Estimating Sparse Graphical Models: Insights Through Simulation

by

Yunan Zhu

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

STATISTICS

Department of Mathematical and Statistical Sciences

Faculty of Science

University of Alberta

Abstract

Graphical models are frequently used to explore networks among a set of variables. Several methods for estimating sparse graphs have been proposed and their theoretical properties have been explored. There are also several selection criteria to select the optimal estimated models. However, their practical performance has not been studied in detail. In this work, several estimation procedures (glasso, bootstrap glasso, adptive lasso, SCAD, DP-glasso and Huge) and several selection criteria (AIC, BIC, CV, ebic, ric and stars) are compared under various simulation settings, such as different dimensions or sample sizes, different types of data, and different sparsity levels of the true model structures. Then we use several evaluation criteria to compare the optimal estimated models and discuss in detail the superiority and deficiency of each combination of estimating methods and selection criteria.

Acknowledgment

First and foremost, I express herein my deepest gratitude to my co-supervisors Dr. Ivor Cribben and Prof. Dr. Rohana Karunamuni for their guidence and support throughout my graduate studies.

After being brought to U of A by Dr. Karunamuni, my life as a graduate student started with a knowledgable and patient mentor. All sorts of difficulties, confusion and challenges happened to me during the last two years but, however, Dr. Karunamuni was the one who always sent me quiet but strong support to lift me up to a new stage. I thank Dr. Karunamuni for any of his continual attention to, interest in and concerns with my coursework and research, thought I have not had a chance to take a course with this excellent professor.

At the same time, Dr. Cribben opened the gate of research for me and pointed out a path towards the fascinating unknown. The topic about estimating sparse graphical models has been truly motivating me from the bottom of heart and occupying almost all of my efforts. Not only my degree thesis but also everything exchanged during our meetings boosts the quality of myself. It is doubtlessly a period of happy hours to pour my sweats into the field of statistical sciences with Dr. Cribben, especially in his gentle and considerate manner of mentoring and working.

I would like to thank Dr. Christoph Frei for not only chairing my committee of dissertation but also offering me a wonderful course on time series analysis. It is also an honor to have excellent scholars like Dr. Bei Jiang and Dr. Yuan Yan sit in my committee, to whom I send my appreciation for their interest and evaluation on my academic works. The instructors, friends and colleagues who shared their time and experience with me at U of A please also receive a special thank from me.

Financial support from the Department of Mathematical and Statistical Sciences at U of A and, especially from Dr. Karunamuni is so important to my pursuit that I sincerely hope my outcome of research deserves such an opportunity.

Finally, there is no word other than ”thanks” that can convey or convey better my appreciation, dedication and tribute to my family members. My thesis belongs to each of them as well.

Table of Contents

Abstract ii

Acknowledgment iii

Table of Contents viii

List of Tables xi

List of Figures xiv

1 Introduction 1

2 Estimating Methods 5

2.1 Sample (Partial) Correlation Matrix 6

2.1.1 Sample correlation matrix 6

2.1.2 Sample Partial Correlation Matrix 7

2.2 Graphical Lasso 7

2.2.1 Background 7

2.2.2 glasso algorithm 8

2.2.3 The R package glasso 9

2.2.4 Highlights and deficiencies 9

2.3 Bootstrap glasso 11

2.3.1 Background 11

2.3.2 Stability selection (SS) 11

2.3.3 BG algorithm 12

2.3.4 Summary and comments 14

2.4 glasso with Adaptive Lasso and SCAD penalties . . . 15
2.4.1 Background . . . 15
2.4.2 Adaptive Lasso . . . 16
2.4.3 SCAD penalty . . . 17
2.4.4 Summary and comments . . . 18
2.5 DP-glasso . . . 19
2.5.1 Background . . . 19
2.5.2 Algorithm and properties . . . 19
2.5.3 Summary and comments . . . 21
2.6 High-dimensional Undirected Graph Estimation (Huge) . . . 22
2.6.1 Background . . . 22
2.6.2 Features and properties . . . 22
2.6.3 `huge` computational requirements . . . 23
2.6.4 Expectations for `huge` . . . 24

3 Selection Criteria **25**
3.1 Selection criteria for all methods other than Huge . . . 25
3.1.1 Akaike information criterion (AIC) . . . 25
3.1.2 Bayesian information criterion (BIC) . . . 26
3.1.3 K-fold Cross Validation (CV) . . . 27
3.2 Selection criteria for Huge . . . 27
3.2.1 ric . . . 27
3.2.2 ebic . . . 28
3.2.3 stars . . . 28

4 Simulations **30**
4.1 The procedure of simulation . . . 30
4.2 Evaluation Standards . . . 32
4.2.1 True Positives . . . 32
4.2.2 True Negatives . . . 33
4.2.3 Average Sparsity Pattern Plot . . . 34
4.3 Simulation details . . . 35

4.3.1 Settings for the estimating methods . . . 35
4.3.2 Settings for the data samples . . . 37
4.3.3 Settings for the regularization path . . . 37
4.3.4 Low dimensional simulations . . . 38
4.3.5 High dimensional simulations . . . 39
4.3.6 Matrix Generating Scheme and Parameter Choices . . . 39

5 Results **43**
5.1 Results for Normally Distributed Data . . . 44
5.1.1 Low dimensional cases . . . 44
5.1.2 High dimensional cases . . . 48
5.2 Results for t-distributed Data . . . 65
5.2.1 Low dimensional cases . . . 65
5.2.2 High dimensional cases . . . 76
5.2.3 Nonparanormal transformation effects (for MVt_4 *data*) . . . 97
5.3 AR1 Auto-correlated *N-Data* (*MVN AR1 data*) . . . 109
5.3.1 Low dimensional cases . . . 109
5.3.2 High dimensional case . . . 112

6 Conclusions **120**
6.1 Computational speed comparisons . . . 120
6.2 The choice of π_{thr} and H in the BG algorithm . . . 121
6.2.1 How the resampling iterations H effects BG . . . 123
6.2.2 How the threshold value π_{thr} effects BG . . . 124
6.3 Estimation Accuracy Comparisons . . . 124
6.3.1 Preliminaries . . . 124
6.3.2 Overall Conclusions . . . 125
6.3.3 Conclusions for the multivariate normal data . . . 126
6.3.4 Conclusions for the t-distributed data . . . 127
6.3.5 Conclusions for the multivariate normal data with AR1 autocorrelation structure . . . 130

Bibliography **138**

Appendix A Tables **139**

A.1 Results for the tridiagonal matrix with $a = 0.85$ 140

A.1.1 *MVN data* ($p = 30$) results 140

A.1.2 *MVt*$_3$ *data* ($p = 30$) results 141

A.1.3 *MVt*$_4$ *data* ($p = 30$) results 142

A.1.4 *MVN AR1 data* ($p = 30$) results 144

A.2 Negative estimates frequencies 145

Appendix B Figures **147**

List of Tables

1 *Summary of the entries of the tridiagonal matrix with* $a = 0.85$ . . . 41
2 *Summary of the entries of the tridiagonal matrix with* $a = 1.7$ 41
3 *Summary of the entries of the exponential decay matrix* 41
4 *Summary of the entries' absolute values of the general matrix* 42

5 *The TPs for the* ***MVN data*** $(p = 5)$ *at* 4 *different sample sizes* 44
6 *The TNs for the* ***MVN data*** $(p = 5)$ *at* 4 *different sample sizes* 46
7 *The TPs and TNs for the tridiagonal matrix with* $a = 1.7$ *for the* ***MVN data*** $(p = 30)$ 62
8 *The TPs and TNs for the exponential decay matrix and the general matrix for the* ***MVN data*** $(p = 30)$ 63
9 *The TPs for the* ***MV***t_3 ***data*** $(p = 5)$ *at* 4 *different sample sizes* 65
10 *The TNs for the* ***MV***t_3 ***data*** $(p = 5)$ *at* 4 *different sample sizes* . . . 68
11 *The TPs for the* ***MV***t_4 ***data*** $(p = 5)$ *at* 4 *different sample sizes* 72
12 *The TPs for the* ***MV***t_4 ***data*** $(p = 5)$ *at* 4 *different sample sizes* 73
13 *The TPs and TNs for the tridiagonal matrix with* $a = 1.7$ *for the* ***MV***t_3 ***data*** $(p = 30)$ 90
14 *The TPs and TNs for the exponential decay matrix and the general matrix for the* ***MV***t_3 ***data*** $(p = 30)$ 91
15 *The TPs and TNs for the tridiagonal matrix with* $a = 1.7$ *for the* ***MV***t_4 ***data*** $(p = 30)$ 93
16 *The TPs and TNs for the tridiagonal matrix with* $a = 1.7$ *for the* ***MV***t_4 ***data*** $(p = 30)$ 94
17 *The TPs for the* ***MV***t_4 ***data*** $(p = 5)$ *using the nonparanormal transformation function* `huge.npn()` *at* 4 *different sample sizes* 97

18 *The TNs for the* ***MV*** t_4 ***data*** *($p = 5$) using the nonparanormal transformation function* `huge.npn()` *at 4 different sample sizes* 98

19 *The TPs and TNs for the tridiagonal matrix with $a = 1.7$ for the* ***MV*** t_4 ***data*** *($p = 30$) using the nonparanormal transformation function* `huge.npn()` 107

20 *The TPs and TNs for the exponential decay matrix and the general matrix for the* ***MV*** t_4 ***data*** *($p = 30$) using the nonparanormal transformation function* `huge.npn()` 108

21 *The TPs for the* ***MVN AR1 data*** *($p = 5$) at 4 different sample sizes* . 109

22 *The TNs for the* ***MVN AR1 data*** *($p = 5$) at 4 different sample sizes* . 110

23 *The TPs and TNs for the tridiagonal matrix with $a = 1.7$ for the* ***MVN AR1 data*** *($p = 30$)* 118

24 *The TPs and TNs for the exponential decay matrix and the general matrix for the* ***MVN AR1 data*** *($p = 30$)* 119

25 *The computational time of each combination for the* ***MVN data*** *($p = 5$) and the* ***MV*** t_3 ***data*** *($p = 5$) using an Intel(R) Core(TM) i3-2350 M 2.30GHz CPU* 121

26 *The TNs for the* ***MVN data*** *($p = 5$ and $n = 200$) using glasso and BG for $H = 50, 100, 150, 200$, $\pi_{thr} = 0.75$ and selection criteria AIC, BIC and CV* 122

27 *The TNs for the* ***MVN data*** *($p = 5$ and $n = 200$) using glasso and BG for $H = 50, 100, 150, 200$, $\pi_{thr} = 0.8$ and selection criteria AIC, BIC and CV* 122

28 *The TNs for the* ***MVN data*** *($p = 5$ and $n = 200$) using glasso and BG for $H = 50, 100, 150, 200$, $\pi_{thr} = 0.85$ and selection criteria AIC, BIC and CV* 122

29 *The TNs for the* ***MVN data*** *($p = 5$ and $n = 200$) using glasso and BG for $H = 50, 100, 150, 200$, $\pi_{thr} = 0.9$ and selection criteria AIC, BIC and CV* 123

30 *The TNs for the **MVN data** ($p = 5$ and $n = 200$) using glasso and BG for $H = 50, 100, 150, 200$, $\pi_{thr} = 0.95$ and selection criteria AIC, BIC and CV* 123

31 *The TPs and TNs for the tridiagonal matrix with $a = 0.85$ for the **MVN data** ($p = 30$)* 140

32 *The TPs and TNs for the tridiagonal matrix with $a = 0.85$ for the **MVt_3 data** ($p = 30$)* 141

33 *The TPs and TNs for the tridiagonal matrix with $a = 0.85$ for the **MVt_4 data** ($p = 30$)* 142

34 *The TPs and TNs for the tridiagonal matrix with $a = 0.85$ for the **MVt_4 data** ($p = 30$) using the nonparanormal transformation function* `huge.npn()` 143

35 *The TPs and TNs for the tridiagonal matrix with $a = 0.85$ for the **MVN AR1 data** ($p = 30$)* 144

36 *The frequencies of getting negative estimates on the main diagonal for the low dimensional datasets* 145

37 *The frequencies of getting negative estimates on the main diagonal for the high dimensional datasets* 146

List of Figures

1 *An example of an 8-dimensional undirected graph* 2

2 *The sparsity pattern plot of the tridiagonal matrix with* $a = 0.85$ *and* $a = 1.7$ 48

3 *The sparsity pattern plots of the exponential decay matrix* 48

4 *The sparsity pattern plots of the general matrix* 49

5 *The ASPs of the sample correlation matrix and the sample partial correlation matrix* 49

6 *The ASP plots for the* ***MVN data*** *(*$p = 30$*) generated from the tridiagonal matrix with* $a = 1.7$ 51

7 *The ASP plots for the* ***MVN data*** *(*$p = 30$*) generated from the tridiagonal matrix with* $a = 0.85$. 54

8 *The ASP plots for the* ***MVN data*** *(*$p = 30$*) generated from the exponential decay matrix* 57

9 *The ASP plots for the* ***MVN data*** *(*$p = 30$*) generated from the general matrix* 60

10 *The ASP plots for the* ***MV*** t_3 ***data*** *(*$p = 30$*) generated from the tridiagonal matrix with* $a = 1.7$*, using the wrong AIC/BIC formula* . . . 76

11 *The ASP plots for the* ***MV*** t_3 ***data*** *(*$p = 30$*) generated from the tridiagonal matrix with* $a = 1.7$*, using the correct AIC/BIC formula* . . . 77

12 *The ASP plots for the* ***MV*** t_3 ***data*** *(*$p = 30$*) generated from the tridiagonal matrix with* $a = 0.85$*, using the wrong AIC/BIC formula.* 78

13 *The ASP plots for the* ***MV*** t_3 ***data*** *($p = 30$) generated from the tridiagonal matrix with $a = 0.85$, using the correct AIC/BIC formula.* 79

14 *The ASP plots for the* ***MV*** t_3 ***data*** *($p = 30$) generated from the exponential decay matrix, using the wrong AIC/BIC formula* 80

15 *The ASP plots for the* ***MV*** t_3 ***data*** *($p = 30$) generated from the exponential decay matrix, using the correct AIC/BIC formula* 81

16 *The ASP plots for the* ***MV*** t_3 ***data*** *($p = 30$) generated from the general matrix, using the wrong AIC/BIC formula* 82

17 *The ASP plots for the* ***MV*** t_3 ***data*** *($p = 30$) generated from the general matrix, using the correct AIC/BIC formula* 83

18 *The ASP plots for the* ***MV*** t_4 ***data*** *($p = 30$) generated from the tridiagonal matrix with $a = 1.7$, using the nonparanormal transformation function* `huge.npn()` *and the wrong AIC/BIC formula* 101

19 *The ASP plots for the* ***MV*** t_4***-data*** *($p = 30$) generated from the tridiagonal matrix with $a = 0.85$, using the nonparanormal transformation function* `huge.npn()` *and the wrong AIC/BIC formula.* 102

20 *The ASP plots for the* ***MV*** t_4 ***data*** *($p = 30$) generated from the exponential decay matrix, using the nonparanormal transformation function* `huge.npn()` *and the wrong AIC/BIC formula* 103

21 *The ASP plots for the* ***MV*** t_4 ***data*** *($p = 30$) generated from the general matrix, using the nonparanormal transformation function* `huge.npn()` *and the wrong AIC/BIC formula* 104

22 *The ASP plots for the* ***MVN AR1 data*** *($p = 30$) generated from the tridiagonal matrix with $a = 1.7$* 112

23 *The ASP plots for the tridiagonal matrix with $a = 0.85$ for the* ***MVN AR1 data*** *($p = 30$)* 113

24 *The ASP plots for the* ***MVN AR1 data*** *($p = 30$) generated from the exponential decay matrix* 114

25 *The ASP plots for the* ***MVN AR1 data*** *($p = 30$) generated from the general matrix* 115

26 *The ASP plots for the* ***MV****t*$_4$ ***data*** *($p = 30$) generated from the tridiagonal matrix with $a = 1.7$, using the wrong AIC/BIC formula* . . . 148
27 *The ASP plots for the* ***MV****t*$_4$ ***data*** *($p = 30$) generated from the tridiagonal matrix with $a = 1.7$, using the correct AIC/BIC formula* . . . 149
28 *The ASP plots for the* ***MV****t*$_4$ ***data*** *($p = 30$) generated from the exponential decay matrix, using the wrong AIC/BIC formula* 150
29 *The ASP plots for the* ***MV****t*$_4$ ***data*** *($p = 30$) generated from the exponential decay matrix, using the correct AIC/BIC formula* 151
30 *The ASP plots for the* ***MV****t*$_4$ ***data*** *($p = 30$) generated from the general matrix, using the wrong AIC/BIC formula* 152
31 *The ASP plots for the* ***MV****t*$_4$ ***data*** *($p = 30$) generated from the general matrix, using the correct AIC/BIC formula* 153

Chapter 1

Introduction

Graphical models have been extensively used to explore the underlying relationships between a set of variables, whose nodes represent the variables and the edges represent the dependence between them. For example, in neuroscience, a network can be estimated between various functioning regions of the brain (Cribben et al. [2012]). The portfolio of various financial assets can also be described using a graphical model where the nodes are the assets and the edges depict the interaction between their rates of return. A sparse graphical model is usually preferred in practice for its simplicity and ease of interpretation.

The work carried out here is concerned with the estimation of the sparse precision matrix (inverse of the covariance matrix) $\Omega = (\omega_{ij})_{1\leq i,j\leq p}$ of p variables $\mathbf{X}_{n\times p} = (X_1, ..., X_p)$, which can be illustrated by an undirected graphical model. In the graphical model, the p nodes denote the p variables $X_1, ..., X_p$, respectively, and the edge between the ith and jth nodes correponds to the (i, j)th entry ω_{ij} of the precision matrix Ω. The larger the entries in the precision matrix Ω, the thicker the edges in the undirected graph, indicating stronger conditional dependence between the corresponding variables. The absence of an edge between two nodes in the undirected graph indicates conditional independence between the corresponding variables. The entries of the standardized precision matrix are the partial correlations of the set of variables, which quantify their dependence with the influence from all other variables removed. An example of an 8-dimensional estimated

graphical model or precision matrix is given in Figure 1. To interpret this graph, for example, the edge between nodes 3 and 4 is thicker than the edge between nodes 2 and 6, indicating that the conditional dependence between the third and fourth variables is stronger than the conditional dependence between the second and and sixth variables. Also, since there is no edge between nodes 1 and 8, we can say that these two variables are conditionally independent.

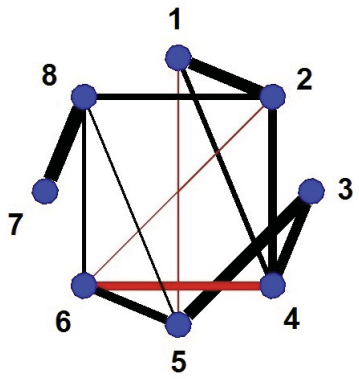

Figure 1: *An example of an* 8*-dimensional undirected graph*

Pourahmadi [2011] pointed out that precision matrices are of more interest than covariance matrices. In addition, it has been shown that the regularized precision matrix is sensitive to capturing the network connection on good quality functional magenetic resonance imaging (fMRI) data (Smith et al. [2011]). Thus, the central theme of this work is the estimation of sparse standardized precision matrices, whose results can be illustrated by undirected graphs.

There have been many estimating methods for estimating sparse precision matrices and many of them are penalized log-likelihood methods. A foundational work on this topic is the l_1-penalized method glasso introduced by Friedman et al. [2008] which is very fast, easy to implement, and produces sparse estimates of the precision matrix. The newly developed algorithm by Mazumder and Hastie [2012b], called the DP-glasso, differs from glasso in that it solves the primal penalized log-likelihood while glasso solves the dual problem. The estimating method, the bootsrap glasso (BG), inspired by the stability selection technique of Meinshausen and Bühlmann [2010], combines both glasso and bootstrap resampling (Efron and Tib-

shirani [1993]) to improve the estimation performance. The estimating methods Adaptive Lasso (AL) and the Smoothly Clipped Absolute Deviation (SCAD) are modifications of glasso in the sense that the original l_1-penalty is replaced respectively by the Adaptive Lasso penalty (Zou [2006]) and the SCAD penalty (Fan and Li [2001]), both of which are non-convex penalties. In addition, Zhao et al. [2014] introduced the estimating method High-dimensional Undirected Graph Estimation (Huge) and a companion `R` package `huge`, which integrates many functions about estimating graphical models such as semiparametric transformation, graph estimation and model selection.

As well as the several estimating methods, there also exist many selection criteria to select the optimal estimate among the candidates. Akaike [1974] proposed the well-known Akaike Information Criterion (AIC) to select the potential optimal model out of a collection of candidate models. Schwarz [1978] proposed the Bayesian Information Criterion (BIC) which can consistently select the optimal model. Also, Fan et al. [2009] employed a K-fold Cross Validation score (see (3.3) to follow) to conduct selection of the optimal model. Moreover, the method Huge is accompanied with three selection criteria Rotation Information Criterion (ric, Lysen [2009]), Extended Bayesian Information Criterion (ebic, Foygel and Drton [2010]) and Stability Approach for Regularization Selection (stars, Liu et al. [2010]), all of which are provied by the function `huge.select()` of the package `huge` (Zhao et al. [2012] and Zhao et al. [2014]).

The theoretical properties of the aforementioned estimating methods and selection criteria have been explored thoroughly, however, the practical performance of the combinations of the estimating methods with selection criteria remains unclear. More specifically, it is not known which combination performs the best for the datasets with different sample sizes, dimension, sparsities, and distributions. Also, no previous comparisons were conducted among the best results that combinations can provide along a given set of regularization parameters ρs. The previous studies only focused on the results for several fixed ρs. Therefore, the major contribution of this work is: (1) to apply all the combinations (of the above estimating methods and selection criteria) to three types of simulated data: multivariate normal data (***MVN***

data), multivariate t-distributed data with 3 and 4 degrees of freedom (***MV***t_3 ***data*** and ***MV***t_4 ***data***) and the multivariate normal data with an AR(1) structure (***MVN AR1 data***), using 100 regularization parameters ρs, and (2) to conduct a comparative evaluation of the best performance that each combination can produce based on the 100 ρs, under different simulation scenarios.

The rest of this work is organized as follows. Chapter 2 and Chapter 3 explain the theoretical background and interesting features of the estimating methods and selection criteria. Chapter 4 is devoted to the simulation descriptions, and the results from the simulations are discussed in Chapter 5. Finally, Chapter 6 provides a set of conclusions based on the simulation results. The chapters are accompanied with Appendix A Tables and Appendix B Figures which contains a set of tables and figures from the simulation results.

Chapter 2

Estimating Methods

Consider a p dimensional dataset

$$\mathbf{X}_{n\times p} = (X_1, X_2, \ldots, X_p)$$

with mean $\mu_{p\times 1}$ and a positive definite covariance matrix $\Sigma_{p\times p} = (\sigma_{ij})_{1\leq i,j\leq p}$, and

$$X_i = (X_{i1}, X_{i2}, \ldots, X_{in}),\ i = 1, ..., p,$$

where n is the sample size. The (i,j)th entry σ_{ij} of a covariance matrix $\Sigma_{p\times p}$ is the covariance between X_i and X_j, where

$$\sigma_{ij} = \mathrm{Cov}(X_i, X_j) = \mathbb{E}\left((X_i - \mu_i)(X_j - \mu_j)\right).$$

The (i,j)th entry of the standardized covariance matrix is the correlation coefficients between X_i and X_j.

The sample covariance matrix $\mathbf{S}_{p\times p} = (s_{ij})_{1\leq i,j\leq p}$ is an empirical statistic calculated from a sample dataset $\mathbf{X}_{n\times p}$, whose (i,j)th entry s_{ij} is the sample covariance between the set of observations of X_i and X_j. s_{ij} is calculated by

$$s_{ij} = \frac{1}{n-1}\sum_{q=1}^{n}(x_{iq} - \bar{x}_i)(x_{jq} - \bar{x}_j), \tag{2.1}$$

where n is the number of observations of X_i and X_j, or the sample size of X_i and X_j. The precision matrix $\Omega = (\omega_{ij})_{1\leq i,j\leq p}$ is the inverse of the covariance matrix Σ and the (i,j)th entry of the standardized precision matrix is the partial correlation between X_i and X_j.

The first step of our work is to estimate the precision matrix $\Omega = \Sigma^{-1}$, and we denote the estimates of Ω and Σ as $\widehat{\Omega}$ and $\widehat{\Sigma}$, respectively. Seven methods are applied in this work, including the sample correlation matrix $\mathbf{C}_S$, the sample partial correlation matrix $\mathbf{P}_S$ (these two are simply used as reference methods), Graphical Lasso (glasso), Bootstrape Graphical Lasso (BG), Graphical Lasso with Adaptive Lasso Penalties (AL), Graphical Lasso with SCAD penalties (SCAD) and High-dimensional Undirected Graph Estimation (Huge).

glasso, BG, AL and SCAD are all penalized log-likelihood methods, who add various weighted l_1-penalties to the log-likelihood formula. Hence the estimate of Ω is the solution to the following formula:

$$\max_{\Omega} \log\det\Omega - \mathrm{tr}(\mathbf{S}\Omega) - \sum_{i=1}^{p}\sum_{j=1}^{p} p_{\rho_{ij}}\left(|\omega_{ij}|\right), \tag{2.2}$$

where *tr* denotes the trace of a matrix which is the sum of all elements on the main diagonal, and $p_{\rho_{ij}}(\cdot)$ is the penalty function on ω_{ij} with ρ_{ij} being the corresponding regularization parameter that controls the sparsity level.

2.1 Sample (Partial) Correlation Matrix

2.1.1 Sample correlation matrix

The (i,j)th element, r_{ij}, of the sample correl ation matrix $\mathbf{C}_S = (r_{ij})_{i,j\leq p}$ is the sample correlation between the ith and the jth variable X_i and X_j, who measures the direction and strength of the linear relationship between them. r_{ij} is calculated using

$$r_{ij} = \frac{\sum_{q=1}^{n}\left(x_{iq}-\bar{x}_i\right)\left(x_{jq}-\bar{x}_j\right)}{\sqrt{\sum_{q=1}^{n}\left(x_{iq}-\bar{x}_i\right)^2\sum_{q=1}^{n}\left(x_{jq}-\bar{x}_j\right)^2}}, \tag{2.3}$$

where n is the sample size of variables X_i and X_j.

2.1.2 Sample Partial Correlation Matrix

The (i, j)th element, γ_{ij}, of the sample partial correlation matrix $\mathbf{P}_S$ is the sample partial correlation between X_i and X_j, measuring the relationship between them while controlling for the other variables and indicating the conditional dependence between these two variables. Partial correlation γ_{ij} becomes more essential when other variables are very likely to have effects on X_i and X_j. In addition, γ_{ij} can be obtained from the corresponding elements in the precision matrix Ω ([Pourahmadi, 2011, pp5]) by

$$\gamma_{ij} = -\frac{\omega_{ij}}{\sqrt{\omega_{ii}\omega_{jj}}}. \tag{2.4}$$

2.2 Graphical Lasso

2.2.1 Background

The Lasso (l_1-) penalty proposed by Tibshirani [1996] has been widely used to estimate sparse undirected graphs. For example, Meinshausen and Bühlmann [2006], provided an approximation for the exact problem in equation (2.5) below which fits a Lasso model to each variable, using the others as predictors. Besides this approximation, algorithms for the exact problem were also proposed, such as [Yuan and Lin, 2007, pp1-2], Banerjee et al. [2008] and Dahl et al. [2008]. After applying the Lasso (l_1-) penalty on Ω, the penalized log-likelihood (2.2) becomes

$$\max_{\Omega} \log\det\Omega - \mathrm{tr}(\mathbf{S}\Omega) - \rho\|\Omega\|_1, \tag{2.5}$$

where $\|\Omega\|_1$ denotes the Lasso (l_1-) penalty on Ω, the sum of all absolute values of the elements of Ω.

Friedman et al. [2008] solved (2.5) using the blockwise coordinate descent approach (Banerjee et al. [2008]), resulting in a simple, yet extremely fast approach, the Graphical Lasso (glasso). In the graphical modeling framework, some elements

of $\widehat{\Omega}$ can be shrunk exactly to zeros by the glasso algorithm due to the l_1-penalty on Ω, because in order to maximize (2.5), $\rho\|\Omega\|_1$ has to be small so that the sum of all absolute values of Ω should be small for a fixed ρ, and thereby some entries of Ω are supposed to be zeros.

2.2.2 glasso algorithm

glasso takes turns to fit a modified lasso regression to each variable, which is solved by coordinate descent. Friedman et al. [2008] also pointed out that the solution to (2.5) and its dual problem (2.6) are equivalent :

$$\min_{\beta}\left\{\frac{1}{2}\|\mathbf{W}_{11}^{1/2}\beta - b\|^2 + \rho\|\beta\|_1\right\}, \tag{2.6}$$

where $\mathbf{W} = \widehat{\Sigma}$, $b = \mathbf{W}_{11}^{-1/2}s_{12}$, $\mathbf{W}$ and $\mathbf{S}$ have the partition as

$$\mathbf{W} = \begin{pmatrix} \mathbf{W}_{11} & \boldsymbol{\omega_{12}} \\ \Omega_{12}^T & \Omega_{22} \end{pmatrix}, \ \mathbf{S} = \begin{pmatrix} \mathbf{S}_{11} & \mathbf{s_{12}} \\ \mathbf{s}_{12}^T & \mathbf{s}_{22} \end{pmatrix}. \tag{2.7}$$

Then, the glasso algorithm ([Friedman et al., 2008, P5]) works as follows:

Step 1 Start with $\mathbf{W} = \mathbf{S} + \rho\mathbf{I}$. The diagonal of $\mathbf{W}$ remains unchanged in what follows.

Step 2 For each $j = 1, 2, \ldots, p$, solves the lasso problem (the dual problem)

$$\min_{\beta}\left\{\frac{1}{2}\|\mathbf{W}_{11}^{1/2}\beta - b\|^2 + \rho\|\beta\|_1\right\},$$

which takes as input the inner products $\mathbf{W}_{11}$ and $\mathbf{s}_{12}$. This gives a $p-1$ vector solution $\hat{\beta}$. Fill in the corresponding row and column of $\mathbf{W}$ using $\boldsymbol{\omega_{12}} = \mathbf{W}_{11}\hat{\beta}$.

Step 3 Continue until convergence.

2.2.3 The R package glasso

The R package glasso (Friedman et al. [2014]) can be downloaded to conduct the above glasso algorithm. $\mathbf{S}$ and ρ are two of its inputs. One can also specify the maximum number of iterations of the outer loop (default $10,000$), the type of start (starting values for estimating Σ and Ω, with default cold start being $\mathbf{S}+\rho\mathbf{I}$; another option, warm start, provides a customized starting value), and threshold for convergence (default $1e-4$), in the package . The glasso package can return $\widehat{\Sigma}$, $\widehat{\Omega}$, the maximized value from equation (2.5) and the number of iterations of the outer loop used by algorithm, and many other outputs.

We denote the estimated precision matrix and covariance matrix from glasso as $\widehat{\Omega}_G$ and $\widehat{\Sigma}_G$, respectively.

2.2.4 Highlights and deficiencies

Generally, glasso is a simple and an efficient algorithm for estimating a sparse Ω. In addition, the popular package 'glasso' in R makes it convenient to implement. More specifically, as discussed in Friedman et al. [2008], glasso provides sparse estimates, because the l_1 penalty shrinks some elements of Ω exactly to zero. Also, another attractive feature of glasso is its desirable computational speed. For example, it has been shown in Friedman et al. [2008] that based on an Intel Xeon 2.80GH processor, for example, with 2 to 8 iterations of the outer loop, $p = 400$ in a sparse Σ case, it only takes glasso 1.23 seconds (the CPU time spent in the C program since glasso was coded in Fortran and linked to an R language function) to get the results. Another advantage of glasso is the positive definiteness of each updated Σ is ensured.

However, as can be seen from its detailed algorithm above, glasso essentially deals with the dual problem (2.6) in each step, rather than the primal penalized log-likelihood (2.5) itself. In the dual problem, the optimized variable is Σ rather than Ω. Thus there are some consequences of operating on the dual problem whose target is the covariance matrix and some of glasso's main deficiencies are as follows:

1. The penalized log-likelihood of the primal problem does not monotonically

increase with the iteration index, whereas the penalized log-likelihood of its dual problem does. Moreover, divergence of the algorithm may occur with undesirable warm starts.

2. It is very likely that the converged $\widehat{\Omega}$ is not equal to $\widehat{\Sigma}^{-1}$, which means $\widehat{\Omega}$ is not numerically the inverse of $\widehat{\Sigma}$ and this undoubtedly contradicts their mathematical relationship. It has also been shown that only upon (asymptotic) convergence, will $\widehat{\Omega}$ be equal to Σ^{-1}.

3. $\widehat{\Omega}$ is not necessarily positive definite, because the optimized matrix in glasso algorithms is Σ and $\widehat{\Omega}$ is not the exact inverse of $\widehat{\Sigma}$.

4. $\widehat{\Omega}$ need not to be sparse.

5. The positive definiteness of a matrix after every update step does not ensure the positive definiteness of it after the final update. Here is a short example. Adopt the $p = 30$ tridiagonal matrix with constant $a = 0.85$ being the true precision matrix Ω. Then `set.seed(136)` in `R` and generate a $n = 120$ multivariate normal distributed dataset from it (or say from its inverse, the convariance matrix $\Sigma = \Omega^{-1}$). Obtain the best glasso estimated matrix by BIC first, then use it as the initial value for iterations for SCAD algorithm and selected by BIC. Then at the 5th iteration step for by SCAD penalty, the estimated covariance matrix from glasso is not positive definite who has a negative eigenvalue (-0.02189027).

2.2.4.1 Comments and summaries

glasso is a basic and fast algorithm for estimating a sparse Ω. It applies a l_1-penalty to Ω which effectively shrink some ω_{ij} estimates to zero. Moreover, the `glasso` package in `R` makes the algorithm practical. Subsequently, many approaches have evolved from it. Although $\widehat{\Omega}_G$ shows some undesirable properties, the advantages and importance of glasso still can not be neglected.

For further improvement and comparison, we introduce several more methods in the following pages.

2.3 Bootstrap glasso

2.3.1 Background

The bootstrap glasso (BG) is a hybridized algorithm of glasso and the bootstrap (resampling scheme), which is inspired by a technique called Stability Selection (SS) attributing to Meinshausen and Bühlmann [2010]. BG was first introduced in Cribben et al. [2013] and Cribben [2012]. We will introduce SS and its properties first, and then introduce our BG algorithm.

2.3.2 Stability selection (SS)

SS combines subsampling with existing high-dimensional structural selection schemes. Meinshausen and Bühlmann [2010] pointed out that SS is not a new variable selection technique, but simply aims at heightening existing methods, such as variable selection methods, graphical modeling methods or cluster analysis methods. However, SS results can not be reconstructed by choosing a proper ρ, since SS provides a basically new solution.

Unlike traditional approaches, SS does not choose one best model among the whole regularization path, instead, SS disturbs (e.g. subsampling) the original dataset a number of times, then keeps structures or variables whose occurrences reach to a certain threshold level. More specifically, estimates with high selection probabilities are maintained, namely whose selection probabilities are greater than a prechosen cutoff $0 < \pi_{thr} < 1$, otherwise they will be abandoned.

SS has many good properties as follows:

1. *Error Control Theorem* ([Meinshausen and Bühlmann, 2010, Theorem 1]): Let V denote the number of false selections. Thus $\mathbb{E}(V)$ stands for the expected value of the number of false selections and $\bar{V}$ represents the average number of falsely selected variables.

   A strong benefit of SS is that, under certain assumptions, it can choose ρs such that $\mathbb{E}(V)$ is bounded. Additionally, the assumptions have been justified that they are either reasonably easy to fulfill, or only slightly weaker

results will be obtained if some of them are violated. Moreover, one of the assumptions is only needed for controlling the error and unnecessary for other advantages of SS. This theorem also indicates that choosing less variables or increasing π_{thr} will reduce $\mathbb{E}(V)$ with an achievable minimal value of $1/p^2$.
The bounded expected value for the number of false selections is a very advantageous property, especially for high-dimensional problems, because a certain limit for finite sample is more practical than asymptotic properties, requiring tending to infinity sample sizes, which is almost impossible in real life.

2. *Consistent Variable Selection*: Herein, the consistent variable selection for a procedure $\widehat{S}$ means that the probability of the procedure $\widehat{S}$ being the same as the true procedure S tends to 1, as the sample sizes increase to infinity. Mathematically, it means

$$\mathbb{P}\left(\widehat{S} = S\right) \to 1 \text{ as } n \to \infty, \tag{2.8}$$

where S is the true value and n is the sample size.
For randomized Lasso, a generalisation of the Lasso, where the Lasso penalises Ω proportional to the sum of all its absolute values with the overall regularization parameter being ρ, while the randomised Lasso changes the penalty for each entry of Ω to a randomly chosen value in the range $[\rho, \rho/\alpha]$, it has been shown that SS will consistently choose variables even if some required conditions for consistency selection are violated for the original method, which means SS will asymptotically choose the correct model when randomized Lasso can't.
This is an benefit for the original approach, especially computational efficient ones, because they always can't select variales consistently in high-dimensional cases, even in fairly simple settings.

2.3.3 BG algorithm

We borrow the subsampling idea from SS, applying it to glasso, leading to our BG algorithm. The notations for BG are as follows:

1. H denotes the resampling time.

2. $\mathbf{X}_B$ denotes the resampled datasets with $\mathbf{X}_B^h$ being the hth resampled dataset, $h = 1, 2, \ldots, H$.

3. $\widehat{\Omega}_G^h$ is the glasso estimate based on the hth resampled dataset $\mathbf{X}_B^h$, $h = 1, 2, \ldots, H$.

4. $\mathbf{Z}$ is the non-zero frequency estimates matrix for each resampled dataset and $\mathbf{Z}^h$ is the non-zero estimate frequency matrix for $\mathbf{X}_B^h$ defined as

$$\mathbf{Z}^h(i,j) = \begin{cases} 1 & \text{if } \widehat{\Omega}_G^h(i,j) \neq 0 \\ 0 & \text{if } \widehat{\Omega}_G^h(i,j) = 0 \end{cases}, \; h \in \{1, 2, ..., H\}, \tag{2.9}$$

5. which indicates that if the (i, j)th element in the glasso estimate $\widehat{\Omega}_G^h$ for the hth resampled dataset is non-zero, the corresponding (i, j)th value in hth non-zero estimate frequency matrix is set to 1, otherwise it is 0. $\mathbf{Z}^h$ describes whether each position of Ω is estimated as non-zero or zero, based on the corresponding resampled dataset $\mathbf{X}_B^h$.

6. $\mathbf{F}$ is the overall non-zero frequency matrix, based on all the resampled dataset $\mathbf{X}_B^h$, $h = 1, 2, \ldots, H$, defined as

$$\mathbf{F}(i,j) = \frac{1}{H} \sum_{h=1}^{H} \mathbf{Z}^h(i,j), \; i, j = 1, 2, ..., p. \tag{2.10}$$

7. $\mathbf{F}$ is the average frequency of a non-zero estimate for each position in Ω after

resampling H times. For example, if $\mathbf{F}(i,j) = 0.9$, then 90% of the glasso re-estimations are non-zero for $\Omega(i,j)$ among all the resamplings.

8. π_{thr} is the prechosen threshold value which is usually set to 0.75 to 0.9.

9. $\widehat{\Omega}_{BG}$ is the BG estimate for Ω.

The BG algorithm executes the following four steps for each ρ_i, $i = 1, ..., 100$:

Step 1 Apply glasso to the original dataset $\mathbf{X}_{n\times p}$ and obtain glasso estimate $\widehat{\Omega}_G$.

Step 2 Resample $\mathbf{X}_{n\times p}$ H times without changing sample size, obtaining H resampled datasets, say $\mathbf{X}_B = (\mathbf{X}_B^h)_{1\le h\le H}$.

Step 3 Apply glasso to each resampled dataset $\mathbf{X}_B^h$ and obtain a new glasso estimate $\widehat{\Omega}_G^h$, $h = 1, 2, \ldots, H$.

Step 4 The estimate $\widehat{\Omega}_{BG}$ by BG is obtained by setting

$$\widehat{\Omega}_{BG}(i,j) = \begin{cases} \widehat{\Omega}_G(i,j), & \text{if } \mathbf{F}(i,j) \ge \pi_{thr} \\ 0, & \text{if } \mathbf{F}(i,j) < \pi_{thr} \end{cases}, \tag{2.11}$$

where $\mathbf{F}$ is the overall non-zero frequency matrix calculated by (2.10).

Intuitively, $\widehat{\Omega}_{BG}$ is at least as sparse as $\widehat{\Omega}_G$, since $\widehat{\Omega}_{BG}$ only retains those non-zero elements of $\widehat{\Omega}_G$ which are always estimated as non-zeros, and abandons those estimates in $\widehat{\Omega}_G$ that are estimated as zeros with relatively high frequencies, which can reasonably be believed as zeros in the true matrix.

2.3.4 Summary and comments

Instead of choosing the best regularization parameter, SS offers a new view for structural estimations by combining subsampling with primal methods, producing many desirable properties. We use glasso to be the initial structure estimation method, then apply SS to it, resulting in our new algorithm Bootstrap glasso (BG). Based on the above theoretical properties of SS, we have the following expectations for BG:

1. The results for BG won't vary much with π_{thr}.
2. The percentage of false estimates (estimates for zeros being non-zeros, or estimates for non-zeros being zeros) is controlled, or bounded.
3. BG provides accuracy improvements over the primal method (glasso).
4. However, a disadvantage is that it is more time consuming than the original estimation method (glasso).

2.4 glasso with Adaptive Lasso and SCAD penalties

2.4.1 Background

2.4.1.1 Motivations

Sparse precision matrices are used to explore network structures. Two major challenges for estimating a sparse precision matrix are the requirement of positive definiteness of Ω when optimizing the penalized log-likelihood, and the reduction of the biases arising from the penalties. For example, our first method glasso using the Lasso penalty on Ω, it has been shown in Fan and Li [2001] that the Lasso penalty increases linearly with the magnitude of the regression coefficients, thus a mass of biases will be induced when having a lot regression coefficients. To remedy this, the nonconcave Smoothly Clipped Absolute Deviation (SCAD) penalty and the Adaptive Lasso (AL) penalty, were proposed in Fan and Li [2001] and Zou [2006], respectively.

2.4.1.2 Improved properties by AL and SCAD penalties

Although glasso can achieve sparsity for the estimated precision matrix, it produces substantial biases. However, improvements can be achieved by the two newly proposed penalties, the AL penalty and the SCAD penalty, which achieve the following three desirable properties simultaneously:

1. sparse estimations
2. consistent model selections
3. unbiased estimates for large coefficients

Necessary conditions can be found in Fan and Li [2001] for any penalty functions that achieve the above three properties.

The optimal estimate of Ω by AL, along a given regularization path, is denoted by $\widehat{\Omega}_{AL}$ and the best SCAD estimate is denoted by $\widehat{\Omega}_{SCAD}$.

2.4.2 Adaptive Lasso

The AL assigns various weights to each Ω element, which depend on the magnitude of elements of a particularly chosen consistent estimate $\widehat{\Omega}$. Here the larger the element the smaller the weight. Thus essentially the AL is a properly weighted version of glasso.

After applying the AL penalty, mathematically the penalized log-likelihood becomes

$$\max_{\Omega} \log\det\Omega - \mathrm{tr}(\mathbf{S}\Omega) - \rho \sum_{i=1}^{p}\sum_{j=1}^{p} \omega_{ij}\left(|\omega_{ij}|\right), \tag{2.12}$$

where w_{ij} is the adaptive weight function (penalty function) and $\hat{\omega}_{ij}$ is an estimate for $\Omega(i,j)$. Fan et al. [2009] defined the adaptive weights to be

$$w_{ij} = 1/|\widetilde{\omega}_{ij}|^{\gamma} \tag{2.13}$$

for some tuning parameters $\gamma > 0$, where $\widetilde{\omega}_{ij}$ is the (i,j)th entry for any consistent estimate $\widetilde{\Omega} = (\widetilde{\omega}_{ij})_{1\le i,j\le p}$.

AL has good asymptotic properties. Fan and Li [2001] showed the following oracle property of AL:

1. Asymptotically, the estimate $\widehat{\Omega}$ of Ω has the same sparsity pattern as Ω.

2. The nonzero entries in $\widehat{\Omega}$ are $\sqrt{n}$-consistent and asymptotically normal.

[Fan et al., 2009, Theorem 5.3] showed that when $\sqrt{n}\rho = O_\rho(1)$ and $\rho\sqrt{n}a_n^\gamma \to \infty$, the above oracle property is achieved by the AL penalty with weights being (2.13) for some $\gamma > 0$ and any estimate $\widetilde{\Omega} = (\widetilde{\omega_{ij}})_{1\le i,j\le p}$ that is a_n-consistent, i.e. $a_n(\widetilde{\Omega} - \Omega) = O_\rho(1)$.

2.4.3 SCAD penalty

By applying the approach of local linear approximation (LLA, Zou and Li [2008]) to the SCAD penalty, the original nonconcave penalized log-likelihood is transformed into a series of weighted Lasso penalized log-likelihood problems, where the weights are controlled by the derivative of the SCAD penalty function. Thus, optimizing the penalized log-likelihood subject to a positive definite Ω can be nicely solved by our efficient first algorithm glasso iteratively. Consequently, the bias is well controlled without losing computational efficiency.

Mathematically, the SCAD penalty is symmetric and a quadratic spline on $[0,\infty)$, whose first order derivative is

$$\mathrm{SCAD}'_{\rho,a}(x) = \rho\left(I(|x|\le\rho) + \frac{(a\lambda - |x|)_+}{(a-1)\rho} I(|x|>\rho)\right), \tag{2.14}$$

for $x \ge 0$, where I is an indicator function, with $\rho > 0$ and $a > 2$ being two tuning parameters. If $a = \infty$, (2.14) becomes the Lasso penalty.

Applying the SCAD penalty, the log-likelihood (2.2) becomes

$$\max_{\Omega} \log\det\Omega - \mathrm{tr}(\mathbf{S}\Omega) - \sum_{i=1}^{p}\sum_{j=1}^{p} \mathrm{SCAD}_{\rho,a}\left(|\omega_{ij}|\right), \tag{2.15}$$

where we use $\rho_{ij} = \rho$ for convenience.

For SCAD, the original penalized log-likelihood (2.15) is iteratively optimized. In each step, in order to take the advantage of glasso, the SCAD penalty is locally linearly approximated (LLA). By the Taylor expansion, for any ω_0, $p_\rho(|\omega|)$ can be approximated in a neighborhood of $|\omega_0|$ by

$$p_\rho(|\omega|) \approx p_\rho(|\omega_0|) + p'_\rho(|\omega_0|)(|\omega| - |\omega_0|), \tag{2.16}$$

where $p'_\rho(|\omega|) = \frac{\partial}{\partial\omega} p_\rho(\omega)$. Note that (2.16) is nonnegative for $\omega \in [0,\infty)$.

At step k, denoting the glasso estimated precision matrix by $\widehat{\Omega}_G^{(k)}$, the penalized log-likelihood is up to a constant,

$$\max_{\Omega} \log\det\Omega - \mathrm{tr}(\mathbf{S}\Omega) - \sum_{i=1}^{p}\sum_{j=1}^{p} p_\rho^{'}(|\widehat{\omega}_{ij}^{(k)}|), \tag{2.17}$$

where $\widehat{\omega}_{ij}^{(k)}$ is the (i,j)-element of $\widehat{\Omega}_G^{(k)}$. glasso can solve the above function efficiently that lead to sparse solutions.

For SCAD, an estimated zero in $\widehat{\Omega}$ in one step does not necessarily mean it is a zero in the next iteration step, whereas for AL, zero estimates will remain zeros for in each iteration step, hence the initial value is always denser the estimates.

[Fan et al., 2009, Theorem 5.1] showed that for a differentiable concave penalty function on $[0,\infty)$, the penalized log-likelihood function monotonically increases with iteration indexes by LLA algorithm, which is the aforementioned algorithm applied to the SCAD penalty when maximize the penalized log-likelihood iteratively.

[Fan et al., 2009, Theorem 5.2] showed that when $\rho \to 0$ and $\sqrt{n}\rho \to \infty$ as $n \to \infty$, the oracle property of AL estimates in Subsection 2.4.2 also holds for the SCAD estimate $\widehat{\Omega}_{SCAD}$ which optimizes (2.15).

2.4.4 Summary and comments

The AL and SCAD penalties are considered improvements over the Lasso penalty. AL and SCAD can achieve sparse estimates, consistent model selection and unbiased estimates simultaneously, some of are not achieved by glasso. However, the basic yet fast glasso algorithm can still be applied to AL and SCAD penalties to estimate sparse Ω conveniently, as long as the SCAD penalty is locally linearly approximated.

2.5 DP-glasso

2.5.1 Background

As already noted, glasso is a popular and efficient algorithm for estimating undirected graphs. However, glasso operates on the dual problem of (2.5) with the target estimation matrix being the covariance matrix Σ, rather than the primal problem itself, which results in many undesirable outcomes. Consequently, Mazumder and Hastie proposed a new method called DP-glasso in Mazumder and Hastie [2012b], which also operates by block coordinate descent, yet directly solves the primal problem whose optimized matrix is the precision matrix Ω, not the covariance matrix. Several advantages come from this new algorithm. Additionally, a `R` package called `dpglasso` allows us to implement this algorithm, and compare it with the other existing methods.

In the `glasso` package, the input regularization parameter ρ can be a scalar and a matrix, thus the AL and SCAD methods can be implemented easily by `glasso`. However, unlike `glasso`, the regularization parameter required by `dpglasso` package has to be a scalar (matrices unallowed), which means the AL and SCAD cannot take advantage of DP-glasso algorithm directly. Therefore, under the constraint of inputting ρ, in order to see the improvements for AL and SCAD by the DP-glasso algorithm, we use the DP-glasso estimated precision matrices as initial values for AL and SCAD, while the `glasso()` function is used in inner steps of AL and SCAD, leading to another two evolved algorithms, called DP-AL and DP-SCAD. For BG,

`dpglasso()` is applied in every step where estimating precision matrices is needed, resulting in another algorithm DP-BG.

2.5.2 Algorithm and properties

The algorithm of DP-glasso ([Mazumder and Hastie, 2012b, Algorithm 3]) is as follows:

Step 1 Initialize $\Omega = \mathrm{diag}(\mathbf{S}+\rho\mathbf{I})^{-1}$.

Step 2 Cycle around the columns repeatedly, performing the following steps till convergence:

Step 2-1 Rearrange the rows/columns so that the target column is last (implicitly).

Step 2-2 Solve (2.18) for $\tilde{\gamma}$ and update

$$\begin{aligned} &\widehat{\Omega}_{12} = -\Omega_{11}(\mathbf{s}_{12}+\tilde{\gamma})/\mathbf{w}_{22}, \\ &\min_{\gamma\in\mathbb{R}^{p-1}} \frac{1}{2}(\mathbf{s}_{12}+\gamma)'\Omega_{11}(\mathbf{s}_{12}+\gamma);\ \text{subject to } \|\gamma\|_\infty \le \rho, \end{aligned} \tag{2.18}$$

where Ω_{11} is the left top part of the partition of Ω:

$$\Omega = \begin{pmatrix} \Omega_{11} & \Omega_{12} \\ \Omega_{21} & \Omega_{22} \end{pmatrix} \tag{2.19}$$

Step 2-3 Solve for ω_{22} using

$$\widehat{\Omega}_{22} = \frac{1-(\mathbf{s}_{12}+\tilde{\gamma})'\widehat{\Omega}_{12}}{\mathbf{w}_{22}}. \tag{2.20}$$

Step 2-4 Update the working covariance $\mathbf{w}_{12} = \mathbf{s}_{12}+\tilde{\gamma}$.

We denote the precision matrix estimated by DP-glasso as $\widehat{\Omega}_{DP}$.

By using DP-glasso, the primal optimization problem (2.5) is tackled and the precision matrix Ω becomes the target now. Unlike glasso, DP-glasso returns a sparse and positive definite estimated precision matrix even if the row/column updates are mistakenly stopped early. The precision matrix Ω is assured to be positive definite after every row/column iteration. For warm starts of DP-glasso, as shown in [Mazumder and Hastie, 2012b, Lemma 4], by starting the DP-glasso with a positive definite matrix Φ, every row/column iteration of the working precision matrix is positive definite, or DP-glasso will certainly converge from any positive definite warm start, thanks to the unconstrained primal problem (2.5).

The working covariance matrix obtained by DP-glasso is not necessarily the exact inverse of the precision matrix. For large regularization values, similar computational time is consumed by the glasso and DP-glasso algorithms. For smaller regularization parameters, DP-glasso is more appealing. For DP-glasso, $O(p(p-k))$ operations per full cycle is needed, where k is the number of non-zeros in $\hat{\Omega}$. However for glasso, the consumption is $O(p^2)$ per cycle.

For DP-glasso, in every row/column updates, varying number of updates are demanded in order to converge. Thus it is difficult to analyze every factor and conclude accurate convergence rates for the overall algorithm. However, Mazumder and Hastie have observed that with warm starts, along a relatively dense series of ρs, the computation costs given above are pretty much accurate for DP-glasso, especially when small/moderate accuracy is of interest.

By the experiments of Mazumder and Hastie [2012b], the DP-glasso with warm-start outperforms all the other combinations, including DP-glasso with cold start, glasso with warm start and glasso with cold start, across all the different simulations. For example, the simulation study included eight $p > n$ scenarios, with two types of covariance structures respectively. More specifically, large regularization parameters lead to faster convergence, while smaller ρs will slow down the convergence. For large p small ρ cases, algorithms converge slowly. Additionally, warm starts are not always helpful to accelerate the convergence of glasso, which may further affirm that the warm starts for glasso should be chosen cautiously to speed up convergence.

2.5.3 Summary and comments

The newly developed algorithm DP-glasso is similar to our first approach glasso, except its optimization variable is the precision matrix and it deals with the primal problem rather than the dual problem. Additionally, starting with any positive definite matrices, DP-glasso will produce sparse and positive definite precision matrix estimates. However, for neither glasso nor DP-glasso, one member of the pair (Ω, Σ) is not the inverse of the other. Moreover, not only theoretically but also experimentally, it has been shown that DP-glasso is computationally more efficient than glasso.

2.6 High-dimensional Undirected Graph Estimation (Huge)

2.6.1 Background

The Huge approach employed many suggestions from Friedman et al. [2007], Friedman et al. [2008] and Friedman et al. [2010], which aims at high-dimensional undirected graph estimation and integrates many functions together, such as data generating, graph estimation, model selection, estimation visualization, etc. More specifically, it merges many up-to-date proposals and results, such as nonparanormal and correlation screening approaches for estimating graphs (Liu et al. [2009] and Fan and Lv [2008]), as well as the StARS approach for stability-based graphical model selection (Liu et al. [2010]), etc. Additionally, two screening rules are offered, lossless screening (Witten et al. [2011] and Mazumder and Hastie [2012a]) and lossy screening. The method Huge we use comes from an `R` package, called '`huge`'.

2.6.2 Features and properties

The following features are provided by `huge`:

1. The package `huge` is coded in `C`, which makes the code more portable and easier to modify, rather than in Fortran, which glasso was coded in.

2. Besides estimating Gaussian graphical models, `huge` can also fit high-dimensional semiparametric Gaussian copula models.

3. More functions are included in `huge` besides graph estimation, such as data generator, model selection, neighborhood screening and graph visualization, etc.

4. A minor divergence problem in `glasso R` package was fixed, regarding to warm start settings.

5. `huge` offers two screening rules to scale up large scale problems, making a tradeoff between computational and statistical efficiency.

The optimal estimated $\widehat{\Omega}$ by `huge`, along the given regularization parameter path, is denoted by $\widehat{\Omega}_{Huge}$.

Many functions are provided in the package `huge`. The first, `huge()`, is used to estimate high-dimensional undirected graphs and the second, `huge.select()`, selects the best estimated graph along the whole regularization path, based on three embedded selection criteria, namely ric, ebic and stars.

Three graph estimation methods are available in `huge()`: the Meinshausen-Bühlmann approximation (`mb`) (see Meinshausen and Bühlmann [2006]), the Graphical Lasso algorithm (`glasso`) (Friedman et al. [2008] and Banerjee et al. [2008]), and the correlation thresholding graph estimation (`ct`). In addition, the first two methods can be further speeded up by the lossy screening rule which prechooses each variable's nearby variables by correlation thresholding before graph estimation (via `scr` argument in `huge()`) , and the third method is computational efficient and has been widely used in biomedical research Langfelder and Horvath [2008].

The function `huge.select()` selects the best estimate of the high-dimensional undirected graph, via three regularization parameter selection criteria: the stability approach for regularization selection (StARS) (Liu et al. [2010]); a modified

rotation information criterion (RIC) (Lysen [2009]); and the extended Bayesian information criterion (ebic) (Foygel and Drton [2010]). The detailed properties of these three selection criteria will be discussed in Chapter 3.

`huge.npn()` applies the nonparanormal method in Liu et al. [2009] for estimating a semiparametric Gaussian copula model by truncated normal or normal score. It implements Gaussianization to $\mathbf{X}_{n\times p}$ to help relax the normality assumption.

2.6.3 `huge` computational requirements

Based on the simulation results in [Zhao et al., 2012, Table 1 and Table 2] (with sample sizes and dimensions being $(n,p) \in \{(100,1000),(150,2000),(200,3000),(300,4000)\}$), the authors found that for Meinshausen-Bühlmann approximation (`mb`), `huge` is faster than `glasso`, also the lossy screening rule in `huge` can accelerate the computation up to 400%. Specifically, after applying lossy screening rule, each Lasso problem with dimension p is reduced to its sample size n, thus leading to greater efficiencies for $p \gg n$ cases. With Graphical Lasso (`glasso`) being the method argument, the computation is reduced a lot by the lossless screening rule (Witten et al. [2011] and Mazumder and Hastie [2012a]) provided by `huge`, especially when the estimator is very sparse. In addition, the lossy screening rule can also accelerate the algorithm.

2.6.4 Expectations for `huge`

Therefore we have the following expectations for `huge` based on the above properties:

1. less time consumed than the `glasso` package even without applying lossy screening rule (at least for $p \gg n$ scenarios)

2. it may suffer from overselected or underselected results due to the defects of ric, ebic and stars.

Chapter 3

Selection Criteria

All the methods used in this work are penalized log-likelihood methods, which require regularization parameters to control the sparsity of the precision matrix. However, it is unknown which value of the regularization parameter is optimal to give the best estimate. Hence, we use several selection criteria to choose the optimal regularization parameter among all the possible values for each method under each graphical structure setting. The selection criteria *AIC*, *BIC* and 5-*fold Cross Validation* are applied to the methods glasso, BG, AL, SCAD, DP-glasso, DP-BG, DP-AL and DP-SCAD. For the method Huge, we apply criteria *ric* (rotation information criterion, Lysen [2009]), *ebic* (extended Bayesian information criterion, Foygel and Drton [2010]) and *stars* (stability approach for regularization selection, Liu et al. [2010]) which are embedded in the package `huge`.

3.1 Selection criteria for all methods other than Huge

3.1.1 Akaike information criterion (AIC)

The AIC formula is

$$\mathrm{AIC}_\rho = 2k - 2\ln(L), \tag{3.1}$$

where $2k$ is be the number of non-zeros in $\widehat{\Omega}$ and $\ln(L)$ is the log-likelihood, namely (2.5) without the penalized term $\rho\|\widehat{\Omega}\|_1$. The regularization parameter corresponding to the minimum AIC value reports the best estimation selected by AIC.

AIC has the asymptotic optimality property (under the average squared error loss) for regression, that is, if the true model is not contained in the estimating models, and if the number of same-dimensional models does not grow fast in dimension, then the average squared error of the AIC-selected model is asymptotically equivalent to the smallest error among all candidate models can possibly provide (e.g. Shibata [1983], Li [1987], Polyak and Tsybakov [1990], Shao [1997] and Yang [2005]). However, AIC cannot choose models consistently.

Hereafter, let ρ_a denote the optimal regularization parameter selected by AIC.

3.1.2 Bayesian information criterion (BIC)

The formula for BIC is

$$\mathrm{BIC}_\rho = k \cdot \ln(n) - 2\ln(L), \tag{3.2}$$

where n is the sample size, $2k$ is be the number of non-zeros in $\widehat{\Omega}$ and $\ln(L)$ is the log-likelihood (Schwarz [1978]). The minimum BIC value reports the best ρ who gives the optimal estimate selected by BIC.

The assumptions of BIC are that observations should be independent and identically distributed (Schwarz [1978]). BIC consistently selects regression models, that is, if the true model is among the estimating regression models, the probability of selecting the true model by BIC approaches 1 as $n \to \infty$ (Nishii [1984]). For linear regression models, due to the heavier penalty, the model chosen by BIC is either the same or simpler than that chosen by AIC (Shao [1997]).

Hereafter, let ρ_b be the optimal regularization parameter selected by BIC.

3.1.3 K-fold Cross Validation (CV)

The K-fold Cross Validation score

$$\mathrm{CV}(\rho) = \sum_{k=1}^{K} \left(n_k \log |\widehat{\Omega}_{-k}(\rho)| - \sum_{i \in T_k} \left(x^{(i)} \right)^T \widehat{\Omega}_{-k}(\rho) x^{(i)} \right) \tag{3.3}$$

is addressed in [Fan et al., 2009, pp527], where n_k is the size of kth fold T_k and $\widehat{\Omega}_{-k}(\rho)$ is the estimate based on the sample $\left(\bigcup_{k=1}^{K} T_k\right) \setminus T_k$ (the training data). The minimum CV value reports the ρ providing the best estimate selected by cross validation. In our work, we use 5-fold Cross Validation to choose the optimal regularization parameter (different from the 6-fold cross-validation scheme used in Fan et al. [2009]).

Asymptotically, minimizing AIC is equivalent to minimizing CV, which is true for any model, not just linear models (Stone [1977]). Generally, Cross Validation does not select models consistently (Shao [1993]). Cross Validation performs poorly for high-dimensional data, sometimes dramatically ([Meinshausen and Bühlmann, 2010, P21]).

Hereafter, let ρ_c be the optimal regularization parameter selected by Cross Validation.

3.2 Selection criteria for Huge

The function `huge.select()` (Zhao et al. [2012] and Zhao et al. [2014]) provides three selection criteria to choose the best estimation by the method Huge, including ric, ebic and stars.

3.2.1 ric

ric is a newly developed and very efficient selection criterion. It directly estimates the best ρ based on some random rotations rather than finding the best ρ over the whole regularization path by some time consuming techniques such as cross validation or subsampling. More specifically, the variables for each sample are randomly rotated several times so that the minimum ρ which generates all zeros estimated by using the rotated data will be selected.

Thus far, there is no theoretical proof for the consistent selection by ric. Also, ric suffers from overselection and especially underselection frequently. Hence, there is a suggestion in [Zhao et al., 2014, pp17] that if desirable false negative levels (few missing selections) are expected, then the number of rotations for ric should be raised, or stars should be applied to make the selection. Moreover, ric is available for all three estimation methods provided by `huge`.

3.2.2 ebic

ebic is also denoted as BIC_γ, where $0 \leq \gamma \leq 1$ is called the ebic parameter. The original BIC is equivalent to BIC_0 ($\gamma = 0$). γ is the input `ebic.gamma` in the function `huge.select()` which can be tuned accordingly. $\gamma = 0.5$ is set as default in `huge.select()` and all our simulation results for ebic criteria stick to this value.

It has been shown in Chen and Chen [2008] that BIC_1 ($\gamma = 1$) is consistent as long as the dimension p does not grow exponentially with the sample size. $\text{BIC}_{0.5}$ ($\gamma = 0.5$) is consistent when $\kappa < 1$ and $p = O(n^\kappa)$; the original BIC (BIC_0) is consistent when $\kappa < 0.5$. [Foygel and Drton, 2010, Main Theorem] showed, under certain conditions ([Foygel and Drton, 2010, (3)]), that if ebic is applied to all decomposable models containing the true model, then the probability of choosing the smallest true model tends to one as the sample size tends to infinity. Non-zero entries are easily discovered for small γs, while false discoveries are better controlled for larger γs. In `huge`, `glasso` is the only available estimation method for ebic.

3.2.3 stars

stars selects the best estimated graph by similar techniques to subsampling so it is not very efficient. Under certain regularity condition, stars is shown to be partially consistent. This criterion suffers from the problem of overselection in estimating Gaussian graphical models while its performance also depends on the regularization parameters used. Moreover, stars can be used for all three estimation methods in `huge`, which are Meinshausen-Bühlman approximation (`mb`), glasso (`glasso`) and correlation thresholding estimation (`ct`).

Chapter 4

Simulations

There has been many theoretical conclusions about the estimating methods and the selection criteria described in Chapter 2 and Chapter 3. However, not all of them has experimental results and, more importantly, they have never been compared together under different model settings. Our work organizes all these methods and criteria together and compare their performance under various model dimensions, sample sizes and underlying distributions, etc. Even datasets with autocorrelation structure are included in our analyses. The next section explains the procedure of our simulation. Section 4.2 introduces the evaluation standards for comparing the estimating methods and selection criteria. Moreover, many important details about our simulation are elaborated in Section 4.3.

4.1 The procedure of simulation

We list the basic procedure of simution below in order to help a better understanding of our work:

Step 1 Generate the original dataset $\mathbf{X}_{n\times p}$ (or a real dataset $\mathbf{X}_{n\times p}$).

Step 2 Fix 100 equally spaced regularization parameters $\rho \in (0.01, 1]$ and $\rho_i = i \times 0.01$, $i = 1, \ldots, 100$.

Step 3 Apply each estimating method to $\mathbf{X}_{n\times p}$ and obtain 100 estimated matrices

$$\widehat{\Omega}_{M_q}(\rho_1), \widehat{\Omega}_{M_q}(\rho_2), ..., \widehat{\Omega}_{M_q}(\rho_{100}) \tag{4.1}$$

corresponding to 100 regularization parameters $\rho_1, ..., \rho_{100}$. In (4.1), M_q denotes the qth estimating method. For example, M_1 denotes the sample correlation matrix method, M_3 represents the glasso algorithm, M_4 denotes the bootstrap glasso algorithm and so forth. Then, $\widehat{\Omega}_{M_q}(\rho_i)$ denotes the precision matrix estimated by method M_q with regularization parameter ρ_i, $i = 1, ..., 100$, $q = 1, \ldots, 11$. For example, $\widehat{\Omega}_{M_3}(\rho_2)$ represents the best estimated precision matrix estimated by glasso when the regularization parameter is 0.02.

Step 4 For each method M_q, $q = 1, \ldots, 11$, let C_1, C_2, C_3, C_4, C_5 and C_6 denote the selection criteria AIC, BIC, CV, ric, ebic and stars, respectively.

Step 5 Apply different selection criteria C_c, $c = 1, \ldots 6$ to the estimated matrices $\widehat{\Omega}_{M_q}(\rho_1)$, $\widehat{\Omega}_{M_q}(\rho_2)$, ..., $\widehat{\Omega}_{M_q}(\rho_{100})$ in Step 2, choosing the estimate which minimizes the selection criteria formula. Let $\widehat{\Omega}^{C_c}_{M_q}(\rho_i)$ denote the precision matrix estimated by method M_q and selected by C_c criterion when the regularization parameter is ρ_i. Then, $\rho(M_q \star C_c)$ denotes the best regularization parameter among $\rho_1, ..., \rho_{100}$, i.e. $\widehat{\Omega}^{C_c}_{M_q}(\rho(M_q \star C_c))$ is the best estimate produced by estimating method M_q and selected by criterion C_c.

Step 6 For each method M_q, $q = 1, \ldots, 11$, use $\rho(M_q \star C_c)$ to re-estimate the precision matrix for the original dataset. Then, the resulting estimated precision matrix $\widehat{\Omega}(M_q \star C_c)$ is considered to be the best estimate for estimating method M_q and selection criterion C_c.

Step 7 For each method M_q and selection criterion C_c, repeat the above selection and re-estimating procedure L times. Then, we get L best estimated precision matrices for each combination of estimating method and selection criterion.

Step 8 Use these L estimated precision matrices to evaluate the performance of estimating methods and selection criteria.

4.2 Evaluation Standards

All the estimated precision matrices are compared under three evaluation standards. The first two standards called *True Positives* (TPs) and *True Negatives* (TNs) are numeric, measuring the estimation accuracy. The third standard, called the *Average Sparsity Pattern Plot* (ASP), is a plot that provides a visually depiction of the sparsity levels of the estimated matrices. Our definitions of TPs and TNs are different from the definitions used in Fan et al. [2009] and may lead to different conclusions. In the mean time, our definition of ASP is adapted from Fan et al. [2009].

4.2.1 True Positives

Our definition of the True Positives (TPs) is

$$\mathrm{TP} = \frac{\left(\sum_{l=1}^{L}\left(\sum_{1\le i,j\le p, i\neq j}\Omega_l^{tp}(i,j)\right)\right)/L}{N_2 - p}, \tag{4.2}$$

where L is the repetition time and N_2 is the number of non-zeros in the true precision matrix. The True Positives Matrix Ω_l^{tp} for the lth repetition is defined by

$$\Omega_l^{tp}(i,j) \doteq \begin{cases} 1, & \text{if } \Omega(i,j)\neq 0, \widehat{\Omega}_l(i,j)\neq 0, \\ 0, & \text{otherwise,} \end{cases}, \tag{4.3}$$

where $\widehat{\Omega}_l$ is the estimated precision matrix in the lth repetition.

The definition of the True Positives Matrix indicates that for each repetition, if both the true and estimated entry in (i,j)th position are non-zeros, the (i,j)th entry of the True Positives Matrix will be defined as 1, otherwise it is 0. Thus each True Positives Matrix Ω_l^{tp} records whether each non-zero true entry is successfully

detected in each repetition, for each method and selection criterion. Note that

$$\left(\sum_{l=1}^{L}\left(\sum_{i,j=1}^{p}\Omega_l^{tp}(i,j)\right)\right)/L \tag{4.4}$$

is the percentage of estimating each non-zero entry correctly over all repetitions. For example, if the (i,j)th entry of (2.7) is 0.85, then 85% of the time among all the repetitions the (i,j)th non-zero entry is estimated as non-zero. Consequently, TP is a number between 0 and 1 and reflects the average proportion of how many times each estimated precision matrix estimates non-zero entries correctly.

The larger the TP, the better the method. TP $= 1$ means that all the non-zero entries in Ω were estimated as non-zeros in every repetition. TP $= 0$ indicates that none of the real non-zero entries were estimated correctly (all of them were estimated as zeros in every repetition). TP $\neq 1$ indicates some non-zero entries are wrongly estimated as zeros, or from a graphical model perspective, that some edges are missing in the estimated graphs. Generally, the larger the TP is, the estimates are closer to the truth, and the estimated graphs (matrices) are denser.

4.2.2 True Negatives

The second standard True Negatives (TNs) is defined similarly by

$$\mathrm{TN} = \frac{\left(\sum_{l=1}^{L}\left(\sum_{i,j=1}^{p}\Omega_l^{tn}(i,j)\right)\right)/L}{N_1}, \tag{4.5}$$

where N_1 is the number of zeros in Ω. The True Negatives Matrix Ω_l^{tn} for the lth repetition is defined by

$$\Omega_l^{tn}(i,j) \stackrel{\circ}{=} \begin{cases} 1, & \text{if } \Omega(i,j) = \widehat{\Omega}_l(i,j) = 0, \\ 0, & \text{otherwise,} \end{cases}. \tag{4.6}$$

Similar to the True Positives Matrix, the True Negatives Matrix marks down whether each zero entry in Ω is successfully estimated to be zero. TN tells us how often zero entries are estimated as zeros. TN is also a numeric value between 0

and 1, with higher values indicating a better method. For example, TN $= 0.87$ indicates that on average, 87% of the zero entries were estimated as zeros among all repetitions. TN $\neq 1$ means that some zero entries are estimated as non-zeros, or that there are some mistakenly added edges in the estimated graphs. Thus typically, larger TN means sparser estimated graphs (matrices).

4.2.3 Average Sparsity Pattern Plot

The Average Sparsity Pattern (ASP) Plot is obtained by plotting the Overall Average Sparsity Matrix $\text{ASP}_{p\times p}$. For each $i, j = 1, ..., p$, $\text{ASP}(i, j)$ is defined by

$$\text{ASP}(i,j) = \frac{\sum_{l=1}^{L} \text{ASP}_l(i,j)}{L}, \tag{4.7}$$

and the Average Sparsity Matrix $\text{ASP}_l(i, j)$ of the lth repetition is defined by

$$\text{ASP}_l(i,j) \triangleq \begin{cases} 1, & \text{if } \widehat{\Omega}_l(i,j) \neq 0 \\ 0, & \text{otherwise,} \end{cases}, \; l = 1, \ldots, L. \tag{4.8}$$

Hence, the Overall Average Sparsity Matrix shows the percentage of times each element is estimated as non-zero among all the repetitions. We plot this matrix in `Matlab`, leading to our Average Sparsity Pattern Plot, with each rectangular area corresponding to an element of the ASP matrix. The larger the ASP value is, the darker the corresponding rectangular area becomes. So, the denser the estimated precision matrices are, the darker the Average Sparsity Pattern Plot is. This is an intuitive and clear way to show the overall sparsity levels of the precision estimates.

4.3 Simulation details

4.3.1 Settings for the estimating methods

4.3.1.1 Parameters of BG

When comparing BG to other estimating methods, we fix the number of resamples to be 50 and the threshold value $\pi_{thr} = 0.9$.

4.3.1.2 Parameters and initial values of AL

We use $\gamma = 0.5$ in the AL penalty (2.12) in all our simulations, which is the same as the choice in Fan et al. [2009]. They recommended $\gamma = 0.5$ because according to their numerical experience, there are no obvious differences among estimates by different γ.

Since the initial values of the AL penealty need to be any consistent estimates, in our simulations, we choose the glasso estimated precision matrices $\widehat{\Omega}_G$ to be its initial values, which means $\widetilde{\Omega} = \widehat{\Omega}_G$ and $\widetilde{\omega}_{ij} = \widehat{\Omega}_G(i,j)$, $1 \leq i,j \leq p$ in the AL penalty (2.12). More specifically, Fan et al. [2009] pointed out that $\widetilde{\Omega}$ can be the inverse sample covariance matrix $\mathbf{S}^{-1}$ in the low dimensional cases ($p < n$). $\widehat{\Omega}_G$ can be the candidate in the high dimensional cases ($p \geq n$). However, $\mathbf{S}^{-1}$ might be inconsistent if p increases in pace with n. Thus we choose $\widehat{\Omega}_G$ to be our initial matrices for AL in all our cases. This requirement for a consistent initial value is one of the drawback of AL.

The `R` package `glasso` makes it is convenient to implemente the AL algorithm. In our simulations, we get the Adaptive Lasso penalty matrix for each ρ first, which contains the penalties for all the elements of the Ω, then apply this penalty matrix as the `rho` argument in the package `glasso` to conduct the estimations by the AL method. The following estimation procedures are the same as glasso, including estimating and selecting as well.

4.3.1.3 Parameter and initial values of SCAD

In order to minimize the Bayes risk, $a = 3.7$ was recommended in Fan and Li [2001], which will also be used in all our simulations. The glasso estimated precision matrices $\widehat{\Omega}_G$ is employed to be the initial value of iterations for SCAD in our simulations.

The estimation by SCAD can also take the advantage of the package `glasso`. In our simulations, we obtain the SCAD penalty matrix for each ρ first, also set it as the `rho` argument in the package `glasso`. Then we choose the best ρ that minimize the selection criteria and use this optimal ρ to iteratively obtain new estimated precision matrices by using the glasso estimated precision matrices $\widehat{\Omega}_G$ as the initial value to get the new penalty matrix and the re-estimated precision matrices iteratively. We stop the iterations when the differences of the sum of the absolute values between the next two estimated precision matrices are less than a threshold, which we set it to be $1e-04$.

4.3.1.4 Settings for Huge

In the package `huge`, the function `huge()` only estimates the graphs along the whole regularization path without selecting the optimal estimate, while the function `huge.select()` helps to pick out the best estimate among all the results provided by the regularization parameters.

In the function `huge()`, we choose `glasso` in the `method` argument and set `scr` = FALSE, indicating that we use the glasso algorithm to estimate graphs and the lossy screening rule will not be applied to preselect the neighborhood before graph estimation. After implementing the function `huge()`, an object with the S3 class will be returned, named `Results_huge` for example, containing values including `icov`, a list of $p \times p$ estimated precision matrices corresponding to the regularization path, and `loglik`, a length(ρ) dimensional vector containing log-likelihood values along the regularization path and so on.

To implement `huge.select()`, its first argument is required to be an S3 class object. Thus we set it to be the returned value `Results_huge` from `huge()`. Accord-

ingly, this means we use the selection function `huge.select()` to pick the best estimate among all the estimates obtained by the estimating function `huge()` along the regularization path. All three criteria (ric, ebic, stars) provided by `huge.select()` will be applied in our simulation. Afterwards, `$opt.icov` shows the optimal estimated precision matrices by the Huge algorithm, corresponding to each selection criterion (ric, ebic, stars).

Additionally, we also study the effect of the nonparanormal transformation function `huge.npn()` provided in the package `huge`, which relaxes the assumption of the normal distributions of the given datasets. Thus we apply this function to our t-distributed datasets first in both low and high dimensions, and then let all the combinations operate on the transformed datasets to see how this function works practically, which is expected to improve the estimates for the non-normally distributed datasets

4.3.2 Settings for the data samples

The repetition time N is fixed to be 100 in all our simulations. To be specific, we apply each combination of estimating method and selection criteria under each simulation scenario to 100 datasets, and then observe how the combinations perform on average based on the 100 repetitions.

In order to reveal the differences between the combinations better, we set $N =$ 100 fixed different seeds in `R` so that in each repetition, all the different combinations operate on the same 100 datasets under each simulatio scenario. In so doing, the randomness of the data samples is removed and, hence, the differences among the results only arise from the estimating methods and selection criteria themselves.

4.3.3 Settings for the regularization path

In all our simulations, 100 equally spaced regularization parameters ρs are used, namely $\rho \in (0.01, 1]$ and $\rho_i = i \times 0.01$, $i = 1, \ldots, 100$.

The regularization parameters we use in our simulations are all between 0.01 and 1, thus the smallest value we can reach is 0.01 and 1 is the largest. One possi-

ble drawback of setting $\rho \in (0.01, 1]$ is that there may exist some smaller ρs (less than 0.01, e.g. 0.002) that could provide superior estimates than the best estimate that $\rho \in (0.01, 1]$ can provide. Another drawback is that our testing regularization parameters are discrete. Hence, it is also likely that some other ρs which are not a multiple of 0.01 (e.g. 0.233) could produce better estimates.

Nevertheless, we believe our regularization parameter choices and working procedures are convincing and the estimating methods are comparable. Although there are two deficiencies mentioned above, all the methods are applied to the same path of ρs and they are working under the same level of parameter precision and carrying out the same operations. Also, our working regularization parameters lie in a relatively dense and pretty wide range, which makes the results more meaningful.

4.3.4 Low dimensional simulations

For the low dimensional cases ($p = 5$), we fix only one true precision matrix Ω, apply all the estimating methods and selection criteria to various datasets, and see how they perform to estimate this fixed Ω under different types of data. The true precision matrix Ω we are going to estimate is

$$\begin{pmatrix} 1 & 0 & 0 & 0.6 & 0.5 \\ 0 & 1 & 0.4 & 0 & 0 \\ 0 & 0.4 & 1 & 0 & 0.6 \\ 0.6 & 0 & 0 & 1 & 0 \\ 0.5 & 0 & 0.6 & 0 & 1 \end{pmatrix}. \tag{4.9}$$

The various datasets we use in the low dimensional cases include the *multivariate normally distributed data* (the ***MVN data***), the *multivariate t-distributed data* with degrees of freedom (df for short) being 3 and 4 (the ***MV*t_3 *data*** and ***MV*t_4 *data***), and the *multivariate normally distributed data* of which every variable has the AR1 auto-correlation structure (***MVN AR1 data***). Additionally, we apply the non-paranormal transformation function `huge.npn()` in the package `huge` to the ***MV*t_4 *data*** to see how it performs to ease the violation of the normality assumption of the

datasets. All the datasets are studied in four sample sizes $(100, 200, 500, 1000)$ to see how the variations of the sample size affect the estimating results.

4.3.5 High dimensional simulations

For the high dimensional cases ($p = 30$), we study three different true precision matrices instead of a fixed Ω and evaluate how each combination performs. The three types of true precision matrices are the *tridiagonal matrices*, the *exponential decay matrices* and the *general matrices* as is suggested in Fan et al. [2009]. The detailed schemes of generating these matrices are introduced in the next subsection. Similarly to the low dimensional cases, for each of the three true Ωs, we apply all the combinations to the various types of data and then compare the performances of all the combinations among the various datasets as well as the different Ωs.

The types of the datasets we use for the high dimensional cases are the same as the low dimensional cases, which are the ***MVN data***, the ***MV*t_3 *data*** and ***MV*t_4 *data***, ***MVN AR1 data*** and the transformed ***MV*t_4 *data*** (apply to the function `huge.npn()`). The only differences are that the datasets herein are of dimension 30 and generated from three different precision matrices rather than a fixed matrix. In addition, we fix the sample size n to be 100 for all the high dimensional datasets.

4.3.6 Matrix Generating Scheme and Parameter Choices

For the three Ω in the high dimensional cases, we employ the same matrix generating scheme as [Fan et al., 2009, 4. Monte Carlo simulation]:

4.3.6.1 Tridiagonal case

The (i, j)th element of Ω is defined to be

$$\omega_{ij} = \exp\left(-a\left|s_i - s_j\right|\right), \tag{4.10}$$

where a is a positive constant and s_i, s_j are random values such that $s_1 < s_2 < \cdots < s_p$ and

$$s_i - s_{i-1} \overset{\text{i.i.d.}}{\sim} \text{Unif}(0.5, 1),\ i = 2, \ldots, p. \tag{4.11}$$

Obviously, larger a will produce smaller off-diagonal elements in Ω.

In the high dimensional case, after trying a number of different values for a, we found that the value 0.85 is the smallest constant that is able to generate a positive definite matrix very quickly. In other words, $a = 0.85$ efficiently produces a positive definite matrix with the largest entries, or this tridiagonal matrix is on the edge of satisfying positive definiteness. Thus for the tridiagonal case with dimension being 30, one matrix is generated by setting $a = 0.85$, whose non-zero entries are between 0.4 and 0.5, with several entries near 0.6. Besides this tridiagonal matrix with large off-diagonal entries, we also employ another tridiagonal matrix with $a = 1.7$. By changing the constant to 1.7, non-zero entries in the precision matrix become smaller, which are closer to 0.3. The summaries of these two tridiagonal matrices are given in Table 1and Table 2, respectively.

4.3.6.2 Exponential decay case

No element of the precision matrix is exactly zero, but Ω contains a number of entries close to 0. The (i, j)th element of the true exponential decay precision matrix is defined to be

$$\omega_{ij} = \exp(-2 \mid i - j \mid), \tag{4.12}$$

which can be extremely small when $\mid i - j \mid$ is large.

Since none of the entries of the exponential decay true matrix Ω are exactly zero, if we don't apply a threshold to the matrix, TN will be reported to be NA and TP will be dramatically small as well. This does not effectively reflect the estimation procedure. Thus we set a threshold of $1e-03$ when calculating TP and TN in this case, but the exact Ω without thresholding is still used to generate the original dataset. Also in the Average Sparsity Pattern Plots, a threshold is applied to the true Ω and $\widehat{\Omega}$ via different estimating methods. The summary of entries of the 30-dimensional exponential decay matrix we will use in our simulation is given

in Table 3.

General matrix case

We generate an upper triangular matrix first. Each element in the upper triangular matrix is generated uniformly over $[-1,-0.5]\cup[0.5,1]$. By symmetrizing this upper triangular matrix, we get a matrix with main diagonals being 0s. We set the (i,i)th entry in this symmetric matrix to be a multiple of the sum of the absolute values of the ith row elements. Here we choose a multiple of 2, which is the same as in Fan et al. [2009], in order to ensure the resulting Ω is positive definite.

For the general case, we also set the threshold to be $1e-03$ when calculating TPs and TNs, as well as for Average Sparsity Pattern Plots. After thresholding, most entries in Ω are between 0.1 and $1e-02$, and the second majority of entries are between $1e-02$ and $1e-03$, while just a small amount of entries are larger than 0.1. On average, the entries are much smaller than the tridiagonal case and the summary of its entries is displayed in Table 4.

Summary of entries

The following tables give a summary of the entries of the three precision matrices generated as above.

Table 1: *Summary of the entries of the tridiagonal matrix with* $a = 0.85$

smallest	**largest**	**0.1 to 1**	**0.01 to 0.1**	**less than 0.01**
0.431 (except 0)	0.65	0.067	0	0.933

Table 2: *Summary of the entries of the tridiagonal matrix with* $a = 1.7$

smallest	**largest**	**0.1 to 1**	**0.01 to 0.1**	**less than 0.01**
0.192 (except 0)	0.425	0.067	0	0.933

Table 3: *Summary of the entries of the exponential decay matrix*

smallest	**largest**	**0.1 to 1**	**0.01 to 0.1**	**less than 0.01**
$6.47 \cdot e-26$	0.135	0.067	0.064	0.869

Table 4: *Summary of the entries' absolute values of the general matrix*

smallest	**largest**	**0.1 to 1**	**0.01 to 0.1**	**less than 0.01**
$1.11 \cdot e-16$	0.181	0.037	0.726	0.259

Chapter 5

Results

5.1 Results for Normally Distributed Data

5.1.1 Low dimensional cases

Table 5: *The TPs for the **MVN data** ($p = 5$) at 4 different sample sizes*

	$n = 100$			$n = 200$			$n = 500$			$n = 1000$		
	TP			**TP**			**TP**			**TP**		
	AIC	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**
CM	1			1			1			1		
PCM	1			1			1			1		
glasso	1	1	1	1	1	1	1	1	1	1	1	1
AL	1	1	1	1	1	1	1	1	1	1	1	1
SCAD	0.999	1	0.993	0.995	0.996	0.999	1	1	1	1	1	1
DP												
glasso	1	1	1	1	1	1	1	1	1	1	1	1
AL	1	1	1	1	1	1	1	1	1	1	1	1
SCAD	0.999	0.998	0.998	0.996	0.996	0.999	1	1	1	1	1	1
BG												
$H = 50$	1	1	1	1	1	1	1	*1*	*1*	1	1	1
DP-boo	1	1	1	1	1	1	1	1	1	1	1	1
Huge												
ric	1			1			1			1		
stars	1			1			1			1		
ebic	1			1			1			1		

Table 5 displays the True Positives (TPs) for the multivariate normally distributed dataset (***MVN data*** for short hereafter) with dimension $p = 5$ and four different sample sizes. From Table 5 we can see that large value non-zero entries are easily estimated correctly, especially when the sample size n is large, with less than 1% false detections.

Among all the estimating methods, SCAD is the only method that occasionally provides zero estimates for non-zero entries for the low dimensional ***MVN data***. In terms of TPs for the low dimensional ***MVN data***, all the other methods perform fairly well and all selection criteria perform similarly in their abilities of estimating non-zeros. In addition, we do not observe obvious improvements for glasso, AL, SCAD and BG when applying the newly proposed DP-glasso algorithm. For such a low dimensional case ($p = 5$), $n = 100$ should be large enough to estimate large value non-zero entries. Moreover, increasing sample size does not appear to improve the estimates.

Table 6: *The TNs for the **MVN data** ($p = 5$) at 4 different sample sizes*

	$n = 100$			$n = 200$			$n = 500$			$n = 1000$		
	TN			**TN**			**TN**			**TN**		
	AIC	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**
CM	0			0			0			0		
PCM	0			0			0			0		
glasso	0.213	0.333	0.335	0.143	0.295	0.242	0.193	0.353	0.249	0.24	0.353	0.25
AL	0.499	0.647	0.678	0.52	0.673	0.649	0.687	0.815	0.73	0.843	0.854	0.843
SCAD	0.708	0.9	0.855	0.75	0.91	0.868	0.937	0.982	0.943	0.955	0.977	0.963
DP												
glasso	0.213	0.33	0.335	0.143	0.297	0.245	0.193	0.352	0.248	0.242	0.352	0.252
AL	0.499	0.645	0.678	0.52	0.672	0.648	0.687	0.815	0.73	0.842	0.853	0.842
SCAD	0.708	0.902	0.857	0.755	0.912	0.868	0.937	0.982	0.943	0.955	0.977	0.963
BG												
$H = 50$	0.672	0.858	0.558	0.595	0.836	0.496	0.7	0.877	0.695	0.828	0.893	0.83
DP-boo	0.673	0.858	0.56	0.593	0.837	0.5	0.702	0.878	0.697	0.702	0.878	0.697
Huge												
ric	0.252			0.17			0.172			0.175		
stars	0.243			0.29			0.353			0.374		
ebic	0.4			0.338			0.353			0.374		

Table 6 presents the True Negatives (TNs) for the low dimensional ***MVN data*** for all sample sizes. It is clear that the zero entries of the true precision matrix Ω are much more difficult to estimate than the non-zero entries, even for low dimensional ***MVN data*** with large sample sizes.

Generally, increasing the sample size can effectively enhance the chances of estimating zeros correctly, especially when the ratio of the sample sizes to the dimension is over 100. Among all the estimating methods, the SCAD method has the strongest power to detect zero entries, especially when the estimates are selected by

BIC. AL and BG are a little weaker than SCAD. More specifically, BG selects more zeros than AL when $n \leq 500$, while AL starts to outperform BG when the sample size is greater than 500. glasso is always a lot worse at capturing zero entries than AL, BG and SCAD. In addition, there appear to be no obvious improvements due to the application of the DP-glasso algorithm to glasso, AL, SCAD and BG. Huge becomes better at estimating zero entries when n increases from 200, yet it is still always worse than glasso. Typically, all estimating methods perform better when the sample size increases.

The BIC criterion selects estimates with the largest number of correct zero entries for the glasso, BG, AL and SCAD methods (as well as DP-glasso, DP-BG, DP-AL and DP-SCAD) in almost all situations, with the only exception being glasso and AL methods at $n = 100$, where CV selects slightly more zeros than BIC. With regards to AIC, it always selects less zero entries than BIC and CV for all sample sizes for glasso, AL and SCAD methods. However, BG behaves differently with AIC selecting more zeros than CV when $n < 1000$, while still being much inferior to BIC.

Generally, the ebic criterion provides the estimates with the most zeros among all three selection criteria for Huge, which are slightly better than glasso estimates. There is no noticeable improvements in the TNs of Huge$\star$ric results as sample size increases. Notice also that the TN decreases when sample size increases from 100 to 200 for Huge$\star$ric. stars is the only criterion in Huge that the TNs increase with the sample size. Another interesting fact is that stars and ebic have exactly the same TNs when $n \geq 500$. Also they both are greater than the TNs of ric.

To conclude, in terms of TNs, SCAD is the best method compared to glasso, AL, BG and Huge. AL is regarded as the second best approach if considering computational time. BG provides competitive estimates to AL, while glasso and Huge are less effective than other methods. As for selection criteria, BIC selects the best estimates for glasso, BG, AL and SCAD most of the time (with the most zero entries). ebic can be considered as the best criterion for Huge. In the view of combinations of estimating methods with selection criteria, SCAD with any of the AIC, BIC or CV criterion outperforms all other combinations, except BG$\star$BIC per-

forms better than SCAD⋆AIC when $n \leq 200$, which is an indication that sometimes the superiority of a selection criterion can remedy the inefficiency of an estimating method.

5.1.2 High dimensional cases

In order to better understand the average sparsity pattern (ASP) plots of the estimates for all methods and selection criteria combinations, we will introduce the sparsity patterns of the three true precision matrices first.

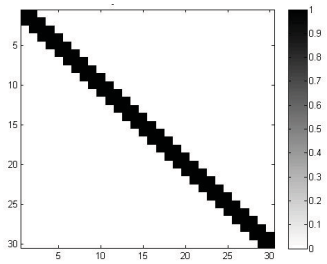

Figure 2: *The sparsity pattern plot of the tridiagonal matrix with* $a = 0.85$ *and* $a = 1.7$

Figure 2 shows the sparsity patterns of the tridiagonal matrices with the constant a being 0.85 and 1.7 in (2.19). Since the sparsity pattern only measures whether an entry is zero or non-zero, rather than the exact value, the sparsity pattern plots of the two tridiagonal matrices (with a being 0.85 and 1.7) are the same.

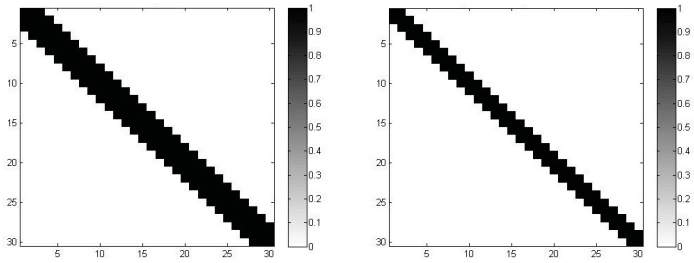

Figure 3: *The sparsity pattern plots of the exponential decay matrix*

Figure 3 corresponds to the exponential decay matrix. If a $1e-03$ threshold is applied to the original matrix, it indicates that some non-zero entries of the new metrix are between 0.001 and 0.01. Accordingly, if the threshold is $1e-02$, all the non-zero entries will be larger than 0.01.

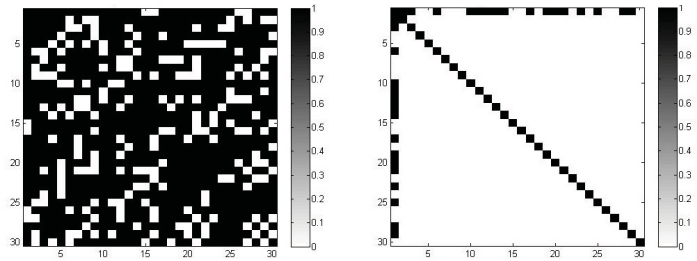

Figure 4: *The sparsity pattern plots of the general matrix*

Figure 4 illustrates the sparsity patterns of the general matrix, where the first plot has a $1e-02$ threshold and the latter one has a $1e-01$ threshold. Thus the first plot shows the spread of non-zeros of the general matrix which are greater than 0.01 and the second plot shows where entries > 0.1 are located.

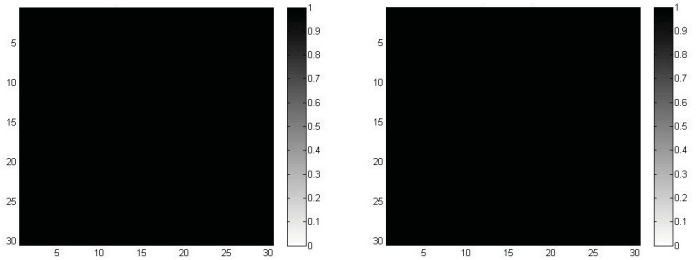

Figure 5: *The ASPs of the sample correlation matrix and the sample partial correlation matrix*

Figure 5 displays the ASPs of the estimates for the sample correlation matrices (CM) and the sample partial correlation matrices (PCM), which are identical for all dimensions, Ω types and sample sizes. Thus we will not show these two ASP plots for CM and PCM hereafter. It is obvious from Figure 5 that all the entries of the sample correlation matrices and the sample partial correlation matrices are always

non-zero, which is a strong reason that these two matrices are improper to be used as effective estimates of the true precision matrix Ω.

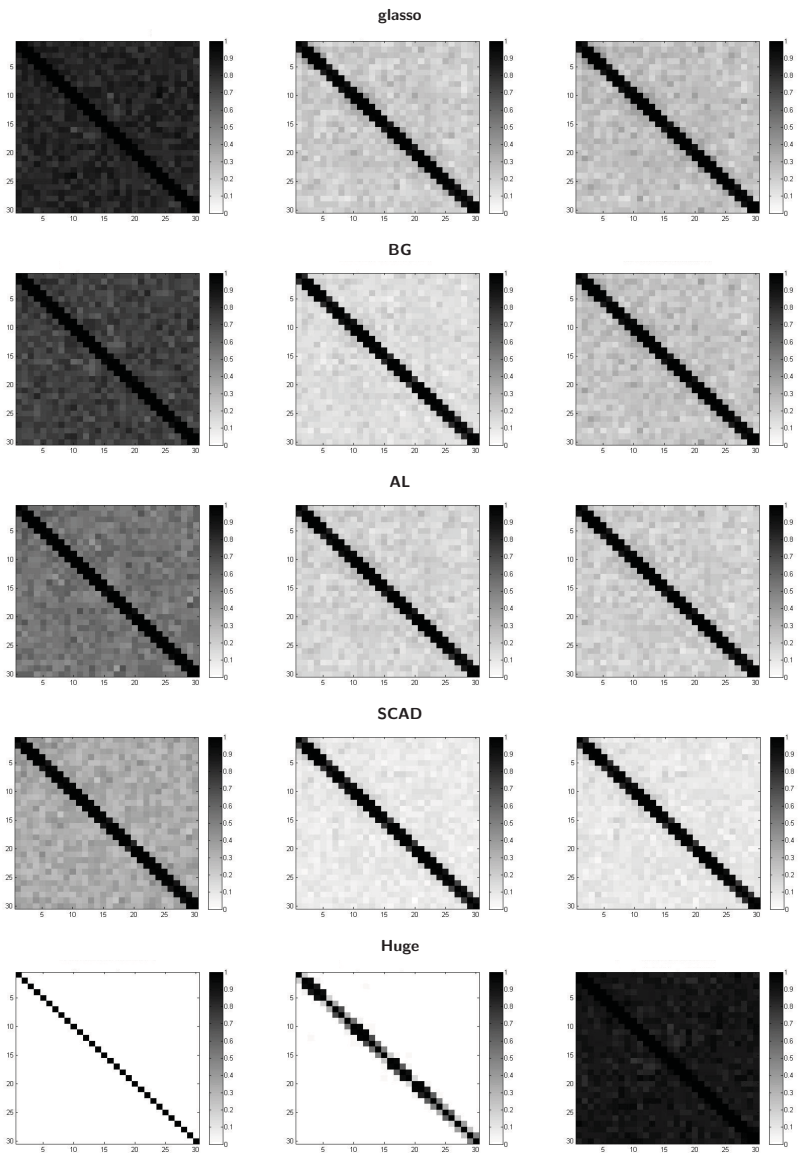

Figure 6: *The ASP plots for the **MVN data** ($p = 30$) generated from the tridiagonal matrix with $a = 1.7$*

[Left panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the AIC criterion. [Middle Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the BIC criterion. [Right Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the CV criterion.

Figure 6 contains all ASP plots of a ***MVN data*** generated from the $a = 1.7$ tridiagonal true precision matrix, corresponding to all the combinations of the estimating methods and the selection criteria. As can be seen from Figure 6, SCAD⋆BIC and SCAD⋆CV produce the sparsest estimates without losing the true graphical structure, where nearly 90% of zero entries are successfully captured. BG⋆BIC performs slightly worse than the best combinations, which can be regarded as the second best combination with about 85% correct detections of zeros. AL⋆BIC, glasso⋆BIC, glasso⋆CV, BG⋆CV and AL⋆CV result in denser estimates than the second best combination with 80% correctly estimated zeros and we see them as the third best choices to estimate Ω. SCAD⋆AIC is the only combination that doesn't lead to fairly dense estimates among all estimates selected by AIC. AL⋆AIC results in much denser estimates than AL⋆BIC and AL⋆CV do. Additionally, BG⋆AIC and glasso⋆AIC estimates are too dense to see the true graphical structure, especially glasso⋆AIC.

In terms of non-zero estimates, all combinations work well in that they hardly miss any of the non-zero entries. This may indicate that non-zero entries greater than 0.19 are large enough to be always sensitively detected by glasso, BG, AL and SCAD, where 0.19 is the smallest value of the tridiagonal matrix with $a = 1.7$.

In terms of the estimation accuracies for the DP-glasso algorithm, DP-glasso performs almost the same as glasso, as does DP-BG and BG , DP-AL and AL, DP-SCAD and SCAD.

Huge estimates are either excessively dense or excessively sparse. Huge⋆ebic shrinks the original entries so much that all of their estimates are zeros, thus it completely loses the true model structure. Since 0.425 is the largest non-zero entry of the tridiagonal matrix, Huge⋆ebic ASPs show that 0.425 may be not large enough to be successfully detected by Huge⋆ebic. Contrarily, Huge⋆stars produces

too many non-zero estimates and it is only able to estimate 11.6% of zero entries accurately, which makes the estimated matrices extremely dense. Huge$\star$ric has a stronger ability to detect non-zero entries than Huge$\star$ebic and 58.2% of non-zeros are correctly estimated by it. As can be seen from its ASP, the dark areas in the first row above and below the main diagonal have actually relatively large non-zero values (> 0.25). This reflects the fact that Huge$\star$ric has the strongest power of distinguishing zeros and non-zero entries correctly among all three criteria for Huge. However, it only has the power to discover relatively large non-zero entries (e.g. > 0.25) and other smaller non-zero entries (e.g. less than 0.25) are often missed by Huge$\star$ric, similar to Huge$\star$ebic. More specifically, we find that if an entry is larger than 0.31, it is very likely that this entry will never be missed by Huge$\star$ric.

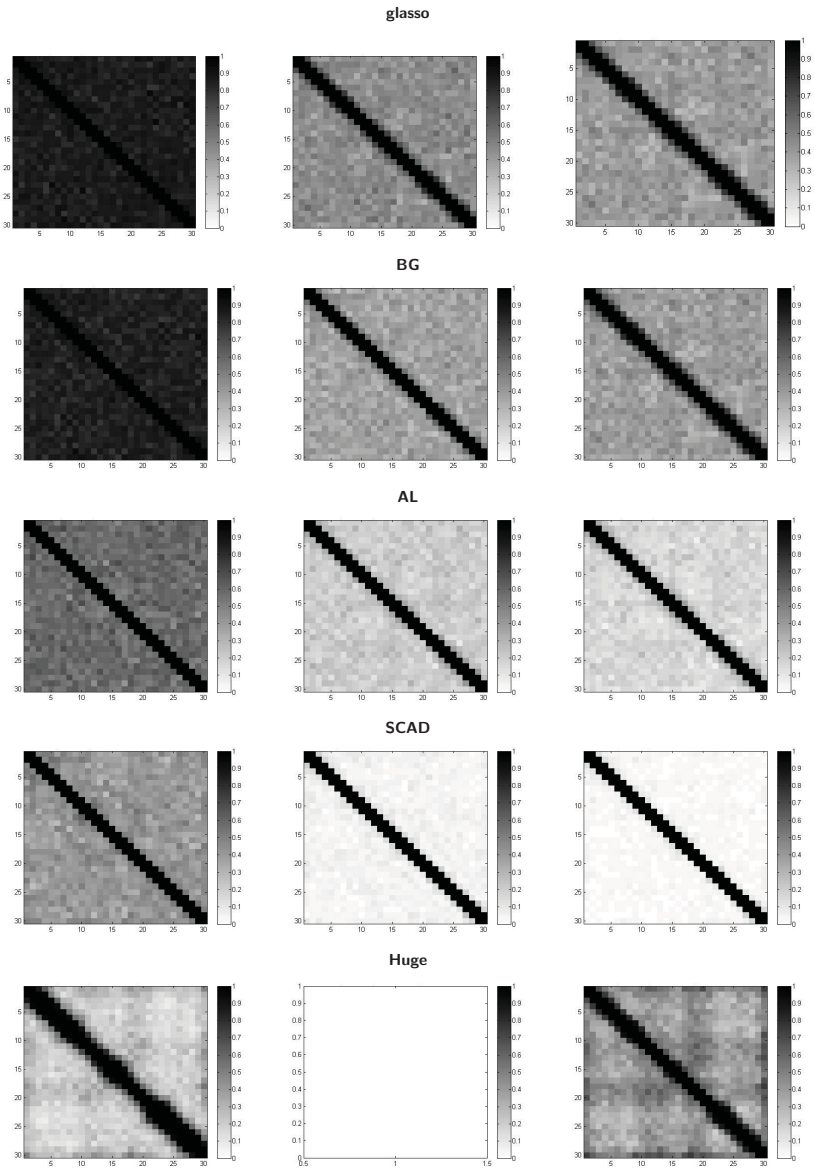

Figure 7: *The ASP plots for the **MVN data** ($p = 30$) generated from the tridiagonal matrix with $a = 0.85$.*

[Left panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the AIC criterion. [Middle Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the BIC criterion. [Right Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the CV criterion.

Figure 7 are the ASP plots of the ***MVN data*** generated from the tridiagonal true precision matrix with $a = 0.85$. Compared to Figure 6, we found that the Huge combinations produce the biggest difference of the results between the two tridiagonal matrices with different as.

For the ***MVN data*** generated from the tridiagonal matrix with $a = 0.85$, we cannot get estimates of Huge⋆ric, since there is no return value from this combination in R, which does not happen to the ***MVN data*** generated from the tridiagonal matrix with smaller entries ($a = 1.7$). In contrast, the other two Huge combinations, the Huge⋆ebic and Huge⋆stars, provide much better results than the tridiagonal matrix with $a = 1.7$, and are able to capture the true graphical structure now. More specifically, the estimation of zero entries of Huge⋆stars and the estimation of non-zero entries of Huge⋆ebic are dramatically boosted, which outperform glasso⋆AIC/BIC combinations with Huge⋆ebic even being competitive with AL combinations. In other words, if the true precision matrix is on the edge of being positive definite (containing many large entries and barely satisfying the requirement of positive definiteness), the Huge combinations will either perform well (such as Huge⋆stars and Huge⋆ebic), providing much better estimates than the true precision matrix with small entries, or the Huge combination is not able to provide estimates, such as the Huge⋆ric. This implies that the level of the positive definiteness (how close it is to be not positive definite) of the true precision matrix will indeed significantly affect the estimation of the Huge combinations. More specifically, if a Huge combination can produce estimates, they will be significantly better for a matrix with larger entries than the matrix with smaller entries. However, in the meanwhile, a combination is also possibly unable to provide estimates, as a matrix with larger entries is closer to be not positive definite and cause estimation problems, such as the Huge⋆ric combination.

For the glasso, AL, BG and SCAD combinations, all the non-zero entries are estimated better, while the majority of the zero entry estimates becomes worse than the tridiagonal matrix with $a = 1.7$, with only AL⋆CV, SCAD⋆BIC, SCAD⋆CV and BG⋆AIC providing better results. This tells us that the stronger true connections are detected better, but in the meanwhile, there are also more wrongly estimated zero

entries for most of the combinations, which implies that even though the entries are larger, the whole graphical structure is not necessarily to be captured more accurate, possibly due to the closeness of the true precision matrix being not positive definite.

The DP-glasso algorithm is again only able to slightly improve the estimates, which means the estimates of DP-glasso, DP-AL and DP-SCAD algorithms are only slightly better than the glasso, AL and SCAD results. An interesting phenomenon happened to the tridiagonal matrix with $a = 0.85$ is that the DP-BG algorithm will get stuck after several times of repetition, so that the remaining estimation procedure (other repetitions) can not proceed (but we randomly set the same seeds in `R` for all the simulation scenarios). Thus we can see that besides the worse or better estimates, the closeness to be not positive definite can also result in the inapplicability of some estimating methods, such as DP-BG.

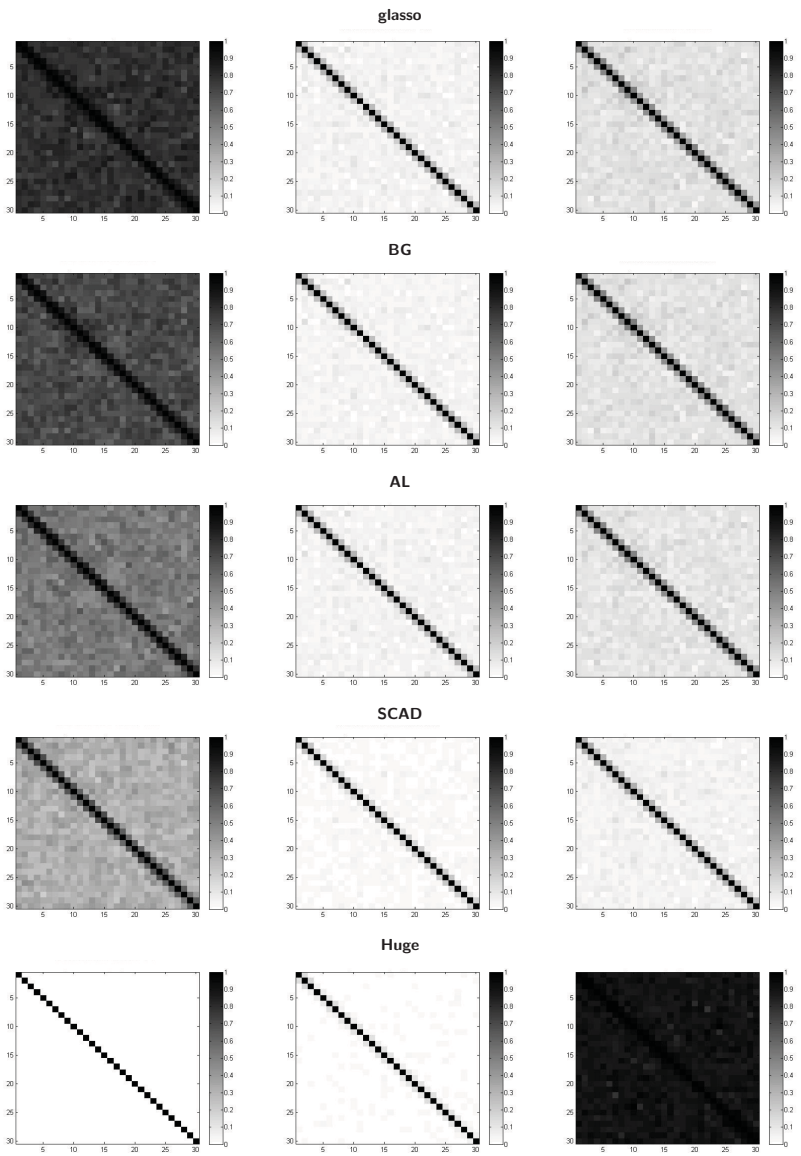

Figure 8: *The ASP plots for the **MVN data** ($p = 30$) generated from the exponential decay matrix*

[Left panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the AIC criterion. [Middle Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the BIC criterion. [Right Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the CV criterion.

Figure 8 are the ASPs of the exponential decay matrix, which are very similar to tridiagonal case. In the exponential decay matrix, 0.1353 is the only non-zero

entry that is larger than 0.1. As can be seen from Figure 8, SCAD⋆BIC, AL⋆BIC, BG⋆BIC, glasso⋆BIC and SCAD⋆CV work the best, providing very sparse estimates, with only entries in the first row next to the main diagonal being estimated to be non-zeros at times, where the real entry is 0.1353. AL⋆CV, BG⋆CV and glasso⋆CV perform similarly well, leading to slightly denser estimates than the best combinations, with about 20% of correctly detected non-zeros and nearly 90% detections of zero entries. These methods achieve a pretty good sparsity level and maintain the true graphical structure, since the majority of the missed non-zero entries are fairly small ones (e.g. < 0.1). SCAD⋆AIC is the third best combination without losing the true structure. AL⋆AIC estimates are even denser than the SCAD⋆AIC results, just about keeping the basic structure. BG⋆AIC and glasso⋆AIC produce pretty dense estimates that lose the actual graphical structure, especially glasso⋆AIC.

Thus we can deduce that entries of the order of 0.1 are hard to accurately estimate using glasso, BG, AL and SCAD, especially when selected by BIC or CV, which tend to produce sparse estimates. AIC is able to detect non-zero entries around 0.1, however, it provides many unnecessarily estimated non-zeros and makes the matrices very dense.

Another noticeable fact is that for estimates with many improperly estimated non-zeros, it seems that the false non-zeros are randomly spread throughout Ω. Hence, any entry smaller than a threshold (say, 0.08) has an equal chance to be estimated as zero or non-zero, regardless of its exact difference from the threshold.

Additionally, the DP-glasso algorithm hardly improves the glasso results, as well as the DP-BG, DP-AL and DP-SCAD methods.

Huge⋆ebic seriously underselects non-zero entries and leads to the identity matrix being estimated all the time. Huge⋆ric also suffers from severe underselection problems, which is slightly better than Huge⋆ebic with about 3.6% of correct detections, compared to the zero correct estimates for Huge⋆ebic. This may confirm the conclusion we get from the tridiagonal case that 0.1353 is too small for Huge⋆ebic and Huge⋆ric to sensitively detect. Huge⋆stars suffers from a severe overselection again, resulting in extremely dense estimates. All Huge estimates lose the true

graphical structure.

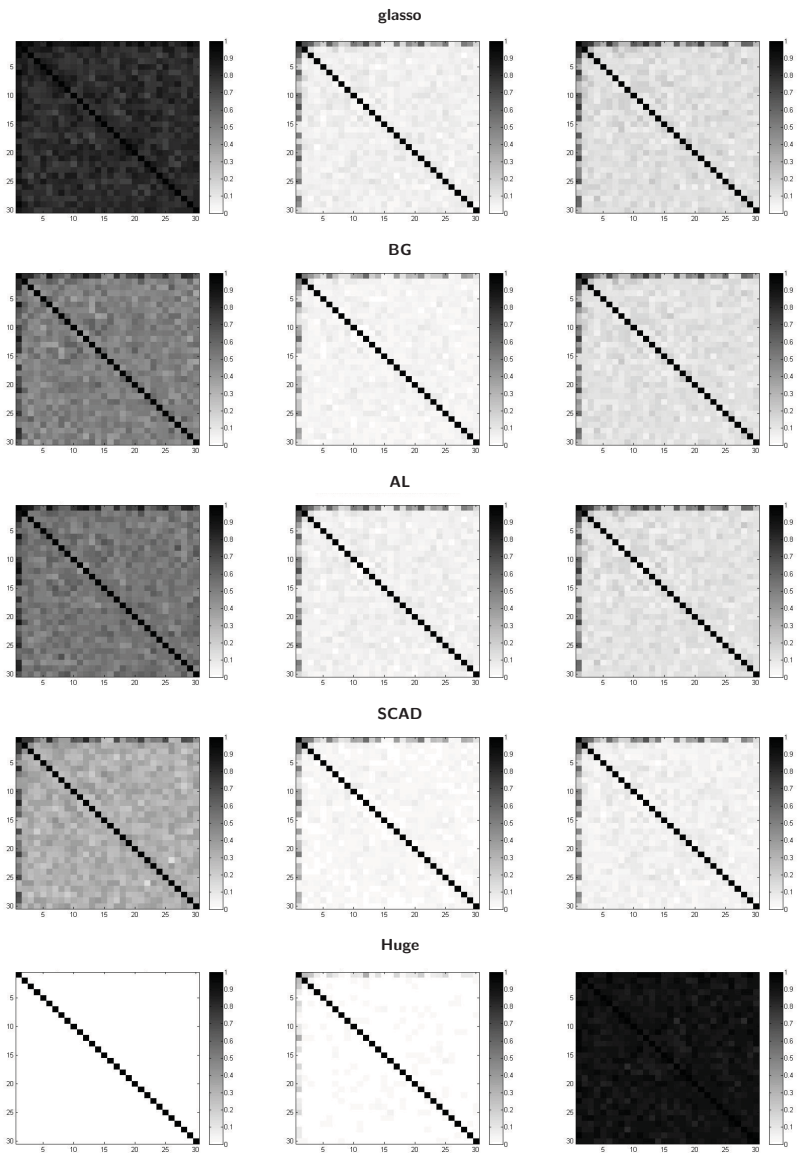

Figure 9: *The ASP plots for the **MVN data** ($p = 30$) generated from the general matrix*

[Left panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the AIC criterion. [Middle Panel] The ASP plots of the glasso, BG ,AL, SCAD and Huge estimates selected by the BIC criterion. [Right Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the CV criterion.

Figure 9 corresponds to all the ASPs for the general matrix. Most of the non-zero entries of this general matrix are less than 0.1 and its largest value is 0.1812,

and all entries are randomly spread.

Results from the general matrix show very similar conclusions to the exponential decay estimates. Again, SCAD⋆BIC, AL⋆BIC, BG⋆BIC, glasso⋆BIC and SCAD⋆CV provide the sparsest estimates, with around 10% detections of the non-zeros and 95% for the zero entries, providing non-zero estimates only for the relatively large entries, such as entries larger than 0.1. AL⋆CV, BG⋆CV and glasso⋆CV lead to slightly denser estimates than the sparsest results, with larger frequencies of estimating non-zero entries, thus those large values (> 0.1) are estimated correctly more often. BG⋆AIC and AL⋆AIC results are still very dense, yet they are able to maintain the basic graphical structure, giving about 60% of accurately estimated non-zeros and 50% detections of the zero entries. glasso⋆AIC estimates are so dense that the original structure is lost. Additionally, the DP-glasso algorithm slightly reduces the TPs and TNs over glasso, with decreases in TPs and TNs also occurring in DP-BG, DP-AL and DP-SCAD.

Huge⋆ebic and Huge⋆ric suffer from severe underselections again with only approximately 2.2% and 3.45% of non-zeros are correctly estimated, respectively. Nevertheless, almost all zero entries are detected. Huge⋆stars still greatly overselects non-zeros so that only 10.4% zero entries are correctly estimated. All of them lose the true model structure.

Table 7: *The TPs and TNs for the tridiagonal matrix with* $a = 1.7$ *for the* ***MVN data*** *(*$p = 30$*)*

Ω	**tri** $a = 1.7$					
	TP			**TN**		
	AIC	**BIC**	**CV**	**AIC**	**BIC**	**CV**
CM	1			0		
PCM	1			0		
glasso	0.995	0.958	0.967	0.172	0.783	0.741
AL	0.983	0.95	0.958	0.452	0.819	0.798
SCAD	0.961	0.915	0.925	0.661	0.908	0.897
DP						
glasso	0.995	0.958	0.967	0.172	0.783	0.741
AL	0.983	0.95	0.958	0.452	0.819	0.798
SCAD	0.958	0.915	0.925	0.684	0.908	0.897
BG						
$H = 50$	0.989	0.932	0.962	0.267	0.855	0.756
DP-boo	0.965	0.9	0.948	0.497	0.9	0.799
Huge						
ric	0.582			0.998		
stars	0.997			0.116		
ebic	0			1		

Table 8: *The TPs and TNs for the exponential decay matrix and the general matrix for the **MVN data** ($p = 30$)*

Ω	exp						gen					
	TP			TN			TP			TN		
	AIC	BIC	CV	AIC	BIC	CV	AIC	BIC	CV	AIC	BIC	CV
CM	1			0			1			0		
PCM	1			0			1			0		
glasso	0.838	0.128	0.235	0.209	0.951	0.877	0.838	0.128	0.235	0.209	0.951	0.877
AL	0.626	0.123	0.219	0.472	0.954	0.893	0.626	0.123	0.219	0.472	0.954	0.893
SCAD	0.428	0.074	0.134	0.692	0.98	0.951	0.428	0.074	0.134	0.692	0.98	0.951
DP												
glasso	0.838	0.1278	0.235	0.209	0.951	0.877	0.813	0.105	0.178	0.203	0.947	0.875
AL	0.626	0.123	0.219	0.472	0.955	0.893	0.563	0.101	0.162	0.459	0.953	0.894
SCAD	0.428	0.074	0.134	0.692	0.981	0.951	0.351	0.065	0.097	0.681	0.977	0.949
BG												
$H = 50$	0.731	0.084	0.182	0.295	0.958	0.881	0.577	0.102	0.224	0.509	0.966	0.888
DP-boo	0.577	0.102	0.224	0.509	0.966	0.888	0.521	0.08	0.164	0.498	0.963	0.886
Huge												
ric	0.036			0.994			0.035			0.993		
stars	0.924			0.098			0.906			0.104		
ebic	0			1			0.022			1		

Table 7 and Table 8 are the exact TPs and TNs based on the high dimensional ***MVN data*** with $n = 100$ estimated by each combination, showing the proportions of correctly estimated non-zero and zero entries, corresponding to different Ωs, respectively. The TP and TN table of the tridiagonal matrix with $a = 0.85$ can be found in the appendix. The tables show consistent conclusions to the ASP plots. Generally, the larger the TPs are, the darker the ASP plots become and the denser the estimates are. Similarly, the larger the TNs are, the lighter the ASP plots show and the sparser the estimates become.

5.2 Results for t-distributed Data

5.2.1 Low dimensional cases

5.2.1.1 df=3

Table 9: *The TPs for the* $\boldsymbol{MVt_3}$ ***data*** *(*$p = 5$*) at* 4 *different sample sizes*

	$n = 100$			$n = 200$			$n = 500$			$n = 1000$		
	TP			**TP**			**TP**			**TP**		
	AIC	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**
CM	1			1			1			1		
PCM	1			1			1			1		
glasso	0.995	0.995	0.988	0.998	0.998	0.993	1	1	1	1	1	1
AL	0.995	0.99	0.983	0.998	0.998	0.985	1	1	0.998	1	1	1
SCAD	0.99	0.98	0.871	0.99	0.988	0.878	1	1	0.974	0.998	0.998	0.978
new	**AIC**	**BIC**										
glasso	0.995	0.993	0.988	0.998	0.998	0.993	1	1	1	1	1	1
AL	0.993	0.99	0.983	0.998	0.998	0.985	1	1	0.998	1	1	1
SCAD	0.96	0.956	0.871	0.964	0.939	0.878	0.999	0.994	0.974	0.985	0.969	0.978
DP												
glasso	0.995	0.995	0.988	0.998	0.998	0.993	1	1	1	1	1	1
AL	0.995	0.99	0.983	0.998	0.998	0.985	1	1	0.996	1	1	1
SCAD	0.99	0.98	0.869	0.99	0.985	0.8775	1	1	0.974	0.998	0.998	0.978
BG												
$H = 50$	0.99	0.983	0.985	0.993	0.985	0.993	1	1	1	1	1	1
new A/BIC	0.99	0.978	0.985	0.998	0.99	0.993	1	1	1	1	1	1
DP-boo	0.99	0.99	0.985	0.993	0.985	0.993	1	1	1	1	1	1
Huge												
ric	0.747			0.886			0.99			1		
stars	0.995			0.998			1			1		
ebic	0.995			0.998			1			1		

Table 9 illustrates the TP values for each combination operating on a 5 dimensional t-distributed dataset (***MVt data*** for short) with the degrees of freedom with 3 (***MV*t_3 *data*** for short), at sample sizes from 100 to 1000. As discussed in the Simulation section, we apply both the wrong and correct log-likelihood of the t-distribution (corresponding to the multivariate normal distribution and t-distribution, respectively) to the AIC formula (3.1) and BIC formula (3.2) and provide in Table 9 both the estimating results using the wrong and correct AIC/BIC formula. The results using the correct log-likelihood in the AIC and BIC formulae are denoted as new AIC BIC in Table 9, corresponding to the optimal results of glasso, BG, AL and SCAD methods selected by the corrected AIC/BIC formula.

It can be seen from Table 9 that though there are decreases in all TPs compared to the ***MVN data***, most of them still remain at a pretty high level ($> 99\%$), especially when $n \geq 500$. Almost all non-zero entries can be effectively estimated by any combination. SCAD$\star$CV produces slightly less non-zeros than other combinations of glasso, BG, AL and SCAD. Using the DP-glasso algorithm again does not affect the estimation results, neither does using DP-BG, DP-AL and DP-SCAD algorithms.

It is also noticeable that the corrected AIC/BIC formula has the strongest influence on SCAD in terms of the reductions of its TPs, though not much, 3% for SCAD$\star$AIC and SCAD$\star$BIC when $n = 100$, SCAD$\star$AIC when $n = 200$, as well as 5% for SCAD$\star$BIC when $n = 200$, which means the estimates of the non-zero entries by SCAD is affected most by changing the AIC/BIC formula, especially for SCAD$\star$BIC.

Compared to the ***MVN data*** Huge results, Huge$\star$ric shows the most dramatic drop of TPs when $n \leq 200$. Huge still provides fairly desirable estimates for the non-zeros in other situations. Also, all three criteria have higher power of detecting non-zeros when the sample size increases. More specifically, Huge$\star$stars and Huge$\star$ebic have the same ability of estimating non-zeros at all four sample sizes $(100, 200, 500, 1000)$, with more than 99.5% of the non-zeros detected when $n \leq 200$ Moreover, no non-zeros will be missed when $n \geq 500$. Huge$\star$ric has less power of detecting non-zeros than these two, with 74.7% discoveries of the non-

zeros when $n = 100$ and 88.6% when $n = 200$, 99% when $n = 500$ and all non-zeros are revealed when n increases to 1000.

Table 10: *The TNs for the* ***MV*** t_3 ***data*** *(*$p = 5$*) at* 4 *different sample sizes*

	$n = 100$			$n = 200$			$n = 500$			$n = 1000$		
	TN			**TN**			**TN**			**TN**		
	AIC	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**
CM	0			0			0			0		
PCM	0			0			0			0		
glasso	0.107	0.177	0.367	0.072	0.145	0.343	0.075	0.157	0.305	0.0583	0.142	0.29
AL	0.273	0.363	0.682	0.213	0.343	0.675	0.24	0.387	0.65	0.25	0.388	0.715
SCAD	0.417	0.592	0.884	0.412	0.567	0.845	0.504	0.643	0.839	0.681	0.746	0.896
new	**AIC**	**BIC**										
glasso	0.083	0.23	0.367	0.038	0.115	0.343	0.032	0.083	0.305	0.037	0.0467	0.29
AL	0.297	0.432	0.682	0.203	0.375	0.675	0.208	0.32	0.65	0.252	0.307	0.715
SCAD	0.833	0.895	0.884	0.78	0.921	0.845	0.888	0.963	0.839	0.933	0.963	0.896
DP												
glasso	0.107	0.177	0.367	0.073	0.145	0.343	0.075	0.157	0.305	0.058	0.142	0.29
AL	0.273	0.363	0.682	0.213	0.343	0.675	0.24	0.387	0.65	0.25	0.388	0.715
SCAD	0.415	0.592	0.886	0.412	0.566	0.843	0.504	0.643	0.839	0.681	0.753	0.896
BG												
$H = 50$	0.373	0.636	0.508	0.284	0.597	0.463	0.253	0.591	0.421	0.17	0.547	0.416
new A/BIC	0.368	0.72	0.508	0.137	0.522	0.463	0.053	0.302	0.421	0.073	0.116	0.416
DP-boo	0.378	0.642	0.508	0.292	0.597	0.462	0.257	0.59	0.422	0.18	0.553	0.417
Huge												
ric	0.67			0.532			0.315			0.198		
stars	0.153			0.15			0.217			0.226		
ebic	0.28			0.228			0.227			0.226		

Table 10 shows the TN values for all estimation combinations corresponding to the 5-dimensional ***MVt data*** with df $= 3$ (***MV*** t_3 ***data*** for short). Similar to the ***MVN data***, the zero entries are harder to estimate accurately than the non-zero entries, especially compared to the large non-zeros. Moreover, applying the

estimating methods to a t-distributed dataset will diminish the power of detecting true zero entries, even for the low dimensional dataset or given a large number of samples.

Before correcting the AIC/BIC formula, for glasso, AL and SCAD, CV selects noticeable more zeros than BIC and AIC at all sample sizes. However, for BG, BIC still selects the most sparse estimates at all sample sizes. Among all combinations, SCAD⋆CV is the best at estimating zero entries, leading to over 84% of successfully detected zeros when the sample size lies in 100 to 1000. AL⋆CV and BG⋆BIC perform the second best when $n \leq 200$ with over 60% discoveries of zeros, while AL⋆CV and SCAD⋆BIC become the second best combinations with about 65% detections when $n = 500$ and more than 70% detections when $n = 1000$. More specifically, SCAD⋆BIC outperforms AL⋆CV when $n = 1000$, yet it is still worse than the best combination SCAD⋆CV. SCAD⋆AIC can be seen as the third best choice which has around 40% of correct estimations when $n \leq 200$, increasing to 50% and 68% when n rises. glasso⋆CV, AL⋆BIC and AL⋆AIC have 20% to 36.7% chance of successful detection. Specifically, the TNs for glasso⋆CV drop as n increases, while the TNs for AL⋆BIC and AL⋆AIC decrease when n increases from 100 to 200 and start to increase when n rises from 200 to 1000. Lastly, glasso⋆BIC and glasso⋆AIC are two worst combinations in terms of the TNs, with lower than 17.6% detections for glasso⋆BIC and lower than 10% for glasso⋆AIC.

Another interesting fact is that all TNs for the three combinations of BG (BG⋆AIC, BG⋆BIC and BG⋆CV) decline when the sample size increases. In contrast, there is not any significant monotone trend for other combinations when n varies.

In addition, the DP-glasso series algorithms (DP-glasso, DP-BG, DP-AL and DP-SCAD) still have no noticeable effects on glasso series algorithms (glasso, BG, AL and SCAD).

Using the wrong AIC/BIC formula, which adopts the log-likelihood of the ***MVN data***, all combinations of glasso, BG, AL and SCAD with the AIC/BIC perform significantly worse for the ***MV*t_3 *data*** than the ***MVN data***, for all sample sizes. However, about half of the CV selected estimates contain more correctly estimated zeros for all sample sizes compared to the ***MVN data***. These estimates are glasso⋆CV

and BG⋆CV at all four sample sizes, AL⋆CV when $n = 100, 200$ and SCAD⋆CV at $n = 100$. This implies that it is possible for CV to work better when there are misuses, such as the variations of the distributions of the dataset. This happens because there is no requirement for any particular distribution of the dataset in CV's formula. By contrast, AIC and BIC are based on the exact log-likelihood of the dataset which is related to the distributions of the dataset. Thus the power of AIC and BIC are more likely to be affected by the changes of the distributions of the dataset. Additionally, the SCAD⋆CV results for the ***MV*t_3 *data*** are fairly desirable with at least 83.9% correct estimations for the zero entries at any sample sizes, which means that even if the dataset is not normally distributed and we do not replace the log-likelihood of the AIC/BIC formula with the correct one, we can still obtain satisfying estimates via SCAD⋆CV.

After modifying the AIC/BIC formula with the correct log-likelihood for the ***MVt data,*** significantly more zeros are correctly estimated by the SCAD combinations (SCAD⋆AIC, SCAD⋆BIC and SCAD⋆CV) compared to the misspecified AIC/BIC formula. Additionally, the TNs of glasso⋆BIC at $n = 100$, AL⋆AIC at $n = 10, 1000$, AL⋆BIC at $n = 100, 200$ and BG⋆BIC at $n = 100$ increase as well. Unexpectedly, less zero entries are correctly estimated for the rest of the combinations and sample sizes.

In conclusion, for the low dimensional ***MV*t_3 *data,*** estimates selected by the true AIC/BIC formula are not necessarily more accurate than using the log-likelihood of the ***MVN data***, unless we are applying the SCAD method to conduct estimation. In other words, the corrected estimates are not certainly sparser than the original ones.

When using the correct log-likelihood of the ***MV*t_3 *data*** in the AIC/BIC calculations, the majority of the TNs statistics are still worse than the results using the ***MVN data*** log-likelihood. AL⋆AIC, AL⋆BIC, glasso⋆AIC and glasso⋆BIC suffer from dramatic deterioration, reducing TNs to less than half from using the ***MVN data*** log-likelihood. The increases of the TNs are very tiny, which appear in glasso⋆CV at all four sample sizes, AL⋆CV when $n = 100, 200$, SCAD⋆CV when $n = 100$, SCAD⋆AIC when $n = 100, 200$ and SCAD⋆BIC when $n = 200$. However, interestingly, those TN decreases of SCAD are very minor. More specifically,

SCAD⋆BIC has the slightest reduction, while SCAD⋆CV results in the most obvious deterioration among all three selection criteria for SCAD. This tells us that even if the dataset is not normally distributed, by adopting the correct log-likelihood for AIC/BIC, SCAD method can still provide pretty satisfying estimates, especially by SCAD⋆BIC, which leads to more than 90% correct detections of the zero entries at any sample size.

The TNs for the ***MV***t_3 ***data*** in all four sample sizes for Huge⋆ric are higher than them for the ***MVN data***, while there is a decreasing trend as the sample size increases. Both Huge⋆stars and Huge⋆ebic lead to decreases of TNs compared to the ***MVN data*** for all sample sizes. Interestingly, TNs of Huge⋆stars and Huge⋆ebic become exactly the same again when $n = 1000$, which also happens to them for the ***MVN data*** with $p = 5$ and $n = 1000$. In addition, the TNs for Huge⋆stars slightly increase with the sample size, while they slightly decrease with the sample size for Huge⋆ebic.

5.2.1.2 df=4

Table 11: *The TPs for the* $\boldsymbol{MVt_4}$ ***data*** *(*$p = 5$*) at* 4 *different sample sizes*

	$n = 100$			$n = 200$			$n = 500$			$n = 1000$		
	TP			**TP**			**TP**			**TP**		
	AIC	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**
CM	1			1			1			1		
PCM	1			1			1			1		
glasso	1	1	1	1	1	1	1	1	1	1	1	1
AL	1	1	1	1	1	1	1	1	1	1	1	1
SCAD	0.995	0.99	0.955	1	0.995	0.965	0.999	0.998	0.985	0.999	0.995	0.994
new	**AIC**	**BIC**										
glasso	1	1	1	1	1	1	1	1	1	1	1	1
AL	1	1	1	1	1	1	1	1	1	1	1	1
SCAD	0.983	0.966	0.955	0.986	0.97	0.965	0.991	0.976	0.985	0.991	0.984	0.994
DP												
glasso	1	1	1	1	1	1	1	1	1	1	1	1
AL	1	1	1	1	1	1	1	1	1	1	1	1
SCAD	0.995	0.99	0.953	1	0.995	0.968	0.999	0.998	0.985	0.999	0.995	0.994
BG												
$H = 50$	0.998	0.99	1	1	0.998	1	1	1	1	1	1	1
new A/BIC	0.998	0.988	1	1	1	1	1	1	1	1	1	1
DP-boo	0.998	0.99	1	1	0.998	1	1	1	1	1	1	1
Huge												
ric	0.884			0.988			1			1		
stars	1			1			1			1		
ebic	1			1			1			1		

Table 12: *The TPs for the* $\boldsymbol{MVt_4}$ ***data*** *(*$p = 5$*) at* 4 *different sample sizes*

	$n = 100$			$n = 200$			$n = 500$			$n = 1000$		
	TN			**TN**			**TN**			**TN**		
	AIC	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**
CM	0			0			0			0		
PCM	0			0			0			0		
glasso	0.12	0.198	0.376	0.093	0.2	0.34	0.075	0.157	0.3	0.085	0.182	0.258
AL	0.311	0.458	0.662	0.3	0.448	0.688	0.325	0.457	0.678	0.399	0.508	0.693
SCAD	0.493	0.684	0.861	0.505	0.74	0.86	0.655	0.761	0.872	0.832	0.883	0.922
new	**AIC**	**BIC**										
glasso	0.107	0.227	0.376	0.048	0.157	0.34	0.045	0.107	0.3	0.075	0.1	0.258
AL	0.334	0.52	0.662	0.28	0.493	0.688	0.313	0.433	0.678	0.393	0.445	0.693
SCAD	0.844	0.942	0.861	0.901	0.969	0.86	0.939	0.99	0.872	0.955	0.995	0.922
DP												
glasso	0.12	0.198	0.377	0.093	0.2	0.342	0.075	0.157	0.3	0.087	0.182	0.258
AL	0.311	0.458	0.662	0.3	0.448	0.688	0.325	0.457	0.678	0.4	0.508	0.693
SCAD	0.497	0.684	0.863	0.505	0.74	0.86	0.656	0.761	0.872	0.833	0.883	0.922
BG												
$H = 50$	0.473	0.725	0.548	0.402	0.728	0.474	0.288	0.683	0.458	0.334	0.688	0.458
new A/BIC	0.46	0.78	0.548	0.205	0.69	0.474	0.149	0.509	0.458	0.237	0.383	0.458
DP-boo	0.478	0.725	0.548	0.403	0.728	0.477	0.292	0.678	0.458	0.342	0.69	0.462
Huge												
ric	0.555			0.328			0.22			0.163		
stars	0.156			0.221			0.243			0.312		
ebic	0.301			0.319			0.253			0.312		

Table 11 and Table 12 are the TPs and TNs for the 5-dimensional ***MVt data*** with df= 4 (***MV***t_4 ***data*** for short). They have very similar results, as well as almost the same variation trends, to the low dimensional ***MV***t_3 ***data*** , thus here we do not explain them detailedly again. The major differences between the ***MV***t_4 ***data*** and ***MV***t_3 ***data*** are that almost all the TPs and TNs of the ***MV***t_4 ***data*** are slightly higher than the ***MV***t_3 ***data*** for glasso, BG, AL and SCAD methods selected by any of the three criteria. This means higher estimating accuracies (both non-zeros and zeros) are achieved by glasso, BG, AL and SCAD when the dataset is t-distributed with a higher df. This is consistent with the fact that the t-distributions are closer to the normal distributions as their degrees of freedom increase and these estimating algorithms are developed based on the normal distributions. However, the TNs of Huge$\star$ric at all four sample sizes are lower than those of the 5-dimensional ***MV***t_3 ***data***. Huge$\star$stars and Huge$\star$ebic show slightly better estimates for zeros and non-zeros when the degrees of freedom increases, and interestingly, their TNs coincide again when $n = 1000$.

5.2.2 High dimensional cases

5.2.2.1 df=3

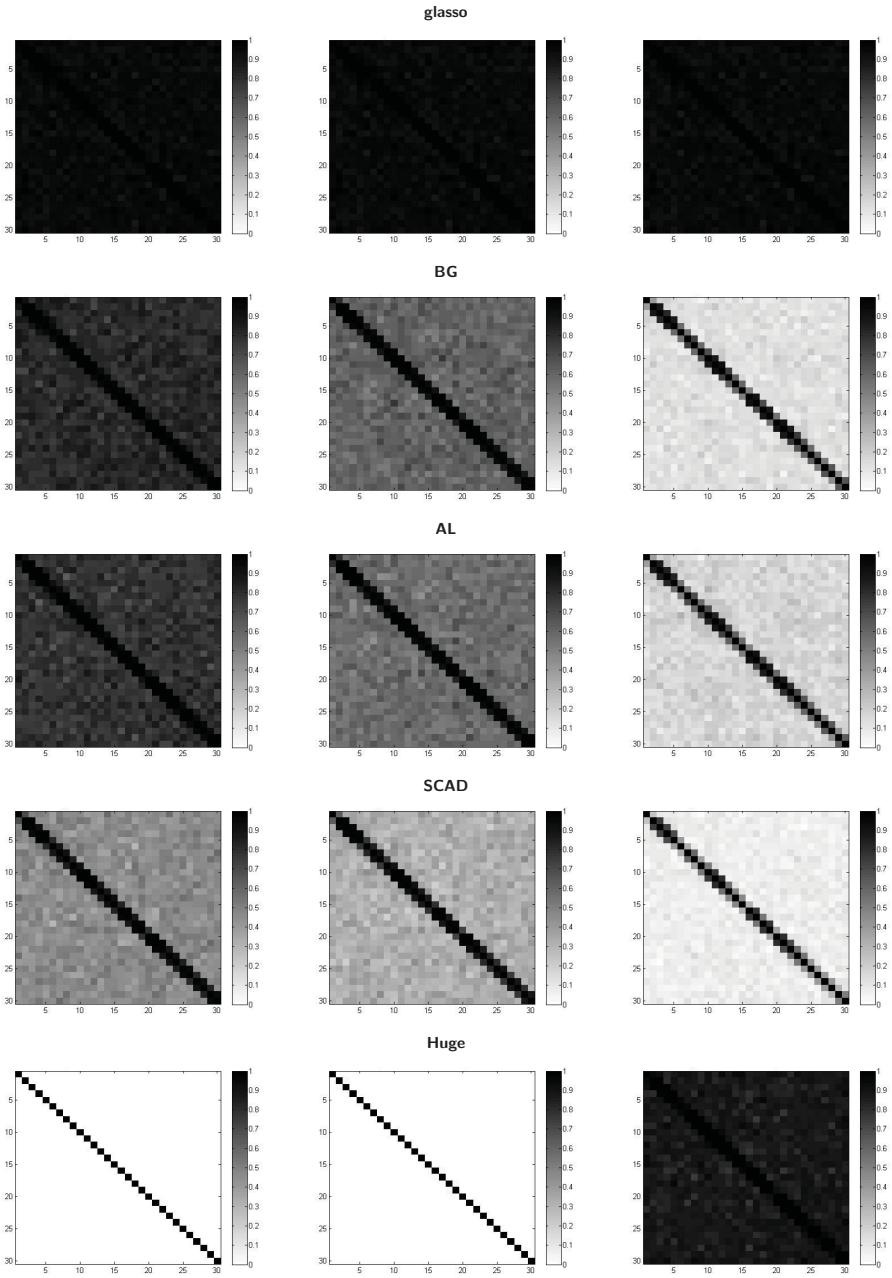

Figure 10: *The ASP plots for the* $\boldsymbol{MVt_3}$ ***data*** *(*$p = 30$*) generated from the tridiagonal matrix with* $a = 1.7$*, using the wrong AIC/BIC formula*

[Left panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the wrong AIC formula. [Middle Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the wrong BIC formula. [Right Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the CV criterion.

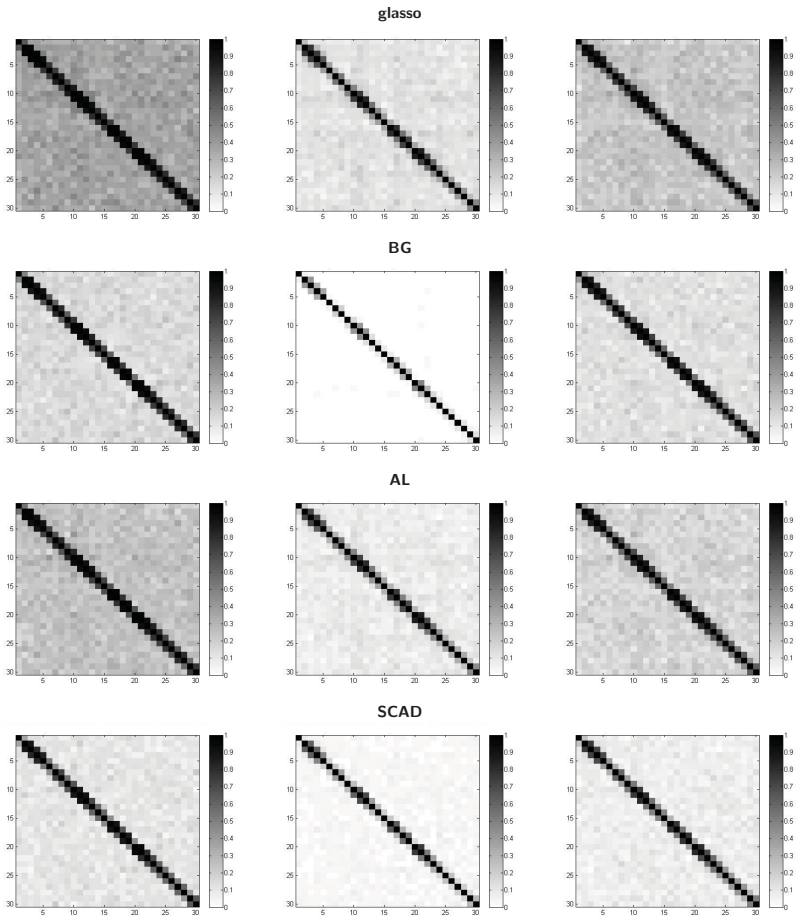

Figure 11: *The ASP plots for the* $\boldsymbol{MVt_3}$ ***data*** *(*$p = 30$*) generated from the tridiagonal matrix with* $a = 1.7$*, using the correct AIC/BIC formula*

[Left panel] The ASP plots of the glasso, BG, AL and SCAD estimates selected by the correct AIC formula. [Middle Panel] The ASP plots of the glasso, BG, AL and SCAD estimates selected by the correct BIC formula. [Right Panel] The ASP plots of the glasso, BG, AL and SCAD estimates selected by the CV criterion.

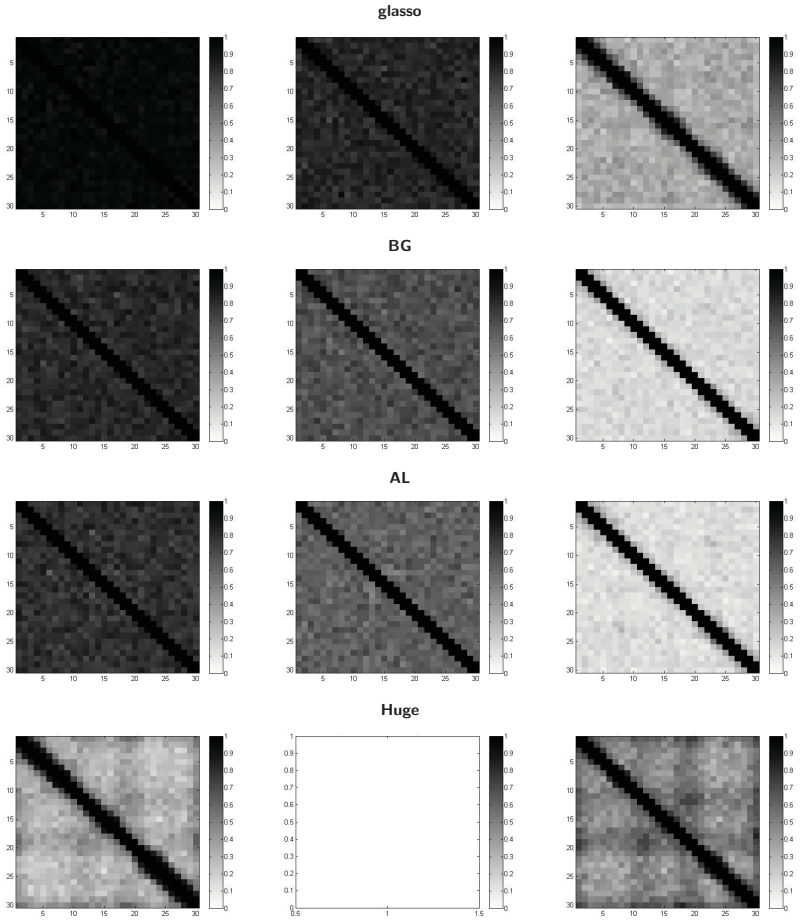

Figure 12: *The ASP plots for the* $\boldsymbol{MVt_3}$ ***data*** *(*$p = 30$*) generated from the tridiagonal matrix with* $a = 0.85$*, using the wrong AIC/BIC formula.*

[Left panel] The ASP plots of the glasso, BG, AL and Huge estimates selected by the wrong AIC formula. [Middle Panel] The ASP plots of the glasso, BG, AL and Huge estimates selected by the wrong BIC formula.[Right Panel] The ASP plots of the glasso, BG, AL and Huge estimates selected by the CV criterion.

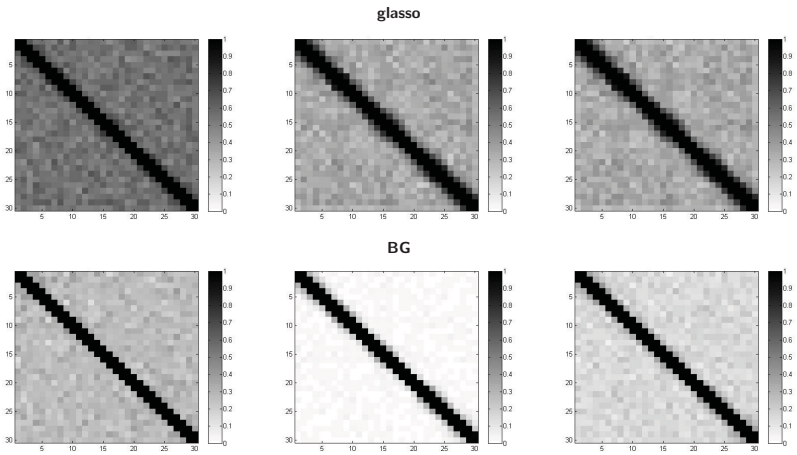

Figure 13: *The ASP plots for the* $\boldsymbol{MVt_3}$ ***data*** *(*$p = 30$*) generated from the tridiagonal matrix with* $a = 0.85$*, using the correct AIC/BIC formula.*

[Left panel] The ASP plots of the glasso and BG estimates selected by the correct AIC formula.

[Middle Panel] The ASP plots of the glasso and BG estimates selected by the correct BIC formula.

[Right Panel] The ASP plots of the glasso and BG estimates selected by the CV criterion.

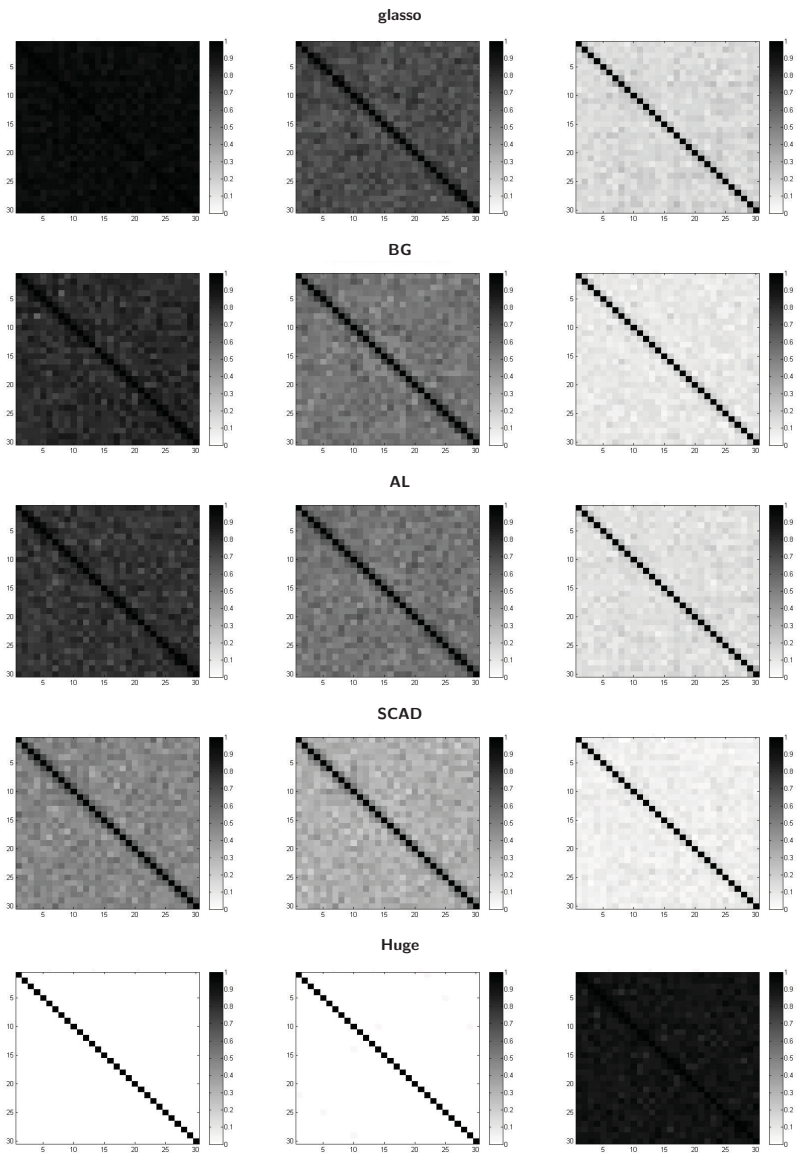

Figure 14: *The ASP plots for the* $\boldsymbol{MVt_3}$ ***data*** *(*$p = 30$*) generated from the exponential decay matrix, using the wrong AIC/BIC formula*

[Left panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the wrong AIC formula. [Middle Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the wrong BIC formula. [Right Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the CV criterion.

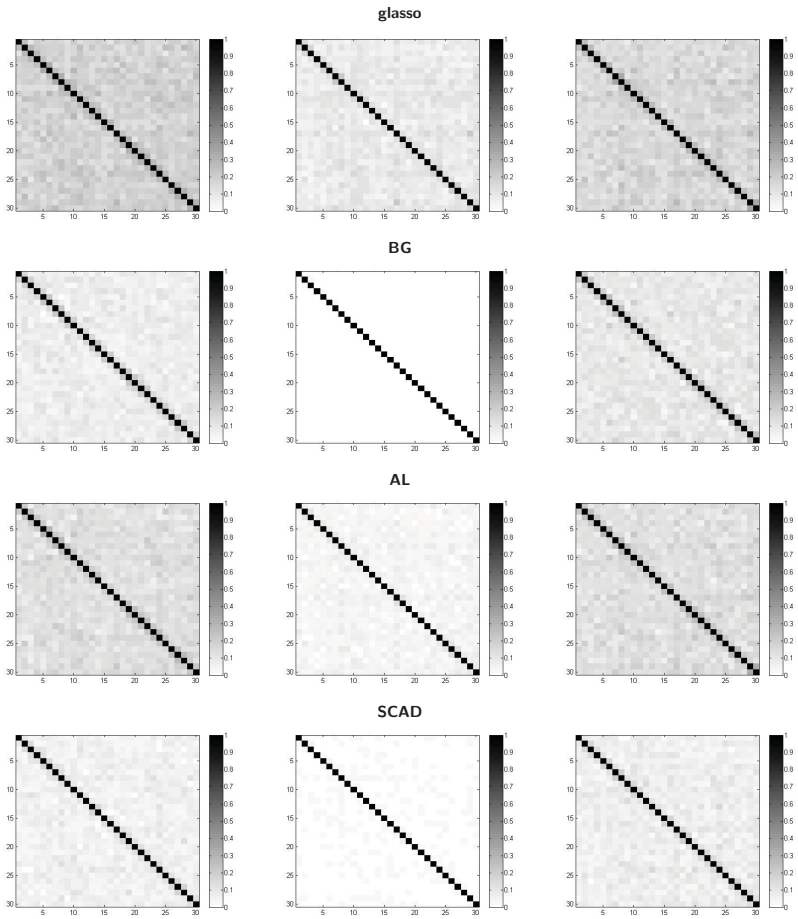

Figure 15: *The ASP plots for the* $\boldsymbol{MVt_3}$ ***data*** *(*$p = 30$*) generated from the exponential decay matrix, using the correct AIC/BIC formula*

[Left panel] The ASP plots of the glasso, BG, AL and SCAD estimates selected by the correct AIC formula. [Middle Panel] The ASP plots of the glasso, BG, AL and SCAD estimates selected by the correct BIC formula. [Right Panel] The ASP plots of the glasso, BG, AL and SCAD estimates selected by the CV criterion.

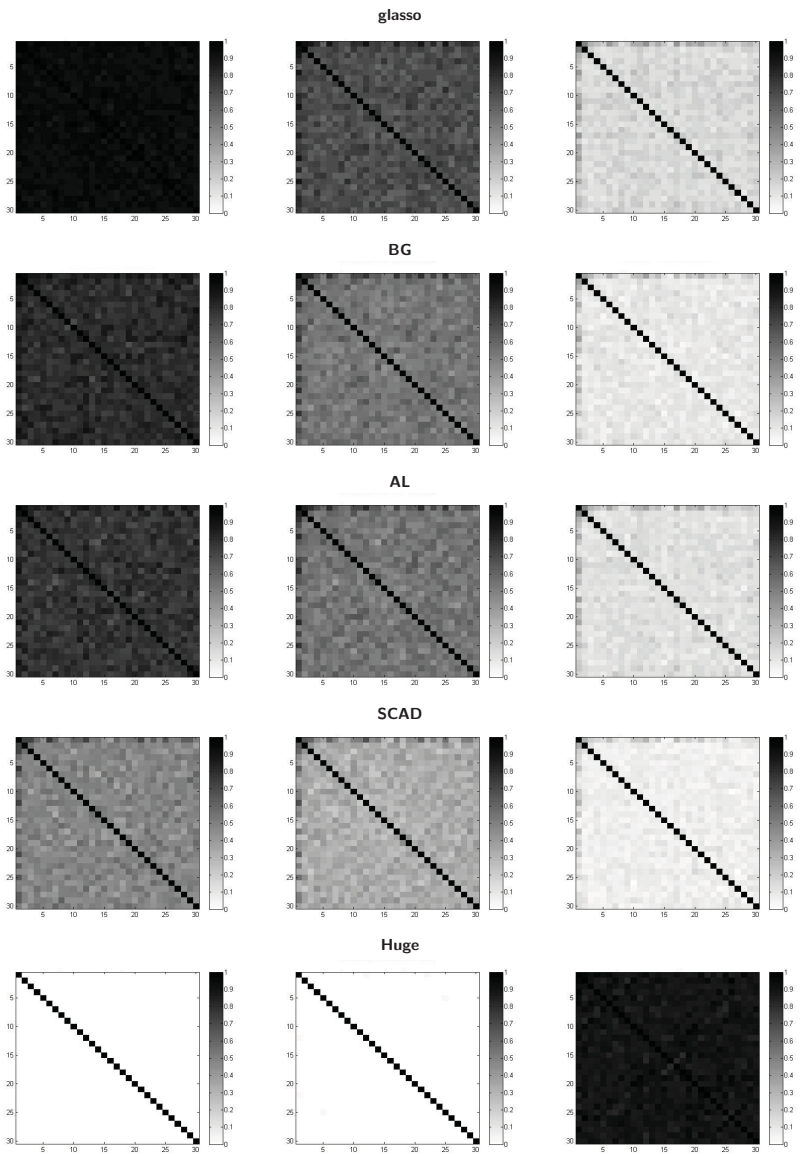

Figure 16: *The ASP plots for the* $\boldsymbol{MVt_3}$ ***data*** *($p = 30$) generated from the general matrix, using the wrong AIC/BIC formula*

[Left panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the wrong AIC formula. [Middle Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the wrong BIC formula. [Right Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the CV criterion.

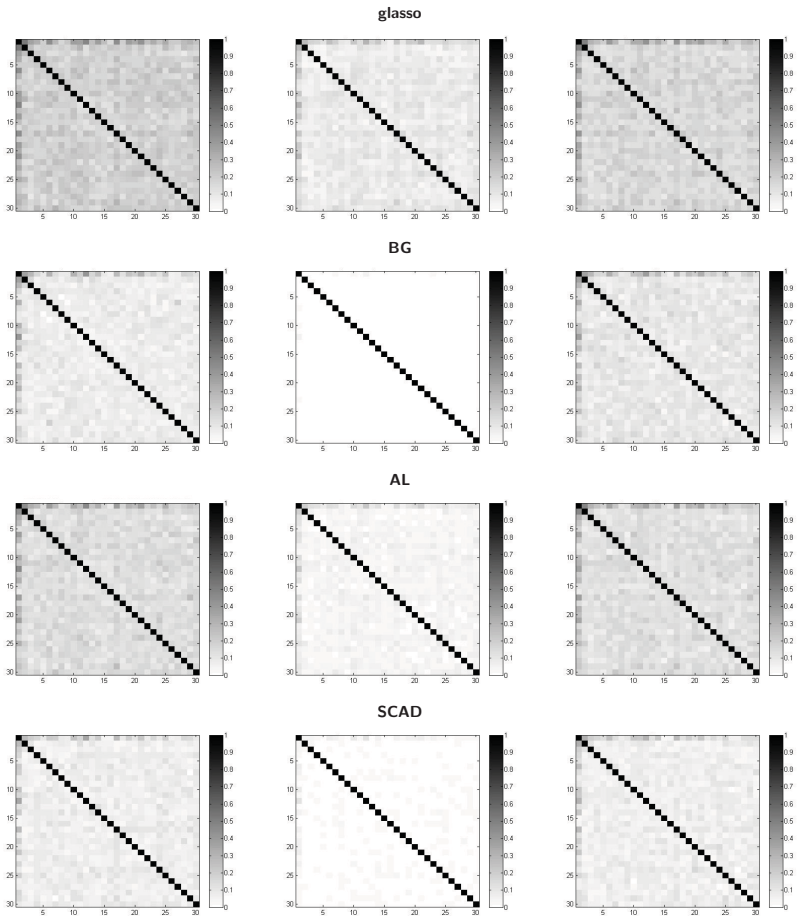

Figure 17: *The ASP plots for the* ***MVt***$_3$ ***data*** *($p = 30$) generated from the general matrix, using the correct AIC/BIC formula*

[Left panel] The ASP plots of the glasso, BG, AL and SCAD estimates selected by the correct AIC formula. [Middle Panel] The ASP plots of the glasso, BG, AL and SCAD estimates selected by the correct BIC formula. [Right Panel] The ASP plots of the glasso, BG, AL and SCAD estimates selected by the CV criterion.

Figures 10, 11, 14, 15, 16 and 17 are the ASPs for all the estimation combinations for the high dimensional ***MVt***$_3$ ***data*** corresponding to the three true precision matrices, including the tridiagonal matrix with $a = 1.7$, the exponential decay matrix and the general matrix. As we do for the the low dimensional ***MVt***$_3$ ***data,*** we present the ASPs for the glasso, BG, AL and SCAD methods using both the wrong (log-likelihood for the ***MVN data***) and correct log-likelihood (log-likelihood for the ***MVt***$_3$ ***data***) in their AIC and BIC formulae. Thus, each of the three matrices has

two ASPs figures accordingly, with the first figure being the glasso, BG, AL, SCAD and Huge results using the misspecified AIC/BIC formula and the latter figure being the ASPs for the glasso, BG, AL and SCAD results using the correct formulae.

Using the *MVN data* log-likelihood In terms of the principal and basic trends, before applying the correct log-likelihood to AIC/BIC, all combinations for the high dimensional $\boldsymbol{MVt_3}$ ***data*** perform the same as their behaviors for the high dimensional ***MVN data***, which indicates that the change of the distributions of the dataset does not affect the nature behaviors of the estimating methods and the selection criteria. More specifically, SCAD again generally produces the most accurate and sparsest estimates without losing the true graphical structure. glasso$\star$AIC results are always too dense to see the actual structure. BIC always selects sparser estimates than AIC does. Huge$\star$stars estimates are always excessively dense, while Huge$\star$ric and Huge$\star$ebic are always severely sparse and all of the Huge estimates lose the original graphical structure.

Besides the similarities to the ***MVN data*** performance, there are also some distinct behaviors of the estimating methods and the selection criteria for the $\boldsymbol{MVt_3}$ ***data***.

Using the ***MVN data*** log-likelihood, the first obvious difference for the high dimensional $\boldsymbol{MVt_3}$ ***data*** is that the AIC results of glasso, BG and AL methods are extremely dense so that the true graphical structure is lost. Among all the methods AIC is applied to, SCAD is the only method whose results are not excessively dense, even if use the log-likelihood of the ***MVN data*** for AIC when the dataset is actually t-distributed. This fact further implies the superiority or the stability of SCAD over the other estimating methods.

Another obvious difference is that CV selects much sparser estimates for the high dimensional $\boldsymbol{MVt_3}$ ***data*** than BIC does, while BIC selects sparser estimates than CV for the high-dimensional ***MVN data***.

We recall that for the low dimensional $\boldsymbol{MVt_3}$ ***data,*** although CV selects much sparser estimates for glasso, AL and SCAD, BIC selects the most sparse estimates for BG at all sample sizes. However, for the high dimensional $\boldsymbol{MVt_3}$ ***data,*** the

sparsest estimates of BG are selected by CV, instead of BIC, which makes CV produce the sparsest results for all glasso, BG, AL and SCAD methods with respect to all three precision matrices, provided that the correct log-likelihood is not applied to the AIC/BIC formula.

When we consider t-distributed data, the estimation precision of some of the combinations is also affected. Firstly, the smallest non-zero value that can be detected by CV increases the most, which means a decrease in the estimation precision in estimating non-zeros. Although CV is the best selection criterion which provides the sparsest estimates, it loses the power of detecting non-zero entries, which is what it can achieve for the ***MVN data***. More specifically, for example in the tridiagonal matrix with $a = 1.7$, where the smallest entry is 0.19 and the largest one is 0.425, all methods selected using CV at times fail to detect the non-zero entries which are less than 0.32. For the entries between 0.19 and 0.32, smaller entries have smaller frequencies of detection. However, all entries in this tridiagonal matrix can always be successfully detected by all combinations of CV for the ***MVN data***. This may indicate that for the ***MVt_3 data***, CV may shrink the entries more than for the ***MVN data***, leading to not only sparser estimates, but also over shrinkage for some relatively small entries (e.g. ≤ 0.32). For the exponential decay matrix, where the largest off diagonal entry is 0.1353, the non-zero entries are very hard to estimate. Moreover, it seems that since all the non-zero entries are smaller than the lowest limit that CV combinations can detect, all the estimated non-zeros are randomly located. This is consistent with the conclusions from the ***MVN data*** results. Similarly, for the general matrix with the largest entry being 0.1811, all its non-zero entries can only be detected occasionally, with slightly higher frequencies of detections for entries which are larger than 0.15 compared to other smaller entries.

Using the wrong log-likelihood formula while calculating the AIC/BIC, all their results become denser for the ***MVt_3 data*** than the ***MVN data***. For BIC results in the tridiagonal case with $a = 1.7$, there is no obvious changes in the power to detect non-zeros, which may due to the fact that its non-zero entries are larger than the other two matrices. For the exponential decay matrix and the general matrix, relatively large non-zero values (e.g. 0.1353 for the exponential decay matrix and

≥ 0.12 for the general matrix) can be estimated more often by BIC, but at the same time results in more false estimations of non-zeros in other positions, which makes the estimates much denser but the basic structure is maintained. However, interestingly, even though AIC leads to extremely dense estimates that the original structure is lost for glasso, BG and AL methods, the lowest limit of the non-zero entries that can be sensitively detected by AIC seems decrease slightly, so that some relatively large values in the exponential decay matrix and the general matrix (e.g. 0.1353 for the exponential decay matrix and ≥ 0.14 for the general matrix) are detected less often than what can be estimated for the ***MVN data***. On the other hand, the non-zero entries in the tridiagonal matrix with $a = 1.7$ are not affected. There are significantly more false estimations of the non-zeros for both AIC and BIC results, where glasso⋆AIC, BG⋆AIC and AL⋆AIC lose their abilities to reflect the true graphical model.

Huge⋆stars still provides excessively dense estimates, while Huge⋆ebic and Huge⋆ric suffers from severe underselection problems. All of them lose the true graphical structure.

Using the *MVt data* log-likelihood Using the correct AIC/BIC formula, there appears to be many more differences in the results than the low dimensional ***MVt*$_3$ *data***. Both AIC and BIC results become significantly sparser than the results based on the wrong log-likelihood. As a result, the BIC provides sparser estimates than CV again, just as we saw for the ***MVN data***. Additionally, after applying the correct log-likelihood, BIC is still sparser than AIC, which is to be expected.

However, the price paid for sparser results is that the ability of correctly estimating non-zeros is also diminished. In other words, the smallest value that can always be successfully detected rises. Consequently, in the tridiagonal matrix with $a = 1.7$, the frequencies of some of the non-zeros (e.g. between 0.19 and 0.32) decline, resulting in a very similar level to the Huge⋆ric results for the high dimensional ***MVN data***. The most dramatic decreases occur to the BG method, compared to glasso, AL and SCAD methods. The power of AIC to detect non-zeros is not affected much for the tridiagonal matrix. This may be due to the nature of AIC to

produce many non-zero estimates and the relatively large non-zero entries of the tridiagonal matrix compared to the other two matrices. For the exponential decay matrix and the general matrix with smaller non-zero entries than the tridiagonal matrix, their estimation results are even sparser than the estimates without using the correct formula, consequently, they almost lose the true graphical structure. This is especially true for BIC.

Using the correct AIC/BIC formula, BG and SCAD provide sparser estimates than glasso and AL, for both AIC and BIC cases. In addition, the estimates selected by AIC are at almost the same sparsity levels as the CV results. Thus if the true entries consist of many large values, such as values greater than 0.4, we recommend to use glasso$\star$BIC, AL$\star$BIC, SCAD$\star$AIC or SCAD$\star$BIC. However, if the majority of the true entries are small, say < 0.2, the BIC combinations are strongly not recommended because they will fail to capture the original structure due to the excessively high sparsity invoked in its estimates. Instead, the AIC combinations are recommended, especially the SCAD$\star$AIC and BG$\star$AIC if one is expecting more sparsity.

Figure 12 contains all the ASP plots of all the combinations that can produce estimates and finish all the repetitions, without being stuck on the way. For all the provided results of the ***MV*t_3 *data***, they show very similar trends to the ***MVN data***, while comparing the results of the two tridiagonal matrices (with a being 1.7 and 0.85, respectively).

The Huge results still vary the most among all the combinations, while changing the constant a in the tridiagonal matrix. Instead of providing either excessively sparse or excessively dense estimates for the tridiagonal matrix with $a = 1.7$, Huge$\star$stars and Huge$\star$ebic combinations are able to capture the true graphical structure now. The performance of estimating non-zeros are dramatically improved for Huge$\star$ebic, though the ability of correctly detecting zero entries weakens a bit. Also, the sparsity of the Huge$\star$stars estimates is greatly boosted, with a lot more correctly estimated zeros. In addition, there are still no return values of the Huge$\star$ric combination for the ***MV*t_3 *data*** generated from the tridiagonal matrix with $a = 0.85$, which is the same situation as the ***MVN data***.

For the glasso, AL, BG and SCAD combinations using the wrong AIC/BIC formula, all the non-zero entries are estimated better, which is the same as the ***MVN data*** case. In terms of the zero entry detection, only AL$\star$CV performs better. What's different from the ***MVN data*** is that the SCAD method gets stuck on a repetition and can not finish all the estimation, making SCAD is not applicable to the ***MV*t_3 *data***.

If the correct AIC/BIC formula is used for the glasso, AL, BG and SCAD combinations, AL and SCAD methods get stuck halfway the estimation. For the obtained glasso and BG results, all the non-zero entries are estimated better, while all the zero entries are detected worse. All the AIC and BIC results using the correct formula are much sparser than those using the wrong formulae, which is the same behaviour as the ***MVN data***.

The above results tell us that the tridiagonal matrix with $a = 0.85$, which contains many large entries and is very clsoe to be not positive definite, will cause more problems for the ***MV*t_3 *data*** than the ***MVN data***, with more estimation combinations become inapplicable. Also, the correct AIC/BIC formula increases the risk

of getting stuck for the ***MV***t_3 ***data***, resulting in more inapplicable combinations. In terms of the results that can be obtained from the ***MV***t_3 ***data***, they present very similar trends to the ***MVN data***, between two tridiagonal matrices.

Table 13: *The TPs and TNs for the tridiagonal matrix with* $a = 1.7$ *for the* $\boldsymbol{MVt}_3$ ***data*** *(*$p = 30$*)*

Ω	**tri** $a = 1.7$					
	TP			**TN**		
	AIC	**BIC**	**CV**	**AIC**	**BIC**	**CV**
CM	1			0		
PCM	1			0		
glasso	0.993	0.971	0.723	0.049	0.242	0.768
AL	0.972	0.942	0.683	0.218	0.422	0.8278
SCAD	0.904	0.874	0.535	0.541	0.675	0.927
new						
glasso	0.849	0.525	0.723	0.635	0.881	0.768
Adaptive	0.801	0.488	0.683	0.753	0.919	0.828
SCAD	0.664	0.353	0.535	0.898	0.969	0.927
DP						
glasso	0.993	0.971	0.723	0.049	0.243	0.768
AL	0.972	0.941	0.683	0.218	0.423	0.827
SCAD	0.903	0.873	0.535	0.541	0.675	0.927
BG						
$H = 50$	0.974	0.947	0.693	0.185	0.387	0.87
new A/BIC	0.777	0.181	0.693	0.845	0.998	0.87
DP-boo	0.974	0.947	0.693	0.185	0.39	0.869
Huge						
ric	0.005			1		
stars	0.985			0.124		
ebic	0.008			0.996		

Table 14: *The TPs and TNs for the exponential decay matrix and the general matrix for the* $\boldsymbol{MVt_3}$ ***data*** *(*$p = 30$*)*

Ω	exp						gen					
	TP			TN			TP			TN		
	AIC	BIC	CV	AIC	BIC	CV	AIC	BIC	CV	AIC	BIC	CV
CM	1			0			1			0		
PCM	1			0			1			0		
glasso	0.945	0.726	0.188	0.059	0.303	0.839	0.941	0.7	0.185	0.054	0.283	0.837
AL	0.799	0.595	0.16	0.223	0.455	0.869	0.784	0.562	0.156	0.205	0.428	0.868
SCAD	0.53	0.374	0.089	0.525	0.681	0.932	0.497	0.352	0.093	0.493	0.653	0.941
new	**AIC**	**BIC**										
glasso	0.213	0.102	0.188	0.817	0.913	0.839	0.216	0.11	0.185	0.8	0.916	0.837
Adaptive	0.158	0.051	0.16	0.87	0.962	0.869	0.166	0.064	0.156	0.855	0.956	0.868
SCAD	0.076	0.013	0.089	0.944	0.991	0.932	0.095	0.032	0.093	0.935	0.991	0.941
DP												
glasso	0.944	0.726	0.188	0.06	0.304	0.839	0.941	0.7	0.185	0.054	0.283	0.838
AL	0.799	0.596	0.16	0.223	0.454	0.869	0.784	0.562	0.156	0.205	0.428	0.868
SCAD	0.53	0.373	0.089	0.525	0.681	0.932	0.497	0.353	0.093	0.493	0.653	0.941
BG												
$H = 50$	0.829	0.59	0.127	0.193	0.456	0.904	0.812	0.558	0.123	0.19	0.436	0.912
new A/BIC	0.105	0.002	0.127	0.929	1	0.904	0.105	0.023	0.123	0.93	1	0.912
DP-boo	0.829	0.59	0.127	0.193	0.457	0.904	0.813	0.558	0.123	0.188	0.438	0.912
Huge												
ric	0.002			0.998			0.023			0.999		
stars	0.905			0.103			0.897			0.102		
ebic	0			1			0.022			1		

Table 13 and Table 14 detail the TPs and TNs for the high dimensional $\boldsymbol{MVt_3}$ ***data***, corresponding to the three precision matrices respectively. Each of the dataset is estimated by all the combinations, and the results selected by the correct AIC/BIC

formula (applying the log-likelihood of $\boldsymbol{MVt_3}$ ***data*** to the formula) are also included. The TPs and TNs for the $\boldsymbol{MVt_3}$ ***data*** generated from the tridiagonal matrix with $a = 0.85$ are displayed in the appendix. The tables show consistent conclusions with the ASP plots, thus we don't explain them in detail again.

5.2.2.2 df=4

Table 15: *The TPs and TNs for the tridiagonal matrix with $a = 1.7$ for the* ***MVt***$_4$ ***data*** *($p = 30$)*

Ω	**tri** $a = 1.7$					
	TP			**TN**		
	AIC	**BIC**	**CV**	**AIC**	**BIC**	**CV**
CM	1			0		
PCM	1			0		
glasso	0.993	0.969	0.779	0.062	0.343	0.768
AL	0.97	0.937	0.741	0.252	0.506	0.827
SCAD	0.911	0.869	0.595	0.576	0.73	0.926
new						
glasso	0.874	0.487	0.779	0.658	0.925	0.768
Adaptive	0.837	0.482	0.741	0.756	0.941	0.827
SCAD	0.738	0.395	0.595	0.886	0.973	0.926
DP						
glasso	0.993	0.969	0.779	0.062	0.341	0.768
AL	0.97	0.937	0.741	0.252	0.506	0.827
SCAD	0.911	0.869	0.595	0.576	0.73	0.926
BG						
$H = 50$	0.967	0.929	0.749	0.242	0.511	0.856
new A/BIC	0.809	0.155	0.749	0.833	0.999	0.856
DP-boo						
Huge						
ric	0.033			1		
stars	0.989			0.117		
ebic	0			1		

Table 16: *The TPs and TNs for the tridiagonal matrix with $a = 1.7$ for the* ***MV****t*$_4$ ***data*** *($p = 30$)*

Ω	**exp**						**gen**					
	TP			**TN**			**TP**			**TN**		
	AIC	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**
CM	1			0			1			0		
PCM	1			0			1			0		
glasso	0.934	0.581	0.184	0.071	0.463	0.862	0.93	0.57	0.177	0.065	0.434	0.863
AL	0.769	0.508	0.16	0.257	0.548	0.885	0.753	0.485	0.15	0.249	0.531	0.886
SCAD	0.501	0.324	0.103	0.561	0.75	0.935	0.468	0.277	0.094	0.547	0.749	0.939
new	**AIC**	**BIC**										
glasso	0.184	0.057	0.184	0.854	0.958	0.862	0.181	0.069	0.177	0.856	0.958	0.863
Adaptive	0.151	0.04	0.16	0.887	0.974	0.885	0.154	0.054	0.15	0.888	0.973	0.886
SCAD	0.098	0.013	0.103	0.936	0.994	0.935	0.01	0.031	0.094	0.935	0.993	0.939
DP												
glasso	0.934	0.582	0.184	0.071	0.463	0.862	0.93	0.569	0.177	0.065	0.433	0.863
AL	0.769	0.508	0.16	0.257	0.548	0.885	0.753	0.485	0.15	0.249	0.527	0.886
SCAD	0.501	0.324	0.103	0.562	0.75	0.935	0.468	0.277	0.094	0.547	0.747	0.939
BG												
$H = 50$	0.775	0.446	0.138	0.249	0.616	0.907	0.758	0.423	0.128	0.233	0.598	0.912
new A/BIC	0.107	0.001	0.138	0.932	1	0.907	0.101	0.022	0.128	0.933	1	0.912
DP-boo	0.775	0.447	0.138	0.249	0.616	0.907	0.758	0.422	0.128	0.233	0.598	0.913
Huge												
ric	0.006			0.996			0.026			0.996		
stars	0.911			0.1			0.905			0.102		
ebic	0			1			0.022			1		

Table 15 and Table 16 are the TPs and TNs for the high dimensional ***MV****t*$_4$ ***data***, corresponding to the same three precision matrices as the ***MV****t*$_3$ ***data***. The ***MV****t*$_4$ ***data*** shows very similar estimation results and almost the same trends as the ***MV****t*$_3$

data does. Also their results are so close that the ASPs are very similar and thus can hardly reflect their differences, so we don't include the ASPs for the $\boldsymbol{MVt_4}$ ***data*** here, but include them in the appendix. All the plots and tables of the $\boldsymbol{MVt_4}$ ***data*** generated from the tridiagonal matrix with $a = 0.85$ are shown in the appendix too.

However, we can still find the small differences between the $\boldsymbol{MVt_4}$ ***data*** and the $\boldsymbol{MVt_3}$ ***data*** using their TP and TN tables. Using the incorrect log-likelihood for the AIC/BIC formula, all the TPs reduce while all the TNs increase for all AIC and BIC combinations (with glasso, BG, AL and SCAD methods), which means the estimated graphs become slightly sparser.

Using the correct log-likelihood, all the TPs of the AIC combinations (with glasso, BG, AL and SCAD methods) increase and all the TNs decrease for the tridiagonal matrix with $a = 1.7$, indicating denser estimates than the $\boldsymbol{MVt_3}$ ***data*** results. However, for the BIC combinations for this matrix, all the TPs decline and all the TNs rise, indicating sparser estimates when the degrees of freedom (df for short) increases. For the exponential decay matrix and the general matrix, all the TPs become smaller and all TNs are larger for all the glasso and AL combinations with AIC or BIC, which are the same behaviors of AIC and BIC using the incorrect log-likelihood. If we apply SCAD to these two matrices, the estimated graphs will become denser if selected by AIC and sparser when selected by BIC. For BG$\star$BIC, the estimates are slightly sparser. However, for BG$\star$AIC, both the TP and TN increase for the exponential decay matrix and both of them decrease for the general matrix.

For the tridiagonal matrix with $a = 1.7$, all the TPs increase and all the TNs decrease for all the CV combinations, which is completely the opposite of the AIC/BIC behaviors for all three precision matrices. Thus the estimates of the tridiagonal matrix selected by CV will become slightly denser when the df increases. However, for the exponential decay matrix and the general matrix, there are both increases and decreases for the TPs and the TNs of glasso, AL and SCAD, while there are only increases of TPs and TNs for BG.

For Huge, the ric and stars estimates for the $\boldsymbol{MVt_4}$ ***data*** become denser than the $\boldsymbol{MVt_3}$ ***data*** for all three precision matrices, since their TPs rise and TNs drop. The

ebic estimates remain almost unchanged for all three cases.

Although what changes are small, we can still conclude that the df is indeed an influential factor on the results, especially for the tridiagonal matrix which leads to the same variation patterns for all methods. Also for most of the estimating methods and the true precision matrices, after correcting the log-likelihood, the AIC estimates tend to be denser and the BIC estimates become sparser than the dataset with the smaller df. Thus we believe it is reasonable to conjecture that the changes will be magnified when the df is further increased.

5.2.3 Nonparanormal transformation effects (for MVt_4 *data*)

5.2.3.1 The low dimensional cases

Table 17: *The TPs for the* MVt_4 ***data*** *($p = 5$) using the nonparanormal transformation function* `huge.npn()` *at* 4 *different sample sizes*

	$n = 100$			$n = 200$			$n = 500$			$n = 1000$		
	TP			**TP**			**TP**			**TP**		
	AIC	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**
CM	**1**			**1**			**1**			**1**		
PCM	**1**			**1**			**1**			**1**		
glasso	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
AL	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
SCAD	**1**	**1**	**0.998**	**1**	**1**	**1**	**1**	**0.999**	**0.999**	**1**	**1**	**1**
BG												
$H = 50$	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Huge												
ric	**1**			**1**			**1**			**1**		
stars	**1**			**1**			**1**			**1**		
ebic	**1**			**1**			**1**			**1**		

Table 17 contains the TPs at all four sample sizes for the low dimensional MVt_4 ***data*** to which applied the nonparanormal transformation function `huge.npn()` provided in the package `Huge` in `R` is applied.

On comparing the results to the original dataset, all the TPs increase for all combinations at all four sample sizes. This implies that the nonparanormal transformation function is indeed helpful in improving the ability of detecting non-zero entries for the low dimensional MVt_4 ***data*** with sample sizes from 100 to 1000, no matter which method or criterion is adopted.

Table 18: *The TNs for the* $\boldsymbol{MVt_4}$ ***data*** *($p = 5$) using the nonparanormal transformation function* `huge.npn()` *at* 4 *different sample sizes*

	$n = 100$			$n = 200$			$n = 500$			$n = 1000$		
	TN			**TN**			**TN**			**TN**		
	AIC	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**
CM	0			0			0			0		
PCM	0			0			0			0		
glasso	0.218	0.237	0.298	0.238	0.245	0.244	0.299	0.299	0.299	0.347	0.347	0.347
AL	0.565	0.57	0.6	0.663	0.663	0.665	0.815	0.815	0.815	0.867	0.867	0.867
SCAD	0.646	0.81	0.863	0.788	0.898	0.888	0.809	0.873	0.866	0.823	0.823	0.823
BG												
$H = 50$	0.712	0.737	0.72	0.758	0.758	0.758	0.865	0.865	0.865	0.857	0.857	0.857
Huge												
ric	0.265			0.295			0.31			0.308		
stars	0.218			0.238			0.302			0.347		
ebic	0.352			0.327			0.32			0.347		

Table 18 consists of the TN values corresponding to all methods and criteria combinations based on the 5-dimensional transformed $\boldsymbol{MVt_4}$ ***data*** using the function `huge.npn()`.

Generally, most of the TNs become larger for all the combinations at all sample sizes, indicating more zero entries are correctly estimated in most of the cases. We found that the most dramatic increases of the TNs are more likely to occur to the combinations with the TNs between 0.3 to 0.5, and their TNs are often doubled to between 0.7 and 0.867 after applying the function `huge.npn()`. This means that if one combination can only detect less than half of the zero entries (but no less than 30%), the function `huge.npn()` is very likely to be effective in improving the estimation results, leading to a fairly desirable sparsity level of the estimates. However, if the original TN value is relatively large (e.g. more than 0.64) or fairly small (e.g. less than 0.2) before applying the transformation function, the improvements are more likely to be insignificant. For example, a TN$= 0.12$ is only increased to 0.218

(glasso⋆AIC at $n = 100$), and a TN$= 0.761$ is increased to 0.873 (SCAD⋆BIC at $n = 500$).

However, there are also decreases in the TNs, which are the glasso⋆CV and AL⋆CV at $n = 100, 200$, SCAD⋆CV at $n = 500, 1000$, SCAD⋆AIC/BIC at $n = 1000$ and Huge⋆ric at $n = 200$. An interesting fact behind these decreases is that they either happen to the CV combinations with glasso or AL at moderate sample sizes (100 and 200), or they happen to SCAD combinations at large sample sizes (500 and 1000). The former situation illustrates the different behaviors of CV from AIC/BIC, or the less desirable cooperations of CV than AIC/BIC with the transformation function when the ratio of the sample size to the dimension is no larger than 40 ($p = 5$, $n = 100, 200$). The latter situation is more interesting, since the original TNs in those situations are pretty large (more than 0.832), thus this indicates that if most of the zero entries are already successfully detected (e.g. more than 83% of them are captured), the function `huge.npn()` may not be helpful to further improve the estimates, and sometimes the results can get slightly worse if applying the transformation function.

Moreover, the only decrease in the TN for Huge method occurs to Huge⋆ric at $n = 200$, from 0.328 to 0.295, yet the reduction is obviously not due to the original high level of detecting zeros (one of the reasons for decreases in AIC/BIC results).

5.2.3.2 The high dimensional cases

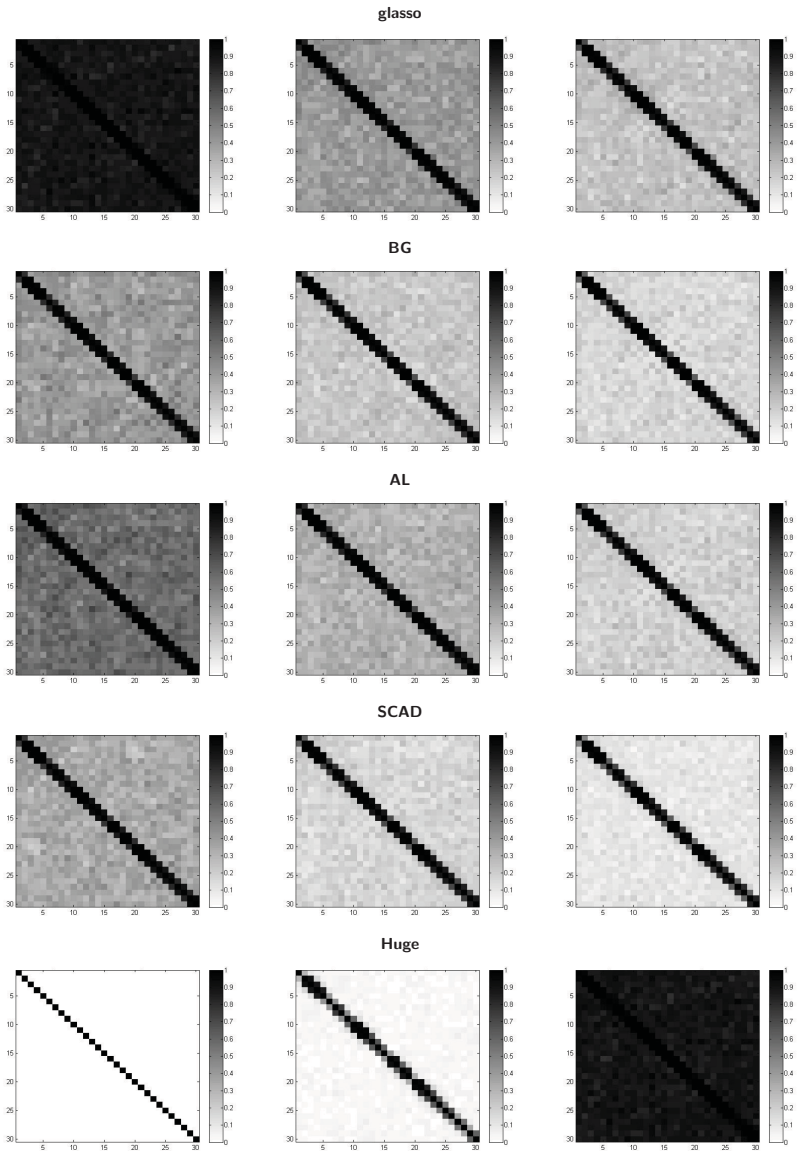

Figure 18: *The ASP plots for the* $\boldsymbol{MVt_4}$ ***data*** *(*$p = 30$*) generated from the tridiagonal matrix with* $a = 1.7$*, using the nonparanormal transformation function* `huge.npn()` *and the wrong AIC/BIC formula*

[Left panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the wrong AIC formula, based on the transformed data. [Middle Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the wrong BIC formula, based on the transformed data. [Right Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the CV criterion, based on the transformed data.

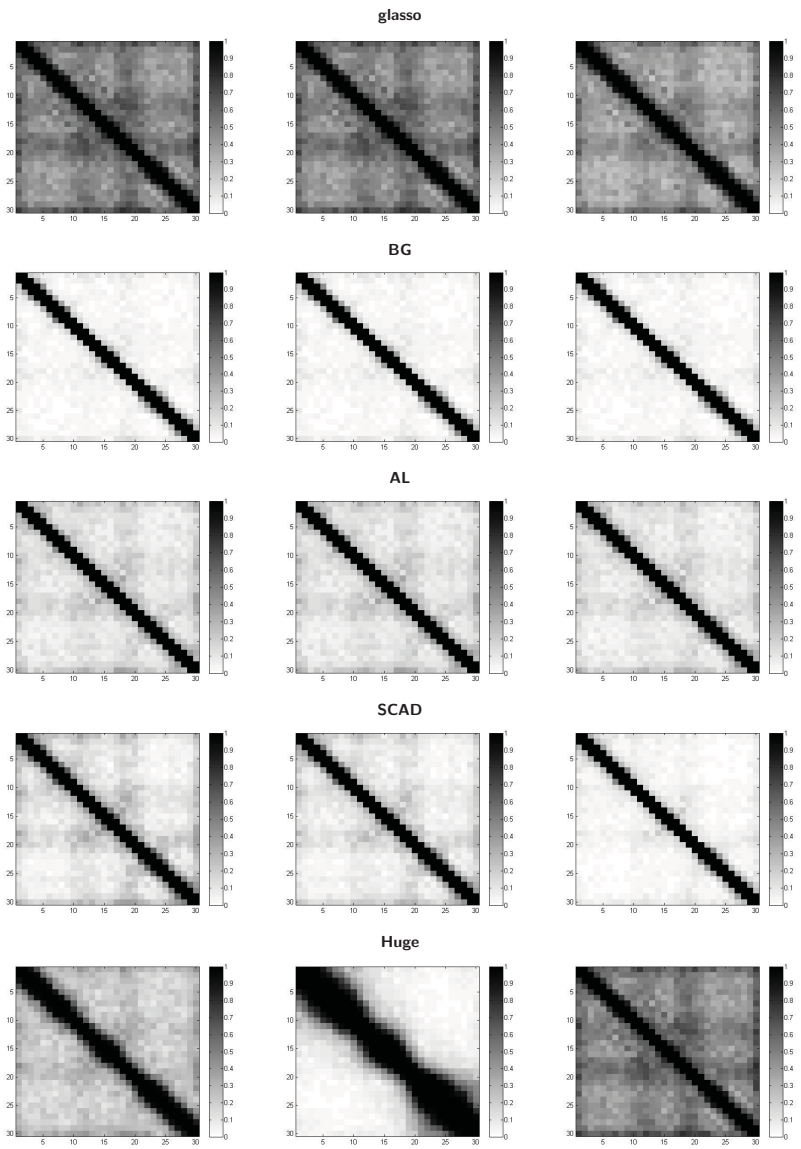

Figure 19: *The ASP plots for the* ***MV****t*$_4$***-data*** *(*$p = 30$*) generated from the tridiagonal matrix with* $a = 0.85$*, using the nonparanormal transformation function* `huge.npn()` *and the wrong AIC/BIC formula.*

[Left panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the wrong AIC formula, based on the transformed data. [Middle Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the wrong BIC formula, based on the transformed data. [Right Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the CV criterion, based on the transformed data.

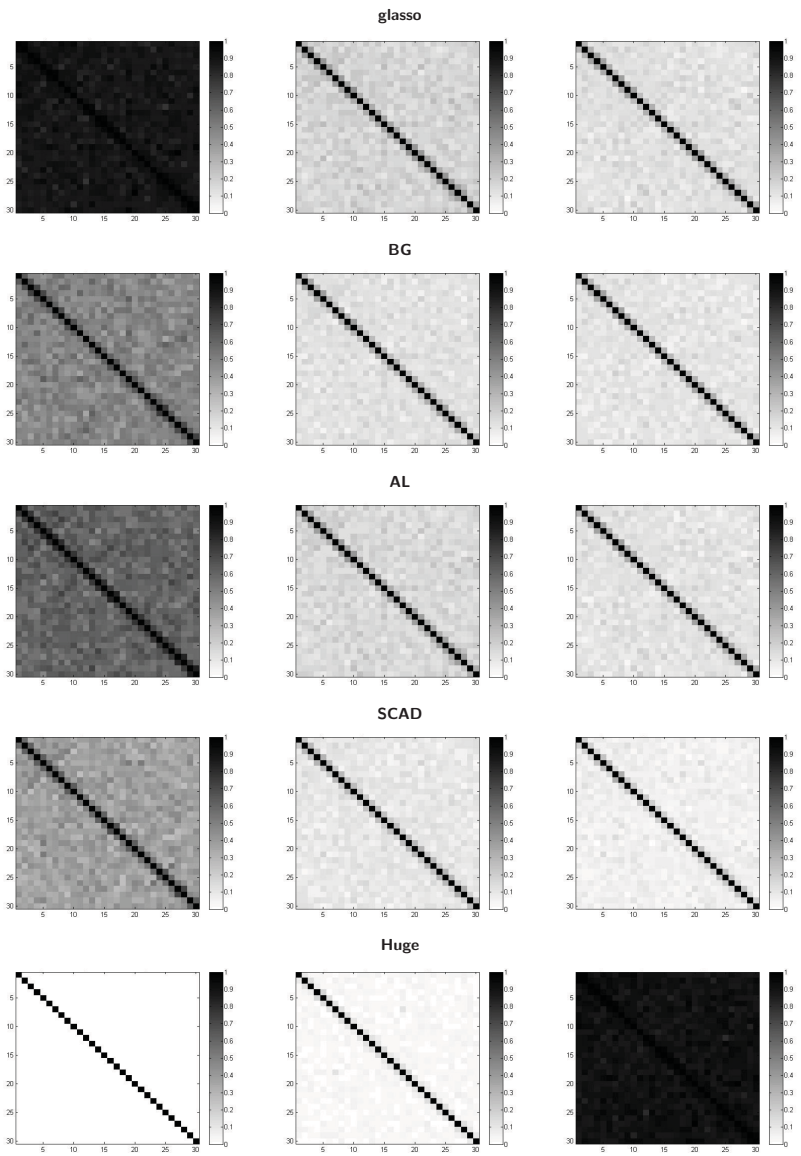

Figure 20: *The ASP plots for the* $\boldsymbol{MVt_4}$ ***data*** *(*$p = 30$*) generated from the exponential decay matrix, using the nonparanormal transformation function* `huge.npn()` *and the wrong AIC/BIC formula*

[Left panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the wrong AIC formula, based on the transformed data. [Middle Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the wrong BIC formula, based on the transformed data. [Right Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the CV criterion, based on the transformed data.

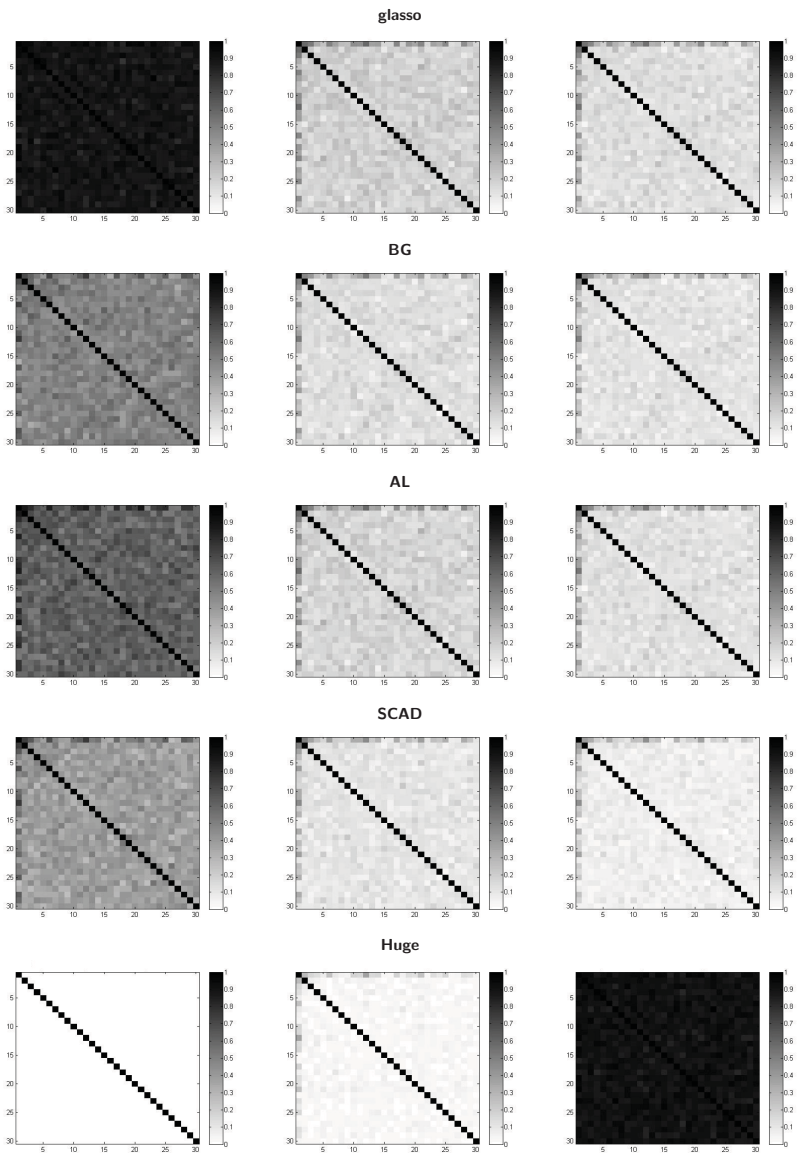

Figure 21: *The ASP plots for the* $\boldsymbol{MVt_4}$ ***data*** *($p = 30$) generated from the general matrix, using the nonparanormal transformation function* `huge.npn()` *and the wrong AIC/BIC formula*

[Left panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the wrong AIC formula, based on the transformed data. [Middle Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the wrong BIC formula, based on the transformed data. [Right Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the CV criterion, based on the transformed data.

Figures 18, 20 and 21 display the ASP plots after applying the nonparanormal transformation function to the $\boldsymbol{MVt_4}$ ***data*** with dimensions 30.

After transforming the original ***MVt data,*** all the TPs are smaller and all the TNs become larger for all the AIC/BIC combinations with respect to all three precision matrices, which indicates all the estimated graphs selected by AIC or BIC become sparser than before. This tells us that if the original dataset is not normally distributed, the function `huge.npn()` can effectively help to provide sparser estimates without losing the actual graphical structure if we use the combinations of AIC/BIC. Moreover, the estimates based on the transformed dataset are still denser (no better) than the estimates based on the original dataset selected by the corrected AIC/BIC formula, which means the corrected formula is the most powerful way to increase the sparsity levels of the estimates, if one accepts slight over-shrinkage.

However, the CV combinations behave differently from the AIC/BIC combinations. Almost all the TPs increase for all the CV combinations with respect to all the three precision matrices. However, all the TNs of the CV combinations become smaller for the tridiagonal matrix with $a = 1.7$. For the exponential decay matrix and the general matrix, all the TNs of the CV combinations with glasso, AL and SCAD either become slightly larger or remain unchanged, while the TNs of BG$\star$CV become smaller. Thus most of the estimated graphs selected by CV become slightly denser after applying the transformation function.

All the TPs of Huge$\star$ric and Huge$\star$stars become larger and their TNs are smaller for all three true precision matrices. More specifically, the TPs of Huge$\star$ric increase remarkably, especially for the tridiagonal matrix, and its TNs decrease negligibly. Thus the estimated graphs uisng Huge$\star$ric contain more correct estimated non-zero entries, with only a very small loss of detections of correct zero entries. Although Huge$\star$stars has the same results as Huge$\star$ric to the function `huge.npn()`, its results change little, remaining excessively dense. Also, the Huge$\star$ebic results seem to be insensitive to the transformation function, resulting in excessively sparse graphs. This may reflect the superiority of Huge$\star$ric over Huge$\star$ebic and Huge$\star$stars. Even if the original dataset is not normally distributed, Huge$\star$ric can still be effective to estimate the graph if the function `huge.npn()` is applied, especially if the matrix has

moderate entry values (e.g. larger than 0.2).

The effects of the transformation function `huge.npn()` on the ***MV*t_4*-data*** generated from the tridiagonal matrix with $a = 0.85$ are basically the same as the $a = 1.7$ tridiagonal matrix case, where the sparsity of the AIC and BIC estimates is greatly increased, making the BG combinations providing the sparest results, the AL and SCAD combinations with similar sparsity and the glasso results still being the densest among all the results based on the transformed data.

An interesting thing happened to the tridiagonal matrix with $a = 0.85$ using thetransformation function `huge.npn()` on the ***MV*t_4*-data*** is that `huge.npn()` cured the absence of retuen value of the Huge$\star$ric combination, which means the combination is able to provide estimates for all repetitions if the function `huge.npn()` is applied to the data. This may tell us that `huge.npn()` can not only help to improve the sparsity of the estimates of some combinations, but also help to ease the problem due to the closeness of the true precision matrix to be not positive definite. In addition, the zero entry estimates of the Huge$\star$stars and Huge$\star$ebic become slightly less and slightly more than the results before applying the `huge.npn()` function.

Table 19: *The TPs and TNs for the tridiagonal matrix with* $a = 1.7$ *for the* ***MV****t*$_4$ ***data*** *(*$p = 30$*) using the nonparanormal transformation function* `huge.npn()`

Ω	**tri** $a = 1.7$					
	TP			**TN**		
	AIC	**BIC**	**CV**	**AIC**	**BIC**	**CV**
CM	1			0		
PCM	1			0		
glasso	0.991	0.939	0.9	0.112	0.594	0.75
AL	0.959	0.923	0.881	0.42	0.681	0.806
SCAD	0.914	0.866	0.818	0.655	0.827	0.893
BG						
$H = 50$	0.916	0.884	0.862	0.595	0.769	0.831
Huge						
ric	0.651			0.968		
stars	0.991			0.107		
ebic	0			1		

Table 20: *The TPs and TNs for the exponential decay matrix and the general matrix for the* ***MV***t_4 ***data*** *(*$p = 30$*) using the nonparanormal transformation function* `huge.npn()`

Ω	exp						gen					
	TP			TN			TP			TN		
	AIC	BIC	CV	AIC	BIC	CV	AIC	BIC	CV	AIC	BIC	CV
CM	**1**			**0**			**1**			**0**		
PCM	**1**			**0**			**1**			**0**		
glasso	**0.901**	**0.244**	**0.195**	**0.115**	**0.825**	**0.875**	**0.894**	**0.217**	**0.166**	**0.089**	**0.816**	**0.879**
AL	**0.663**	**0.22**	**0.185**	**0.393**	**0.841**	**0.885**	**0.631**	**0.201**	**0.155**	**0.368**	**0.831**	**0.897**
SCAD	**0.455**	**0.155**	**0.112**	**0.624**	**0.903**	**0.938**	**0.409**	**0.136**	**0.099**	**0.611**	**0.901**	**0.939**
BG												
$H = 50$	**0.541**	**0.187**	**0.173**	**0.519**	**0.879**	**0.893**	**0.505**	**0.163**	**0.144**	**0.507**	**0.866**	**0.901**
Huge												
ric	**0.057**			**0.973**			**0.058**			**0.976**		
stars	**0.921**			**0.093**			**0.909**			**0.077**		
ebic	**0**			**1**			**0.022**			**1**		

Table 19 and Table 20 show the TPs and the TNs of the high dimensional t_4-***data*** which applied the non-paranormal transformation function. The table of the tridiagonal matrix with $a = 0.85$ is displayed in the appendix. They also show consistent conclusions to the ASP plots.

5.3 AR1 Auto-correlated *N-Data* (*MVN AR1 data*)

5.3.1 Low dimensional cases

Table 21: *The TPs for the **MVN AR1 data** ($p = 5$) at 4 different sample sizes*

	$n = 100$			$n = 200$			$n = 500$			$n = 1000$		
	TP			**TP**			**TP**			**TP**		
	AIC	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**
CM	1			1			1			1		
PCM	1			1			1			1		
glasso	1	1	0.99	1	1	1	1	1	1	1	1	1
AL	0.99	0.988	0.973	0.998	0.998	0.998	1	1	1	1	1	1
SCAD	0.983	0.963	0.865	0.998	0.995	0.94	1	0.998	0.998	1	0.998	0.996
BG												
$H = 50$	0.988	0.97	0.978	0.998	0.998	0.998	1	1	1	1	1	1
Huge												
ric	0.393			0.419			0.648			0.938		
stars	1			1			1			1		
ebic	1			1			1			1		

Table 21 contains the TPs of the low dimension ***MVN data*** with an AR1 auto-correlation structure added to each of its variables (***MVN AR1 data*** for short hereafter), at four different sample sizes from 100 to 1000. As expected, there are indeed decreases in the TPs for some combinations at some sample sizes. The decreases mainly happen to the AL combinations at $n = 100, 200$, SCAD combinations (especially selected by CV) and Huge$\star$ric at all four sample sizes. However, when n increases, the decreases of TPs become smaller, or increasing the sample sizes will lessen the losses of power to detect the non-zeros because of the auto-correlation within each variable.

Table 22: *The TNs for the **MVN AR1 data** ($p = 5$) at 4 different sample sizes*

	$n = 100$			$n = 200$			$n = 500$			$n = 1000$		
	TN			**TN**			**TN**			**TN**		
	AIC	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**
CM	0			0			0			0		
PCM	0			0			0			0		
glasso	0.083	0.14	0.387	0.067	0.125	0.367	0.083	0.17	0.313	0.074	0.173	0.263
AL	0.192	0.288	0.675	0.202	0.333	0.707	0.247	0.388	0.677	0.275	0.427	0.683
SCAD	0.358	0.518	0.858	0.386	0.593	0.902	0.532	0.668	0.846	0.777	0.845	0.885
BG												
$H = 50$	0.314	0.473	0.528	0.335	0.518	0.501	0.369	0.544	0.418	0.339	0.556	0.382
Huge												
ric	0.905			0.883			0.83			0.568		
stars	0.145			0.17			0.238			0.288		
ebic	0.204			0.228			0.243			0.288		

Table 22 consists of the TNs for the low dimension ***MVN AR1 data***, at four sample sizes.

All the AIC/BIC combinations have smaller TNs at all four sample sizes than the ***MVN data*** and a number of them are reduced to around half of the original values. Also, generally, the TNs of the ***MVN AR1 data*** increase with the sample sizes for all the AIC/BIC combinations. More specifically, the SCAD method seems to perform the best, or it is the most resistant method to the AR1 auto-correlation, at all sample sizes. Since AL provides pretty desirable estimates for the original ***MVN data*** with relatively high level TNs, though its reduced TNs are still higher than glasso's, they suffer from the most severe declines in TNs, especially for AIC. Moreover, both AL and BG provide similarly good estimates for the original ***MVN data*** when n is large, however, for the ***MVN AR1 data***, BG provides much better results than AL. Thus for now, SCAD still performs the best, followed by BG, AL and glasso. In terms of the selection criteria, BIC selects better results (more

correctly estimated zeros) than AIC.

With regard to CV, glasso⋆CV at all sample sizes, AL⋆CV at $n = 200$, SCAD⋆CV at $n = 100, 200$ have higher TNs for the ***MVN AR1 data*** than the ***MVN data***. This indicates that if the samples of each variable are not independently collected, which violates the independency assumption for the estimating methods, CV combinations detect more zero entries at times than the normally distributed dataset with independent samples. However, interestingly, BG⋆CV has lower TNs at all sample sizes, which may indicate that the resampling procedure can neutralize the advantage of CV.

Huge⋆ric has larger TNs at all sample sizes with a decreasing trend (the TNs decrease with the sample sizes), while the TNs of Huge⋆ebic and Huge⋆stars become smaller at all four sample sizes with an increasing trend. Huge⋆ric correctly estimates much more zero entries than Huge⋆ebic and Huge⋆stars at all sample sizes. Moreover, the TNs of Huge⋆ebic and Huge⋆stars become exactly the same again, just as they do for the low dimensional ***MVN data*** case.

5.3.2 High dimensional case

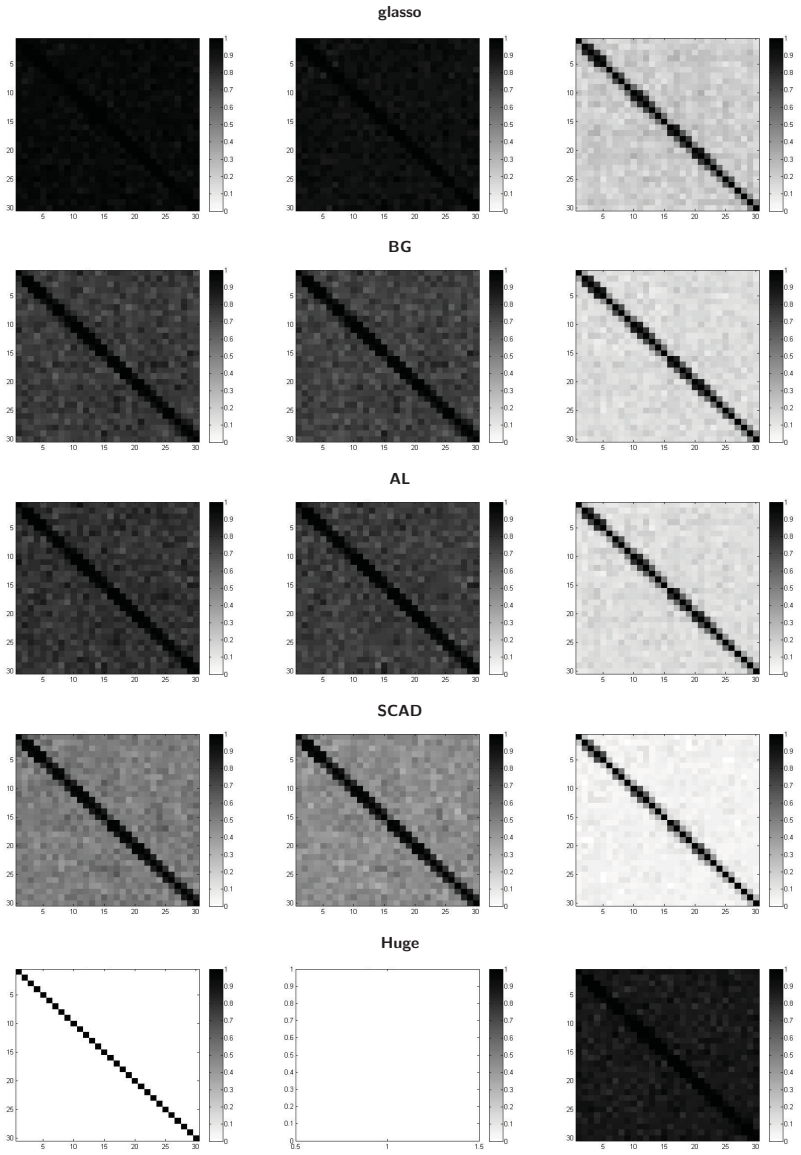

Figure 22: *The ASP plots for the **MVN AR1 data** ($p = 30$) generated from the tridiagonal matrix with $a = 1.7$*

[Left panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the AIC criterion. [Middle Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the BIC criterion. [Right Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the CV criterion.

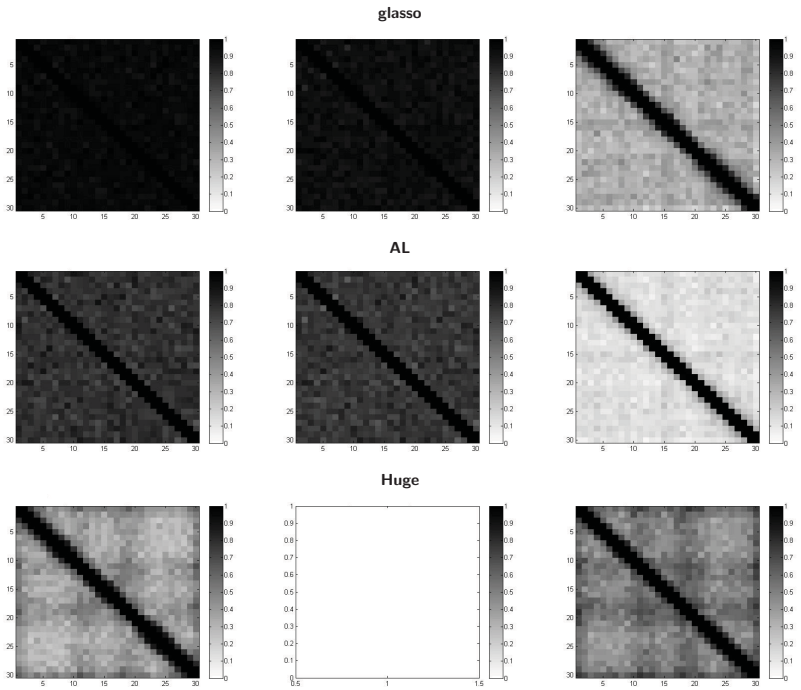

Figure 23: *The ASP plots for the tridiagonal matrix with* $a = 0.85$ *for the* ***MVN AR1 data*** *(*$p = 30$*)*

[Left panel] The ASP plots of the glasso, AL and Huge estimates selected by the AIC criterion.

[Middle Panel] The ASP plots of the glasso, AL and Huge estimates selected by the BIC criterion.

[Right Panel] The ASP plots of the glasso, AL and Huge estimates selected by the CV criterion.

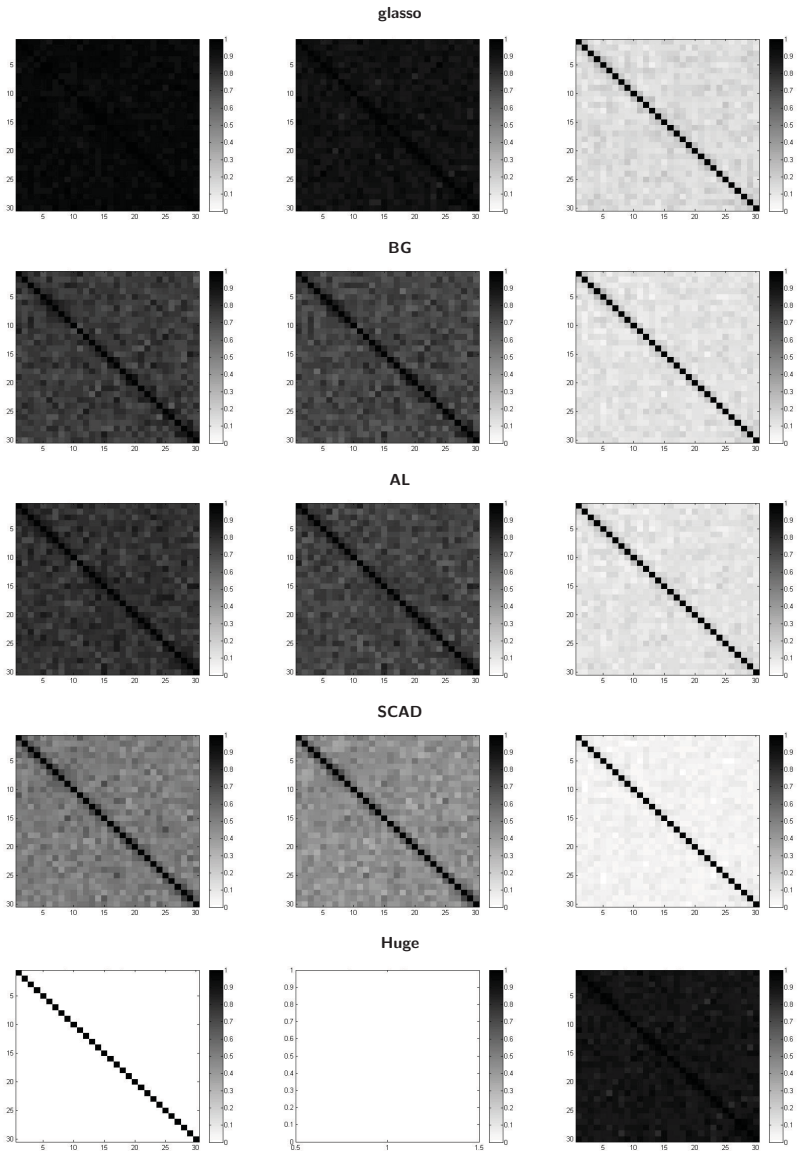

Figure 24: *The ASP plots for the **MVN AR1 data** ($p = 30$) generated from the exponential decay matrix*

[Left panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the AIC criterion. [Middle Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the BIC criterion. [Right Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the CV criterion.

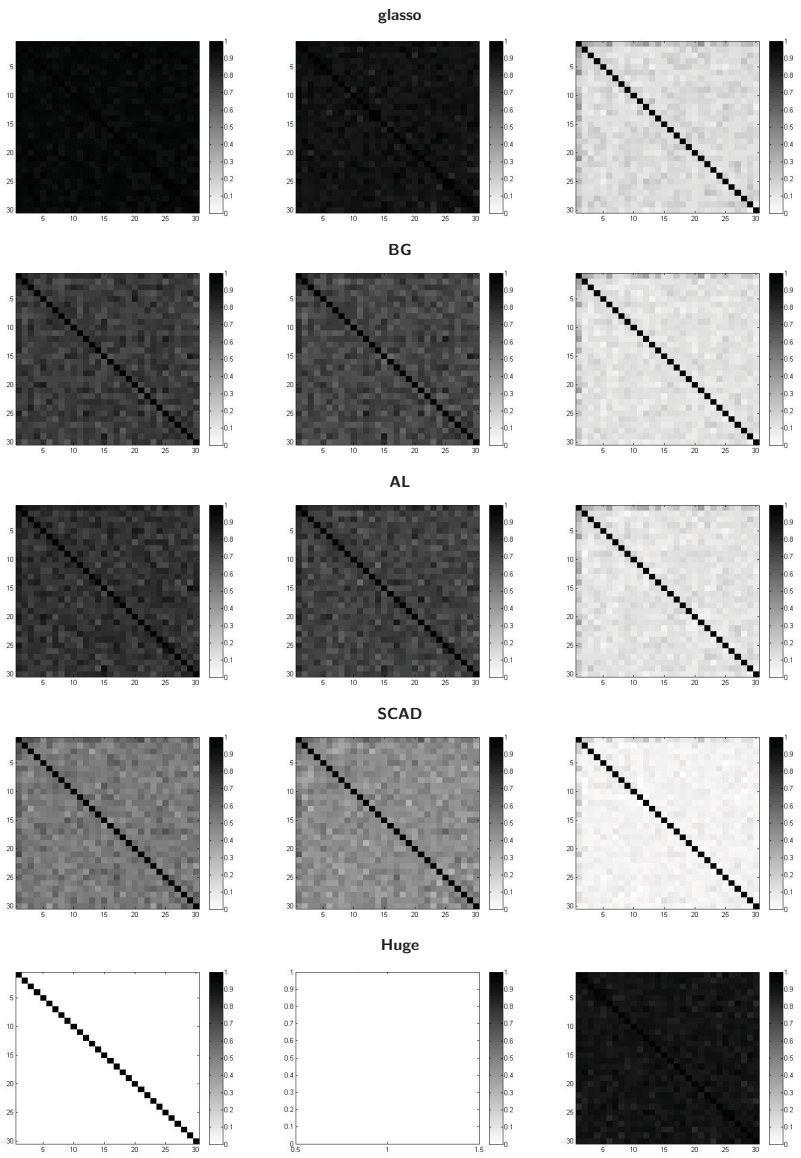

Figure 25: *The ASP plots for the **MVN AR1 data** ($p = 30$) generated from the general matrix*

[Left panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the AIC criterion. [Middle Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the BIC criterion. [Right Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the CV criterion.

Figures 22, 24 and 25 are ASPs corresponding to the high dimensional ***MVN AR1 data*** with respect to the tridiagonal matrix with $a = 1.7$, the exponential de-

cay matrix and the general matrix, respectively. Figure 23 contains all the ASP plots we can get from the tridiagonal matrix with $a = 0.85$, where the rest of the combinations get stuck halfway.

For the tridiagonal matrix with $a = 1.7$, the exponential decay matrix and the general matrix, all the estimates of AIC and BIC combinations become significantly denser based on the ***MVN AR1 data***. BIC can provide both accurate and sparse estimates for the high dimensional ***MVN data***, however, when the samples are not independently collected, BIC results deteriorate dramatically, with matrices almost as dense as the AIC matrix and it also loses some of the true graphical structure, even for the tridiagonal matrix which contains many large entries. Among all AIC/BIC combinations, SCAD combinations are the only ones that do not completely lose the actual model structure. However, both of their abilities of detecting non-zeros and zeros are spoiled. Only the basic structure can be captured by the SCAD combinations. Also, the matrices with large entries are easier to estimate, such as the tridiagonal matrix, while the general matrix structure is almost lost, even if it is estimated by SCAD$\star$AIC/BIC.

However, almost all of the TPs of CV combinations go down and all of the TNs increase for all the three precision matrices, leading to sparser estimates for the ***MVN AR1 data*** when compared to the ***MVN data***. More specifically, for the tridiagonal matrix with $a = 1.7$, the ability of CV to detect non-zeros declines the most. It not only declines more than the AIC/BIC, but also declines the most among all three precision matrices, which indicates that when the dataset is not independently collected, CV is more inclined to over shrink the entries, even for relatively large entries, or CV loses some power of detecting non-zeros. This may confirm again that CV is more resistant to misuses, i.e. CV results are likely to be sparser. Moreover, for all the CV combinations, SCAD$\star$CV is still the sparsest one.

For Huge estimates on the ***MVN AR1 data***, Huge$\star$ebic results are still always excessively sparse and Huge$\star$stars results remain excessively dense. However, for Huge$\star$ric, an interesting observation is that there is no returned values from the package `Huge` in `R` for all the 100 repetitions, which means that we cannot get the estimated precision matrices from Huge$\star$ric by the package `Huge`. This

never happened in other scenarios, even for the high dimensional ***MVt data***. This may indicate that the auto-correlation is more devastating to the results than the t-distribution, at least for Huge$\star$ric.

For the ***MVN AR1 data*** generated from the tridiagonal matrix with $a = 0.85$, it is noticed that the BG and SCAD methods can not finish all the repetions, while they were able to provide estimates for the ***MVN data*** generated from this precision matrix. Thus the AR(1) auto correlation structure will also cause estimation problems for some of the combinations, making them inapplicable. Also, again, the Huge$\star$ric combination can not return estimates for the ***MVN AR1 data*** generated from the tridiagonal matrix with $a = 0.85$.

Among all the results we can get for the ***MVN AR1 data*** generated from the tridiagonal matrix with $a = 0.85$, AL$\star$CV provides the best estimates and glasso$\star$CV can be regarded as the second best choice. Though the Huge$\star$ebic and Huge$\star$stars results are slightly denser than the best two combinations, they are still able to capture the true graphical structure. However, the estimates of glasso$\star$AIC, glasso$\star$BIC, AL$\star$AIC and AL$\star$BIC are so dense that the true graphical structure is completely lost.

Compared to the results of the ***MVN data*** generated from the tridiagonal matrix with $a = 0.85$, the results of the glasso$\star$AIC, glasso$\star$BIC, AL$\star$AIC and AL$\star$BIC combinations are significantly denser, which lose the true graphical structure. On the contrary, the glasso$\star$CV and AL$\star$CV results become slightly sparser than the ***MVN data*** results, which is the same as the other three precision matrices. In addition, the Huge$\star$ebic and Huge$\star$stars combinations produce slightly denser estimates for the ***MVN AR1 data*** than the ***MVN data***.

Table 23: *The TPs and TNs for the tridiagonal matrix with* $a = 1.7$ *for the* ***MVN AR1 data*** *(*$p = 30$*)*

Ω	**tri** $a = 1.7$					
	TP			**TN**		
	AIC	BIC	CV	AIC	BIC	CV
CM	***1***			***0***		
PCM	***1***			***0***		
glasso	0.991	0.982	0.62	0.049	0.078	0.792
AL	0.951	0.944	0.552	0.217	0.253	0.857
SCAD	0.87	0.844	0.397	0.483	0.545	0.945
BG						
$H = 50$	0.943	0.936	0.6	0.247	0.269	0.849
Huge						
ric	NA			NA		
stars	0.976			0.123		
ebic	0			1		

Table 24: *The TPs and TNs for the exponential decay matrix and the general matrix for the **MVN AR1 data** ($p = 30$)*

Ω	**exp**						**gen**					
	TP			**TN**			**TP**			**TN**		
	AIC	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**	**AIC**	**BIC**	**CV**
CM	1			*0*			1			*0*		
PCM	1			*0*			*1*			*0*		
glasso	0.956	0.919	0.175	0.049	0.093	0.854	0.953	0.904	0.173	0.052	0.102	0.853
AL	0.808	0.754	0.143	0.218	0.277	0.886	0.789	0.738	0.142	0.228	0.286	0.878
SCAD	0.566	0.491	0.491	0.481	0.556	0.947	0.542	0.468	0.077	0.49	0.569	0.943
BG												
$H = 50$	0.777	0.748	0.147	0.249	0.285	0.881	0.755	0.72	0.145	0.253	0.294	0.88
Huge												
ric	NA			NA			NA			NA		
stars	0.903			0.108			0.897			0.115		
ebic	0			1			0.022			1		

Table 23 and Table 24 contain the detailed TPs and TNs for the high dimensional ***MVN AR1 data***, they show identical conclusions to their corresponding ASP plots. The TPs and TNs of the tridiagonal matrix with $a = 0.85$ can be found in the appendix.

Chapter 6

Conclusions

Using the same notations as in the Results section, the combination of an estimating method M_q with the selection criterion C_c is denoted as $M_q \star C_c$. For example, if one uses the glasso algorithm to estimate the precision matrix, then apply AIC to select the best estimate, this combination is denoted as glasso$\star$AIC.

6.1 Computational speed comparisons

In this subsection we illustrate the computational speed of each estimation combination, as the cost of time is also an important issue that should be considered for practical reasons. When recording the computational time for each combination $M_q \star C_c$, we fix the repetition time N to be 100 and the sample size $n = 100$ for all cases, then apply 100 regularization parameters ρs to each combination with $\rho = 0.01 \times i$ for $i = 1, \ldots, 100$. For the BG and DP-BG methods, we fix the resampling number H to be 50 and the threshold to be $\pi_{thr} = 0.9$. For the low dimensional cases ($p = 5$) the computational time for each combination is shown in Table 25 (in seconds):

Table 25: *The computational time of each combination for the* ***MVN data*** *($p = 5$) and the* ***MV*** t_3 ***data*** *($p = 5$) using an Intel(R) Core(TM) i3-2350 M 2.30GHz CPU*

Methods	***MVN data***	***MVt* Data**	**Methods**	***MVN data***	***MVt* Data**
glasso⋆AIC	3.44	3.61	**DP-G⋆AIC**	76.8	119.51
glasso⋆BIC	3.44	3.54	**DP-G⋆BIC**	76.66	119.53
glasso⋆CV	37.48	38.47	**DP-G⋆CV**	394.09	616.71
BG⋆AIC	170.31	117.87	**DP-BG⋆AIC**	3973.94	6440.35
BG⋆BIC	171.83	177.70	**DP-BG⋆BIC**	3971.14	6430.92
BG⋆CV	2021.85	2054.96	**DP-BG⋆CV**	24129.8	38605.94
AL⋆AIC	8.65	7.22	**DP-AL⋆AIC**	80.84	121.51
AL⋆BIC	9.17	9.00	**DP-AL⋆BIC**	80.64	121.55
AL⋆CV	80.35	98.30	**DP-AL⋆CV**	431.15	644.83
SCAD⋆AIC	15.82	16.77	**DP-SCAD⋆AIC**	82.34	122.12
SCAD⋆BIC	14.57	16.19	**DP-SCAD⋆BIC**	82.27	123.56
SCAD⋆CV	107.16	86.44	**DP-SCAD⋆CV**	432.75	644.38
Huge⋆ric	115.64	116.27	**Huge⋆ebic**	87.77	90.50
Huge⋆stars	1981.21	2021.82			

Each of the times in Table 25 corresponds to the time cost to finish $N = 100$ repetitions of the whole estimating and selection procedure for each combination $M_q \star C_c$. More specifically, the whole procedure is the time to estimate the true precision matrix Ω 100 times (based on $N = 100$ different datasets) by each Method M_q and then select the best estimate by its Criterion C_c among all the 100 estimated precision matrices produced by the 100 regularization parameters.

6.2 The choice of π_{thr} and H in the BG algorithm

Next we study the optimal choice for the key parameters for BG, the number of resampling iterations H and the threshold π_{thr}. Then, we chose the values that provide desirable estimates and are computational friendly as well.

Table 26: *The TNs for the **MVN data** ($p = 5$ and $n = 200$) using glasso and BG for $H = 50, 100, 150, 200$, $\pi_{thr} = 0.75$ and selection criteria AIC, BIC and CV*

			AIC		**BIC**		**CV**	
π_{thr}	n	H	**glasso**	**BG**	**glasso**	**BG**	**glasso**	**BG**
0.75	200	50	0.143	0.145	0.295	0.465	0.242	0.242
		100	0.143	0.145	0.295	0.473	0.242	0.242
		150	0.143	0.143	0.295	0.442	0.242	0.242
		200	0.143	0.143	0.295	0.448	0.242	0.242

Table 27: *The TNs for the **MVN data** ($p = 5$ and $n = 200$) using glasso and BG for $H = 50, 100, 150, 200$, $\pi_{thr} = 0.8$ and selection criteria AIC, BIC and CV*

			AIC		**BIC**		**CV**	
π_{thr}	n	H	**glasso**	**BG**	**glasso**	**BG**	**glasso**	**BG**
0.8	200	50	0.143	0.203	0.295	0.61	0.242	0.252
		100	0.143	0.162	0.295	0.605	0.242	0.242
		150	0.143	0.158	0.295	0.618	0.242	0.242
		200	0.143	0.15	0.295	0.619	0.242	0.242

Table 28: *The TNs for the **MVN data** ($p = 5$ and $n = 200$) using glasso and BG for $H = 50, 100, 150, 200$, $\pi_{thr} = 0.85$ and selection criteria AIC, BIC and CV*

			AIC		**BIC**		**CV**	
π_{thr}	n	H	**glasso**	**BG**	**glasso**	**BG**	**glasso**	**BG**
0.85	200	50	0.143	0.3	0.295	0.714	0.242	0.305
		100	0.143	0.319	0.295	0.743	0.242	0.288
		150	0.143	0.272	0.295	0.735	0.242	0.265
		200	0.143	0.275	0.295	0.748	0.242	0.267

Table 29: *The TNs for the **MVN data** ($p = 5$ and $n = 200$) using glasso and BG for $H = 50, 100, 150, 200$, $\pi_{thr} = 0.9$ and selection criteria AIC, BIC and CV*

			AIC		**BIC**		**CV**	
π_{thr}	n	H	**glasso**	**BG**	**glasso**	**BG**	**glasso**	**BG**
0.9	200	50	0.143	0.595	0.295	0.836	0.242	0.496
		100	0.143	0.591	0.295	0.855	0.242	0.486
		150	0.143	0.595	0.295	0.848	0.242	0.465
		200	0.143	0.599	0.295	0.852	0.242	0.463

Table 30: *The TNs for the **MVN data** ($p = 5$ and $n = 200$) using glasso and BG for $H = 50, 100, 150, 200$, $\pi_{thr} = 0.95$ and selection criteria AIC, BIC and CV*

			AIC		**BIC**		**CV**	
π_{thr}	n	H	**glasso**	**BG**	**glasso**	**BG**	**glasso**	**BG**
0.95	200	50	0.143	0.777	0.295	0.907	0.242	0.738
		100	0.143	0.852	0.295	0.932	0.242	0.794
		150	0.143	0.814	0.295	0.932	0.242	0.76
		200	0.143	0.837	0.295	0.935	0.242	0.781

Since all the TPs of our 5-dimensional datasets are 1, we only show the TNs here to see the variations of the glasso and BG results. Tables 26, 27, 28, 29 and 30 illustrate the TNs of glasso and BG combinations (with AIC, BIC and CV), with the 5 π_{thr}s being $0.75, 0.8, 0.85, 0.9, 0.95$. For each value of π_{thr}, there are 4 different resampling iterations $H = 50, 100, 150, 200$ corresponding to a fixed sample size $n = 200$.

6.2.1 How the resampling iterations H effects BG

From the above tables we can see that for the 5 dimensional dataset with the sample size $n = 200$, by increasing the number of resampling iterations from 50 to 200, we did not improve the estimates of BG. Moreover, resampling too many times can also harm the results. We found that the largest H should not be greater than

the sample size n. However, a larger H will lead to noticeably more computational time, especially for the high dimensional datasets. In conclusion, increasing H can slightly boost the precision of the estimate, but also consumes significantly more time.

Since we fix $n = 100$ in our high dimensional datasets and $n = 100, 200, 500, 1000$ in our low dimensional datasets, for simplicity, we fix $H = 50$ in all of our simulations of BG. This H is at most half of the sample sizes but we believe it can effectively improve the estimates without requiring too much computational time. However, $H = 50$ may be not large enough for those datasets with large sample sizes. Indeed, for the datasets with a large number of samples, we believe that increasing H can produce better estimates for BG, but we do not think it is worth obtaining those small improvements by sacrificing the computational time.

6.2.2 How the threshold value π_{thr} effects BG

From the above tables it is evident that π_{thr} is much more influential on the BG estimates than the choice of H. Only slightly increasing the value of π_{thr} leads to a major improvement. However, we believe that if one chooses an excessively large π_{thr}, the non-zero estimate frequencies will have a hard time becoming larger than the threshold. Thus many estimated entries will be set to be zeros and the estimated graphs will be very sparse with many false estimated zeros.

Another appealing feature of π_{thr} is that no additional computational time is required. As a consequence, we choose $\pi_{thr} = 0.9$ in all our BG simulations, which is believed to effectively improve the estimates without the risk of providing inordinately sparse results.

6.3 Estimation Accuracy Comparisons

6.3.1 Preliminaries

We discuss in this section the precision of the estimates for all the datasets considered in Chapter 4, including the ***MVN data***, the ***MVt data*** and the ***MVN AR1***

data. We indicate their closeness to the truth and also their desirable sparsities. It should be noted that the accuracy and the sparsity of an estimate are not equivalent to each other. Sometimes an estimate can be very sparse but it is unnecessarily sparse in that it can not effectively reveal the true graphical structure, which is regarded as undesirable. On the other hand, an estimate may be denser than another one, but it can be closer to the true graph, which we believe can be regarded as the better estimate. The reason that the sparsity and the accuracy are not equivalent to each other is that non-zero entries are easy to correctly estimate most of the time, however, in many cases zero entries are mistakenly estimated. Thus, provided that almost all non-zeros are estimated correctly, a sparser estimated graph indicates more zero entries are estimated correctly, which will be more accurate and closer to the true graph. Additionally, the results of an estimating method/selection criterion in this section represent the best estimated precision matrices (or the estimated undirected graphs) that this method/selection criterion can reveal, under the restriction that certain reasonable parameters were chosen, such as the 100 ρs, H and π_{thr}, etc. Accordingly, one method M_q is better than another M_l if the best estimates M_q provides are better than the best estimates M_l can provide. The same representations are used for selection criteria as well.

As our results are based on the parameters we choose, it is also possible that the best results of a method are further improved if some parameters are properly changed. However, the results may become worse with some undesirable settings, such as choosing a narrow range of ρs or only one fixed ρ, using a too small π_{thr}, and using too little or too many resampling iterations.

6.3.2 Overall Conclusions

In general, for all dimensions and types of datasets, we have the following conclusions:

1. SCAD $>$ AL $>$ BG $>$ glasso $>$ Huge.

2. The DP-glasso algorithm is not effective in improving the estimates. In other words, DP-glasso has very similar results to glasso, and so does DP-BG to

BG, DP-AL to AL and DP-SCAD to SCAD.

3. BIC always chooses the sparser estimates than AIC. CV is sparser than AIC most of the time. All of them are able to keep the true graphical structure, and the only differences occur in the sparsities of the estimated graphs.

4. Typically, SCAD⋆BIC provides the best estimates for all the cases. AL⋆BIC or BG⋆BIC or AL⋆CV are the second best choices most of the time. glasso⋆AIC and Huge always provide undesirable estimates.

5. Some of the combinations can produce negative estimates on the main diagonal of an estimated precision matrix occasionally, which is unallowed. Thus we record the frequencies for the simulation scenarios where this kind of mistake happens. The frequency tables can be found in Appendix .

6.3.3 Conclusions for the multivariate normal data

For the ***MVN data***, in both low dimensional and high dimensional cases, when n is of moderate size, SCAD⋆BIC or SCAD⋆CV performs the best, when n is large, we recommend using SCAD⋆BIC, SCAD⋆CV, AL⋆BIC, AL⋆CV and BG⋆BIC. For the low dimensional ***MVN data***, the least recommended methods are the glasso and Huge combinations. For the high dimensional cases, we also suggest not to use glasso⋆AIC, Huge⋆ebic and Huge⋆stars.

By increasing the dimensions of the ***MVN data***, we do not see a deterioration in the estimation of glasso, AL, BG or SCAD. On the contrary, with increasing dimensions we find remarkably better detections of the zero entries by glasso⋆BIC, glasso⋆CV, along with the moderate increases by AL⋆BIC, AL⋆CV and BG⋆CV.

However, Huge does not perform as well as glasso, AL, BG and SCAD as the dimension increases. Huge⋆ric is recommended only if the true precision matrix contains many large value non-zeros. Otherwise, all the three Huge combinations should not be adopted, due to their incapability of maintaining the true graphical structures.

If the dataset is normally distributed, through properly chosen combinations, such as SCAD⋆BIC, SCAD⋆CV or BG⋆BIC, accurate estimates can be achieved. For example, more than 90% of all the entries of the true precision matrix can be succesfully detected.

Note that glasso, AL, BG and SCAD have a stronger ability of detecting smaller non-zero entries than Huge. In other words, the estimation precisions of glasso, AL, BG and SCAD are better than Huge. Thus, if the connections within a model are not very intense, glasso, AL, BG and SCAD are better choices than Huge, with SCAD being the strongest recommendation.

Moreover, the tridiagonal matrix with large entries are more likely to get worse estimates for most of the glasso, AL, BG and SCAD combinations, with only AL⋆CV, SCAD⋆BIC, SCAD⋆CV and BG⋆AIC results being better. The tridiagonal matrix with large entries (close to be not positive definite) also causes estimation problems for the Huge⋆ric combination, which can not provide any estimate for any of the repetition. However, the other two Huge combinations provide much better results than the tridiagonal matrix with smaller entries.

6.3.4 Conclusions for the t-distributed data

The major trends (see Subsection 6.3.2) of the estimating methods and selection criteria will not be changed when the distributions of the datasets are altered. In general, SCAD is the most resistant estimating method to the abuses of the distributions of the datasets. CV is more stable against the changes of the distributions of the datasets than AIC and BIC.

Changing the distribution of the datasets from the normal distribution to the t-distribution will definitely harm all the AIC/BIC combinations results and brings about much denser estimates. In particular, the AIC results suffer from devastating decreases of sparsities in which only the SCAD⋆AIC combination does not completely lose the graphical structure. In contrast, some of the CV combinations produce better estimates for the ***MVt data*** than the ***MVN data***. In particular, SCAD⋆CV provides the best estimates if the AIC and BIC use the wrong log-

likelihood in their formulas. In the low dimensional case, Huge⋆ric detects more zero entries for the ***MVt data*** than the ***MVN data***, while Huge⋆stars and Huge⋆ebic detect slightly less zeros for the ***MVt data*** than the ***MVN data***. All of these three combinations maintain the true graphical structure. When the dimension of the ***MVt data*** increases, Huge is very likely to provide very poor estimates and lose the original graphical structures.

Using the correct log-likelihood in the AIC/BIC formula, SCAD is the only method which dramatically increases its ability of estimating zero entries using both the AIC and BIC in all sample sizes for the low dimensional ***MVt data***. In other words, the correct AIC/BIC formula show either small improvements or perform slightly worse using the other methods. However, when the dimension of the ***MVt data*** increases, the effects of the correct AIC/BIC formula are magnified. The sparsities of all the AIC/BIC results using all methods are dramatically increased, resulting in significantly sparser estimated graphs, which enables AIC results to be at the same sparsity levels as the CV results, and the BIC results are the sparsest. However, BIC leads to somewhat excessively sparse estimates so that the true structures are lost, especially when the true entries are relatively small.

The effect of df on the estimates are concluded as follows. Generally, the results of ***MV***t_3 ***data*** and ***MV***t_4 ***data*** are very similar, in both low and high dimensional cases. However, higher dimensional datasets lead to slightly sparser estimates using the AIC/BIC combinations before correcting the AIC/BIC formula. For CV, if the true entries are not very small, the estimates become slightly denser for the ***MV***t_4 ***data***. After applying the corrected AIC/BIC formula, if the true entries are relatively large in the high dimensional case, AIC leads to denser estimates for the ***MV***t_4 ***data*** than the ***MV***t_3 ***data,*** while BIC produces sparser estimates for ***MV***t_4 ***data*** than the ***MV***t_3 ***data***. If the true entries are relatively small, glasso and AL combinations (with both AIC and BIC) produce sparser estimates for the ***MV***t_4 ***data*** than the ***MV***t_3 ***data***. SCAD⋆AIC will produce denser estimates for the ***MV***t_4 ***data*** than the ***MV***t_3 ***data*** and SCAD⋆BIC will produce sparser estimates for the ***MV***t_4 ***data*** than the ***MV***t_3 ***data***. The effect of df on BG estimates is ambiguous. Moreover, Huge⋆ric and Huge⋆stars produce denser estimates for the ***MV***t_4 ***data*** than the ***MV***t_3 ***data,***

while Huge⋆ebic results remain almost the same.

In addition, we also studied the effect of the nonparanormal transformation function huge.npn() on the ***MV***t_4 ***data***. In the low dimensional case, all the combinations have stronger power of detecting non-zeros. As for the zero entries, if the original estimates for the non-transformed ***MV***t_4 ***data*** are moderately good (e.g. 30% to 50% correct detections), then the estimates will benefit the most from using the transformation function huge.npn(). In this case, the correct estimation proportions are always doubled (e.g. to nearly 70% to 87%). If the original detections of the non-transformed ***MV***t_4 ***data*** are already relatively good (e.g. more than 64%) or fairly poor (e.g. less than 20%), then there will be only minor improvements. However, there is also a deterioration in results due to the transformation function huge.npn(), which can either happen to the combinations glasso⋆CV and AL⋆CV at moderate sample sizes or the SCAD combinations at large sample sizes. The latter case has one feature in that all the original performances of the SCAD combinations are pretty good (e.g. more than 83% detections) before transforming the dataset. To be more specific, if the original estimates are already good before transforming the dataset, the transformation function huge.npn() may help little to improve them or even make them worse. In addition, almost all the Huge results are improved by the transformation function huge.npn().

For the high dimensional ***MV***t_4 ***data,*** all the AIC/BIC results for the ***MV***t_4 ***data*** using the transformation function huge.npn() are notably sparser than those for the non-transformed results, while they are still denser than the results for non-transformed ***MV***t_4 ***data*** using the correct AIC/BIC formula. This indicates that the correct AIC/BIC formula has the stronger power to produce sparse estimates than huge.npn(). In contrast to AIC/BIC, huge.npn() makes most of the CV estimates slightly denser. Moreover, if the entries of the true precision matrices are relatively large (e.g. > 0.2), Huge⋆ric results are improved a lot after applying huge.npn(), which is the only case that a Huge combination can effectively estimate the graphical model without losing its structure.

6.3.5 Conclusions for the multivariate normal data with AR1 autocorrelation structure

In general, SCAD is the most stable estimating method against autocorrelation and it produces the best estimates among all the estimating methods. AL suffers the most from correctly estimating zero among all the estimating methods. CV is the most resistant selection criterion to the auto-correlation structure. Furthermore, the ability of correctly detecting the true entries of all estimating methods increases with the sample sizes.

For the low dimensional ***MVN AR1 data***, the top recommendations are SCAD⋆CV and AL⋆CV at any sample size, and SCAD⋆BIC is also recommended at large sample sizes. For the Huge combinations on the low dimensional ***MVN AR1 data***, the Huge⋆ric performs much better than the other combinations. The power of detecting the zero entries by Huge⋆ric dramatically rises, while its ability of estimating non-zeros is seriously weaken, especially for small sample sizes, which indicates that the estimated graphs by Huge⋆ric will become much sparser for the ***MVN AR1 data***. In contrast, Huge⋆ebic and Huge⋆stars estimates are slightly denser. All of them maintain the true graphical structure.

For the high dimensional ***MVN AR1 data***, AL⋆CV, BG⋆CV and glasso⋆CV are suggested for the matrices with relatively small entries, since SCAD⋆CV will over shrink entries and lead to excessively sparse estimates. If the true entries are relatively large, SCAD⋆CV is recommended due to its property of achieving more sparsity than other combinations. Compared to the low dimensional case, the Huge combinations have quite different performance in the high dimensional case. Interestingly, no values are returned for the Huge⋆ric combination from the package huge for all the three precision matrices, thus we obtain no estimated precision matrices by Huge⋆ric. Huge⋆stars and Huge⋆ebic still produced excessively denser and sparse estimates as they do for other high dimensional datasets.

Conclusively, the major influence of the AR1 autocorrelation structure added to each variable of the dataset is that it remarkably reduce the sparsities of the estimated graphs. In other words, for both low and high dimensions, not only the

results of the ***MVN AR1 data*** are much denser than the ***MVN data*** estimates but also the ability of correctly estimating both non-zeros and zeros become much weaker. The excessively dense graph estimated by most of the methods for this data type and the problems with Huge⋆ric indicate that the autocorrelation structure inherent in the data cause more harm to the estimates than assuming a wrong distribution for the data.

Moreover, the AR(1) auto correlation structure disenable the SCAD and BG methods to conduct estimation, which will get stuck halfway through the repetitions. Among all the obtained results, AIC and BIC results become too dense to capture the true graphical structure, while the CV results become slightly sparser than the ***MVN data*** estimates. Also, there is no return value of Huge⋆ric for any of the repetition and the other two Huge combinations become slightly denser than the ***MVN data***.

Bibliography

Hirotugu Akaike. A new look at the statistical model identification. *IEEE Trans. Automatic Control*, AC-19:716–723, 1974. ISSN 0018-9286. System identification and time-series analysis.

Onureena Banerjee, Laurent El Ghaoui, and Alexandre d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *J. Mach. Learn. Res.*, 9:485–516, June 2008. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1390681.1390696.

Jiahua Chen and Zehua Chen. Extended Bayesian information criteria for model selection with large model spaces. *Biometrika*, 95(3):759–771, 2008. ISSN 0006-3444. doi: 10.1093/biomet/asn034. URL http://dx.doi.org/10.1093/biomet/asn034.

Ivor Cribben. *Detecting Dependence Change Points in Multivariate Time Series with Applications in Neuroscience and Finance*. ProQuest LLC, Ann Arbor, MI, 2012. ISBN 978-1267-62451-2. URL http://gateway.proquest.com/openurl?url_ver=Z39.88-2004&rft_val_fmt=info:ofi/fmt:kev:mtx:dissertation&res_dat=xri:pqm&rft_dat=xri:pqdiss:3527904. Thesis (Ph.D.)–Columbia University.

Ivor Cribben, Ragnheidur Haraldsdottir, Lauren Y. Atlas, Tor D. Wager, and Martin A. Lindquist. Dynamic connectivity regression: Determining state-related changes in brain connectivity. *NeuroImage*, 61(4):907 – 920, 2012. ISSN 1053-8119. doi: http://dx.doi.org/10.1016/j.neuroimage.2012.03.070. URL http://www.sciencedirect.com/science/article/pii/S1053811912003515.

Ivor Cribben, Tor Wager, and Martin Lindquist. Detecting functional connectivity change points for single-subject fmri data. *Frontiers in Computational Neuroscience*, 7(143), 2013. ISSN 1662-5188. doi: 10.3389/fncom.2013.00143. URL http://www.frontiersin.org/computational_neuroscience/10.3389/fncom.2013.00143/abstract.

Joachim Dahl, Lieven Vandenberghe, and Vwani Roychowdhury. Covariance selection for nonchordal graphs via chordal embedding. *Optim. Methods Softw.*, 23 (4):501–520, 2008. ISSN 1055-6788. doi: 10.1080/10556780802102693. URL http://dx.doi.org/10.1080/10556780802102693.

Bradley Efron and Robert J. Tibshirani. *An introduction to the bootstrap*, volume 57 of *Monographs on Statistics and Applied Probability*. Chapman and Hall, New York, 1993. ISBN 0-412-04231-2. doi: 10.1007/978-1-4899-4541-9. URL http://dx.doi.org/10.1007/978-1-4899-4541-9.

Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *J. Amer. Statist. Assoc.*, 96(456):1348–1360, 2001. ISSN 0162-1459. doi: 10.1198/016214501753382273. URL http://dx.doi.org/10.1198/016214501753382273.

Jianqing Fan and Jinchi Lv. Sure independence screening for ultrahigh dimensional feature space. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 70(5):849–911, 2008. ISSN 1369-7412. doi: 10.1111/j.1467-9868.2008.00674.x. URL http://dx.doi.org/10.1111/j.1467-9868.2008.00674.x.

Jianqing Fan, Yang Feng, and Yichao Wu. Network exploration via the adaptive lasso and SCAD penalties. *Ann. Appl. Stat.*, 3(2):521–541, 2009. ISSN 1932-6157. doi: 10.1214/08-AOAS215. URL http://dx.doi.org/10.1214/08-AOAS215.

Rina Foygel and Mathias Drton. Extended bayesian information criteria for gaussian graphical models. In J.D. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in*

Neural Information Processing Systems 23, pages 604–612. Curran Associates, Inc., 2010. URL http://papers.nips.cc/paper/4087-extended-bayesian-information-criteria-for-gaussian-graphical-models.pdf.

Jerome Friedman, Trevor Hastie, Holger Höfling, and Robert Tibshirani. Pathwise coordinate optimization. *Ann. Appl. Stat.*, 1(2):302–332, 2007. ISSN 1932-6157. doi: 10.1214/07-AOAS131. URL http://dx.doi.org/10.1214/07-AOAS131.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008. doi: 10.1093/biostatistics/kxm045. URL http://biostatistics.oxfordjournals.org/content/9/3/432.abstract.

Jerome. Friedman, Trevor. Hastie, and Rob Tibshirani. Package "glasso". 2014. URL http://www-stat.stanford.edu/~tibs/glasso.

Jerome H. Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2 2010. ISSN 1548-7660. URL http://www.jstatsoft.org/v33/i01.

Peter Langfelder and Steve Horvath. Wgcna: an R package for weighted correlation network analysis. *BMC Bioinformatics*, (1):559, 2008.

Ker-Chau Li. Asymptotic optimality for C_p, C_L, cross-validation and generalized cross-validation: discrete index set. *Ann. Statist.*, 15(3):958–975, 1987. ISSN 0090-5364. doi: 10.1214/aos/1176350486. URL http://dx.doi.org/10.1214/aos/1176350486.

Han Liu, John Lafferty, and Larry Wasserman. The nonparanormal: semiparametric estimation of high dimensional undirected graphs. *J. Mach. Learn. Res.*, 10:2295–2328, 2009. ISSN 1532-4435.

Han Liu, Kathryn Roeder, and Larry Wasserman. Stability approach to regularization selection (stars) for high dimensional graphical models. In J. Lafferty, C. Williams, J. Shawe-taylor, R.s. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1432–1440. 2010. URL http://books.nips.cc/papers/files/nips23/NIPS2010_0834.pdf.

Shaun Lysen. *Permuted inclusion criterion: A variable selection technique*. ProQuest LLC, Ann Arbor, MI, 2009. ISBN 978-1109-70929-2. URL http://gateway.proquest.com/openurl?url_ver=Z39.88-2004&rft_val_fmt=info:ofi/fmt:kev:mtx:dissertation&res_dat=xri:pqdiss&rft_dat=xri:pqdiss:3405354. Thesis (Ph.D.)–University of Pennsylvania.

Rahul Mazumder and Trevor Hastie. Exact covariance thresholding into connected components for large-scale graphical lasso. *J. Mach. Learn. Res.*, 13:781–794, 2012a. ISSN 1532-4435.

Rahul Mazumder and Trevor Hastie. The graphical lasso: new insights and alternatives. *Electron. J. Stat.*, 6:2125–2149, 2012b. ISSN 1935-7524. doi: 10.1214/12-EJS740. URL http://dx.doi.org/10.1214/12-EJS740.

Nicolai Meinshausen and Peter Bühlmann. High-dimensional graphs and variable selection with the lasso. *Ann. Statist.*, 34(3):1436–1462, 2006. ISSN 0090-5364. doi: 10.1214/009053606000000281. URL http://dx.doi.org/10.1214/009053606000000281.

Nicolai Meinshausen and Peter Bühlmann. Stability selection. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 72(4):417–473, 2010. ISSN 1369-7412. doi: 10.1111/j.1467-9868.2010.00740.x. URL http://dx.doi.org/10.1111/j.1467-9868.2010.00740.x.

Ryuei Nishii. Asymptotic properties of criteria for selection of variables in multiple regression. *Ann. Statist.*, 12(2):758–765, 1984. ISSN 0090-5364. doi: 10.1214/aos/1176346522. URL http://dx.doi.org/10.1214/aos/1176346522.

B. T. Polyak and A. B. Tsybakov. Asymptotic optimality of the C_p-test in the projection estimation of a regression. *Teor. Veroyatnost. i Primenen.*, 35(2):305–317, 1990. ISSN 0040-361X. doi: 10.1137/1135037. URL http://dx.doi.org/10.1137/1135037.

Mohsen Pourahmadi. Covariance estimation: the GLM and regularization perspectives. *Statist. Sci.*, 26(3):369–387, 2011. ISSN 0883-4237. doi: 10.1214/11-STS358. URL http://dx.doi.org/10.1214/11-STS358.

Gideon Schwarz. Estimating the dimension of a model. *Ann. Statist.*, 6(2):461–464, 1978. ISSN 0090-5364.

Jun Shao. Linear model selection by cross-validation. *J. Amer. Statist. Assoc.*, 88(422):486–494, 1993. ISSN 0162-1459. URL http://links.jstor.org/sici?sici=0162-1459(199306)88:422⟨486:LMSBC⟩2.0.CO;2-C&origin=MSN.

Jun Shao. An asymptotic theory for linear model selection. *Statist. Sinica*, 7(2):221–264, 1997. ISSN 1017-0405. With comments and a rejoinder by the author.

Ritei Shibata. Asymptotic mean efficiency of a selection of regression variables. *Ann. Inst. Statist. Math.*, 35(3):415–423, 1983. ISSN 0020-3157. doi: 10.1007/BF02480998. URL http://dx.doi.org/10.1007/BF02480998.

Stephen M. Smith, Karla L. Miller, Gholamreza Salimi-Khorshidi, Matthew Webster, Christian F. Beckmann, Thomas E. Nichols, Joseph D. Ramsey, and Mark W. Woolrich. Network modelling methods for {FMRI}. *NeuroImage*, 54(2):875 – 891, 2011. ISSN 1053-8119. doi: http://dx.doi.org/10.1016/j.neuroimage.2010.08.063. URL http://www.sciencedirect.com/science/article/pii/S1053811910011602.

M. Stone. An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion. *J. Roy. Statist. Soc. Ser. B*, 39(1):44–47, 1977. ISSN 0035-9246.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc. Ser. B*, 58(1):267–288, 1996. ISSN 0035-9246. URL http://links.jstor.org/sici?sici=0035-9246(1996)58:1⟨267:RSASVT⟩2.0.CO;2-G&origin=MSN.

Daniela M. Witten, Jerome H. Friedman, and Noah Simon. New insights and faster computations for the graphical lasso. *J. Comput. Graph. Statist.*, 20(4):892–900, 2011. ISSN 1061-8600. doi: 10.1198/jcgs.2011.11051a. URL http://dx.doi.org/10.1198/jcgs.2011.11051a.

Yuhong Yang. Can the strengths of AIC and BIC be shared? A conflict between model indentification and regression estimation. *Biometrika*, 92(4):937–950, 2005. ISSN 0006-3444. doi: 10.1093/biomet/92.4.937. URL http://dx.doi.org/10.1093/biomet/92.4.937.

Ming Yuan and Yi Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94(1):19–35, 2007. ISSN 0006-3444. doi: 10.1093/biomet/asm018. URL http://dx.doi.org/10.1093/biomet/asm018.

Tuo Zhao, Han Liu, Kathryn Roeder, John Lafferty, and Larry Wasserman. The huge package for high-dimensional undirected graph estimation in r. *J. Mach. Learn. Res.*, 13:1059–1062, 2012. ISSN 1532-4435.

Tuo Zhao, Han Liu, Kathryn Roeder, John Lafferty, and Larry Wasserman. Package "huge". 2014. URL cran.r-project.org/web/packages/huge/huge.pdf.

Hui Zou. The adaptive lasso and its oracle properties. *J. Amer. Statist. Assoc.*, 101(476):1418–1429, 2006. ISSN 0162-1459. doi: 10.1198/016214506000000735. URL http://dx.doi.org/10.1198/016214506000000735.

Hui Zou and Runze Li. One-step sparse estimates in nonconcave penalized likelihood models. *Ann. Statist.*, 36(4):1509–1566, 2008. ISSN 0090-5364. doi: 10.1214/009053607000000802. URL http://dx.doi.org/10.1214/009053607000000802.

Appendix A

Tables

A.1 Results for the tridiagonal matrix with $a = 0.85$

A.1.1 *MVN data* ($p = 30$) results

Table 31: *The TPs and TNs for the tridiagonal matrix with* $a = 0.85$ *for the* ***MVN data*** *(*$p = 30$*)*

Ω	**tri** $a = 0.85$					
	TP			**TN**		
	AIC	**BIC**	**CV**	**AIC**	**BIC**	**CV**
CM	1			0		
PCM	1			0		
Glasso	1	1	1	0.139	0.53	0.58
AL	1	1	1	0.415	0.786	0.827
SCAD	1	1	1	0.581	0.947	0.97
DP						
Glasso	1	1	1	0.14	0.53	0.58
AL	1	1	1	0.416	0.782	0.827
SCAD	1	1	1	0.584	0.948	0.972
BG						
$H = 50$	1	1	1	0.486	0.815	0.732
DP-boo	stuck					
Huge						
ric	NA			NA		
stars	1			0.558		
ebic	1			0.712		

A.1.2 MVt_3 *data* ($p = 30$) results

Table 32: *The TPs and TNs for the tridiagonal matrix with* $a = 0.85$ *for the* MVt_3 ***data*** *(*$p = 30$*)*

Ω	**tri** $a = 0.85$					
	TP			**TN**		
	AIC	**BIC**	**CV**	**AIC**	**BIC**	**CV**
CM	1			0		
PCM	1			0		
Glasso	1	1	1	0.048	0.174	0.626
AL	1	1	0.995	0.211	0.376	0.838
SCAD	stuck					
new	**AIC**	**BIC**				
Glasso	1	1	1	0.439	0.625	0.626
AL	stuck					
SCAD	stuck					
BG						
$H = 50$	1	1	0.999	0.183	0.316	0.827
new A/BIC	0.999	1	0.999	0.725	0.961	0.827
Huge						
ric	NA			NA		
stars	1			0.483		
ebic	1			0.62		

A.1.3 MVt_4 *data* ($p = 30$) results

Table 33: *The TPs and TNs for the tridiagonal matrix with* $a = 0.85$ *for the* MVt_4 ***data*** *(*$p = 30$*)*

Ω	**tri** $a = 0.85$					
	TP			**TN**		
	AIC	**BIC**	**CV**	**AIC**	**BIC**	**CV**
CM	1			0		
PCM	1			0		
Glasso	1	0.999	1	0.06	0.238	0.618
AL	0.769	0.508	0.16	0.257	0.548	0.885
SCAD	0.5	0.324	0.103	0.561	0.75	0.935
new	**AIC**	**BIC**				
Glasso	1	1	1	0.453	0.643	0.618
AL	0.999	0.999	0.999	0.703	0.859	0.836
SCAD	stuck					
BG						
$H = 50$	1	0.999	0.999	0.238	0.412	0.805
new A/BIC	0.999	0.998	0.999	0.708	0.964	0.805
Huge						
ric	NA			NA		
stars	1			0.492		
ebic	1			0.65		

Table 34: *The TPs and TNs for the tridiagonal matrix with* $a = 0.85$ *for the* ***MV***t_4 ***data*** *(*$p = 30$*) using the nonparanormal transformation function* `huge.npn()`

Ω	**tri** $a = 0.85$					
	TP			**TN**		
	AIC	**BIC**	**CV**	**AIC**	**BIC**	**CV**
CM	1			0		
PCM	1			0		
glasso	1	1	1	0.477	0.477	0.564
AL	1	1	1	0.848	0.848	0.86
SCAD	1	0.999	1	0.839	0.871	0.938
BG						
$H = 50$	0.999	0.999	0.999	0.942	0.942	0.942
Huge						
ric	1			0.711		
stars	1			0.477		
ebic	1			0.696		

A.1.4 *MVN AR1 data* ($p = 30$) results

Table 35: *The TPs and TNs for the tridiagonal matrix with* $a = 0.85$ *for the* ***MVN AR1 data*** *(*$p = 30$*)*

Ω	**tri** $a = 0.85$					
	TP			**TN**		
	AIC	**BIC**	**CV**	**AIC**	**BIC**	**CV**
CM	1			0		
PCM	1			0		
glasso	1	1	0.999	0.048	0.075	0.637
AL	0.998	0.998	0.997	0.211	0.231	0.843
SCAD	stuck					
BG						
$H = 50$	stuck					
Huge						
ric	NA			NA		
stars	1			0.458		
ebic	1			0.575		

A.2 Negative estimates frequencies

Table 36: *The frequencies of getting negative estimates on the main diagonal for the low dimensional datasets*

	n		
MVN AR1 data $(p = 5)$	100	**Huge⋆ric**	0.93
	200	**Huge⋆ric**	0.63
	500	**Huge⋆ric**	0.12
MV*t_3 *data $(p = 5)$	100	**SCAD⋆CV**	0.01
	100	**Huge⋆ric**	0.12
	200	**Huge⋆ric**	0.01
MV*t_4 *data $(p = 5)$	100	**SCAD⋆CV**	0.01
	100	**Huge⋆ric**	0.03

Table 37: *The frequencies of getting negative estimates on the main diagonal for the high dimensional datasets*

	Ω		
MVN data *($p = 30$)*	tri_0.85	**SCAD⋆BIC**	0.01
		DP-SCAD⋆BIC	0.01
		DP-SCAD⋆CV	0.01
		Huge⋆ric	1
MVN AR1 data *($p = 30$)*	tri_0.85	**Huge⋆ric**	1
	tri_1.7	**Huge⋆ric**	1
	exp	**Huge⋆ric**	1
	gen	**Huge⋆ric**	1
MVt_3 data *($p = 30$)*	tri_0.85	**Huge⋆ric**	1
	tri_1.7	**Huge⋆ric**	0.11
	exp	**SCAD⋆CV**	0.01
	exp	**Huge⋆ric**	0.03
	gen	**Huge⋆ric**	0.05
MVt_4 data *($p = 30$)*	tri_0.85	**Huge⋆ric**	1
	tri_1.7	**Huge⋆ric**	0.01

Appendix B

Figures

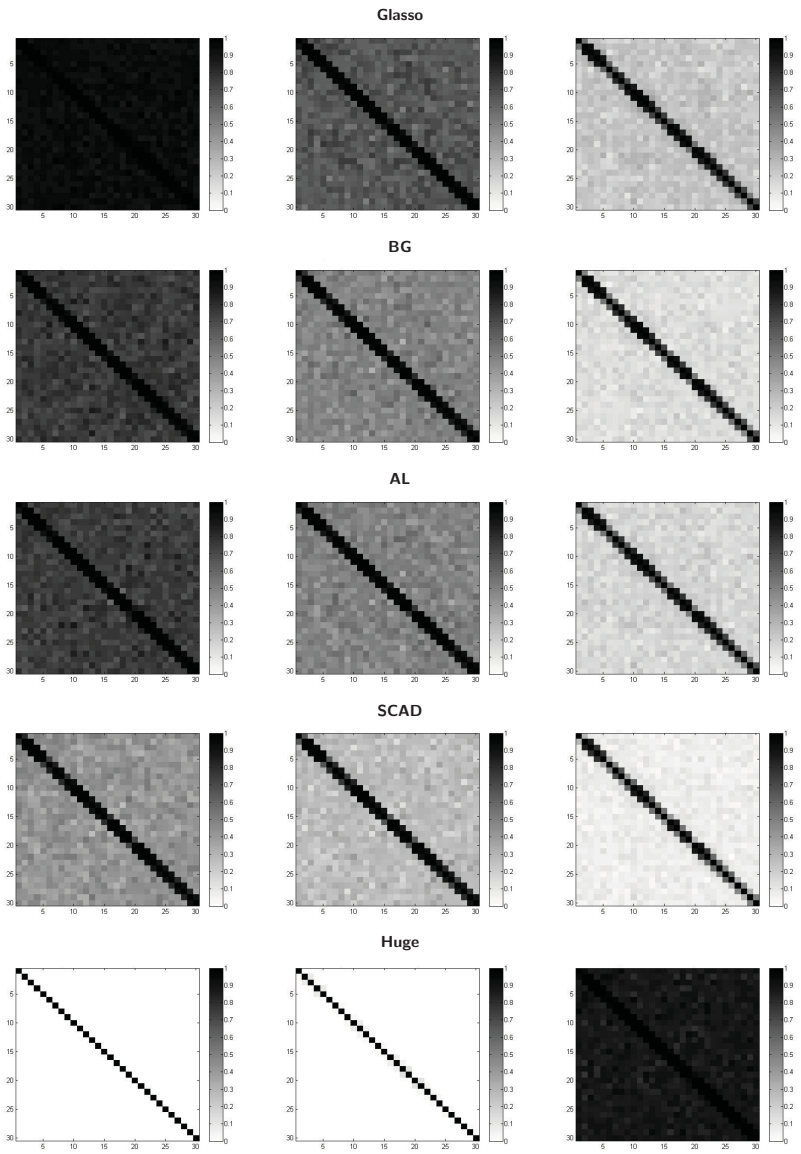

Figure 26: *The ASP plots for the* $\boldsymbol{MVt_4}$ ***data*** *(*$p = 30$*) generated from the tridiagonal matrix with* $a = 1.7$*, using the wrong AIC/BIC formula*

[Left panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the wrong AIC formula. [Middle Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the wrong BIC formula. [Right Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the CV criterion.

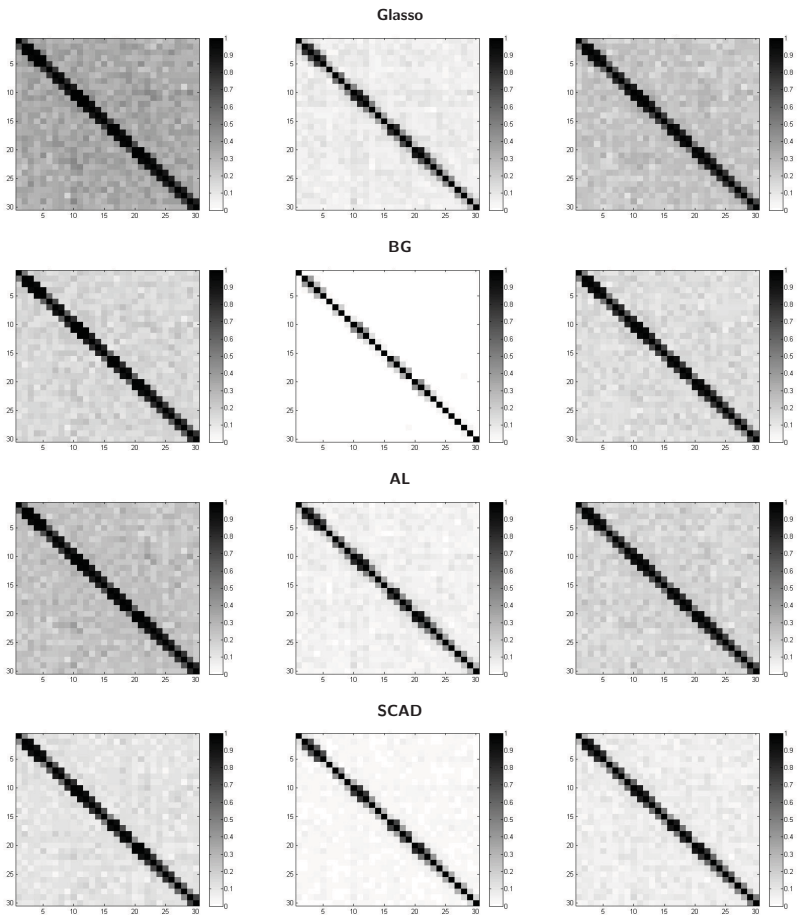

Figure 27: *The ASP plots for the* $\boldsymbol{MVt_4}$ ***data*** *(*$p = 30$*) generated from the tridiagonal matrix with* $a = 1.7$*, using the correct AIC/BIC formula*

[Left panel] The ASP plots of the glasso, BG, AL and SCAD estimates selected by the correct AIC formula. [Middle Panel] The ASP plots of the glasso, BG, AL and SCAD estimates selected by the correct BIC formula. [Right Panel] The ASP plots of the glasso, BG, AL and SCAD estimates selected by the CV criterion.

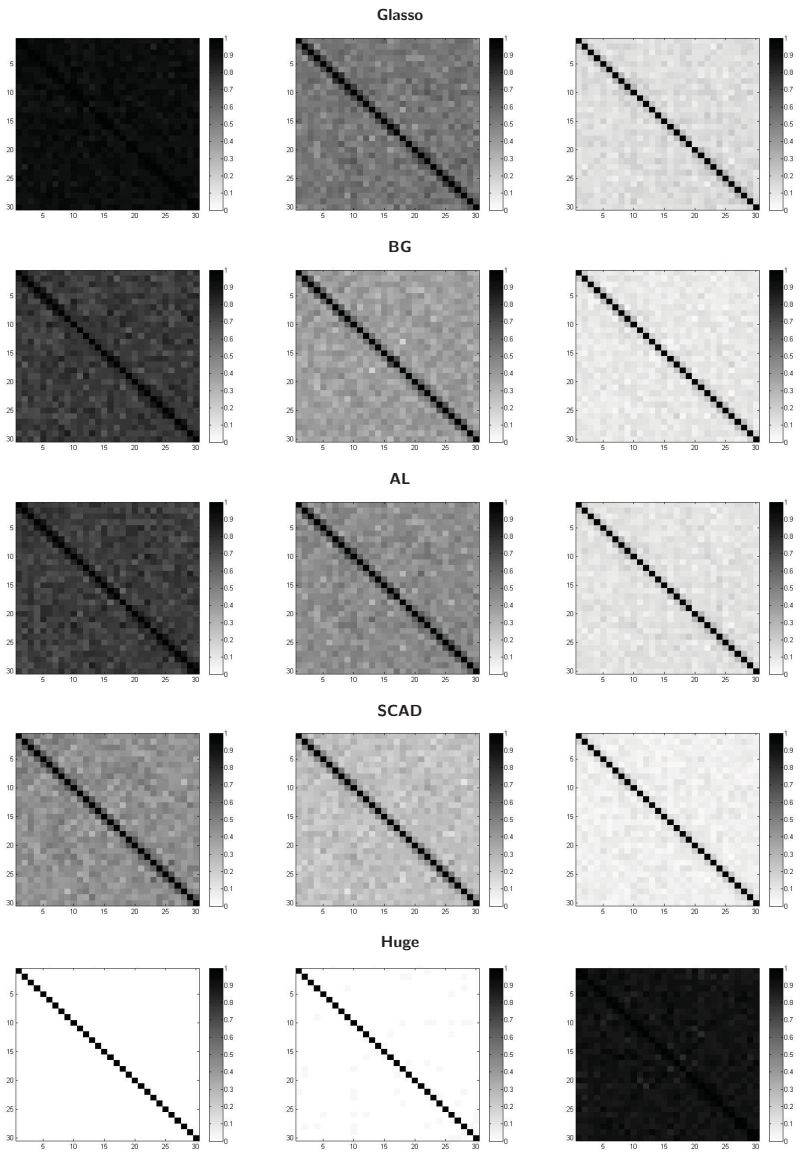

Figure 28: *The ASP plots for the* ***MV****t*$_4$ ***data*** *(*$p = 30$*) generated from the exponential decay matrix, using the wrong AIC/BIC formula*

[Left panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the wrong AIC formula. [Middle Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the wrong BIC formula. [Right Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the CV criterion.

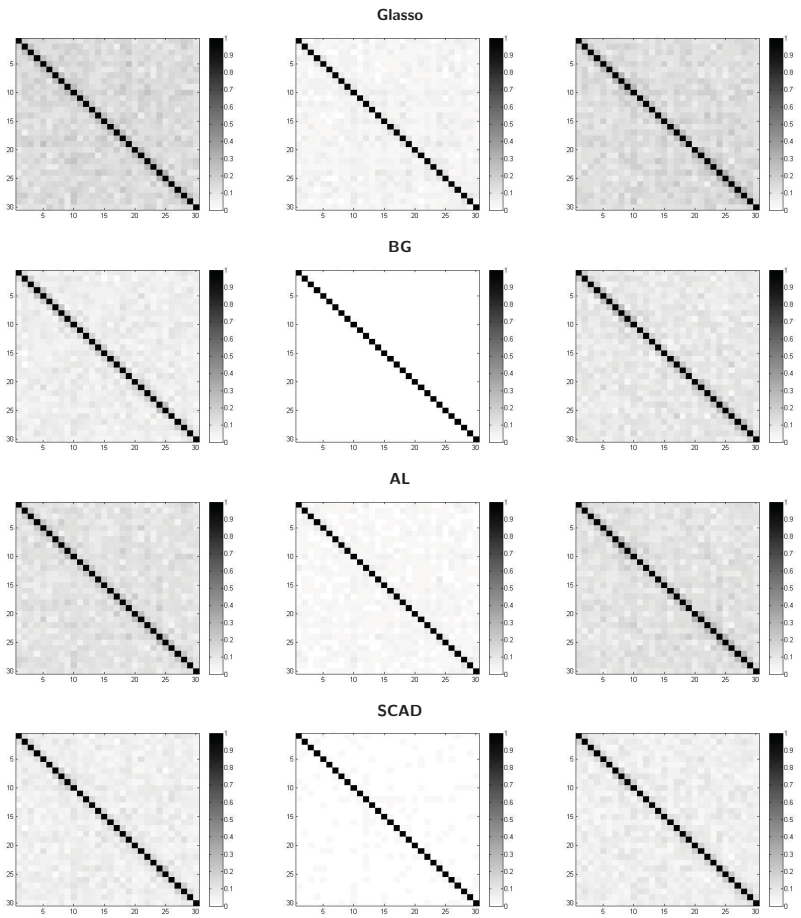

Figure 29: *The ASP plots for the* $\boldsymbol{MVt_4}$ ***data*** *(*$p = 30$*) generated from the exponential decay matrix, using the correct AIC/BIC formula*

[Left panel] The ASP plots of the glasso, BG, AL and SCAD estimates selected by the correct AIC formula. [Middle Panel] The ASP plots of the glasso, BG, AL and SCAD estimates selected by the correct BIC formula. [Right Panel] The ASP plots of the glasso, BG, AL and SCAD estimates selected by the CV criterion.

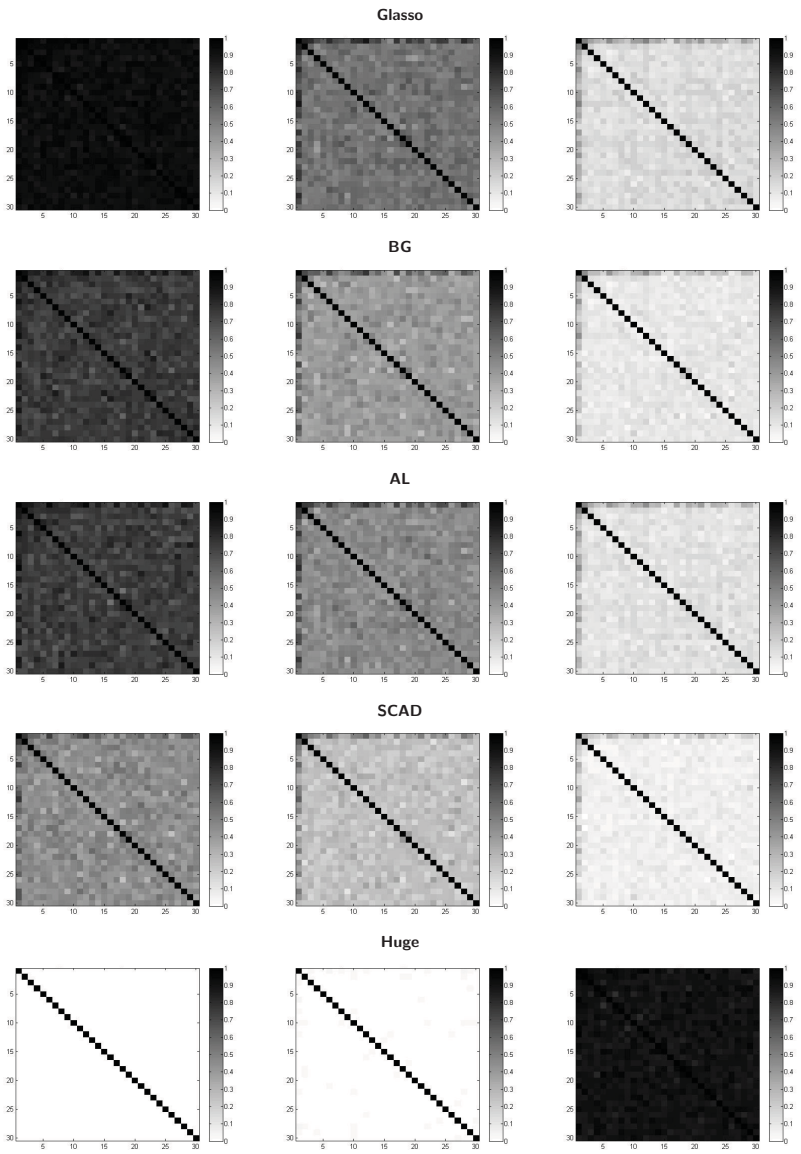

Figure 30: *The ASP plots for the* $\boldsymbol{MVt_4}$ ***data*** $(p = 30)$ *generated from the general matrix, using the wrong AIC/BIC formula*

[Left panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the wrong AIC formula. [Middle Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the wrong BIC formula. [Right Panel] The ASP plots of the glasso, BG, AL, SCAD and Huge estimates selected by the CV criterion.

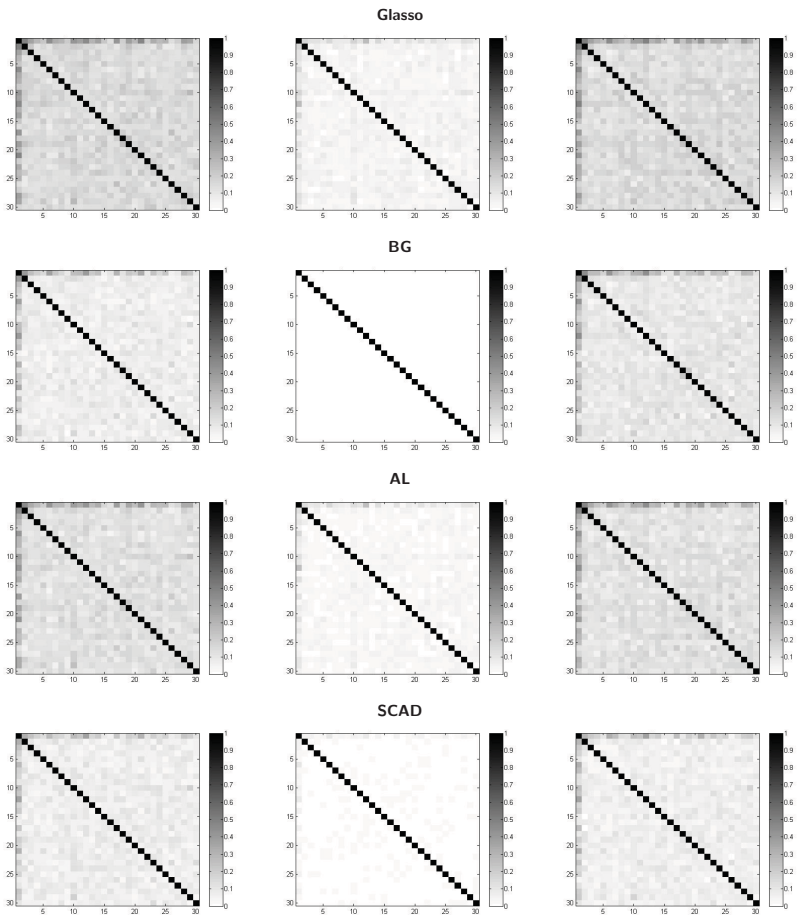

Figure 31: *The ASP plots for the* $\boldsymbol{MVt_4}$ ***data*** *(*$p = 30$*) generated from the general matrix, using the correct AIC/BIC formula*

[Left panel] The ASP plots of the glasso, BG, AL and SCAD estimates selected by the correct AIC formula. [Middle Panel] The ASP plots of the glasso, BG, AL and SCAD estimates selected by the correct BIC formula. [Right Panel] The ASP plots of the glasso, BG, AL and SCAD estimates selected by the CV criterion.