PENALIZED MAXIMUM LIKELIHOOD FACTOR ANALYSIS

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Jang Hoon Choi

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Gary W. Oehlert and
Hui Zou, Advisers

August 2010

Acknowledgements

I would like to thank all people who have assisted and inspired me to successfully complete my doctoral study and contributed to my time in graduate school.

I am very fortunate that I could finish my dissertation with the guidance of my two advisors, Dr. Gary Oehlert and Dr. Hui Zou. I am deeply thankful to Dr. Gary Oehlert, my Ph.D. advisor, for his creative ideas for the thesis topic and broad knowledge related to the research. I could establish ground work for my research and finish my thesis due to his help and advice. He also helped me to write good statements. I am greatly indebted to Dr. Hui Zou, my Ph.D. co-advisor, for his exceptional knowledge and clear directions. He always showed me how to approach the problems and helped me to solve them. I am especially grateful to Dr. Hui Zou for his continuous support for my research work from beginning to end so that I could publish a paper and complete my thesis through his guidance.

I thank Dr. Snigdhansu Chatterjee for relieving my life with his flexibility and easy-going attitude and for serving on my Ph.D. committee. I thank Dr. Saonli Basu for her teaching and for serving on my Ph.D. committee.

I am grateful to Dr. Yuhong Yang for his kindness and positive attitude. I am thankful to Dr. Birgit Grund for her assistance and guidance with respect to statistical knowledge and career.

I am thankful to the staff members Mary Hildre, Jane Sell, and Myrna Klitzke for their administrative help.

I am thankful to Wonho Chung for his emotional help when I was in difficult times and Ka Young Park and Do Hyang Kim for the help while I was away from the school.

Finally I thank my parents for their assistance. I could not study and complete my graduate study without their love and support.

Abstract

Factor analysis represents multivariate data as linear combinations of fewer random quantities called factors. Maximum likelihood is a popular method for finding the factor loadings, that is, the linear coefficients, and the ML estimates can be found using the EM algorithm. One problem with factor analysis, including ML estimation, is that the loadings are not unique. Practitioners use "factor rotation" to choose a unique set of coefficients that is more interpretable. This dissertation introduces a penalized maximum likelihood technique for estimating loadings that produces interpretable loadings without rotation. This method takes advantage of the propensity of the Lasso penalty to estimate some coefficients as exact zeros. We also provide an efficient algorithm for computing the estimates.

Contents

1 Introduction **1**

1.1 Definition, Brief History, and Application of the Factor Analysis . . . 1

1.2 Review on Factor Analysis 5

1.3 Estimation of Factor Models 8

1.3.1 The Heywood Case 8

1.3.2 The Principal Component Method 9

1.3.3 The Maximum Likelihood Method 13

1.3.4 Minres Method 18

1.3.5 Canonical Factor Analysis 19

1.3.6 Multiple-Group Method 20

1.4 Review of Factor Rotation 24

1.4.1 Quartimax Criterion 26

1.4.2 Varimax Criterion 28

1.4.3 Orthomax Criteria 30

1.5 Factor Scores 31

2 Review on Lasso and Adaptive Lasso **33**

2.1 Lasso 33

2.1.1 Lasso Regression 34

2.1.2 Least Angle Regression for Lasso 35

2.2 Adaptive Lasso 38

3 Proposed Approach and Main Results 40

3.1 Notation for the model 40

3.2 ℓ_1-Penalized Factor Analysis 41

3.3 Algorithm 44

3.4 Simulation 49

3.5 Real Data Example 61

3.5.1 Body fat data 61

3.5.2 Holzinger and Swineford (HS) 66

3.5.3 Parkinsons 72

3.5.4 Image Segmentation 78

4 Conclusions and Future Research 90

4.1 Conclusions 90

4.2 Future Research 92

4.2.1 Extension to Other Factor Models and Analysis Methods . . . 92

4.2.2 Measuring instruments 93

4.2.3 Confidence Level 94

References 95

A 5-Fold Cross Validation 98

B Q-norm Measurement 112

B.1 Simulations by Q-norm 112

B.2 Real data by Q-norm 120

B.2.1 Body fat data 120

B.2.2 Holzinger and Swineford (HS) 123

B.2.3 Parkinsons 124
B.2.4 Image Segmentation 129

C Proof of Ascent property of Algorithm 1 132

D R code for algorithm of Sparse Factor Analysis using Lasso (SFAL) 134

D.1 SFAL1 134
D.2 Bodyfat 136
D.2.1 Bodyfat: Scree Plot 136
D.2.2 Bodyfat: Lasso Plot 138
D.3 Parkinsons 140
D.3.1 Parkinsons: Scree Plot 140
D.3.2 Parkinsons: Lasso Plot 142
D.4 Image Segmentation 145
D.4.1 Brickface: Scree Plot 145
D.4.2 Brickface: Lasso Plot 147
D.4.3 Sky: Scree Plot 149
D.4.4 Sky: Lasso Plot 151
D.4.5 Foliage: Scree Plot 153
D.4.6 Foliage: Lasso Plot 155
D.4.7 Cement: Scree Plot 158
D.4.8 Cement: Lasso Plot 160
D.4.9 Windows: Scree Plot 162
D.4.10 Windows: Lasso Plot 164
D.4.11 Path: Scree Plot 166
D.4.12 Path: Lasso Plot 168
D.4.13 Grass: Scree Plot 171
D.4.14 Grass: Lasso Plot 173

List of Tables

1.1 The estimated factor loadings, communalities, and uniquenesses where 97 percent of the total standardized sample variance is explained by three common factors 12

1.2 Comparison of the factor model using the maximum likelihood method with the one using the principal component method for $q = 3$ factor with the same data as in section Section 1.3.2 15

1.3 The original factor loadings and the rotated factor loadings by the maximum likelihood method are shown using the same data 25

2.1 The values of β for different values of λ in "diabetes" data of R package; As λ increases, $\sum \mid \beta_j \mid$ decreases and some β_j become zero; λ is chosen such that some β start to become zero 36

3.1 Averages based on 100 replications(top: model 1, middle: model 2, bottom: model 3). Numbers in $(\cdot)$ are standard errors. 52

3.2 A sequence of q and corresponding KL loss for Body fat data 62

3.3 A sequence of λ and corresponding KL for Body fat data with q = 5 :Lasso 63

3.4 A sequence of λ and corresponding KL for Body fat data with q = 5: ALasso 63

3.5 The estimates of factor loadings and uniqueness of body fat data from Lasso (top) with penalty 18, ALasso (middle) with penalty 2, and rotation (bottom), respectively. For ALasso penalty, γ of 1 is used. Iteration converging gap of 0.0001 was used for β 65
3.6 A sequence of λ and corresponding KL for HS data with q = 4 :Lasso 69
3.7 A sequence of λ and corresponding KL for HS data with q = 4 :ALasso 69
3.8 HS data: Three estimates for factor loadings. The starting values are given as close values by factanal in R packages without rotation, with rotation, and random choices, respectively. 71
3.9 HS data: Comparison of factor loadings. top: Lasso with λ of 25; middle: ALasso with λ of 15; bottom: rotation (varimax) curtailed at 0.1. 73
3.10 $q = 8$ gives the best MLE model for Parkinson's disease data. 74
3.11 $\lambda = 3$ gives the best Lasso model for Parkinson's disease data. 74
3.12 $\lambda = 1$ gives the best ALasso model for Parkinson's disease data. . . . 75
3.13 The estimates of factor loadings and uniqueness of Parkinson's disease data from Lasso (left) and ALasso (right) with penalty 4 and 1 respectively. For ALasso penalty, γ of 1 is used. Iteration converging gap of 0.0001 was used for β 77
3.14 $q = 7$ gives the best MLE model for brickface image data. 79
3.15 $\lambda = 1$ gives the best Lasso model for brickface image data. 79
3.16 $q = 7$ gives the best MLE model for sky image data. 80
3.17 $\lambda = 3$ gives the best Lasso model for sky image data. 80
3.18 $q = 8$ gives the best MLE model for foliage image data. 82
3.19 $\lambda = 10$ gives the best Lasso model for foliage image data. 82
3.20 $q = 3$ gives the best MLE model for cement image data. 83
3.21 $\lambda = 13$ gives the best Lasso model for cement image data. 83

3.22 $q = 7$ gives the best MLE model for window image data. 85
3.23 $\lambda = 29$ gives the best Lasso model for window image data. 85
3.24 $q = 8$ give the best MLE model for path image data. 87
3.25 $\lambda = 12$ gives the best Lasso model for path image data. 87
3.26 $q = 5$ gives the best MLE model for grass image data. 88
3.27 $\lambda = 38$ gives the best Lasso model for grass image 89

A.1 Averages based on 30 replications(from top to bottom: model 1, 2, 3, 4). Numbers in $(\cdot)$ are standard errors. 111

B.1 Q-norm vs λ: 50 replications for each λ: top: Lasso, bottom: ALasso 116
B.2 A sequence of q and corresponding Q-norm loss for Body fat data . . 121
B.3 A sequence of λ and corresponding Q-norm for Body fat data with q = 5 :Lasso 121
B.4 A sequence of λ and corresponding Q-norm for Body fat data with q = 5 :ALasso 121
B.5 A sequence of λ and corresponding Q-norm for HS data with q = 4 :Lasso 124
B.6 A sequence of λ and corresponding Q-norm for HS data with q = 4 :ALasso 124
B.7 $q = 4$ gives the best MLE model for Parkinson's disease data. 126
B.8 $\lambda = 250$ gives the best Lasso model for Parkinson's disease data. . . . 126
B.9 $\lambda = 60$ gives the best ALasso model for Parkinson's disease data. . . 127
B.10 $q = 2$ gives the best MLE model for cement image data. 129
B.11 $\lambda = 50$ gives the best Lasso model for cement image data. 130
B.12 $q = 3$ gives the best MLE model for grass image data. 130
B.13 $\lambda = 100$ gives the best Lasso model for grass image 131

List of Figures

2.1 $-$: coefficient shrinkage in the orthogonal design case; $\cdots$: 45° -line for reference 35

3.1 The y-axis shows the value of $KL(\text{mle})_{\text{v}}$ for q factors; top: model 1, middle: model 2, bottom: model 3. 55

3.2 Boxplots of RKLs of the Lasso, ALasso and oracleestimators with respect to the MLE (top: model 1, middle: model 2, bottom: model 3). 56

3.3 Pairwise comparison of ALasso and oracle. Thesolid straight line is the 45 degree line (top: model 1, middle: model 2, bottom: model 3). 57

3.4 Comparing the sparsity pursuit performance ofLasso, ALasso, and rotation of model 1 (top: Lasso, middle: ALasso, bottom:rotation(varimax) curtailed at 0.1). 58

3.5 Comparing the sparsity pursuit performance ofLasso, ALasso, and rotation of model 2 (top: Lasso, middle: ALasso, bottom:rotation(varimax) curtailed at 0.1). 59

3.6 Comparing the sparsity pursuit performance ofLasso, ALasso, and rotation of model 3 (top: Lasso, middle: ALasso, bottom:rotation(varimax) curtailed at 0.1). 60

3.7 Body fat data: KL loss vs. number of factors 63

3.8 Body fat data: KL vs. λ (Lasso): q = 5 64

3.9 Body fat data: KL vs. λ (ALasso): q = 5 64
3.10 LL of body fat data 66
3.11 PLL of body fat data for Lasso 67
3.12 PLL of body fat data for ALasso 67
3.13 HS Data: KL loss vs. number of factors 68
3.14 HS data: KL vs. λ (Lasso): q = 4 70
3.15 HS data: KL vs. λ (ALasso): q = 4 70
3.16 Parkinsons data: q = 8 gives the best MLE model. 75
3.17 Parkinsons data: KL loss vs. λ: q = 8 76
3.18 brickface: KL loss vs. number of factors and lambda 79
3.19 sky image: $\lambda = 3$ gives the best Lasso model. 81
3.20 foliage image: NLL vs. number of factors and lambda 83
3.21 cement image: KL loss vs. number of factors and lambda 84
3.22 window image: KL loss vs. number of factors and lambda 86
3.23 path image: KL loss vs. number of factors and lambda 88
3.24 grass image: KL loss vs. number of factors and lambda 89

4.1 Q-norm vs. number of factors. 93

A.1 The y-axis shows the value of $KL(\text{mle})_{\text{v}}$ for q factors; top: model 1; bottom: model 2. 101
A.2 The y-axis shows the value of $KL(\text{mle})_{\text{v}}$ for q factors; top: model 3; bottom: model 4. 102
A.3 Boxplots of RKLs of the Lasso, ALasso and oracleestimators with respect to the MLE (top: model 1; bottom: model 2). 103
A.4 Boxplots of RKLs of the Lasso, ALasso and oracleestimators with respect to the MLE (top: model 3; bottom: model 4). 104

A.5 Pairwise comparison of ALasso and oracle. Thesolid straight line is the 45 degree line (top: model 1; bottom: model 2). 105
A.6 Pairwise comparison of ALasso and oracle. Thesolid straight line is the 45 degree line (top: model 3; bottom: model 4). 106
A.7 Comparing the sparsity pursuit performance ofLasso and ALasso of model 1 (top: Lasso, bottom: ALasso). 107
A.8 Comparing the sparsity pursuit performance ofLasso and ALasso of model 2 (top: Lasso, bottom: ALasso). 108
A.9 Comparing the sparsity pursuit performance ofLasso and ALasso of model 3 (top: Lasso, bottom: ALasso). 109
A.10 Comparing the sparsity pursuit performance ofLasso and ALasso of model 4 (top: Lasso, bottom: ALasso). 110

B.1 Q-norm vs. number of factors: top-model 1, middle-model 2, bottom-model3. 114
B.2 Model 1: Q-norm and zero counts vs λ: 50 replications for each λ . . 117
B.3 Model 2: Q-norm and zero counts vs λ: 50 replications for each λ . . 118
B.4 Model 3: Q-norm and zero counts vs λ: 50 replications for each λ . . 119
B.5 Body fat data: Q-norm vs. number of factors 121
B.6 Body fat data: Q-norm vs. λ (Lasso): q = 5 122
B.7 Body fat data: Q-norm vs. λ (ALasso): q = 5 122
B.8 HS Data: Q-norm vs. number of factors 123
B.9 HS data: Q-norm vs. λ (Lasso): q = 4 125
B.10 HS data: Q-norm vs. λ (ALasso): q = 4 125
B.11 Parkinsons data: q = 4 gives the best MLE model. 127
B.12 Parkinsons data: Q-norm vs. λ: q = 4 128
B.13 cement image: Q-norm vs. number of factors and lambda 130

B.14 grass image: Q-norm vs. number of factors and lambda 131

Chapter 1

Introduction

1.1 Definition, Brief History, and Application of the Factor Analysis

Factor analysis is a popular multivariate analysis method that is used to describe observed variables as linear combinations of a smaller number of unobserved random quantities called factors (Gorsuch, 1983). Researchers generally utilize this method to explore or establish the covariance or correlation relationships among observed random variables. Factors are random quantities that are unobservable and describe these relationships. Although they cannot be measured directly, they can be inferred indirectly from the observable attributes. These internal attributes can be structured as a smaller number of random variables than the surface attributes because of the correlations among the observed data. Therefore, factor analysis represents the underlying structure of the covariance relationships among many variables in simple and interpretable quantities. For example, in the domain of mental abilities, numerical ability influences both the student's addition and product tests where numerical ability is the internal attribute. In addition, the student's addition and product tests are the surface attributes. Here, surface attributes are influenced by internal attributes according to the traditional theory.

Harman (1976) stated that the originator of factor analysis is Spearman (1904). Spearman (1904) developed a psychological theory associated with one general factor and a number of specific factors that lead to the development of a two-factor theory. Some time later, some experimenters used group factors to overcome the inadequacies of Spearman's two-factor theory. Then Garnett (1919) developed the concept of multiple-factor analysis.

A natural approach involves representing the variables with sequentially determined factors where the factors are estimated to have a maximum of the variance in each stage. Pearson (1901) suggested this method of principal axes, and in the 1930s Hotelling developed a full method that is currently known as the principal component method (Harman, 1976).

The centroid method is another approach. This method was introduced as the computer became a useful device. It was then used when the principal component method was too arduous. The centroid method is usually viewed as a historical interest. Through the centroid, which refers to the average of the coordinates, the first axis of reference is obtained. The coordinate of each variable can be obtained on the reference axis such that the sum of the coordinates is zero, except the reference axis (Harman, 1976).

However, subject-matter specialists would not agree with the result of either the principal component or centroid method because they want to attain meaningful, unique results that are applicable to their areas of interest, such as intelligence and personality. Holzinger's bi-factor Theory and Thurstone's simple structure theory are good examples of attempting to attain meaningful, unique results (Harman, 1976).

Factor analysis has primarily been applied in the field of psychology. Kelley (1940) proposed a method for obtaining the highest social utility while maintaining individ-

ual liberties and rights as a way of extending the psychological use. Since 1950, as computer accessibility became widespread, applications of factor analysis have become popular other fields, such as meteorology, medicine, political science, taxonomy, archaeology, economics, sociology, and regional science (Harman, 1976).

Recent developments of factor analysis, along with the developments of high-speed computers, resolved early controversies over the psychological interpretations by facilitating a statistical method. In addition, most of the original techniques have been abandoned (Johnson and Wichern, 2007).

The following example illustrates how factor analysis works. Factor model can be written as factors multiplied by corresponding factor loadings plus error terms. As previously explained, factors are unobserved random variables attained from correlation relationships among observed variables; therefore, there are fewer factors than observed variables. Factor loadings are the correlation coefficients between the observed variables and factors. Then, the following covariance matrix

$$\Sigma = \begin{bmatrix} 7 & 4 & 15 & 6 \\ 4 & 8 & 12 & -3 \\ 15 & 12 & 50 & 18 \\ 6 & -3 & 18 & 43 \end{bmatrix}$$

can be divided into two terms, $\beta\beta' + \tau^2$

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 6 \\ -4 & 5 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & -4 \\ 2 & 1 & 6 & 5 \end{bmatrix} + \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

where β is a covariance matrix between data variable and factors and is called a matrix of factor loadings.

Σ is represented by a q = 2 orthogonal factor model. Since

$$\beta = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \\ b_{41} & b_{42} \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 6 \\ -4 & 5 \end{bmatrix}$$

and

$$\tau^2 = \begin{bmatrix} \tau_1^2 & 0 & 0 & 0 \\ 0 & \tau_2^2 & 0 & 0 \\ 0 & 0 & \tau_3^2 & 0 \\ 0 & 0 & 0 & \tau_4^2 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

where τ^2 is the portion of the variance not explained by factors, the communality of Y_{i1}, which is the portion of the variance explained by the q common factors, is

$$h_1^2 = b_{11}^2 + b_{12}^2 = 1^2 + 2^2 = 5$$

and the variance of Y_{i1} is

$$\begin{aligned} \sigma_{11}^2 &= (b_{11}^2 + b_{12}^2) + \tau_1^2 \\ &= h_1^2 + \tau_1^2 \\ &= 5 + 2 \\ &= 7 \end{aligned}$$

The other variables can similarly be divided into communality and uniqueness. More details are given in section Section 1.2.

Factor analysis is closely related to principal component analysis. Both methods analyze data by approximating the covariance matrix. However, there are some important differences between them. In factor analysis the variances associated with factors are modeled, whereas all of the observed variances are modeled in principal component analysis. Conceptually, factor analysis attempts to relate the measured underlying factors to theoretical concepts as well as data reductions; however, principal component analysis only tries to obtain data reductions.

1.2 Review on Factor Analysis

The factor analysis model is

$$Y_i - \mu_y = \beta X_i + \epsilon_i, i = 1, \ldots, n$$

where n is the number of observations. Y_i is an observed vector with p components with mean μ_y and β is a $p \times q$ matrix called factor loadings. X_i is an unobserved vector with q components, $q < p$, the components of which are called common factors and ϵ_i is an unobserved error with mean $0_{(p\times 1)}$ and is called the specific factor. We assume that X_i follows a normal distribution with mean $0_{(q\times 1)}$ and variance $\Sigma_{(q\times q)}$. At this time, however, we only consider where $Cov(X_i) = E(X_i X_i') = I_{(q\times q)}$, that is, the orthogonal case. Then $Y_i|X_i$ follows $N(\mu_y + \beta X_i, Cov(\epsilon_i))$, where $Cov(\epsilon_i) = \tau^2 = diag(\tau_1^2, \ldots, \tau_p^2)$. τ^2 is called uniqueness or specific variance.

Consequently, a covariance structure for Y is

$$\begin{aligned}\Sigma &= Cov(Y_i) \\ &= E(Y_i - \mu_y)(Y_i - \mu_y)' \\ &= E(\beta X_i X_i' \beta') + E(\epsilon_i X_i' \beta') + E(\beta X_i \epsilon_i') + E(\epsilon_i \epsilon_i') \\ &= \beta\beta' + \tau^2\end{aligned}$$

and $Cov(Y_i, X_i) = E(Y_i - \mu_y)X_i' = E(\beta X_i + \epsilon_i)X_i' = \beta$.

Now, communality is introduced. $Cov(Y_i) = \beta'\beta + \tau^2$ can be written as

$$\begin{aligned}Var(Y_{ij}) &= \beta_{j1}^2 + \ldots + \beta_{jq}^2 + \tau_j^2 \\ Cov(Y_{ij}, Y_{ik}) &= \beta_{j1}\beta_{k1} + \ldots + \beta_{jq}\beta_{kq}\end{aligned}$$

and $Cov(Y_i, X_i) = \beta$ can be written as

$$Cov(Y_{ij}, X_{ik}) = \beta_{jk}.$$

Communality is the portion of the variance of the variable contributed by the q common factors. Suppose the j-th communality is h_j^2. Then

$$\begin{aligned}Var(Y_{ij}) = \sigma_{jj} &= communality + uniqueness \\ &= h_j^2 + \tau_j^2 \\ &= (\beta_{j1}^2 + \ldots + \beta_{jq}^2) + \tau_j^2\end{aligned}$$

where $j = 1, \ldots, p$.

The j-th communality is the sum of squares of the loadings of the j-th variable on the q common factors (Johnson and Wichern, 2007).

When $q > 1$, there are multiple factor loadings that generate the same covariance matrix. Let T be any $q \times q$ orthogonal matrix. If we let $\beta^* = \beta T$ and $X_i^* = T'X_i$, then X_i^* has the same statistical properties as X_i since $E(X_i^*) = T'E(X_i) = 0$ and $Cov(X_i^*) = T'Cov(X_i)T = T'T = I_{(q\times q)}$. β and β^* yield the same covariances because $\beta\beta' = \beta^*\beta^{*'}$. The factor analysis model

$$Y_i - \mu_y = \beta X_i + \epsilon_i = \beta TT'X_i + \epsilon_i = \beta^* X_i^* + \epsilon_i$$

produces the same covariance matrix Σ since

$$\Sigma = \beta\beta' + \tau^2 = \beta TT'\beta' + \tau^2 = (\beta^*)(\beta^*)' + \tau^2.$$

A particular set of loadings needs to be chosen. A good set is one that is easily interpreted. Usually, this means a sparse solution with many zeros. Factor rotation is one approval to finding the solution, as it rotates the coordinate system for Y on X to ease interpretation. Harman (1976) described three methods of orthogonal factor rotation: quartimax, varimax, and orthomax.

Quartimax rotation is an orthogonal rotation that minimizes the inner-product function of the columns of the factor matrix or equivalently maximizes the variance of squared factor loadings or the sum of the fourth powers of factor loadings. This type of rotation often tends to increase the large original loadings and reduce the small original loadings. It also attempts to simplify the description of each variable.

Varimax rotation is an orthogonal rotation that maximizes the variance of the squared loadings of a factor on all of the variables after modifying the vectors of the variables to unit length. After the rotation is completed, the vectors are brought back to their original length. This rotation will allow each factor to have either large or small loadings of any variable. Therefore, for each factor, a few variables will have high loadings, whereas the rest of the variables will have near zero loadings. This is the most common rotation method.

Orthomax rotation is a weighted composite between quartimax rotation and varimax rotation. If the weight on quartimax is α and the weight on varimax is β, it is suggested that the best results are obtained when $\beta/(\alpha+\beta)$ is approximately half the number of common factors (Saunders, 1962).

1.3 Estimation of Factor Models

1.3.1 The Heywood Case

Spearman's fundamental theorem shows that a set of p variables is represented by only one factor and p unique factors if and only if

$$r_{jk}r_{lm} - r_{lk}r_{jm} = 0$$

where r_{ij} is the correlation between two variables i and j, and $j,k,l,m = 1,\ldots,p; j \neq k \neq l \neq m$.

The two factor model may be given by

$$\hat{z}_j = a_{j0}F_0 + u_jY_j, j = 1,\ldots,p, \tag{1.1}$$

where F_0 is the general factor and the Y_j are the p unique factors.

The correlations produced from the equation in (1.1) are $r_{jk} = a_{j0}a_{k0}$ assuming that the residuals disappear. After modification, we have

$$a_{e0}^2 \sum_{j<k=1}^{p} r_{jk} = \sum_{j<k=1}^{p} r_{ej}r_{ek}$$

where e is fixed, $j,k \neq e$. The above equation may be rewritten by

$$a_{e0}^2 = h_e^2 = \frac{\sum(r_{ej}r_{ek}; j,k = 1,\ldots,p, j,k \neq e, j<k)}{\sum(r_{jk}; j,k = 1,\ldots,p, j,k \neq e, j<k)} \tag{1.2}$$

$$= \frac{(\sum_{j=1}^{p} r_{ej})^2 - \sum_{j=1}^{p} r_{ej}^2}{2(\sum_{j<k=1}^{p} r_{jk} - \sum_{j=1}^{p} r_{ej})} \tag{1.3}$$

where e is fixed, $j \neq e$.

The conditions for the solution in the two-factor model are disappearance of all tetrads, which means, zero second-order minors (except a diagonal element) of the correlation matrix. The "Heywood" case is the situation that if we have negative uniquenesses for the fit, we can meet those conditions (Harman, 1976).

1.3.2 The Principal Component Method

The covariance matrix Σ is represented by the spectral decomposition. If Σ has eigenvalue λ_i with $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_p \geq 0$ and corresponding eigenvector e_i, then

$$\begin{aligned} \Sigma &= \lambda_1 e_1 e_1' + \ldots + \lambda_p e_p e_p' \\ &= \beta_{(p\times p)} \beta_{(p\times p)}' \end{aligned}$$

where $\beta_{(p\times p)} = (\sqrt{\lambda_1} e_1 \ldots \sqrt{\lambda_p} e_p)$. Because this matrix product form represents the covariance matrix perfectly, there is no uniqueness. However, this form is not useful for factor analysis because the number of factors should be less than the number of variables. It is desirable to have a model that explains the covariance with a small number of common factors and leave the differences between the two as uniquenesses. Suppose the number of variables is p and the number of factors is q. Then we find q such that the last p - q eigenvalues are so small that we can ignore the contribution of $\lambda_{q+1} e_{q+1} e_{q+1}' + \ldots + \lambda_p e_p e_p'$ to Σ. We then have

$$\begin{aligned} \Sigma &\doteq \lambda_1 e_1 e_1' + \ldots + \lambda_q e_q e_q' + diag(\tau_1^2, \ldots, \tau_p^2) \\ &= \beta_{(p\times q)} \beta_{(p\times q)}' + \tau^2 \end{aligned}$$

where $\beta_{(p\times q)} = (\sqrt{\lambda_1} e_1 \ldots \sqrt{\lambda_q} e_q)$ and $\tau_1^2, \ldots, \tau_p^2$ are very small. This representation is the principal component solution.

Because the units of the variables of the original data may be different, standardization is required for a factor model. That is,

$$Z_{ij} = \frac{(y_{ij} - \bar{y}_j)}{\sqrt{s_{jj}}}, i = 1, \ldots, n, j = 1, \ldots, p$$

is required because some variables with large variances influence the determination of factor loadings too much. In this case, the sample covariance matrix, S, becomes the sample correlation matrix, R.

For the principal component method, the estimate of each factor loading is fixed independent of the number of factors. If $\tilde{\beta} = (\sqrt{\hat{\lambda}_1}\hat{e}_1, \ldots, \sqrt{\hat{\lambda}_m}\hat{e}_m)$ for $q = m$, and $\tilde{\beta} = (\sqrt{\hat{\lambda}_1}\hat{e}_1, \ldots, \sqrt{\hat{\lambda}_m}e_m, \ldots, \sqrt{\hat{\lambda}_n}\hat{e}_n)$ for $q = n$, $m < n$, $\sqrt{\hat{\lambda}_1}\hat{e}_1, \ldots, \sqrt{\hat{\lambda}_m}\hat{e}_m$ are the same for both cases.

One way of determining the number of factors q is to consider the residual matrix

$$S - (\tilde{\beta}\tilde{\beta}' + \tau^2).$$

If q's are chosen to ensure that the residual matrices are small enough, the least number of q among all q's that satisfy the small residual matrix condition is appropriate. The sum of squared entries of $(S - (\tilde{\beta}\tilde{\beta}' + \tau^2)) \leq \hat{\lambda}_{q+1}^2 + \ldots + \hat{\lambda}_p^2$. This means that if the right side of the inequality is small, then the left side should also be small.

The contribution to the total sample variance from the k-th common factor is

$$\tilde{\beta}_{1k}^2 + \ldots + \tilde{\beta}_{pk}^2 = (\sqrt{\hat{\lambda}_k e_k})'(\sqrt{\hat{\lambda}_k e_k}) = \hat{\lambda}_k.$$

Therefore, the proportion of the total sample variance due to the k-th factor equals $\frac{\hat{\lambda}_k}{s_{11}+\ldots+s_{pp}}$ for a sample covariance matrix, S, and $\frac{\hat{\lambda}_k}{p}$ for a sample correlation matrix, R. From this proportion, q is chosen to obtain the appropriately high proportion.

Consider the following correlation matrix:

$$\Sigma = \begin{bmatrix} 1.00 & (0.95) & 0.27 & 0.39 & 0.38 & 0.42 \\ 0.95 & 1.00 & 0.27 & 0.41 & 0.44 & 0.51 \\ 0.27 & 0.27 & 1.00 & (0.93) & 0.37 & 0.37 \\ 0.39 & 0.41 & 0.93 & 1.00 & 0.39 & 0.40 \\ 0.38 & 0.44 & 0.37 & 0.39 & 1.00 & (0.96) \\ 0.42 & 0.51 & 0.37 & 0.40 & 0.96 & 1.00 \end{bmatrix}.$$

Entries in parentheses in the above matrix show high correlations. Variables 1 and 2 and variables 5 and 6 are highly correlated and, thus, form groups. Variables 3 and 4 are also highly correlated. Therefore it is expected that two or three common factors can explain the linear relationships among the variables. Typically, q is set to equal the number of eigenvalues of the correlation matrix, R, greater than one, if the sample correlation matrix is factored (Johnson and Wichern, 2007). In this example, the eigenvalues λs are $(3.49, 1.28, 1.07, 0.07, 0.05, 0.03)$. In addition, q is set to be three because there are three eigenvalues greater than one. The proportion of the total sample variance due to the three common factors is

$$\frac{\hat{\lambda}_1 + \hat{\lambda}_2 + \hat{\lambda}_3}{p} = \frac{3.49 + 1.28 + 1.07}{6} = 0.97.$$

Consequently, 97% of the total standardized sample variance is explained by three common factors. The estimated factor loadings, communalities, and uniquenesses are given in Table Table 1.1.

To determine if this two-factor model is appropriate, we can check that

$$
\begin{aligned}
\tilde{\beta}\tilde{\beta}' + \tilde{\tau}^2 &= \begin{bmatrix} -.75 & .45 & -.47 \\ -.79 & .46 & -.38 \\ -.68 & -.69 & -.14 \\ -.76 & -.59 & -.22 \\ -.79 & .12 & .59 \\ -.81 & .16 & .55 \end{bmatrix} \begin{bmatrix} -.75 & -.79 & -.68 & -.76 & -.79 & -.81 \\ .45 & .46 & -.69 & -.59 & .12 & .16 \\ -.47 & -.38 & -.14 & -.22 & .59 & .55 \end{bmatrix} \\
&+ \begin{bmatrix} 0.02 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.02 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.04 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.03 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.02 \end{bmatrix} \\
&= \begin{bmatrix} 1.00 & 0.97 & 0.26 & 0.40 & 0.37 & 0.42 \\ 0.97 & 1.00 & 0.27 & 0.41 & 0.45 & 0.50 \\ 0.26 & 0.27 & 1.00 & 0.96 & 0.37 & 0.37 \\ 0.40 & 0.41 & 0.96 & 1.00 & 0.40 & 0.40 \\ 0.37 & 0.45 & 0.37 & 0.40 & 1.00 & 0.98 \\ 0.42 & 0.50 & 0.37 & 0.40 & 0.98 & 1.00 \end{bmatrix}
\end{aligned}
$$

Table 1.1: The estimated factor loadings, communalities, and uniquenesses where 97 percent of the total standardized sample variance is explained by three common factors

Var	$\tilde{\beta}_1$	$\tilde{\beta}_2$	$\tilde{\beta}_3$	$\tilde{h}_i^2$	$\tilde{\tau}_i^2$
1	-0.75	0.45	-0.47	0.98	0.02
2	-0.79	0.46	-0.38	0.98	0.02
3	-0.68	-0.69	-0.14	0.96	0.04
4	-0.76	-0.59	-0.22	0.97	0.03
5	-0.79	0.12	0.59	0.99	0.01
6	-0.81	0.16	0.55	0.98	0.02
Eigenvalues	3.49	1.28	1.07		
Cum. proportion	0.58	0.80	0.97		

is close to the correlation matrix R. The residual matrix between R and $\beta\beta' + \tau^2$ is

$$R - \beta\beta' - \tau^2 = \begin{bmatrix} 0.00 & -0.02 & 0.01 & -0.01 & 0.01 & -0.01 \\ -0.02 & 0.00 & -0.00 & 0.00 & -0.01 & 0.00 \\ 0.01 & -0.00 & 0.00 & -0.03 & 0.00 & -0.00 \\ -0.01 & 0.00 & -0.03 & 0.00 & -0.00 & 0.00 \\ 0.01 & -0.01 & 0.00 & -0.00 & 0.00 & -0.02 \\ -0.01 & 0.00 & -0.00 & 0.00 & -0.02 & 0.00 \end{bmatrix}.$$

The communalities $(0.98, 0.98, 0.96, 0.97, 0.99, 0.98)$ also show that a high portion of the sample variance is explained by three factors.

1.3.3 The Maximum Likelihood Method

If the distributions of X_i and the specific factors, ϵ_i, are assumed to be normal, then estimates of the factor loadings and uniquenesses can be obtained using the maximum likelihood method (Johnson and Wichern, 2007). The distribution of Y given β, τ^2 is normal with mean 0 and covariance matrix $\tau^2 + \beta'\beta$, and the likelihood is

$$L(\beta, \tau^2) \propto \det(\Sigma)^{-\frac{n}{2}} e^{-\frac{1}{2}\sum_{i=1}^{n}(Y_i - \bar{Y})\Sigma^{-1}(Y_i - \bar{Y})'}$$

The resulting log likelihood (LL) is

$$LL(\beta, \tau^2) \propto -\frac{n}{2}\ln\det(\Sigma) - \frac{1}{2}\sum_{i=1}^{n}(Y_i - \bar{Y})\Sigma^{-1}(Y_i - \bar{Y})'$$

The maximum likelihood estimates of β, τ^2 (called $\hat{\beta}$ and $\hat{\tau}^2$) can be obtained by maximizing the LL. Based on the invariance property of maximum likelihood estimates, the maximum likelihood estimate of the communality due to j-th factor is $\hat{\beta}_{1j}^2 + \ldots + \hat{\beta}_{pj}^2$. As a result, the proportion of the total sample variance due to the j-th factor is $\frac{\hat{\beta}_{1j}^2 + \ldots + \hat{\beta}_{pj}^2}{s_{11} \ldots + s_{pp}}$ where s_{ii} is the (i,i)-th entry of the sample covariance matrix

which is an estimate of the unknown population covariance matrix, Σ.

If Y_i is standardized to be $Z_i = V^{-1/2}(Y_i - \mu_i)$, the covariance matrix ρ is

$$\begin{aligned} \rho = V^{-1/2}\Sigma V^{-1/2} &= (V^{-1/2}\beta)(V^{-1/2}\beta)' + V^{-1/2}\tau^2 V^{-1/2} \\ &= \beta_z\beta_z' + \tau_z^2 \end{aligned}$$

where $V^{-1/2}$ is the diagonal matrix with the reciprocal of the sample standard deviations on the main diagonal of Y_i. Based on the invariance property of maximum likelihood estimators, the maximum likelihood estimator of ρ is

$$\rho = \hat{\beta}_z\hat{\beta}_z' + \hat{\tau}_z^2$$

where $\hat{\beta}_z = V^{-1/2}\hat{\beta}$.

The proportion of total standardized sample variance due to j-th factor $= \frac{\hat{\beta}_{1j}^2 + \ldots + \hat{\beta}_{pj}^2}{s_{11}\ldots + s_{pp}}$ $= \frac{\hat{\beta}_{1j}^2 + \ldots + \hat{\beta}_{pj}^2}{p}$.

Table Table 1.2 compares the factor model using the maximum likelihood method with the one using the principal component method by using $q = 3$ factor with the same data as in the principal component method section.

The residual matrix between R and $\beta\beta' + \tau^2$ is

$$R - \beta\beta' - \tau^2 = \begin{bmatrix} 0.00 & 0.00 & 0.00 & -0.00 & 0.01 & -0.00 \\ 0.00 & -0.00 & -0.00 & 0.00 & -0.00 & 0.00 \\ 0.00 & -0.00 & -0.00 & 0.00 & 0.00 & -0.00 \\ -0.00 & 0.00 & 0.00 & -0.00 & -0.00 & 0.00 \\ 0.01 & -0.00 & 0.00 & -0.00 & -0.00 & 0.00 \\ -0.00 & 0.00 & -0.00 & 0.00 & 0.00 & -0.00 \end{bmatrix}$$

The residual matrix from the maximum likelihood method is much smaller than that from the principal component method. Therefore, it seems that the maximum likelihood approach might be a more accurate method than the principal component approach.

One important objective of factor analysis is to determine the appropriate number of common factors. The likelihood ratio test can be used to ascertain the number of common factors, due to the normal assumption of the distribution. If the null hypothesis is that the q common-factor model is appropriate, then the null and alternate hypotheses are

$$\begin{aligned} H_0 &: \Sigma_{p\times p} = \beta_{p\times q}\beta'_{q\times p} + \tau^2_{p\times p} \\ H_1 &: \Sigma_{p\times p} \neq \beta_{p\times q}\beta'_{q\times p} + \tau^2_{p\times p} \end{aligned}$$

Table 1.2: Comparison of the factor model using the maximum likelihood method with the one using the principal component method for $q = 3$ factor with the same data as in section Section 1.3.2

	Principal Component				Maximum Likelihood			
Var	$\tilde{\beta}_1$	$\tilde{\beta}_2$	$\tilde{\beta}_3$	$\tilde{\tau}_i^2$	$\tilde{\beta}_1$	$\tilde{\beta}_2$	$\tilde{\beta}_3$	$\tilde{\tau}_i^2$
1	-0.75	0.45	-0.47	0.02	0.17	0.92	0.17	0.09
2	-0.79	0.46	-0.38	0.02	0.26	0.95	0.15	0.01
3	-0.68	-0.69	-0.14	0.04	0.18	0.09	0.98	0.01
4	-0.76	-0.59	-0.22	0.03	0.19	0.24	0.89	0.11
5	-0.79	0.12	0.59	0.01	0.93	0.19	0.20	0.07
6	-0.81	0.16	0.55	0.02	0.95	0.25	0.18	0.01
Cum. proportion	0.58	0.80	0.97		0.32	0.64	0.95	

Here, H_1 is a completely general alternative, so the alternate hypothesis is any other positive definite matrix.

Under H_0, log likelihood function is

$$\begin{aligned} LL(\beta, \tau^2) &= -\frac{n}{2}\ln\det(\hat{\Sigma}) - \frac{1}{2}tr[\sum_{i=1}^{n}(Y_i - \bar{Y})'\hat{\Sigma}^{-1}(Y_i - \bar{Y})] \\ &= -\frac{n}{2}\ln\det(\hat{\beta}\hat{\beta}' + \tau^2) - \frac{n}{2}tr[(\hat{\beta}\hat{\beta}' + \tau^2)^{-1}S_n] \end{aligned}$$

Suppose L_0 and L are maximum likelihood under H_0 and the unrestricted model. The likelihood ratio statistic for testing H_0 is

$$\begin{aligned} -2\ln\frac{L_0}{L} &= -2\ln\left(\frac{det(\hat{\Sigma})}{det(S_n)}\right)^{-n/2} + n[tr(\hat{\Sigma}^{-1}S_n) - p] \\ &= n\ln\left(\frac{det(\hat{\Sigma})}{det(S_n)}\right) \end{aligned}$$

since $tr(\hat{\Sigma}^{-1}S_n) = p$ under the unrestricted model.

$$\begin{aligned} \upsilon = df - df_0 &= \frac{1}{2}p(p+1) - [p(q+1) - \frac{1}{2}q(q-1)] \\ &= \frac{1}{2}[(p-q)^2 - p - q] \end{aligned}$$

Then, we reject H_0 if

$$n\ln\left(\frac{det(\hat{\Sigma})}{det(S_n)}\right) > \chi^2_{\upsilon} \tag{1.4}$$

The likelihood ratio test is not appropriate when $q \geq \frac{1}{2}(2p+1-\sqrt{8p+1})$ because this means $\upsilon \leq 0$. For the example in this section, the likelihood ratio test cannot be tested because degree of freedom $= 0$ with $p = 6$ and $q = 3$. Therefore, if $\upsilon \leq 0$ it is recommended to use asymptotic χ^2 distribution in the following test form:

$$n\{log \mid \hat{\beta}\hat{\beta}' + \hat{\tau}^2 \mid -log \mid R \mid +tr[R(\hat{\beta}\hat{\beta}' + \hat{\tau}^2)^{-1}] - p\} \tag{1.5}$$

with degrees of freedom is

$$\upsilon = \frac{1}{2}[(p-q)^2 + p - q].$$

It is clear that the degrees of freedom are always positive whenever $p > q$. The degrees of freedom is different because it is not assumed that $\hat{S}igma$ is the maximum likelihood estimate. This test can be applied for all types of factor solutions. The form in ((1.5)) was derived based on the sampling variation of the observed correlation matrix. Because the sample size influences the variability, large samples are recommended when applying factor analysis (Harman, 1976).

If the data example in this section with $q = 2$ instead of $q = 3$ is used and the model is tested, the test statistic is

$$\begin{aligned}
n \ln\left(\frac{det(\hat{\Sigma})}{det(S_n)}\right) &= n \ln\left(\frac{det(\hat{\beta}\hat{\beta}' + \hat{\tau}^2)}{det(S_n)}\right) \\
&= n \ln \frac{det(\hat{V}^{-1/2})det(\hat{\beta}\hat{\beta}' + \hat{\tau}^2)det(\hat{V}^{-1/2})}{det(\hat{V}^{-1/2})det(S_n)det(\hat{V}^{-1/2})} \\
&= n \ln \frac{det(\hat{\beta}_z\hat{\beta}_z' + \hat{\tau}^2)}{det(R)} \\
&= n \ln det(\hat{\beta}_z\hat{\beta}_z' + \hat{\tau}^2) - n \ln det(R) \\
&= -84.36 + 113.73 \\
&= 29.36.
\end{aligned}$$

with $\upsilon = \frac{1}{2}[(6-2)^2 - 6 - 2] = 4$, leading to a very small p-value. So, we should reject the H_0 and determine that the two factor model is not appropriate.

1.3.4 Minres Method

"Minres" is a contraction of minimum residuals (Harman, 1976). The distributions of the data are not important when using this method so this method can be used when the distributions of the data are unknown. It allows for the production of a small number of factors with acceptable residuals by considering the minimization of the off-diagonal residual correlation matrix.

In the factor analysis model, specific factor, $\epsilon_i, i = 1, \ldots, n$, n is the number of observations. It can be replaced by $U_i S_i$ where U_i has mean of 0 and its variance is uniqueness, and S_i has mean of 0 and unit variance. Then, the classical factor model with Z_i which is the standardization of Y_i is

$$Z_i = \beta_z X_i + U_i S_i, i = 1, \ldots, n.$$

The only parameters to be estimated are the factor loadings β. Assuming that factors are orthogonal, the estimated correlation matrix, $\hat{R}$, is

$$\hat{R} = \beta\beta'$$

and its communalities are in the principal diagonal.

The goal is to obtain a minimum value of the difference between the observed correlation matrix, R, and the reproduced correlation matrix, $\hat{R}$.

By fitting R by $(\hat{R}+U_i^2)$, or $(R-I)$ by $(\hat{R}-H^2)$ where $H^2 = I - U_i^2 = diag(\beta\beta')$, a least-squares fit can be obtained. In the former case, the principal component method can be used to minimize the residuals of the matrix. In the latter case, the minres method is utilized to minimize the off-diagonal residuals, that is,

$$min_\beta[\{R - I\} - \{\beta\beta' - diag(\beta\beta')\}]$$

and also can be shown as:

$$min[f(\beta)] = min[\sum_{k=j+1}^{p}\sum_{j=1}^{p-1}(r_{jk} - \sum_{l=1}^{q}\beta_{jl}\beta_{kl})^2]. \tag{1.6}$$

As seen in the objective function in (1.6), the off-diagonal residuals depend on p, so are not the best criterion. Instead, the root-mean-square (rms) deviation is used:

$$rms = \sqrt{2f(\beta)/p(p-1)}.$$

However, in some cases, the communality is greater than one. Therefore, an additional condition is required:

$$h_j^2 = \sum_{l=1}^{q}\beta_{jl}^2 \leq 1, j = 1, \ldots, p.$$

This forces the range of the communalities to be between 0 and 1.

1.3.5 Canonical Factor Analysis

In this model, a factor is searched from the observed data matrix by maximizing precision. Then, a second factor that is orthogonal to the first factor and has a maximal relationship with the observed data is searched. This process is repeated until a specified number of factors is obtained (Harman, 1976).

Assuming that the factors are orthogonal, the determinant equation $\mid \beta\beta' - \nu R \mid = 0$ leads to the roots, ν, which are the squared canonical correlations between the linear function of the observed variables and the linear function of the factors (Harman, 1976).

Because $R - U_i^2$ is an approximation of $\beta\beta'$, the determinant equation becomes

$$[(R - U_i^2) - \nu R]b = 0 \tag{1.7}$$

where b is a weights vector for the linear function of the variables in the zs. The equation in (1.7) is equivalent to

$$[U_i^{-1}(R - U_i^2)U_i^{-1} - \lambda I]m = 0 \tag{1.8}$$

or

$$[U_i^{-1}RU_i^{-1} - (\lambda + 1)I]m = 0 \tag{1.9}$$

where

$$\lambda = \frac{\mu}{1 - \mu}$$

and

$$m = U_i b.$$

m is the eigenvector for the largest eigenvalue$(\lambda_1 + 1)$ of $U_i^{-1}RU_i^{-1}$. According to Harris (1962), the canonical correlation equals the roots of $U_i^{-1}RU_i^{-1}$ where the roots of $U_i^{-1}RU_i^{-1}$ are greater than one.

The factor loading β can be written as $\beta = U_i Q \Lambda^{1/2}$. Q is the matrix of eigenvectors of unit-length related to the q largest eigenvalues of (1.8). Further, Λ is the diagonal matrix of these eigenvalues. It is important to note that in canonical factor analysis factors are invariant on rescaling.

The factors x can be obtained by $x = (\beta'\beta)^{-1}\beta' c$ where $c = \beta x$ is the common part. However, c cannot be obtained directly, so the factors are roughly estimated from the observed data.

1.3.6 Multiple-Group Method

According to Harman (1976), the multiple-group method selects linearly independent groups and obtains a number of common factors in one operation. Usually the

common factors chosen in one operation are oblique to each other. Therefore, two matrices are produced from the multiple-group method. One is a factor pattern that is the estimates the factor loadings, and the other is the estimate of the correlations between the variables and the factors. As a result, a matrix of reproduced correlations is obtained and thus the residual matrix is estimated. This method is repeated with the residual matrix if the residual matrix and the null matrix are not close enough. The common factors produced at each step are oblique to each other, although the factors obtained at one step are orthogonal to the factors obtained at other steps.

Oblique solution

Reference axes that go through the cores of each group of variables represent the factors. The factors are oblique with each other because the groups of variables typically not orthogonal with one another. First, p variables are grouped into q groups $G_l, l = 1, \ldots, q$.

In this method, a composite variable T_l is obtained by adding the variables in a group G_l:

$$T_l = \sum(z_k; k \in G_l), l = 1, \ldots, q.$$

$T_l's$ are oblique factors and usually do not have unit variances.

To facilitate the computation of variances and covariances among the factors, some correlations are summed. The first sum is the sum of the correlations of variable z_j so that:

$$w_{jl} = \sum(r_{jk}; k \in G_l), j = 1, \ldots, p; l = 1, \ldots, q$$

where all the variables in G_l are summed. Another useful sum is that of the correla-

tions among all variables in G_l :

$$W_{lm} = \sum(w_{jm}; j \in G_l) \tag{1.10}$$

$$= \sum(r_{jk}; j \in G_l, k \in G_m), l, m = 1, \ldots, q. \tag{1.11}$$

The variance of T_p is

$$s^2_{T_l} = \sum(r_{jk} : j, k \in G_l) \tag{1.12}$$

$$= W_{ll}. \tag{1.13}$$

The correlation between T_l and T_m is

$$r_{T_l T_m} = \frac{\sum_{i=1}^{n} T_{li} T_{mi}}{n s_{T_l} s_{T_m}} \tag{1.14}$$

From the equation in (1.12)) we know that $W_{lm} = \sum_{i=1}^{n} T_{li} T_{mi}/n$. By combining (1.14) with (1.12), the correlation becomes

$$r_{T_l T_m} = \frac{W_{lm}}{\sqrt{W_{ll} W_{mm}}}.$$

Now, the correlations of the variables with the factors can be computed in the below equation:

$$r_{z_j T_l} = \frac{w_{jl}}{\sqrt{W_{ll}}}.$$

Finally, the factor pattern P can be computed:

$$P = S\Phi^{-1} \tag{1.15}$$

where S is the known $p \times q$ factor structure, $s_{jl} = r_{z_j x_l}$, and $\Phi = XX'$.

For the oblique solution to be evaluated, the reproduced correlations and the

residuals need to be computed. The reproduced correlations are computed as:

$$\hat{R} = PS'$$

and the residuals with q factors excluded are computed as:

$$R_q = R - \hat{R}.$$

Orthogonal solution

The orthogonal solution can be obtained from the oblique solution by transforming the pattern values to the orthogonal factor loadings (Harman, 1976).

The first axes are the same on the transformed (orthogonal) system and the oblique system. The second axis of the transformed system is orthogonal to the first axis but on the same plane of the first two oblique axes, and so on. For example, let p be a point on the two factor axes. The coordinates of p are (b_1, b_2) in the oblique system and (a_1, a_2) in the orthogonal system. The vector p from the origin can be expressed in the orthogonal system as:

$$\tilde{z} = a_1 F_1 + a_2 F_2$$

and expressed in the oblique system as:

$$\tilde{z} = b_1 T_1 + b_2 T_2$$

where F_1 and F_2 are orthogonal factors, and T_1 T_2 are oblique factors.

The relationships between the two coordinate systems are

$$\begin{aligned} a_1 &= b_1 + b_2 cos\theta_{12} \\ a_2 &= b_2 + b_2 cos\theta_{12} \end{aligned}$$

where θ_{12} is the angle from the axis T_1 to the axix T_2, as is the correlation $r_{T_1T_2} = r$. Then, the relationships in the matrix form are

$$(a_{j1} \;\; a_{j2}) = (b_{j1} \;\; b_{j2}) \begin{bmatrix} 1 & 0 \\ r & \sqrt{1-r^2} \end{bmatrix}. \tag{1.16}$$

The generalization of the equation in (1.16) for p variables and q factors is

$$A = PT' \tag{1.17}$$

where A is $p \times q$ matrix (a_{jl}) and P is $p \times q$ matrix (b_{jl}), and T, a $q \times q$ matrix, is obtained by the square root operation on the matrix Φ of factor correlations. The orthogonal coordinates A in the form (1.17) can be represented by the oblique structure values. If the equation in (1.15) is used in the equation in (1.17), A will become

$$A = S\Phi^{-1}T'. \tag{1.18}$$

Since $\Phi = T'T$, the equation in (1.18) becomes

$$A = ST^{-1}$$

which is the transformation needed from the oblique system to the orthogonal system.

1.4 Review of Factor Rotation

There is an equivalence class of factor loadings that yields the same $\beta\beta$. That is, the factor loadings have the same covariance matrices and so the same $\beta\beta$ and τ^2. This is because an orthogonal transformation can produce the same covariance matrix.

This orthogonal transformation of the factor loadings is referred to as factor rotation. Suppose $\hat{\beta}$ is the $p \times q$ matrix of estimated factor loadings. Then

$$\hat{\beta}^* = \hat{\beta}T$$

where $TT' = T'T = I$, is a $p \times q$ matrix of rotated loadings, and

$$\hat{\beta}\hat{\beta}' + \tau^2 = \hat{\beta}TT'\hat{\beta}' + \tau^2 = \hat{\beta}^*\hat{\beta}^{*'} + \tau^2.$$

This equivalence means that the residual matrices, $S_n - \hat{\beta}\hat{\beta}' - \tau^2$ and $S_n - \hat{\beta}^*\hat{\beta}^{*'} - \tau^2$, are the same, the communalities, $\hat{h}_i^2$ and $\hat{h}_i^{2*}$, are the same, and likelihoods are also the same.

Table Table 1.3 shows two sets of loadings, an ML solution and the solution rotated (by the varimax method).

Table 1.3: The original factor loadings and the rotated factor loadings by the maximum likelihood method are shown using the same data

Var	$\tilde{\beta}_1$	$\tilde{\beta}_2$	$\tilde{\beta}_3$	$\tilde{\beta}_1^*$	$\tilde{\beta}_2^*$	$\tilde{\beta}_3^*$	$\tilde{h}_i^2$
1	0.722	-0.389	0.487	0.169	(0.924)	0.165	0.91
2	0.786	-0.433	0.434	0.258	(0.952)	0.150	1.00
3	0.670	0.725	0.141	0.176	0.086	(0.978)	0.99
4	0.721	0.571	0.204	0.191	0.237	(0.891)	0.89
5	0.796	-0.117	-0.532	(0.925)	0.186	0.198	0.93
6	0.837	-0.169	-0.516	(0.949)	0.246	0.181	0.99
Cum. proportion	0.574	0.779	0.952	0.320	0.640	0.952	

It is not clear which variables are closely correlated from the factor loadings on the left side in the above table, but it is clear from the rotated factor loadings on the right side. Variables 1 and 2 depend on factor 2, variables 3 and 4 depend on factor 3, and variables 5 and 6 depend on factor 1 from the parenthesized values on the right side of Table Table 1.3.

There are numerous rotation methods in factor analysis; three standard orthogonal rotation methods are described by Harman (1976). The quartimax method emphasizes simplifying variables, whearas the varimax method emphasizes simplifying factors. The orthomax method represents a compromise between the two methods. The varimax method provides rotations that easily identify each variable with a single factor, which is the standard recommendation.

1.4.1 Quartimax Criterion

The original factor matrix should be transformed so that, for each variable, large factor loadings are increased and small factor loadings are decreased (Harman, 1976). Because algebraic sign does not matter, The use of squares of factor loadings is suggested. To achieve simple factors, quartimax maximizes the variance of the squared factor loadings for each variable on the rotation.

$$\boldsymbol{B} = \boldsymbol{AT}$$

where

$$\begin{aligned} \boldsymbol{A} &= (a_{jl}), \quad initial\ factor\ loadings\ matrix \\ \boldsymbol{B} &= (b_{jl}), \quad final\ factor\ loadings\ matrix \\ \boldsymbol{T} &= (t_{ml}), \quad orthogonal\ transformation\ matrix. \end{aligned}$$

Then, the equation in (1.19) is maximized:

$$s^2_{b^2} = \frac{1}{pq}\sum_{j=1}^{p}\sum_{l=1}^{q}(b^2_{jl} - \bar{b}^2)^2 \tag{1.19}$$

$$= \frac{1}{pq}\sum_{j=1}^{p}\sum_{l=1}^{q} b^4_{jl} - (\bar{b}^2)^2. \tag{1.20}$$

Maximizing the equation in (1.19) is the same as maximizing the sum of fourth powers of factor loadings because $\bar{b}^2$ is a constant term under orthogonal rotation. Thus,

$$Q = \sum_{j=1}^{p} \sum_{l=1}^{q} b_{jl}^4 \tag{1.21}$$

is maximized.

The procedure of Neuhaus and Wrigley (1954) was followed in developing the theory in this section. The relationship between the original coordinates as and the new coordinates bs in the plane of factors l and m through an angle ϕ is

$$b_{jl} = a_{jl}cos\phi + a_{jm}sin\phi \tag{1.22}$$

$$b_{jm} = -a_{jl}sin\phi + a_{jm}cos\phi. \tag{1.23}$$

where bs represent all intermediate values and the final value resulting from the rotation of as. This procedure works for two factors at a time and reduces the problem to a sequence of one-dimensional optimizations.

The loadings of the two rotated factors are determined so that the fourth powers of the new loadings are summed:

$$Q_{lm}(\phi) = \sum_{j=1}^{p} (b_{jl}^4 + b_{jm}^4). \tag{1.24}$$

In the equation in (1.24), the rotation angle ϕ can be obtained by maximizing $Q_{lm}(\phi)$ function. Then a final transformed factor matrix B is computed by the product of the transformation matrices T_{lm} of all possible combinations of factor pairs (order does not matter):

$$\boldsymbol{B} = \boldsymbol{A}\boldsymbol{T}_{12}\boldsymbol{T}_{13}\ldots\boldsymbol{T}_{(q-1),q}. \tag{1.25}$$

The procedure in (1.25) is called one cycle and this cycle should be repeated until Q is converged.

The angle ϕ can be computed by differentiating Q in (1.24) after substituting the equation in ((1.22)) into the equation in (1.24). The result is

$$tan4\phi = \frac{2\sum_{j=1}^{p} 2a_{jl}a_{jm}(a_{jl}^2 - a_{jm}^2)}{\sum_{j=1}^{p}[(a_{jl}^2 - a_{jm}^2)^2 - (2a_{jl}a_{jm})^2]} = \frac{\nu}{\delta}. \tag{1.26}$$

To make sure Q has the maximum, use the inequality that the second derivative should be less than zero.

When ϕ is determined, the transformation matrix becomes

$$T_{lm} = \begin{pmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{pmatrix}.$$

The procedure in (1.25) should be repeated until Q converges.

1.4.2 Varimax Criterion

Whereas the quartimax method emphasizes a simplified description of each variable, the varimax method emphasizes the simplification of the factors (Harman, 1976).

Proposed by Kaiser (1958), the varimax method is a modification of the quartimax method. His simplicity of a factor l is expressed as the variance s_l^2 of its squared loadings:

$$s_l^2 = \frac{1}{p}\sum_{j=1}^{p}(b_{jl}^2)^2 - \frac{1}{p^2}(\sum_{j=1}^{p} b_{jl}^2)^2, l = 1, \ldots, q. \tag{1.27}$$

which leads factor loadings toward unity and zero. The criterion for a complete factor

matrix is expressed as the maximization of the sum of variance s_l^2 in (1.27):

$$s^2 = \sum_{l=1}^{q} s_l^2 = \frac{1}{p}\sum_{l=1}^{q}\sum_{j=1}^{p} b_{jl}^4 - \frac{1}{p^2}\sum_{l=1}^{q}(\sum_{j=1}^{p} b_{jl}^2)^2 \tag{1.28}$$

which is called the "raw" varimax criterion. However, the contributions of the factors are not the same. To make the contributions equal, weights are attached to the variables so that they adjust the vectors that describe the variables to unit length in the factor space. After the rotations, the vectors are brought back to their original length.

The improved varimax criterion that replaces the "raw" varimax criterion (1.28) is

$$V = p\sum_{l=1}^{q}\sum_{j=1}^{p}(b_{jl}/h_j)^4 - \sum_{l=1}^{q}(\sum_{j=1}^{p} b_{jl}^2/h_j^2)^2. \tag{1.29}$$

As seen in the equation in (1.29), the factors are of unit length because $h_j^2 = b_{j1}^2 + \ldots + b_{jq}^2$.

The algorithm is the same as in the quartimax criterion but maximizing V in (1.29) instead of maximizing Q in (1.24).

Let's introduce some transformation form. Suppose the "normalized" factor loadings of a variable z_j for a pair of factors l and m be $x_j = a_{jl}/h_j$ and $y_j = a_{jm}/h_j$. If the rotated loadings are X_j and Y_j, then the transformation form becomes

$$(X_j \;\; Y_j) = (x_j \;\; y_j)\begin{pmatrix} cos\phi & -sin\phi \\ sin\phi & cos\phi \end{pmatrix}$$

where ϕ is the angle of rotation. To compute ϕ, several notations are defined as

$$\begin{aligned} \mu_j &= x_j^2 - y_j^2 \qquad & C &= \sum(\mu_j^2 - \nu_j^2) \\ \nu_j &= 2x_j y_j \qquad & D &= 2\sum \mu_j \nu_j \\ A &= \sum \mu_j \qquad & E &= D - 2AB/p \\ B &= \sum \nu_j \qquad & F &= C - (A^2 - B^2)/p \end{aligned}$$

where $j = 1, \ldots, p$.

The angle ϕ can be obtained from

$$tan4\phi = E/F.$$

1.4.3 Orthomax Criteria

The orthomax criteria are built from a weighted composite of two types of orthogonal rotations, quartimax rotation and varimax rotation (**??**) that is,

$$cQ + dV$$

is maximized where Q is taken from (1.21) and V is taken from (1.28) multiplied by p. Then, the orthomax criteria are maximizing

$$\sum_{l=1}^{q}(\sum_{j=1}^{p} b_{jl}^4 - \frac{\gamma}{p}(\sum_{j=1}^{p} b_{jl}^2)^2)$$

where $\gamma = d/(c+d)$. Therefore, if $\gamma = 0$, then this criteria become the quartimax criterion and if $\gamma = 1$, then this criteria become the varimax criterion.

The best results may be obtained when $\gamma = q/2$ (Saunders, 1962) and is called "equamax" criterion.

1.5 Factor Scores

The observed variables in the factor model can be estimated from the factor loadings and uniqueness as well as the factor scores, which are the values estimated from the common factors. Usually, factor loadings and uniqueness are the interest in a factor model. However, factor scores are often desired when the intention is to relate the factors to other variables. In addition, future research does not require a factor analysis; it only requires the scoring procedure for the factors. There are two important elements in approaching factor score estimation. The first is that factor loadings and uniquenesses are treated as if they were true values, not estimated values. The second is that the original data are linearly transformed in computing factor scores when rotated factor loadings are involved (Johnson and Wichern, 2007).

$$
\begin{aligned}
E(X) &= 0_{(q\times 1)}, Cov(X) = E[XX'] = I_{(q\times q)} \\
E(\epsilon) &= 0_{(p\times 1)}, Cov(\epsilon) = E[\epsilon\epsilon'] = \tau^2_{(p\times p)} = \begin{bmatrix} \tau_1^2 & 0 & \dots & 0 \\ 0 & \tau_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \tau_p^2 \end{bmatrix}
\end{aligned}
$$

Thus,

$$Y - \mu = \beta X + \epsilon \sim N_p(0, \beta\beta' + \tau^2)$$

and

$$(Y - \mu, X) \sim N_{p+q}(0, \left[\begin{array}{c|c} \beta\beta' + \tau^2 & \beta \\ \hline \beta' & I \end{array}\right])$$

Also,

$$X|Y \sim N(\beta'(\beta\beta' + \tau^2)^{-1}(y - \mu), I - \beta'(\beta\beta' + \tau^2)^{-1}\beta)$$

Therefore, the i-th vector of the estimated factor score is

$$\hat{x}_i = \hat{\beta}'\hat{\Sigma}^{-1}(y_i - \bar{y}), \quad i = 1, 2, \ldots, n$$

where $\hat{\beta}$ and $\hat{\tau}^2$ are the maximum likelihood estimates and considered as the true values, and $\hat{\Sigma} = \hat{\beta}\hat{\beta}' + \hat{\tau}^2$.

Practically, for the data whose dimension is not very high, $\hat{\Sigma}$ is replaced by S, sample covariance matrix, to reduce the effects of the number of factors not properly chosen. Then,

$$\hat{x}_i = \hat{\beta}'\hat{S}^{-1}(y_i - \bar{y}), \quad i = 1, 2, \ldots, n$$

If the factor loadings are replaced by the rotated loadings $\hat{X}^* = \hat{X}T$, the factor scores $\hat{x}_i$ are also replaced by $\hat{x}_i^*$ where

$$\hat{x}_i^* = T'\hat{x}_i, \quad i = 1, 2, \ldots, n.$$

Chapter 2

Review on Lasso and Adaptive Lasso

2.1 Lasso

Two well-known methods for supplementing the ordinary least squares (OLS) method are ridge regression and subset selection. However, both have weaknesses. Ridge regression offers stable models because coefficients are shrunk by a continuous process, but interpretation is difficult because predictors are not reduced. Subset selection allows for easy interpretation but may yield a highly variable model because it selects predictors in a discrete manner. Therefore, a new method, the least absolute shrinkage and selection operator (lasso) method was proposed by Tibshirani (1996) to complement ridge regression and subset selection. It solves the OLS problem subject to a constraint on the sum of the absolute value of the coefficients. As a result of this constraint, some shrunken coefficients or zero coefficients can be obtained. The lasso method increases prediction accuracy by reducing variance while sacrificing a small bias. It also allows for easy interpretation by excluding variables via zero coefficients.

2.1.1 Lasso Regression

If y_i is the response and x_i is the predictor variable, $i = 1, \ldots, n$, and the conditional distribution of y_i given x_i is independent, then the lasso estimate $\hat{\beta}$ is

$$\hat{\beta} = argmin[\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}\beta_j x_{ij})^2] \ \ subject \ \ to \ \ \sum_{j=1}^{p}|\beta_j| \leq t \tag{2.1}$$

where $\hat{\beta} = (\hat{\beta}_1, \ldots, \hat{\beta}_p)'$ and $t \geq 0$ is a tuning parameter. In the definition of the lasso in (2.1), the constant term is omitted because $\bar{y} = 0$ is set. It is also assumed that x_{ij} are standardized with mean 0 and variance 1. The parameter t determines the shrinkage level. If $\hat{\beta}_j^0$ is the least squares estimate, then shrinkage of the estimates and zero estimates will occur when $t < \sum|\hat{\beta}_j^0|$.

The Lagrangian form of the lasso estimate is

$$\hat{\beta} = argmin[\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}|\beta_j|] \tag{2.2}$$

Two equations in (2.1) and (2.2) are equivalent, that is, there exists a $\lambda \geq 0$ if and only if there exists a $t \geq 0$ such that the solutions are the same (Osborne et al., 2000). For some $\lambda \geq 0$ in the orthonormal design case, $\hat{\beta}_j = \hat{\beta}_j^0$ if $|\hat{\beta}_j^0| > \lambda$ and shrinkage occurs otherwise.

When $\mid \beta_j \mid^1$ is considered as the log-prior density for β, the Lagrangian form in (2.2) can be considered as an independent double exponential or Laplace distribution, with density $(1/2)exp(- \mid \beta \mid /\tau)$ and $\tau = 1/\lambda$ (Hastie et al., 2009). Therefore, the lasso is, in this sense, a Bayes estimate.

The plot in figure Figure 2.1 is shown to compare the OLS coefficients to the Lasso coefficients in the orthogonal design case. The 45° dotted line is the unrestricted coefficient estimate line when Y axis is the OLS estimate and given as a reference,

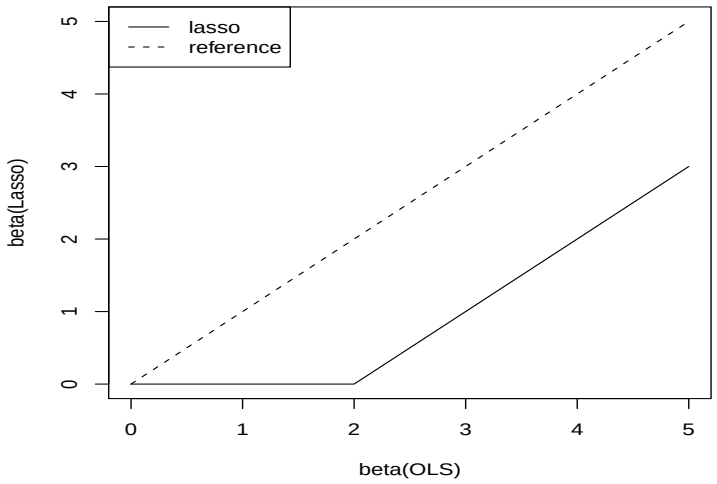

Figure 2.1: −: coefficient shrinkage in the orthogonal design case; $\cdots$: 45° -line for reference

whereas the line is the restricted coefficient estimate line when Y axis is the Lasso estimate. This figure illustrates that the lasso regression shrinks the coefficients by a constant amount and truncates at 0. That is, the lasso produces not only shrinkage but also exactly 0 for the estimators when the OLS estimate is small.

As an example, the "diabetes" data in R package are used with least squares estimate model:

$$Y_i = \beta_1 x_{i,1} + \ldots + \beta_{10} x_{i,10}$$

where $i = 1, \ldots, 442$. Table Table 2.1 illustrates the values of $\boldsymbol{\beta}$ for different values of λ. As λ increases, $\sum \mid \beta_j \mid$ decreases and some β_j become exact 0s.

2.1.2 Least Angle Regression for Lasso

The Lasso solution is found by using the sequential inequality constraints and searching for a solution with which the Kuhn-Tucker conditions are satisfied (Lawson and

Hansen, 1974). However, the Lasso can be solved with much less computation costs if Least Angle Regression (LARS) is used.

Model selection algorithms, such as forward selection, backward elimination, and all subsets regression are the algorithms of selecting a linear model for estimating a response y based on some predictors $x_1, x_2, \ldots, x_p$. However, these algorithms go through many steps before reaching a final model. So, LARS, which is related to the traditional forward selection methods, was developed to reach a final model with much simpler steps by accelerating the computations (Efron et al., 2003). Because the LARS can be used for computing Lasso estimates with small modification, it is a useful method for Lasso computation.

The LARS procedure is as follows:

1. Start with zero coefficients.
2. Find the most correlated predictor $(x_{j_1}, j = 1, \ldots, p)$ with the response.
3. Take the largest step in the direction of this predictor and stop when one other predictor (x_{j_2}) has the same correlation with the current residual.
4. Move a direction in the same angles between the two predictors and stop when

Table 2.1: The values of β for different values of λ in "diabetes" data of R package; As λ increases, $\sum \mid \beta_j \mid$ decreases and some β_j become zero; λ is chosen such that some β start to become zero

λ	β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8	β_9	β_{10}
0	-10.0	-239.8	519.8	324.4	-792.2	476.8	101.0	177.1	751.3	67.6
5.5	0	-227.2	526.4	315.0	-237.3	33.6	-134.6	111.4	545.5	64.6
88.8	0	-112.0	512.0	252.5	0	0	-196.1	0	452.4	12.1
452.9	0	0	434.8	79.2	0	0	0	0	374.9	0
949.4	0	0	60.1	0	0	0	0	0	0	0

a third variable (x_{j_3}) attains the most correlated set.

5. Move a direction in the same angles among three predictors and stop when a fourth variable attains the most correlated set. The process continues for a fifth variable, a sixth variable, etc.

The LARS algorithm is as efficient as that of OLS. Suppose $\hat{\mu}_0 = 0$, $\hat{\mu}_A$ is the current LARS estimate, and $\hat{c} = X'(y - \hat{\mu}_A)$ is the vector of correlations with the current residuals. Then, the algorithm is fully described as:

$$\hat{C} = max_j\{\mid \hat{c}_j \mid\} \quad and \quad A = \{j :\mid \hat{c}_j \mid = \hat{C}\}$$

If

$$s_j = sign\{\hat{c}_j\} \quad for \quad j \in A,$$

then,

$$\mu_A = X_A[1'_A(X'_AX_A)^{-1}1_A]^{-1/2}(X'_AX_A)^{-1}1_A$$

where 1_A is a vector of $1's$ with the length of $\mid A \mid$, and $X_A = (\ldots, s_jx_j \ldots)_{j\in A}$. Also the inner product vector

$$a \equiv X'\mu_A.$$

So the update $\hat{\mu}_{A+}$ of the current LARS estimate $\hat{\mu}_A$ is

$$\hat{\mu}_{A+} = \hat{\mu}_A + \hat{\gamma}\mu_A \tag{2.3}$$

where

$$\hat{\gamma} = min^+_{j\in A^c}\{\frac{\hat{C} - \hat{c}}{(1'_A(X'_AX_A)^{-1}1_A)^{-1/2} - a_j}, \frac{\hat{C} + \hat{c}}{(1'_A(X'_AX_A)^{-1}1_A)^{-1/2} + a_j}\}; \tag{2.4}$$

"min^+" means that, over positive components for each j, only the minimum is chosen. $\hat{\gamma}$ in (2.4) is the smallest positive value of γ that leads new index $\hat{j}$ to become the active set where $\hat{j}$ is the minimizing index in (2.4). The new active set $A_+ = A \cup \{\hat{j}\}$ and the new maximum absolute correlation is $\hat{C}_+ = \hat{C} - \hat{\gamma}(1'_A(X'_A X_A)^{-1}1_A)^{-1/2}$.

Now, suppose if $\tilde{\gamma} < \hat{\gamma}$, stop the LARS step at $\gamma = \tilde{\gamma}$. Then, LARS algorithm is applied into Lasso, that is, the following equation is used instead of (2.3):

$$\hat{\mu}_{A_+} = \hat{\mu}_A + \tilde{\gamma}\mu_A \quad and \quad A_+ = A - \tilde{\tilde{j}}.$$

2.2 Adaptive Lasso

The Lasso method can simultaneously estimate and select predictive variables and increase prediction accuracy from continuous shrinkage due to the bias-variance trade off. It can also select variables automatically because of the nondifferentiality of the ℓ_1 penalty at the origin. However, the Lasso estimates are biased for the large coefficients. Therefore, in terms of estimation risk, it could be suboptimal (Fan and Li, 2001). It is also shown that the Lasso variable selection might be inconsistent and lacks the oracle properties (Fan and Li, 2001) and (Meinshausen and Bühlmann, 2004). The oracle properties are those that the coefficient estimator uses to identify the right subset model with the optimal estimation rate. That is, $\sqrt{n}$ multiplied by the difference between the coefficient estimator and the true coefficient becomes normally distributed with mean 0 and covariance matrix knowing the true subset model (Zou, 2006). It is thought that a good procedure should have oracle properties (Fan and Li, 2001) and (Fan and Peng, 2004).

Consequently, a new penalty method, called the adaptive lasso (ALasso) was introduced by Zou (2006). It is different from the Lasso because it places data dependent and differing weights on coefficients across the variables, whereas the Lasso places

the same penalty on the coefficients. Suppose that $\hat{\beta}$ is a root n-consistent estimator to β^* where β^* is the estimate knowing the true subset model where the coefficients are nonzeros. As in the Lasso, if y_i is the response and x_i is the predictor variable, $i = 1, \ldots, n$, and the conditional distribution of y_i given x_i is independent, then the ALasso estimates $\hat{\beta}^{*(n)}$ is

$$\hat{\beta}^{*(n)} = argmin_{\beta}[\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda_n \sum_{j=1}^{p}\hat{w}_j|\beta_j|], \tag{2.5}$$

where $\hat{w}_j$ is a weights vector and defined by $1/\mid \hat{\beta}_j \mid^{\gamma}$, $\gamma > 0$ must be selected, and λ_n varies with n.

The form in (2.5) is a convex function for all positive γ as the form in the Lasso; it does not have a multiple local minimum problem. The adaptive lasso estimates can be computed using the current lasso algorithm because the ALasso uses a Lasso type penalization method. The computational cost is the same as the one from the OLS fit, so the ALasso method is very attractive.

One of important properties of ALasso is the oracle properties, that is, consistency in variable selection and asymptotic normality are satisfied in the adaptive lasso estimates if $\lambda_n/\sqrt{n} \to 0$ and $\lambda_n n^{(\gamma-1)/2} \to \infty$ (Zou, 2006).

$\hat{w}$ depends on data and is critically important in the oracle properties of ALasso. $\hat{w}$ for zero-coefficient predictors grows to infinity and $\hat{w}$ for nonzero coefficient predictors converges to a finite constant as the sample size n increases. Therefore, large coefficient and small threshold unbiased estimates can simultaneously and asymptotically be estimated. A smoothly clipped absolute deviation (SCAD) has the same rationale as this (Zou, 2006).

Chapter 3

Proposed Approach and Main Results

3.1 Notation for the model

Suppose we have n independent and identically distributed (i.i.d.) random vectors in $\mathbb{R}^p$: $\{Y_1, \ldots, Y_n\}$. Without loss of generality, assume the mean of Y_i is zero and its covariance is $\mathbf{\Sigma}$. The factor model is represented by

$$Y_i = \boldsymbol{\beta}^T X_i + e_i, \tag{3.1}$$

where X_i is an *unobserved* random vector of length q, $\boldsymbol{\beta}$ is a $q \times p$ matrix and e_i represents a p-dimension random error vector whose mean is zero and covariance is a diagonal matrix denoted by $\boldsymbol{\tau}^2 = \text{diag}(\tau_1^2, \ldots, \tau_p^2)$. $\boldsymbol{\tau}^2$ is called the uniqueness matrix. It is assumed that X_i has zero mean and covariance $\boldsymbol{I}_q$. As a consequence, the covariance of Y_i can be expressed as $\mathbf{\Sigma}_Y = \boldsymbol{\beta}^T\boldsymbol{\beta} + \boldsymbol{\tau}^2$. In a matrix form we write the model as

$$\mathbf{Y}_{n\times p} = \mathbf{X}_{n\times q}\boldsymbol{\beta}_{q\times p} + \boldsymbol{\epsilon}_{n\times p} \tag{3.2}$$

where the i-th rows of $\mathbf{Y}$, $\mathbf{X}$ and $\boldsymbol{\epsilon}$ are Y_i, X_i and e_i, respectively. In factor analysis, $\mathbf{X}$ is called the factor score matrix and $\boldsymbol{\beta}$ is called the factor loading matrix. For statistical inference, it is usually assumed that the hidden factors are normally distributed and hence Y_is are i.i.d. $N(0, \boldsymbol{\beta}^T\boldsymbol{\beta}+\boldsymbol{\tau}^2)$. Maximum likelihood estimation can be carried out by using the Expectation-Maximization algorithm (Rubin and Thayer, 1982).

The factor model (Dempster et al., 1977) is invariant under an orthogonal rotation, as is the maximum likelihood estimator. This property makes it possible to rotate the estimated factor loading matrix so that the rotated loading matrix exhibits an interesting structure or pattern that can help to interpret the fitted factor model. Rotation is often necessary in real applications of factor analysis when the number of factors is not small. The most common rotation technique is *varimax*, which aims to yield either large or small loadings. Often, small loadings are further truncated at some threshold (e.g., 0.1), as zero loadings greatly enhance the interpretability. We can understand the idea behind varimax rotation as follows. Suppose that the factor model can be represented by some sparse $\boldsymbol{\beta}$ matrix that makes the factor model easy to interpret. The MLE $\widehat{\boldsymbol{\beta}}$ is an estimator of $U\boldsymbol{\beta}$ with U being an unknown orthonormal matrix. Varimax aims to find U^T in order to recover the sparse $\boldsymbol{\beta}$ matrix using $U^T\widehat{\boldsymbol{\beta}}$, the varimax rotated loading matrix.

3.2 ℓ_1-Penalized Factor Analysis

In this section we define the ℓ_1 penalized factor analysis. Under the normality assumption, the log-likelihood can be written as

$$LL(\boldsymbol{\tau}^2, \boldsymbol{\beta}) = -\frac{n}{2}\log det(\boldsymbol{\tau}^2 + \boldsymbol{\beta}^T\boldsymbol{\beta}) - \frac{1}{2}\sum_{i=1}^{n} Y_i^T(\boldsymbol{\tau}^2 + \boldsymbol{\beta}^T\boldsymbol{\beta})^{-1}Y_i, \tag{3.3}$$

or equivalently

$$LL(\boldsymbol{\tau}^2, \boldsymbol{\beta}) = -\frac{n}{2}\left(\log det(\boldsymbol{\tau}^2 + \boldsymbol{\beta}^T\boldsymbol{\beta}) + tr\left((\boldsymbol{\tau}^2 + \boldsymbol{\beta}^T\boldsymbol{\beta})^{-1}\boldsymbol{\Sigma}^s\right)\right), \tag{3.4}$$

where $\boldsymbol{\Sigma}^s = \frac{1}{n}Y^TY$ is the sample covariance matrix of Y. In the above equations we have assumed the mean of Y_i is zero which is done in practice by centering the data matrix. The classical factor analysis uses the maximum likelihood estimator given by

$$\arg\min\left\{\log det(\boldsymbol{\tau}^2 + \boldsymbol{\beta}^T\boldsymbol{\beta}) + tr\left((\boldsymbol{\tau}^2 + \boldsymbol{\beta}^T\boldsymbol{\beta})^{-1}\boldsymbol{\Sigma}^s\right)\right\}. \tag{3.5}$$

Generally speaking, the objective function in (3.5) is equivalent to the Kullback-Leibler loss between $\boldsymbol{\tau}^2+\boldsymbol{\beta}^T\boldsymbol{\beta}$ and the sample covariance matrix $\boldsymbol{\Sigma}^s$. Without causing any confusion, from now on we still call $LL(\widehat{\boldsymbol{\tau}^2}, \widehat{\boldsymbol{\beta}})$ the log-likelihood.

Interpretability of the factor model becomes very important in applications when the number of factors is not too small. In the classical factor analysis, rotation techniques are often used to obtain more understandable factor loadings. Factor analysis is closely related to principal component analysis in the sense that both methods attempt to explain the variability among correlated variables by several factors/components. The ℓ_1 penalization idea has been successfully used to develop sparse principal component analysis (Zou et al., 2006). We use the sparse penalization idea to develop sparse factor analysis.

Consider the penalized log-likelihood defined by

$$PLL(\boldsymbol{\tau}^2, \boldsymbol{\beta}) = -\frac{n}{2}\log det(\boldsymbol{\tau}^2+\boldsymbol{\beta}^T\boldsymbol{\beta}) - \frac{n}{2}tr[\boldsymbol{\Sigma}^s(\boldsymbol{\tau}^2+\boldsymbol{\beta}^T\boldsymbol{\beta})^{-1}] - \frac{1}{2}\sum_{l=1}^{q}\sum_{j=1}^{p}P_\lambda(|\beta_{lj}|) \tag{3.6}$$

where $P_\lambda(\cdot)$ is a non-negative penalty function. In recent literature there has been a lot of work on the use of sparsity-inducing penalty functions in various penalized models. The reference list is too long to be listed here. Readers are referred to two

good review papers (Fan and Lv, 0101) and (Hesterberg et al., 2008). In this work we use $P_\lambda(|\beta_{lj}|) = \lambda|\beta_{lj}|$ which is the Lasso penalty (Tibshirani, 1996). The Lasso estimator, denoted by $(\widehat{\boldsymbol{\tau}^2}, \widehat{\boldsymbol{\beta}})$, is then defined as $\arg\max PLL(\boldsymbol{\tau}^2, \boldsymbol{\beta})$ or

$$(\widehat{\boldsymbol{\tau}^2}, \widehat{\boldsymbol{\beta}}) = \arg\min \log det(\boldsymbol{\tau}^2 + \boldsymbol{\beta}^T\boldsymbol{\beta}) + tr[\boldsymbol{\Sigma}^s(\boldsymbol{\tau}^2 + \boldsymbol{\beta}^T\boldsymbol{\beta})^{-1}] + \frac{\lambda}{n}\sum_{l=1}^{q}\sum_{j=1}^{p}|\beta_{lj}|. \quad (3.7)$$

Note that the ℓ_1 penalty is not invariant under orthogonal transformation. Therefore, the estimator in (3.7) is no longer rotation invariant.

It has been shown in Zou (2006) that the adaptively weighted Lasso penalty can achieve better prediction and sparsity trade off than the Lasso and the ALasso estimator enjoys the oracle properties using the language of Fan and Li (2001). In this work we also consider the ALasso estimator in which the adaptive weights are computed from the Lasso estimator. The ALasso estimator is computed by the following two-step procedure:

1. Compute the Lasso estimator in (3.7).

2. If $\hat{\beta}_{lj} = 0$ let $\hat{w}_{lj} = \infty$, otherwise $\hat{w}_{lj} = \frac{1}{|\hat{\beta}_{lj}|}$. Then compute the ALasso penalized estimator

$$\arg\min \log det(\boldsymbol{\tau}^2 + \boldsymbol{\beta}^T\boldsymbol{\beta}) + tr[\boldsymbol{\Sigma}^s(\boldsymbol{\tau}^2 + \boldsymbol{\beta}^T\boldsymbol{\beta})^{-1}] + \frac{\lambda^*}{n}\sum_{l=1}^{q}\sum_{j=1}^{p}\hat{w}_{lj}|\beta_{lj}|.$$

In principle we can also use other penalty functions such as the SCAD (Fan and Li, 2001) to derive sparse factor analysis. We choose the Lasso and ALasso penalties primarily for computational considerations. In the next section we develop a generalized expectation-maximization (GEM) algorithm for maximizing the objective function in (3.6) with a penalty function. The ℓ_1 penalty allows for much more efficient computations in the M-step than other concave penalties like the SCAD.

3.3 Algorithm

Rubin and Thayer (1982) derived an E-M algorithm for computing the MLE for the factor model. In this section we derive an E-M algorithm for computing the ℓ_1 penalized estimator. It turns out that the penalized estimator can be computed by iterative Lasso-penalized least squares.

For convenience we define some notation. We use $M[i,]$ to denote the i-th row vector of a matrix $\mathbf{M}$. Likewise, $M[,j]$ represents the j-th column vector of $\mathbf{M}$. The i,j entry of $\mathbf{M}$ is M_{ij}.

Finding the "missing data" is a key component in the derivation of an E-M algorithm. From the model (3.2) we naturally take $\mathbf{X}$ as the missing data. By $X_i \sim N(0, \boldsymbol{I}_q)$ we write down the joint likelihood of $(\mathbf{Y}, \mathbf{X})$ as

$$\begin{aligned} L_{y,x}(\tau^2, \beta) &= [2\pi \prod_{j=1}^{p} \tau_j^2]^{-n/2} \exp[-\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{p} \frac{(Y_{ij} - X[i,]\beta[,j])^2}{\tau_j^2}] \\ &\times [2\pi det\mathbf{I}]^{-n/2} \exp[-\frac{1}{2} \sum_{i=1}^{n} X[i,]X[i,]^T] \end{aligned}$$

EM algorithms iterate between the E-step and the M-step. Let $(\boldsymbol{\beta}_{(k)}, \boldsymbol{\tau}^2_{(k)})$ be the estimates of step k. At the E-step, we need compute the conditional expectation of the log-likelihood given $\mathbf{Y}$ and $(\boldsymbol{\beta}_{(k)}, \boldsymbol{\tau}^2_{(k)})$. Let $ELL_{(k)}$ be the conditional expectation

of the log-likelihood. We have

$$\begin{aligned} ELL_{(k)}(\boldsymbol{\beta},\boldsymbol{\tau}^2) &= E(\log P(\mathbf{X},\mathbf{Y}|\boldsymbol{\beta},\boldsymbol{\tau}^2)|\mathbf{Y},\boldsymbol{\beta}_{(k)},\boldsymbol{\tau}^2_{(k)}) \\ &= -\frac{n}{2}\sum_{j=1}^{p}\log\tau_j^2 - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{p}\frac{1}{\tau_j^2}(Y_{ij}^2 - 2Y_{ij}E(X[i,]|\mathbf{Y},\boldsymbol{\beta}_{(k)},\boldsymbol{\tau}^2_{(k)})\beta[,j] \\ &\quad +\beta[,j]^T E(X[i,]^T X[i,]|\mathbf{Y},\boldsymbol{\beta}_{(k)},\boldsymbol{\tau}^2_{(k)})\beta[,j]) \\ &\quad -\frac{1}{2}\sum_{i=1}^{n}E(X[i,]X[i,]^T|\mathbf{Y},\boldsymbol{\beta}_{(k)},\boldsymbol{\tau}^2_{(k)}) + constant \end{aligned}$$

Since

$$\mathbf{X}|\mathbf{Y},\boldsymbol{\beta},\boldsymbol{\tau}^2 \sim N(\mathbf{Y}(\boldsymbol{\tau}^2+\boldsymbol{\beta}^T\boldsymbol{\beta})^{-1}\boldsymbol{\beta}^T, \mathbf{I}-\boldsymbol{\beta}(\boldsymbol{\tau}^2+\boldsymbol{\beta}^T\boldsymbol{\beta})^{-1}\boldsymbol{\beta}^T)$$

we can write

$$\begin{aligned} E(X[i,]|\mathbf{Y},\boldsymbol{\beta}_{(k)},\boldsymbol{\tau}^2_{(k)}) &= \boldsymbol{\delta}^T Y[i,]^T \\ Var(X[i,]|\mathbf{Y},\boldsymbol{\beta}_{(k)},\boldsymbol{\tau}^2_{(k)}) &= \boldsymbol{\Delta} \\ E(X[i,]^T X[i,]|\mathbf{Y},\boldsymbol{\beta}_{(k)},\boldsymbol{\tau}^2_{(k)}) &= \boldsymbol{\Delta}+\boldsymbol{\delta}^T Y[i,]^T Y[i,]\boldsymbol{\delta} \end{aligned}$$

where

$$\begin{aligned} \boldsymbol{\delta} &= (\boldsymbol{\tau}^2_{(k)}+\boldsymbol{\beta}^T_{(k)}\boldsymbol{\beta}_{(k)})^{-1}\boldsymbol{\beta}^T_{(k)}, \\ \boldsymbol{\Delta} &= \mathbf{I}-\boldsymbol{\beta}_{(k)}(\boldsymbol{\tau}^2_{(k)}+\boldsymbol{\beta}^T_{(k)}\boldsymbol{\beta}_{(k)})^{-1}\boldsymbol{\beta}^T_{(k)}. \end{aligned}$$

We treat $\frac{1}{2}\sum_{i=1}^{n}E(X[i,]X[i,]^T|\mathbf{Y},\boldsymbol{\beta}_{(k)},\boldsymbol{\tau}^2_{(k)})$ as a constant because it does not involve

$\boldsymbol{\beta}$ or $\boldsymbol{\tau}^2$. Hence without the constants $ELL_{(k)}(\boldsymbol{\beta}, \boldsymbol{\tau}^2)$ can be expressed as

$$-\frac{1}{2}\sum_{j=1}^{p}\left[n\log\tau_j^2+\sum_{i=1}^{n}\frac{(Y_{ij}^2-2Y_{ij}Y[i,]\boldsymbol{\delta}\beta[,j]+\beta[,j]^T[\boldsymbol{\Delta}+\boldsymbol{\delta}^TY[i,]^TY[i,]\boldsymbol{\delta}]\beta[,j])}{\tau_j^2}\right].$$

As the M step, we maximize the so-called Q function defined as

$$Q(\boldsymbol{\beta}, \boldsymbol{\tau}^2) = ELL_{(k)}(\boldsymbol{\beta}, \boldsymbol{\tau}^2) - \frac{1}{2}P_\lambda(\boldsymbol{\beta}). \tag{3.8}$$

However, it would take another iterative process to find the maximizer of the Q function. To mitigate the computation difficulty, we just find an update to increase the Q function rather than maximize it. This idea was introduced in the original EM paper (Dempster et al., 1977). First, we find $\boldsymbol{\tau}^2_{(k+1)}$ by letting

$$\boldsymbol{\tau}^2_{(k+1)} = \arg\max_{\tau^2}[Q(\boldsymbol{\beta}, \boldsymbol{\tau}^2)|\boldsymbol{\beta} = \boldsymbol{\beta}_{(k)}]. \tag{3.9}$$

Then we compute $\boldsymbol{\beta}_{(k+1)}$ by

$$\boldsymbol{\beta}_{(k+1)} = \arg\max_{\beta}[Q(\boldsymbol{\beta}, \boldsymbol{\tau}^2)|\boldsymbol{\tau}^2 = \boldsymbol{\tau}^2_{(k+1)}]. \tag{3.10}$$

It is easy to see that

$$\tau^2_{(k+1),j} = \frac{1}{n}\sum_{i=1}^{n}\{Y_{ij}^2-2Y_{ij}Y[i,]\boldsymbol{\delta}\beta_{(k)}[,j]+\beta_{(k)}[,j]^T[\boldsymbol{\Delta}+\boldsymbol{\delta}^TY[i,]^TY[i,]\boldsymbol{\delta}]\beta_{(k)}[,j]\}. \tag{3.11}$$

Given $\boldsymbol{\tau}^2_{(k+1)}$, we solve p separate maximization problems to get $\beta_{(k+1)}[,j], j = 1, 2, \ldots, p$. By straightforward calculations, we have

$$\beta_{(k+1)}[,j] = \arg\min_{\beta} -\frac{2(\sum_{i=1}^{n}Y_{ij}Y[i,])\boldsymbol{\delta}\beta}{\tau^2_{(k+1),j}}+\frac{\beta^T[n\boldsymbol{\Delta}+\boldsymbol{\delta}^T\mathbf{Y}^T\mathbf{Y}\boldsymbol{\delta}]\beta}{\tau^2_{(k+1),j}}+\sum_{l=1}^{q}\lambda|\beta_{lj}| \tag{3.12}$$

Note that (3.12) can be regarded as a Lasso-penalized least square problem. Thus we can efficiently compute $\beta_{(k+1)}[,j]$ by using the LARS-Lasso algorithm (Efron et al., 2003).

Let $\mathbf{\Sigma}^s = \frac{1}{n}\mathbf{Y}^T\mathbf{Y}$. We can rewrite (3.11) and (3.12) as

$$\tau^2_{(k+1),j} = \mathbf{\Sigma}^2_{jj} - 2\mathbf{\Sigma}^s[j,]\boldsymbol{\delta}\beta_{(k)}[,j] + \beta_{(k)}[,j]^T[\mathbf{\Delta} + \delta^T\mathbf{\Sigma}^s\delta]\beta_{(k)}[,j]. \tag{3.13}$$

$$\beta_{(k+1)}[,j] = \arg\min_{\beta} -\frac{2\Sigma^s[j,]\boldsymbol{\delta}}{\tau^2_{(k+1),j}}\beta + \frac{\beta^T[\mathbf{\Delta} + \delta^T\mathbf{\Sigma}^s\delta]\beta}{\tau^2_{(k+1),j}} + \sum_{l=1}^{q}\frac{\lambda}{n}|\beta_{lj}|. \tag{3.14}$$

The above procedure is summarized in Algorithm 1. We call algorithm 1 a generalized expectation-maximization (GEM) algorithm, because in the M-step the Q function is penalized condition log-likelihood and we increase the Q function rather than maximize it.

Algorithm 1 can also be used to compute the penalized estimator using a general penalty function $P_\lambda(|\beta|)$. We just replace $\lambda|\beta_l|$ with $P_\lambda(|\beta_l|)$ in step (3.b). The ℓ_1 penalty enjoys great computational advantages, for we can use the LARS-Lasso algorithm to solve the ℓ_1-penalized least squares problem in the same order of computations of an ordinary least-squares fit (Efron et al., 2003).

As a generalized E-M algorithm, algorithm 1 enjoys the nice ascent property which is formally proven in the Appendix. We should also point out that the ascent property has nothing to do with the normality assumption of the data, although we interpret the objective function as penalized log-likelihood of normal data.

Algorithm 1: GEM for sparse factor analysis

Step 0. Compute $\mathbf{\Sigma}^s = \mathbf{Y}^T\mathbf{Y}/n$.

Step 1 : Set initial values for $\boldsymbol{\beta}$ and $\boldsymbol{\tau}^2$.

Step 2 : Calculate $\boldsymbol{\delta}$, $\mathbf{\Delta}$:

$$\begin{aligned} \boldsymbol{\delta} &= (\boldsymbol{\tau}^2 + \boldsymbol{\beta}^T\boldsymbol{\beta})^{-1}\boldsymbol{\beta}^T \\ \mathbf{\Delta} &= \mathbf{I} - \boldsymbol{\beta}(\boldsymbol{\tau}^2 + \boldsymbol{\beta}^T\boldsymbol{\beta})^{-1}\boldsymbol{\beta}^T \end{aligned}$$

Compute the Cholesky decomposition: $\mathbf{Z}^T\mathbf{Z} = \mathbf{\Delta} + \delta^T\mathbf{\Sigma}^s\delta$.

Step 3 : For $j = 1, \ldots, p$

(3.a) Compute τ_j^2 by (3.13).

(3.b) Compute $\tilde{Y} = (\Sigma^s[j,]\boldsymbol{\delta}\mathbf{Z}^{-1})^T$. Then solve the following penalized least squares problem:

$$\beta[,j] = \arg\min_{\beta} \|\tilde{Y} - \mathbf{Z}\beta\|_2^2 + \frac{\tau_j^2}{n}\sum_{l=1}^{q}\lambda|\beta_l|.$$

Step 4 : Repeat Steps 2-3 till convergence.

3.4 Simulation

We examine the performance of the Lasso, ALasso, and rotated (varimax) estimators in situations where the factor loadings matrix is sparse. The first simulation model is generated by taking i.i.d. random vectors Y_i of length 12 from a normal distribution with mean 0 and covariance $\boldsymbol{\Sigma} = \boldsymbol{\beta}^T\boldsymbol{\beta} + \boldsymbol{\tau}^2$ where

$$\boldsymbol{\beta} = \begin{bmatrix} 1.8 & 1.8 & 1.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.6 & 1.6 & 1.6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.7 & 1.7 & 1.7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.5 & 1.5 & 1.5 \end{bmatrix},$$

$$\boldsymbol{\tau}^2 = \text{diag}(0.50, 0.13, 0.08, 0.89, 0.12, 0.32, 0.58, 0.71, 0.83, 0.36, 0.09, 0.10),$$

and the second simulation model is generated in the same manner where

$$\boldsymbol{\beta} = \begin{bmatrix} 1.5 & 0 & 0 & 0 & 1.5 & 0 & 0 & 0 & 1.5 & 0 & 0 & 0 \\ 0 & 1.7 & 0 & 0 & 0 & 1.7 & 0 & 0 & 0 & 1.7 & 0 & 0 \\ 0 & 0 & 1.6 & 0 & 0 & 0 & 1.6 & 0 & 0 & 0 & 1.6 & 0 \\ 0 & 0 & 0 & 1.8 & 0 & 0 & 0 & 1.8 & 0 & 0 & 0 & 1.8 \end{bmatrix},$$

$$\boldsymbol{\tau}^2 = \text{diag}(0.03, 0.77, 0.76, 0.99, 0.91, 0.89, 0.43, 0.51, 0.25, 0.05, 0.65, 0.43),$$

and the third simulation model is generated where

$$\boldsymbol{\beta} = \begin{bmatrix} 1.8 & 1.8 & 1.8 & 1.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.6 & 1.6 & 1.6 & 1.6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.7 & 1.7 & 1.7 & 1.7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.5 & 1.5 & 1.5 \end{bmatrix},$$

$$\boldsymbol{\tau}^2 = \text{diag}(0.50, 0.13, 0.08, 0.89, 0.12, 0.32, 0.58, 0.71, 0.83, 0.36, 0.09, 0.10).$$

The interpretation of the first factor model is that variables $3k-2$, $3k-1$, and $3k$ are random perturbations of factor k, $k = 1, 2, 3, 4$, and for the second factor model, variables k,$k+4$, and $k+8$ are random perturbations of factor k. The third factor model is the same as the first factor model except for the fourth, seventh, and tenth columns. Within each of the 100 replications we generated 100 training data and separate 100 validation data for the three simulation models. In these simulation studies we compared four methods: the varimax rotation (ordinary MLE), the Lasso and ALasso estimators, and the oracle estimator, which is defined as the MLE when it is known which entries of the factor loading matrix should be zero.

Suppose a method μ produces an estimator $\widehat{\boldsymbol{\beta}}(\mu)$ and $\widehat{\boldsymbol{\tau}^2}(\mu)$. Write

$$\boldsymbol{\Sigma}(\mu) \equiv \widehat{\boldsymbol{\beta}}(\mu)^T \widehat{\boldsymbol{\beta}}(\mu) + \widehat{\boldsymbol{\tau}^2}(\mu).$$

We define two K-L measurements of μ as follows

$$KL(\mu) = \frac{1}{2}\log(det(\boldsymbol{\Sigma}(\mu)) + \frac{1}{2}tr(\boldsymbol{\Sigma}(\mu)^{-1}\boldsymbol{\Sigma})) - \frac{1}{2}\log(det(\boldsymbol{\Sigma})) - \frac{p}{2} \tag{3.15}$$

$$KL(\mu)_{\text{v}} = \frac{1}{2}\log(det(\boldsymbol{\Sigma}(\mu)) + \frac{1}{2}tr(\boldsymbol{\Sigma}(\mu)^{-1}\boldsymbol{\Sigma}_{\text{v}})) - \frac{1}{2}\log(det(\boldsymbol{\Sigma}_{\text{v}})) - \frac{p}{2} \tag{3.16}$$

where $\boldsymbol{\Sigma}_{\text{v}}$ is the sample covariance matrix computed using the validation data. The KL loss in ((3.15)) measures the goodness of fit of μ and KL_{v} in ((3.16)) is used to select meta-parameters (if any) of μ. We report the relative K-L loss (RKL) defined as $\frac{KL(\mu)}{KL(\text{mle})}$.

The true model has $q = 4$ factors. We did not use this information in the study. We treated q as a meta-parameter of the MLE and used the q minimizing $KL(\text{mle})_{\text{v}}$. K-L loss is expected to be large when q is small or large, whereas it is expected to be small when q is appropriate. Therefore, we select q when q value is the minimizer of

the K-L loss curve. That is, K-L losses are computed for a sequence of q with MLE models evaluated on the validation data set to obtain its model error. Then, q is selected when K-L loss has a minimum value.

In all 100 replications, the selected q was 4. Figure Figure 3.1 shows three plots of $KL(\text{mle})_{\text{v}}$ vs. q for simulation 1, 2, and 3.

From Figure Figure 3.2 and the second column of Table Table 3.1, it is clear that both the Lasso and the ALasso estimators are more accurate than the MLE. Moreover, ALasso is more accurate than Lasso and is also very close to the oracle. When we compare the three models, the performance of model 3 is not as good as that of models 1 and 2. The RKLs of model 3 are slightly higher than that of models 1 and 2. Further, the number of zeros is slightly lower than that of models 1 and 2. Therefore, based on this information, our analysis would probably work best when true factor loadings have only one nonzero entry on each column in the loadings matrix and the performance decreases when there are more than one nonzero entries on each column (also see Appendix Appendix A). Figure Figure 3.3 shows paired RKL values evaluated on validation data for ALasso and oracle in the 100 replications. It is interesting to see that 4 or 5 out of 100 times the ALasso did slightly better than the oracle in all three models as we can see that the RKLs of ALasso are smaller than those of oracle in those cases.

In the third column of Table Table 3.1, it is seen that the ALasso discovered as many zeros as the rotation method (varimax) and many more zero loadings than the Lasso did. For the rotation, loadings are truncated to zeros when they are less than 0.1. Without truncation, rotation method does not have any zeros. To illustrate their difference, we made the histogram of the number of estimated zero loadings for Lasso, ALasso, and rotation, as shown in Figures Figure 3.4, Figure 3.5, and Figure 3.6. The performance of the rotation is better than we expect. However, the performance depends upon the threshold. If we choose a threshold other than 0.1, the performance

would not be as good. The problem is that it might not be easy to find an appropriate threshold in some dataset.

method	RKL	Number of zeros
Oracle	0.732 (0.092)	36
Lasso	0.960 (0.025)	11 (3.39)
ALasso	0.767 (0.083)	33 (3.17)
Rotation	1	33 (2.42)
method	RKL	Number of zeros
Oracle	0.718 (0.093)	36
Lasso	0.949 (0.024)	12 (3.09)
ALasso	0.749 (0.083)	33 (2.84)
Rotation	1	33 (2.67)
method	RKL	Number of zeros
Oracle	0.777 (0.082)	33
Lasso	0.975 (0.019)	8 (3.83)
ALasso	0.820 (0.068)	30 (3.00)
Rotation	1	30 (2.55)

Table 3.1: Averages based on 100 replications(top: model 1, middle: model 2, bottom: model 3). Numbers in $(\cdot)$ are standard errors.

The ℓ_1 penalized method is comparable to the rotation method. It might not be easy to compare the factor loadings provided by the two methods, but the loadings are similar in terms of $\boldsymbol{\beta}^T\boldsymbol{\beta}+\boldsymbol{\tau}^2$. In the following three matrices, the absolute differences of the element-wise factor loadings (left side of each slash) between the Lasso and rotation and the loadings produced by Lasso (right side of each slash) are shown with averages after 100 repetitions (top: model 1; middle: model 2; bottom: model 3). The absolute differences range from 0.04 to 0.25.

$Model1$	$c1$	$c2$	$c3$	$\cdots$	$c10$	$c11$	$c12$
$r1$	.14/.86	.14/.90	.14/.91	$\cdots$	.05/.13	.05/.13	.05/.14
$r2$	.23/.81	.23/.86	.23/.87	$\cdots$	.10/.04	.10/.04	.10/.04
$r3$	.25/.80	.25/.85	.25/.86	$\cdots$	.11/.03	.11/.03	.11/.03
$r4$	.13/.25	.13/.25	.13/.25	$\cdots$	.12/.26	.12/.25	.12/.26
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$r10$	.06/.14	.06/.14	.06/.14	$\cdots$	.13/.86	.13/.91	.13/.90
$r11$	.10/.04	.10/.04	.10/.04	$\cdots$	.23/.81	.23/.86	.23/.86
$r12$	.09/.05	.09/.04	.09/.05	$\cdots$	.23/.81	.23/.86	.23/.86

$Model2$	$c1$	$c2$	$c3$	$\cdots$	$c10$	$c11$	$c12$
$r1$	.14/.86	.04/.02	.05/.02	$\cdots$	.05/.02	.05/.02	.05/.02
$r2$	.19/.20	.07/.87	.18/.20	$\cdots$	.07/.95	.18/.20	.17/.21
$r3$	.20/.22	.20/.22	.09/.87	$\cdots$	.20/.22	.09/.89	.21/.23
$r4$	.22/.24	.21/.23	.22/.23	$\cdots$	.21/.23	.23/.23	.09/.92
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$r10$	.05/.03	.12/.77	.05/.03	$\cdots$	.13/.86	.05/.02	.05/.03
$r11$	.17/.19	.17/.19	.07/.85	$\cdots$	.17/.18	.06/.87	.18/.19
$r12$	.10/.12	.09/.12	.11/.12	$\cdots$	.09/.12	.11/.12	.06/.87

$Model3$	$c1$	$c2$	$c3$	$\cdots$	$c10$	$c11$	$c12$
$r1$	.13/.87	.13/.91	.13/.92	$\cdots$	.05/.14	.05/.13	.05/.13
$r2$	.22/.82	.22/.87	.22/.87	$\cdots$	.10/.04	.10/.04	.10/.04
$r3$	.24/.81	.24/.86	.24/.87	$\cdots$	.12/.03	.11/.03	.11/.02
$r4$	.10/.68	.10/.71	.10/.72	$\cdots$	.05/.13	.05/.13	.05/.13
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$r10$	.14/.07	.15/.07	.15/.07	$\cdots$	.18/.88	.14/.61	.14/.61
$r11$	.10/.04	.09/.04	.09/.04	$\cdots$	.16/.58	.20/.89	.20/.89
$r12$	.10/.04	.09/.04	.09/.04	$\cdots$	.16/.58	.20/.89	.20/.89

Finally, the likelihood ratio tests are performed to determine if the ℓ_1 penalized factor models are acceptable by the likelihood criterion. In this test, only training data are used. By performing the likelihood ratio test in (1.4), we attain p-values near 1 for all three Lasso models, and .15 for both ALasso models 1 and 2. The p-value for ALasso model 3 is approximately .22. The p-values for ALasso models are less than those for the Lasso models, but both of the ℓ_1 penalty models are within the confidence intervals of the maximum likelihood model.

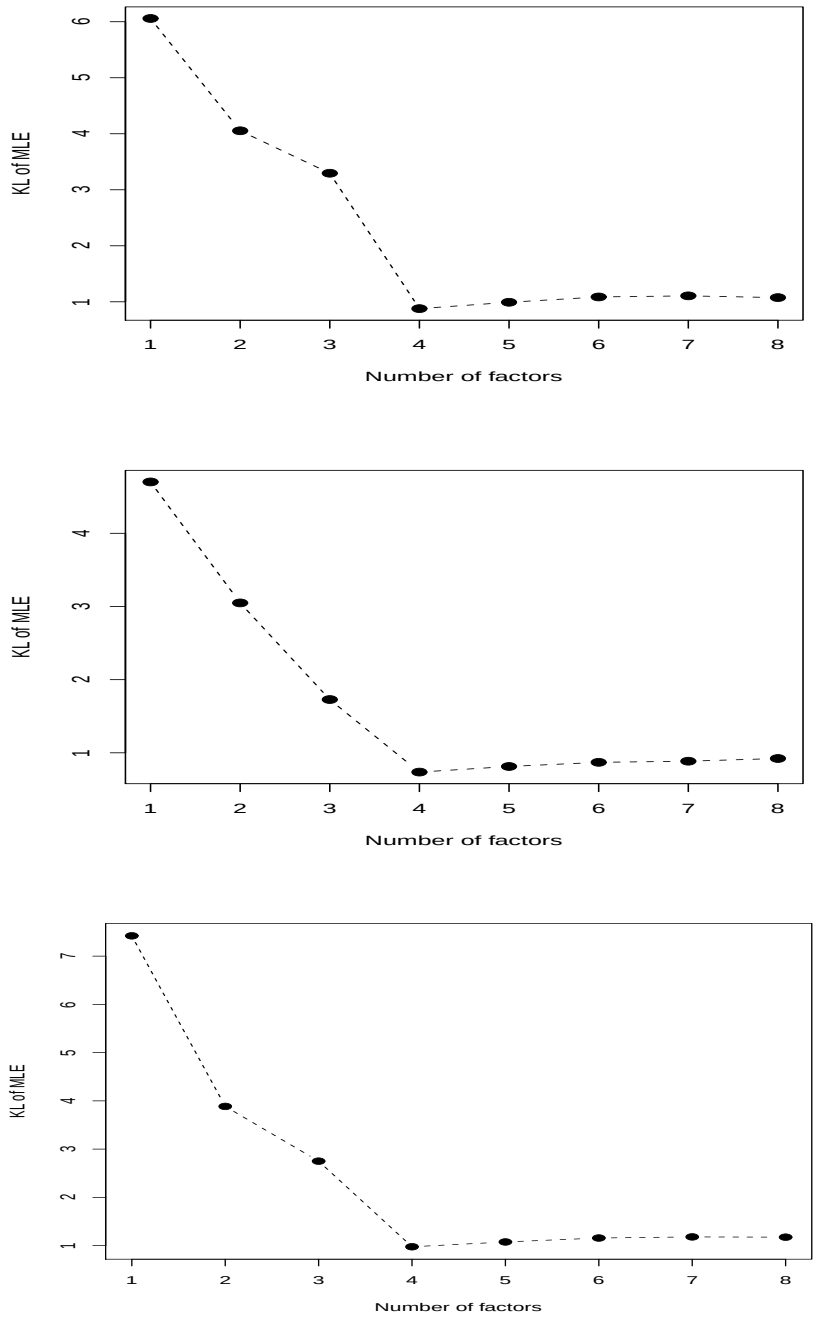

Figure 3.1: The y-axis shows the value of $KL(\text{mle})_{\text{v}}$ for q factors; top: model 1, middle: model 2, bottom: model 3.

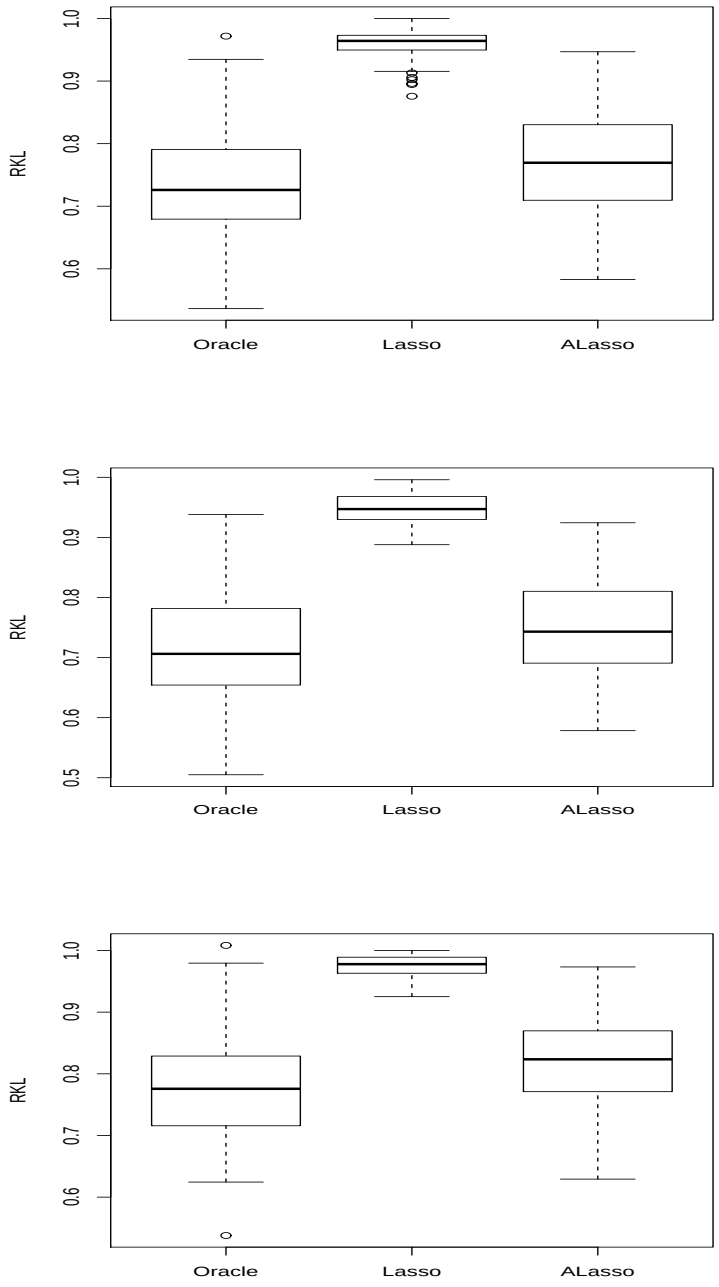

Figure 3.2: Boxplots of RKLs of the Lasso, ALasso and oracleestimators with respect to the MLE (top: model 1, middle: model 2, bottom: model 3).

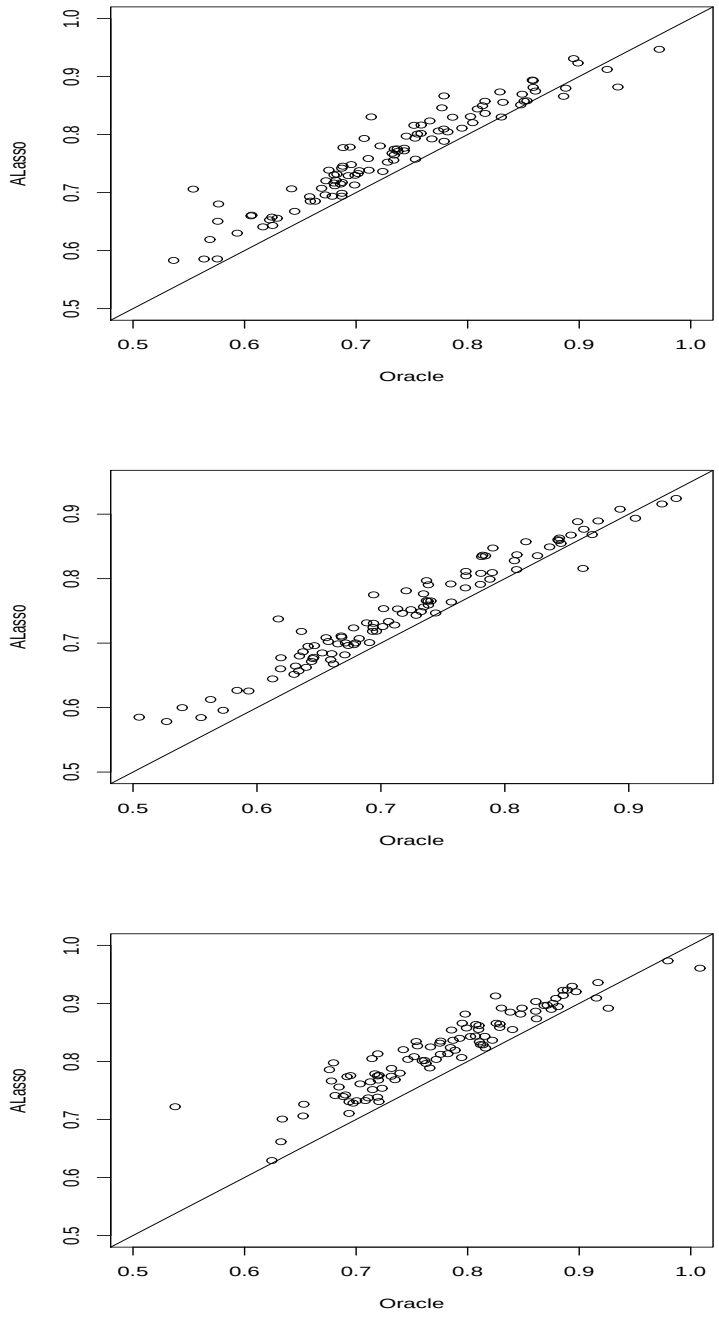

Figure 3.3: Pairwise comparison of ALasso and oracle. Thesolid straight line is the 45 degree line (top: model 1, middle: model 2, bottom: model 3).

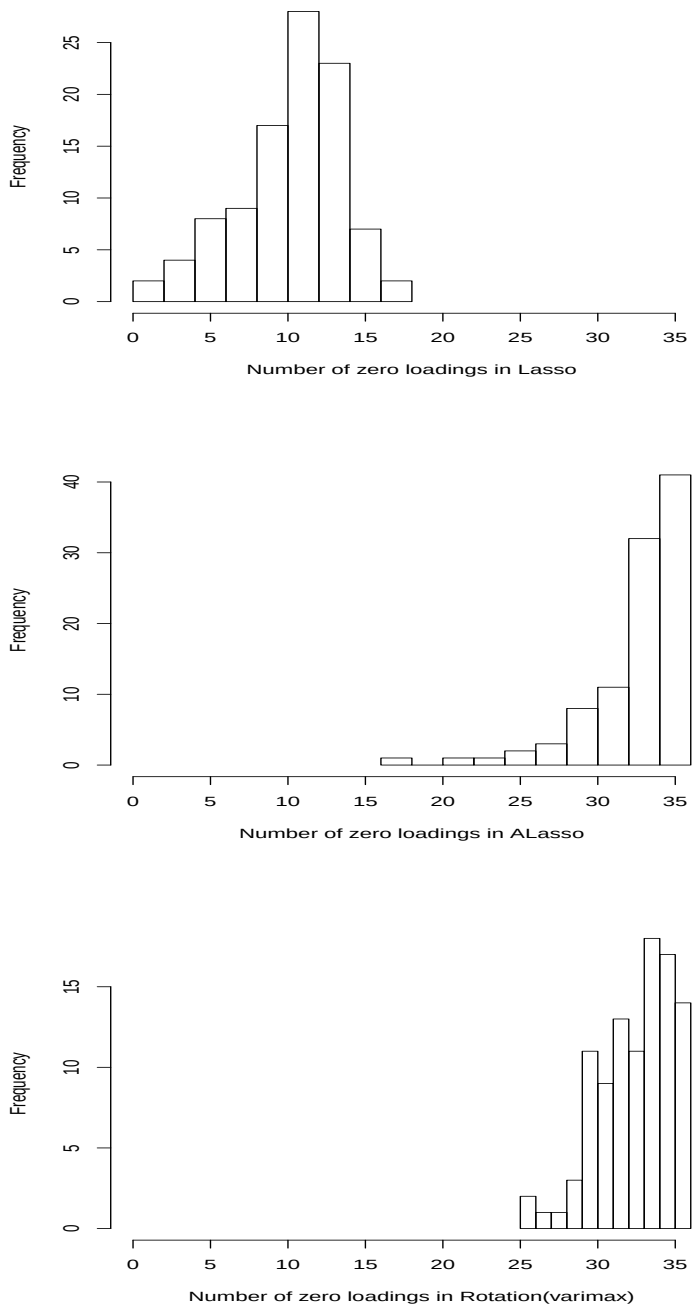

Figure 3.4: Comparing the sparsity pursuit performance ofLasso, ALasso, and rotation of model 1 (top: Lasso, middle: ALasso, bottom:rotation(varimax) curtailed at 0.1).

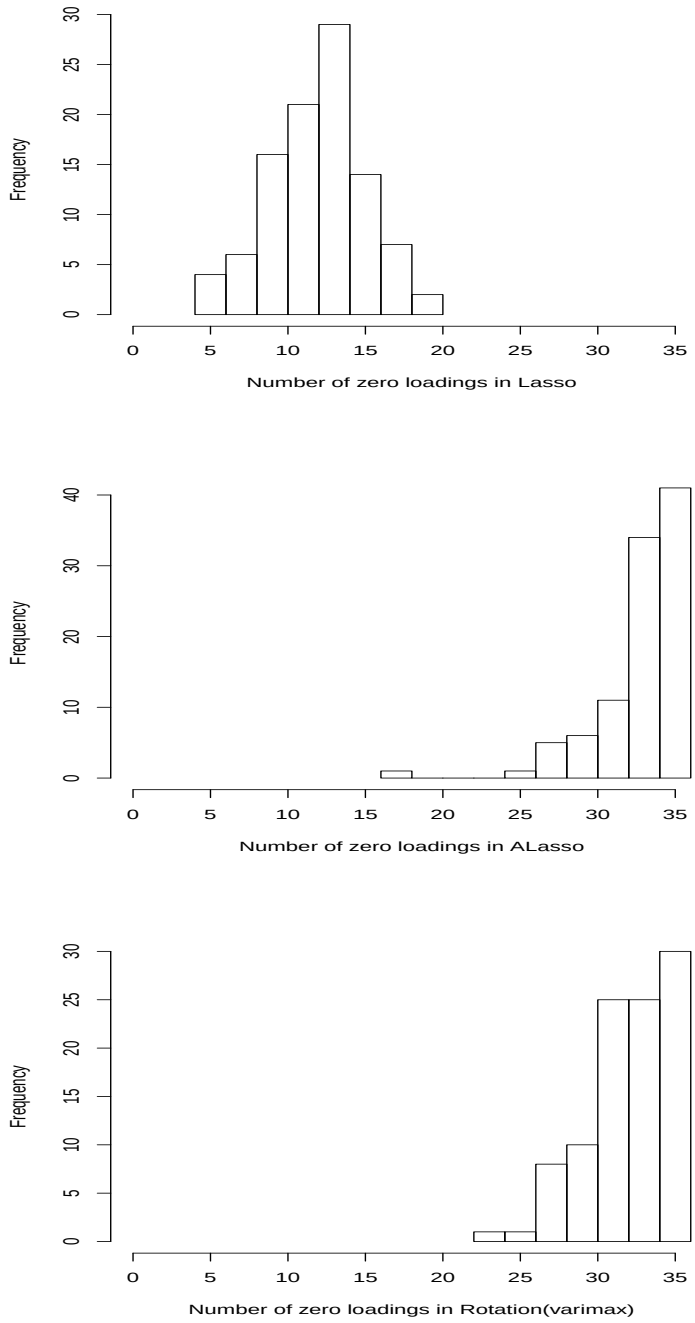

Figure 3.5: Comparing the sparsity pursuit performance ofLasso, ALasso, and rotation of model 2 (top: Lasso, middle: ALasso, bottom:rotation(varimax) curtailed at 0.1).

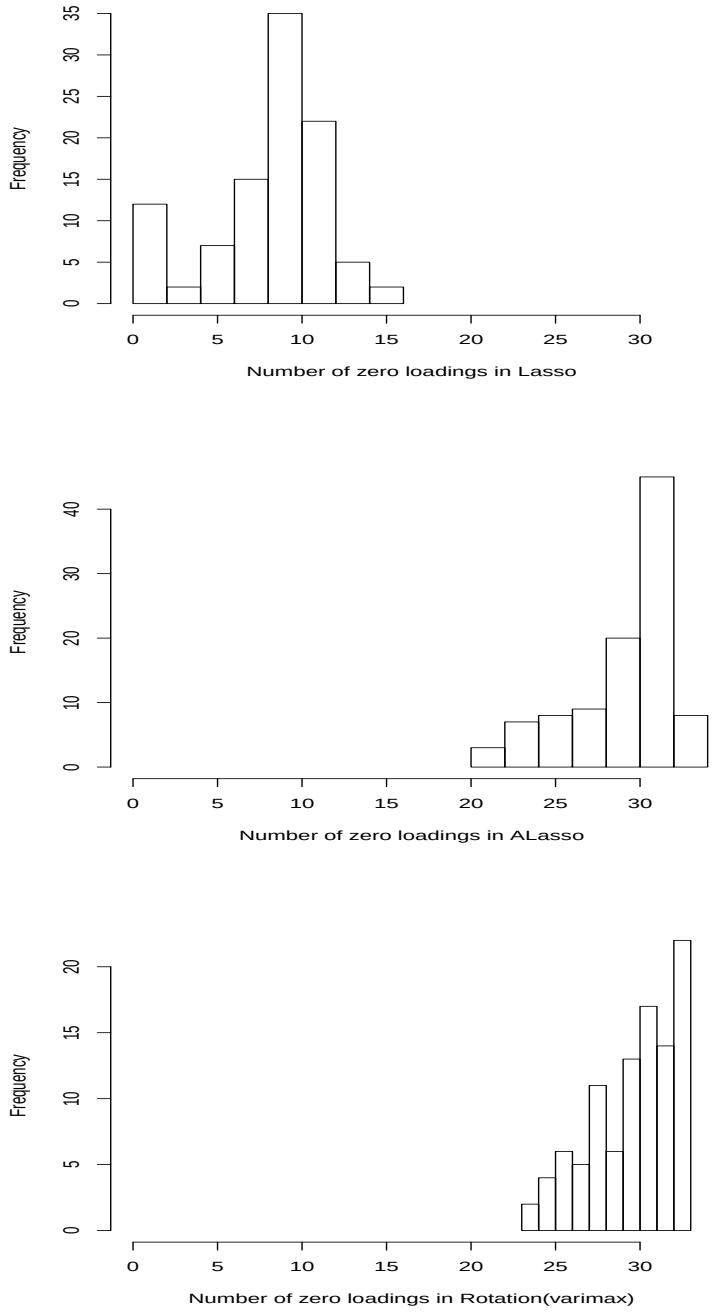

Figure 3.6: Comparing the sparsity pursuit performance ofLasso, ALasso, and rotation of model 3 (top: Lasso, middle: ALasso, bottom:rotation(varimax) curtailed at 0.1).

3.5 Real Data Example

3.5.1 Body fat data

The dataset is from Medicine and Science in Sports and Exercise vol. 17, no. 2, April 1985, p. 189 (Penrose et al., 1985). The data are the estimates of the percentage of body fat determined by underwater weighing and various body circumference measurements for 252 men. In this dataset, we use all of the variables except body fat to examine whether relationships exist among them. The 14 variables listed are Density determined from underwater weighing, Age (years), Weight (lbs), Height (inches), Neck circumference (cm), Chest circumference (cm), Abdomen 2 circumference (cm), Hip circumference (cm), Thigh circumference (cm), Knee circumference (cm), Ankle circumference (cm), Biceps (extended) circumference (cm), Forearm circumference (cm), Wrist circumference (cm). We first select q, the number of common factors, and then λ by computing KL loss. For KL loss calculations, the data are divided into two parts: the training data and the validation data. One third of the original data are randomly assigned to the validation dataset (84 observations), and the rest are assigned to the training dataset (168 observations). Using the same method as in the simulation model, KL loss is computed using the estimated parameters from the training dataset and the sample covariance from the validation dataset. First KL loss is computed for choosing q with $\lambda = 0$ and then for selecting λ with chosen q. q is selected at the minimum point on the KL loss curve from a sequence of q and λ is chosen at the minimum point on the KL loss curve from equally spaced λs. $q = 5$ is chosen and $\lambda = 10$ is selected for the Lasso penalty and 1 is chosen for the ALasso penalty (with γ of 1) for this dataset. Table Table B.2 and Figure Figure 3.7 show a sequence of q and corresponding KL loss. The KL loss has a minimum for $q = 5$. In Tables Table B.3 and Table B.4 and Figures Figure B.6 and Figure B.7, a series of λs and corresponding KL loss are computed for the Lasso and ALasso methods. The

smallest KL loss by the Lasso and ALasso models are 1.664 and 1.676, respectively, whereas the KL loss of the MLE is 1.683. Because there is little room for improving the accuracy of the MLE using ℓ_1 penalization, it seems more reasonable to use the sparsity-first rule, namely that we use a sparse factor model with the highest sparsity as long as its KL loss is smaller than that of the MLE. The sparsity-first rule chooses a Lasso model with 17 exact zero loadings when $\lambda = 18$ and an ALasso model with 16 exact zero loadings when $\lambda = 2, \Gamma = 1$. A varimax rotation model has 21 zero loadings when curtailed at 0.1. Therefore, the Lasso, ALasso, and rotation models show similar performances in this example.

The ℓ_1 penalty models are tested if they are within the confidence interval of the maximum likelihood model at the 0.01 level. By performing the likelihood ratio test in ((1.4)), we attain a p-value of 0.88 for the Lasso model with $\lambda = 18$ and a p-value of 0.99 for the ALasso model with $\lambda = 2, \Gamma = 1$. The ℓ_1 penalties models are within the confidence interval of the maximum likelihood model.

Table Table 3.5 shows β (factor loadings) and τ^2 (uniqueness) for Lasso and ALasso penalties, as well as the factor rotation method. When we compare the β and τ^2 in both penalties, they show similar trends. Further, when we compare $\beta'\beta$ rather than β between the penalized and the rotation methods, they show somewhat similar trends.

Table 3.2: A sequence of q and corresponding KL loss for Body fat data

q	1	2	3	4	5	6	7	8	...	14
KL	3.33	2.45	2.00	1.80	1.76	1.86	1.80	1.83	...	1.79

Figures Figure 3.10, Figure 3.11, and Figure 3.12 show LL and penalized log-likelihoods (PLL) with Lasso and ALasso estimates of body fat data, respectively. In recording LL (PLL), we start at the fifth iteration and record LL (PLL) after every 50

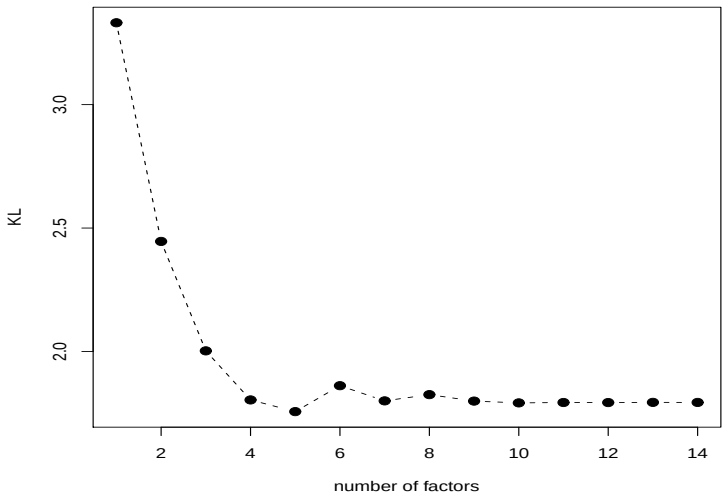

Figure 3.7: Body fat data: KL loss vs. number of factors

Table 3.3: A sequence of λ and corresponding KL for Body fat data with q = 5 :Lasso

λ	0	2	4	6	8	<u>10</u>	12	14	16	18
KL	1.683	1.678	1.670	1.666	1.665	<u>1.664</u>	1.666	1.669	1.673	1.677

Table 3.4: A sequence of λ and corresponding KL for Body fat data with q = 5: ALasso

λ	0	<u>1</u>	2	3	4	5
KL	1.683	<u>1.676</u>	1.682	1.687	1.706	1.735

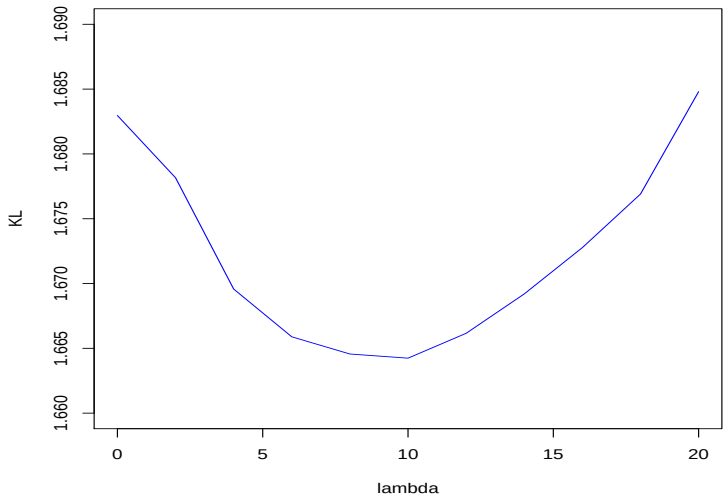

Figure 3.8: Body fat data: KL vs. λ (Lasso): q = 5

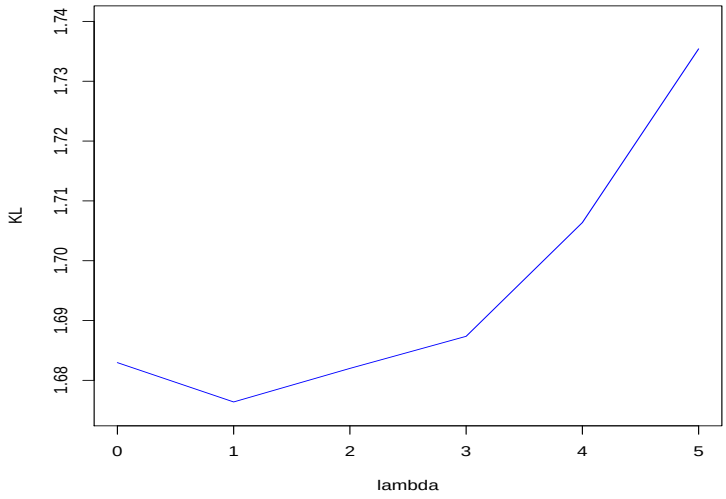

Figure 3.9: Body fat data: KL vs. λ (ALasso): q = 5

Table 3.5: The estimates of factor loadings and uniqueness of body fat data from Lasso (top) with penalty 18, ALasso (middle) with penalty 2, and rotation (bottom), respectively. For ALasso penalty, γ of 1 is used. Iteration converging gap of 0.0001 was used for β

	C1	C2	C3	C4	C5	C6	C7	...	C14
F1	-0.47	0.00	0.82	0.21	0.71	0.74	0.72	...	0.64
F2	0.60	-0.39	0.00	0.33	0.00	-0.25	-0.45	...	0.16
F3	0.00	0.58	0.00	0.00	0.17	0.00	0.00	...	0.44
F4	0.00	0.00	0.00	0.21	0.00	0.00	0.00	...	0.00
F5	0.00	0.00	-0.12	-0.31	0.00	-0.13	0.00	...	0.00
$\hat{\tau}^2$	0.23	0.41	0.01	0.64	0.18	0.11	0.02	...	0.16
F1	-0.61	0.00	0.95	0.23	0.79	0.89	0.89	...	0.67
F2	0.57	-0.56	0.00	0.40	0.00	-0.18	-0.38	...	0.00
F3	0.14	0.49	0.00	0.25	0.22	0.00	0.00	...	0.46
F4	0.00	0.00	0.00	0.00	0.20	0.00	0.00	...	0.00
F5	0.00	0.14	0.00	-0.13	0.19	0.00	0.00	...	0.34
$\hat{\tau}^2$	0.23	0.41	0.01	0.64	0.19	0.11	0.02	...	0.16
F1	-0.77	0.00	0.80	0.00	0.57	0.80	0.91	...	0.33
F2	-0.19	0.00	0.41	0.14	0.57	0.42	0.29	...	0.54
F3	0.23	0.00	0.38	0.58	0.30	0.21	0.00	...	0.42
F4	-0.25	0.75	0.00	0.00	0.13	0.19	0.24	...	0.33
F5	0.00	0.00	0.22	0.00	0.18	0.00	0.13	...	0.48
$\hat{\tau}^2$	0.26	0.42	0.01	0.63	0.22	0.10	0.01	...	0.10

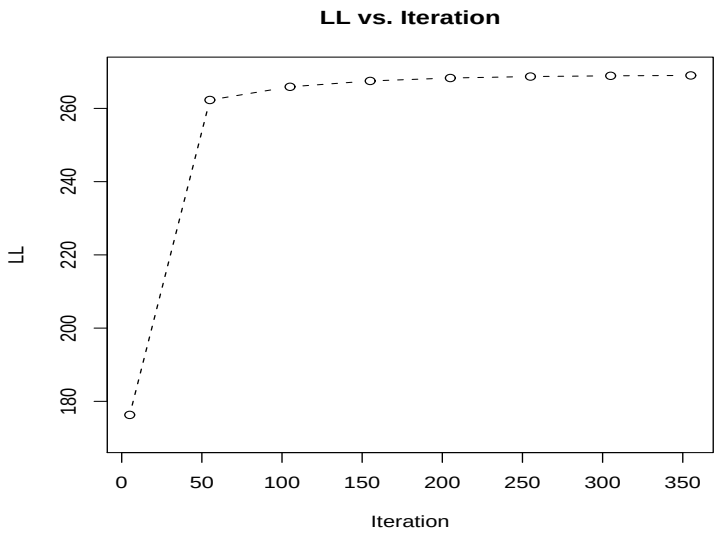

Figure 3.10: LL of body fat data

iterations. PLL_L1 means PLL for Lasso, PLL_ALASSO is PLL for ALasso. As we proved in the Appendix, the LL and two PLLs consistently increase. The iterations in ALasso are much less than the iterations in Lasso as seen in Figures Figure 3.11 and Figure 3.12. λ of 60 is used for Lasso whereas a λ of 10 is used for ALasso with γ of 1.

3.5.2 Holzinger and Swineford (HS)

The dataset is from Holzinger and Swineford (1939). Primarily used for factor analysis, the dataset represents nine cognitive abilities.

We first select q, and then λ by computing KL loss. For KL loss calculations, the data are divided into two parts: the training data and validation data. One third of the original data are randomly assigned to the validation dataset, and the rest are assigned to the training dataset. KL loss is computed using the estimated

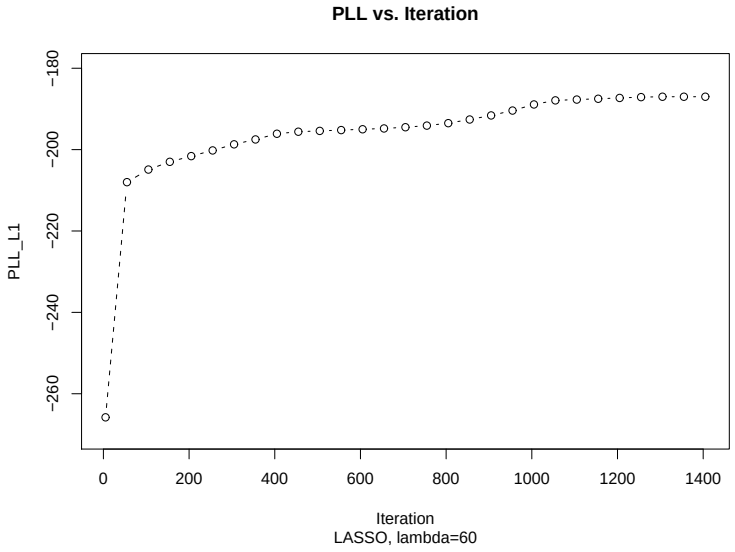

Figure 3.11: PLL of body fat data for Lasso

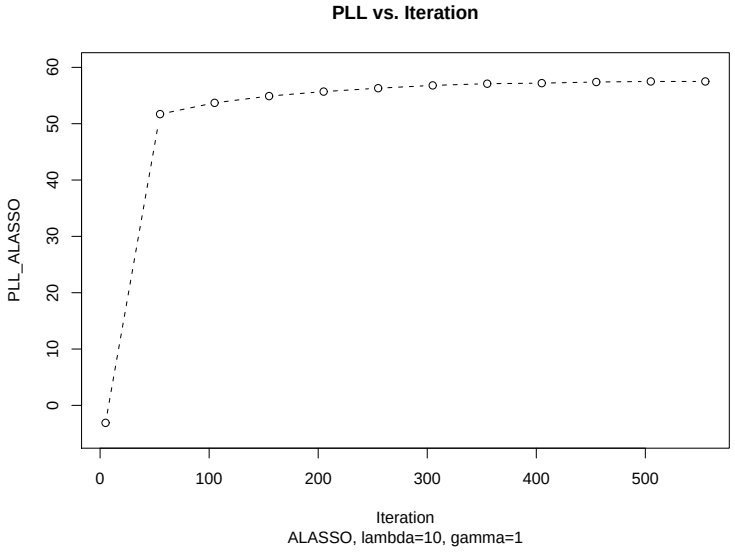

Figure 3.12: PLL of body fat data for ALasso

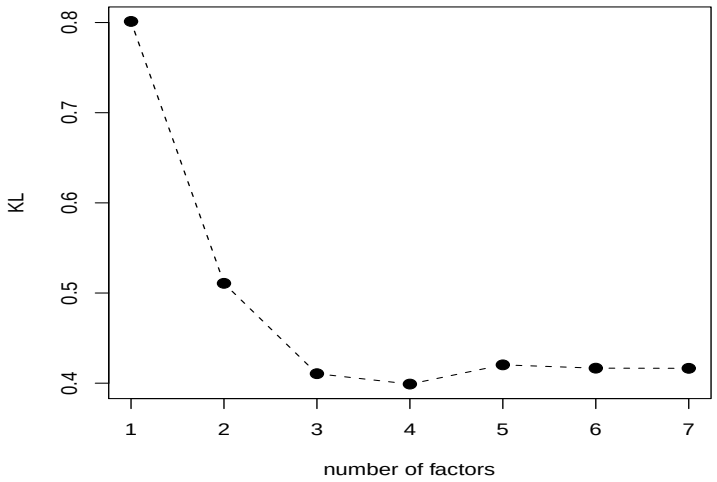

Figure 3.13: HS Data: KL loss vs. number of factors

parameters from the training dataset and the sample covariance from the validation dataset. First, KL loss is computed for choosing q with $\lambda = 0$ and then for selecting λ with the chosen q. q is selected at the minimum point on the KL loss curve from a sequence of q and then λ is chosen at the minimum point on the KL loss curve from equally spaced λs. $q = 4$ is chosen and $\lambda = 15$ is selected for the Lasso penalty, whereas 10 is chosen for the ALasso penalty (with γ of 1) for this dataset. Figure Figure 3.13 shows a sequence of q and corresponding KL loss.

In Tables Table 3.6 and Table 3.7 and Figures Figure 3.14 and Figure 3.15, a series of λ and corresponding KL loss are computed for the Lasso and ALasso method. The smallest KL loss by the lasso and the ALasso models are 0.394 and 0.376, respectively, whereas the KL loss of the MLE is 0.400. Because there is little room for improving the accuracy of the MLE by ℓ_1 penalization, it seems more reasonable to use the sparsity-first rule, namely that we utilize the sparse factor model with the highest sparsity as long as its KL loss is smaller than that of the MLE. The sparsity-first

rule chooses a lasso model with 11 exact zero loadings when $\lambda = 25$ and an ALasso model with 8 exact zero loadings when $\lambda = 15, \Gamma = 1$. A varimax rotation model has 18 zero loadings when curtailed at 0.1. The ℓ_1 penalty models are tested if they are within the confidence interval of the maximum likelihood model at the 0.01 level. By performing the likelihood ratio test in (1.4), we attain a p-value of 0.001 for the Lasso model with $\lambda = 25$ and a p-value of almost zero for the ALasso model with $\lambda = 15, \Gamma = 1$. Therefore, the ℓ_1 penalty models are not within the confidence interval of the maximum likelihood model.

Table 3.6: A sequence of λ and corresponding KL for HS data with q = 4 :Lasso

λ	0	5	10	15	20	25	30	$\cdots$
KL	0.400	0.395	0.394	0.394	0.396	0.399	0.404	$\cdots$

Table 3.7: A sequence of λ and corresponding KL for HS data with q = 4 :ALasso

λ	0	5	10	15	20	$\cdots$
KL	0.400	0.392	0.376	0.382	0.411	$\cdots$

Table Table 3.8 shows the three estimates for the factor loadings. The starting values of the EM algorithm are provided as close values by factanal in R packages without rotation, with rotation, and random choices, respectively. In addition, 0.0001 is used for the iteration converging gap of $\boldsymbol{\beta}$.

The values of these three factor loadings are different depending on the starting values. However, all three of the factor loadings actually have only one solution. That is, $\boldsymbol{\beta\beta}$ and $\boldsymbol{\tau}^2$ are all the same.

The comparisons of the three factor loadings by the Lasso, ALasso, and rotation (varimax) in Table Table 3.9 generally provide a consistent explanation of the data.

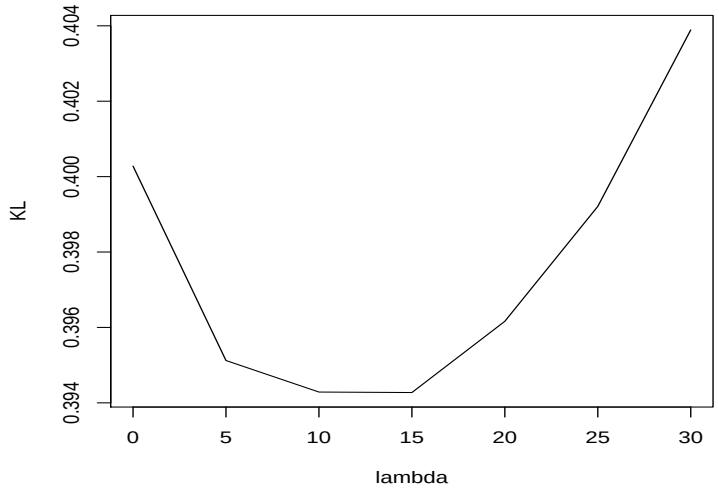

Figure 3.14: HS data: KL vs. λ (Lasso): q = 4

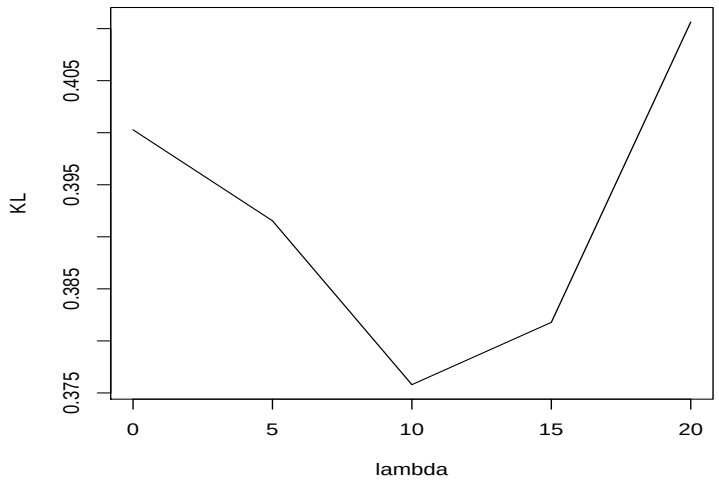

Figure 3.15: HS data: KL vs. λ (ALasso): q = 4

Table 3.8: HS data: Three estimates for factor loadings. The starting values are given as close values by factanal in R packages without rotation, with rotation, and random choices, respectively.

	Factor1	Factor2	Factor3	Factor4	$\hat{\underline{\boldsymbol{\tau}}}^2$
visual	0.47	0.05	0.49	-0.02	0.53
cubes	0.28	-0.07	0.39	-0.06	0.76
flags	0.27	0.08	0.63	-0.26	0.45
paragrap	0.83	0.15	-0.12	-0.09	0.27
sentence	0.86	0.07	-0.18	0.12	0.21
wordm	0.82	0.11	-0.05	-0.05	0.31
addition	0.00	0.99	0.00	0.00	0.01
counting	0.12	0.48	0.30	0.31	0.56
straight	0.29	0.34	0.47	0.35	0.46
visual	0.29	0.57	0.22	-0.03	0.54
cubes	0.15	0.44	0.09	-0.11	0.76
flags	0.06	0.74	0.06	0.07	0.45
paragrap	0.82	0.17	0.02	0.15	0.27
sentence	0.87	0.04	0.15	0.01	0.21
wordm	0.79	0.21	0.07	0.09	0.31
addition	0.00	0.00	0.38	0.92	0.01
counting	0.03	0.17	0.56	0.29	0.57
straight	0.13	0.36	0.62	0.11	0.45
visual	0.19	-0.02	0.65	-0.05	0.53
cubes	0.05	-0.05	0.46	-0.13	0.76
flags	-0.09	0.18	0.71	-0.13	0.45
paragrap	0.25	-0.36	0.47	0.56	0.27
sentence	0.42	-0.47	0.38	0.49	0.21
wordm	0.28	-0.36	0.50	0.47	0.31
addition	0.32	0.77	0.06	0.57	0.01
counting	0.49	0.39	0.22	0.05	0.56
straight	0.54	0.26	0.42	-0.09	0.45

The columns are reorganized so that they match on all three methods. Clearly, the variables of paragrap, sentence, and wordm form factor 1, the variables of flags, visual, cubes, and straight form factor 4, and the variables of addition and counting form factor 3. However, the variable of sentence positively belongs to factor 1 on the Lasso and ALasso but seems to belong to both factors 1 and 2 on the rotation.

3.5.3 Parkinsons

The Oxford Parkinson's data (Little et al., 2008) is from "Suitability of Dysphonia Measurements for Tele-monitoring of Parkinson's Disease" by Max A. Little, Patrick E. McSharry, Eric J. Hunter, Lorraine O. Ramig (2008), IEEE Transactions on Biomedical Engineering (to appear). The dataset consists of 23 columns and 195 rows. Each column is a particular voice measure, and each row is one of 195 voice recordings from 31 individuals, 23 of which have Parkinson's disease (PD). We apply the proposed sparse factor analysis method. We randomly split the data into a training set (130 observations) and a validation set (65 observations) and measure the accuracy of the model by computing the KL loss evaluated on the validation set. Before fitting any factor model, we standardized the data so that each feature has mean 0 and standard deviation 1.

We select q at the minimum point on the KL loss curve from a sequence of q and then λ at the minimum point on the KL loss curve from equally spaced λ's. The KL loss for q = 8 has a minimum value. Therefore, we select q = 8 and $\lambda = 3$ for the Lasso penalty and 1 for the ALasso penalty (with γ of 1) for this dataset. The smallest KL loss by the lasso is 14.87, whereas the KL loss of the MLE model is 14.89. Because there is little room for improving the accuracy of the MLE using ℓ_1 penalization, it seems more reasonable to use the sparsity-first rule. The sparsity-first rule chooses a lasso model with 15 zero loadings when λ is 4. The ALasso model has 27 zero loadings when $\lambda = 1$.

Table 3.9: HS data: Comparison of factor loadings. top: Lasso with λ of 25; middle: ALasso with λ of 15; bottom: rotation (varimax) curtailed at 0.1.

	Factor1	Factor2	Factor3	Factor4	$\hat{\underline{\boldsymbol{\tau}}^2}$
visual	0.18	0.19	0.00	0.60	0.41
cubes	0.04	0.00	0.00	0.39	0.78
flags	0.06	0.00	0.00	0.54	0.60
paragrap	0.63	0.54	0.00	0.06	0.14
sentence	0.92	0.00	0.02	0.00	0.01
wordm	0.61	0.24	0.00	0.17	0.37
addition	0.00	0.03	0.58	0.00	0.58
counting	0.13	-0.06	0.66	0.17	0.38
straight	0.13	0.00	0.40	0.36	0.54
visual	0.22	0.23	0.00	0.60	0.41
cubes	0.07	0.00	0.00	0.42	0.76
flags	0.09	0.05	0.00	0.56	0.60
paragrap	0.66	0.57	0.00	0.04	0.11
sentence	0.94	0.00	0.02	0.00	0.01
wordm	0.64	0.26	0.01	0.17	0.37
addition	0.02	0.03	0.62	0.00	0.56
counting	0.16	-0.08	0.67	0.20	0.38
straight	0.17	0.00	0.41	0.39	0.54
visual	0.27	0.00	0.14	0.63	0.50
cubes	0.00	0.00	0.00	0.49	0.74
flags	0.00	0.00	0.13	0.64	0.57
paragrap	0.87	0.00	0.00	0.17	0.11
sentence	0.93	0.33	0.00	0.00	0.01
wordm	0.75	0.00	0.00	0.23	0.38
addition	0.00	0.00	0.74	0.00	0.42
counting	0.00	0.00	0.69	0.18	0.49
straight	0.13	0.00	0.52	0.42	0.53

Then, the ℓ_1 penalty models are tested if they are within the confidence interval of the maximum likelihood model. By performing the likelihood ratio test in (1.4), we attain a p-value of 0.99 for the Lasso model with $\lambda = 4$ and also a p-value of 0.99 for the ALasso model with $\lambda = 1, \Gamma = 1$. Therefore, there are no significant differences between the ℓ_1 penalty models and the maximum likelihood model.

Table Table B.7 and Figure Figure 3.16 show a sequence of q and corresponding KL loss. Tables Table 3.11 and Table 3.12 show a sequence of λ and corresponding KL loss for the Lasso and ALasso, and Figure Figure 3.17 shows a sequence of λ and corresponding KL loss for the Lasso and ALasso. Table Table 3.13 illustrates the estimates of the factor loadings and uniqueness from the Lasso (left) and the ALasso (right) with penalty 3 and 1 respectively. For the ALasso penalty, γ of 1 is used. An iteration converging gap of 0.0001 was utilized for β. When we compare the patterns between the two methods, 18 out of 23 variables have similar patterns, although 5 variables (variable 1, 2, 13, 18, 19) are dissimilar.

Table 3.10: $q = 8$ gives the best MLE model for Parkinson's disease data.

q	...	5	6	7	8	9	10	...
KL	...	16.625	15.748	15.500	15.402	15.502	15.509	...

Table 3.11: $\lambda = 3$ gives the best Lasso model for Parkinson's disease data.

λ	0	1	2	3	4	5
KL	14.886	14.878	14.871	14.869	14.874	14.890

Table 3.12: $\lambda = 1$ gives the best ALasso model for Parkinson's disease data.

λ	0	1	2	3	4
KL	14.886	14.878	14.966	15.035	15.120

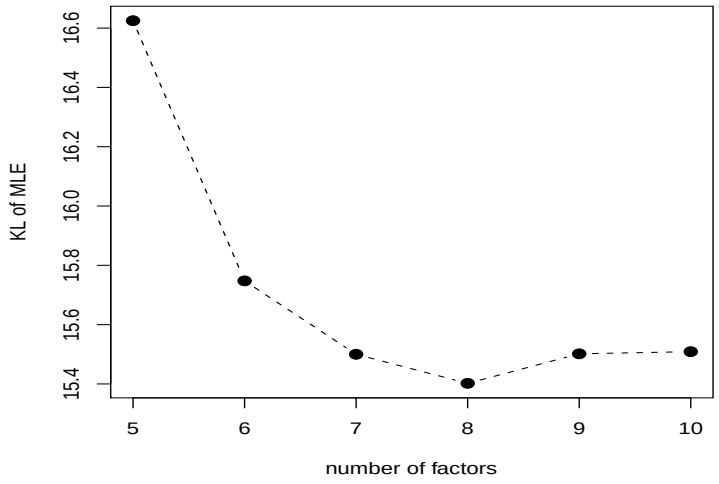

Figure 3.16: Parkinsons data: q = 8 gives the best MLE model.

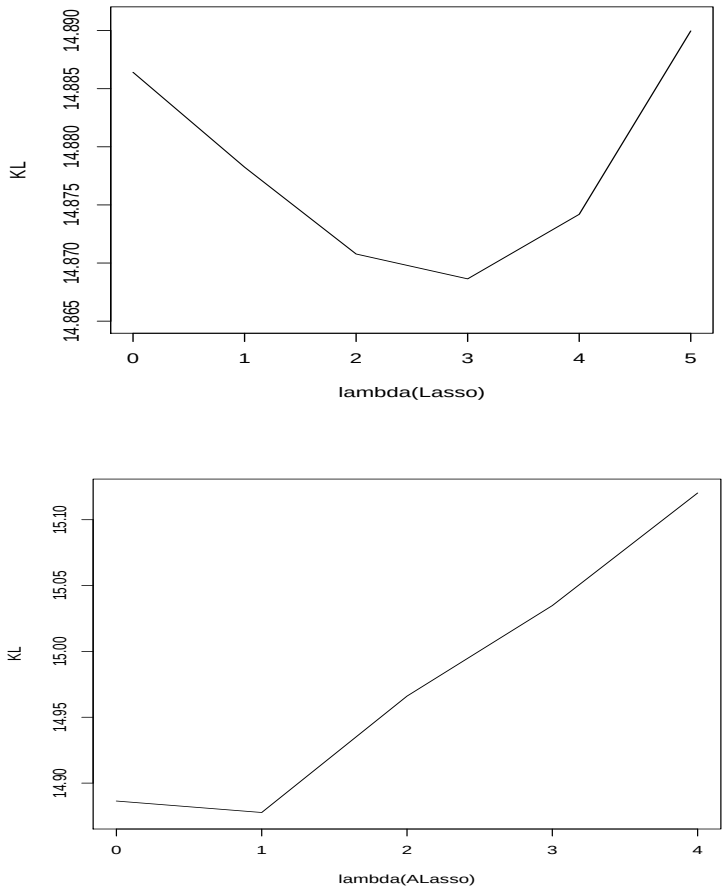

Figure 3.17: Parkinsons data: KL loss vs. λ: q = 8

Table 3.13: The estimates of factor loadings and uniqueness of Parkinson's disease data from Lasso (left) and ALasso (right) with penalty 4 and 1 respectively. For ALasso penalty, γ of 1 is used. Iteration converging gap of 0.0001 was used for β

	F1	F2	F3	...	F8	$\hat{\tau}^2$	F1	F2	F3	...	F8	$\hat{\tau}^2$
V1	0.05	0.59	0.00	...	-0.02	0.01	0.00	0.19	0.00	...	0.11	0.01
V2	0.00	0.00	-0.32	...	0.20	0.64	-0.23	-0.28	0.00	...	0.36	0.65
V3	-0.14	0.52	0.00	...	-0.03	0.51	-0.11	0.27	0.00	...	0.00	0.51
V4	-0.02	0.15	-0.03	...	0.84	0.01	-0.02	0.19	0.01	...	0.88	0.01
V5	0.07	-0.01	-0.11	...	0.71	0.03	0.03	0.08	-0.10	...	0.74	0.03
V6	-0.00	0.20	-0.00	...	0.82	0.01	-0.00	0.25	0.00	...	0.87	0.01
V7	-0.03	0.20	-0.00	...	0.89	0.01	-0.00	0.18	0.01	...	0.90	0.01
V8	-0.00	0.20	-0.00	...	0.82	0.01	-0.00	0.25	0.00	...	0.87	0.01
V9	0.33	-0.02	0.24	...	0.79	0.01	0.43	0.02	0.30	...	0.74	0.01
V10	0.22	-0.00	0.22	...	0.85	0.01	0.32	0.03	0.33	...	0.80	0.01
V11	0.43	-0.03	0.20	...	0.75	0.01	0.51	0.00	0.23	...	0.72	0.01
V12	0.31	0.04	0.24	...	0.78	0.01	0.43	0.03	0.30	...	0.71	0.01
V13	0.06	-0.01	0.34	...	0.83	0.01	0.24	0.11	0.46	...	0.73	0.01
V14	0.43	-0.03	0.20	...	0.75	0.01	0.51	0.00	0.23	...	0.73	0.01
V15	-0.00	0.05	0.04	...	0.74	0.08	0.00	0.15	0.10	...	0.79	0.08
V16	-0.24	-0.00	-0.19	...	-0.64	0.16	-0.38	-0.06	-0.17	...	-0.61	0.16
V17	0.17	-0.13	0.24	...	0.16	0.57	0.15	0.00	0.15	...	0.09	0.58
V18	0.00	-0.39	0.00	...	0.24	0.01	0.32	-0.13	0.00	...	0.15	0.01
V19	0.00	0.02	0.00	...	0.16	0.40	0.00	0.02	0.00	...	0.02	0.41
V20	0.13	-0.06	0.03	...	0.50	0.03	0.12	0.00	0.00	...	0.48	0.03
V21	0.00	-0.26	0.02	...	0.33	0.38	0.00	-0.23	0.20	...	0.28	0.38
V22	0.10	-0.01	0.30	...	0.32	0.35	0.01	0.00	0.36	...	0.34	0.35
V23	0.13	0.00	0.00	...	0.56	0.02	0.13	0.00	0.00	...	0.53	0.02

3.5.4 Image Segmentation

Image segmentation data were created by Vision Group at the University of Massachusetts in November 1990, and are stored in UCI Machine Learning Repository (Asuncion and Newman, 2007). The data consist of 7 outdoor images with training data from 2100 observations. The images were hand segmented, so that a classification for every pixel could be created. Each instance is a 3×3 region. There are 19 variables with all real numbers, but the data on the third column (region−pixel−count: the number of pixels in a region = 9) are all the same number: consequently, column is excluded. We determine q and λ for each of the 7 images according to the classes of brickface, sky, foliage, cement, window, path, and grass. Each image consists of data from 300 instances. Two thirds of each data set are randomly selected for training data, and the remaining one third of each dataset is used for validation data.

As a measurement of performance, KL loss is utilized for this data to determine the appropriate number of q and λ.

The first image data is brickface. From Tables Table 3.14 and Table 3.15 and Figure Figure 3.18 we determined that the appropriate number of q is 7 and the appropriate λ is 4. The smallest KL loss by the Lasso model is 55.792, whereas the KL loss of the MLE model is 55.815. The difference is so small that we use the sparsity-first rule. The rule chooses a lasso model with 30 zero loadings when $\lambda = 14$.

Then, the ℓ_1 penalty model is tested if it is within the confidence interval of the maximum likelihood model at the 0.01 level. By performing the likelihood ratio test in (1.4), we attain a p-value of 0.99 for the Lasso model with $\lambda = 14$. Therefore, there is no significant difference between the lasso model and maximum likelihood model.

The second image data is sky. In this dataset, the column 5 (short-line-density-2) is excluded in addition to the column 3 because the covariance matrix cannot be computed with the column 5. For the sky image data, KL loss cannot be computed because the determinant of the covariance matrix is 0 so we utilize negative

Table 3.14: $q = 7$ gives the best MLE model for brickface image data.

q	...	4	5	6	7	8	9	...
KL	...	58.89	58.63	58.60	58.54	58.56	58.63	...

Table 3.15: $\lambda = 1$ gives the best Lasso model for brickface image data.

λ	0	2	4	6	8	10	12	14	...
KL	55.815	55.803	55.792	55.798	55.801	55.807	55.809	55.813	...

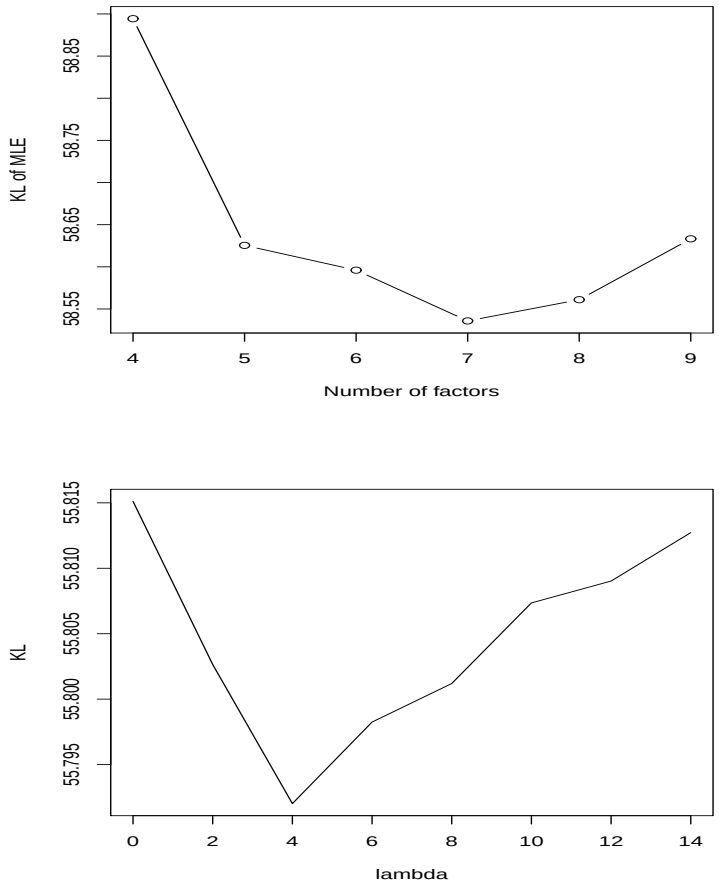

Figure 3.18: brickface: KL loss vs. number of factors and lambda

log-likelihood (NLL) which is equivalent to KL loss. The accuracy of the model is measured by its NLL evaluated on the validation set. From Tables Table 3.16 and Table 3.17 and Figure Figure 3.19 we determine the appropriate number of q is 7 and the appropriate λ is 3. The smallest NLL is -2506 while the NLL of the MLE model is -2503. The sparsity-first rule chooses a Lasso model with 32 zero loadings when $\lambda = 16$.

Then, the ℓ_1 penalty model is tested whether it is within the confidence interval of the maximum likelihood model at the 0.01 level. By performing the likelihood ratio test, we attain a p-value of 0.99 for the Lasso model with $\lambda = 16$. Therefore, there is no difference between the Lasso model and the maximum likelihood model.

Table 3.16: $q = 7$ gives the best MLE model for sky image data.

q	...	5	6	7	8	9	...
NLL	...	-2218	-2235	-2254	-2243	-2241	...

Table 3.17: $\lambda = 3$ gives the best Lasso model for sky image data.

λ	0	1	2	3	4	5	6
NLL	-2503	-2503	-2505	-2506	-2505	-2504	-2504

The third image data is foliage. The determinant of the covariance is too small to compute KL loss. Therefore, NLL is used instead of KL loss. From Tables Table 3.18 and Table 3.19 and Figure Figure 3.20 we determine the appropriate number of q is 8 and the appropriate λ is 10. The smallest NLL of the Lasso model is -2250, whereas the NLL of the MLE model is -2236. A Lasso model has 45 zero loadings at λ of 10.

Then, the ℓ_1 penalty model is tested if it is within the confidence interval of the maximum likelihood model at the 0.01 level. By performing the likelihood ratio test,

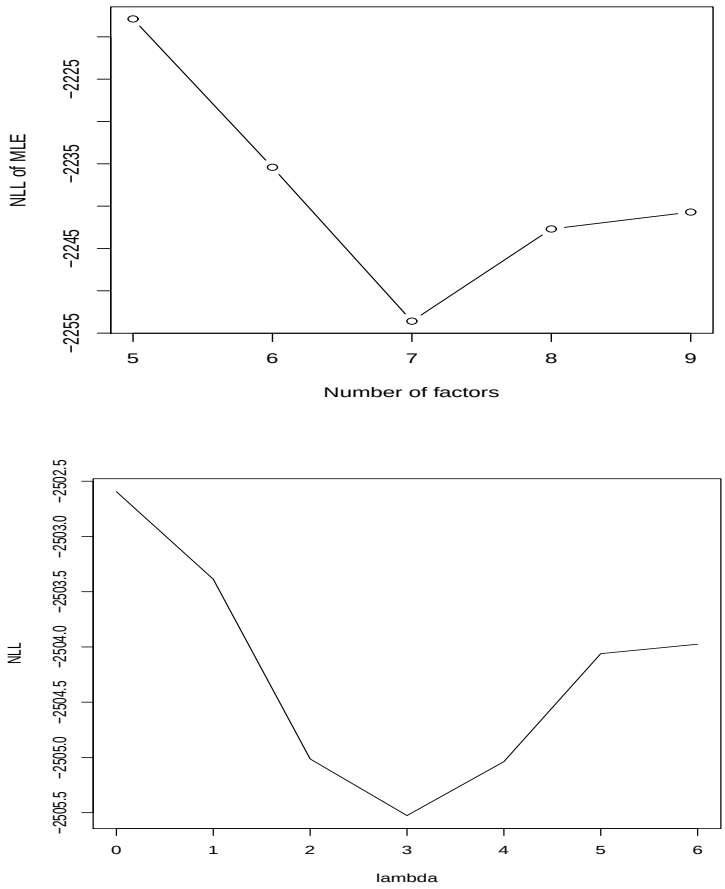

Figure 3.19: sky image: $\lambda = 3$ gives the best Lasso model.

we attain a p-value of 0.99 for the Lasso model with $\lambda = 14$. Therefore, there is no difference between the Lasso model and the maximum likelihood model.

Table 3.18: $q = 8$ gives the best MLE model for foliage image data.

q	...	6	7	8	9	10	...
NLL	...	-2180	-2236	-2240	-2237	-2237	...

Table 3.19: $\lambda = 10$ gives the best Lasso model for foliage image data.

λ	...	7	8	9	10	11	12	...
NLL	...	-2248	-2249	-2249	-2250	-2250	-2249	...

The fourth image data is cement. We fit the Lasso factor model by estimating KL loss. From Tables Table 3.20 and Table 3.21 and Figure Figure 3.21 we determine the appropriate number of q is 3. The appropriate λ of 13 is obtained estimating KL loss for a grid of penalization parameter. The smallest KL loss by the Lasso model is 62.3196, whereas the KL loss of the MLE model is 62.3525. A Lasso model has 5 zero loadings at the λ of 13.

Then, the ℓ_1 penalty model is tested if it is within the confidence interval of the maximum likelihood model at the 0.01 level. By performing the likelihood ratio test, we attain a p-value of 0.99 for the Lasso model with $\lambda = 13$. Therefore, there is no difference between the Lasso model and the maximum likelihood model.

The fifth image data is window. The accuracy of the model is measured by its KL loss evaluated on the validation set. From Tables Table 3.22 and Table 3.23 and Figure Figure 3.22 we determine the appropriate number of q is 7 and the appropriate

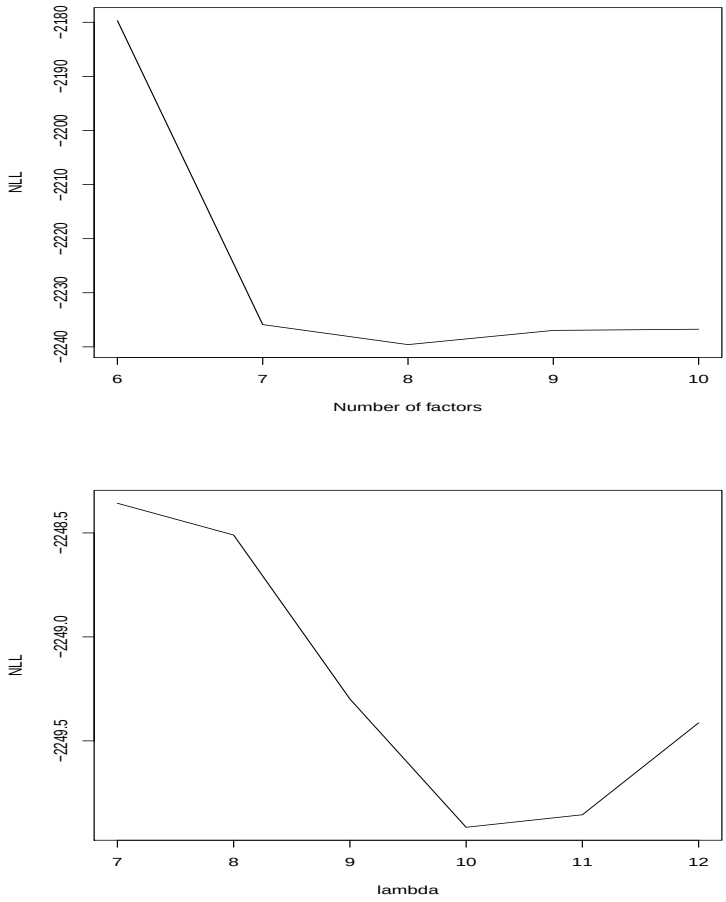

Figure 3.20: foliage image: NLL vs. number of factors and lambda

Table 3.20: $q = 3$ gives the best MLE model for cement image data.

q	1	2	3	4	5	6	...
KL	71.724	64.626	63.811	63.851	64.127	64.128	...

Table 3.21: $\lambda = 13$ gives the best Lasso model for cement image data.

λ	...	11	12	13	14	15	16	...
KL	...	62.3203	62.3198	62.3196	62.3197	62.3199	62.3203	...

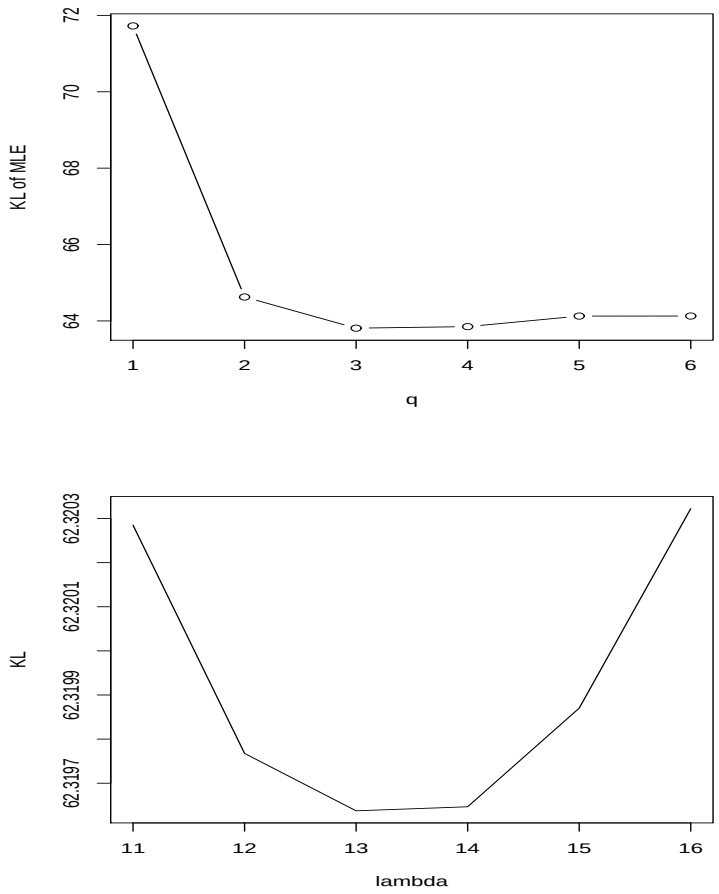

Figure 3.21: cement image: KL loss vs. number of factors and lambda

λ is 29. The smallest KL loss by the Lasso model is 60.425, whereas the KL loss of the MLE model is 60.591. We choose a Lasso model with 44 exact zero loadings at the λ of 29.

Then, the ℓ_1 penalty model is tested if it is within the confidence interval of the maximum likelihood model at the 0.01 level. By performing the likelihood ratio test, we attain a p-value of 0.04 for the Lasso model with $\lambda = 29$. The p-value is quite low but the model is still within the confidence interval of the maximum likelihood model at the 0.01 level.

Table 3.22: $q = 7$ gives the best MLE model for window image data.

q	...	5	6	7	8	9	...
KL	...	65.737	63.735	63.667	63.847	63.971	...

Table 3.23: $\lambda = 29$ gives the best Lasso model for window image data.

λ	...	25	26	27	28	29	30	...
KL	...	60.430	60.429	60.427	60.426	60.425	60.478	...

The sixth image data is path. From Tables Table 3.24 and Table 3.25 and Figure Figure 3.23 we determine the appropriate number of q is 8 and the appropriate λ is 12. The smallest KL loss by the Lasso model is 54.231, whereas the KL loss of the MLE model is 54.315. Therefore, we use the sparsity-first rule and choose a Lasso model with 61 zero loadings at the λ of 31.

Then, the ℓ_1 penalty model is tested if it is within the confidence interval of the maximum likelihood model at the 0.01 level. By performing the likelihood ratio test, we attain a p-value of less than 0.01 for the Lasso model with $\lambda = 31$. The p-value is

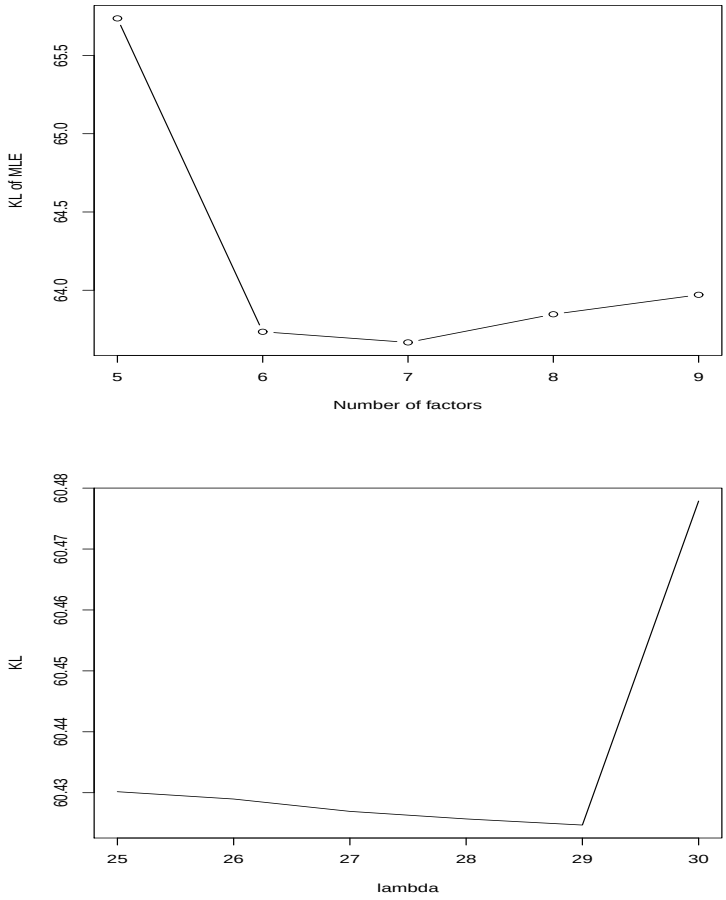

Figure 3.22: window image: KL loss vs. number of factors and lambda

very low and the model is not within the confidence interval of the maximum likelihood model at the 0.01 level.

Table 3.24: $q = 8$ give the best MLE model for path image data.

q	...	3	4	5	6	7	8	9	10	...
KL	...	54.428	54.132	54.048	53.902	53.804	53.795	53.817	53.822	...

Table 3.25: $\lambda = 12$ gives the best Lasso model for path image data.

λ	...	8	10	12	14	16	...
KL	...	54.240	54.238	54.231	54.235	54.245	...

The last image data is grass. From Tables Table 3.26 and Table 3.27 and Figure Figure 3.24 we determine the appropriate number of q is 5 and the appropriate λ is 38. The smallest KL loss by the Lasso model is 57.476, whereas the KL loss of the MLE model is 57.839. Therefore, we choose a Lasso model with 27 exact zero loadings at the λ of 38.

Then, the ℓ_1 penalty model is tested if it is within the confidence interval of the maximum likelihood model at the 0.01 level. By performing the likelihood ratio test, we attain a p-value of 0.60 for the Lasso model with $\lambda = 38$. The p-value is high. Therefore, the model is within the confidence interval of the maximum likelihood model at the 0.01 level.

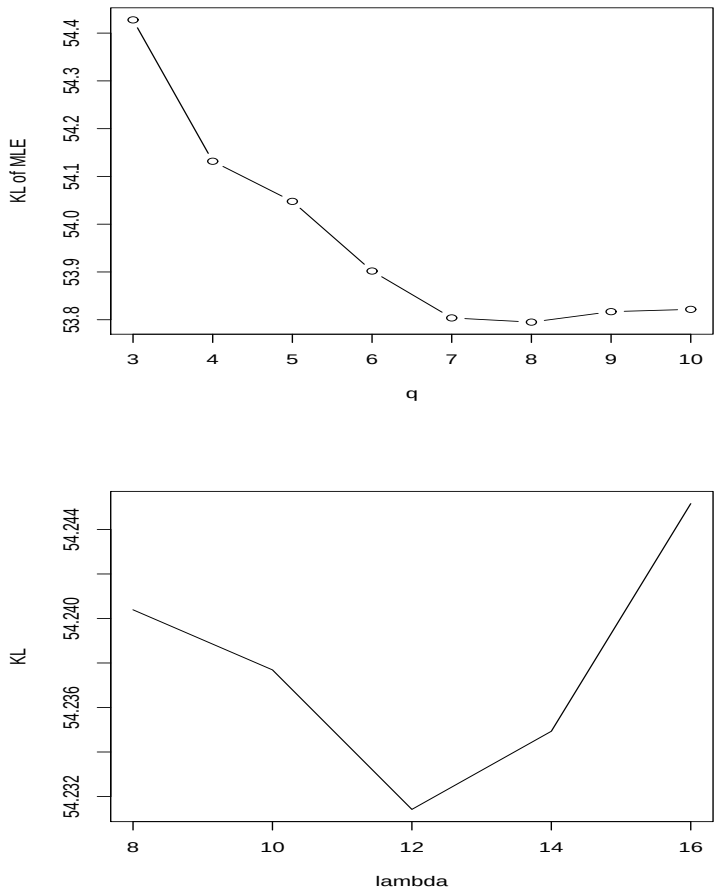

Figure 3.23: path image: KL loss vs. number of factors and lambda

Table 3.26: $q = 5$ gives the best MLE model for grass image data.

q	1	2	3	4	5	6	7	8	...
KL	64.673	61.674	56.780	56.523	56.258	56.707	56.515	56.574	...

Table 3.27: $\lambda = 38$ gives the best Lasso model for grass image

λ	...	36	38	40	42	44	46	48	50	...
KL	...	57.747	57.476	57.560	57.562	57.564	57.566	57.567	57.569	...

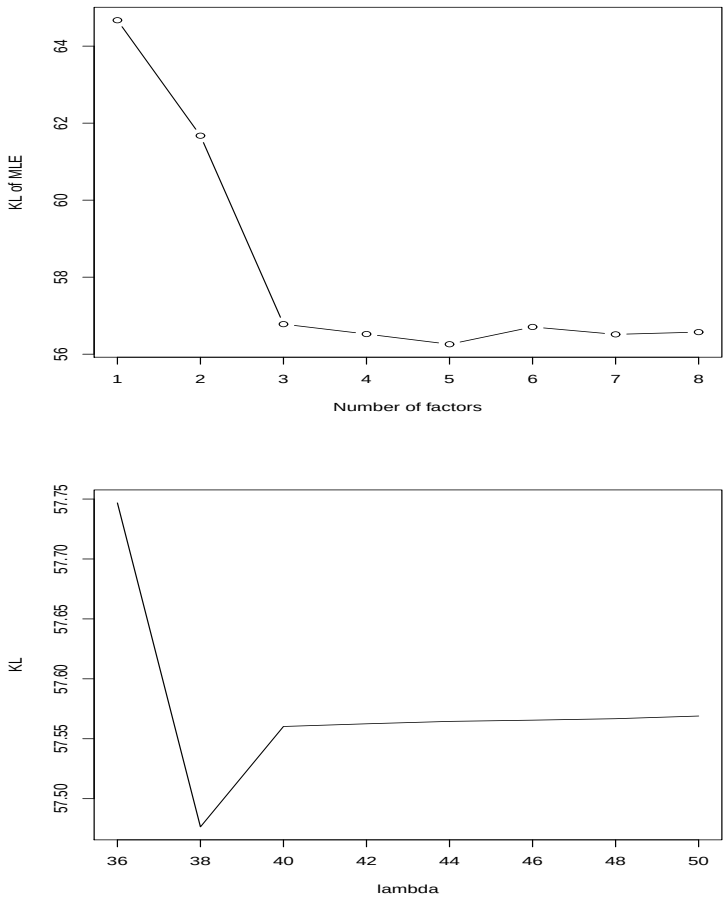

Figure 3.24: grass image: KL loss vs. number of factors and lambda

Chapter 4

Conclusions and Future Research

4.1 Conclusions

Factor models are most interpretable when the factor loadings are sparse (i.e., when most of the factor loadings are zero). The Lasso is a technique that can successfully estimate sparse models in regression and related contexts. We adapt the MLE of the factor model to include ℓ_1 penalties and, consequently achieve sparse factor loadings. The maximum likelihood method does not usually produce a sparse model even when the true factor loadings are sparse. However, by including the ℓ_1 penalties, the ability of recognizing zeros is greatly improved. This improvement can be measured by KL loss, where the penalized estimates show lower K-L loss than the standard estimates.

We derive an efficient algorithm by adapting the expectation-maximization (EM) algorithm, which is equivalent to iterative ℓ_1-penalized least squares, for maximizing the objective function to include a ℓ_1 penalty function. The modified EM algorithm increases the penalized likelihood at each iteration and converges to the unique optimum.

We have observed that if the data are generated from a truly sparse factor model, the ℓ_1 penalized models not only discover zero loadings, but also are significantly more accurate than the MLE model. One advantage of ℓ_1 penalized models is that the effort

to find the interpretable form by rotation is not necessary because ℓ_1 penalized models are not rotation invariant and produce sparse models that are easy to interpret. The ALasso model performs better than the Lasso model. The ALasso model discovers zero loadings more accurately than the Lasso model and performs similarly to the oracle MLE. The simulations show that the average number of zeros discovered by the Lasso model is quite below the number of true zeros, but the average number of zeros discovered by the ALasso model is close to the number of true zeros. The rotation method cannot discover any exact zeros, so it is not a good method for sparse factor loadings. However, if properly truncated, it shows quite good performance.

The likelihood ratio test shows that most of the likelihoods of the real data examples analyzed by the ℓ_1 penalized model are within the confidence interval of the maximum likelihood. Further, many cases are observed equivalent to the maximum likelihood because they have very high p-values.

In some applications, ℓ_1 penalization only slightly improves the accuracy of the MLE model. In such situations we suggest the use of the sparsity-first rule to select the optimal penalization parameter in ℓ_1 penalized models because ℓ_1 penalization is primarily used to pursue sparsity.

The LL and PLL have increasing property in the EM algorithm. In addition, the EM algorithm always converges to the point where β and τ^2 can be obtained. The iteration required for convergence is much less in ALasso than in Lasso.

4.2 Future Research

4.2.1 Extension to Other Factor Models and Analysis Methods

In this thesis, we applied the Lasso and ALasso penalties to the maximum likelihood method for an orthogonal factor model. Although we gained important insights and discovered ways of determining the number of factors and λ, future research is warrented. There are many factor analysis methods in addition to the maximum likelihood method, such as principal factor method, Minres method, canonical factor analysis, image factor analysis, and multiple-group methods. We do not know if we can use the Lasso and ALasso penalties for all of these factor analysis methods, but we need to try them to both find out if they are applicable and compare them with the existing methods. Further, there are other models in addition to the orthogonal factor model. The model is simple because factors are not correlated with one another. However, we need to apply to the oblique factor model in which the factors are correlated. Problems might arise due to the correlations, but it would be worth investigating. Another model is the Bayesian model. It is not clear whether a lasso-type penalty can be included in a Bayesian factor model because this type of penalty can be replaced by a prior distribution. However, we might develop a new method of estimation by applying a lasso-type penalty to a Gibbs sampling algorithm. We have not tried other penalty models such as SCAD and Elastic Net. We need to consider these penalties and compare them with Lasso and ALasso penalties in terms of accuracy and computation cost.

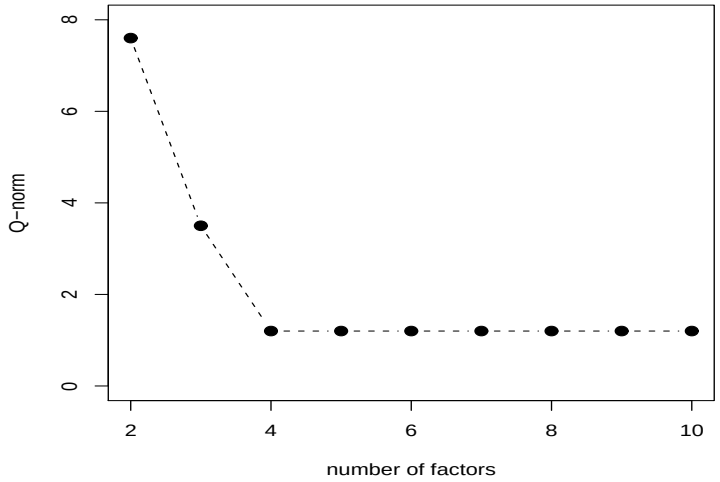

Figure 4.1: Q-norm vs. number of factors.

4.2.2 Measuring instruments

In this thesis, we used KL loss to measure the appropriate number of factors and λ. We have also attempted Q-norm as a measurement though it is not shown in this thesis. Q-norm is defined as

$$Q - norm = [tr(\hat{\Sigma}\Sigma^{-1} - I)^2]^{1/2}$$

where $\hat{\Sigma}$ is a sample covariance matrix and Σ is an estimated covariance matrix, and I is an identity matrix. Choosing the appropriate number of factors from Q-norm can be determined by plot. A typical plot of Q-norm vs. number of factors from the simulations are shown in Figure Figure 4.1.

However, Q-norm plots in real data do not look so good as in the Q-norm plots in the simulations and hard to determine practically the exact bend point because the bend point is not a minimum point while KL loss (or NLL) is easy to determine the accurate point because we can choose the minimum point. Additionally, the bend

point on Q-norm and minimum point on KL loss have not shown consistency in the real data examples though they are consistent in the simulation data. Therefore, exploring a way of choosing the exact point on Q-norm curve and the reason of inconsistency between Q-norm and KL loss is a work we need to perform in future research.

4.2.3 Confidence Level

The likelihood ratio tests on the simulation data show that the lasso penalized factor model is as good as the maximum likelihood model and the ALasso model is within the confidence level of the MLE model. The tests on the real data show that most of ℓ_1 penalized models are also within the confidence interval of the maximum likelihood model. However, in some cases, p-values are very low and one real data case has a p-value of less than 0.01. We do not know why this case does not fit to the maximum likelihood model when most of the other cases did. We need to perform tests on more data and investigate the reason.

References

Asuncion, A. and Newman, D. (2007). Uci machine learning repository [http://www.ics.uci.edu/ mlearn/mlrepository.html]. *University of California, School of Information and Computer Science*.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm (with discussion). *Journal of the Royal Statistical Society, series B*, 39:1–38.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2003). Least angle regression. *The Annals of Statistics*, 32:407–499.

Fan, J., Fan, Y., and Lv, J. (2008). High dimensional covariance matrix estimation using a factor mode. *Journal of Econometrics*, 147:186–197.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96:1348–1360.

Fan, J. and Lv, J. (20101). A selective overview of variable selection in high dimensional feature space. *Statistica Sinica*, 20:101–148.

Fan, J. and Peng, H. (2004). "on nonconcave penalized likelihood with diverging number of parameters,". *The Annals of Statistics*, 32:928–961.

Garnett, J. C. M. (1919). On certain independent factors in mental measurement. *Proc. Roy. Soc.Lon.*, 96:91–111.

Gorsuch, R. (1983). Factor analysis. *Hillsdale, NJ: Lawrence Erlbaum.*

Harman, H. H. (1976). *Modern Factor Analysis.* The University of Chicago Press, Chicago 60637.

Harris, C. (1962). Some rao-guttman relationships. *Psych.*, 27:247–63.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning.* Springer.

Hesterberg, T., Choi, N., Meier, L., and Fraley, C. (2008). Least angle and ℓ_1 penalized regression: A review. *Statistical Survery*, 2:61–93.

Holzinger, K. J. and Swineford, F. A. (1939). A study in factor analysis: The stability of a bi-factor solution. *Supplementary Education Monographs; University of Chicago*, 48.

Johnson, R. and Wichern, D. (2007). *Applied Multivariate Statistical Analysis.* New Jersey: Pearson Education, Inc.

Kaiser, H. (1958). The varimax criterion for analytic rotation in factor analysis. *Psychometrika*, 23(3).

Kelley, T. (1940). Talents and tasks: Their conjunction in a democracy for wholesome living and national defense. *Harvard education papers: Harvard University, Graduate School of Education*, (1).

Lawson, C. and Hansen, R. (1974). Solving least squares problems. *Englewood Cliffs: Prentice Hall.*

Little, M., McSharry, P., Hunter, E., and Ramig, L. (2008). Suitability of dysphonia measurements for telemonitoring of parkinson's disease. *IEEE Transactions on Biomedical Engineering*, 56:1015–1022.

Meinshausen, N. and Bühlmann, P. (2004). "variable selection and high dimensional graphs with the lasso,'. *technical report, ETH Zürich*.

Neuhaus, J. and Wrigley, C. (1954). The quartimax method: An analytical approach to orthogonal simple structure.

Osborne, M., Presnell, B., and Turlach, B. (2000). On the lasso and its dual. *Journal of Computational and Graphical Statistics*, 9(2):319–337.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Phil. Mag.*, 2(6):559–72.

Penrose, K., Nelson, A., and Fisher, A. (1985). Generalized body composition prediction equation for men using simple measurement techniques. *Medicine and Science in Sports and Exercise*, 17(2):189.

Rubin, D. and Thayer, D. (1982). Em algorithms for ml factor analysis. *Psychometrika*, 47:69–76.

Saunders, D. (1962). Trans-varimax. *Psych.*, 17:395.

Spearman, C. (1904). General intelligence, objectively determined and measured. *The American Journal of Psychology*, 15:201–293.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, series B*, 58:265–288.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101:1418–1429.

Zou, H., Hastie, T., and Tibshirani, R. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15:265–286.

Appendix A

5-Fold Cross Validation

We repeat the simulations but used 5-fold cross validation. We use the model 1 and 2, 3, and a new model 4. The simulation model 1 is generated by taking i.i.d. random vectors Y_i of length 12 from normal distribution with mean 0 and covariance $\boldsymbol{\Sigma} = \boldsymbol{\beta}^T\boldsymbol{\beta} + \boldsymbol{\tau}^2$ where

$$\boldsymbol{\beta} = \begin{bmatrix} 1.8 & 1.8 & 1.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.6 & 1.6 & 1.6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.7 & 1.7 & 1.7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.5 & 1.5 & 1.5 \end{bmatrix},$$

$$\boldsymbol{\tau}^2 = \text{diag}(0.50, 0.13, 0.08, 0.89, 0.12, 0.32, 0.58, 0.71, 0.83, 0.36, 0.09, 0.10),$$

and the second simulation model is generated where

$$\boldsymbol{\beta} = \begin{bmatrix} 1.5 & 0 & 0 & 0 & 1.5 & 0 & 0 & 0 & 1.5 & 0 & 0 & 0 \\ 0 & 1.7 & 0 & 0 & 0 & 1.7 & 0 & 0 & 0 & 1.7 & 0 & 0 \\ 0 & 0 & 1.6 & 0 & 0 & 0 & 1.6 & 0 & 0 & 0 & 1.6 & 0 \\ 0 & 0 & 0 & 1.8 & 0 & 0 & 0 & 1.8 & 0 & 0 & 0 & 1.8 \end{bmatrix},$$

$$\boldsymbol{\tau}^2 = \text{diag}(0.03, 0.77, 0.76, 0.99, 0.91, 0.89, 0.43, 0.51, 0.25, 0.05, 0.65, 0.43),$$

and the third simulation model is generated where

$$\boldsymbol{\beta} = \begin{bmatrix} 1.8 & 1.8 & 1.8 & 1.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.6 & 1.6 & 1.6 & 1.6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.7 & 1.7 & 1.7 & 1.7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.5 & 1.5 & 1.5 \end{bmatrix},$$

$$\boldsymbol{\tau}^2 = \text{diag}(0.50, 0.13, 0.08, 0.89, 0.12, 0.32, 0.58, 0.71, 0.83, 0.36, 0.09, 0.10).$$

The interpretation of the first factor model is that variables $3k-2$, $3k-1$, and $3k$ are random perturbations of factor k, $k = 1, 2, 3, 4$, and for the second factor model, variables k,$k+4$, and $k+8$ are random perturbations of factor k. The third factor model is the same as the first factor model except for the fourth, seventh, and tenth columns. The fourth simulation model is generated where

$$\boldsymbol{\beta} = \begin{bmatrix} -.25 & .00 & -.41 & .00 & .00 & -.22 & -.41 & -.58 & .00 & .00 & .00 & -.26 \\ .00 & .00 & .00 & .37 & .26 & .38 & -.22 & .00 & .00 & .21 & .70 & .66 \\ .00 & -.29 & .00 & .00 & -.46 & .00 & -.36 & .00 & -.35 & .53 & -.89 & .00 \\ .44 & .36 & .00 & .00 & 1.16 & .39 & .00 & .88 & .00 & .49 & .31 & .66 \end{bmatrix},$$

$$\boldsymbol{\tau}^2 = \text{diag}(0.022, 0.939, 0.224, 0.869, 0.089, 0.949, 1.136, 0.490, 0.699, 0.912, 0.691, 0.419).$$

Each element in $\boldsymbol{\beta}^T$ is generated from $N(0, 0.5)$ and curtailed at 0.2, and $\boldsymbol{\tau}^2$ is generated from $\mid N(0, 0.5) \mid$.

Within each of 30 replications we generate 200 data for the four simulations. Then we randomly split the data into five data sets and one of the five sets were used for validation data and the other four sets were grouped into one data set for training data. The process is then repeated 5 times, with each of the 5 data set used exactly once as the validation data. The 5 results can be averaged to produce

a single estimation. In these simulation studies we compared four methods as we did in section (Section 3.4): the varimax rotation (ordinary MLE), the Lasso and ALasso estimators and the oracle estimator.

We choose $q = 4$ factors for all of the true model 1,2, 3, and 4 from Figures Figure A.1 and Figure A.2.

From Figures Figure A.3 and Figure A.4 and the second column of Table Table A.1, it is very clear that, for models 1, 2, 3, both the Lasso and the ALasso estimators are more accurate than the MLE. Moreover, the ALasso is more accurate than the Lasso and is also very close to the oracle. Figures Figure A.5 and Figure A.6 display paired RKL values for the ALasso and oracle in the 30 replications. It is interesting to see that some out of 30 times the ALasso did slightly better than the oracle in the simulations. However, for the model 3, the performance is not as good as those of the model 1 and 2 because the true factor loadings have more than one nonzero entries on some columns of the true loadings matrix. Table Table A.1 illustrates that the RKLs of the model 3 are higher than the other two models, and the ability of recognizing zeros is not as good as those of the models 1 and 2. The model 4 is the worst among all models. the RKLs of the model 4 are higher than those of the other models, and the ability of recognizing zeros is not satisfied. Observed from these simulations, the performance of ℓ_1 penalized method might be best when the true factor loadings matrix has only one nonzero entry on each column and decreases when it has more than one nonzero entries on each column of the matrix.

From the third column of Table Table A.1 we see that the ALasso discovers more zero loadings than the Lasso did. To visualize their difference, we make the histogram of the number of estimated zero loadings for the Lasso and ALasso, as shown in Figures Figure A.7, Figure A.8, Figure A.9, and Figure A.10.

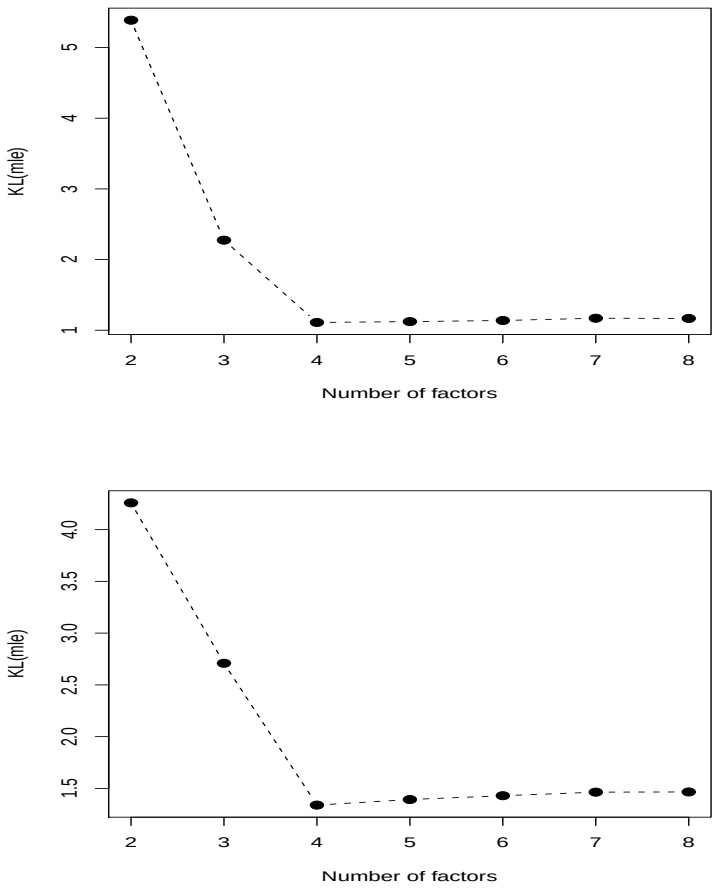

Figure A.1: The y-axis shows the value of $KL(\text{mle})_{\text{v}}$ for q factors; top: model 1; bottom: model 2.

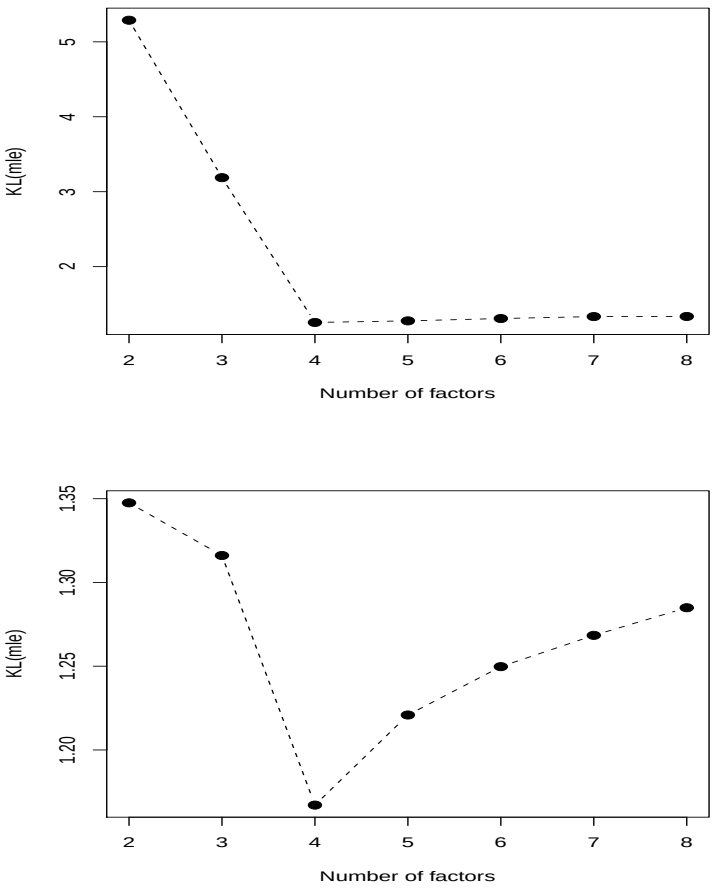

Figure A.2: The y-axis shows the value of $KL(\text{mle})_{\text{v}}$ for q factors; top: model 3; bottom: model 4.

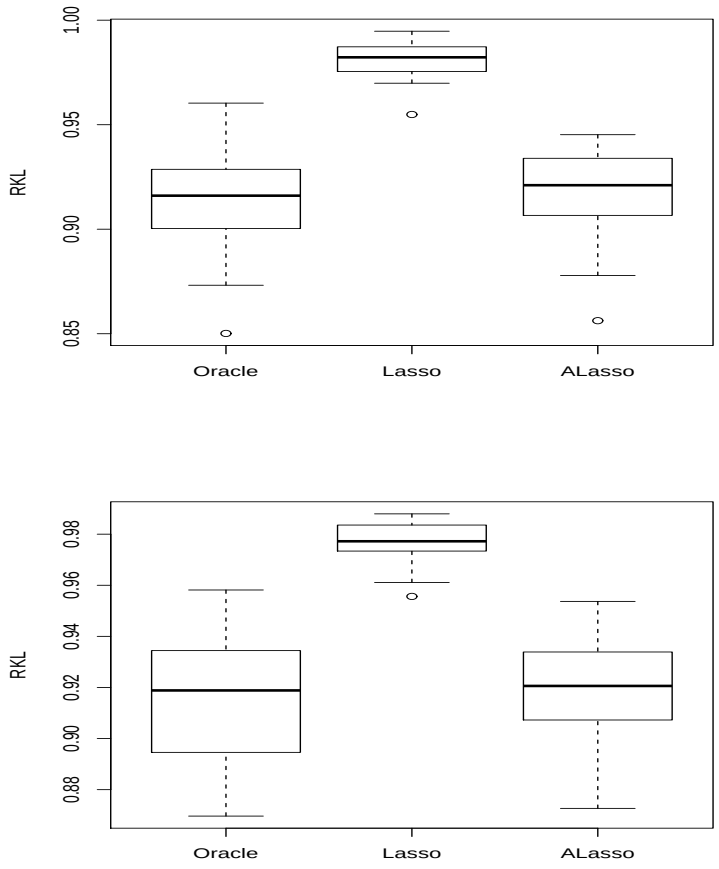

Figure A.3: Boxplots of RKLs of the Lasso, ALasso and oracleestimators with respect to the MLE (top: model 1; bottom: model 2).

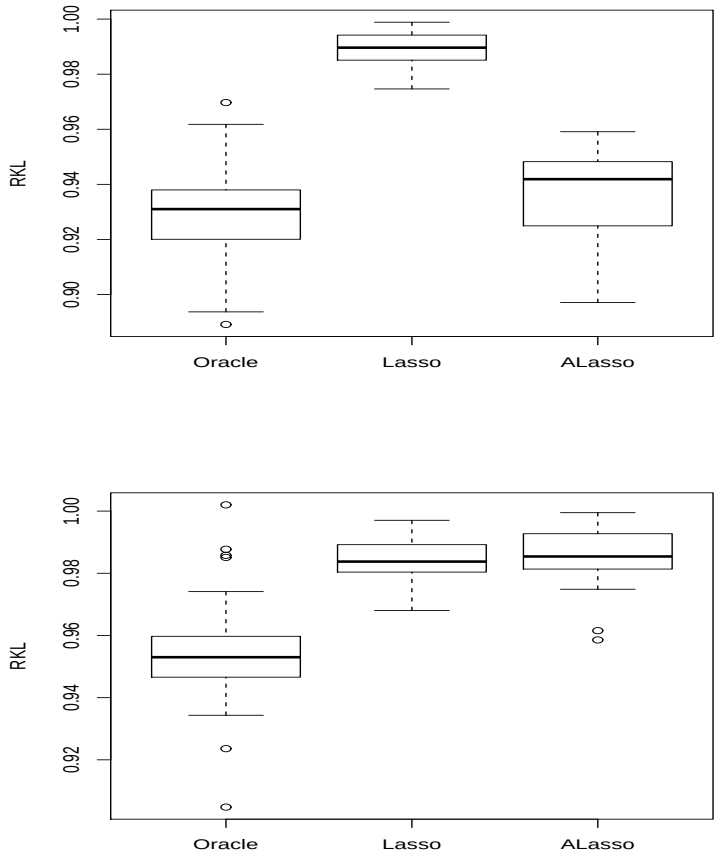

Figure A.4: Boxplots of RKLs of the Lasso, ALasso and oracleestimators with respect to the MLE (top: model 3; bottom: model 4).

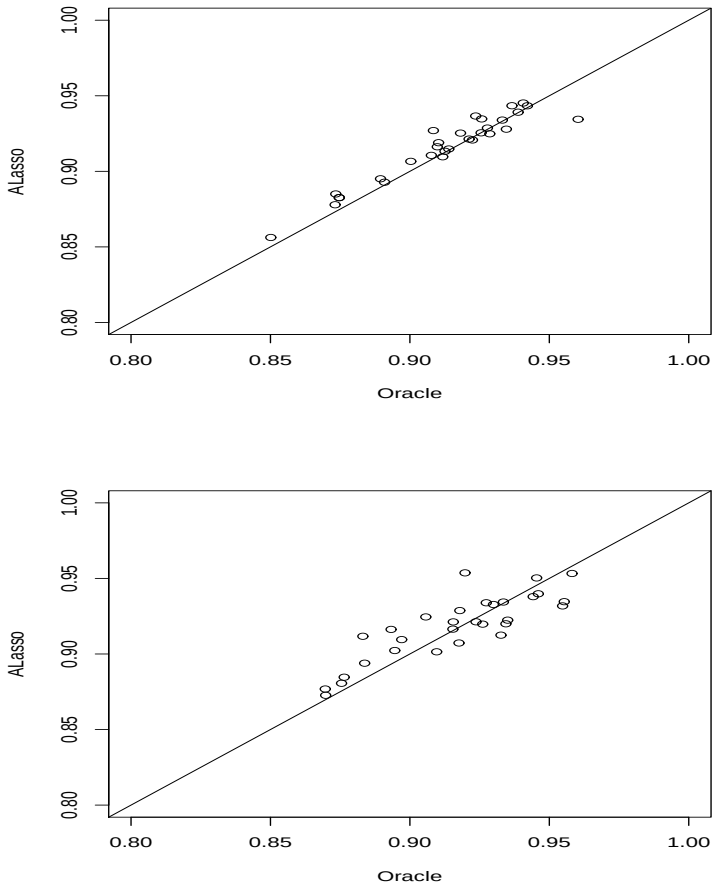

Figure A.5: Pairwise comparison of ALasso and oracle. Thesolid straight line is the 45 degree line (top: model 1; bottom: model 2).

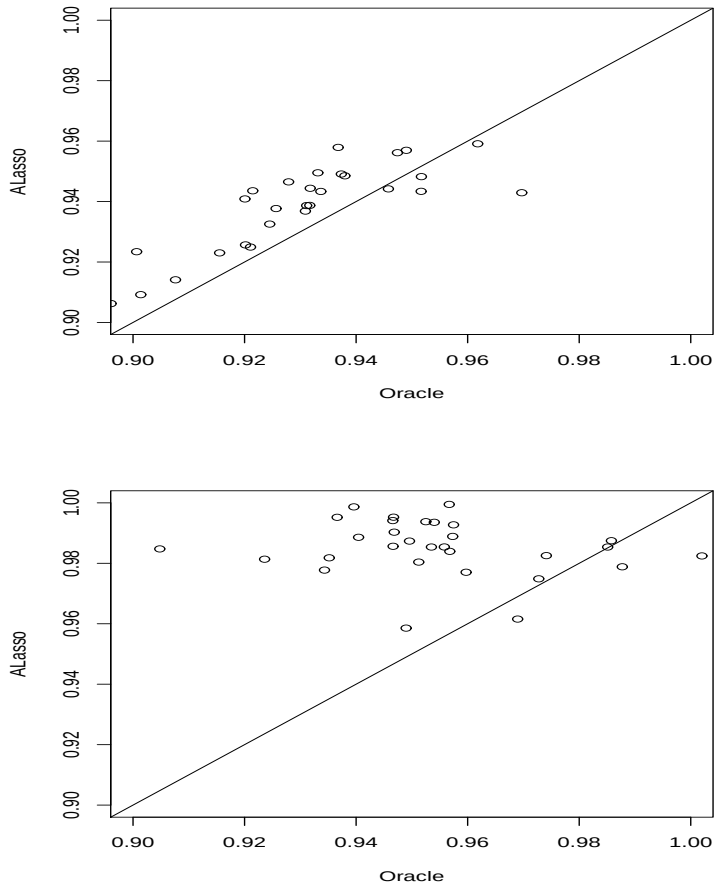

Figure A.6: Pairwise comparison of ALasso and oracle. Thesolid straight line is the 45 degree line (top: model 3; bottom: model 4).

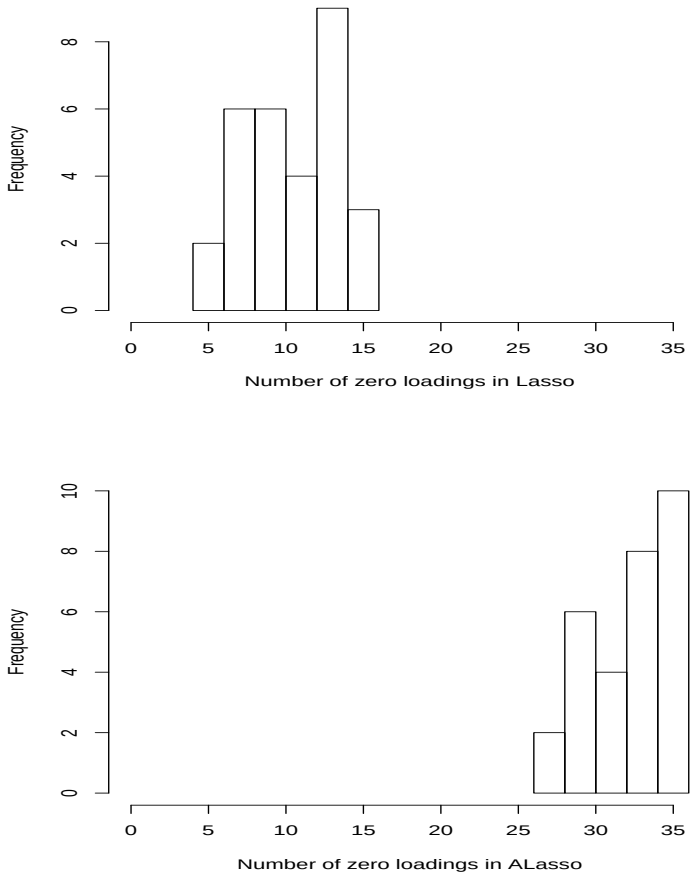

Figure A.7: Comparing the sparsity pursuit performance ofLasso and ALasso of model 1 (top: Lasso, bottom: ALasso).

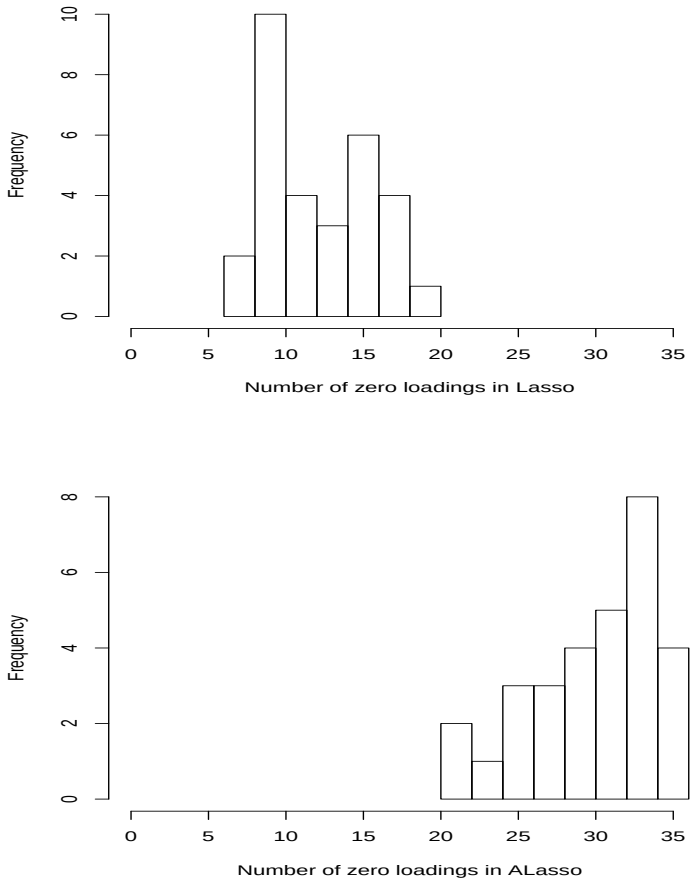

Figure A.8: Comparing the sparsity pursuit performance ofLasso and ALasso of model 2 (top: Lasso, bottom: ALasso).

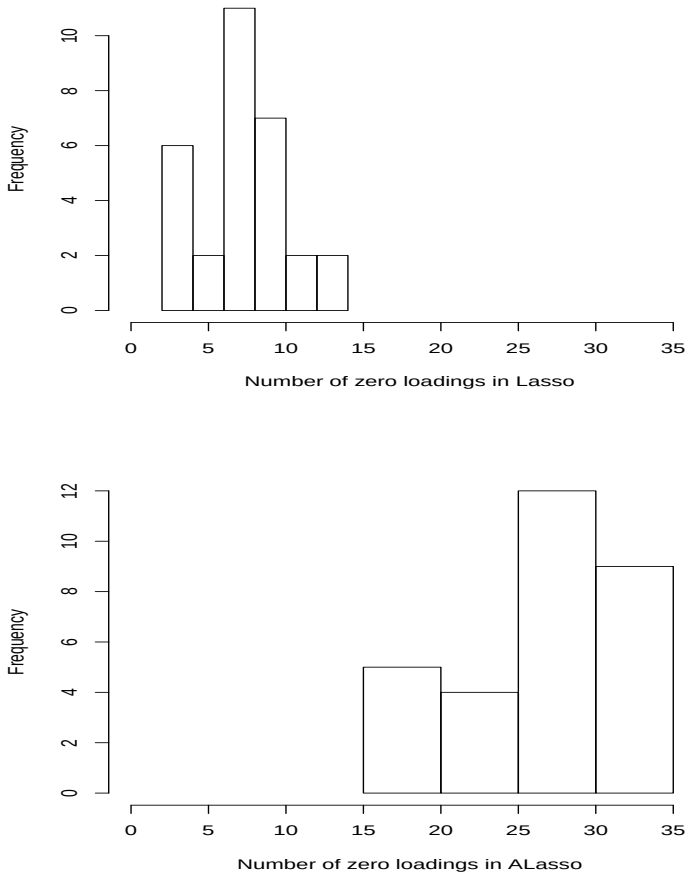

Figure A.9: Comparing the sparsity pursuit performance ofLasso and ALasso of model 3 (top: Lasso, bottom: ALasso).

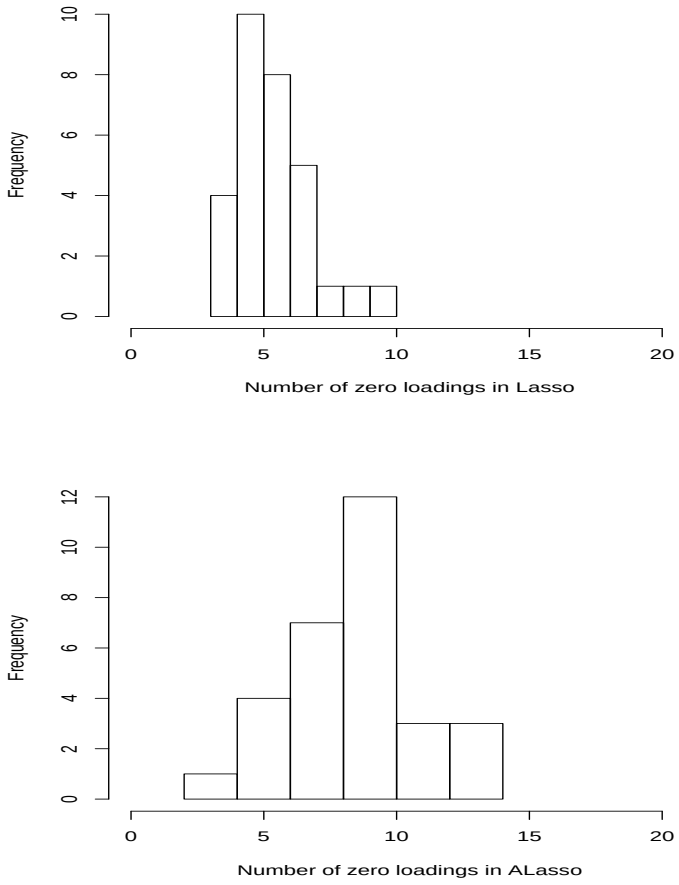

Figure A.10: Comparing the sparsity pursuit performance ofLasso and ALasso of model 4 (top: Lasso, bottom: ALasso).

model1	RKL	Number of zeros
Oracle	0.913 (0.025)	36
Lasso	0.981 (0.008)	11 (3.02)
ALasso	0.916 (0.022)	33 (2.78)
model2	RKL	Number of zeros
Oracle	0.916 (0.026)	36
Lasso	0.978 (0.008)	12 (3.40)
ALasso	0.918 (0.022)	30 (4.24)
model3	RKL	Number of zeros
Oracle	0.928 (0.020)	33
Lasso	0.989 (0.006)	7 (2.95)
ALasso	0.936 (0.016)	27 (4.95)
model4	RKL	Number of zeros
Oracle	0.954 (0.020)	21
Lasso	0.983 (0.007)	6 (1.49)
ALasso	0.985 (0.009)	9 (2.53)

Table A.1: Averages based on 30 replications(from top to bottom: model 1, 2, 3, 4). Numbers in $(\cdot)$ are standard errors.

Appendix B

Q-norm Measurement

B.1 Simulations by Q-norm

In factor analysis, the residual covariance matrix which is the difference between sample covariance matrix and estimated covariance matrix can be measured for choosing an appropriate number of common factor q. Therefore, if the residual covariance matrix is decently small for some number of factors, the number of factors chosen is appropriate. However, it is subjective whether to accept the residual covariance matrix. Q-norm is introduced in this section as an objective criterion to compute the distance between the sample covariance and the estimated covariance (Fan et al., 2008), instead of the residual matrix.. Q-norm is defined as:

$$Q - norm = p^{1/2}||\hat{\Sigma} - \Sigma||_{\Sigma} = [tr(\hat{\Sigma}\Sigma^{-1} - I)^2]^{1/2}$$

The Q-norm is used for choosing q, the number of common factors. Q-norm is expected to be large if q is small and to be small if q is large. Q-norms are computed for a sequence of q. Then, similar to a scree plot (Johnson and Wichern, 2007) used for determining an appropriate number of principal components, a plot of Q-norm versus q can be drawn. There is always a bend in the plot for sparse data, and the bend point is determined to be the appropriate number of factors because, from that

bend point, Q-norms are relatively small and almost the same levels. We show this in the simulation models.

A training dataset and a validation dataset are used for evaluating the number of common factors. β and τ^2 are estimated from the training dataset for a sequence of q without penalty. Then Q-norms are computed from those estimated parameters and the covariance matrix from the validation dataset. As q increases, there is a bend point on the Q-norm curve. q is chosen at this point that is the appropriate number of common factors. Figure Figure B.1 shows a series of q and corresponding Q-norm. From the figures, q is chosen to be 4 for all of three simulations.

As the similar method of selecting q, Q-norm is also utilized for selecting λ. Let cc be defined to be the number of correctly estimated zero β and nc be the number of wrongly estimated none zero β. That is,

$$\begin{aligned} cc &= \sum(\hat{\beta}_{ij} = 0 \cap \beta_{ij} = 0) \\ nc &= \sum(\hat{\beta}_{ij} = 0 \cap \beta_{ij} \neq 0) \end{aligned}$$

For the Lasso or ALasso method, Q-norm is computed for an equally spaced grid of λ for fixed q. Then λ is chosen right at the point where Q-norm abruptly jumps up. Based on our simulations, as λ increases before that jump point, cc also increases, whereas nc is almost always zero. On the other hand, as λ increases after that jump point, nc starts to increase, whereas cc is almost always level. The number of errors starts at the jump point and that point is best for both cc and nc. In the simulations, two data set are generated, a training dataset with 100 observations and a validation dataset with 100 observations. First, the training data are utilized to compute the estimates of β and τ^2 on a grid of λ. After computing β and τ^2, the validation dataset is used in addition to the training dataset to compute Q-norm. For Q-norm calculation, $\hat{\Sigma}$ is computed from the estimates of β and τ^2 on the training

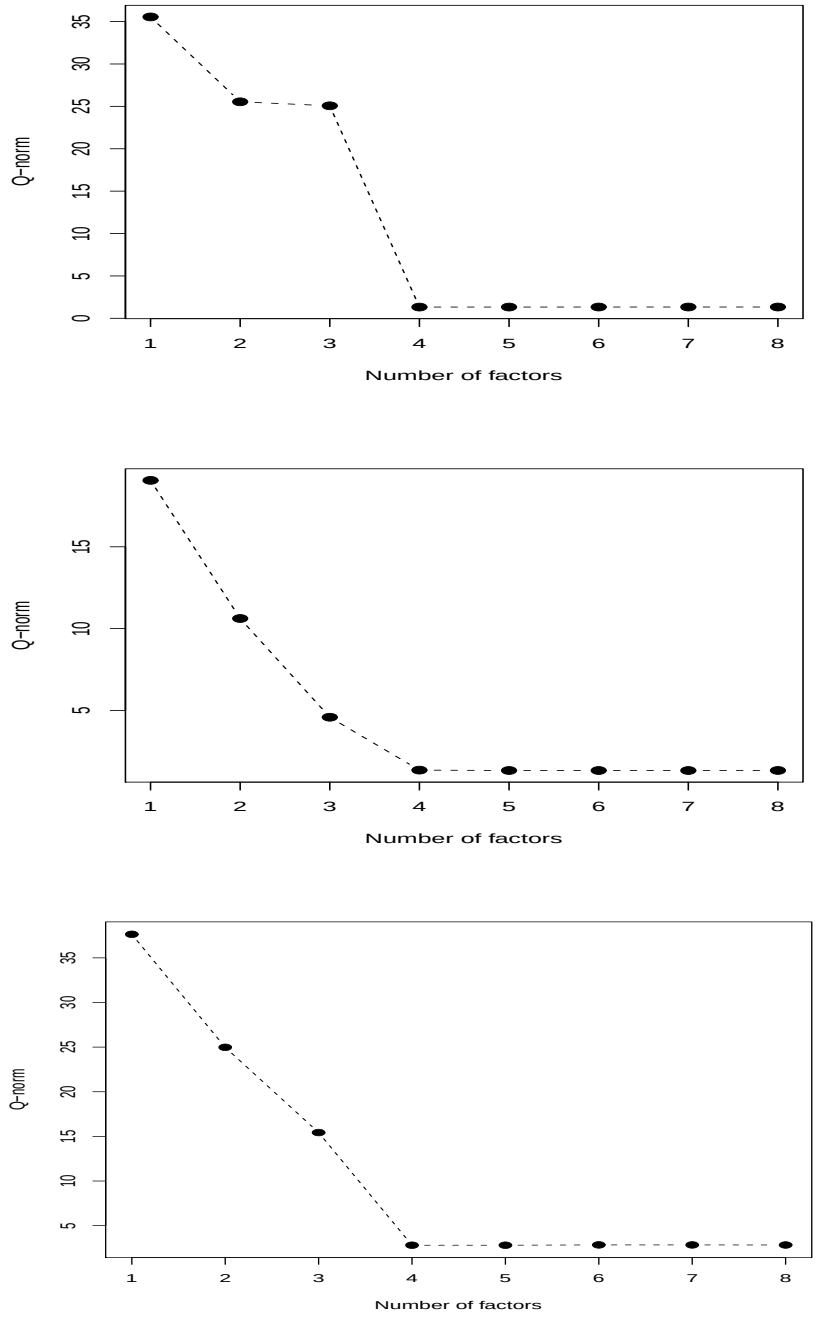

Figure B.1: Q-norm vs. number of factors: top-model 1, middle-model 2, bottom-model3.

dataset and the Σ is computed as the sample covariance matrix from the validation set. In a series of λ's, λ at the jump-up point on the Q-norm curve is selected and cc and nc are counted. This process is repeated 50 times for each λ. From the Q-norm curve, we can find a reasonably good λ. In model 1, for the Lasso method, reasonably good average λ is approximately 10. For the ALasso method, the average λ is also 10. Similarly, in model 2, the adequate average λ is 10 for both the Lasso and ALASSO methods. In model 3, the adequate λ is not as clear as in models 1 and 2 because nc does not have increasing trends for the Lasso and ALasso methods. The reason is that there are more than one nonzero entries in some columns in the true β matrix in model 3. Therefore, we may presume that the ℓ_1 penalized model works best in the models where each column in true β matrix has only one nonzero entry. In Table Table B.1 and Figures Figure B.2,Figure B.3, and Figure B.4, the relationships of Q-norm, cc, and nc with λ are shown. Although the figures may not be clear, you would agree with the selected λ when you investigate nc in Table Table B.1.

Table B.1: Q-norm vs λ: 50 replications for each λ: top: Lasso, bottom: ALasso

Lasso	λ	0	10	20	30	40	50	60
Model 1	Q-norm	0.92	0.80	2.58	16.37	22.96	30.30	33.41
	cc	0.00	11.86	16.62	24.80	29.50	33.22	35.20
	nc	0.00	0.00	0.24	2.34	3.96	6.48	8.16
Model 2	Q-norm	0.89	0.78	1.45	3.36	4.89	6.15	7.47
	cc	0.00	12.48	18.30	23.54	27.84	30.66	33.10
	nc	0.00	0.00	0.38	1.56	2.52	3.42	4.68
Model 3	Q-norm	1.59	1.37	1.26	1.79	8.54	15.51	19.61
	cc	0.00	10.40	14.36	16.88	20.88	24.54	28.16
	nc	0.00	0.06	0.00	0.06	1.08	2.24	4.16
ALasso	λ	0	5	10	15	20	25	30
Model 1	Q-norm	0.92	0.80	0.76	1.15	4.24	13.47	18.32
	cc	0.00	8.24	11.86	13.98	16.62	20.56	24.80
	nc	0.00	0.00	0.00	0.06	0.24	1.32	2.34
Model 2	Q-norm	0.89	0.79	0.76	0.85	1.68	3.07	4.13
	cc	0.00	8.22	12.48	15.20	18.30	20.70	23.54
	nc	0.00	0.00	0.00	0.04	0.38	1.02	1.56
Model 3	Q-norm	1.59	1.41	1.33	1.26	1.21	1.80	3.04
	cc	0.00	7.16	10.40	12.82	14.36	15.68	16.88
	nc	0.00	0.04	0.06	0.00	0.00	0.00	0.06

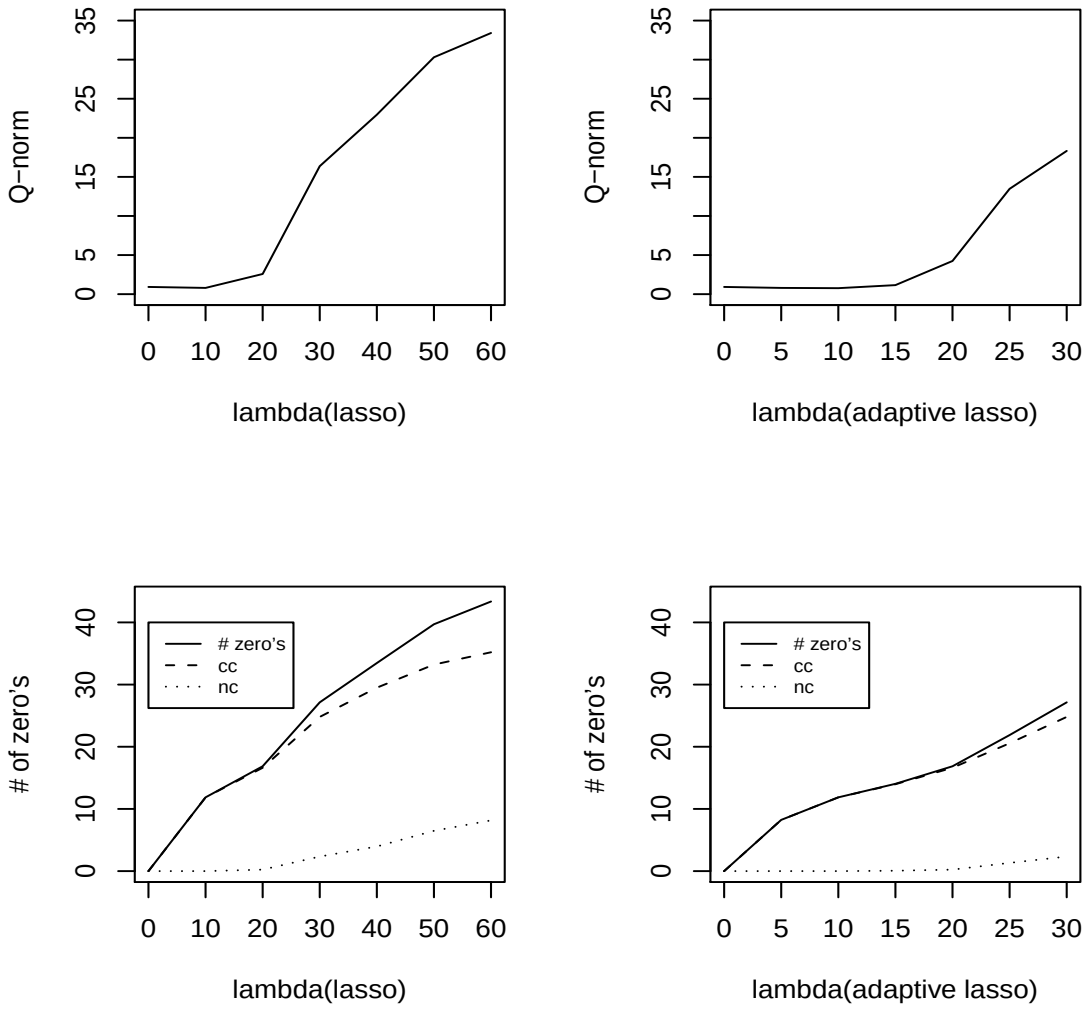

Figure B.2: Model 1: Q-norm and zero counts vs λ: 50 replications for each λ

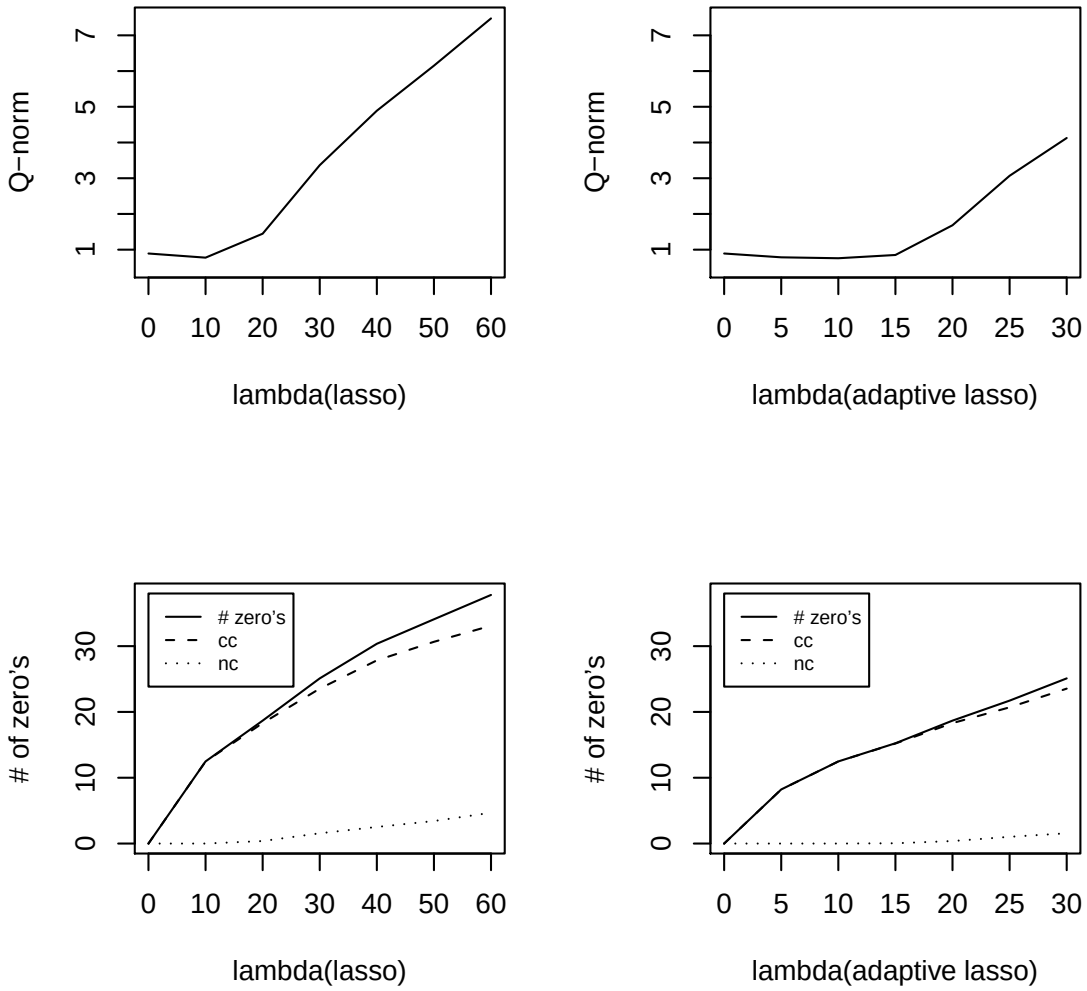

Figure B.3: Model 2: Q-norm and zero counts vs λ: 50 replications for each λ

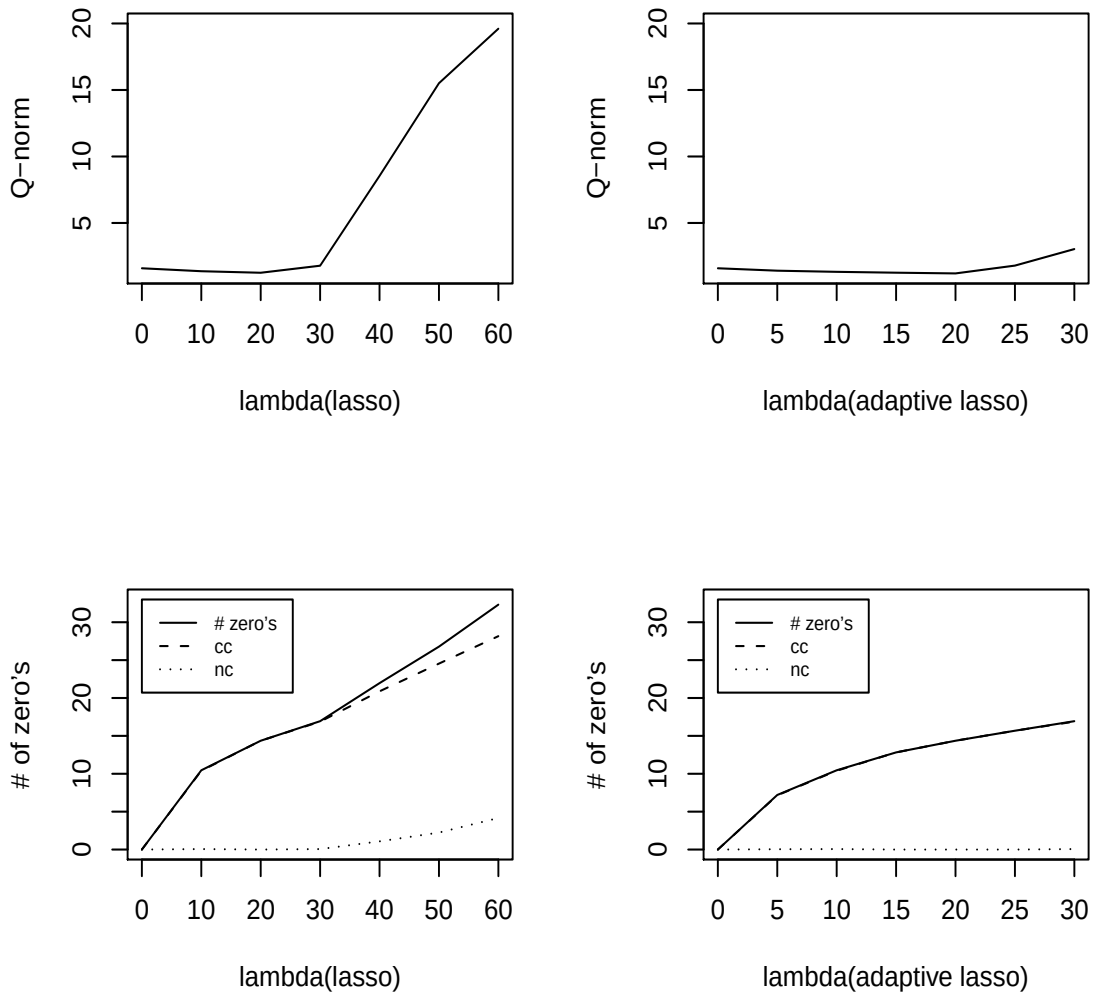

Figure B.4: Model 3: Q-norm and zero counts vs λ: 50 replications for each λ

B.2 Real data by Q-norm

B.2.1 Body fat data

The dataset is from Medicine and Science in Sports and Exercise vol. 17, no. 2, April 1985, p. 189 (Penrose et al., 1985). The data are the estimates of the percentage of body fat determined by underwater weighing and various body circumference measurements for 252 men. In this dataset, we use all variables except body fat to determine if relationships exist among the variables. The 14 variables listed are Density determined from underwater weighing, Age (years), Weight (lbs), Height (inches), Neck circumference (cm), Chest circumference (cm), Abdomen 2 circumference (cm), Hip circumference (cm), Thigh circumference (cm), Knee circumference (cm), Ankle circumference (cm), Biceps (extended) circumference (cm), Forearm circumference (cm), Wrist circumference (cm). We first select q, the number of common factors, and then λ by computing Q-norm. For Q-norm calculations, the data are divided into two parts, training data and validation data. One third of the original data are randomly assigned to the validation dataset (84 observations), and the remaining data are assigned to the training dataset (168 observations). Q-norm is computed using the estimated parameters from the training dataset and sample covariance from the validation dataset. First, Q-norm is computed for choosing q with $\lambda = 0$. Then, for selecting λ with the chosen q. q is selected at the jump point on the Q-norm curve from a sequence of q. Then, λ is chosen at the jump point on the Q-norm curve from equally spaced λ. $q = 5$ is chosen and $\lambda = 35$ is selected for the Lasso penalty and 2 for ALasso penalty (with γ of 1) for this dataset. Table Table B.2 and Figure Figure B.5 show a sequence of q and corresponding Q-norm. In Tables Table B.3 and Table B.4 and Figures Figure B.6 and Figure B.7, a series of λ and corresponding Q-norm are computed for the Lasso and ALasso methods.

Table B.2: A sequence of q and corresponding Q-norm loss for Body fat data

q	1	2	3	4	5	6	7	8	...	14
Q-norm	6.10	3.64	3.07	2.65	2.28	2.28	2.11	2.10	...	2.06

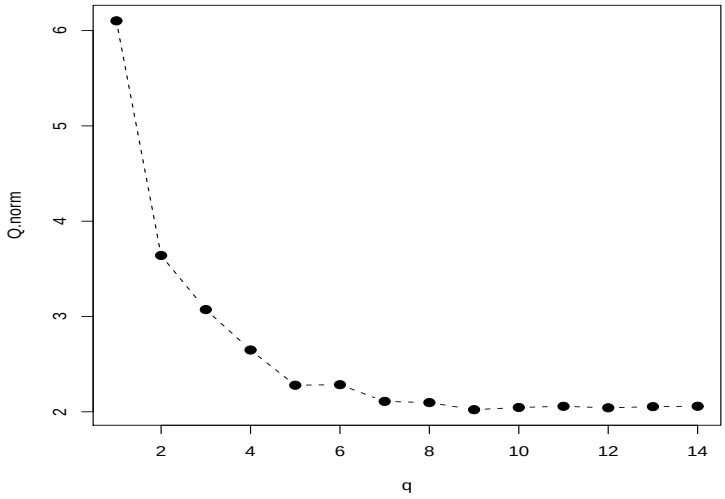

Figure B.5: Body fat data: Q-norm vs. number of factors

Table B.3: A sequence of λ and corresponding Q-norm for Body fat data with q = 5 :Lasso

λ	20	25	30	35	40
Q-norm	2.048	2.041	2.036	2.035	2.041

Table B.4: A sequence of λ and corresponding Q-norm for Body fat data with q = 5 :ALasso

λ	0	2	4	6
Q-norm	2.223	2.239	2.314	2.413

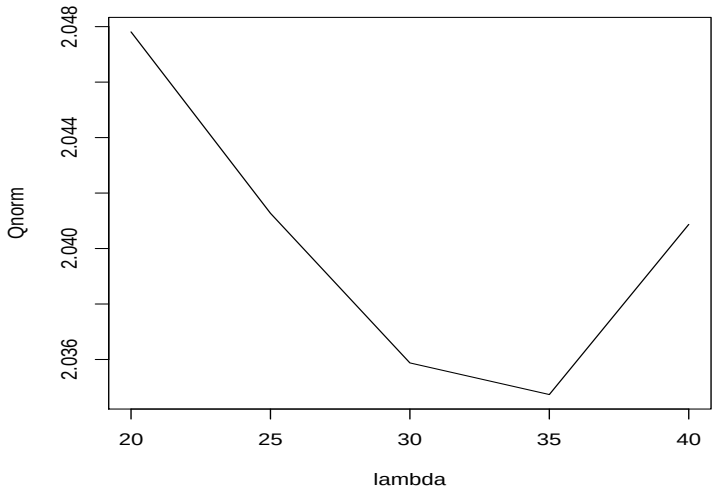

Figure B.6: Body fat data: Q-norm vs. λ (Lasso): q = 5

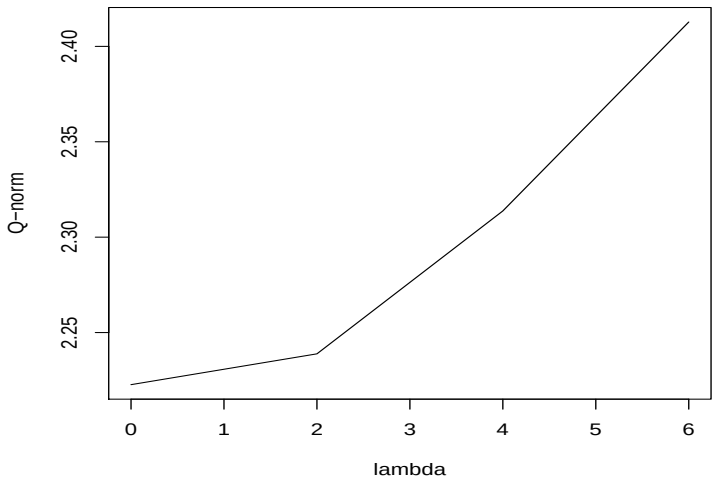

Figure B.7: Body fat data: Q-norm vs. λ (ALasso): q = 5

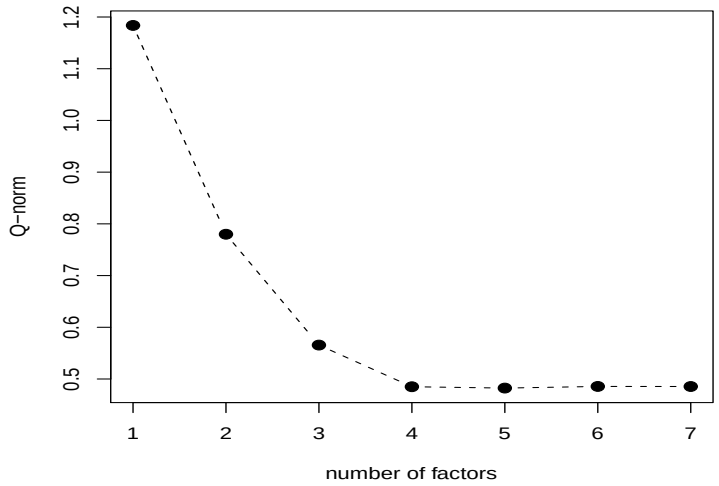

Figure B.8: HS Data: Q-norm vs. number of factors

B.2.2 Holzinger and Swineford (HS)

The dataset is from Holzinger and Swineford (1939). We first select q, and then λ by computing Q-norm. For Q-norm calculations, the data are divided into two parts, training data and validation data. One third of the original data are randomly assigned to the validation dataset, and the remaining data are assigned to the training dataset. Q-norm is computed using the estimated parameters that come from the training dataset and sample covariance that comes from the validation dataset. First, Q-norm is computed for choosing q with $\lambda = 0$ and then for selecting λ with the chosen q. q is selected at the bend point on the Q-norm curve from a sequence of q and then λ is chosen at the jump-up point on the Q-norm curve from equally spaced λ. $q = 4$ is chosen and $\lambda = 30$ is selected for the Lasso penalty and 10 for the ALasso penalty (with γ of 1) for this dataset. Figure Figure B.8 show a sequence of q and corresponding Q-norm.

In Tables Table B.5 and Table B.6 and Figures Figure B.9 and Figure B.10, a series

of λ and corresponding Q-norm are computed for the Lasso and ALasso method. The appropriate Q-norm by the lasso and ALasso models are 30 and 10, respectively. A Lasso model has 15 exact zero loadings when $\lambda = 30$ and an ALasso model has 7 exact zero loadings when $\lambda = 10, \Gamma = 1$.

The ℓ_1 penalty models are tested if they are within the confidence interval of the maximum likelihood model at the 0.01 level. By performing the likelihood ratio test in ((1.4)), we attain a p-values of almost zero for the Lasso model with $\lambda = 30$ and for the ALasso model with $\lambda = 10, \Gamma = 1$. Therefore, certainly ℓ_1 penalty models are out of the confidence interval of the maximum likelihood model.

Table B.5: A sequence of λ and corresponding Q-norm for HS data with q = 4 :Lasso

λ	0	10	20	30	40	50	$\cdots$
Q-norm	0.490	0.471	0.466	0.464	0.474	0.527	$\cdots$

Table B.6: A sequence of λ and corresponding Q-norm for HS data with q = 4 :ALasso

λ	0	5	10	15	20	$\cdots$
Q-norm	0.403	0.433	0.497	0.515	0.533	$\cdots$

B.2.3 Parkinsons

The Oxford Parkinson's data (Little et al., 2008) is from "Suitability of Dysphonia Measurements for Tele-monitoring of Parkinson's Disease" by Max A. Little, Patrick E. McSharry, Eric J. Hunter, Lorraine O. Ramig (2008), IEEE Transactions on Biomedical Engineering (to appear). The dataset consists of 23 columns and 195 rows. Each column is a particular voice measure, and each row is one of 195 voice

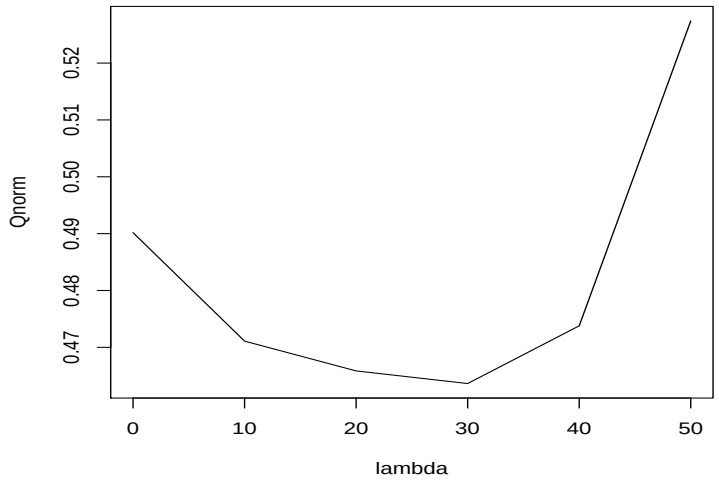

Figure B.9: HS data: Q-norm vs. λ (Lasso): q = 4

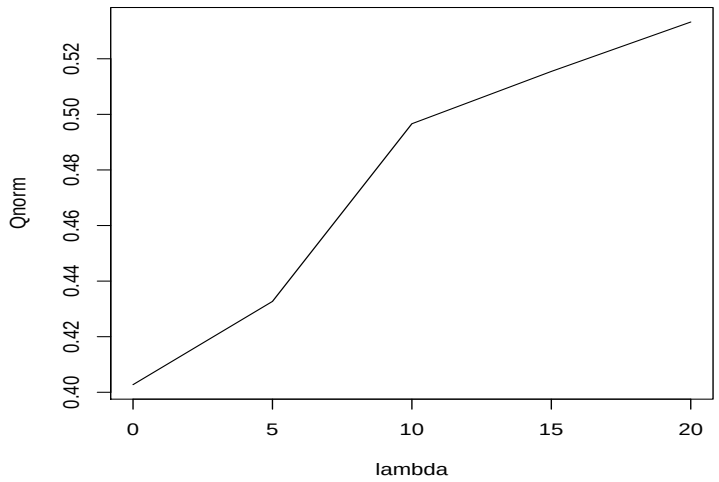

Figure B.10: HS data: Q-norm vs. λ (ALasso): q = 4

recordings from 31 individuals, 23 of which have Parkinson's disease (PD). We apply the proposed sparse factor analysis method. We randomly split the data into a training set (130 observations) and a validation set (65 observations) and measure the accuracy of the model by computing the KL loss evaluated on the validation set. Before fitting any factor model, we standardized the data so that each feature has mean 0 and standard deviation 1.

We select q at the bend point on the Q-norm curve from a sequence of q and then λ at the jump-up point on the Q-norm curve from equally spaced λ. The Q-norm for q = 4 has a bend point. Therefore, we select q = 4 and $\lambda = 250$ for the Lasso penalty and 60 for the ALasso penalty (with γ of 1) for this dataset.

Table Table B.7 and Figure Figure B.11 show a sequence of q and corresponding Q-norm. Tables Table B.8 and Table B.9 illustrate a sequence of λ and corresponding Q-norm for the Lasso and ALasso, and Figure Figure B.12 shows a sequence of λ and corresponding Q-norm for the Lasso and ALasso.

Table B.7: $q = 4$ gives the best MLE model for Parkinson's disease data.

q	...	2	3	4	5	6	...
Q-norm	...	516467	357998	107106	107106	107106	...

Table B.8: $\lambda = 250$ gives the best Lasso model for Parkinson's disease data.

λ	$\cdots$	100	150	200	250	300	350	$\cdots$
Q-norm	$\cdots$	160550	160549	160546	160548	948591	983244	$\cdots$

Table B.9: $\lambda = 60$ gives the best ALasso model for Parkinson's disease data.

λ	$\cdots$	30	40	50	60	70	$\cdots$
Q-norm	$\cdots$	603328	607832	612362	616899	1209234	$\cdots$

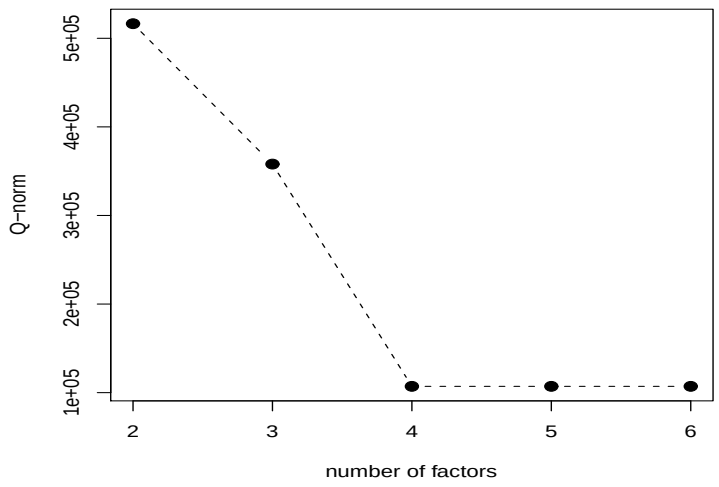

Figure B.11: Parkinsons data: q = 4 gives the best MLE model.

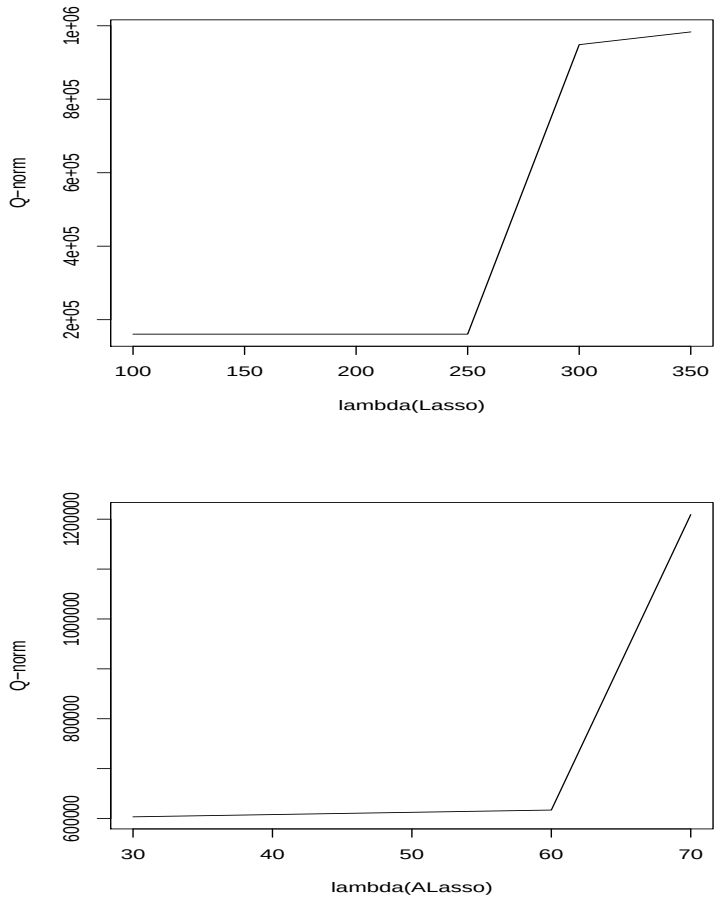

Figure B.12: Parkinsons data: Q-norm vs. λ: q = 4

B.2.4 Image Segmentation

Image segmentation data were created by Vision Group at the University of Massachusetts in November 1990, and are stored in UCI Machine Learning Repository (Asunction & Newman, 2007). The data consist of 7 outdoor images with training data from 2100 observations. The images were hand segmented, so that a classification for every pixel could be created. Each instance is a 3×3 region. There are 19 variables with all real numbers, but the data on the third column (region−pixel−count: the number of pixels in a region = 9) are all the same number: consequently, column is excluded. We determine q and λ for each of the 7 images according to the classes of brickface, sky, foliage, cement, window, path, and grass. Each image consists of data from 300 instances. Two thirds of each data set are randomly selected for training data, and the remaining one third of each dataset is used for validation data.

As a measurement, Q-norm is utilized for this data to determine the appropriate number of q and λ.

The first data is the fourth image data, cement. We fit the Lasso factor model by estimating Q-norm. From Tables Table B.10 and Table B.11 and Figure Figure B.13 we determine the appropriate number of q is 2. The appropriate λ of 50 is obtained by estimating Q-norm for a grid of penalization parameter.

Table B.10: $q = 2$ gives the best MLE model for cement image data.

q	1	2	3	4	5	...
Q-norm	2.198e+13	1.175e+12	1.175e+12	1.050e+12	1.050e+12	...

The other data is the last image data, grass. From Tables Table B.12 and Table B.13 and Figure Figure B.14 we determine the appropriate number of q is 3 and the appropriate λ is 100.

Table B.11: $\lambda = 50$ gives the best Lasso model for cement image data.

λ	0	25	50	75	...
KL	1.163e+12	1.154e+12	1.154e+12	2.190e+13	...

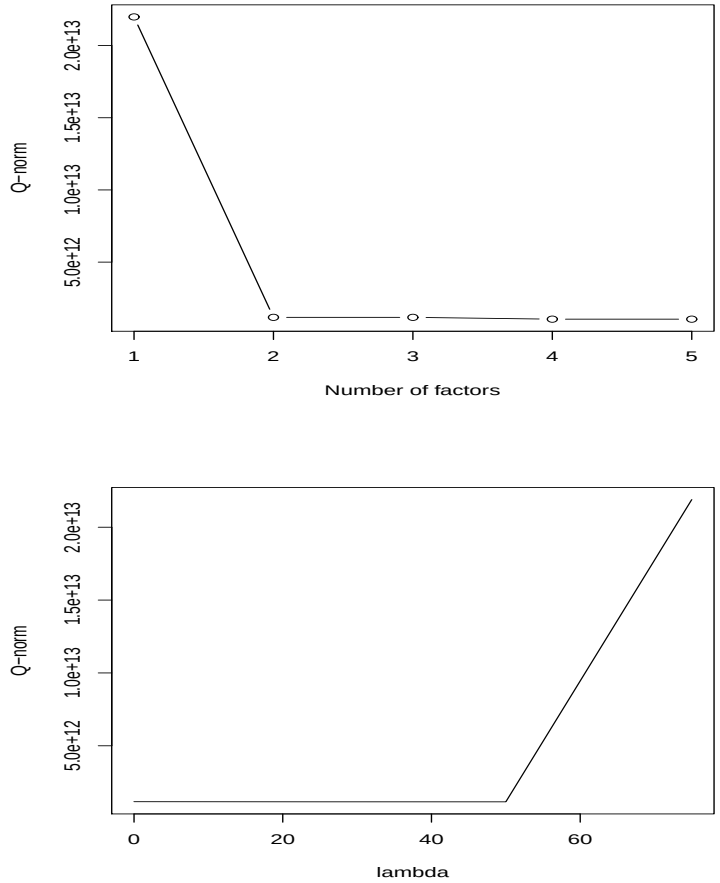

Figure B.13: cement image: Q-norm vs. number of factors and lambda

Table B.12: $q = 3$ gives the best MLE model for grass image data.

q	···	2	3	4	5	6	···
Q-norm	···	2.79e+13	9.16e+11	9.16e+11	9.16e+11	9.16e+11	···

Table B.13: $\lambda = 100$ gives the best Lasso model for grass image

λ	0	25	50	75	<u>100</u>	125	$\cdots$
Q-norm	6.95e+11	6.97e+11	6.98e+11	7.00e+11	<u>7.02e+11</u>	2.70e+13	$\cdots$

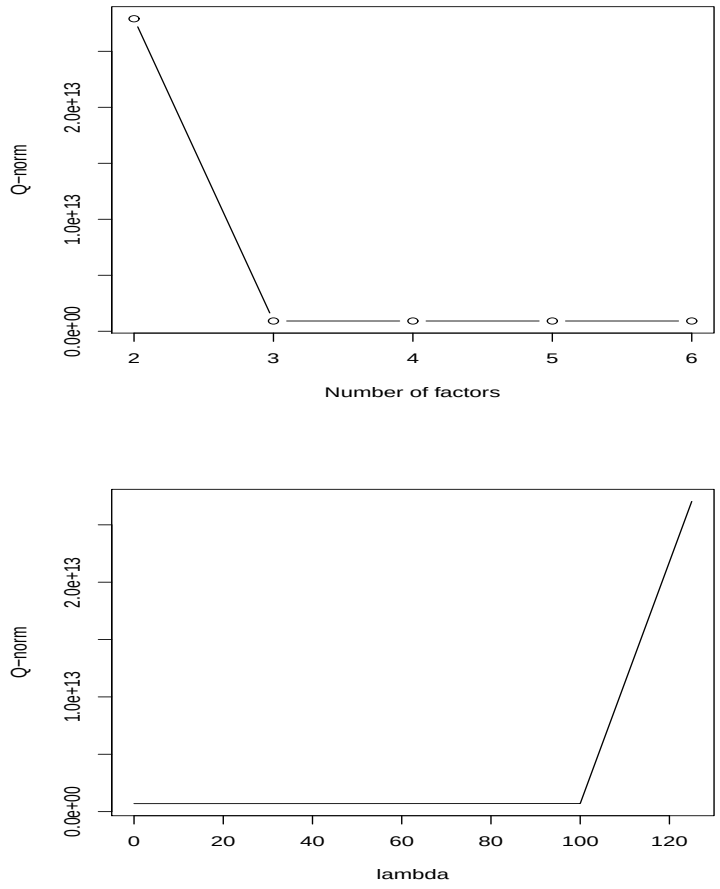

Figure B.14: grass image: Q-norm vs. number of factors and lambda

Appendix C

Proof of Ascent property of Algorithm 1

Let the conditional density of $\mathbf{X}$ given $\mathbf{Y}$ and $\boldsymbol{\tau^2}, \boldsymbol{\beta}$ be

$$f(\mathbf{X}|\boldsymbol{\tau^2}, \boldsymbol{\beta}, \mathbf{Y}) = \frac{f(\mathbf{Y},\mathbf{X}|\boldsymbol{\tau^2}, \boldsymbol{\beta})}{f(\mathbf{Y}|\boldsymbol{\tau^2}, \boldsymbol{\beta})}$$

and Penalized log-likelihood (PLL) of $\mathbf{Y}$ be

$$PLL(\boldsymbol{\tau^2}, \boldsymbol{\beta}) = LL(\boldsymbol{\tau^2}, \boldsymbol{\beta}) - P_\lambda(\boldsymbol{\beta})$$

where $P_\lambda(\boldsymbol{\beta})$ is the penalty function.

Also define

$$ELLP_{(k)}(\boldsymbol{\tau^2}, \boldsymbol{\beta}) = ELL_{(k)}(\boldsymbol{\tau^2}, \boldsymbol{\beta}) - E[P_\lambda(\boldsymbol{\beta})|\boldsymbol{\tau^2_{(k)}}, \boldsymbol{\beta_{(k)}}, \mathbf{Y}]$$

Then,

$$PLL(\boldsymbol{\tau^2}, \boldsymbol{\beta}) = ELLP_{(k)}(\boldsymbol{\tau^2}, \boldsymbol{\beta}) - E[\log f(\mathbf{X}|\boldsymbol{\tau^2}, \boldsymbol{\beta}, \mathbf{Y})|\boldsymbol{\tau^2_{(k)}}, \boldsymbol{\beta_{(k)}}, \mathbf{Y}]$$

Since the (k+1)th $ELLP_{(k)}(\boldsymbol{\tau^2}, \boldsymbol{\beta})$ is obtained by maximizing the expected log-likelihood,

$$ELLP_{(k)}(\boldsymbol{\tau^2_{(k+1)}}, \boldsymbol{\beta_{(k+1)}}) \geq ELLP_{(k)}(\boldsymbol{\tau^2_{(k+1)}}, \boldsymbol{\beta}) \geq ELLP_{(k)}(\boldsymbol{\tau^2}, \boldsymbol{\beta})$$

for any $\boldsymbol{\tau^2}, \boldsymbol{\beta}$. On the other hand, by Jensen's inequality,

$$\begin{aligned} E[\log f(\mathbf{X}|\boldsymbol{\tau^2}, \boldsymbol{\beta}, \mathbf{Y})|\boldsymbol{\tau}^{\mathbf{2}}_{(\boldsymbol{k})}, \boldsymbol{\beta}_{(\boldsymbol{k})}, \mathbf{Y}] &= \int \log f(\mathbf{X}|\boldsymbol{\tau^2}, \boldsymbol{\beta}, \mathbf{Y}) f(\mathbf{X}|\boldsymbol{\tau}^{\mathbf{2}}_{(\boldsymbol{k})}, \boldsymbol{\beta}_{(\boldsymbol{k})}, \mathbf{Y})\, dx \\ &\leq \int \log f(\mathbf{X}|\boldsymbol{\tau}^{\mathbf{2}}_{(\boldsymbol{k})}, \boldsymbol{\beta}, \mathbf{Y}) f(\mathbf{X}|\boldsymbol{\tau}^{\mathbf{2}}_{(\boldsymbol{k})}, \boldsymbol{\beta}_{(\boldsymbol{k})}, \mathbf{Y})\, dx \\ &\leq \int \log f(\mathbf{X}|\boldsymbol{\tau}^{\mathbf{2}}_{(\boldsymbol{k})}, \boldsymbol{\beta}_{(\boldsymbol{k})}, \mathbf{Y}) f(\mathbf{X}|\boldsymbol{\tau}^{\mathbf{2}}_{(\boldsymbol{k})}, \boldsymbol{\beta}_{(\boldsymbol{k})}, \mathbf{Y})\, dx \\ &= E[\log f(\mathbf{X}|\boldsymbol{\tau}^{\mathbf{2}}_{(\boldsymbol{k})}, \boldsymbol{\beta}_{(\boldsymbol{k})}, \mathbf{Y})|\boldsymbol{\tau}^{\mathbf{2}}_{(\boldsymbol{k})}, \boldsymbol{\beta}_{(\boldsymbol{k})}, \mathbf{Y}] \end{aligned}$$

Therefore,

$$\begin{aligned} & PLL(\boldsymbol{\tau}^{\mathbf{2}}_{(\boldsymbol{k+1})}, \boldsymbol{\beta}_{(\boldsymbol{k+1})}) - PLL(\boldsymbol{\tau}^{\mathbf{2}}_{(\boldsymbol{k})}, \boldsymbol{\beta}_{(\boldsymbol{k})}) \\ = \; & [ELLP_{(k)}(\boldsymbol{\tau}^{\mathbf{2}}_{(\boldsymbol{k+1})}, \boldsymbol{\beta}_{(\boldsymbol{k+1})}) - ELLP_{(k)}(\boldsymbol{\tau}^{\mathbf{2}}_{(\boldsymbol{k})}, \boldsymbol{\beta}_{(\boldsymbol{k})})] \\ - \; & [E[\log f(\mathbf{X}|\boldsymbol{\tau}^{\mathbf{2}}_{(\boldsymbol{k+1})}, \boldsymbol{\beta}_{(\boldsymbol{k+1})}, \mathbf{Y})|\boldsymbol{\tau}^{\mathbf{2}}_{(\boldsymbol{k})}, \boldsymbol{\beta}_{(\boldsymbol{k})}, \mathbf{Y}] \\ - \; & E[\log f(\mathbf{X}|\boldsymbol{\tau}^{\mathbf{2}}_{(\boldsymbol{k})}, \boldsymbol{\beta}_{(\boldsymbol{k})}, \mathbf{Y})|\boldsymbol{\tau}^{\mathbf{2}}_{(\boldsymbol{k})}, \boldsymbol{\beta}_{(\boldsymbol{k})}, \mathbf{Y}]] \\ \geq \; & 0 \end{aligned}$$

Appendix D

R code for algorithm of Sparse Factor Analysis using Lasso (SFAL)

D.1 SFAL1

```
SFAL1<-
function(Y0,center,scale,Beta1,Tau_sq1,q,lambda,conv_gap,noitr){
#-----------------------------------------------------------------
Y <- scale(Y0, center, scale)
n <- nrow(Y)
#----------- the sufficient statistic for using within the loop -----
y_j.y <- yty <- t(Y)%*%Y
Cyy <- yty/(n-1)
y_j.sq <- diag(t(Y)%*%Y)
#-----------------------------------------------------------------
count <- 0           # number of iterations
Beta<-matrix(rep(0,q*p),q,p) # set 0 beta matrix
#------------------step 5 : repeat step 2 - 4 until converges--------
#-----------------------------------------------------------------
while(max(abs(Beta1 - Beta)) > conv_gap && count <= noitr ) {
```

```
Beta <- Beta1
Tau_sq <- Tau_sq1
#-----------------step 2 : compute delta and Delta-------------------
delta <- solve(Tau_sq+t(Beta)%*%Beta)%*%t(Beta) # p * q matrix
Iq<-diag(1,q)
Delta <- (Iq-Beta%*%solve(Tau_sq+t(Beta)%*%Beta)%*%t(Beta))
#--------------------------------------------------------------------
ZTZ.numerator <- n*Delta+t(delta)%*%yty%*%delta
# step 4 : compute sufficient statistic for for-loop
for(j in 1:p) {
#------------------step 3 : compute Tau_sq--------------------------
Tau_sq1[j,j] <- (1/n)*(y_j.sq[j] - 2*y_j.y[j,]%*%delta%*%Beta[,j]
+ t(Beta[,j])%*%(n*Delta + t(delta)%*%yty%*%delta)%*%Beta[,j])
if(Tau_sq1[j,j] < 0.005) Tau_sq1[j,j]<-0.005
#Set a lower bound of uniqueness is 0.005
#--------------------------------------------------------------------
#------------------step 4 : compute Z, Y_tilde, and beta-------------
Z <- chol(ZTZ.numerator/Tau_sq1[j,j])
Y.tilde <- t(y_j.y[j,]%*%delta%*%solve(Z)/Tau_sq1[j,j])
b <- lars(Z,Y.tilde, intercept = FALSE, normalize=F)
Beta1[,j] <- coef(predict.lars(b, s=lambda/2, type = "coefficients",
mode = "lambda"))
#--------------------------------------------------------------------
}
count<-count+1
}
#--------------------------------------------------------------------
```

```
output<-list(Beta1, diag(Tau_sq1))
output
}
```

D.2 Bodyfat

D.2.1 Bodyfat: Scree Plot

```
library(lars)
library(mvtnorm)
set.seed(100)
#----------------------------choices---------------------------------
re <- 1                   # number of simulations
q <- c(1,2,3,4,5,6,7,8,9,10,11,12,13,14)    # number of factors
conv_gap <- 0.0001        # convergence gap for "while loop"
#-------------------------- Download Data --------------------------
Y0<-read.delim("http://www.stat.umn.edu/.../bodyfat.txt")
Y1 <- Y0[,-c(1)]
n.total <- nrow(Y1)
p <-ncol(Y1)
#--------------------Training data set-------------------------------
flow <- trunc(n.total/3)
Ytemp <- Y1[-(1:flow)*3,]
#--------------------------------------------------------------------
#--------------------Validation data set-------------------------
Ytemp_v <- Y1[(1:flow)*3,]
Y_v <- scale(Ytemp_v, center=T, scale=T)
```

```
Cyy_v <- t(Y_v)%*%Y_v/(flow - 1)
#-----------------------------------------------------------------
KL <- rep(0,length(q))
for(dd in 1:length(q)){
#-----------------step 1 : Set initial values----------------------
set.seed(100)
Btrue_initial <- matrix(rnorm(p*q[dd], 0, 1),q[dd],p)
for(aa in 1:p) for(bb in 1:q[dd]) if(abs(Btrue_initial[bb,aa]) >= 1)
                                    Btrue_initial[bb,aa]<-0.9

Ttrue_initial<-diag(rep(0.5,p))
#-----------------------------------------------------------------
BetaTau <- SFAL1(Y0=Ytemp, center=T, scale=T, Beta1=Btrue_initial,
Tau_sq1=Ttrue_initial,q=q[dd],lambda=0,conv_gap=0.0001,noitr=1000)
#=================================================================
#=================Beta is obtained from the above code============
#=================================================================
Beta1 <- BetaTau[[1]]
Tau_sq1 <- diag(BetaTau[[2]])
KL[dd] <- (1/2)*log(det(Tau_sq1+t(Beta1)%*%Beta1))
          +(1/2)*sum(diag((solve(Tau_sq1+t(Beta1)%*%Beta1))%*%Cyy_v))
          -(1/2)*log(det(Cyy_v))-p/2
}       # <- for(dd in 1:length(q))
#--------------------Results of Beta and Tau_sq--------------------
q
KL
plot(q, KL)
```

D.2.2 Bodyfat: Lasso Plot

```
library(lars)
library(mvtnorm)
set.seed(200)

Y1<-read.delim("http://www.stat.umn.edu/.../bodyfat.txt")
Y1 <- Y1[,-c(1)]

n.total <- nrow(Y1)
p <-ncol(Y1)

q <- 5                    # number of factors
conv_gap <- 0.0001        # convergence gap for "while loop"
lambda <- c(0,2,4,6,8,10,12,14,16,18,20)    # lambda for lasso
#----------------------Training data set----------------------------
flow <- trunc(n.total/3)
rorder_temp <- sample(1:n.total)
rorder <- rorder_temp[1:flow]
rorder <- rorder[order(rorder, decreasing=F)]
Ytemp <- Y1[-rorder,]
#-------------------------------------------------------------------
#----------------------Validation data set-------------------------
Ytemp_v <- Y1[rorder,]
Y_v <- scale(Ytemp_v, center=T, scale=T)
Cyy_v <- t(Y_v)%*%Y_v/(flow - 1)
#-------------------------------------------------------------------
KL <- rep(0, length(lambda))
```

```
#--------------------step 1 : Set initial values---------------------
Btrue_initial<-
t(factanal(Y1,factors=q,scores="regression",rotation="none")
          $loadings)
Ttrue_initial<-
diag(factanal(Y1,factors=q,scores="regression",rotation="none")
                  $uniqueness)

for(la in 1:length(lambda)){

#-----------------------------------------------------------------------

BetaTau <- SFAL1(Y0=Ytemp, center=T, scale=T, Beta1=Btrue_initial,
Tau_sq1=Ttrue_initial,q=q,lambda=lambda[la],conv_gap=0.0001,
               noitr=1000)
#=======================================================================
#================Beta is obtained from the above code================
#=======================================================================
Beta1 <- BetaTau[[1]]
Tau_sq1 <- diag(BetaTau[[2]])

KL[la] <- (1/2)*log(det(Tau_sq1+t(Beta1)%*%Beta1))
         +(1/2)*sum(diag((solve(Tau_sq1+t(Beta1)%*%Beta1))%*%Cyy_v))
         -(1/2)*log(det(Cyy_v))-p/2

}       # <- for(la in 1:length(lambda))
```

```
lambda
KL
#-------------------------------------------------------------------

plot(lambda, KL)
```

D.3 Parkinsons

D.3.1 Parkinsons: Scree Plot

```
library(lars)
library(mvtnorm)
set.seed(15)
#----------------------------choices---------------------------------
q <- c(5,6,7,8,9,10)                  # number of factors
conv_gap <- 0.0001       # convergence gap for "while loop"
#-------------------------- Download Data ---------------------------
Y0<-read.table("http://www.stat.umn.edu/.../parkinsons.txt",header=T)
Y1 <- Y0[,-1]
Y1 <- scale(Y1, center=T, scale=T)

n.total <- nrow(Y1)
p <-ncol(Y1)
#--------------------Training data set-------------------------------
flow <- trunc(n.total/3)
rorder_temp <- sample(1:n.total)
rorder <- rorder_temp[1:flow]
```

```
Ytemp <- Y1[-rorder,]
#-------------------------------------------------------------------
#---------------------Validation data set---------------------------
Ytemp_v <- Y1[rorder,]
Y_v <- scale(Ytemp_v, center=F, scale=F)
Cyy_v <- t(Y_v)%*%Y_v/(flow - 1)
#-------------------------------------------------------------------
KL <- rep(0,length(q))

for(dd in 1:length(q)){
#------------------step 1 : Set initial values----------------------
set.seed(100)

Btrue_initial <- matrix(rnorm(p*q[dd], 0, 1),q[dd],p)
for(aa in 1:p) for(bb in 1:q[dd]) if(abs(Btrue_initial[bb,aa]) >= 1)
                                     Btrue_initial[bb,aa]<-0.9

Ttrue_initial<-diag(rep(0.5,p))
#-------------------------------------------------------------------

BetaTau <- SFAL1(Y0=Ytemp, center=F, scale=F, Beta1=Btrue_initial,
 Tau_sq1=Ttrue_initial,q=q[dd],lambda=0,conv_gap=0.0001,noitr=1000)
#===================================================================
#==============Beta is obtained from the above code=================
#===================================================================
Beta1 <- BetaTau[[1]]
Tau_sq1 <- diag(BetaTau[[2]])
```

```
#------------------Example--------------------------------------
KL[dd] <- (1/2)*log(det(Tau_sq1+t(Beta1)%*%Beta1))
         +(1/2)*sum(diag((solve(Tau_sq1+t(Beta1)%*%Beta1))%*%Cyy_v))
         -(1/2)*log(det(Cyy_v))-p/2

}       # <- for(dd in 1:length(q))
#-----------------------------------------------------------------
lambda
KL
#-----------------------------------------------------------------

plot(q, KL)
```

D.3.2 Parkinsons: Lasso Plot

```
library(lars)
library(mvtnorm)
set.seed(200)

Y0<-read.table("http://.../parkinsons.txt",header=TRUE)
Y1 <- Y0[,-1]
Y1 <- scale(Y1, center=T, scale=T)

n.total <- nrow(Y1)
p <-ncol(Y1)

q <- 8                    # number of factors
```

```
conv_gap <- 0.0001        # convergence gap for "while loop"
lambda <- c(0,1,2,3,4,5)

#----------------------Training data set----------------------------
flow <- trunc(n.total/3)
rorder_temp <- sample(1:n.total)
rorder <- rorder_temp[1:flow]
Ytemp <- Y1[-rorder,]
#----------------------Validation data set-------------------------
Ytemp_v <- Y1[rorder,]
Y_v <- scale(Ytemp_v, center=F, scale=F)
Cyy_v <- t(Y_v)%*%Y_v/(flow - 1)
#------------------------------------------------------------------
#-------------------------a grid of lambda-------------------------
KL <- rep(0,length(lambda))
#--------------------step 1 : Set initial values--------------------
set.seed(100)
Btrue_initial <- matrix(rnorm(p*q, 0, 1),q,p)
for(aa in 1:p) for(bb in 1:q) if(abs(Btrue_initial[bb,aa]) >= 1)
                                Btrue_initial[bb,aa]<-0.9

Ttrue_initial<-
diag(factanal(Y1,factors=12,scores="regression",rotation="none")
                     $uniqueness)

for(la in 1:length(lambda)){
```

```
#-----------------------------------------------------------------------
BetaTau <- SFAL1(Y0=Ytemp, center=F, scale=F, Beta1=Btrue_initial,
Tau_sq1=Ttrue_initial,q=q,lambda=lambda[la],conv_gap=0.0001,
            noitr=1000)
#=======================================================================
#================Beta is obtained from the above code===============
#=======================================================================
Beta1 <- BetaTau[[1]]
Tau_sq1 <- diag(BetaTau[[2]])

#-----------------------------------------------------------------------

KL[la] <- (1/2)*log(det(Tau_sq1+t(Beta1)%*%Beta1))
         +(1/2)*sum(diag((solve(Tau_sq1+t(Beta1)%*%Beta1))%*%Cyy_v))
         -(1/2)*log(det(Cyy_v))-p/2

}       # <- for(la in 1:length(lambda))


lambda
KL

plot(lambda, KL, type="l", xlab="lambda(Lasso)",ylab="KL")
```

D.4 Image Segmentation

D.4.1 Brickface: Scree Plot

```
library(lars)
library(mvtnorm)
set.seed(100)
#----------------------------choices---------------------------------
q <- c(4,5,6,7,8,9)                 # number of factors
conv_gap <- 0.0001       # convergence gap for "while loop"
#------------------------- Download Data --------------------------
Y0<-read.csv("http://.../segment_test.txt",header=F)
Y2<-Y0[which(Y0[,1]=="BRICKFACE"),]
Y1<-Y2[,-c(1,4)]
Y1 <- scale(Y1, center=T, scale=T)

n.total <- nrow(Y1)
p <-ncol(Y1)

#-------------------Training data set-----------------------------
flow <- trunc(n.total/3)
#Ytemp <- Y1[-(1:flow)*3,]
rorder_temp <- sample(1:n.total)
rorder <- rorder_temp[1:100]
Ytemp <- Y1[-rorder,]
#----------------------Validation data set-------------------------
#Ytemp_v <- Y1[(1:flow)*3,]
Ytemp_v <- Y1[rorder,]
```

```
Y_v <- scale(Ytemp_v, center=F, scale=F)
Cyy_v <- t(Y_v)%*%Y_v/(flow - 1)
#-------------------------------------------------------------------
KL <- rep(0,length(q))

for(dd in 1:length(q)){
#--------------------step 1 : Set initial values--------------------
set.seed(100)

Btrue_initial <- matrix(rnorm(p*q[dd], 0, 1),q[dd],p)
for(aa in 1:p) for(bb in 1:q[dd]) if(abs(Btrue_initial[bb,aa]) >= 1)
                                  Btrue_initial[bb,aa]<-0.9

Ttrue_initial<-diag(rep(0.5,p))
#-------------------------------------------------------------------

BetaTau <- SFAL1(Y0=Ytemp, center=F, scale=F, Beta1=Btrue_initial,
 Tau_sq1=Ttrue_initial,q=q[dd],lambda=0,conv_gap=0.0001,noitr=1000)
#===================================================================
#=================Beta is obtained from the above code==============
#===================================================================
Beta1 <- BetaTau[[1]]
Tau_sq1 <- diag(BetaTau[[2]])

#------------------Example----------------------------------------
KL[dd] <- 1/2*log(det(Tau_sq1+t(Beta1)%*%Beta1))
         + 1/2*sum(diag(solve(Tau_sq1+t(Beta1)%*%Beta1)%*%Cyy_v))
```

```
         - 1/2*log(det(Cyy_v)) - p/2

}       # <- for(dd in 1:length(q))
#-------------------------------------------------------------------
q
KL
#-------------------------------------------------------------------
plot(q, KL, type="b", xlab="Number of factors", ylab="KL of MLE")
```

D.4.2 Brickface: Lasso Plot

```
library(lars)
library(mvtnorm)
set.seed(400)

Y0<-read.csv("http://.../segment_test.txt",header=F)
Y2<-Y0[which(Y0[,1]=="BRICKFACE"),]
Y1<-Y2[,-c(1,4)]
Y1<-scale(Y1, center=T, scale=T)

n.total <- nrow(Y1)
p <-ncol(Y1)

q <- 7                          # number of factors
conv_gap <- 0.0001              # convergence gap for "while loop"
lambda <- c(0,2,4,6,8,10,12,14)
#----------------------Training data set----------------------------
flow <- trunc(n.total/3)
```

```
Ytemp <- Y1[-(1:flow)*3,]
#----------------------Validation data set-------------------------
Ytemp_v <- Y1[(1:flow)*3,]
Y_v <- scale(Ytemp_v, center=F, scale=F)
Cyy_v <- t(Y_v)%*%Y_v/(flow - 1)
#-------------------------a grid of lambda-------------------------
KL <- rep(0,length(lambda))

Btrue_initial <- matrix(rnorm(p*q, 0, 1),q,p)
for(aa in 1:p) for(bb in 1:q) if(abs(Btrue_initial[bb,aa]) >= 1)
                              Btrue_initial[bb,aa]<-0.9

Ttrue_initial<-diag(rep(0.5,p))

for(la in 1:length(lambda)){

#--------------------step 1 : Set initial values--------------------
BetaTau <- SFAL1(Y0=Ytemp, center=F, scale=F, Beta1=Btrue_initial,
Tau_sq1=Ttrue_initial,q=q,lambda=lambda[la],conv_gap=0.0001,
                noitr=1000)
#=====================================================================
#==================Beta is obtained from the above code==============
#=====================================================================
Beta1 <- BetaTau[[1]]
Tau_sq1 <- diag(BetaTau[[2]])

KL[la] <- 1/2*log(det(Tau_sq1+t(Beta1)%*%Beta1))
```

```
         + 1/2*sum(diag(solve(Tau_sq1+t(Beta1)%*%Beta1)%*%Cyy_v))
         - 1/2*log(det(Cyy_v)) - p/2

}       # <- for(la in 1:length(lambda))
#-------------------------------------------------------------------
#------------------Results of Beta and Tau_sq-----------------------
lambda
LL
KL
#-------------------------------------------------------------------
plot(lambda, KL, type="l", xlab="lambda", ylab="KL")
```

D.4.3 Sky: Scree Plot

```
library(lars)
library(mvtnorm)
set.seed(200)
#----------------------------choices---------------------------------
q <- c(5,6,7,8,9)                 # number of factors
conv_gap <- 0.0001       # convergence gap for "while loop"
#-------------------------- Download Data --------------------------
Y0<-read.csv("http://.../segment_test.txt",header=F)
Y2<-Y0[which(Y0[,1]=="SKY"),]
Y1<-Y2[,-c(1,4,6)]
Y1 <- scale(Y1, center=T, scale=T)

n.total <- nrow(Y1)
p <-ncol(Y1)
```

```
#--------------------Training data set-------------------------------
flow <- trunc(n.total/3)
rorder_temp <- sample(1:n.total)
rorder <- rorder_temp[1:100]
Ytemp <- Y1[-rorder,]
#----------------------Validation data set-------------------------
Ytemp_v <- Y1[rorder,]
Y_v <- scale(Ytemp_v, center=F, scale=F)
Cyy_v <- t(Y_v)%*%Y_v/(flow - 1)
#------------------------------------------------------------------
LL <- rep(0,length(q))

for(dd in 1:length(q)){
#--------------------step 1 : Set initial values--------------------
set.seed(100)

Btrue_initial <- matrix(rnorm(p*q[dd], 0, 1),q[dd],p)
for(aa in 1:p) for(bb in 1:q[dd]) if(abs(Btrue_initial[bb,aa]) >= 1)
                                  Btrue_initial[bb,aa]<-0.9

Ttrue_initial<-diag(rep(0.5,p))
#------------------------------------------------------------------

BetaTau <- SFAL1(Y0=Ytemp, center=F, scale=F, Beta1=Btrue_initial,
 Tau_sq1=Ttrue_initial,q=q[dd],lambda=0,conv_gap=0.0001,noitr=1000)
#==================================================================
#==================Beta is obtained from the above code============
```

```
#=====================================================================
Beta1 <- BetaTau[[1]]
Tau_sq1 <- diag(BetaTau[[2]])

#------------------Example-----------------------------------------
LL[dd] <- -(n.total-flow)/2*log(det(Tau_sq1+t(Beta1)%*%Beta1))
 -(n.total-flow)/2*sum(diag(Cyy_v%*%solve(Tau_sq1+t(Beta1)%*%Beta1)))
}      # <- for(dd in 1:length(q))
#---------------------------------------------------------------------
#-----------------Results of Beta and Tau_sq----------------------
q
-LL
#---------------------------------------------------------------------
plot(q, -LL, type="b", xlab="Number of factors", ylab="NLL of MLE")
```

D.4.4 Sky: Lasso Plot

```
library(lars)
library(mvtnorm)
set.seed(600)

Y0<-read.csv("http:/.../segment_test.txt",header=F)
Y2<-Y0[which(Y0[,1]=="SKY"),]
Y1<-Y2[,-c(1,4,6)]
Y1<-scale(Y1, center=T, scale=T)

n.total <- nrow(Y1)
p <-ncol(Y1)
```

```
q <- 7                    # number of factors
conv_gap <- 0.0001        # convergence gap for "while loop"
lambda <- c(0,1,2,3,4,5,6)
#----------------------Training data set----------------------------
flow <- trunc(n.total/3)
rorder_temp <- sample(1:n.total)
rorder <- rorder_temp[1:flow]
Ytemp <- Y1[-rorder,]
#----------------------Validation data set-------------------------
Ytemp_v <- Y1[rorder,]
Y_v <- scale(Ytemp_v, center=F, scale=F)
Cyy_v <- t(Y_v)%*%Y_v/(flow - 1)
#-------------------------a grid of lambda-------------------------
LL <- rep(0,length(lambda))

#set.seed(100)
Btrue_initial <- matrix(rnorm(p*q, 0, 1),q,p)
for(aa in 1:p) for(bb in 1:q) if(abs(Btrue_initial[bb,aa]) >= 1)
                                Btrue_initial[bb,aa]<-0.9

Ttrue_initial<-diag(rep(0.5,p))

for(la in 1:length(lambda)){

BetaTau <- SFAL1(Y0=Ytemp, center=F, scale=F, Beta1=Btrue_initial,
Tau_sq1=Ttrue_initial,q=q,lambda=lambda[la],conv_gap=0.0001,
```

```
                noitr=1000)
#=====================================================================
#==================Beta is obtained from the above code==============
#=====================================================================
Beta1 <- BetaTau[[1]]
Tau_sq1 <- diag(BetaTau[[2]])

LL[la] <- -(n.total-flow)/2*log(det(Tau_sq1+t(Beta1)%*%Beta1))
-(n.total-flow)/2*sum(diag(Cyy_v%*%solve(Tau_sq1+t(Beta1)%*%Beta1)))

}       # <- for(la in 1:length(lambda))

#-------------------Results of Beta and Tau_sq-----------------------
lambda
-LL
#--------------------------------------------------------------------
plot(lambda, -LL, type="l", xlab="lambda", ylab="NLL")
```

D.4.5 Foliage: Scree Plot

```
library(lars)
library(mvtnorm)
set.seed(100)
#----------------------------choices-----------------------------------
q <- c(6,7,8,9,10)                 # number of factors
conv_gap <- 0.0001       # convergence gap for "while loop"
#------------------------- Download Data ----------------------------
Y0<-read.csv("http://.../segment_test.txt",header=F)
```

```
Y2<-Y0[which(Y0[,1]=="FOLIAGE"),]
Y1<-Y2[,-c(1,4)]
Y1 <- scale(Y1, center=T, scale=T)

n.total <- nrow(Y1)
p <-ncol(Y1)
#-------------------------------------------------------------------

#----------------------Training data set----------------------------
flow <- trunc(n.total/3)
rorder_temp <- sample(1:n.total)
rorder <- rorder_temp[1:100]
Ytemp <- Y1[-rorder,]
#-------------------------------------------------------------------
#----------------------Validation data set--------------------------
Ytemp_v <- Y1[rorder,]
Y_v <- scale(Ytemp_v, center=F, scale=F)
Cyy_v <- t(Y_v)%*%Y_v/(flow - 1)
#-------------------------------------------------------------------
LL <- rep(0,length(q))

for(dd in 1:length(q)){
#--------------------step 1 : Set initial values--------------------
set.seed(100)

Btrue_initial <- matrix(rnorm(p*q[dd], 0, 1),q[dd],p)
for(aa in 1:p) for(bb in 1:q[dd]) if(abs(Btrue_initial[bb,aa]) >= 1)
```

```
                                    Btrue_initial[bb,aa]<-0.9

Ttrue_initial<-diag(rep(0.5,p))
#-------------------------------------------------------------------

BetaTau <- SFAL1(Y0=Ytemp, center=F, scale=F, Beta1=Btrue_initial,
 Tau_sq1=Ttrue_initial,q=q[dd],lambda=0,conv_gap=0.0001,noitr=1000)
#===================================================================
#================Beta is obtained from the above code===============
#===================================================================
Beta1 <- BetaTau[[1]]
Tau_sq1 <- diag(BetaTau[[2]])

#------------------Example------------------------------------------
LL[dd] <- -(n.total-flow)/2*log(det(Tau_sq1+t(Beta1)%*%Beta1))
 -(n.total-flow)/2*sum(diag(Cyy_v%*%solve(Tau_sq1+t(Beta1)%*%Beta1)))
}       # <- for(dd in 1:length(q))
#-------------------------------------------------------------------
#------------------Results of Beta and Tau_sq-----------------------
q
-LL
#-------------------------------------------------------------------
plot(q, -LL, type="b", xlab="Number of factors", ylab="NLL of MLE")
```

D.4.6 Foliage: Lasso Plot

```
library(lars)
library(mvtnorm)
```

```
set.seed(100)

Y0<-read.csv("http://.../segment_test.txt",header=F)
Y2<-Y0[which(Y0[,1]=="FOLIAGE"),]
Y1<-Y2[,-c(1,4)]
Y1 <- scale(Y1, center=T, scale=T)

n.total <- nrow(Y1)
p <-ncol(Y1)

q <- 8                    # number of factors
conv_gap <- 0.0001        # convergence gap for "while loop"
lambda <- c(7,8,9,10,11,12)
#----------------------Training data set-----------------------------
flow <- trunc(n.total/3)
rorder_temp <- sample(1:n.total)
rorder <- rorder_temp[1:flow]
Ytemp <- Y1[-rorder,]
#-----------------------Validation data set-------------------------
Ytemp_v <- Y1[rorder,]
Y_v <- scale(Ytemp_v, center=F, scale=F)
Cyy_v <- t(Y_v)%*%Y_v/(flow - 1)
#-------------------------a grid of lambda-------------------------
LL <- rep(0,length(lambda))

Btrue_initial <- matrix(rnorm(p*q, 0, 1),q,p)
for(aa in 1:p) for(bb in 1:q) if(abs(Btrue_initial[bb,aa]) >= 1)
```

```
                              Btrue_initial[bb,aa]<-0.9

Ttrue_initial<-diag(rep(0.5,p))

for(la in 1:length(lambda)){

BetaTau <- SFAL1(Y0=Ytemp, center=F, scale=F, Beta1=Btrue_initial,
Tau_sq1=Ttrue_initial,q=q,lambda=lambda[la],conv_gap=0.0001,
             noitr=1000)
#=====================================================================
#===============Beta is obtained from the above code=================
#=====================================================================
Beta1 <- BetaTau[[1]]
Tau_sq1 <- diag(BetaTau[[2]])

LL[la] <- -(n.total-flow)/2*log(det(Tau_sq1+t(Beta1)%*%Beta1))
-(n.total-flow)/2*sum(diag(Cyy_v%*%solve(Tau_sq1+t(Beta1)%*%Beta1)))

}       # <- for(la in 1:length(lambda))

#-----------------Results of Beta and Tau_sq-------------------------
lambda
-LL
#--------------------------------------------------------------------

plot(lambda, -LL, type="l", xlab="lambda", ylab="NLL")
```

D.4.7 Cement: Scree Plot

```
library(lars)
library(mvtnorm)
set.seed(100)
#---------------------------choices---------------------------------
q <- c(1,2)                 # number of factors
conv_gap <- 0.0001          # convergence gap for "while loop"
#-------------------------- Download Data --------------------------
Y0<-read.csv("../segment_test.txt",header=F)
Y2<-Y0[which(Y0[,1]=="CEMENT"),]
Y1<-Y2[,-c(1,4)]
Y1 <- scale(Y1, center=T, scale=T)

n.total <- nrow(Y1)
p <-ncol(Y1)
#-------------------------------------------------------------------
#----------------------Training data set----------------------------
flow <- trunc(n.total/3)
Ytemp <- Y1[-(1:flow)*3,]
#----------------------Validation data set--------------------------
Ytemp_v <- Y1[(1:flow)*3,]
Y_v <- scale(Ytemp_v, center=F, scale=F)
Cyy_v <- t(Y_v)%*%Y_v/(flow - 1)
#-------------------------------------------------------------------
KL <- rep(0,length(q))

for(dd in 1:length(q)){
```

```
#--------------------step 1 : Set initial values---------------------
set.seed(100)

Btrue_initial <- matrix(rnorm(p*q[dd], 0, 1),q[dd],p)
for(aa in 1:p) for(bb in 1:q[dd]) if(abs(Btrue_initial[bb,aa]) >= 1)
                                  Btrue_initial[bb,aa]<-0.9

Ttrue_initial<-diag(rep(0.5,p))
#-----------------------------------------------------------------------

BetaTau <- SFAL1(Y0=Ytemp, center=F, scale=F, Beta1=Btrue_initial,
 Tau_sq1=Ttrue_initial,q=q[dd],lambda=0,conv_gap=0.0001,noitr=1000)
#=======================================================================
#================Beta is obtained from the above code================
#=======================================================================
Beta1 <- BetaTau[[1]]
Tau_sq1 <- diag(BetaTau[[2]])

#-------------------Example-----------------------------------------
KL[dd] <- (1/2)*log(det(Tau_sq1+t(Beta1)%*%Beta1))
         +(1/2)*sum(diag((solve(Tau_sq1+t(Beta1)%*%Beta1))%*%Cyy_v))
         -(1/2)*log(det(Cyy_v))-p/2
}       # <- for(dd in 1:length(q))
#--------------------Results of Beta and Tau_sq---------------------
q
KL
#-----------------------------------------------------------------------
```

```
plot(q, KL, type="b", xlab="Number of factors", ylab="KL of MLE")
```

D.4.8 Cement: Lasso Plot

```
library(lars)
library(mvtnorm)
set.seed(300)

Y0<-read.csv("http://.../segment_test.txt",header=F)
Y2<-Y0[which(Y0[,1]=="CEMENT"),]
Y1<-Y2[,-c(1,4)]
Y1<-scale(Y1, center=T, scale=T)

n.total <- nrow(Y1)
p <-ncol(Y1)

q <- 3                    # number of factors
conv_gap <- 0.0001        # convergence gap for "while loop"
lambda <- c(11,12,13,14,15,16)
#--------------------Training data set-----------------------------
flow <- trunc(n.total/3)
rorder_temp <- sample(1:n.total)
rorder <- rorder_temp[1:flow]
Ytemp <- Y1[-rorder,]
#---------------------Validation data set--------------------------
Ytemp_v <- Y1[rorder,]
Y_v <- scale(Ytemp_v, center=F, scale=F)
Cyy_v <- t(Y_v)%*%Y_v/(flow - 1)
```

```
#-----------------------a grid of lambda---------------------------
KL <- rep(0,length(lambda))

Btrue_initial <- matrix(rnorm(p*q, 0, 1),q,p)
for(aa in 1:p) for(bb in 1:q) if(abs(Btrue_initial[bb,aa]) >= 1)
                              Btrue_initial[bb,aa]<-0.9

Ttrue_initial<-diag(rep(0.5,p))

for(la in 1:length(lambda)){

BetaTau <- SFAL1(Y0=Ytemp, center=F, scale=F, Beta1=Btrue_initial,
Tau_sq1=Ttrue_initial,q=q,lambda=lambda[la],conv_gap=0.0001,
             noitr=1000)
#=====================================================================
#==================Beta is obtained from the above code=============
#=====================================================================
Beta1 <- BetaTau[[1]]
Tau_sq1 <- diag(BetaTau[[2]])

KL[la] <- 1/2*log(det(Tau_sq1+t(Beta1)%*%Beta1))
         + 1/2*sum(diag(solve(Tau_sq1+t(Beta1)%*%Beta1)%*%Cyy_v))
         - 1/2*log(det(Cyy_v)) - p/2

}       # <- for(la in 1:length(lambda))

#-----------------Results of Beta and Tau_sq-----------------------
```

```
lambda
KL
#-------------------------------------------------------------------

plot(lambda, KL, type="l", xlab="lambda", ylab="KL")
```

D.4.9 Windows: Scree Plot

```
library(lars)
library(mvtnorm)
set.seed(200)
#----------------------------choices---------------------------------
q <- c(5,6,7,8,9)                  # number of factors
conv_gap <- 0.0001       # convergence gap for "while loop"
#-------------------------- Download Data --------------------------
Y0<-read.csv("http://.../segment_test.txt",header=F)
Y2<-Y0[which(Y0[,1]=="WINDOW"),]
Y1<-Y2[,-c(1,4)]
Y1 <- scale(Y1, center=T, scale=T)

n.total <- nrow(Y1)
p <-ncol(Y1)
#-------------------------------------------------------------------
#----------------------Training data set----------------------------
flow <- trunc(n.total/3)
rorder_temp <- sample(1:n.total)
rorder <- rorder_temp[1:100]
Ytemp <- Y1[-rorder,]
```

```
#----------------------Validation data set-------------------------
Ytemp_v <- Y1[rorder,]
Y_v <- scale(Ytemp_v, center=F, scale=F)
Cyy_v <- t(Y_v)%*%Y_v/(flow - 1)
#------------------------------------------------------------------
KL <- rep(0,length(q))

for(dd in 1:length(q)){
#--------------------step 1 : Set initial values--------------------
set.seed(100)

Btrue_initial <- matrix(rnorm(p*q[dd], 0, 1),q[dd],p)
for(aa in 1:p) for(bb in 1:q[dd]) if(abs(Btrue_initial[bb,aa]) >= 1)
                                  Btrue_initial[bb,aa]<-0.9

Ttrue_initial<-diag(rep(0.5,p))
#------------------------------------------------------------------

BetaTau <- SFAL1(Y0=Ytemp, center=F, scale=F, Beta1=Btrue_initial,
 Tau_sq1=Ttrue_initial,q=q[dd],lambda=0,conv_gap=0.0001,noitr=1000)
#==================================================================
#=================Beta is obtained from the above code==============
#==================================================================
Beta1 <- BetaTau[[1]]
Tau_sq1 <- diag(BetaTau[[2]])

#------------------Example-----------------------------------------
```

```
KL[dd] <- (1/2)*log(det(Tau_sq1+t(Beta1)%*%Beta1))
         +(1/2)*sum(diag((solve(Tau_sq1+t(Beta1)%*%Beta1))%*%Cyy_v))
         -(1/2)*log(det(Cyy_v))-p/2
}       # <- for(dd in 1:length(q))
#-----------------------------------------------------------------
#--------------------Results of Beta and Tau_sq--------------------
q
KL
#-----------------------------------------------------------------
plot(q, KL, type="b", xlab="Number of factors", ylab="KL of MLE")
```

D.4.10 Windows: Lasso Plot

```
library(lars)
library(mvtnorm)
set.seed(100)

Y0<-read.csv("http://.../segment_test.txt",header=F)
Y2<-Y0[which(Y0[,1]=="WINDOW"),]
Y1<-Y2[,-c(1,4)]
Y1 <- scale(Y1, center=T, scale=T)

n.total <- nrow(Y1)
p <-ncol(Y1)

q <- 7                    # number of factors
conv_gap <- 0.0001        # convergence gap for "while loop"
lambda <- c(25,26,27,28,29,30)
```

```
#----------------------Training data set-----------------------------
flow <- trunc(n/3)
rorder_temp <- sample(1:n)
rorder <- rorder_temp[1:flow]
Ytemp <- Y1[-rorder,]
#--------------------------------------------------------------------
#----------------------Validation data set---------------------------
Ytemp_v <- Y1[rorder,]
Y_v <- scale(Ytemp_v, center=F, scale=F)
Cyy_v <- t(Y_v)%*%Y_v/(flow - 1)
#--------------------------a grid of lambda--------------------------
KL <- rep(0,length(lambda))

Btrue_initial <- matrix(rnorm(p*q, 0, 1),q,p)
for(aa in 1:p) for(bb in 1:q) if(abs(Btrue_initial[bb,aa]) >= 1)
                              Btrue_initial[bb,aa]<-0.9

Ttrue_initial<-diag(rep(0.5,p))

for(la in 1:length(lambda)){

BetaTau <- SFAL1(Y0=Ytemp, center=F, scale=F, Beta1=Btrue_initial,
Tau_sq1=Ttrue_initial,q=q,lambda=lambda[la],conv_gap=0.0001,
     noitr=1000)
#====================================================================
#==================Beta is obtained from the above code=============
#====================================================================
```

```
Beta1 <- BetaTau[[1]]
Tau_sq1 <- diag(BetaTau[[2]])

KL[la] <- 1/2*log(det(Tau_sq1+t(Beta1)%*%Beta1))
        + 1/2*sum(diag(solve(Tau_sq1+t(Beta1)%*%Beta1)%*%Cyy_v))
        - 1/2*log(det(Cyy_v)) - p/2

}       # <- for(la in 1:length(lambda))
#-------------------------------------------------------------------
#-----------------Results of Beta and Tau_sq-----------------------
lambda
KL
#-------------------------------------------------------------------

plot(lambda, KL, type="l", xlab="lambda", ylab="KL")
```

D.4.11 Path: Scree Plot

```
library(lars)
library(mvtnorm)
set.seed(300)
#---------------------------choices-----------------------------------
q <- c(3,4,5,6,7,8,9,10)          # number of factors
conv_gap <- 0.0001                # convergence gap for "while loop"
#------------------------ Download Data ----------------------------
Y0<-read.csv("http://.../segment_test.txt",header=F)
Y2<-Y0[which(Y.0[,1]=="PATH"),]
Y1<-Y2[,-c(1,4)]
```

```
Y1 <- scale(Y1, center=T, scale=T)

n.total <- nrow(Y1)
p <-ncol(Y1)
#-------------------------------------------------------------------
#----------------------Training data set----------------------------
flow <- trunc(n.total/3)
rorder_temp <- sample(1:n.total)
rorder <- rorder_temp[1:100]
Ytemp <- Y1[-rorder,]
#----------------------Validation data set--------------------------
Ytemp_v <- Y1[rorder,]
Y_v <- scale(Ytemp_v, center=F, scale=F)
Cyy_v <- t(Y_v)%*%Y_v/(flow - 1)
#-------------------------------------------------------------------
KL <- rep(0,length(q))

for(dd in 1:length(q)){
#--------------------step 1 : Set initial values--------------------
set.seed(100)

Btrue_initial <- matrix(rnorm(p*q[dd], 0, 1),q[dd],p)
for(aa in 1:p) for(bb in 1:q[dd]) if(abs(Btrue_initial[bb,aa]) >= 1)
                                  Btrue_initial[bb,aa]<-0.9

Ttrue_initial<-diag(rep(0.5,p))
#-------------------------------------------------------------------
```

```
BetaTau <- SFAL1(Y0=Ytemp, center=F, scale=F, Beta1=Btrue_initial,
 Tau_sq1=Ttrue_initial,q=q[dd],lambda=0,conv_gap=0.0001,noitr=1000)
#=========================================================================
#================Beta is obtained from the above code==============
#=========================================================================
Beta1 <- BetaTau[[1]]
Tau_sq1 <- diag(BetaTau[[2]])

#------------------Example-----------------------------------------
KL[dd] <- (1/2)*log(det(Tau_sq1+t(Beta1)%*%Beta1))
         +(1/2)*sum(diag((solve(Tau_sq1+t(Beta1)%*%Beta1))%*%Cyy_v))
         -(1/2)*log(det(Cyy_v))-p/2
}       # <- for(dd in 1:length(q))
#-------------------------------------------------------------------
#--------------------Results of Beta and Tau_sq--------------------
q
KL
#-------------------------------------------------------------------
plot(q, KL, type="b", xlab="Number of factors", ylab="KL of MLE")
```

D.4.12 Path: Lasso Plot

```
library(lars)
library(mvtnorm)
set.seed(10)

Y0<-read.csv("http://.../segment_test.txt",header=F)
```

```
Y2<-Y0[which(Y0[,1]=="PATH"),]
Y1<-Y2[,-c(1,4)]
Y1<-scale(Y1, center=T, scale=T)

n.total <- nrow(Y1)
p <-ncol(Y1)

q <- 8                    # number of factors
conv_gap <- 0.0001        # convergence gap for "while loop"
lambda <- c(0,2,4,6,8,10,12)
#----------------------Training data set----------------------------
flow <- trunc(n.total/3)
rorder_temp <- sample(1:n.total)
rorder <- rorder_temp[1:flow]
Ytemp <- Y1[-rorder,]
#----------------------Validation data set-------------------------
Ytemp_v <- Y1[rorder,]
Y_v <- scale(Ytemp_v, center=F, scale=F)
Cyy_v <- t(Y_v)%*%Y_v/(flow - 1)
#--------------------------a grid of lambda-------------------------
KL <- rep(0,length(lambda))

Btrue_initial <- matrix(rnorm(p*q, 0, 1),q,p)
for(aa in 1:p) for(bb in 1:q) if(abs(Btrue_initial[bb,aa]) >= 1)
                                Btrue_initial[bb,aa]<-0.9

Ttrue_initial<-diag(rep(0.5,p))
```

```
for(la in 1:length(lambda)){

BetaTau <- SFAL1(Y0=Ytemp, center=F, scale=F, Beta1=Btrue_initial,
Tau_sq1=Ttrue_initial,q=q,lambda=lambda[la],conv_gap=0.0001,
         noitr=1000)
#=====================================================================
#==================Beta is obtained from the above code=============
#=====================================================================
Beta1 <- BetaTau[[1]]
Tau_sq1 <- diag(BetaTau[[2]])

KL[la] <- 1/2*log(det(Tau_sq1+t(Beta1)%*%Beta1))
         + 1/2*sum(diag(solve(Tau_sq1+t(Beta1)%*%Beta1)%*%Cyy_v))
         - 1/2*log(det(Cyy_v)) - p/2

}       # <- for(la in 1:length(lambda))

#---------------------------------------------------------------------
#-----------------Results of Beta and Tau_sq--------------------------
lambda
KL
#---------------------------------------------------------------------

plot(lambda, KL, type="l", xlab="lambda", ylab="KL")
```

D.4.13 Grass: Scree Plot

```
library(lars)
library(mvtnorm)
set.seed(200)
#-------------------------choices-----------------------------------
q <- c(1,2,3,4,5,6,7,8)                 # number of factors
conv_gap <- 0.0001       # convergence gap for "while loop"
#-------------------------- Download Data ---------------------------
Y0<-read.csv("http://.../segment_test.txt",header=F)
Y2<-Y0[which(Y0[,1]=="GRASS"),]
Y1<-Y2[,-c(1,4)]
Y1 <- scale(Y1, center=T, scale=T)

n.total <- nrow(Y1)
p <-ncol(Y1)
#-----------------------------------------------------------------
#--------------------Training data set-----------------------------
flow <- trunc(n.total/3)
rorder_temp <- sample(1:n.total)
rorder <- rorder_temp[1:100]
Ytemp <- Y1[-rorder,]
#---------------------Validation data set--------------------------
Ytemp_v <- Y1[rorder,]
Y_v <- scale(Ytemp_v, center=F, scale=F)
Cyy_v <- t(Y_v)%*%Y_v/(flow - 1)
#-----------------------------------------------------------------
KL <- rep(0,length(q))
```

```
for(dd in 1:length(q)){
#------------------step 1 : Set initial values----------------------
set.seed(100)

Btrue_initial <- matrix(rnorm(p*q[dd], 0, 1),q[dd],p)
for(aa in 1:p) for(bb in 1:q[dd]) if(abs(Btrue_initial[bb,aa]) >= 1)
                                  Btrue_initial[bb,aa]<-0.9

Ttrue_initial<-diag(rep(0.5,p))
#-----------------------------------------------------------------------

BetaTau <- SFAL1(Y0=Ytemp, center=F, scale=F, Beta1=Btrue_initial,
 Tau_sq1=Ttrue_initial,q=q[dd],lambda=0,conv_gap=0.0001,noitr=1000)
#=======================================================================
#===================Beta is obtained from the above code=============
#=======================================================================
Beta1 <- BetaTau[[1]]
Tau_sq1 <- diag(BetaTau[[2]])

#-------------------Example-----------------------------------------
KL[dd] <- (1/2)*log(det(Tau_sq1+t(Beta1)%*%Beta1))
         +(1/2)*sum(diag((solve(Tau_sq1+t(Beta1)%*%Beta1))%*%Cyy_v))
         -(1/2)*log(det(Cyy_v))-p/2
}       # <- for(dd in 1:length(q))
#-----------------------------------------------------------------------
#--------------------Results of Beta and Tau_sq---------------------
```

```
q
KL
#-----------------------------------------------------------------
plot(q, KL, type="b", xlab="Number of factors", ylab="KL of MLE")
```

D.4.14 Grass: Lasso Plot

```
library(lars)
library(mvtnorm)
set.seed(100)

Y0<-read.csv("http://.../segment_test.txt",header=F)
Y2<-Y0[which(Y0[,1]=="GRASS"),]
Y1<-Y2[,-c(1,4)]
Y1 <- scale(Y1, center=T, scale=T)

n.total <- nrow(Y1)
p <-ncol(Y1)

q <- 5                    # number of factors
conv_gap <- 0.0001        # convergence gap for "while loop"
lambda <- c(36,38,40,42,44,46,48,50)

#----------------------Training data set----------------------------
flow <- trunc(n.total/3)
rorder_temp <- sample(1:n.total)
rorder <- rorder_temp[1:flow]
Ytemp <- Y1[-rorder,]
```

```
#-------------------------------------------------------------------
#----------------------Validation data set--------------------------
Ytemp_v <- Y1[rorder,]
Y_v <- scale(Ytemp_v, center=F, scale=F)
Cyy_v <- t(Y_v)%*%Y_v/(flow - 1)

#-------------------------a grid of lambda--------------------------
KL <- rep(0,length(lambda))

Btrue_initial <- matrix(rnorm(p*q, 0, 1),q,p)
for(aa in 1:p) for(bb in 1:q) if(abs(Btrue_initial[bb,aa]) >= 1)
                              Btrue_initial[bb,aa]<-0.9

Ttrue_initial<-diag(rep(0.5,p))

for(la in 1:length(lambda)){

BetaTau <- SFAL1(Y0=Ytemp, center=F, scale=F, Beta1=Btrue_initial,
Tau_sq1=Ttrue_initial,q=q,lambda=lambda[la],conv_gap=0.0001,
              noitr=1000)
#===================================================================
#================Beta is obtained from the above code===============
#===================================================================
Beta1 <- BetaTau[[1]]
Tau_sq1 <- diag(BetaTau[[2]])

KL[la] <- 1/2*log(det(Tau_sq1+t(Beta1)%*%Beta1))
```

```
        + 1/2*sum(diag(solve(Tau_sq1+t(Beta1)%*%Beta1)%*%Cyy_v))
        - 1/2*log(det(Cyy_v)) - p/2

}       # <- for(la in 1:length(lambda))

#-------------------------------------------------------------------
#------------------Results of Beta and Tau_sq-----------------------
lambda
KL
#-------------------------------------------------------------------

plot(lambda, KL, type="l", xlab="lambda", ylab="KL")
```