi_{i,j}}\pi(a_0+j-i+1, \overline{X}_{i,j}) \quad (2.15)$$

Combine (2.12) with (2.9) and plug in the (2.15), the (2.14) can be written as

$$
\begin{aligned}
f(\theta_t|\mathcal{X}_n) &\propto \frac{f(\theta_t|\mathcal{X}_t)f(\theta_t|\mathcal{X}_{t+1,n})}{\pi(\theta;a_0,\mu_0)} \\
&= \frac{\sum_{i=1}^{t} p_{it}\pi(\theta_t;a_0+t-i+1,\overline{X}_{i,t})}{\pi(\theta_t;a_0,\mu_0)} \\
&\quad * [p\pi(\theta_t;a_0,\mu_0)+(1-p)\sum_{j=t+1}^{n} q_{j,t+1}\pi(\theta_t;a_0+j-t,\overline{X}_{t+1,j})] \\
&= \sum_{i=1}^{t} pp_{it}\pi(\theta_t;a_0+t-i+1,\overline{X}_{i,t}) \\
&\quad + \frac{(1-p)[\sum_{i=1}^{t} p_{it}\pi(\theta_t;a_0+t-i+1,\overline{X}_{i,t})][\sum_{j=t+1}^{n} q_{j,t+1}\pi(\theta_t;a_0+j-t,\overline{X}_{t+1,j})]}{\pi(\theta_t;a_0,\mu_0)} \\
&= \sum_{i=1}^{t} pp_{it}\pi(\theta_t;a_0+t-i+1,\overline{X}_{i,t}) \\
&\quad + \sum_{1\le i\le t<j\le n} (1-p)p_{it}q_{j,t+1}\frac{\pi_{it}\pi_{t+1,j}}{\pi_{ij}\pi_{00}}\pi(a_0+j-i+1,\overline{X}_{i,j})
\end{aligned}
$$

Note that the first part above can be regarded as $i \le t = j$, thus the posterior density at time t can be obtained from (2.14)

$$
f(\theta_t|\mathcal{X}_n) = \sum_{1\le i\le t\le j\le n} w_{ijt}\pi(\theta_t;a_0+j-i+1,\overline{X}_{i,j}) \tag{2.16}
$$

where $w_{ijt} = w^*_{ijt}/P_t$, $P_t = p + \sum_{1\le i\le t\le j\le n} w^*_{ijt}$, and

$$
w^*_{ijt} = \begin{cases} pp_{it} & \text{if } i \le t = j, \\ (1-p)p_{it}q_{j,t+1}\frac{\pi_{it}\pi_{t+1,j}}{\pi_{ij}\pi_{00}} & \text{if } i \le t < j. \end{cases} \tag{2.17}
$$

From (2.17), it follows that

$$
P(I_{t+1}=1|\mathcal{X}_n) = \frac{p}{P_t}, \quad E(\theta_t|\mathcal{X}_n) = \sum\nolimits_{1\le i\le t\le j\le n} w_{ijt}\overline{X}_{i,j} \tag{2.18}
$$

2.1.4 Estimation of Hyperparameters

Notice that in order to calculate the estimated parameters, we need to know the hyper-parameters p, a_0 and μ_0 which are assumed known previously. However, we still have to estimate them in practice. The method used here is the Maximum Likelihood Estimation (MLE), which can be simply described as the following.

Suppose the observations $x_1, x_2, x_3, ..., x_n$ are i.i.d (independent and identically distributed) from a certain known distribution with unknown parameters θ. Its density function can be written as $f(\cdot|\theta)$. Both x_i and θ can be vectors. Since all observations are independent with each other, the joint density function can be written as

$$f(x_1, x_2, ..., x_n|\theta) = f(x_1|\theta)f(x_2|\theta)...f(x_n|\theta)$$

Since $x_1, x_2, x_3, ..., x_n$ are observations and are fixed, the only variable in the joint density function is the parameter θ. In other words, it can be regarded as a function depending on parameter θ:

$$\mathcal{L}(\theta|x_1, x_2, ..., x_n) = f(x_1, x_2, ..., x_n|\theta) = \prod_{i=1}^{n} f(x_i|\theta)$$

In practice it is often more convenient to work with the logarithm of the likelihood function which known as the log-likelihood:

$$l = ln\mathcal{L}(\theta|x_1, x_2, ..., x_n) = \sum_{i=1}^{n} lnf(x_i|\theta)$$

The maximum likelihood estimation (MLE) is:

$$\hat{\theta}_{mle} = argmax\{l(\theta|x_1, x_2, ..., x_n)\}$$

which can be explained as, we believe our observations are drawn from the one which gives highest probability among all distribution family members.

In the paper of Lai and Xing (2011), they propose a way to estimate the hyperparameters which is described as the following.

The likelihood function hyperparameter $\theta = \theta(p, a_0, \mu_0)$ can be written as

$$\begin{aligned} f_\theta(x_1, ..., x_n) &= f(x_1) f(x_2, ..., x_n | x_1) \\ &= f(x_1) f(x_2|x_1) f(x_3, ..., x_n | x_1, x_2) \\ &= \prod_{t=1}^{n} f(x_t | \mathcal{X}_{t-1}) \end{aligned}$$

Note that the likelihood $f(x_t|\mathcal{X}_t)$ can be decomposed into $\sum_{i=1}^{t} p_{it}^*$. Since p_{it}^* stands for the marginal density (likelihood) of x_t given the latest change occurs at time i. If we sum the i over $1 \leq i \leq t$, it equals to the marginal density $f(x_t|\mathcal{X}_t)$. Thus, the likelihood function finally can be written as

$$L(\theta|\mathcal{X}_n) = \prod_{t=1}^{n} f(x_t|\mathcal{X}_{t-1}) = \prod_{t=1}^{n} (\sum_{i=1}^{t} p_{it}^*) \tag{2.19}$$

p_{it}^* is a function of p, a_0, and μ_0 shown in 2.11. The μ_0 is suggested to be estimated by the sample mean $\hat{\mu} = n^{-1} \sum_{t=1}^{n} x_t$. However, since what we care about here is the covariance matrix, intuitively, we use sample covariance matrix, $\mathbf{S} = (n-1)^{-1} \sum_{t=1}^{n} (x_t - \bar{x})(x_t - \bar{x})^T$, to estimate $\hat{\mu}$ instead of sample mean. As we have assumed a Normal distribution with known mean zero, the sample covariance matrix should be expressed in the form of $\mathbf{S} = n^{-1} \sum_{t=1}^{n} x_t x_t^T$. a_0 is recommended to be set as 1, which can be interpreted as: for the prior knowledge on the parameter θ we have, it's equivalent to the information from one additional observation. Substituting these two estimated values into 2.19, we just need to maximize the log-likelihood function $l(p) = \sum_{t=1}^{n} log(\sum_{i=1}^{t} p_{it}^*)$ only depending on p. It is suggested to maximize the log-likelihood function by grid search which is in the form of $\{2^k/n : k_0 \leq k \leq k_1\}$, where $k_0 < 0 < k_1$ are integers.

2.2 The Explicit Formulas for Normal Distributed Observations

In this section, the explicit formulas for Normal distributed observations will be derived, both for one dimensional case and multi-dimensional case. The formulas from multi-dimensional case will be used in case studies chapter 4.

2.2.1 One Dimensional Case

Suppose $x_1, ..., x_m \sim N(0, \sigma^2)$, where σ^2 are unknown. The density function is:

$$f_{\sigma^2}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\{-\frac{x^2}{2\sigma^2}\}$$

Let $\tau = (2\sigma^2)^{-1}$, $\tau \sim Gamma(g, \lambda)$. The parameter g and λ are assumed known. Then the density function above can be written as

$$f_\tau(x) = \frac{1}{\sqrt{\pi}} exp\{-\tau x^2 - \frac{1}{2} log \frac{1}{\tau}\}$$

Let $\theta = -\tau$, then the density function becomes

$$f_\theta(x) = \frac{1}{\sqrt{\pi}} exp\{\theta x^2 - \frac{1}{2} log(-\frac{1}{\theta})\} \tag{2.20}$$

Since $\tau \sim Gamma(g, \lambda)$,that is

$$\pi(\tau) = \frac{1}{\Gamma(g)\lambda^g} \tau^{g-1} e^{-\frac{\tau}{\lambda}}$$

therefore, the prior density of θ can be easily derived as

$$\begin{aligned} \pi(\theta) &= \pi(\tau(\theta))|\tau'(\theta)| \\ &= \frac{1}{\Gamma(g)\lambda^g}(-\theta)^{g-1} e^{\frac{\theta}{\lambda}} \\ &= \frac{1}{\Gamma(g)\lambda^g} exp\{\frac{1}{\lambda}\theta - 2(g-1)\frac{1}{2} log(-\frac{1}{\theta})\} \end{aligned} \tag{2.21}$$

If we let $\phi(\theta) = \frac{1}{2}log(-\frac{1}{\theta})$, $a_0 = 2(g-1)$, $\mu_0 = \frac{1}{2\lambda(g-1)}$, and $c(a_0, \mu_0) = \frac{1}{\Gamma(g)\lambda^g}$, then (2.20) and (2.21) can be expressed as:

$$f_\theta(x) = \frac{1}{\sqrt{\pi}} exp\{\theta x^2 - \phi(\theta)\} \tag{2.22}$$

$$\pi(\theta; a_0, \mu_0) = c(a_0, \mu_0) exp\{a_0\mu_0\theta - a_0\phi(\theta)\} \tag{2.23}$$

Now, we verify the property stated in (2.3). Since $\tau = (2\sigma^2)^{-1} \sim Gamma(g, \lambda)$, it is simple to find that $\sigma^2 \sim Inverse - Gamma(g, \frac{1}{2\lambda})$. Therefore, by the property of conjugate family, the posterior density of σ^2 given observations $x_1, ..., x_m$ is:

$$\sigma^2 | x_1, ..., x_m \sim Inverse - Gamma(g + \frac{m}{2}, \frac{\frac{1}{\lambda} + \sum_{i=1}^m x_i^2}{2})$$

Thus,

$$\tau | x_1, ..., x_m \sim Gamma(g + \frac{m}{2}, \frac{1}{\frac{1}{\lambda} + \sum_{i=1}^m x_i^2})$$

Since the posterior distribution is still a gamma distribution, again, we can easily write the posterior density of θ in forms of (2.23) by

$$\pi(\theta | x_1, ..., x_m) = \pi(\theta | a_0^*, \mu_0^*)$$

where

$$a_0^* = 2(g + \frac{m}{2} + 1) = 2(g-1) + m = a_0 + m$$

$$\mu_0^* = \frac{1}{2(g + m/2 - 1)\frac{1}{\frac{1}{\lambda} + \sum_{i=1}^m x_i^2}} = \frac{\frac{1}{\lambda} + \sum_{i=1}^m x_i^2}{2(g-1) + m} = \frac{a_0\mu_0 + \sum_{i=1}^m x_i^2}{a_0 + m}$$

The property stated in (2.3) holds. Note that, instead of sample mean, the sample variance is used as updated information. This is intuitive because we are interested in posterior mean of variance instead of posterior mean of mean.

The important formulas that participate in the calculation hereby can be written as:

$$\pi_{0,0} = c(a_0, \mu_0) = \frac{1}{\Gamma(g)}(\frac{1}{\lambda})^g \tag{2.24}$$

$$\pi_{i,j} = c(a_{ij}, \mu_{ij}) = \frac{1}{\Gamma(g_{ij})}(\frac{1}{\lambda_{ij}})^{g_{ij}} \tag{2.25}$$

$$\overline{X}_{ij} = \frac{1}{2\lambda_{ij}(g_{ij} - 1)} \tag{2.26}$$

where $g_{ij} = g + \frac{j-i+1}{2}$ and $\lambda_{ij} = (\frac{1}{\lambda} + \sum_{k=i}^{j} x_k^2)^{-1}$.

2.2.2 Multi-Dimensional Case

Now, we consider a general case in m dimensional space. Suppose $\tilde{x}_1, ..., \tilde{x}_n \sim N(\tilde{0}, \Sigma)$, where $\tilde{x}_1, ..., \tilde{x}_n$ are m dimensional vectors. The density function can be written as

$$\begin{aligned} f_\Sigma(\tilde{x}) &= \frac{1}{(2\pi)^{\frac{m}{2}}|\Sigma|^{\frac{1}{2}}} exp\{-\frac{1}{2}\tilde{x}^T\Sigma^{-1}\tilde{x}\} \\ &= (2\pi)^{\frac{m}{2}} exp\{\tilde{x}^T(-\frac{\Sigma^{-1}}{2})\tilde{x} - \frac{1}{2}log(|\Sigma|)\} \end{aligned}$$

Let $\theta = -\frac{\Sigma^{-1}}{2}$ and $\phi(\theta) = \frac{1}{2}log(|\Sigma|) = \frac{1}{2}log(|-\frac{\theta^{-1}}{2}|)$. Then, the density function of $\tilde{x}$ can be rewritten as

$$f_\theta(\tilde{x}) = (2\pi)^{-\frac{m}{2}} exp\{\tilde{x}^T\theta\tilde{x} - \phi(\theta)\}$$

We assume the prior distribution of Σ is Inverse Wishart Distribution: $\Sigma \sim IW_m(\Psi, n_0)$. Thus, Σ^{-1} has a Wishart distribution: $\Sigma^{-1} \sim W_m(\Psi^{-1}, n_0)$ with density function:

$$\begin{aligned} \pi(\Sigma^{-1}) &= \frac{1}{2^{\frac{n_0 m}{2}}|\Psi^{-1}|^{\frac{n_0}{2}}\Gamma_m(\frac{n_0}{2})} |\Sigma^{-1}|^{\frac{n_0-m-1}{2}} exp\{-\frac{1}{2}tr((\Psi^{-1})^{-1}\Sigma^{-1})\} \\ &= \frac{|\Psi|^{\frac{n_0}{2}}}{2^{\frac{n_0 m}{2}}\Gamma_m(\frac{n_0}{2})} |\Sigma|^{-\frac{n_0-m-1}{2}} exp\{-\frac{1}{2}tr(\Psi\Sigma^{-1})\} \\ &= \frac{|\Psi|^{\frac{n_0}{2}}}{2^{\frac{n_0 m}{2}}\Gamma_m(\frac{n_0}{2})} exp\{tr\left[\Psi(-\frac{\Sigma^{-1}}{2})\right] - \frac{n_0 - m - 1}{2}log|\Sigma|\} \end{aligned}$$

Since $\theta = -\frac{\Sigma^{-1}}{2}(\Sigma^{-1}(\theta) = -2\theta)$, the density function of θ is:

$$\begin{aligned}\pi(\theta) &= \pi(\Sigma^{-1}(\theta))|(\Sigma^{-1}(\theta))'| \\ &= \frac{|\Psi|^{\frac{n_0}{2}}}{2^{\frac{n_0 m}{2}}\Gamma_m(\frac{n_0}{2})}|-2\theta|^{\frac{n_0-m-1}{2}}exp\{tr(\Psi\theta)\}|-2| \\ &= \frac{|\Psi|^{\frac{n_0}{2}}}{2^{\frac{n_0 m}{2}-1}\Gamma_m(\frac{n_0}{2})}exp\{tr(\Psi\theta)-(n_0-m-1)\phi(\theta)\}\end{aligned}$$

Let $c(a_0,\mu_0) = \frac{|\Psi|^{\frac{n_0}{2}}}{2^{\frac{n_0 m}{2}-1}\Gamma_m(\frac{n_0}{2})}$, $a_0 = n_0 - m - 1$, $a_0\mu_0 = \Psi$, and $\mu_0 = \frac{\Psi}{n_0-m-1}$. $\pi(\theta)$ can be expressed as

$$\pi(\theta; a_0, \mu_0) = c(a_0,\mu_0)exp\{tr(a_0\mu_0\theta) - a_0\phi(\theta)\}$$

To verify the property stated in (2.3), by the conclusion of conjugate family, the posterior distribution of Σ given observations $\tilde{x}_1, ..., \tilde{x}_n$ is:

$$\Sigma|\tilde{x}_1, ..., \tilde{x}_n \sim IW_m(\Psi^*, n_0^*)$$

where $\Psi^* = \Psi + \sum_{i=1}^n \tilde{x}_i^T\tilde{x}_i$ and $n_0^* = n_0 + n$. Then, the posterior density of θ is:

$$\pi(\theta; a_0^*, \mu_0^*) = c(a_0^*,\mu_0^*)exp\{tr(a_0^*\mu_0^*\theta) - a_0^*\phi(\theta)\}$$

where

$$a_0^* = n_0^* - m - 1 = n_0 + n - m - 1 = a_0 + n$$

$$\mu_0^* = \frac{\Psi^*}{a_0^*} = \frac{\Psi + \sum_{i=1}^n \tilde{x}_i^T\tilde{x}_i}{a_0+n} = \frac{a_0\mu_0 + \sum_{i=1}^n \tilde{x}_i^T\tilde{x}_i}{a_0+n}$$

Thus, the property holds. Then, we can write those important formulas participating in the calculation:

$$\pi_{0,0} = c(a_0,\mu_0) = \frac{|\Psi|^{\frac{n_0}{2}}}{2^{\frac{n_0 m}{2}-1}\Gamma_m(\frac{n_0}{2})} \tag{2.27}$$

$$\pi_{i,j} = c(a_{ij},\mu_{ij}) = \frac{|\Psi_{ij}|^{\frac{n_{0(ij)}}{2}}}{2^{\frac{n_{0(ij)} m}{2}-1}\Gamma_m(\frac{n_{0(ij)}}{2})} \tag{2.28}$$

$$\overline{X}_{ij} = \frac{a_0\mu_0 + \sum_{k=i}^{j} \tilde{x}_k^T \tilde{x}_k}{a_0 + (j-i+1)} = \frac{\Psi_{ij}}{n_{0(ij)} - m - 1} \tag{2.29}$$

where $\Psi_{ij} = \Psi + \sum_{k=i}^{j} \tilde{x}_k^T \tilde{x}_k$ and $n_{0(ij)} = n_0 + (j-i+1)$.

2.3 PCA to Estimated Variance-Covariance Matrix

In one dimensional case, the estimated parameter at each time is just a number. It's easy for us to check the change-point by simply plotting the estimated value of parameters. However, we are focusing on covariance matrix in multi-dimensional space, and it is impossible to plot the estimated covariance matrix at each time directly. In order to find out the change of a series of matrices, we firstly need to find a way to measure the change of matrix. Motivated by the spirit of PCA, we hereby perform the eigen-decomposition to the estimated covariance matrices. Similarly as the variance in one dimensional space, covariance matrix describes the variation of a random vector in multiple space. PCA makes it possible for us to decomposed the variation into several orthogonal directions so that we can check the variation in each direction.

As demonstrated before, the final estimated covariance matrix is a weighted average of a series of matrices (the posterior means). An intuitive wondering is, if the eigenvalues and eigenvectors are still the weighted average (with the same weights w_{ijt}) of eigenvalues and eigenvectors of the covariance matrices of the posterior means $\overline{X}_{ij}$. If so, the numerical calculation can be simplified by only storing eigenvalues and eigenvectors (usually for top a few). In this way, we are able to avoid storing the whole matrices involved in the calculation, which will save much memory space of a computer. The limit of memory space is a big issue for the calculation when the dimension is high.

Unfortunately, in general, it is not true. The following is a simple example. Let

$$A_1 = \begin{pmatrix} 0.18233835 & 0.03813438 \\ 0.03813438 & 0.11766165 \end{pmatrix}$$

$$A_2 = \begin{pmatrix} 0.4680357 & -0.1974292 \\ -0.1974292 & 0.5319643 \end{pmatrix}$$

$$w_1 = 0.4, \quad w_2 = 0.6$$

$$A = w_1 A_1 + w_2 A_2$$

The eigenvalues of A are 0.4633924 and 0.2566076. However, the weighted average of eigenvalues of A_1 and A_2 are 0.5 and 0.22. It is also easy to verify that the eigenvectors of A are not the weighted average of eigenvectors of A_1 and A_2 either.

However, under a certain condition, the eigenvalues/eigenvectors are the weighted average of the eigenvalues/eigenvectors of each segment. Next, we provide an important observation with which a sufficient condition can be proofed. Under this sufficient condition, the previous statement is true.

An Important Observation. Given a finite set of $m \times m$ matrix, say $A_1, A_2, ..., A_n$ such that each A_i is diagonalizable, and they commute with each other, i.e. $A_i A_j = A_j A_i$, $\forall i, j$. Then $\exists P$ invertible, and $P^{-1} A_i P = D$, where D is a diagonal matrix.

A Sufficient Condition. Suppose $A_1, ..., A_n$ are $m \times m$ matrices. They are all diagonalizable and commute with each other. Let $A = \sum_{i=1}^{n} w_i A_i$, where $\sum_{i=1}^{n} w_i = 1$. Let λ_k and λ_k^i denote the eigenvalue of A and A_i respectively, and let $\tilde{v}_k$ and $\tilde{v}_k^i$ denote the unit eigenvector with same sign of A and A_i respectively, for $k = 1, 2, ..., m$, and $i = 1, 2, ..., n$. Then we have:

$$\lambda_k = \sum\nolimits_{i=1}^{n} w_i \lambda_k^i, \quad \tilde{v}_k = \sum\nolimits_{i=1}^{n} w_i \tilde{v}_k^i$$

The proofs are given in appendix. In fact, the condition given here is too strict. In practice, we can seldom find a series of matrices within which any two can commute with each other. Thus, we still have to store all posterior means (covariance matrices), which makes the computation more difficult.

Chapter 3

Simulation Studies

In order to test the performance of our Bayesian model on detecting change-points in covariance matrix, simulation studies are necessary. The problem is how to shorten the computational time. For one simulation study, the model has to been run at least hundreds of times which will result in an unacceptable computational time. Two major issues should be considered.

Firstly, the Bayes filter uses recursive formulas to calculate the weights p_{it} and $q_{j,t}$. We can see that the number of weight w_{ijt} in final estimation is increasing with the sample size n. In fact, the maximum number of weights equals $n^2/2$ resulting in unbounded memory requirements as well as computational complexity. Secondly, though the traditional programming software "R" provides bunch of packages which make statistical computation very convenient, the computational speed is not satisfactory. BCMIX approximation and programming in C language focus on solving these two problems respectively. In the first two sections, the BCMIX approximation and several algorithms for eigen-decomposition will be introduced. The results from simulation studies will be presented in the last section.

3.1 BCMIX Approximation

To bound the computational complexity and memory requirements in estimating the parameter at each time, Lai and Xing (2011) has proposed an approximated approach called *bounded complexity mixture* approximation which can be described as the following.

The idea is, we select $G(p)$ components in total which has $g(p)$ most recent weights $p_{j,n}$ (with $n - m(p) < j \leq n$ and $g(p) < G(p)$). For example, at stage $t-1$, let $\mathcal{D}_{t-1}(p)$ be the set of indices i for which $p_{i,t-1}$ is kept at stage $t-1$. As mentioned, since $\mathcal{D}_{t-1}(p)$ contains the indices of most recent $g(p)$ weight, the set $\{t-1, ..., t-g(p)\}$ is a subset of $\mathcal{D}_{t-1}(p)$, i.e. $\mathcal{D}_{t-1}(p) \supset \{t-1, ..., t-g(p)\}$. At stage t, the indices set $\mathcal{D}_t(p)$ is obtained by updating $\mathcal{D}_{t-1}(p)$. Since $\mathcal{D}_t(p)$ also must have the most recent $g(p)$ components, we have to add index t into the old set $\mathcal{D}_{t-1}(p)$. Now we have $G(p)+1$ components in total, thus, we need to delete one. Let i_t be the index not belonging to $\{t, ..., t-g(p)+1\}$ such that

$$p^*_{i_t,t} = min\{p^*_{i,t} : i \in \mathcal{D}_{t-1}(p) \quad \text{and} \quad i \leq t - g(p)\}$$

We choose the i_t corresponding to the smallest weight $p^*_{i_t,t}$ to be deleted. If there are two or more smallest $p^*_{i_t,t}$, we choose the index which is farthest from t. Then, $\mathcal{D}_t(p)$ can be expressed as $\mathcal{D}_t(p) = \{t\} \cup (\mathcal{D}_t(p) - \{i_t\})$. By normalizing $p^*_{i,t}$ we get $p_{i,t}$

$$p_{i,t} = (\frac{p^*_{i,t}}{\sum_{j \in \mathcal{D}_t(p)} p^*_{j,t}}), \quad i \in \mathcal{D}_t(p).$$

The above is only for selecting weights in forward filter. Similarly, we can apply the selection process to backward filter. Let $\tilde{\mathcal{D}}_{t+1}(p)$ denote the set of indices j for which $q_{j,t+1}$ is kept at stage $t+1$. $\tilde{\mathcal{D}}_{t+1}(p)$ contains the closest $g(p)$ indices in the future. That is, $\tilde{\mathcal{D}}_{t+1}(p) \supset \{t+1, ..., t+g(p)\}$. The way to obtain $\tilde{\mathcal{D}}_t(p)$ by updating $\tilde{\mathcal{D}}_{t+1}(p)$ is totally the same as that in forward filter. Let j_t be the index not belonging to $\{t, ..., t+g(p)-1\}$ such that

$$q^*_{j_t,t} = min\{q^*_{j,t} : j \in \tilde{\mathcal{D}}_{t+1}(p) \quad \text{and} \quad j \geq t + g(p)\}$$

We choose j_t to be the farthest from t which corresponds to the smallest weight $q^*_{j_t,t}$. Then, $\tilde{\mathcal{D}}_t(p) = \{t\} \cup (\tilde{\mathcal{D}}_{t+1}(p) - \{j_t\})$ and let

$$q_{j,t} = \frac{q^*_{j,t}}{\sum_{j \in \tilde{\mathcal{D}}_t(p)} q^*_{j,t}}$$

where $j \in \tilde{\mathcal{D}}_t(p)$.

Finally, the BCMIX approximation to the posterior density at time t can be written as

$$f(\theta_t|\mathcal{X}_n) \approx \sum_{i \in \mathcal{D}_t(p), j \in \tilde{\mathcal{D}}_{t+1}(p)} \tilde{w}_{ijt} \pi(\theta_t; a_0 + j - i + 1, \overline{X}_{i,j}) \tag{3.1}$$

in which $\tilde{w}_{ijt} = w^*_{ijt}/\tilde{P}_t$, $\tilde{P}_t = p + \sum_{1 \le t \le n, i \in \mathcal{D}_t(p), j \in \tilde{\mathcal{D}}_{t+1}(p)} w^*_{ijt}$. Thus, the BCMIX for the posterior mean at time t is

$$\hat{\theta}_{t|n} = \sum_{i \in \mathcal{D}_t(p), j \in \tilde{\mathcal{D}}_{t+1}(p)} \tilde{w}_{ijt} \overline{X}_{i,j} \tag{3.2}$$

3.2 Programming in C language

As mentioned in the beginning of this chapter, in order to speed up the calculation, we implement the simulation entirely with coding in C language. Coding the Bayesian model is quite similar in both "R" and C since the there is no packages available even in "R". However, for the part of eigen-decompostion, it's different. Bunch of functions make it extremely easy to get eigen-decomposition in "R" while we have to code all of it ourselves in C language. The algorithms used for eigen-decomposition will be briefly introduced. In the end of this section, some difficulties coding in C language will be mentioned along with some possible solutions proposed.

3.2.1 Householder Transformation

Two algorithms for eigen-decomposition are used while programming in C language. One is only able to provide all eigenvalues and eigenvectors at once, while another one is for approximating a few top eigenvalues and eigenvectors. Both algorithms depend on the Householder Triangularization performing the QR factorization to a matrix. The QR decomposition of a matrix is a decomposition of a matrix A into a product

$$A = QR$$

where Q is an orthogonal matrix and R is an upper triangular matrix.

The Householder Reflection is used to implement the QR decomposition. The algorithm can be briefly described as the following. Suppose A is a $m \times n$ matrix with $m \geq n$. We use $|| \cdot ||$ to denote the Euclidean norm and $\mathbf{I}$ to denote an $m \times m$ identity matrix. We firstly let $\mathbf{y}$ to be the first column of matrix A. Then, we set

$$\mathbf{u} = \mathbf{y} - \beta \mathbf{e}_1$$

where $\mathbf{e}_1$ is the vector $(1, 0, ..., 0)^T$ and the scalar β can be calculated from $||\mathbf{y}|| = |\beta|$ and the sign of β is the same as the first element of $\mathbf{y}$. Then, matrix Q can be obtained by

$$\mathbf{v} = \frac{\mathbf{u}}{||\mathbf{u}||},$$
$$Q = \mathbf{I} - 2\mathbf{v}\mathbf{v}^T.$$

Q is usually referred as an $m \times m$ Householder matrix and

$$Q\mathbf{y} = (\beta, 0, ..., 0)^T.$$

Gradually, we are able to transform A into upper triangular form. If we let $\mathbf{y}$ be the first column of A, we can rewrite the Q we have obtained above as Q_1 and the β as β_1. Then,

we have

$$Q_1 A = \left(\begin{array}{c|c} \beta_1 & * \cdots * \\ \hline \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} & A' \end{array}\right)$$

The previous step is repeated for A' (obtained from $Q_1 A$ by removing the first column and first row) above, giving a Householder matrix Q_2'. However, Q_2' is smaller than Q_1 but we want the whole operation is on $Q_1 A$ instead of A'. Therefore, we have to expand it toward the upper left by the following:

$$Q_k = \begin{pmatrix} \boldsymbol{I}_{k-1} & 0 \\ 0 & Q_k' \end{pmatrix}$$

After p iterations, $p = min(n, m-1)$, the matrix R is computed as

$$R = Q_p ... Q_2 Q_1 A$$

which is an upper triangular matrix. The final orthogonal matrix Q can be obtained by

$$Q = Q_1^T Q_2^T ... Q_p^T.$$

3.2.2 Numerical Eigen-Decomposition

With QR decomposition to a matrix A, the QR algorithm can be applied in order to solve the eigen-decomposition problem. Suppose in the t^{th} iteration, $A_t = Q_t R_t$, then, A_{t+1} is constructed by $A_{t+1} = R_t Q_t$ and we perform the QR decomposition to A_{t+1} again. Note that

$$A_{t+1} = R_t Q_t = Q_t^T Q_t R_t Q_t = Q_t^T A_t Q_t = Q_t^{-1} A_t Q_t,$$

Thus all $\{A_t\}$ are similar with each other and they have the same eigenvalues. The whole process is repeated until some certain condition is satisfied. Eigenvalues will appear on the diagonal of matrix R and eigenvectors will appear as the columns of matrix Q.

When the dimensionality is high, usually, we care only about several top eigenvalues which accounts the most variation. In this situation, it's really time and memory consuming to compute all eigenvalues since the number of eigenvalues (eigenvectors) is the same as the dimension. As an alternative solution, we suggest the "Bisection" method to numerically calculate a group of eigenvalues located in a specified interval. The "Bisection" method depends on the tridiagonalization of a symmetric matrix.

At the beginning, we use "Householder Reduction" to reduce a matrix into Hessenberg form. Since the matrix we deal with here is the covariance matrix which is symmetric and positive definite, it is guaranteed can be reduced into tridiagonal form. A little bit different from the "Householder Transformation" mentioned in last section, here we leave the first row of A unchanged and started from the second row. Particularly,

$$Q_1 A = \left(\begin{array}{c|c} * & * * \cdots * * \\ * & * * \cdots * * \\ \hline 0 & \\ \vdots & A' \\ 0 & \end{array}\right)$$

Then, continue the same steps in "Householder Transformation", we can get a series of matrix, $Q_1, ...Q_{m-2}$, such that,

$$H = Q_{m-2}^T ... Q_2^T Q_1^T A Q_1 Q_2 ... Q_{m-2}$$

where matrix H is in Hessenberg form:

$$\begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{pmatrix}$$

Since A here is symmetric, $Q^T AQ$ is also symmetric, and any symmetric Hessenberg matrix is tridiagonal:

$$\begin{pmatrix} * & * & 0 & 0 & 0 \\ * & * & * & 0 & 0 \\ 0 & * & * & * & 0 \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{pmatrix}$$

Now, given a symmetric matrix $A \in \mathcal{R}^{m \times m}$, we assume that A has already been reduced into the tridiagnoal form. Let $A^{(1)}, ..., A^{(m)}$ denote the principals in dimensions $1, ..., m$ of matrix A. The k^{th} principal is a sub-matrix formed by the first k^{th} rows and first k^{th} columns of A. The eigenvalues of $A^{(k)}$ are distinct and let them ordered by $\lambda_1^{(k)} < \lambda_2^{(k)} < ... < \lambda_k^{(k)}$. The important property is that these eigenvalues *strictly interlace*, satisfying the following inequalities

$$\lambda_j^{(k+1)} < \lambda_j^{(k)} < \lambda_{j+1}^{(k+1)} \tag{3.3}$$

for $k = 1, 2, ..., m-1$ and $j = 1, 2, ..., k-1$. This property makes the "Bisection" method powerful. Figure 3.1 is an illustration of the property: The eigenvalues of $A^{(k)}$ interlace those of $A^{(k+1)}$.

Taking the advantage of the *strictly interlace* property, we are able to count the exact number of eigenvalues in any particular interval. Let's look at a simple example.

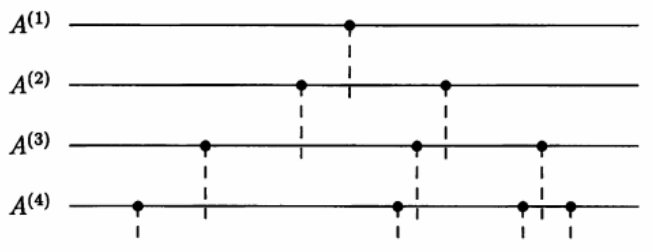

FIGURE 3.1: Illustration of the Property of Strictly Interlace

We construct a 4×4 tridiagonal matrix as

$$A = \begin{pmatrix} 1 & 1 & & \\ 1 & 0 & 1 & \\ & 1 & 2 & 1 \\ & & 1 & -1 \end{pmatrix}$$

The determinant of each principal can be easily calculated:

$$det(A^{(1)}) = 1, \quad det(A^{(2)}) = -1, \quad det(A^{(3)}) = -3, \quad det(A^{(4)}) = 4,$$

The eigenvalue of $A^{(1)}$ is obviously positive. From the *strictly interlace* property, we know that the bigger eigenvalue of $A^{(2)}$ is bigger than the eigenvalue of $A^{(1)}$ while the smaller eigenvalue of $A^{(2)}$ is smaller than the eigenvalue of $A^{(1)}$. Thus, the bigger eigenvalue of $A^{(2)}$ must be positive. The determinant of $A^{(2)}$ is negative thus the smaller eigenvalue is negative. Continue in this way, we can find that $A^{(3)}$ has one negative eigenvalue, and $A^{(4)}$ has two negative eigenvalues. In general, for any symmetric tridiagonal matrix $A \in \mathcal{R}^{m \times m}$, the number of negative eigenvalues equals to the number of sign changes in the "Sturm Sequence" sequence

$$1, det(A^{(1)}), det(A^{(2)}), ..., det(A^{(m)}). \tag{3.4}$$

If there exists zero determinants, we define "sign change" as a change from positive or zero to negative or from negative or zero to positive but not from positive or negative

to zero. Under this definition, the statement about the "Sturm Sequence" is still true. By shifting A with aI, we can count the number of eigenvalues located in any interval $(-\infty, a)$. Since to count the number of negative eigenvalues by the "Sturm Sequence"

$$1 - a, det(A^{(1)} - aI^{(1)}), ..., det(A^{(m)} - aI^{(m)})$$

is equivalent to count the number of eigenvalues of A located in the interval $(-\infty, a)$. Hence, we can calculate the number of eigenvalues in any interval $[a, b)$ by subtracting the number of eigenvalues in $(-\infty, a)$ from the number of eigenvalues in $(-\infty, b)$, for $a < b$.

Another important observation makes the bisection algorithm efficient in practical calculation. That is there is a recursive formula for the determinants of those principals of matrix A. Suppose after tridiagonalization, A has the form

$$A = \begin{pmatrix} a_1 & b_1 & & & \\ b_1 & a_2 & b_2 & & \\ & b_2 & a_3 & ... & \\ & & ... & ... & b_{m-1} \\ & & & b_{m-1} & a_m \end{pmatrix} \tag{3.5}$$

The recursive relationship can be described by the following expression:

$$det(A^{(k)}) = a_k det(A^{(k-1)}) - b_{k-1}^2 det(A^{(k-2)}). \tag{3.6}$$

By introducing the shift yI and denoting $q^{(k)}(y) = det(A^{(k)} - yI)$, the recursive relationship 3.6 can be rewritten as

$$q^{(k)}(y) = (a_k - y)q^{(k-1)}(y) - b_{k-1}^2 q^{(k-2)}(y). \tag{3.7}$$

In order to make this recursive relationship valid for all $k = 1, 2, ..., m$, we define $q^{(-1)}(y) = 0$ and $q^{(0)}(y) = 1$.

By applying 3.7 for a proper values of y and counting sign changes along the way, the bisection algorithm will locate eigenvalues in arbitrarily small intervals which means an approximation to these eigenvalues. The computational cost is in the same order as $O(m)$ for each evaluation of the sequence. If we only need a few top eigenvalues, it will be much more efficient than the QR algorithm with computational complexity $O(m^2)$.

Note that, the bisection algorithm only provide us a way to calculate eigenvalues. The algorithm used to calculate eigenvectors is called "Inverse Iteration" which can be briefly described as the following:

Algorithm 3.1 Inverse Iteration

Initialize an unit vector $v^{(0)}$ arbitrarily,

for $k = 1, 2...$

Solve the equation $(A - \mu I)w = v^{(k-1)}$ for w

$v^{(k)} = w/||w||$

$\lambda^{(k)} = (v^{(k)})^T A v^{(k)}$

This algorithm relies on the fact that: for any $\mu \in \mathcal{R}$ not equal to any eigenvalue of A, $(A - \mu I)^{-1}$ has the same eigenvectors as A, and if we denote $\{\lambda_j\}$ as the eigenvalues of A, $(A - \mu I)^{-1}$ has the eigenvalues $\{(\lambda_j - \mu)^{-1}\}$. Our case is much easier since we have already obtained those eigenvalues we want from bisection algorithm. Thus, we only have to conduct one step ($k = 1$) of "Inverse Iteration" to obtain the corresponding eigenvector. When the μ equals eigenvalue, $A - \mu I$ is singular, however, it won't cause any trouble in the implementation.

As the calculation of inverse of a matrix is involved in the "Inverse Iteration" algorithms, at last, we briefly introduce the algorithms for calculating the inverse of matrix. We firstly conduct a LU decomposition to matrix $A \in \mathcal{R}^{m \times m}$:

$$A = LU$$

where L and U are lower and upper triangular matrix respectively. This is implemented by "Gaussian Elimination" described below.

Algorithm 3.2 Gaussian Elimination

Let $U = A$, $L = I$, $P = I$

for $k = 1$ to $m - 1$

Select the $i \geq k$ which maximizes $|u_{ik}|$

$u_{k,k:m} \leftrightarrow u_{i,k:m}$ (switch rows)

$l_{k,1:k} \leftrightarrow l_{i,1:k-1}$ (switch rows)

$p_{k,:} \leftrightarrow p_{i,:}$ (switch rows)

for $j = k + 1$ **to** m

$l_{jk} = u_{jk}/u_{kk}$

$u_{j,k:m} = u_{j,k:m} - l_{jk}u_{k,k:m}$

After the LU decomposition of A, the problem of invert a matrix can be written as:

$$AA^{-1} = (LU)A^{-1} = I$$

If we let x_i and b_i denote the columns of A^{-1} and I respectively, for $1 \leq i \leq m$. Then, the problem can be expressed as solving the following linear equation system:

$$(LU)x_i = b_i, \quad 1 \leq i \leq m.$$

For each i, the linear equation can be solved by: let $y_i = Ux_i$ and firstly solve the equation

$$Ly_i = b_i$$

by the well known process: forward substitution. At last, we get each column x_i of the inverse matrix A^{-1} with solving the linear equations

$$Ux_i = y_i$$

by backward substitution.

3.2.3 Difficulties

C language has a much faster computational speed than R though, as a trade-off, it has some drawbacks. One is that the memory size referred by a variable is very limited. In the process of calculation, bunch of big values need to be stored, however, these values are out of bound. In another word, the variables defined in C can't store such big values. This problem may lead to a wrong results.

A common solution to this problem is to take logarithm of the value before storing it. The advantage of logarithm function is that, it reduces wide-ranging quantities to smaller scopes. Figure 3.2 is an illustration of logarithm function (2 based). The logarithm function is a monotone function, however, it slows down the values getting big. From the figure we can see $log_2(8)$ is just around 3. If the value is big, the effect is much more significant. For example, $log_2(8^{10}) = 10log_2(8) \approx 30$, which can be easily handled by a double variable in C language, while 8^{10} far beyond what a double variable can handle. After getting all calculations done, we can simply convert them into true values by applying exponential function.

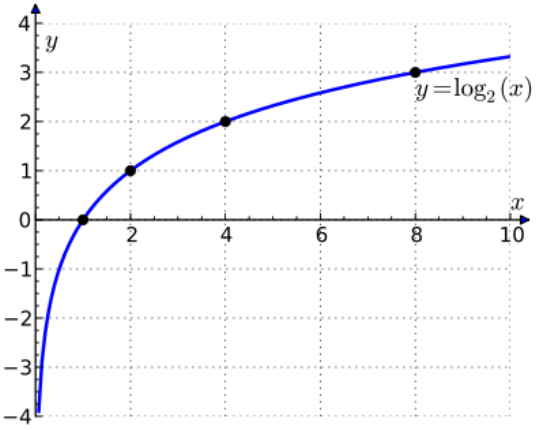

FIGURE 3.2: Illustration of Logarithm Function

One small issue need mention is how to calculate $log(a+b)$ only depending on $log(a)$ and $log(b)$. Since in the process of calculation, all values are stored in the form of log. The appearance of value a or b are avoided due to the risk that they are out of bound. Thus, we recommend the following formula

$$log(a+b) = log(\frac{a+b}{b}b) = log(\frac{a}{b}+1) + log(b) = log(exp(log(a) - log(b)) + 1) + log(b)$$

in which b is always chosen as the bigger one between the two. Then we can promise $log(a) - log(b)$ is less or equal to zero so that $exp(log(a) - log(b))$ won't be greater than 1. If b is much bigger than a, the value $exp(log(a) - log(b))$ will be calculated as 0.

Another difficulty is also related to the memory size. In the process of calculation, there are bunch of values need to be stored. In order to make it easier to be retrieved by the algorithm, those values are stored in matrices. In one dimensional case, they are ordinary matrices. However, in multiple dimensional space, even parameters are matrices. What we do is using four dimensional arrays to store them. In another word, matrices whose every element is a matrix. When the dimension is not high (less than 20) and the sample size is not big (less than 1000), it's okay for the memory size to handle all the matrices. However, when the dimension is high, the memory size is not capable to store all the matrices.

Two potential solutions to this issue. One is the trade-off between time and space which means in order to save the space of memory, we sacrifice the speed of calculation. For example, in the original implementation, we store a series of matrices in a four dimensional array so that the following calculations are able to retrieve them directly. To save the memory space, we no longer store those matrices any more but calculate them whenever needed. For each matrix, it might be re-calculated couple of times, and that is why storing them and retrieving them directly is fast. Obviously, it's a trade-off between the space and time. Another possible solution is to store those matrices on the hard drive. As we know, the space of hard drive is much bigger than the memory. Our

recommendation is, write every matrix into a text file named as its index in the original four dimensional array. Since the name of every text file is right the same as its index in the original four dimensional array, actually we store these text files (matrices) into an abstract four dimensional array on the hard drive. The computation afterward can read any text file from the hard drive whenever necessary. The second solution proposed might be better than the first one if the computation of matrices is time consuming. In that case, read a text file from hard drive is much faster than re-conducting calculation on a matrix.

3.3 Implementation

So far, we have presented all models used, mainly, the Bayes model and PCA model and corresponding algorithms. Before numerical studies, we'd like to provide the details of implementations, which may make the process of computation clear. Let's first look at the implementation of Bayes Model.

Step 1. Calculate the parameters of posterior distribution. As mentioned in section 2.2.2, the posterior distribution given observation $x_i, ..., x_j$ $(1 \leq i \leq j \leq n)$ depends on two parameters:$\Psi_{ij} = \Psi + \sum_{k=i}^{j} \tilde{x}_k^T \tilde{x}_k$ and $n_{0(ij)} = n_0 + (j - i + 1)$. A four dimensional array is used to store Ψ_{ij}:

$$\psi = \begin{pmatrix} \Psi_{11} & & & & \\ \Psi_{12} & \Psi_{22} & & & \\ \Psi_{13} & \Psi_{23} & \Psi_{33} & & \\ ... & ... & ... & ... & ... \\ \Psi_{1n} & \Psi_{2n} & ... & \Psi_{n-1,n} & \Psi_{nn} \end{pmatrix}$$

A two dimensional array (a matrix) is used to store $n_{0(ij)}$:

$$N = \begin{pmatrix} n_{0(11)} & & & & \\ n_{0(12)} & n_{0(22)} & & & \\ n_{0(13)} & n_{0(23)} & n_{0(33)} & & \\ ... & ... & ... & ... & ... \\ n_{0(1n)} & n_{0(2n)} & ... & n_{0(n-1,n)} & n_{0(nn)} \end{pmatrix}$$

Then, we can get the posterior mean based on (2.28) suggested to be stored in a four dimensional array:

$$\mu = \begin{pmatrix} \overline{X}_{11} & & & & \\ \overline{X}_{12} & \overline{X}_{22} & & & \\ \overline{X}_{13} & \overline{X}_{23} & \overline{X}_{33} & & \\ ... & ... & ... & ... & ... \\ \overline{X}_{1n} & \overline{X}_{2n} & ... & \overline{X}_{n-1,n} & \overline{X}_{nn} \end{pmatrix}$$

$log(\pi_{00})$ and $log(\pi_{i,j})$ can be calculated according to formulas (2.26) and (2.27),

$$log(\pi_{0,0}) = \frac{n_0}{2} log(|\Psi|) - (\frac{n_0 m}{2} - 1) log(2) - log(\Gamma_m(\frac{n_0}{2}))$$

$$log(\pi_{i,j}) = \frac{n_{0(ij)}}{2} log(|\Psi_{ij}|) - (\frac{n_{0(ij)} m}{2} - 1) log(2) - log(\Gamma_m(\frac{n_{0(ij)}}{2}))$$

Similarly, $log(\pi_{i,j})$ are suggested to be stored in a two dimensional array (a matrix):

$$log(\Pi) = \begin{pmatrix} log(\pi_{1,1}) & & & & \\ log(\pi_{1,2}) & log(\pi_{2,2}) & & & \\ log(\pi_{1,3}) & log(\pi_{2,3}) & log(\pi_{3,3}) & & \\ ... & ... & ... & ... & ... \\ log(\pi_{1,n}) & log(\pi_{2,n}) & ... & log(\pi_{n-1,n}) & log(\pi_{n,n}) \end{pmatrix}$$

The advantage of storing all these values this way is, there is a one to one mapping

between the location in the matrix and the index of the parameter. Specifically, for $1 \leq i \leq j \leq n$,

$$\Psi_{ij} = \psi_{ji}, \quad n_{0(ij)} = N_{ji}, \quad \overline{X}_{ij} = \mu_{ji}, \quad log(\pi_{i,j}) = log(\Pi)_{j,i}.$$

Step 2. Forward Filter. Calculate $log(p_{it})$ recursively based on $log(p^*_{it})$, for $1 \leq i \leq t \leq n$.We also suggest use two matrices to store $log(p_{it})$ and $log(p^*_{it})$ respectively. As mentioned in section 2.1.1, we start with

$$log(p^*_{11}) = log(\pi_{0,0}) - log(\pi_{1,1}) = log(\pi_{0,0}) - log(\Pi)_{1,1}$$

and store it as

$$log(P^*) = \begin{pmatrix} log(p^*_{11}) & & \\ & & \\ & & \\ & & \end{pmatrix}.$$

Then, by normalizing p^*_{11} with the logarithm form,

$$log(p_{11}) = log(p^*_{11}) - log(p^*_{11}) = 0$$

and also store as

$$log(P) = \begin{pmatrix} log(p_{11}) & & \\ & & \\ & & \\ & & \end{pmatrix}.$$

Follow the recursive process, next we calculate the second row of matrix $log(P^*)$ based on the first row of matrix $log(P)$:

$$log(p^*_{12}) = log(1-p)+log(p_{11})+log(\pi_{1,1})-log(\pi_{1,2}) = log(1-p)+log(P)_{11}+log(\Pi)_{1,1}-log(\Pi)_{2,1}$$

$$log(p^*_{22}) = log(p) + log(\pi_{0,0}) - log(\pi_{2,2}) = log(p) + log(\pi_{0,0}) - log(\Pi)_{2,2}$$

Then, the matrix $log(P^*)$ can be updated:

$$log(P^*) = \begin{pmatrix} log(p^*_{11}) & \\ log(p^*_{12}) & log(p^*_{22}) \\ & \\ & \\ & \end{pmatrix}.$$

By normalization, we can get,

$$log(p_{12}) = log(p^*_{12}) - log(p^*_{12} + p^*_{22}), \quad log(p_{22}) = log(p^*_{22}) - log(p^*_{12} + p^*_{22})$$

Note that, the way to calculate $log(a+b)$ has been mentioned in section 3.2.3. The second row of $log(P)$ now can be updated:

$$log(P) = \begin{pmatrix} log(p_{11}) & \\ log(p_{12}) & log(p_{22}) \\ & \\ & \\ & \end{pmatrix}.$$

By continuing the process, eventually, we can have two lower triangular matrices:

$$log(P^*) = \begin{pmatrix} log(p^*_{1,1}) & & & & \\ log(p^*_{1,2}) & log(p^*_{2,2}) & & & \\ log(p^*_{1,3}) & log(p^*_{2,3}) & log(p^*_{3,3}) & & \\ ... & ... & ... & ... & \\ log(p^*_{1,n}) & log(p^*_{2,n}) & ... & log(p^*_{n-1,n}) & log(p^*_{n,n}) \end{pmatrix}$$

$$log(P) = \begin{pmatrix} log(p_{1,1}) & & & & \\ log(p_{1,2}) & log(p_{2,2}) & & & \\ log(p_{1,3}) & log(p_{2,3}) & log(p_{3,3}) & & \\ ... & ... & ... & ... & \\ log(p_{1,n}) & log(p_{2,n}) & ... & log(p_{n-1,n}) & log(p_{n,n}) \end{pmatrix}$$

Step 3. Estimate the parameter p. In section 2.1.4, we have suggested the value for parameter a_0 and μ_0, which means the only parameter we need to estimate is p. With unknown p, all values and parameters in **Step1** and **Step2** are functions of p. As suggested in 2.1.4, the log-likelihood function can be written as:

$$l(p) = \sum_{t=1}^{n} log(\sum_{i=1}^{t} p^*_{it})$$

Each t corresponds to the row of the matrix $log(P^*)$. Based on each row of $log(P^*)$, $log(\sum_{i=1}^{t} p^*_{it})$ can be calculated with the way suggested in section 3.2.3.

Numerically, we have to search a bunch of values of p to find one maximize the log-likelihood function. The grid of values of p has been suggested in section 2.1.4. Practically, with every value of p from the grid, we do **Step1** to **Step3** getting a corresponding value of log-likelihood function and we choose the p corresponding to the biggest log-likelihood.

After obtaining the proper value of p, we restart from **Step1** but skip **Step3** and go to **Step4** directly.

Step 4. Backward Filter. Calculate $log(q_{t,tj})$ recursively based on $log(q^*_{t,j})$, for $1 \leq t \leq j \leq n$. Note that, the notation here is a little bit tricky, the $q_{t,j}$ and $q^*_{t,j}$ equal to $q_{j,t}$ and $q^*_{j,t}$. We switch the position of j and t since $t \leq j$. Similarly, we use two matrices to store $log(q_{t,j})$ and $log(q^*_{t,j})$ respectively. As mentioned in section 2.1.2, we start with

$$log(q^*_{n,n}) = log(\pi_{0,0}) - log(\pi_{n,n})$$

and store it as

$$log(Q^*) = \begin{pmatrix} & & \\ & & \\ & & \\ & & log(q^*_{n,n}) \end{pmatrix}.$$

By normalization

$$log(q_{n,n}) = log(q^*_{n,n}) - log(q^*_{n,n}) = 0$$

and also store as

$$log(Q) = \begin{pmatrix} & & \\ & & \\ & & \\ & & log(q_{n,n}) \end{pmatrix}.$$

Again, follow the recursive process, next we calculate the $(n-1)th$ column of matrix $log(Q^*)$ based on the nth column of matrix $log(Q)$:

$$log(q^*_{n-1,n}) = log(1-p) + log(q_{n,n}) + log(\pi_{n,n}) - log(\pi_{n-1,n})$$

$$log(q^*_{n-1,n-1}) = log(p) + log(\pi_{0,0}) - log(\pi_{n-1,n-1})$$

Then, the matrix $log(Q^*)$ can be updated:

$$log(Q^*) = \begin{pmatrix} & & \\ & & \\ & log(q^*_{n-1,n-1}) & \\ & log(q^*_{n-1,n}) & log(q^*_{n,n}) \end{pmatrix}.$$

By normalization:

$$log(q_{n-1,n}) = log(q^*_{n-1,n}) - log(q^*_{n-1,n} + q^*_{n-1,n-1}),$$

$$log(q_{n-1,n-1}) = log(q^*_{n-1,n-1}) - log(q^*_{n-1,n} + q^*_{n-1,n-1}).$$

and the matrix $log(Q)$ can be updated:

$$log(Q) = \begin{pmatrix} & & \\ & & \\ & log(q_{n-1,n-1}) & \\ & log(q_{n-1,n}) & log(q_{n,n}) \end{pmatrix}.$$

By continuing doing so, eventually we can get two low triangular matrix:

$$log(Q^*) = \begin{pmatrix} log(q^*_{1,1}) & & & & \\ log(q^*_{1,2}) & ... & & & \\ ... & ... & log(q^*_{n-2,n-2}) & & \\ log(q^*_{1,n-1}) & ... & log(q^*_{n-2,n-1}) & log(q^*_{n-1,n-1}) & \\ log(q^*_{1,n}) & ... & log(q^*_{n-2,n}) & log(q^*_{n-1,n}) & log(q^*_{n,n}) \end{pmatrix}.$$

$$log(Q) = \begin{pmatrix} log(q_{1,1}) & & & & \\ log(q_{1,2}) & ... & & & \\ ... & ... & log(q_{n-2,n-2}) & & \\ log(q_{1,n-1}) & ... & log(q_{n-2,n-1}) & log(q_{n-1,n-1}) & \\ log(q_{1,n}) & ... & log(q_{n-2,n}) & log(q_{n-1,n}) & log(q_{n,n}) \end{pmatrix}.$$

Step 5. Get smooth. In this last step, we calculate the weight w_{ijt} for the final weighted average. Firstly, we calculate $log(w^*_{ijt})$ according to formula (2.16), for $i \leq t = j$:

$$log(w^*_{ijt}) = log(p) + log(p_{it}) = log(p) + log(P)_{ti},$$

for $i \leq t < j$:

$$\begin{aligned} log(w^*_{ijt}) &= log(1-p) + log(p_{it}) + log(q_{j,t+1}) + log(\pi_{it}) + log(\pi_{t+1,j}) - log(\pi_{ij}) - log(\pi_{00}) \\ &= log(1-p) + log(P)_{ti} + log(Q)_{j,t+1} + log(\Pi)_{ti} + log(\Pi)_{j,t+1} - log(\Pi)_{ji} - log(\pi_{00}) \end{aligned}$$

We also suggest a $n \times n$ matrix $log(W^*)$ to store $log(w^*_{ijt})$. However, for each specific t, we select the t^{th} to n^{th} row and 1st to t^{th} column of $log(W^*)$ as a sub-matrix to store $log(w^*_{ijt})$. For example, starting from $t = 1$, we select the first column of $log(W^*)$ to store $log(w^*_{1j1})$, with $1 \leq j \leq n$:

$$log(W^*) = \begin{pmatrix} log(w^*_{111}) & \\ log(w^*_{121}) & \\ ... & \\ log(w^*_{1,n-1,1}) & \\ log(w^*_{1,n,1}) & \end{pmatrix}.$$

Next, when $t = 2$, we select the $2nd$ to n^{th} row and first two columns of $log(W^*)$ to store $log(w^*_{ij2})$, with $1 \leq i \leq 2 \leq j \leq n$:

$$log(W^*) = \begin{pmatrix} & \\ log(w^*_{122}) & log(w^*_{222}) \\ log(w^*_{132}) & log(w^*_{232}) \\ ... & ... \\ log(w^*_{1,n-1,2}) & log(w^*_{2,n-1,2}) \\ log(w^*_{1,n,2}) & log(w^*_{2,n,2}) \end{pmatrix}.$$

Every time with the value of t increasing 1, the length of row of the sub-matrix decrease 1 while the length of column of the sub-matrix increase 1. Thus, in the end with $t = n$, we have:

$$log(W^*) = \begin{pmatrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ log(w^*_{1,n,n}) & log(w^*_{2,n,n}) & ... & log(w^*_{n-1,n,n}) & log(w^*_{n,n,n}) \end{pmatrix}.$$

By normalizing w^*_{ijt}, we can get the weight w_{ijt}:

$$log(P_t) = log(p + \sum\nolimits_{1 \leq i \leq t \leq j \leq n} w^*_{ijt}), \quad w_{ijt} = exp(log(w^*_{ijt}) - log(P_t)).$$

We store w_{ijt} in the matrix named W with the same structure with $log(W^*)$. For each t, the way to store w_{ijt} in W is totally the same as $log(w^*_{ijt})$. Note that, we don't need to keep w_{ijt} in the log form since after normalization, each w_{ijt} will locate in $[0, 1]$. For extremely small value, it will be stored as 0.

According to (2.17), the final estimation at each time t can be obtained. one advantage of the way storing w_{ijt} is, for each pair (i, j) , the physical location of w_{ijt} is the same as $\overline{X}_{i,j}$, which will means we have a clear expression for index.

From the steps above, it's also easy to notice that the computational complexity will be extremely high when the sample size n is very big. Specifically, the number of weights w_{ijt} is equal to $t(n-t+1)$, for $1 \leq t \leq n$. By a simple inequality:

$$xy \leq (\frac{x+y}{2})^2,$$

we know that the equality holds when $x = y$. Thus, the biggest number of weights will occur when $t = n - t$, which is $t = n/2$. Since we are considering the case with big n, without losing generality, assuming n is even, then the biggest number of weights is $n^2/4$. That means, the number of weights will increase as the same order as n^2.

As mentioned before, BCMIX solves this problem reducing the computational complexity by bounding the maximum number of weights. The first step of BCMIX is totally the same as the **Step 1** described above. Starting from **Step 2**, it's different.

Step 2. Forward Filter (BCMIX). Two parameters, KK and MM have to be set up in advance. In the literature, it is suggested that $KK = 20$ and $MM = 10$. The matrices $log(P)$ and $log(P^*)$ are no longer $n \times n$ square matrices, but with dimensionality $n \times KK$ and $n \times (KK + 1)$ respectively. Recall the recursive process filling the matrices $log(P)$ and $log(P^*)$, it's obvious that the first KK rows will be generated the same as in

the original **Step 2**.

$$log(P) = \begin{array}{c} \\ 1 \\ 2 \\ \\ \text{KK} \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \dots & \text{KK} \end{array} \\ \begin{pmatrix} * & & & \\ * & * & & \\ \vdots & \vdots & \ddots & \\ * & * & \dots & * \\ & & & \\ & & & \end{pmatrix} \end{array}$$

$$log(P^*) = \begin{array}{c} \\ 1 \\ 2 \\ \\ \text{KK} \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & \dots & \text{KK} & \text{KK+1} \end{array} \\ \begin{pmatrix} * & & & & \\ * & * & & & \\ \vdots & \vdots & \ddots & & \\ * & * & \dots & * & \\ & & & & \\ & & & & \end{pmatrix} \end{array}$$

Besides, we use a matrix called D to store the index. It's also a $n \times KK$ matrix and its first KK rows are stored in this way:

$$D = \begin{array}{c} \\ 1 \\ 2 \\ \\ \text{KK} \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \dots & \text{KK} \end{array} \\ \begin{pmatrix} 1 & & & \\ 1 & 2 & & \\ \vdots & \vdots & \ddots & \\ 1 & 2 & \dots & \text{KK} \\ & & & \\ & & & \end{pmatrix} \end{array}$$

In fact, for a particular value of t, the index indicates the time up to time t that are involved in "Forward Filter" and KK is the maximum number allowed. Thus, for $1 \leq$

$i \leq t \leq KK$, all time points before t are used. For example, the first row indicates $t = 1$. Only $i = 1 \leq t = 1$ can be used, therefore, the first row of D is 1. The second row indicates $t = 2$ and $i = \{1, 2\}$ can be used, thus 1 and 2 are stored in the second row of D. We keep this going on until $t = KK$ with all time points $1 \leq i \leq t$ can be used but no more allowed. Thus, the KK^{th} row of D stores $\{1, 2, ..., KK\}$.

Starting from $t = KK+1$, we firstly calculate the $(KK+1)^{th}$ row of $log(P^*)$. We use the index on the KKth row of D combining with a single index $(KK + 1)$ to indicates the $KK + 1$ columns in $log(\Pi)$ involved in calculating the $KK + 1$th row of $log(P^*)$. specifically, with

$$\{log(P)_{KK,1}, log(P)_{KK,2}, ..., log(P)_{KK,KK}\},$$

$$\{log(\Pi)_{KK,D_{KK,1}}, log(\Pi)_{KK,D_{KK,2}}, ..., log(\Pi)_{KK,D_{KK,KK}}, log(\Pi)_{KK,KK+1}\},$$

we can get the $(KK + 1)^{th}$ row of $log(P^*)$:

$$log(P^*) = \begin{array}{c} \\ 1 \\ 2 \\ \\ \text{KK} \\ \text{KK+1} \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & \dots & \text{KK} & \text{KK+1} \end{array} \\ \begin{pmatrix} * & & & & & \\ * & * & & & & \\ \vdots & \vdots & \ddots & & & \\ * & * & \dots & * & & \\ * & * & \dots & * & * & \\ & & & & & \\ & & & & & \end{pmatrix} \end{array}$$

Now, we can explain why the matrix $log(P^*)$ has one more column than $log(P)$. This is because $log(P^*)$ need to contain one more nearest time point (here $t = KK + 1$), so that we can start to delete one. Next, we pick the first MM elements on the $(KK + 1)$th row of $log(P^*)$ and delete the smallest one (if two or more smallest value exit, we delete the leftmost one). For example, we delete the l^{th} $(1 \leq l \leq MM)$ element, then the $(KK+1)^{th}$

row of $log(P)$ can be filled by normalizing

$$log(P^*)_{KK+1,1}, ..., log(P^*)_{KK+1,l-1}, log(P^*)_{KK+1,l+1}, ..., log(P^*)_{KK+1,KK+1}$$

and updated as

$$log(P) = \begin{array}{c} \\ 1 \\ 2 \\ \\ \text{KK} \\ \text{KK+1} \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \dots & \text{KK} \end{array} \\ \begin{pmatrix} * & & & \\ * & * & & \\ \vdots & \vdots & \ddots & \\ * & * & \dots & * \\ * & * & \dots & * \\ & & & \\ & & & \end{pmatrix} \end{array}$$

The $(KK+1)^{th}$ row of D is

$$D_{KK,1}, ..., D_{KK,l-1}, D_{KK,l+1}, ..., D_{KK,KK}, KK+1$$

and the structure is updated as

$$D = \begin{array}{c} \\ 1 \\ 2 \\ \\ \text{KK} \\ \text{KK+1} \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & \dots & \text{KK} & \end{array} \\ \begin{pmatrix} * & & & & \\ * & * & & & \\ \vdots & \vdots & \ddots & & \\ * & * & \dots & * & \\ * & * & \dots & * & * \\ & & & & \\ & & & & \end{pmatrix} \end{array}$$

Continue this process, the $(KK+1)^{th}$ to n^{th} row of $log(P^*)$, $log(P)$, and D can be fully filled.

Step 3. Estimate the parameter p. Compared with the construction of log-likelihood function suggested in the **Step 3** of original Bayes model, the only difference is all elements are taken from the matrix $log(P^*)$ shrunken by BCMIX in Forward Filter.

Step 4. Backward Filter (BCMIX). The Backward Filter with BCMIX is very similar to that of Forward Filter (BCMIX) described above. The three matrices finally will have the structure as the following:

$$log(Q^*) = \begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ \text{n- KK} \\ \text{n- (KK-1)} \\ \\ n-1 \\ n \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & \dots & \text{KK} & \text{KK+1} \end{array} \\ \begin{pmatrix} * & * & \dots & * & * \\ * & * & \dots & * & * \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ * & * & \dots & * & * \\ * & * & \dots & * & \\ \vdots & \vdots & \ddots & & \\ * & * & & & \\ * & & & & \end{pmatrix} \end{array}$$

$$log(Q) = \begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ \text{n- (KK-1)} \\ \\ n-1 \\ n \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \dots & \text{KK} \end{array} \\ \begin{pmatrix} * & * & \dots & * \\ * & * & \dots & * \\ \vdots & \vdots & \vdots & \vdots \\ * & * & \dots & * \\ \vdots & \vdots & \ddots & \\ * & * & & \\ * & & & \end{pmatrix} \end{array}$$

$$E = \begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ \text{n- (KK-1)} \\ \\ n-1 \\ n \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \dots & \text{KK} \end{array} \\ \begin{pmatrix} * & * & \dots & * \\ * & * & \dots & * \\ \vdots & \vdots & \vdots & \vdots \\ * & * & \dots & * \\ \vdots & \vdots & \ddots & \\ * & * & & \\ * & & & \end{pmatrix} \end{array}$$

It can be simply regarded as a time-reversed forward filter if we start from the first column of n^{th} row. All recursive calculation are totally the same as that in forward filter. Thus, we are not going to give the details of calculation but only provide the structure of matrices.

Step 5. Get smooth (BCMIX). In this last step, the shrunken matrices $log(P)$ and $log(Q)$ are used to calculate the weights w_{ijt}. According to (2.16), the calculation also need the elements in $log(\Pi)$ which is not a shrunken matrix. In order to select those elements corresponding to the shrunken $log(P)$ and $log(Q)$, we use the index matrices D and E to do so. For example, for a certain time point $1 \leq t \leq n$, firstly, we use a vector $find$ to store all non-zero elements on t^{th} row of matrix D. Assume the number of non-zero element is $d(t)$, then we have:

$$find = D_{t,1:d(t)}$$

Also, we use a vector $bind$ to store all non-zero elements on $(t+1)^{th}$ row of matrix E. Assume the number of non-zero element is $e(t+1)$, then we have:

$$bind = E_{t+1,e(t+1):1}$$

Note that the order to store the $(t+1)^{th}$ row of E is reversed.

Then, (2.16) can be rewritten as the following. For $1 \leq i \leq d(t)$ and $1 \leq j \leq e(t+1)$,

$$log(w^*_{find[i],bind[j],t}) = log(p) + log(p_{find[i],t}), \quad \text{if } find[i] \leq t = bind[j]$$

$$\begin{aligned} log(w^*_{find[i],bind[j],t}) =& log(1-p) + log(p_{find[i],t}) + log(q_{bind[j],t+1}) + log(\pi_{find[i],t}) \\ &+ log(\pi_{t+1,bind[j]}) - log(\pi_{find[i],bind[j]}) - log(\pi_{00}), \\ &\text{if } find[i] \leq t < bind[j]. \end{aligned}$$

The matrices used to store $log(w^*_{ijt})$ and w_{ijt} are $e(t+1) \times d(t)$ matrices. Since both $d(t)$ and $e(t+1)$ have been bounded by the number KK, the total number of weights is bounded by KK^2 which won't increase with the sample size n.

3.4 Measurement and Results

Two measurements are used: Kullback–Leibler (KL) divergence and the L_2 norm of a matrix which is also known as Frobenius Norm (FN). Kullback–Leibler measures the divergence between two probability distributions F and F_0. Specifically, the Kullback–Leibler divergence of F from F_0, denoted as $D_{KL}(F_0||F)$ which is a measure of the information lost when F is used to approximate the true distribution F_0. The Kullback–Leibler divergence between two multivariate normal distribution of the dimension m with the means μ_0, μ_1 and their corresponding nonsingular covariance matrices Σ_0, Σ_1 is:

$$D_{KL}(\mathcal{N}_0||\mathcal{N}_1) = \frac{1}{2}(tr(\Sigma_1^{-1}\Sigma_0) + (\mu_1 - \mu_0)^T\Sigma_1^{-1}(\mu_1 - \mu_0) - m - ln(\frac{det(\Sigma_0)}{det(\Sigma_1)})).$$

Since in our case, we assume that the mean is known as the zero vector, the Kullback–Leibler divergence can be simplified as:

$$D_{KL}(\mathcal{N}_0||\mathcal{N}_1) = D_{KL}(\Sigma_0||\Sigma_1) = \frac{1}{2}(tr(\Sigma_1^{-1}\Sigma_0) - m - ln(\frac{det(\Sigma_0)}{det(\Sigma_1)})).$$

Another measurement is the L_2 Norm, which is also known as the Frobenius Norm, of a $m \times n$ matrix A is:

$$||A||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} = \sqrt{trace(A^T A)}.$$

For our case, let $A = \hat{\Sigma} - \Sigma_0$ in order to measure the divergence from estimated covariance from true covariance.

In this simulation study, we have both 5 and 8 dimensional vectors. For each vector, we have three scenarios (sample size is 1000):

Scenario 1. The data is generated from multi-normal distribution with mean zero and covariance Σ_t. One change-point exists for Σ_t at $t = 501$. That is $\Sigma_t = \Sigma_1$ for $1 \leq t \leq 500$, and $\Sigma_t = \Sigma_2$ for $501 \leq t \leq 1000$, where $\Sigma_1 \neq \Sigma_2$.

Scenario 2. The data is generated from multi-normal distribution with mean zero and covariance Σ_t. Two change-points exist for Σ_t at $t = 301$ and $t = 701$. That is $\Sigma_t = \Sigma_1$ for $1 \leq t \leq 300$, $\Sigma_t = \Sigma_2$ for $301 \leq t \leq 700$, and $\Sigma_t = \Sigma_3$ for $701 \leq t \leq 1000$, where $\Sigma_1 \neq \Sigma_2 \neq \Sigma_3$.

Scenario 3. The data is generated from multi-normal distribution with mean zero and covariance Σ_t. Three change-points exist for Σ_t at $t = 251$, $t = 501$, and $t = 751$. That is $\Sigma_t = \Sigma_1$ for $1 \leq t \leq 250$, $\Sigma_t = \Sigma_2$ for $251 \leq t \leq 500$, $\Sigma_t = \Sigma_3$ for $501 \leq t \leq 750$, and $\Sigma_t = \Sigma_4$ for $751 \leq t \leq 1000$, where $\Sigma_1 \neq \Sigma_2 \neq \Sigma_3 \neq \Sigma_4$.

For each kind of variable and each scenario, we simulated 500 cases within which the sample mean can be calculated for both KL divergence and L_2 Norm.

The result is shown in the table 3.1. The values shown in parentheses are the corresponding stand error. From both measurements, the divergence of our estimation from the true covariance is small, especially for the estimation in multi-dimensional space. Since different from the estimation in one dimensional space, multi-dimensional space

TABLE 3.1: Results of Simulation (Change-Point Fixed)

Number of Change-points		1	2	3
5 Dimension	KL	0.111248	0.022754	0.051042
		(0.015977)	(0.006682)	(0.012530)
	FN	0.080215	0.176822	0.179665
		(0.030539)	(0.040290)	(0.039602)
8 Dimension	KL	0.593686	0.178394	0.387253
		(0.031036)	(0.042207)	(0.027310)
	FN	0.068498	0.267058	0.197097
		(0.033605)	(0.055403)	(0.039484)

contains much more noise which will affect the accuracy of estimation. Despite of the accuracy of estimation, what we care more about is to capture the change-points of the covariance matrix. As we mentioned before, we expect the change of covariance matrix will be reflected by eigenvalues.

Figures 3.3 $\sim$ 3.5 are the plots of eigenvalues in 5 dimensional space for 1 change-point, 2 change-points and 3 change-points respectively. Figures 3.6 $\sim$ 3.8 are the plots of eigenvalues in 8 dimensional space for 1 change-point, 2 change-points and 3 change-points respectively. In each plot, the red line is the eigenvalues of true covariance matrix and the blue line stands for the eigenvalues of estimated covariance matrix. From these figures, we can see that the estimated eigenvalues vary from the true eigenvalues in an acceptable range. This is normal in any statistical estimation which is known as the estimation error. However, we don't care much about how precisely an estimation is at a single time, we expect the the dynamical behavior of the estimation will mimic that of true covariance so that the change-points are captured. In fact, our model has completed the task almost perfectly in this experiment. The blue lines and red lines jump at the same time. In other words, the estimated eigenvalues change at the same time true eigenvalues change. The only miss happens in the second change-point of eigenvalue 8 in figure 3.8. The reason might be both the true eigenvalue and the amount of change are too small. We also note that at some change-points, the estimated eigenvalue is unstable. For example, at the

third change-point of eigenvalue 3 in figure 3.4, the estimated eigenvalue jumps up and jumps back down to the stable value.

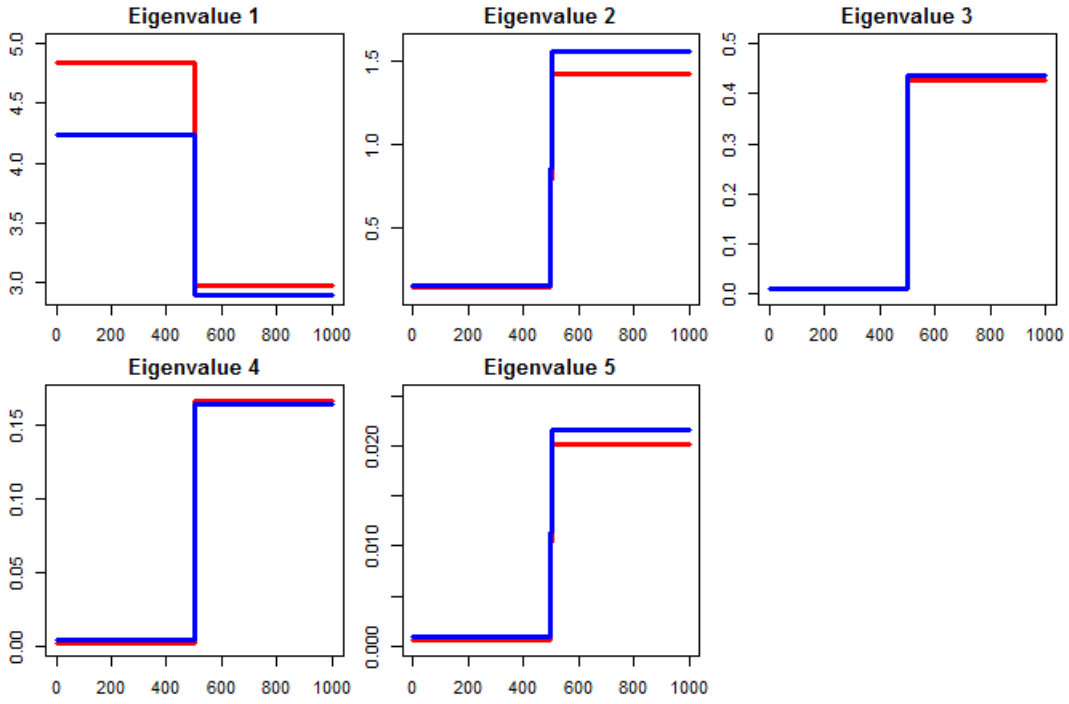

FIGURE 3.3: Eigenvalues for 1 Fixed Change-Point Simulated in 5 Dimensional Space

In the previous simulation experiment, the number and location of change-point are fixed. Next, we simulate change-points randomly with different probabilities. The observations are still from the 5 dimensional and 8 dimensional space. As what we did before, we simulate 500 cases with sample size 1000 for each case. Let the indicate variable $\mathbf{I}_t$ distributed as Bernoulli, indicating if a change occurs at time t:

$$\mathbf{I}_t := \begin{cases} 1, & p, \\ 0, & 1-p. \end{cases}$$

The probability p is set as 0.001, 0.005, and 0.01 respectively. The results can be concluded as in the table 3.2:

Since the change-points are randomly simulated, table 3.3 gives an exact description of the numbers and locations of change-points for each case. "Change-Point" is shorted

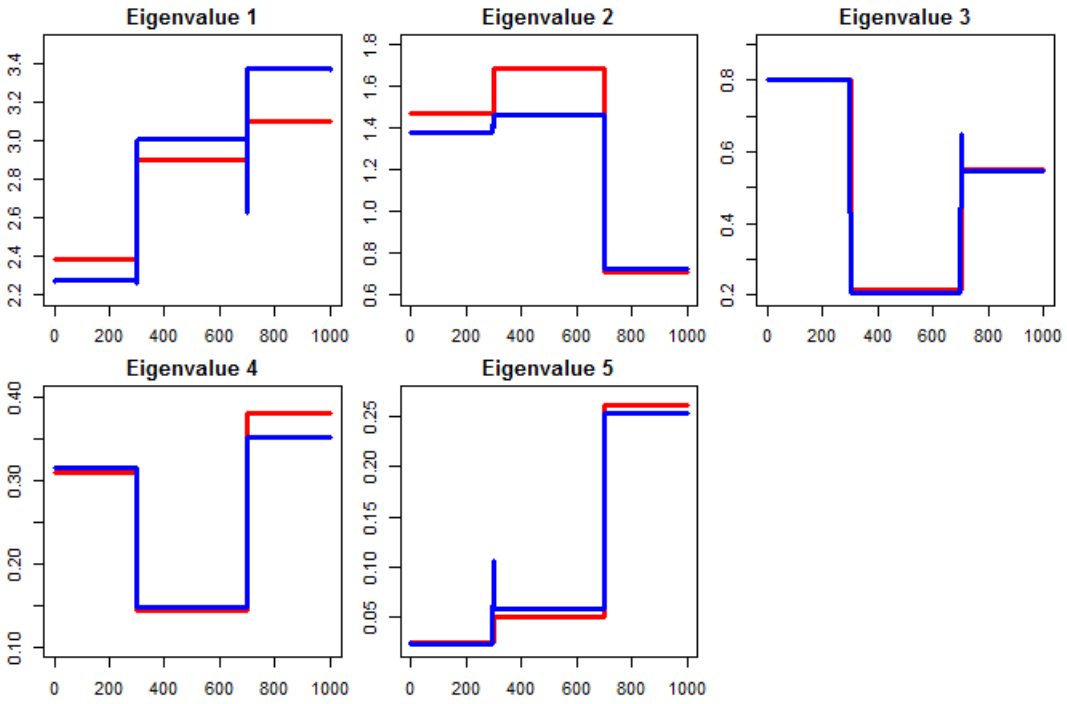

FIGURE 3.4: Eigenvalues for 2 Fixed Change-Points Simulated in 5 Dimensional Space

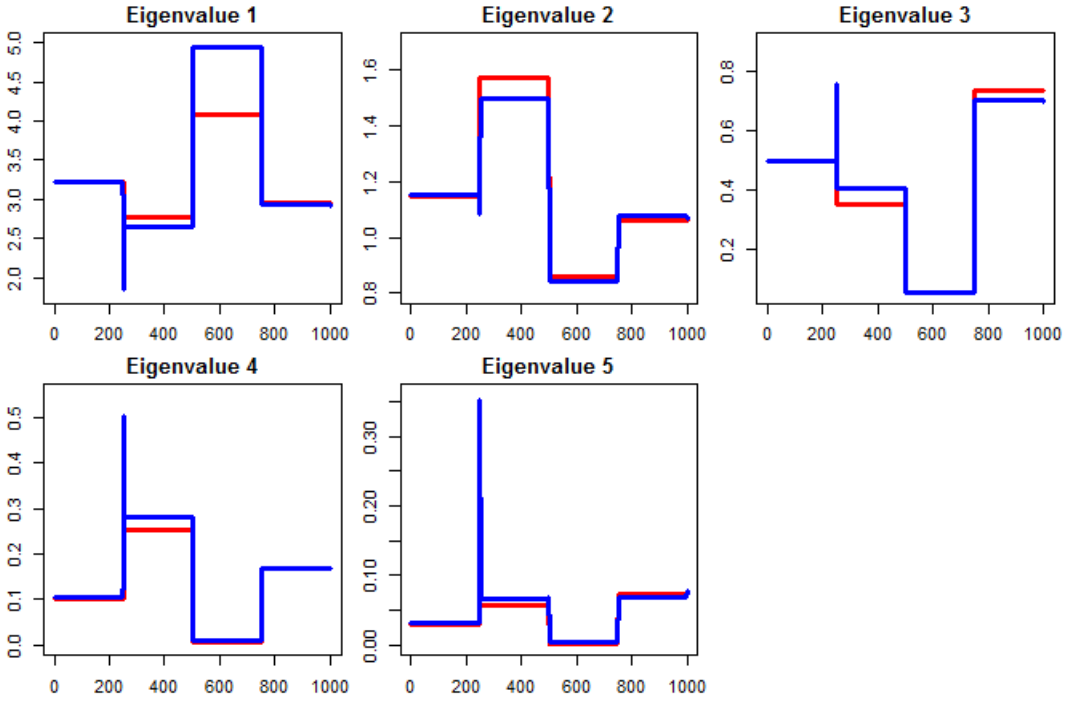

FIGURE 3.5: Eigenvalues for 3 Fixed Change-Points Simulated in 5 Dimensional Space

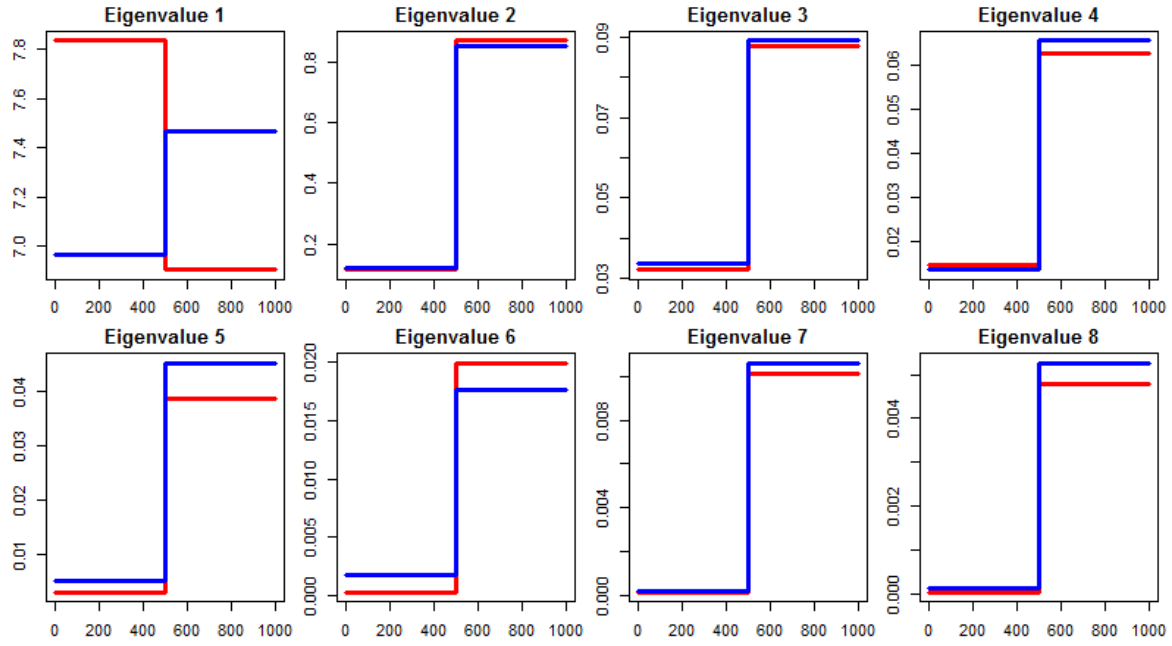

FIGURE 3.6: Eigenvalues for 1 Fixed Change-Point Simulated in 8 Dimensional Space

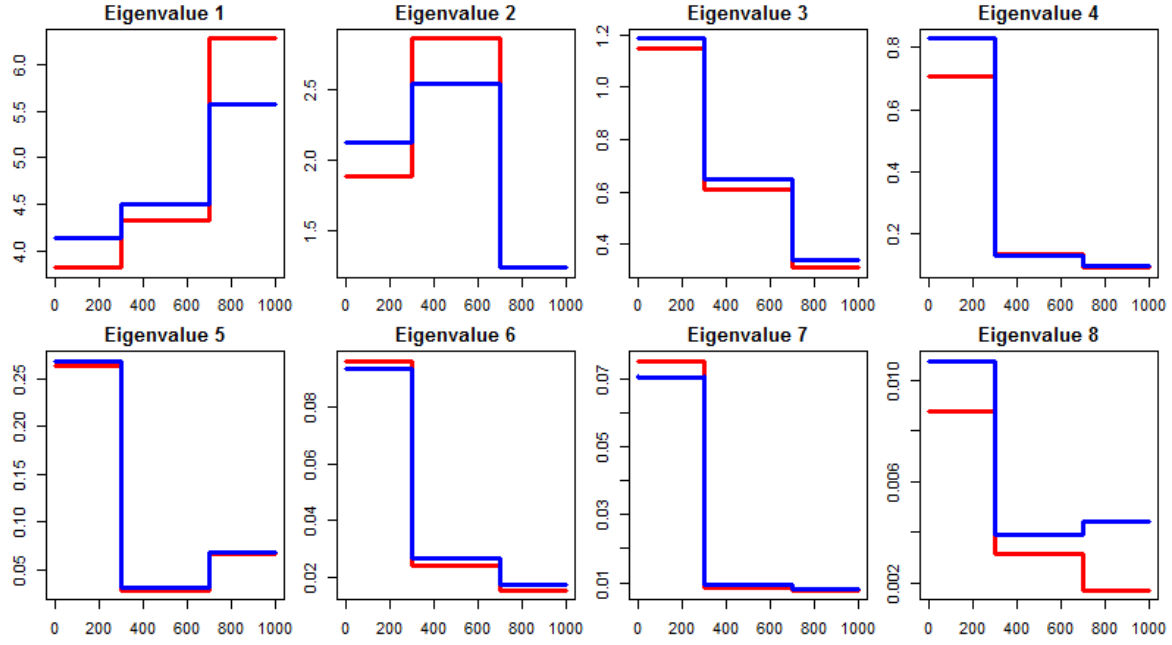

FIGURE 3.7: Eigenvalues for 2 Fixed Change-Points Simulated in 8 Dimensional Space

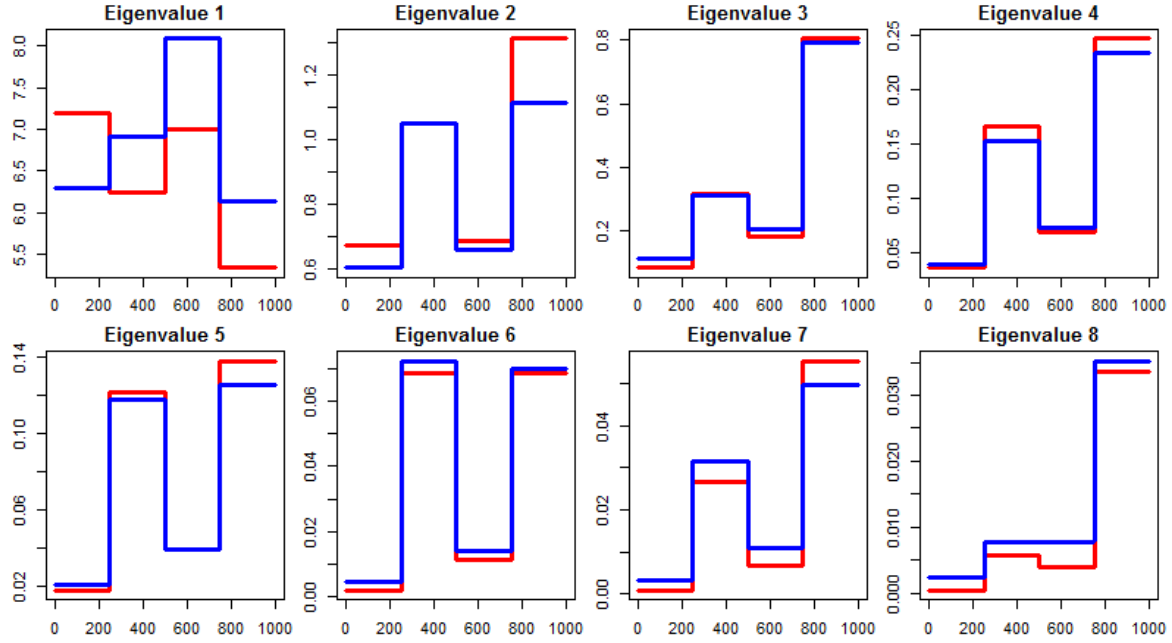

FIGURE 3.8: Eigenvalues for 3 Fixed Change-Points Simulated in 8 Dimensional Space

TABLE 3.2: Results of Simulation (Change-Point Randomly Simulated)

Probability of Change Occurs		0.001	0.005	0.01
5 Dimension	KL	1.054809	0.403882	0.496377
		(0.018151)	(0.013435)	(0.036012)
	FN	1.463088	1.144405	0.990666
		(0.020171)	(0.015470)	(0.020319)
8 Dimension	KL	0.499691	0.600790	0.732257
		(0.016173)	(0.026516)	(0.028544)
	FN	1.155180	1.295160	1.433151
		(0.021665)	(0.030906)	(0.029558)

as "CP" in the table.

Next, we provide the plots of eigenvalues which illustrate how our estimated eigenvalues perform comparing with the true eigenvalues. Figures 3.9 $\sim$ 3.11 are cases in 5 dimensional space with simulated probability 0.001, 0.005, and 0.01 respectively. Figures 3.12 $\sim$ 3.14 are the cases in 8 dimensional space with simulated probability 0.001, 0.005, and 0.01 respectively. The exact locations (indexes) can be find from table 3.3.

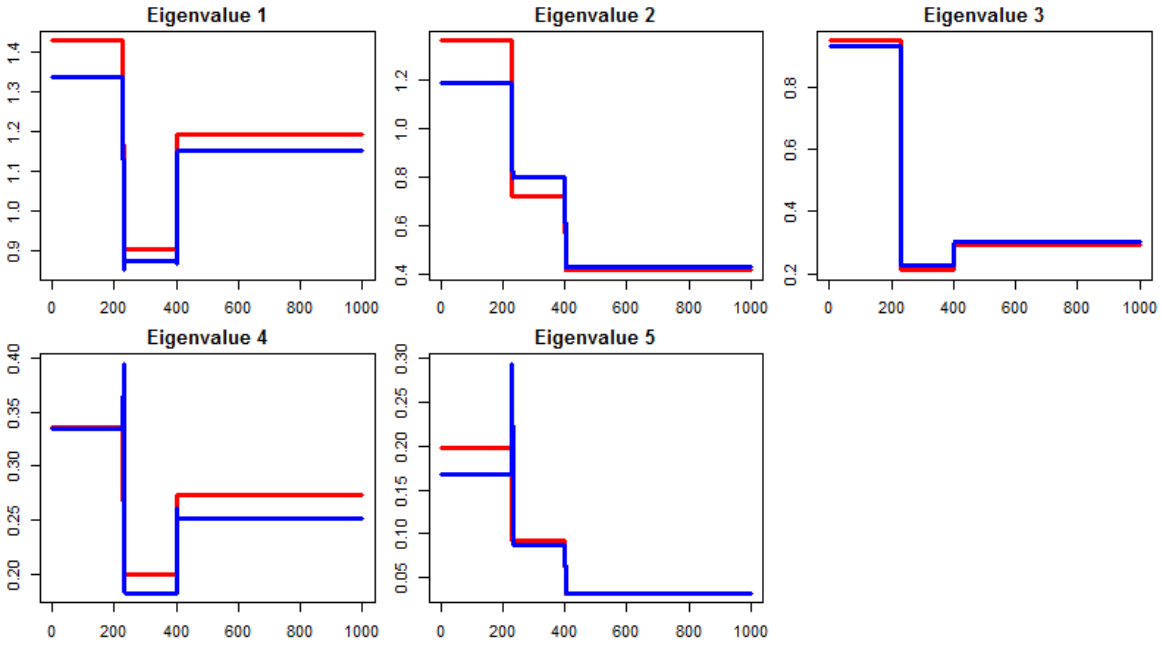

FIGURE 3.9: Eigenvalues for Random Change-Points Simulated with Probability 0.001 in 5 Dimensional Space

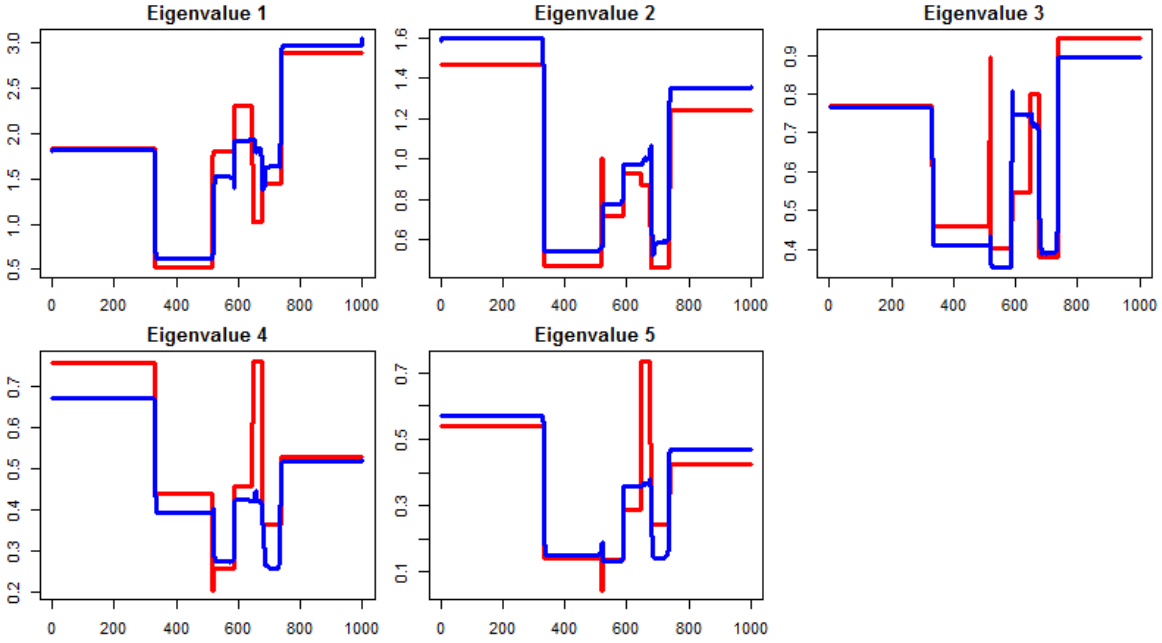

FIGURE 3.10: Eigenvalues for Random Change-Points Simulated with Probability 0.005 in 5 Dimensional Space

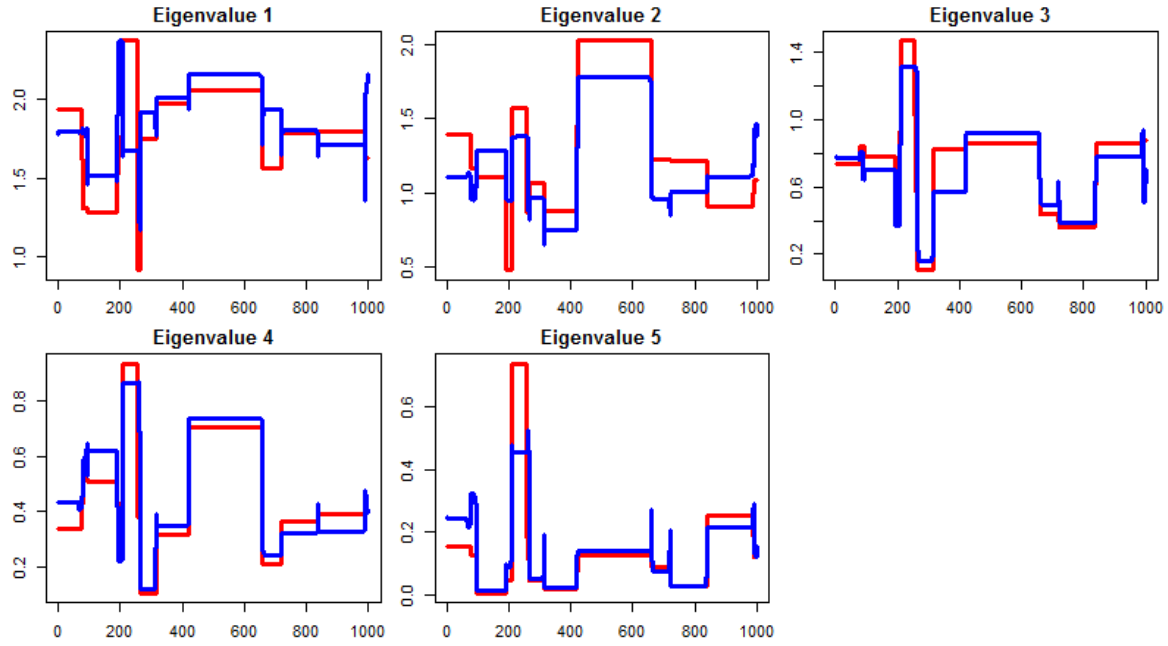

FIGURE 3.11: Eigenvalues for Random Change-Points Simulated with Probability 0.01 in 5 Dimensional Space

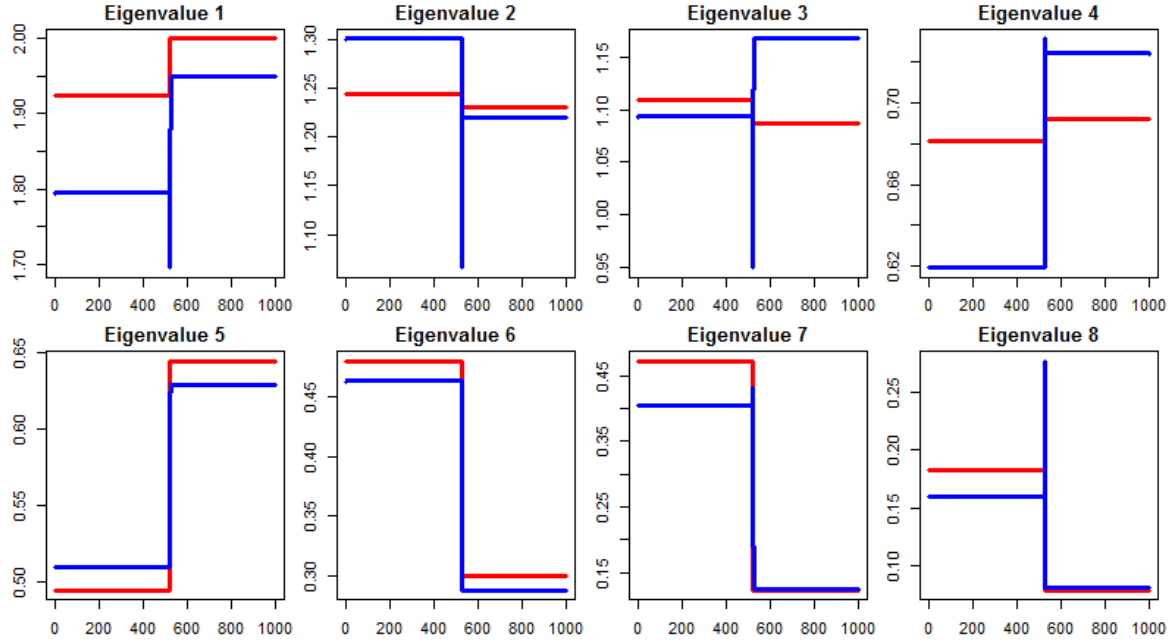

FIGURE 3.12: Eigenvalues for Random Change-Points Simulated with Probability 0.001 in 8 Dimensional Space

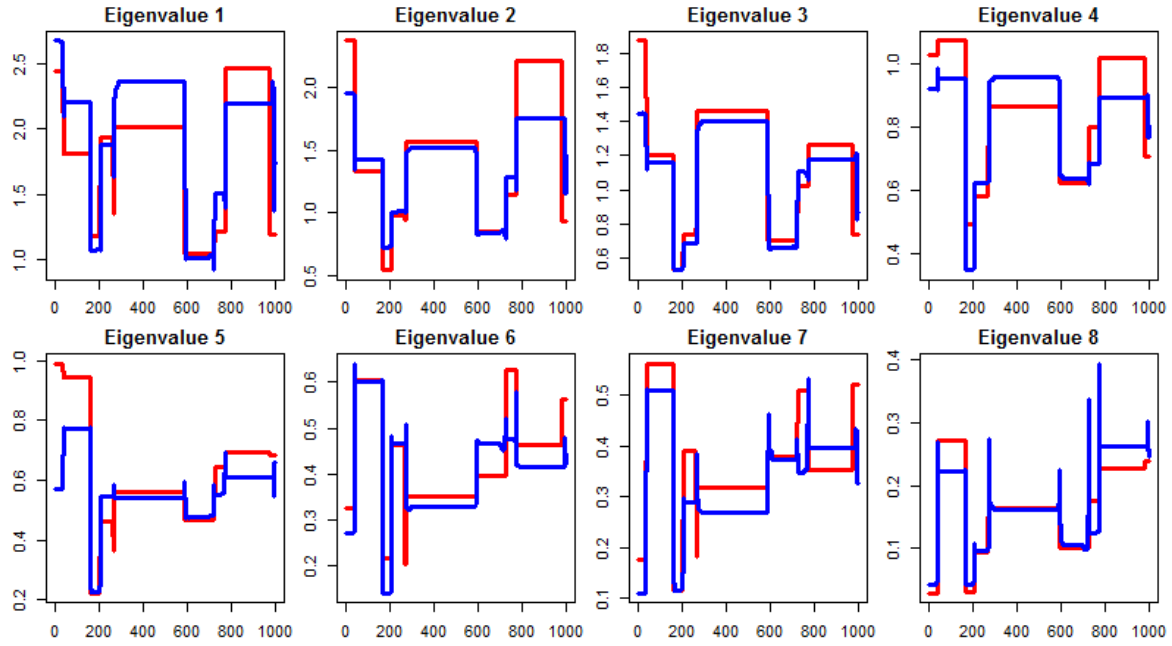

FIGURE 3.13: Eigenvalues for Random Change-Points Simulated with Probability 0.005 in 8 Dimensional Space

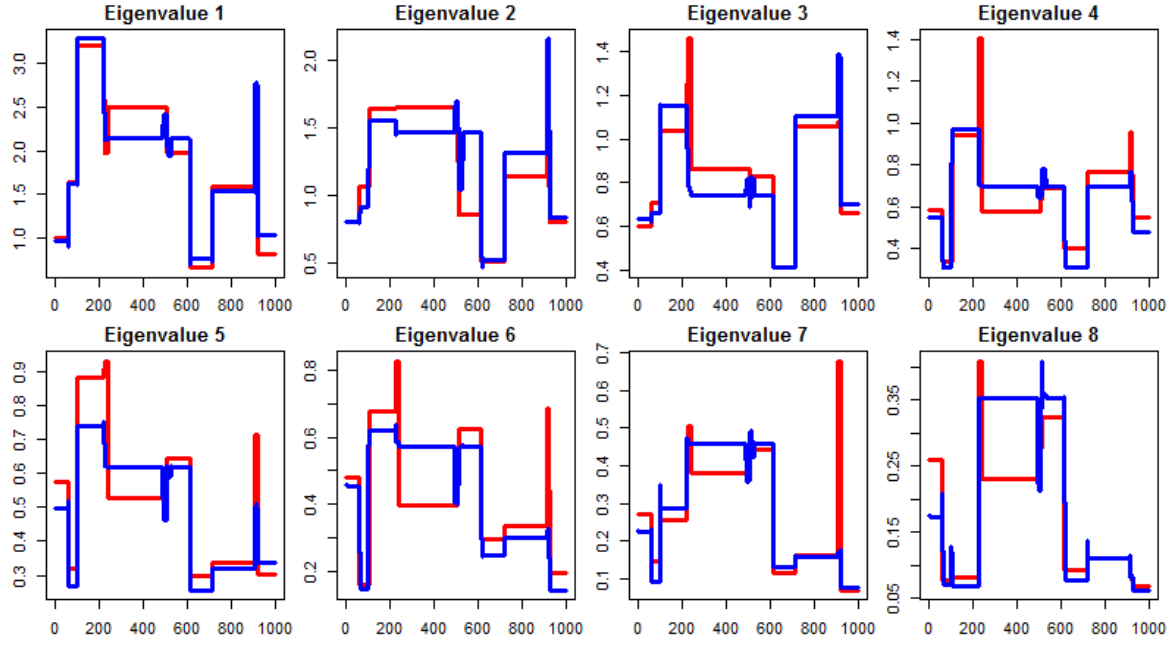

FIGURE 3.14: Eigenvalues for Random Change-Points Simulated with Probability 0.01 in 8 Dimensional Space

TABLE 3.3: Number and Location of Change-Point (Change-Point Randomly Simulated)

Probability of Change Occurs		0.001	0.005	0.01
5 Dimension	Number of CP	2	7	12
	Index of Location	(230,402)	(332,517,521, 588,646,676, 738)	(78,95,192,209, 257,264,316,421, 659,720,839,989)
8 Dimension	Number of CP	1	9	9
	Index of Location	(524)	(39,164,206, 266,271,592, 726,774,978)	(61,104,226, 242,510,615, 719,912,923)

By comparing the value and behavior of blue lines and red lines in the plots, we could make some comments here. For a very small probability of change-point simulated (in this case, $p = 0.001$), our change-point model works very well on both estimating eigenvalues and capturing change-points. This is shown in figure 3.9 and figure 3.12. With a small probability, the expected number of change-point is small. As a result, it's more likely to have more observations before and after a certain change-point, which could provide more information to detect this change-point. As the probability p increases, number of change-points gets bigger. In this situation, it's more likely that several change-points will appear within a short time interval, which will makes detection difficult. See the 5 dimensional case with $p = 0.005$. From figure 3.10 we could find several change-points are very closed to each other and located between time 500 and 800. In fact, from table 3.3, the exact locations of them are (517, 521, 588, 646, 676, and 738). The estimation of this part no longer appears stable and clear. Generally speaking, our model still performs well in this experiment, since most change-points have been captured. However, it's slightly worse than the result from the "fixed" experiment, which might be caused by the randomness of change-points.

Chapter 4

Case Studies

Convinced by the results from the previous simulation studies, we apply our model to some real data sets to see how it performs.

4.1 Swap Rate

Firstly, we apply the model to the swap rate data which has been shown in chapter 1. In chapter 1, we applied PCA to a sequence of subsets of the sample. The plotted eigenvalues and eigenvector have implied the existence of change-point in covariance matrix. For the data we used, each observation is an eight dimensional vector stands for the daily change of swap rate. Each column represents different maturities: maturity of 1 year, 2 years, 3 years, 4 years, 5 years, 7 years, 10 years, and 30 years. The sample size is 1256. The original data can be found here: http://www.ams.sunysb.edu/~xing/statfinbook/_BookData/Chap02/d_swap.txt. The figure 1.3 is the snapshot of the original swap rate data.

The explicit formulas have been derived for observations from Normal distribution with known mean zero. In order to match this assumption, we have to difference the data. The data at time t is obtained by subtracting the original data at time t by time

$t+1$. The figure 1.4 is a snapshot of the differenced swap rate data to which we are going to apply the Bayesian model. Figure 4.1 is the plot of change of swap rate for different maturities. In another word, the plots of each column of the data set shown in figure 1.4.

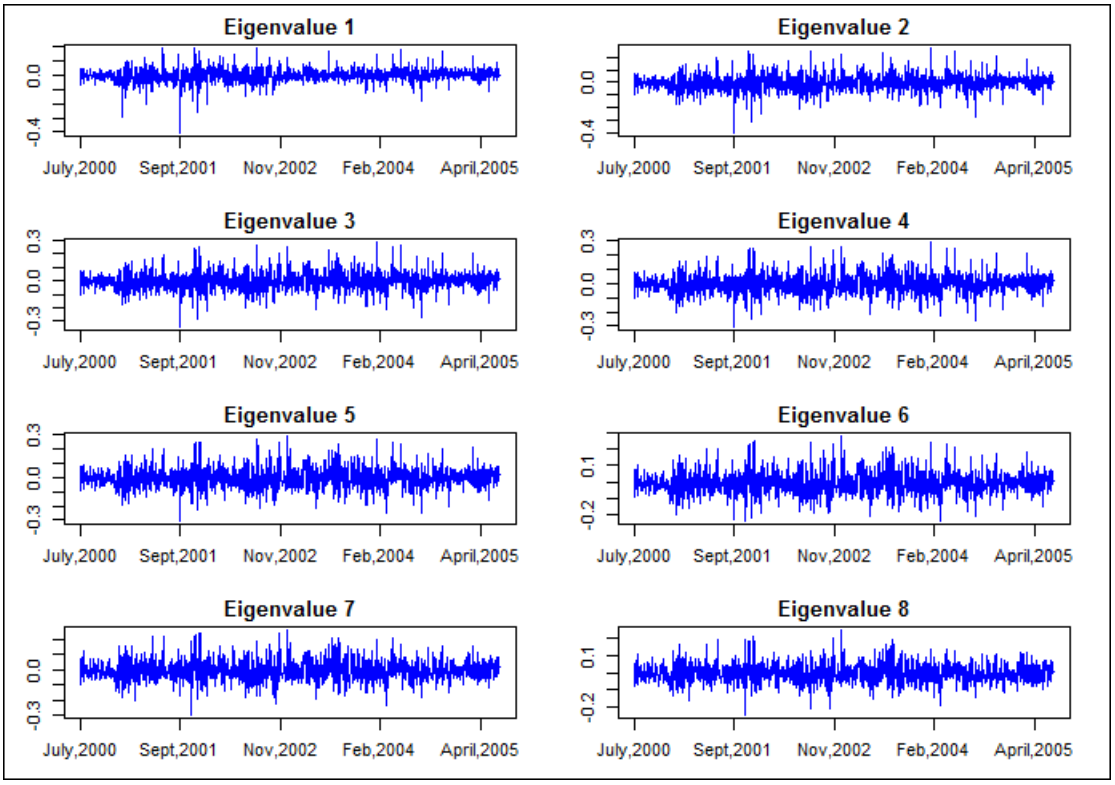

FIGURE 4.1: Change of Swap Rate of Each Maturity

After applying the Bayesian model to the data, we have a sequence of estimated covariance matrices. As mentioned before, in order to make the change visible, we do eigen-decomposition to each estimated covariance matrix. Figure 4.2 shows the first four eigenvalues of the estimated covariance matrices. Figure 4.3 shows the last four eigenvalues of the estimated covariance matrices.

From each figure, we can easily observe some significant "jumps". For example,

from the plot of the first eigenvalue, five significant change-points occur around: July/2000, Dec/2000, Sept/2001, June/2002 and Feb/2004. In fact, the change of eigenvector matches the change of corresponding eigenvalue. Figure 4.4 is the plot of the first eigenvector of estimated covariance matrices. The jumps (look like "ridges") in this figure agree with those changes in the 1st eigenvalue.

From the rest of the eigenvalues, we can find that the most significant five "jumps" all occur at the same time as those of the first eigenvalue. It implies that the whole eigen-space has changed at these five time points, which is a strong evidence for the structure breaks of the covariance matrix. Note that, at some change-points, such as the time around Feb/2004, the first eigenvalue and the second eigenvalue encounter a very small amount of change (increasing and decreasing a little bit respectively). Eigenvalue 3, eigenvalue 4 and eigenvalue 5 decrease at this time point while eigenvalue 6, eigenvalue 7 and eigenvalue 8 increase. The increase and decrease of all the eigenvalues may counteract with each other resulting in an unchanged determinant of the matrix, because the determinant can be calculated by the product of all its eigenvalues. Therefore, it confirms us that it's reasonable to use the eigen-structure rather than the determinant to measure the change.

4.2 Treasury Constant Maturity Rate

The second data used is the Treasury Constant Maturity Rate (monthly and weekly respectively) from the Federal Reserve Bank of St.Louis. In our sample, each observation is an 11-dimensional vector whose columns stand for different maturities: 1 months, 3 months, 6 months, 1 year, 2 years, 3 years, 5 years, 7 years, 10 years, 20 years, and 30 years respectively.

The observations are taken starting from 2001-07-01 to 2012-11-01 (with sample size 136) for monthly data and from 2001-08-10 to 2012-12-28 (with sample size 594) for

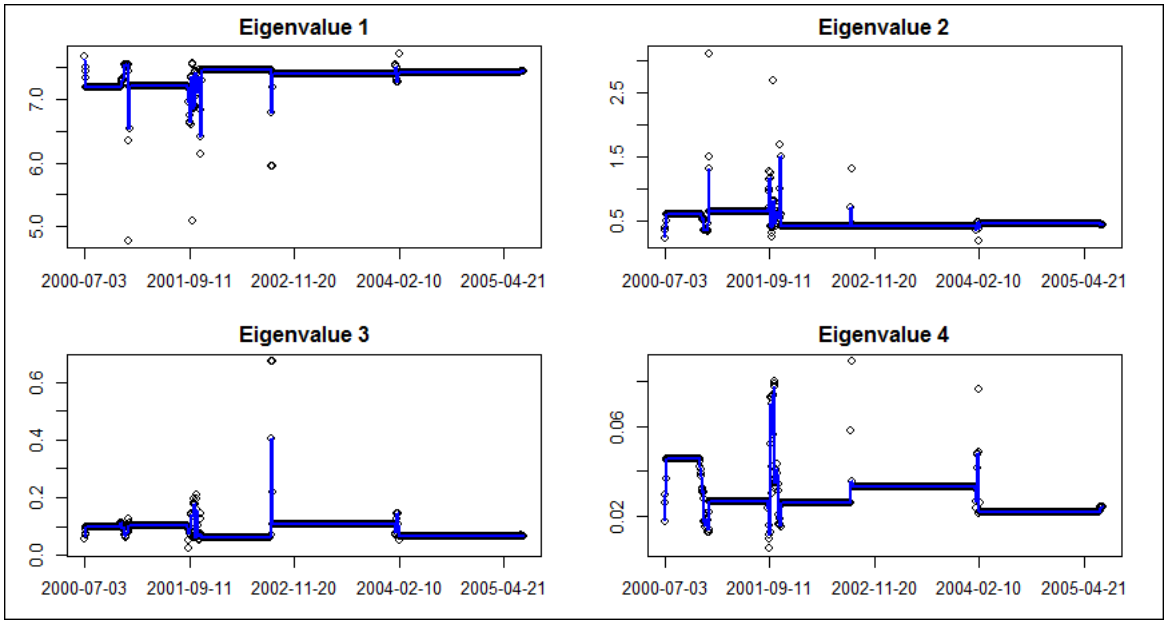

FIGURE 4.2: Swap rate: The First Four Eigenvalues of Estimated Covariance Matrices

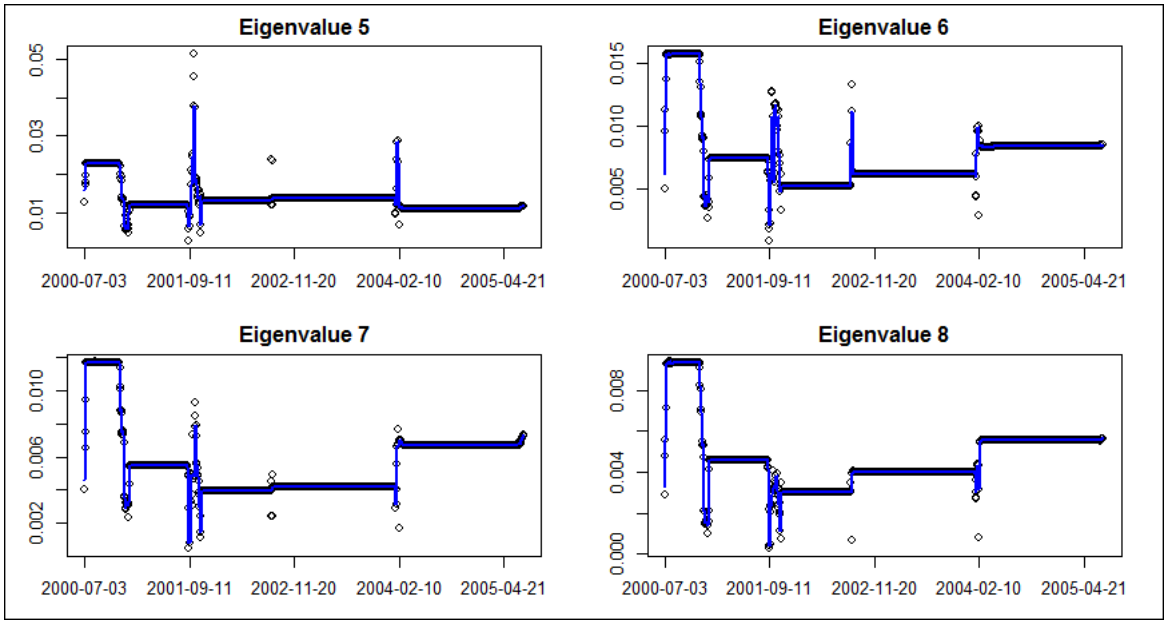

FIGURE 4.3: Swap Rate: The Last Four Eigenvalues of Estimated Covariance Matrices

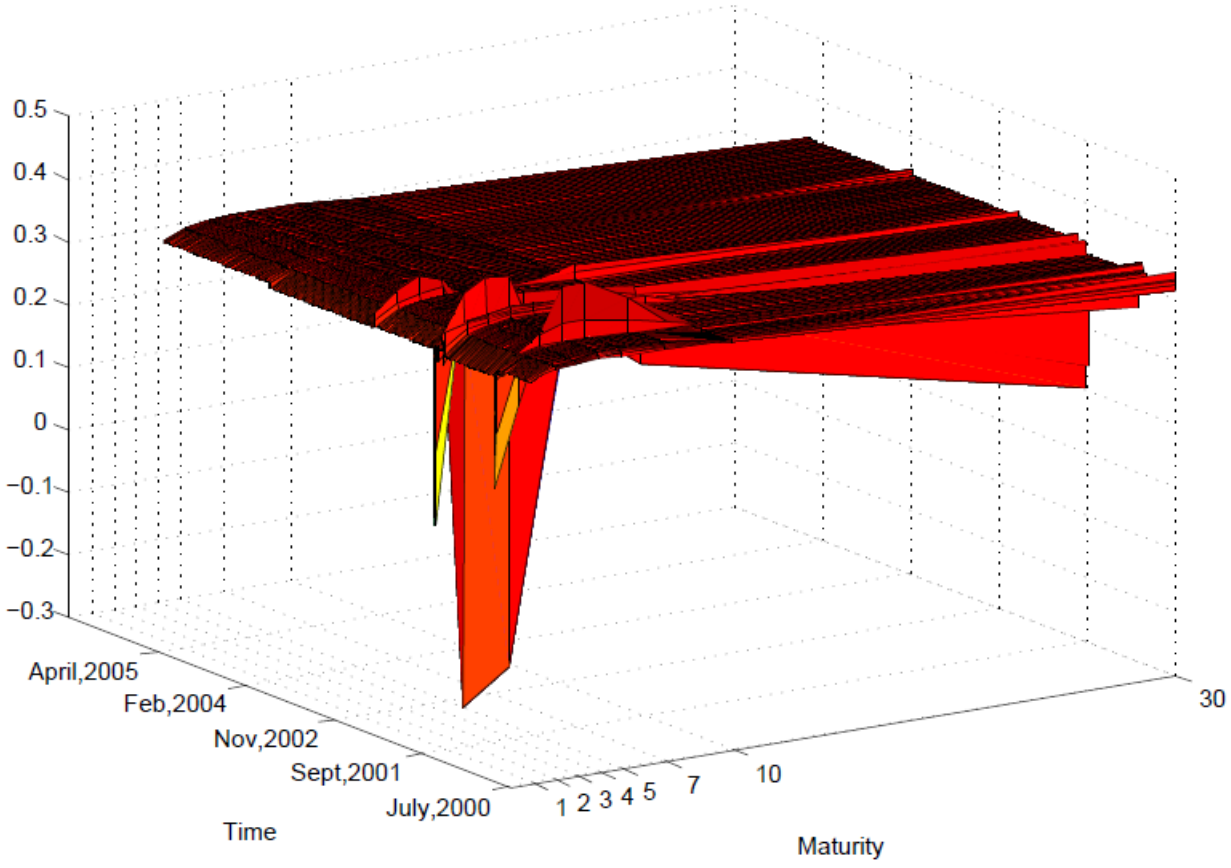

FIGURE 4.4: Swap Rate: 1st Eigenvector of Estimated Covariance Matrices

weekly data. The differenced data (the change of the treasury constant maturity rate) is applied to our model as it will satisfy the assumption of zero mean. Figures 4.5 and 4.6 are the snapshots for monthly and weekly data respectively.

For monthly data, figures 4.7 $\sim$ 4.9 are the plots of all 11 eigenvalues of the estimated covariance matrices. Figure 4.10 is the plot of first eigenvector of the estimated covariance matrices. The most significant change occurs around Dec/2008. All the behaviors of 11 figures agree at this time. Other potential change-points may be implied by a certain eigenvalues. For example, around June/2006, the change is significantly reflected by eigenvalue 3, 4, 6, 7, 8, 9, 10 and 11 but not from the eigenvalue 1, 2 and 5. Combining

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]
[1,]	0.06	0.09	0.11	0.13	0.16	0.15	0.18	0.23	0.27	0.31	0.33
[2,]	0.09	0.12	0.11	0.10	0.08	0.08	0.10	0.09	0.12	0.13	0.14
[3,]	0.01	0.08	0.16	0.16	0.15	0.12	0.07	0.05	0.00	-0.06	-0.05
[4,]	0.19	0.16	0.10	0.06	0.00	-0.02	-0.03	-0.03	-0.02	-0.04	-0.02
[5,]	0.26	0.01	-0.10	-0.14	-0.22	-0.22	-0.22	-0.22	-0.21	-0.17	-0.13
[6,]	-0.39	-0.16	-0.09	-0.11	-0.13	-0.16	-0.15	-0.15	-0.16	-0.15	-0.15
[7,]	0.20	0.12	0.04	0.04	0.03	0.03	0.02	0.01	0.01	0.01	0.00
[8,]	0.24	0.02	0.03	0.00	-0.06	-0.08	-0.11	-0.11	-0.13	-0.16	-0.16
[9,]	-0.34	-0.10	-0.08	-0.07	-0.07	-0.06	-0.05	-0.04	-0.04	0.00	-0.01
[10,]	0.07	0.14	0.08	0.12	0.21	0.21	0.22	0.21	0.20	0.17	0.17

FIGURE 4.5: Treasury Constant Maturity Rate: Monthly Data

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]
[1,]	0.05	0.03	0.02	0.02	0.02	0.02	0.02	0.00	-0.03	-0.06	-0.05
[2,]	0.02	0.04	0.03	0.02	0.01	0.02	0.05	0.05	0.05	0.06	0.07
[3,]	-0.01	-0.02	0.02	0.03	0.02	0.07	0.10	0.12	0.13	0.15	0.14
[4,]	0.04	0.02	0.03	-0.01	-0.07	-0.08	-0.09	-0.07	-0.03	-0.01	0.01
[5,]	0.15	0.05	0.00	0.01	0.05	-0.01	0.00	0.00	-0.02	0.00	-0.01
[6,]	0.04	-0.04	0.02	0.05	0.08	0.10	0.11	0.11	0.11	0.11	0.11
[7,]	-0.04	0.05	0.03	0.03	0.05	0.05	0.06	0.07	0.09	0.10	0.11
[8,]	-0.03	0.02	0.06	0.06	0.06	0.06	0.07	0.07	0.09	0.11	0.11
[9,]	-0.06	0.03	0.00	-0.01	-0.03	-0.02	0.00	0.02	0.04	0.04	0.05
[10,]	0.07	0.05	0.03	0.04	0.04	0.04	0.04	0.05	0.05	0.04	0.05

FIGURE 4.6: Treasury Constant Maturity Rate: Weekly Data

all these 11 figures together, it's reasonable to roughly divide the time into three pieces: July/2001 to June/2006, June/2006 to Dec/2008 and Dec/2008 to Nov/2012. Eigenvalue 3 is a good illustration. It's not very easy to give a clear segmentation since eigenvalues doesn't keep stable within several small interval. This might be caused by the small sample size (which is 136) related to the dimension 11.

For weekly data, the figures 4.11 $\sim$ 4.13 are the plots of all 11 eigenvalues of the estimated covariance matrices. Figure 4.14 is the plot of first eigenvector of the estimated covariance matrices. In this case, the segmentation is much clearer than that of monthly case, since the sample size is much bigger and we have enough information to detect change-point. A few change-points are significantly shown from the figures, such as the time around Sept/2001, May/2004, Oct/2005 and July/2008.

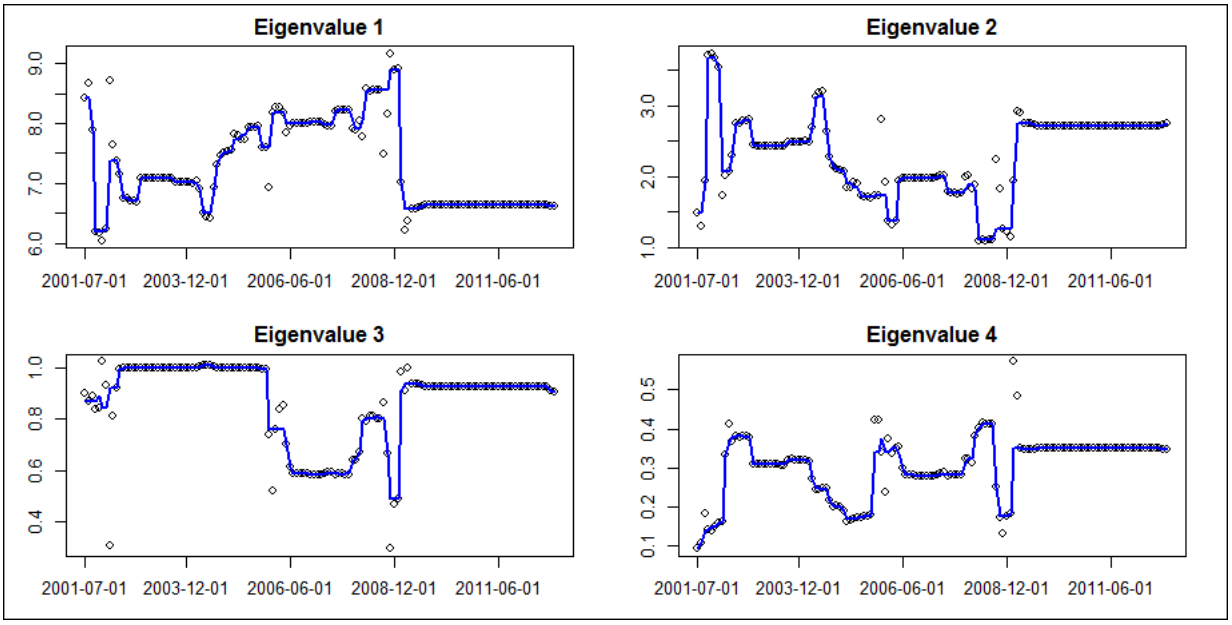

FIGURE 4.7: Treasury Constant Maturity Rate: Eigenvalue 1∼4 of Estimated Covariance Matrices for Monthly Data

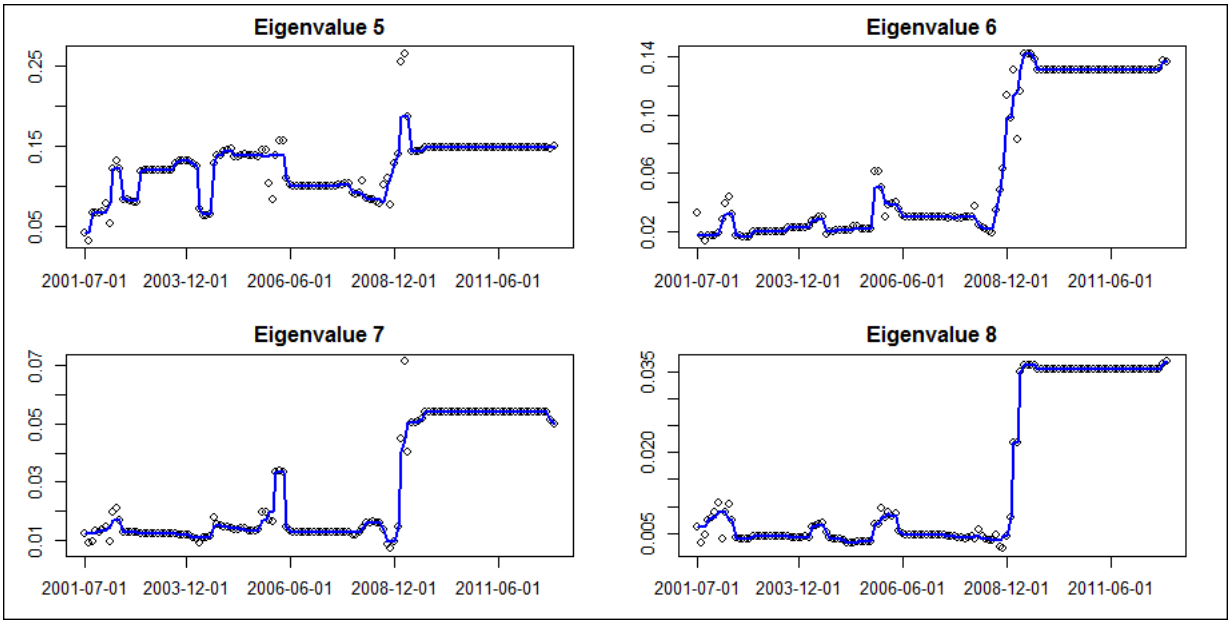

FIGURE 4.8: Treasury Constant Maturity Rate: Eigenvalue 5∼8 of Estimated Covariance Matrices for Monthly Data

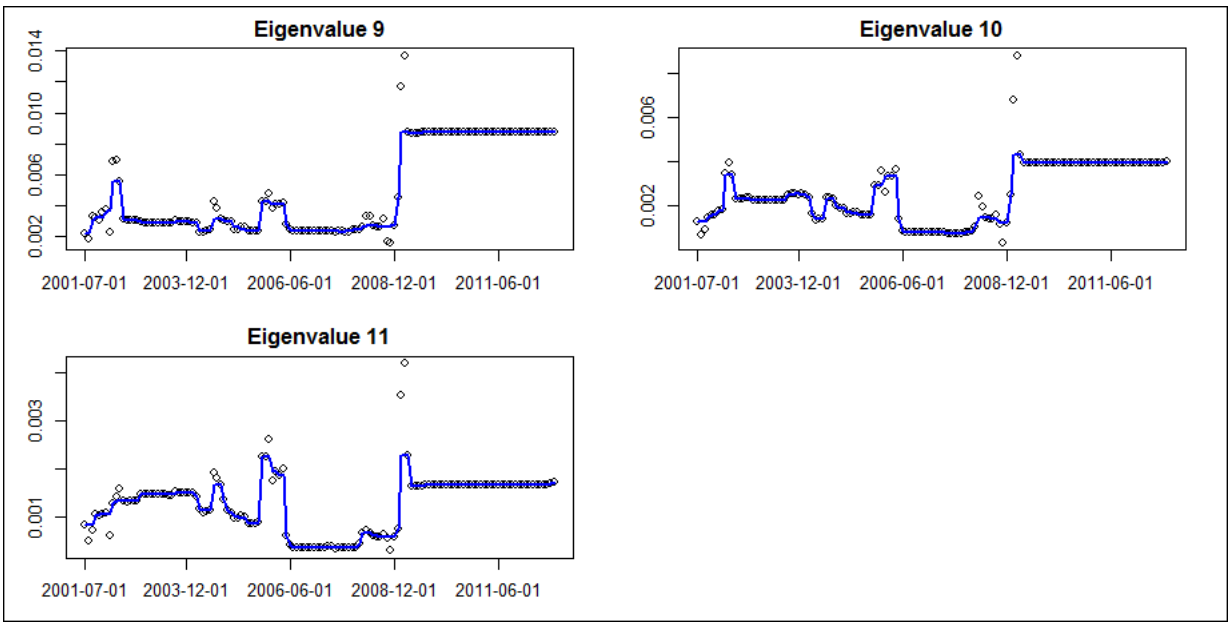

FIGURE 4.9: Treasury Constant Maturity Rate: Eigenvalue 9∼11 of Estimated Covariance Matrices for Monthly Data

In practice, often we are only able to do a rough segmentation according to the plots of eigenvalues, since the parameter of true distribution might change frequently within a short time period. Just like the case in the simulation part with change-point simulated randomly. In this situation, we are unable to get the ideal segmentation like what we achieved in the fixed change-point case. Instead of figuring out a change-point in these intervals, we would like to say, the distribution (parameters) changes frequently within this interval.

Note that, by comparing the monthly data with weekly data, we can find that our model is affected by the sample size, as all statistical inference methods do. However, the situation may get worse if it is the multiple dimensional case. The sample will get extremely sparse when the dimensionality is high. As in the monthly case, the $Dimension/SampleSize$ is around 0.1 which is very big and results in a relatively poor performance. If the case is in a high dimensional space, like 500 dimensional space, it might need a huge sample size to get a satisfactory result. However, the computational

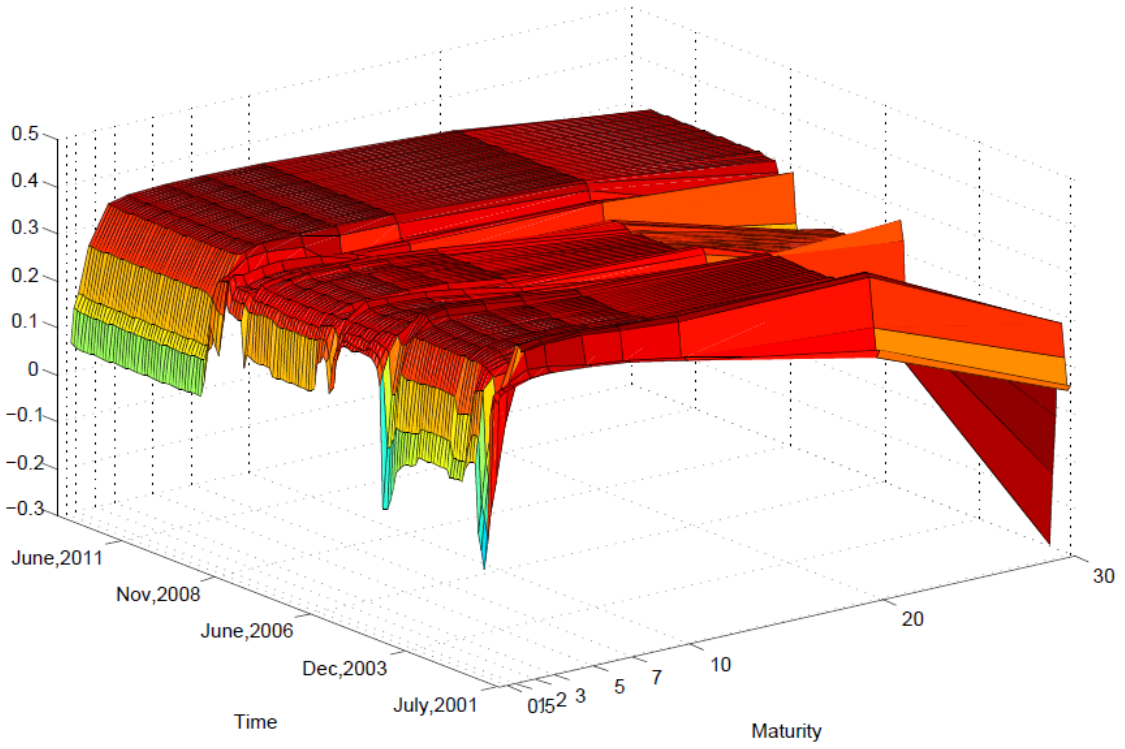

FIGURE 4.10: Treasury Constant Maturity Rate: 1st Eigenvector of Monthly Data

burden usually is unacceptable.

4.3 S&P 500 Stocks

In this section, the data comes from S&P500. S&P500 stands for Standard & Poor's 500 which is a stock market index depending on the market capitalization of 500 big companies who have common stocks on New York Stock Exchange (NYSE) or NASDAQ Stcok Market. S&P500 may be one of the most commonly followed equity indexes across the nation and considered as a very good representation of the U.S. stock market.

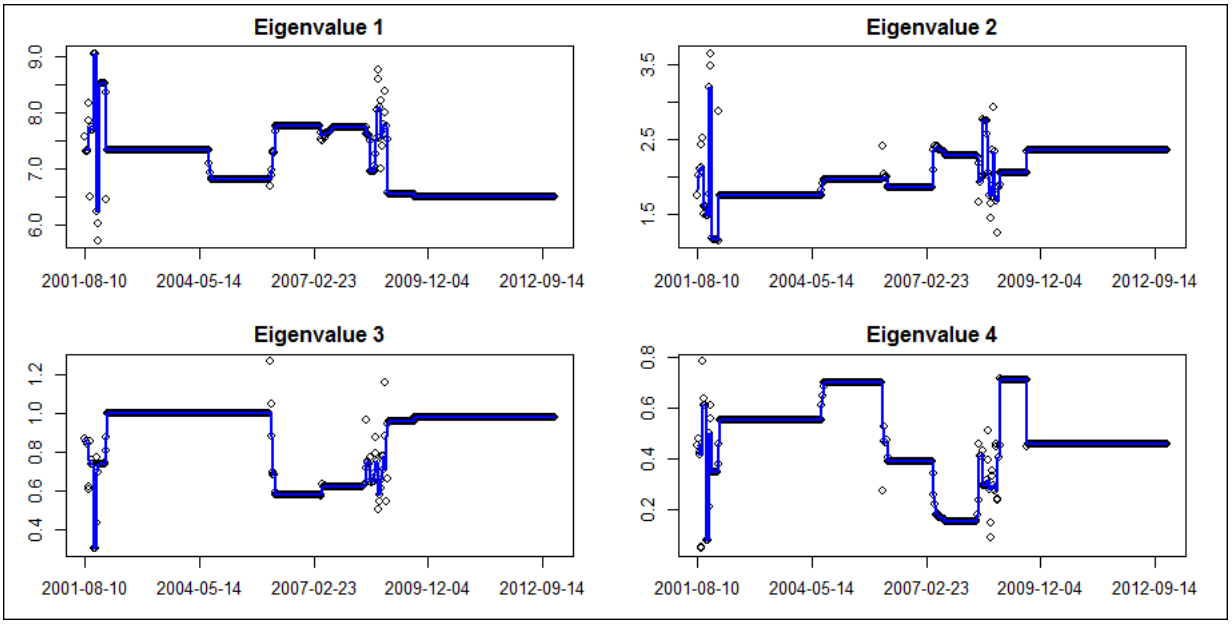

FIGURE 4.11: Treasury Constant Maturity Rate: Eigenvalue 1∼4 of Estimated Covariance Matrices for Weekly Data

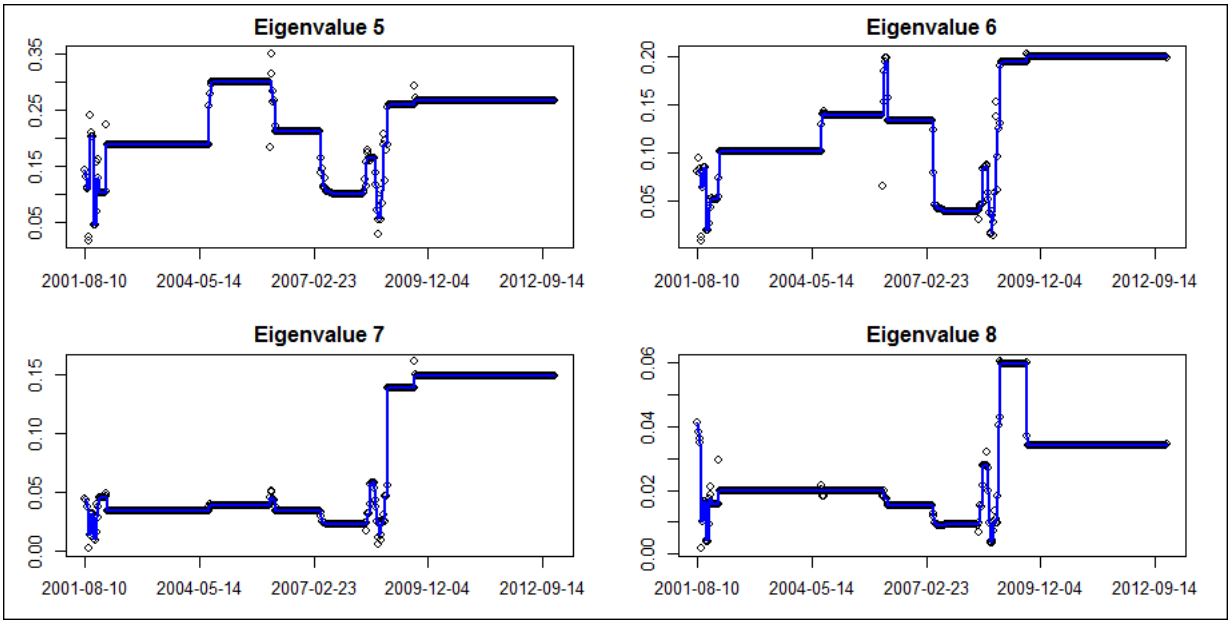

FIGURE 4.12: Treasury Constant Maturity Rate: Eigenvalue 5∼8 of Estimated Covariance Matrices for Weekly Data

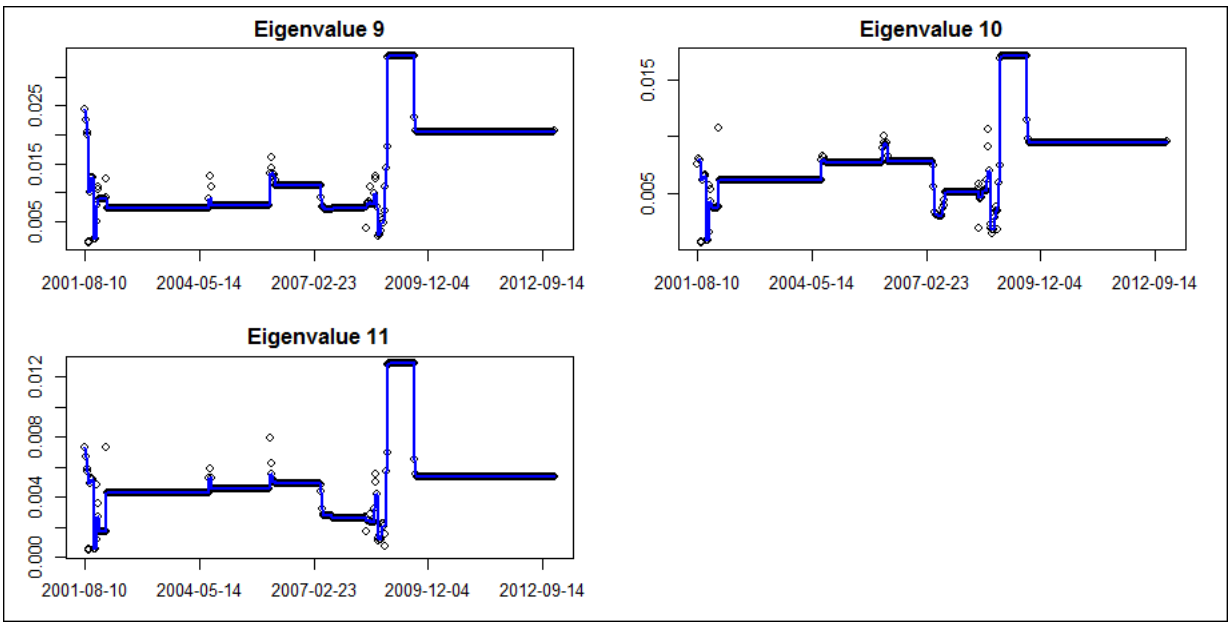

FIGURE 4.13: Treasury Constant Maturity Rate: Eigenvalue 9∼11 of Estimated Covariance Matrices for Weekly Data

The data used comes from 17 different stocks [1], which means the dimension is 17. The data is downloaded from **Yahoo!Finance**. Figure 4.15 is a snapshot of the index of ticker "A" (Agilent Technologies) from **Yahoo!Finance**.

The sample is taken from March 16th 2004 to March 16th 2014, while some dates have been deleted due to index missing. We also difference the data to satisfy the assumption of zero mean. Figure 4.16 is a snapshot (the first 15 observations) of the differenced data which will be applied with the Bayes model. The exact sample size is 2516. Since the dimension is high and the sample size is doubled comparing to the previous cases, we employ BCMIX in order to speed up the calculation. Besides, we trade the speed for saving memory size with the way mentioned in section 3.2.3.

Since our case is in 17 dimensional space, the estimated covariance matrices have 17 eigenvalues. In order to show the behavior of the eigenvalues, we pick the 4 eigenvalues

[1] The name of the 17 tickers:"A", "AA", "AAPL" "ABC" "ABT" "ACE", "ACN", "ACT", "ADBE", "ADI", "ADM", "ADP", "ADSK", "AEE", "AEP", "AES" and "AET".

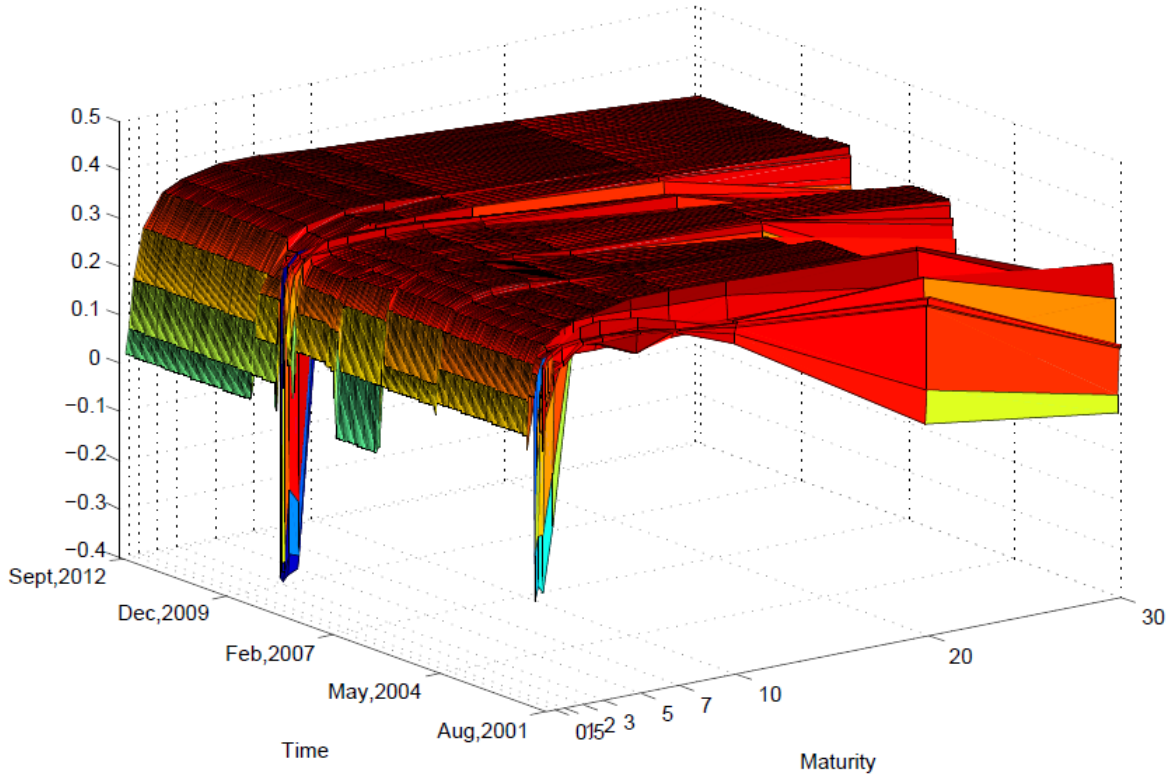

FIGURE 4.14: Treasury Constant Maturity Rate: 1st Eigenvector of Weekly Data

in the first, 4 eigenvalues in the middle and 4 eigenvalues in the last and plot them along the time. See figures 4.17 $\sim$ 4.19. Figure 4.20 illustrates the change of first eigenvector of estimated covariance matrices. Generally speaking, it's easy to make a rough segmentation according to these plots of eigenvalues. The all agrees with each other on the following four major change-points around the time: Mar/2007, May/2009, Mar/2011 and Jan/2013. We can find that eigenvalues "jumps" frequently within the time period Mar/2004 to Mar/2006. As that in previous cases, it might be caused by the nature of the true distribution. However, one thing need to be mentioned is, the "jumps" becomes much more dramatic in this 17 dimensional case. The amount of noise increases fast with the increasing of dimensionality, which makes it much more difficult to only keep useful information for change-point detection. Another appearance of the distraction of noise

FIGURE 4.15: S&P500: Snapshot of Ticker "A" from Yahoo!Finance

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17
[1,]	0.05	-0.18	-0.59	-0.89	0.20	0.60	0.70	0.36	0.31	-0.05	0.22	0.44	-0.32	0.13	0.45	0.41	0.65
[2,]	0.90	0.23	-0.02	-0.45	-1.22	-0.10	-0.04	-0.16	0.60	0.09	0.08	-0.21	0.63	0.40	0.15	0.19	0.73
[3,]	-0.30	0.12	-0.38	0.08	-0.18	-0.70	0.34	0.36	3.30	-0.76	-0.04	-0.22	0.38	-0.14	-0.15	-0.17	-0.63
[4,]	-0.55	-0.06	-0.19	-1.43	-0.20	-0.14	-0.25	-0.21	-0.15	-1.41	0.06	-0.76	-0.44	-0.16	-0.45	-0.22	-0.40
[5,]	-0.15	-0.31	0.51	0.15	-0.20	-0.45	-0.30	0.00	0.09	0.89	-0.41	-0.07	-0.15	-0.28	-0.04	-0.03	-0.88
[6,]	-1.26	-0.80	-0.61	-0.61	0.16	0.19	0.20	-0.25	-0.94	-0.66	-0.27	0.24	-0.14	-0.22	-0.06	-0.13	0.63
[7,]	0.76	0.00	0.87	-0.24	-0.56	-0.34	0.05	0.09	0.53	1.45	0.08	0.21	0.35	0.08	0.10	0.13	-0.95
[8,]	0.81	0.98	0.86	0.55	0.20	0.39	-0.04	0.05	0.87	1.72	0.17	0.17	0.63	-0.12	-0.25	-0.03	0.50
[9,]	0.49	0.20	0.37	0.35	0.80	0.10	0.24	-0.46	-0.24	-0.85	0.42	-0.05	0.64	-0.03	-0.22	0.05	1.70
[10,]	0.30	-0.13	0.37	0.16	0.30	0.65	0.65	-0.48	-0.07	0.83	0.19	0.75	1.48	0.19	0.13	-0.06	0.90
[11,]	0.20	0.62	0.18	-0.01	0.34	-0.07	0.02	-0.40	0.15	-0.05	-0.13	0.06	0.02	0.09	-0.02	0.24	1.08
[12,]	0.13	-0.65	-1.03	0.92	0.21	-0.17	0.05	0.12	-0.15	-0.07	-0.16	-0.21	-0.20	-0.06	0.69	0.15	-0.28
[13,]	0.42	0.23	0.86	-0.04	1.19	0.95	0.38	0.40	1.33	1.77	0.16	1.65	1.18	0.07	-0.30	-0.05	0.50
[14,]	0.18	1.20	-0.27	0.16	-0.01	0.39	0.15	-0.42	-0.04	0.75	0.12	0.99	0.30	-0.32	-0.28	-0.01	1.66
[15,]	-0.13	0.30	0.23	0.12	0.42	0.45	0.40	0.08	-0.13	-0.83	0.22	0.56	0.30	-0.10	-0.57	0.03	2.69

FIGURE 4.16: S&P500: Snapshot of Differenced Data

can be found in the time period Mar/2007 to May/2008. It is still reasonable to group all points in this interval as coming from one same distribution depending on the continuous trend. However, the values are not stable as what we expect. This is significantly shown from the plots of eigenvalues 2, 3, 4, 14 and 15.

Our model still works fine in 17 dimensional space but it's reasonable to guess that the estimation will get quite unstable with the dimension getting higher. We are not going to discuss change-point detection in high dimension space here, since it's another independent topic from low dimensional space. In traditional statistical inference, it is totally different even for the estimation of a single covariance matrix. Bunch of shrinkage methods are already there for the estimation in high dimensional space, however, the hardest part of change-point detection with our Bayes model in high dimensional space is the following. Which shrinkage method is a proper replacement of traditional Bayesian estimation? "Proper" means it filters out the useless noise as much as possible from a high dimensional space while keeping enough information for parameter change. Pessimistically, we wonder if it is possible to find a way to complete this task. So far, it's still an open and difficult question which need to be explored in the future.

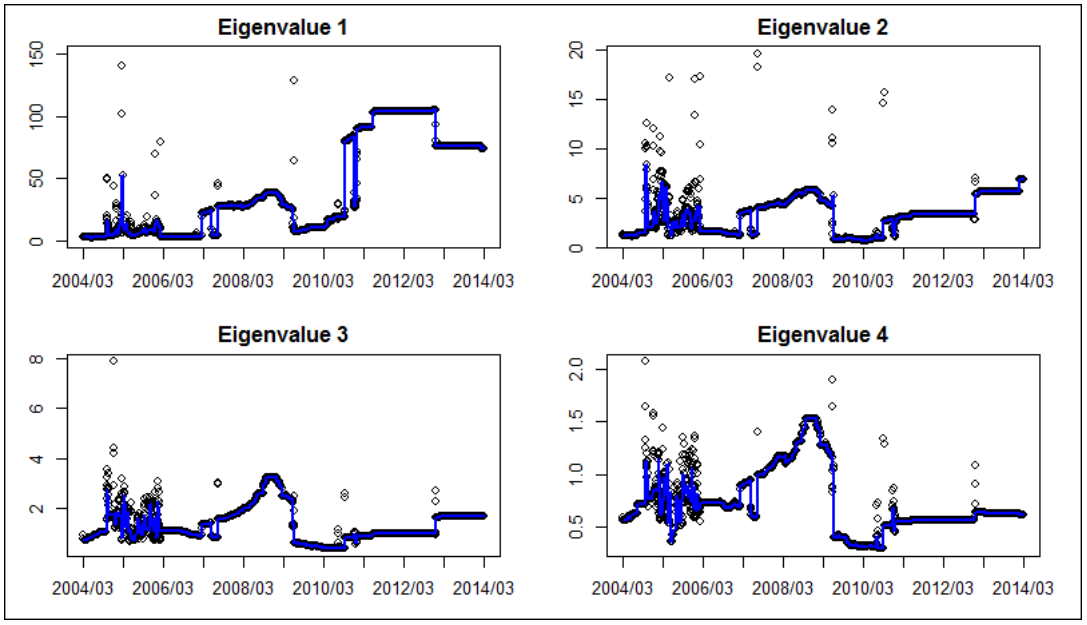

FIGURE 4.17: S&P500: Eigenvalue 1∼4 of Estimated Covariance Matrices

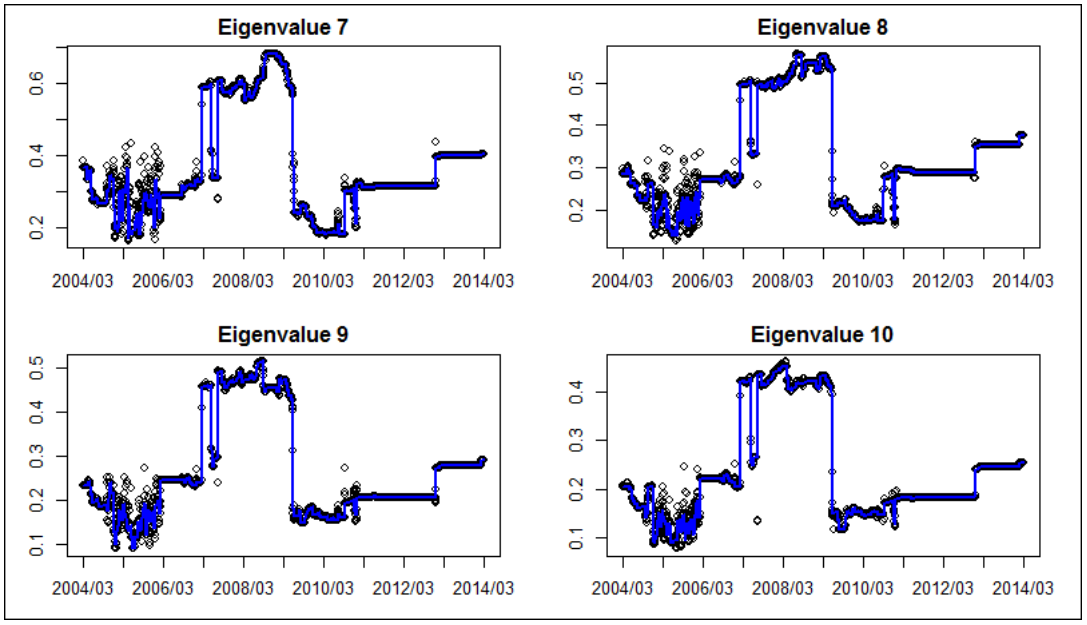

FIGURE 4.18: S&P500: Eigenvalue 7∼10 of Estimated Covariance Matrices

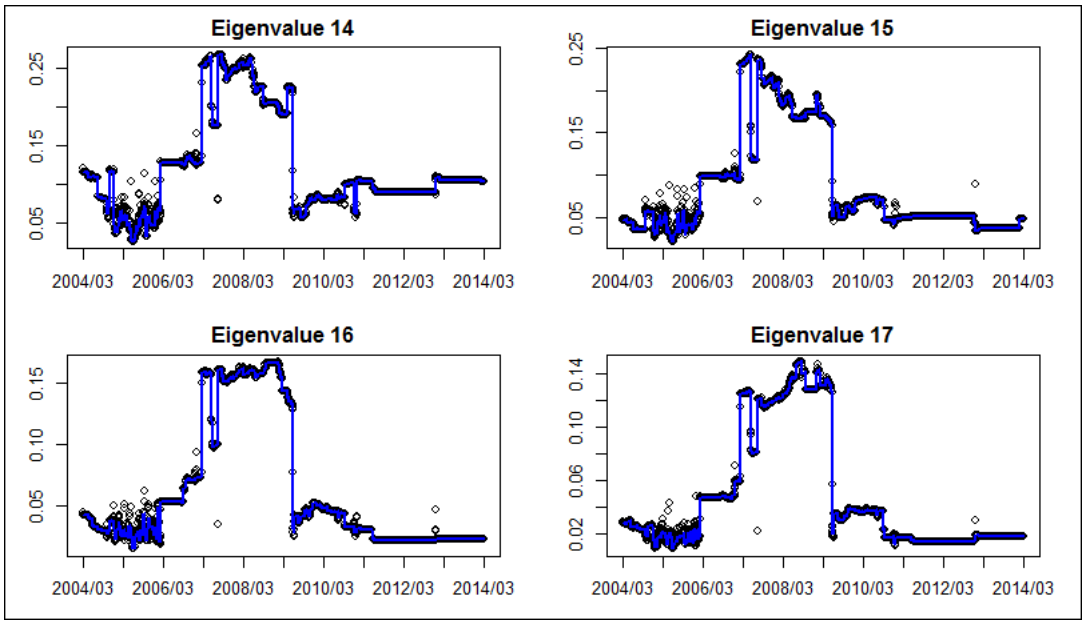

FIGURE 4.19: S&P500: Eigenvalue 14∼17 of Estimated Covariance Matrices

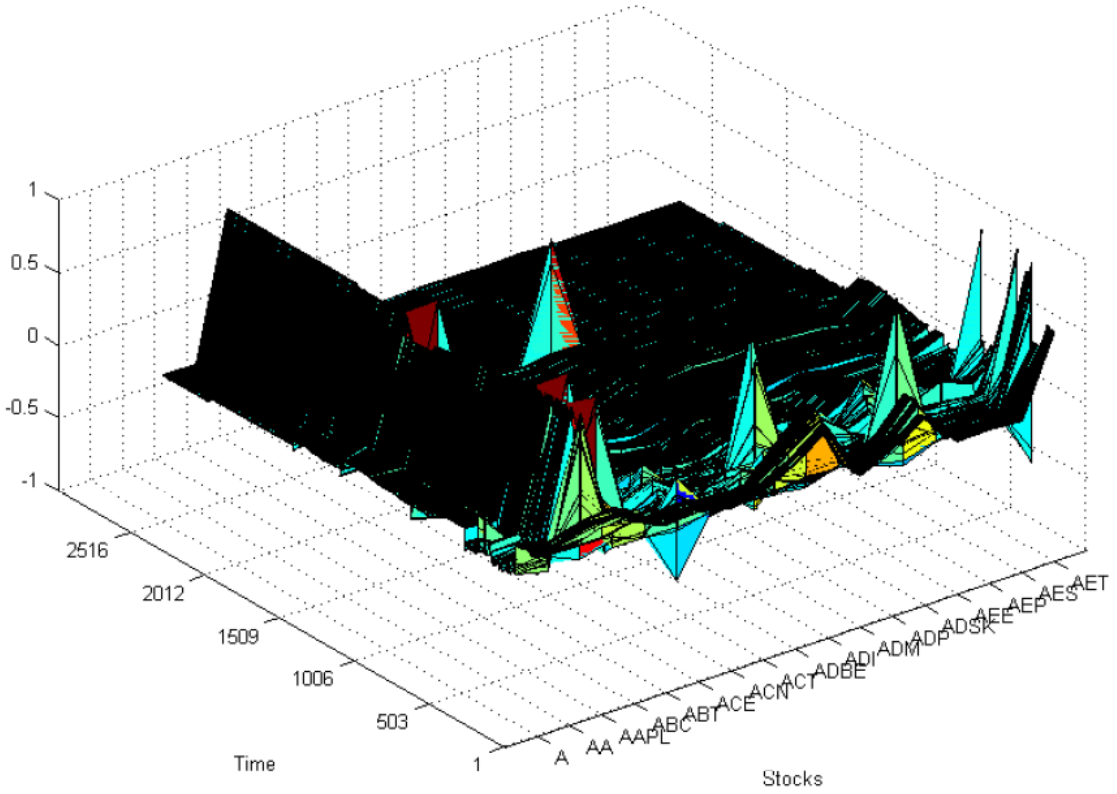

FIGURE 4.20: S&P500: First Eigenvector of Estimated Covariance Matrices

Chapter 5

Conclusions

We focus on change-point detection for the time series data in finance filed. The observations in finance filed are usually vectors, which means coming from a multiple dimensional space. Thus, the parameters are no longer a simple scalar but a vector or even a matrix. In fact, what we are interested in is the change-point in covariance matrix. In multi-dimensional space, we are facing two major challenges. The first one is, much more noise exists in multiple dimensional space than that in one dimensional space. We expect a proper model which is able to filter out the noise as much as possible while it can keep enough useful information for parameter change. The second challenge is, with a sequence of estimated covariance matrices, how do we measure the change. It's quite easy to find the change in one dimensional space since the parameter is a scalar. The most intuitively way is to plot the series of parameters along the time and figure out the existence of significant "jumps". However, for matrices in multiple dimensional space, it's impossible to plot them directly. It's not proper to use the determinant of a matrix as the measure, since a very small change of one element of a matrix may cause the determinant changing dramatically, which should not be considered as a matrix change.

We employ a Bayes model proposed by Lai and Xing(2011) to complete the change-point detection on covariance matrix. By combining the forward filter and backward filter,

the model outputs the estimated parameter at each time. The estimated parameter is a weighted average of a series of posterior means. The assumption of exponential family contributes to the closed form of final estimation, and it simplifies the computational complexity.

The eigen-structure is chosen to measure the the change of matrices, which is motivated by the spirit of principal component analysis (PCA). PCA is equivalent to eigen-decomposition to a covariance matrix. The eigenvectors decompose the variation of a covariance matrix into several orthogonal directions. Note that all these directions are in one dimensional space and the amount of variation in this direction is measured by the corresponding eigenvalue. Therefore, the eigen-structure is a good choice for projecting the variation of multiple dimensional space into several one dimensional spaces. If the eigen-structure of a matrix changes, we believe that the inner structure of a matrix has changed. Besides, it turns out that eigen-structure is a better choice than the determinant, as the determinant is the production of all eigenvalues. The counteraction of the change of eigenvalues may result in a unchanged determinant.

We also have provided a few algorithms to improve the computational speed. In the computation of basic Bayes model, the maximum number of weights involving in calculation is as the same order as n^2, which results in an unbounded computational complexity. The situation may get even worse in multi-dimensional space. Thus, we equip the Bayes model with BCMIX approximation in order to bound the number of weights. Besides, as we have to run the model at least hundreds of times in one scenario in simulation studies, we implement the whole model in C language to save computational time. The trade-off is we have to code all functions ourselves, especially for eigen-decomposition. Two algorithms have been proposed for low and high dimensional case respectively. All the algorithms improve the computational speed significantly.

From the results of simulation, it turns out that, our model performs well on capturing change-point in covariance matrix, especially when the number of change-point

is not big. The model also performs well in the real data analysis, in the swap rate data, the treasury constant maturity rate data and the S&P 500 data. From the plots of eigenvalues of estimated covariance matrix, we are able to do rough segmentation. It's almost impossible to do it precisely because in several short time interval, the eigenvalues jump up and down frequently. However, it might be caused by the natural behavior of the distribution. As shown in the simulation part with change-point occurs randomly with high probability, the true parameter itself changes frequently in this period of time. Thus, in practice, instead of trying to figure out all change-points, it might be better to figure out those intervals with constant value.

Bibliography

[1] AGRESTI, A. *Categorical Data Analysis*, second ed. Wiley & Sons, New Jersey, 2002.

[2] Attaway, S. *MATLAB: A Practical Introduction to Programming and Problem Solving*, second ed. Elsevier, 2012.

[3] Bai, J. Estimation multiple breaks one at a time. *Econometric Theory 13* (1997b), 315–352.

[4] Bai, J. Estimating high dimensional covariance matrices and its applications. *Annals of Economics and Finance 12*, 2 (2011), 199–215.

[5] Bai, J., and Perron, P. Computation and analysis of multiple structural change models. *J. Appl. Econometrics 18* (2003), 1–22.

[6] Barry., D., and Hartigan, J. A. A bayesian analysis for change point problems. *J. Amer. Statist. Assoc. 88* (1993), 309–319.

[7] Barth, W., Martin, R. S., and Wilkinson, J. H. Calculation of the eigenvalues of a symmetric tridiagonal matrix by the method of bisection*. *Numerische Mathematik 9* (1967), 386–393.

[8] Carlin, B. P., Gelfand, A. E., and Smith, A. F. M. Hierarchical bayesian analysis of changepoint problems. *Appl. Statist. 41* (1992), 389–405.

[9] Casella, G., and George, E. I. Explaining the gibbs sampler. *The American Statistician 46*, 3 (1992), 167–174.

[10] Crawley, M. J. *The R Book*, second ed. John Wiley & Sons, 2013.

[11] Deitel, P. J., Deitel, H. M., and Deitel, A. *C: How to Program.* Pearson, 2012.

[12] Diaconis, P., and Ylvisaker, D. Conjugate priors for exponential families. *Ann. Statist. 7* (1979), 269–281.

[13] Draper, N. R., and Smith, H. *Applied Regression Analysis.* Wiley & Sons, 1998.

[14] Enders, C. K. *Applied Missing Data Analysis.* The Guilford Press, New York, 2010.

[15] Francis, J. The QR transformation, I. *The Computer Journal 4*, 3 (1961), 265–271.

[16] Francis, J. The QR transformation, II. *The Computer Journal 4*, 3 (1962), 332–345.

[17] Friedman, J., Hastie, T., and Tibshirani, R. Sparse inverse covariance estimation with the graphical lasso.

[18] Gelfand, A. E., and Smith, A. F. M. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association 85*, 410 (1990), 398–409.

[19] Han, J., Kamber, M., and Pei, J. *Data Mining: Concepts and Techniques*, third ed. Elsevier Inc, 2012.

[20] Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning*, second ed. Springer, 2009.

[21] Horn, R. A., and Johnson, C. R. *Matrix Analysis (Ch5: Norms for Vectors and Matrices)*. Cambridge University Press, Cambridge, England, 1990.

[22] Householder, A. S. Unitary triangularization of a nonsymmetric matrix. *Journal of the ACM 5*, (4) (1958), 339–342.

[23] Jeffers, J. Two case studies in the application of principal component. *Applied Statistics 16* (1967), 225–236.

[24] Jolliffe, I. *Principal Component Analysis*. Springer Verlag, New York, 1986.

[25] Kernighan, B. W., and Ritchie, D. M. *The C Programming Language*. Prentice Hall, 1988.

[26] Kublanovskaya, V. N. On some algorithms for the solution of the complete eigenvalue problem. *USSR Computational Mathematics and Mathematical Physics 1*, 3 (1963), 637–657.

[27] Kullback, S. Letter to the editor: The kullback–leibler distance. *The American Statistician 41*, (4) (1987), 340–341.

[28] Kullback, S., and Leibler, R. On information and sufficiency. *Annals of Mathematical Statistics 22*, (1) (1951), 79–86.

[29] Kwan, C. C. Y. An introduction to shrinkage estimation of the covariance matrix: A pedagogic illustration. *Spreadsheets in Education 4* (2011).

[30] LaBudde, C. The reduction of an arbitrary real square matrix to tridiagonal form using similarity transformations. *Mathematics of Computation (American Mathematical Society) 17*, (84) (1963), 433–437.

[31] Lai, T. L., Liu, T., and Xing, H. A bayesian approach to sequential surveillance in exponential families. *To appear in Comm. Statist. Theory Methods, Special Issue in honor of S. Zacks* (2009).

[32] Lai, T. L., and Xing, H. *Statistical Models and Methods for Financial Markets*. Springer, New York, 2008.

[33] Lai, T. L., and Xing, H. A simple bayesian approach to multiple change-points. *Statistica Sinica 21*, 2011 (2011), 539–569.

[34] Langly, P. *Elements of Machine Learning*. Morgan Kauffmann Publishers, 1996.

[35] Lipschutz, S., and Lipson, M. L. *Theory and Problems of Linear Algebra*, second ed. Mcgraw-Hill, 2004.

[36] McNeil, A. J., Frey, R., and Embrechts, P. *Quantitative Risk Management*. Princeton University Press, 2005.

[37] Ortega, J. M., and Kaiser, H. F. The LL^T and QR methods for symmetric tridiagonal matrices. *The Computer Journal 6*, 1 (1963), 99–101.

[38] Pourahmadi, M. *High-Dimensional Covariance Estimation*. John Wiley & Sons, 2013.

[39] Rencher, A. C. *Methods of Multivariate Analysis*, second ed. John Wiley & Sons, 2002.

[40] Rizzo, M. L. *Statistical Computing with R*. Chapman & Hall/CRC, 2008.

[41] Robert, C., and Casella, G. *Monte Carlo Statistical Methods*, second ed. Springer-Verlag, New York, 2004.

[42] Shao, J. *Mathematical Statistics*, second ed. Springer, 2003.

[43] Shumway, R. H., and Stoffer, D. S. *Time Series Analysis and Its Applications*, third ed. Springer, 2011.

[44] Simar, H. *Applied Multivariate Statistical Analysis*, second ed. Springer.

[45] Spector, P. *Data Manipulation with R*. Springer, 2008.

[46] Stein, E. M., and Shakarchi, R. *Real Analysis*. Princeton University Press, 2005.

[47] Trefethen, L. N., and David Bau, I. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[48] Wang, H. Bayesian graphical lasso models and efficient posterior computation. *Bayesian Analysis 7*, 4 (2012), 867–886.

[49] Watkins, D. S. *Fundamentals of Matrix Computations*, second ed. John Wiley & Sons, 2002.

[50] Whishart, J. The generalised product moment distribution in samples from a normal multivariate population. *Biometrika 20A*, 1-2 (1928), 32–52.

[51] WREDE, R., AND SPIEGEL, M. R. *Theory and Problems of Advanced Calculus*, second ed. Mcgraw-Hill, 2002.

[52] ZOU, H., HASTIE, T., AND TIBSHIRANI, R. Sparse principal component analysis. *Journal of Computational and Graphical Statistics 15*, 2 (2006), 265–286.

Appendix

Proof of the Important Observation. Let's firstly consider the case of $n = 2$, say two matrices A, B such that $AB = BA$ and both of them are diagonalizable. Thus, A and B commute with each other. Suppose λ is an eigenvalue of A, denoting by V_λ the eigenspace of λ. If we regard A, and B as linear transformations on the $m \times m$ vector space V, then, A is diagonalizable means all the eigenspaces span the entire vector space V. Since A,B can commute with each other, we can get the following observation:

$$A(Bv) = B(Av) = \lambda Bv, \quad \forall v \in V_\lambda$$

This means, $\forall v \in V_\lambda$, $Bv \in V_\lambda$. Therefore, every eigenspace V_λ of A is an invariant subspace of B. Then, if we denote all eigenspaces of A as $V_{\lambda_1}, ..., V_{\lambda_s}$, they are the invariant subspaces of B and span the entire vector space V. Thus, we can block B as $diag\{B_1, ..., B_s\}$ by these invariant subspaces. Since B is diagonalizable, so is each B_i. Since the diagonalization for each B_i is completed in each V_{λ_i} and we know A is diagonalized by all V_{λ_i}, thus, A and B are simultaneously diagonalizable.

Induction of hypothesis: suppose for general n, it is correct. Consider A_{n+1} which commutes with every $A_i, i = 1, 2, ..., n$. Then we have a basis $(e_1, ..., e_m)$ of the entire vector space V, each of whose member is a common eigenvector of A_i. Then, similarly, since all the common eigenspaces are the invariant subspaces of A_{n+1} by commutability, so A_{n+1} can firstly be semi-diagonalized as $diag\{A^1_{n+1}, ..., A^t_{n+1}\}$, then diagonalize every block as we do in the case of $n = 2$.

Proof of the Sufficient Condition. Since $A_1, ..., A_n$ commute with each other, so A commute with all $A_1, ..., A_n$. By the conclusion of the important observation, we know that A and $A_1, ..., A_n$ can be diagonalized simultaneously. That is, there exists an invertible matrix P such that,

$$P^{-1}AP = \sum\nolimits_{i=1}^{n} w_i P^{-1} A_i P$$

Then we can have

$$diag\{\lambda_1, ..., \lambda_m\} = \sum\nolimits_{i=1}^{n} w_i diag\{\lambda_1^t, ..., \lambda_m^t\}$$

According to the proof of the important observation we know that for a series of diagonalizable matrices who can commute with each other, they will have the common eigenvectors. Therefore, if we denote $\tilde{v}_k$ and $\tilde{v}_k^i$ as the unit eigenvector with same sign of A and A_i respectively, then $\tilde{v}_k = \tilde{v}_k^i$. Hence,

$$\tilde{v}_k = 1 \cdot \tilde{v}_k = \sum\nolimits_{i=1}^{n} w_i \tilde{v}_k = \sum\nolimits_{i=1}^{n} w_i \tilde{v}_k^i, \quad \text{for } k = 1, 2, ..., m.$$