STRUCTURED REGULARIZED DIMENSIONALITY REDUCTION ON TWO REAL APPLICATIONS

A Dissertation

by

YAN ZHONG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Co-Chair of Committee,	Jianhua Huang
	Xia Hu
Committee Members,	Raymond Ka Wai Wong
	James Cai
Head of Department,	Branislav Vidakovic

May 2021

Major Subject: Statistics

ABSTRACT

Datasets with a large number of observations and variables, called large datasets, become ubiquitous as a consequence of the development of technology. To deal with large datasets, data scientists face challenges such as the overfitting problem and the computational problem. Particularly, an important issue when analyzing large datasets is to study the structural information of observations and features. This dissertation focuses on a currently popular strategy called the structured regularized dimensionality reduction to analyze large datasets, which utilizes dimensionality reduction and regularization techniques to incorporate structural information into the model. We build new machine learning models of structured regularized dimensionality reduction for two real applications. In the first application, we propose a regularized spatially varying coefficient model to select important variables and estimate spatially clustered coefficients simultaneously in the spatial regression problem. In the second application, we build a regularized matrix decomposition model to solve the biclustering problem with a complex layout of latent biclusters in the data matrix.

ACKNOWLEDGMENTS

I appreciate a lot for my co-advisor, Dr. Jianhua Huang and Dr. Xia Hu, who guided me and helped me overcome many difficulties during my doctoral life. I would like to thank my other committee members, Dr. Raymond Ka Wai Wong, and Dr. James Cai, who offered their time, support, and good thoughts to review and improve this dissertation.

I am grateful to Dr. Huiyan Sang and Dr. Xiaohu Xu. Dr. Sang taught me and guided me a lot in doing the spatial data analysis. Dr. Xu guided me in doing the interdisciplinary studies, and I gained a lot of research experiences from him.

I wish to thank all my other cooperators, Dr. Xiao Zhang, Dr. Xiao Huang, Dr. Jundong Li, Daniel Osorio, Guanxun Li, Scott Cook, Dr. Paul Kellstedt, Huiling Liao, and Huang Ke. I enjoyed the time working with them.

I also would like to acknowledge Dr. Kejun He, Xiaomeng Yan, Dr. Guorong Dai, Jiangyuan Li, and the other students of the Department of Statistics for the valuable discussions that helped improve my research.

Finally, I would like to thank my parents. Their love and support never stop during all these years. I would like to express my deepest gratitude to them.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Dr. Jianhua Huang of the Department of Statistics (co-advisor), Dr. Xia Hu of the Department of Computer Science and Engineering (co-advisor), Dr. Raymond Ka Wai of the Department of Statistics, and Dr. James Cai of the Department of Veterinary Integrative Biosciences.

The analyses depicted in Chapter 2 were conducted in part by Dr. Huiyan Sang of the Department of Statistics, Scott Cook of the Department of Political Science, and Dr. Paul Kellstedt of the Department of Political Science.

All other work conducted for the dissertation was completed by the student independently.

Funding Sources

This work is, in part, supported by NSF (#IIS-1900990).

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Large Data Analysis

With the development of technology, the number of observations and variables included in the collected datasets is increasing rapidly. For example, the online social network data contains various features for millions of users, whose volume is much bigger than the conventional social network data. Next generation sequencing enables scientists to access the expression level of genes in the resolution of single cell, which increases the number of observations from less than 100 in the bulk cell analysis to around 10000. Many other examples of large datasets exist for applications from different fields, and the conventional statistical models face several new challenges to deal with these large datasets.

First of all, a lot of structural information can exist for large datasets. Examples include temporal data, spatial data, network data, multi-source data, and so on. With structural information, the basic assumptions of many conventional models will not be satisfied. From the sample aspect, the observations are usually assumed to be independent and homogeneous. However, since the number of observations is large, it is likely that observations come from different clusters and are heterogeneous. Moreover, prior information about the local relationship among observations can exist, showing that they are not independent. From the variable aspect, it is common that some variables are correlated for a large variable set, and high dimensional data may locate in a low dimensional manifold space. Without considerint structural information, the conventional models may not be effective and even provide some misleading results. Thus, it is important to incorporate the structural information into the models.

Second, the number of parameters is also large for large datasets, which causes problems in modeling. For example, the overfitting problem may occur for a model with too many parameters, in which the modeling results are too close to the training observations and cannot perform well for new observations. Also, when the number of parameters is larger than the number of samples,

some models face the problem of unidentifiability. Thus, it requires researchers to find proper methods to decrease the number of parameters or restrict the flexibility of parameters to avoid the above problems in large data analysis.

Third, many conventional statistical models have high computational complexity, and it takes too much time for them to deal with large datasets. For example, Gaussian process is a very effective way to model a smooth function. However, the naive implementation of Gaussian process requires $O(n^3)$ times to build a model for n observations, which is computationally difficult when n is large. As a result, how to deal with the computing issues of the existing methods for large datasets is a very important problem.

There are also some other challenges for large data analysis such as the requirement for the storage, the security problem, and so on. These challenges are also important but are not the focus of this dissertation.

To solve the above three challenges when facing a real application, this dissertation studies a particular group of methods called structured regularized dimensionality reduction.

1.2 Structured Regularized Dimensionality Reduction

Dimensionality reduction is to transform the high dimensional data to the low dimensional data that contain most information of the original data. There are two groups of dimensionality reduction methods: feature selection and feature extraction. For feature selection, researchers select a subset of variables from all variables that relate to the study. For example, in the regression problem, researchers use the step-wise method [1] to find a small number of variables that determine the variability of the response variable. On the other hand, feature extraction focuses on constructing a low dimensional projection space of the original high dimensional data. A conventional example of the feature extraction methods is principal component analysis (PCA) [2]. In PCA, people find the principal components of the high dimensional feature space that have the largest sample variance.

Since the 21st century, the regularization technique has become popular, which restricts the flexibility of the parameters by adding some regularization terms into the model. The famous reg-

ularization terms include Lasso [3], group Lasso [4], SCAD [5], fused Lasso [6], and so on. The regularization technique has several advantages for large data analysis. First, the regularization technique can restrict the flexibility of the parameters and solve the ill-posed problem and overfitting in the model. Second, the formulation of the regularization term can be designed according to the structure information of data and increase the interpretability of the model. Last but not the least, many regularization terms do not have a large computing cost. Many regularization terms are even convex functions and can be efficiently solved by convex optimization.

In conclusion, structured regularized dimensionality reduction refers to the methods that utilize the idea of dimensionality reduction and the regularisation technique to incorporate the spatial information of the data and solve the real problems. It is a powerful tool to solve the problem for large datasets.

1.3 Dissertation Overview

This dissertation solves two real applications by constructing new machine learning models of structured regularized dimensionality reduction. In each application, we first study the requirements of structure for parameters and design a specific regularization term of parameters to pursue the required structure. Then, we build an optimization problem to address the application and provide an efficient algorithm to solve this problem.

In Chapter 2, we introduce our proposed method for a specific spatial regression problem. Our method could achieve simultaneously variable selection and spatially coefficient clustering, which perform well in the analysis of the influence of the covariates on the United States presidential election results.

In Chapter 3, we propose a new biclustering method that solves the biclustering problem with a complex layout of latent biclusters in the data matrix. The experimental results on the single-cell RNA sequencing data demonstrate the effectiveness of our method.

2. ADAPTIVE VARIABLE SELECTION IN SPATIALLY CLUSTERED COEFFICIENT MODEL

2.1 Introduction

Spatial analysis has become an increasingly popular tool in social science studies, allowing researchers to explain social behavior and its relationship with a set of potential factors over a region of interests while accounting for spatial structures in the data [7]. These efforts are, however, complicated, since many factors (known and unknown) are likely to influence social behavior, and the effect of these factors are likely to vary from place to place. To account for this, researchers have developed spatially clustered coefficient (SCC) models, which allow practitioners to capture discontinuous changes in regression relationships across space. Existing SCC methods tend to assume a fixed set of predictors pre-supplied by the researcher. However, when a large number of available covariates need to be considered at the initial stage of modeling, practitioners frequently lack a strong theory to inform model selection when analyzing spatial data. Thus, researchers are often forced to either pre-select a subset of these variables in an ad hoc fashion or ignore potential spatial heterogeneity outright. When the data do not support these assumptions (i.e., accurate model selection or spatial homogeneity), as often the case with real-world spatial data, the resulting model will be inappropriate. Instead, we need a more general model that allows researchers to undergo variable selection and spatial cluster identification flexibly.

Our specific motivation is the correlates of aggregate voting behavior in U.S. presidential elections. While voter behavior in presidential elections has received considerable attention in political science, research tends to focus on individual-level behavior (via surveys) and state- or national-level results. Here we are instead interested in explaining local-level variation in aggregate (i.e., county-level) electoral results, an area which has received less attention to date. The few notable exceptions [8, 9, 10] that have analyzed county-level elections tend to focus on demographic characteristics and economic indicators. For example, [8] focuses on the local economy and the racial

composition in explaining voting results in 2008, [10] studies the relationship between the unemployment rate and the Democratic vote share in presidential and gubernatorial elections, and [9] assesses the influence of local unemployment on presidential vote share.

Yet these studies consider a relatively small number of pre-selected covariates and estimate the model using linear regression with spatially invariant coefficients. That is, they each assume that the relevant set of predictors is known and that the coefficients are the same for each county. These two assumptions seem questionable in the case of aggregate voting behavior. First, while theory can guide the inclusion of some variables, it is doubtful that this set is exhaustive. Many salient inputs may not have been previously considered, and in dynamically evolving systems it may be the case that prior models are no longer sufficient. As such, considering determinants without an obvious theoretical motivation can be worthwhile. Second, assuming that variation in voting behavior is entirely a function of variation in the level of the input (e.g., unemployment rate) neglects the fact that different counties, states, regions, etc., have distinct cultural, social, and economic practices which can modify the effect of these other county-level characteristics on voting behavior. Model predictors can affect voting patterns in these different regions in distinct ways. Moreover, contiguous counties with similar cultural, social, and economic practices are likely to be clustered together [11]. The effect of model predictors is the same within each cluster of counties but has abrupt changes across the boundary of clusters. In sum, in analyzing aggregate voting patterns, it is often unclear which factors are likely to be salient in a given election, and how their effect may vary across space.

Rather than requiring applied researchers to impose these restrictions by default, we are motivated to propose a new spatial regression model to select important social and demographic variables and detect spatial clustering patterns of voting behavior simultaneously. In the spatial statistics literature, a class of spatially varying coefficient (SVC) models has been proposed to accommodate spatial heterogeneity in regression coefficients. [12] extends the ordinary least square regression by fitting a geographically local regression model at each observation. [13] proposes a Bayesian approach by assigning a multivariate Gaussian process prior to the regression coeffi-

cients. Both methods are more suited for dealing with smoothly varying than clustered regression coefficients. For the spatially clustered coefficient regression (SCC) problem, [14] proposes a fused Lasso regularized optimization method, which uses a Euclidean distance based minimum spanning tree as the “spatial order" to encourage homogeneity between the regression coefficients at two adjacent locations. Each variable is also allowed to have its own underlying spatially clustered patterns. [15] proposes an extension to multivariate regression using adaptive Lasso. Elsewhere, [16] constructs a Bayesian model based on Dirichlet process to solve the SCC problem. While each represents advancements in efficiently identifying spatial clusters, none has incorporated variable selection directly in estimation. Introducing a large set of covariates to current SCC models is likely to cause computational issues, risk overfitting, and make model interpretation more difficult.

In high dimensional statistics, a number of methods have been proposed for variable selection problems for spatial data. [17] develop a spatial adaptive Lasso method for simultaneous model selection and parameter estimation in spatial regression for lattice data. [18] and [19] consider binary spatial regression models and proposed penalized quasi-likelihood methods with spatial dependence for variable selections. [20] and [21] propose variable selection methods via sparse regularization for point process models. Yet, each of these methods assume that the regression coefficients of selected spatial variables are constant across space. In so doing, these approaches may eliminate predictors with complex spatial patterns that only emerge once cluster detection in undertaken. For example, if the average coefficient effect is close to zero, it may be eliminated during variable selection. This could include predictors with very interesting spatial heterogeneity, where clusters have significant effects in different directions that offset when average.

In this chapter, we study the problem of variable selection in the SCC model for an analysis of county-level U.S. presidential election results in 2016 and 2020. Our analysis identifies a number of influential factors (e.g., race composition, poverty, and education) that govern electoral politics narrowed from a long list of social and demographic variables. The analysis results also reveal several interesting clustering patterns of voting behaviors among different counties, which provide important information to study U.S. regional politics.

Our main methodological contributions are threefold. First, we propose a new regularized spatially varying coefficient (RSVC) model for selecting important variables and estimating spatially clustered coefficients simultaneously. The regularization takes an additive form of two terms. The first term is a graphical fused Lasso penalty for spatial clustering, and the second term is a group elastic-net penalty to achieve spatial variable selection. Particularly, we adopt the idea of adaptive learning to allow for adaptive weights and adaptive graphs for improved estimation of parameters. To the best of our knowledge, our method is among the first to propose an adaptive regularization for simultaneously selecting variables and estimating spatially clustered coefficient in the literature. Second, we design an efficient proximal gradient algorithm to solve the regularized regression optimization problem. To address the common computation issues encountered in the optimization involving graphical fused Lasso regularization, we carefully select a chain graph and utilize its structure to speed up the algorithm such that it achieves low computational complexity and can be applied to large spatial data sets. Third, we make a theoretical contribution by providing the non-asymptotic theoretical results for the proposed new regularization in our approach. In particular, we derive an improved convergence rate for the adaptive method over its non-adaptive counterpart.

The proposed RSVC model has several other advantages. It allows the investigation of different clustered patterns in different regression coefficients. The method also enjoys great flexibility in the cluster shapes and naturally induces spatially contiguous clusters so that practitioners can interpret clusters as subregions. Moreover, the number of selected variables and the number of clusters are both treated as unknown and determined from data-driven model-based approaches. Finally, since the method is built upon graphs, it can be used beyond the spatial context to solve the variable selection problem in any clustered coefficient model where observations can be related in a graph or a network. We provide several such real examples in Section 2.7.

This chapter is organized as follows. In Section 2.2, we propose our model. In Section 2.3, we present the theoretical results of this model. Section 2.4 describes the efficient algorithm for the implementation of our model. Sections 2.5 and 2.6 include the simulations to illustrate the model performance and the application to the U.S. election data analysis. We offer discussions in

Section 2.7. The R code, proofs, additional simulation results, and information about the real data are provided in the Supplementary.

2.2 Methodology

2.2.1 Spatially Varying Coefficients

To fix concepts, we first introduce the familiar SVC regression model. Suppose that the response variable is collected at n locations with $\mathbf{Y} = (y_1, y_2, \ldots, y_n)^T \in \mathbb{R}^n$, and $\mathbf{X} \in \mathbb{R}^{n \times p}$ is the corresponding feature matrix with p covariates. The first column of $\mathbf{X}$ is for the intercept with all entries equal to 1. Use $\mathbf{x}_i$ to denote the i^{th} row of $\mathbf{X}$. For each location, the SVC regression assumes that y_i and $\mathbf{x}_i$ follow:

$$y_i = \mathbf{x}_i \mathbf{b}_i + \epsilon_i, \quad \epsilon_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2),$$

where $\mathbf{b}_i \in \mathbb{R}^p$ is the coefficient vector for the i^{th} location. Specifically, the first entry b_{i1} of $\mathbf{b}_i$ refers to the spatially varying intercept for the i^{th} location. It can be interpreted as a random spatial adjustment at each location, which captures the spatial dependence that is unexplained by other covariates. Thus, we model ϵ_i as a random error term rather than a spatially dependent error term. One major goal of the SVC regression is to estimate the coefficient matrix $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)^T \in \mathbb{R}^{n \times p}$.

The traditional way to estimate $\mathbf{B}$ is to minimize the sum of the squared loss of the prediction error. However, there are $n \times p$ unknown parameters but only n observations in the SVC regression, and then $\mathbf{B}$ cannot be identified directly. Instead, we need to make additional assumptions about $\mathbf{B}$ to regularize this ill-posed high-dimensional regression problem.

In the proposed model, we allow for two sources of constraints: one focusing on spatial heterogeneity, and one focusing on variable selection. These two constraints permit simultaneous feature selection and cluster identification. Specifically, we propose the regularized SVC (RSVC) model

and estimate $\mathbf{B}$ by minimizing the objective function:

$$L(\mathbf{B}) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \mathbf{x}_i\mathbf{b}_i)^2 + \lambda_1 P_C(\mathbf{B}) + \lambda_2 P_V(\mathbf{B}), \tag{2.1}$$

where λ_1 and λ_2 are two penalty parameters, P_C is the penalty function to pursue the homogeneity of coefficients, and P_V is another penalty function to select relevant variables. We discuss the specific forms of P_C and P_V to use in the following content.

2.2.2 The Selection of P_C

We first introduce the form of P_C to impose spatial homogeneity in the regression coefficients. In many spatial applications, regression coefficients at proximate locations are likely to be similar as homogeneity is expected within a small neighboring area. It is therefore desired to consider spatially contiguous clustering configurations such that only adjacent locations are clustered together. A common approach to encode spatial proximity information is to use a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the vertex set $\mathcal{V}$ includes n observations, and the edge set $\mathcal{E}$ collects the edges between observations. In addition, we assume that each $\mathbf{B}_j$, defined as the j^{th} column vector $\mathbf{B}$, has its own underlying spatially clustered patterns. Therefore, P_C consists of individual penalties for each $\mathbf{B}_j$.

To pursue the latent clustered patterns, we incorporate the graph information by using the graph fused Lasso [22], whose form is $P_C(\mathbf{B}) = \sum_{j=1}^{p}\sum_{(i_1,i_2)\in\mathcal{E}}|b_{i_1,j} - b_{i_2,j}|$. For the j^{th} variable and any edge $(i_1, i_2) \in \mathcal{E}$, the graph fused Lasso pursues a sparse solution on the difference between the coefficients of sample i_1 and i_2, i.e., $b_{i_1,j} - b_{i_2,j}$. The zero elements of the estimated $b_{i_1,j} - b_{i_2,j}$ indicate that sample i_1 and i_2 belong to the same cluster for the j^{th} variable, while the non-zero elements of $b_{i_1,j} - b_{i_2,j}$ correspond to a cut set of edges which, if removed from the graph, will partition the vertices into a number of disjoint connected components. When a spatial graph is used, the method naturally imposes spatial contiguity constraints on the clustering configurations and the resulting connected components can be interpreted by practitioners as subregions.

Adaptive weights can further be adopted following the idea of the adaptive Lasso [23, 24] to reduce the estimation bias of the standard Lasso and improve variable selection accuracy, and P_C

becomes $P_C(\mathbf{B}) = \sum_{j=1}^{p} \sum_{(i_1,i_2)\in\mathcal{E}} w_{(i_1,i_2),j}|b_{i_1,j} - b_{i_2,j}|$, where $w_{(i_1,i_2),j}$ is the weight for the edge (i_1, i_2) of the j^{th} variable often determined from an initial estimator.

The alternating direction methods of multipliers (ADMM) [25] and the path following type of algorithms [26, 27] have been developed to solve the graph fused Lasso problems. However, the computation of these algorithms can be expensive with the above penalty $P_C(\mathbf{B}, \mathcal{G})$ for a large general graph. A similar computational issue arises when minimizing our objective function in Eq. (2.1) if we use the same form of $P_C(\mathbf{B}, \mathcal{G})$. When a graph has certain simple structure such as a chain or a tree graph, several previous works [14, 28] show that one can take advantage of specific graph structures to design efficient algorithms to solve the graph fused Lasso problem. This motivates us to consider a similar strategy to replace the original graph in P_C with a simple graph, which includes the proximate pairs of samples that are likely to share similar coefficients to pursue spatial homogeneity.

In particular, we select to use a chain graph in our model for two reasons. First, among the different choices of the simple graph, the chain graph could achieve the fastest computation as evidenced in previous studies [28]. Its main advantage lies in the fact that it provides an order of n samples so that the graph fused Lasso is reduced to a 1-dimensional fused Lasso problem, for which several existing efficient algorithms [29, 30] can be adapted to solve the problem. Second, we will show in Section 2.3 that using the chain graph we described below leads to some nice theoretical properties in terms of parameter estimation, and therefore achieves a good balance between model accuracy and computational efficiency.

Note that we allow each covariate to have its own clustering configuration. Directly applying the fused Lasso method using a common graph for all covariates fails to adapt to the differences in covariate-specific clustering patterns. Therefore, we propose an adaptive learning approach to build a variable-dependent chain graph for each variable. The way to construct this variable-dependent graph is as below: We begin with an initial estimator $\mathbf{B}^{\text{ini}}$ of $\mathbf{B}$, whose selection will be discussed in Section 2.4.4. For each of the j^{th} variable, we follow the approach in [28] to first construct a minimum spanning tree (MST) of $\mathcal{G}$ with $|b^{\text{ini}}_{i_1,j} - b^{\text{ini}}_{i_2,j}|$ as the distance between

$(i_1, i_2) \in \mathcal{E}$ and then find a chain graph $\mathcal{G}_j$ by applying the depth-first searching (DFS) algorithm to the MST with a random root vertex. It is proved that for any arbitrary signal, the graph fused Lasso penalty along with the DFS-induced chain graph never exceeds twice the graph fused Lasso over the original graph [28]. This nice property enables the DFS-induced chain graph as a good candidate to approximately carry the information from the original graph.

Now we give the specific form of P_C below. For the j^{th} variable, its chain graph $\mathcal{G}_j$ defines a direct path of n samples, i.e. $(j_1 \to j_2 \to \cdots \to j_n)$. Then $P_C(\mathbf{B}) = P_C(\mathbf{B}, \mathcal{G}_1, \ldots, \mathcal{G}_p) = \lambda_1 \sum_{j=1}^{p} \sum_{i=1}^{n-1} w_{(j_i, j_{i+1}),j} |b_{j_i,j} - b_{j_{i+1},j}|$, where $w_{(j_i,j_{i+1}),j}$ denotes the adaptive weight for the edge $(j_i, j_{j_{i+1}})$ in $\mathcal{G}_j$, and $b_{j_i,j}$ denotes the $(j_i, j)^{\text{th}}$ entry of $\mathbf{B}$. For the sake of notation simplicity, we use $w_{j,i}$ to represent $w_{(j_i,j_{i+1}),j}$. To write $P_C(\mathbf{B})$ in a matrix form, we build the incidence matrix $\mathbf{H}_j \in \mathbb{R}^{(n-1)\times n}$ for $\mathcal{G}_j$: The $(i, j_i)^{\text{th}}$ entry of $\mathbf{H}_j$ is equal to 1, the $(i, j_{i+1})^{\text{th}}$ entry of $\mathbf{H}_j$ is equal to -1 for $i = 1, 2, \ldots, n-1$, and all the other entries of $\mathbf{H}_j$ are 0. Denote $\mathbf{W}_j \in \mathbb{R}^{(n-1)\times(n-1)}$ as the diagonal matrix with $w_{j,i}$ along the diagonal. Then,

$$P_C(\mathbf{B}) = \lambda_1 \sum_{j=1}^{p} \left\| \mathbf{W}_j \mathbf{H}_j \mathbf{B}_j \right\|_1 . \tag{2.2}$$

2.2.3 The Selection of P_V

We then introduce the form of P_V to impose sparsity regularization to determine the non-zero columns $\mathbf{B}_j$. Note that when a spatial covariate is not a true predictor, the corresponding column of regression coefficients of this covariate becomes a zero vector. By treating the parameters in each $\mathbf{B}_j$ as a group, P_V should help achieve the group selection of parameters and hence identify important spatial covariates to avoid over-fitting in estimation and improve the interpretation of the model. There are many existing methods that focus on group selection (see [31] for a selective review of the literature), and we choose the adaptive group elastic-net [32] for the form of P_V. The adaptive group elastic-net can be seen as a combination of the adaptive group Lasso and the elastic-net. It is well studied that the adaptive group Lasso method can achieve the oracle property for the regression problem with group selection, and the elastic-net can deal with the

collinearity of the variables. Indeed, for many spatial problems such as the U.S. election data we consider in Section 2.6, spatial covariates (e.g., social demographic variables) often exhibit strong collinearity, and hence it is desired to perform variable selection such that strongly correlated spatial covariates are selected or omitted from the model altogether. Specifically, $P_V(\mathbf{B})$ takes the form $\sum_{j=1}^{p} u_j \left\|\mathbf{B}_j\right\|_2 + \frac{\tau}{2}\left\|\mathbf{B}\right\|_2^2$ where u_j is a weight parameter for adaptive learning that is often determined from an initial estimator, and τ is a tuning parameter to adjust the size of the squared ℓ_2 norm penalty in the elastic-net.

2.2.4 Simultaneous Spatial Clustering and Variable Selection

Setting $\tilde{\mathbf{X}} = \left[\text{diag}(\mathbf{X}_1), \ldots, \text{diag}(\mathbf{X}_p)\right] \in \mathbb{R}^{n\times(np)}$ where $\mathbf{X}_j \in \mathbb{R}^n$ is the j^{th} column vector of $\mathbf{X}$, we have $f(\mathbf{B}) = \frac{1}{2n}\left\|\mathbf{Y} - \tilde{\mathbf{X}}\text{vec}(\mathbf{B})\right\|_2^2$. By incorporating the penalties P_C in Eq. (2.1) and P_V in Eq. (2.2), we now introduce the final matrix form of the objective function which we minimize to obtain an estimator of $\mathbf{B}$:

$$L(\mathbf{B}) = \frac{1}{2n}\left\|\mathbf{Y} - \tilde{\mathbf{X}}\text{vec}(\mathbf{B})\right\|_2^2 + \lambda_1 \sum_{j=1}^{p} \left\|\mathbf{W}_j\mathbf{H}_j\mathbf{B}_j\right\|_1 + \lambda_2\left[\sum_{j=1}^{p} u_j \left\|\mathbf{B}_j\right\|_2 + \frac{\tau}{2}\left\|\mathbf{B}\right\|_2^2\right]. \tag{2.3}$$

It is noticeable that $L(\mathbf{B})$ is a convex function of $\mathbf{B}$, which can be efficiently solved by the convex optimization method to be introduced in Section 2.4.

2.3 Theorem

We now give a theoretical analysis of our model. To make a brief overview, we first provide a non-asymptotic bound for the estimating error of our model that depends on the number of truly related variables and edges of graphs connecting samples from different clusters in Theorem 1. Then we provide Corollary 1 that shows the selection consistency of our estimator. Finally, in Theorem 2, we show the relationship between the number of edges connecting samples from different clusters and the number of clusters for the variable-dependent chain graph, which reflects the advantage of using the variable-dependent chain graph proposed in our method.

We first introduce the notations in this section. Use $\mathbf{B}^*$ to denote the true value of $\mathbf{B}$ and $\hat{\mathbf{B}}$ to denote the estimator of $\mathbf{B}$ from Eq. (2.3). We define a series of index sets $\mathcal{J}, \mathcal{I}_1, \cdots, \mathcal{I}_p$ to indicate

the support space of $\mathbf{B}^*$:

$$\mathcal{J} = \{j : \left\|\mathbf{B}_j^*\right\|_2 \neq 0, j = 1, 2, \ldots, p\},$$
$$\mathcal{I}_j = \begin{cases} \{i \in \{1, 2, \ldots, n-1\} : \mathbf{H}_{i,j}\mathbf{B}_j^* \neq 0\} & \text{if } j \in \mathcal{J}, \\ \emptyset & \text{if } j \notin \mathcal{J}, \end{cases} \tag{2.4}$$

where $\mathbf{H}_{i,j}$ denotes the i^{th} row of $\mathbf{H}_j$. Clearly, $\mathcal{J}$ contains the indexes of truly related variables, and $\mathcal{I}_j$ collects the edges that connect the samples with different coefficient values and from different clusters of the j^{th} variables. We further define $\tilde{\mathbf{H}}_j = (\mathbf{H}_j^T, \mathbf{1}_n/n)^T \in \mathbb{R}^{n\times n}$. It is easy to check that $\tilde{\mathbf{H}}_j$ is always a full rank matrix. Let $\tilde{\mathbf{H}}_j^{-1}$ denote the inverse matrix of $\tilde{\mathbf{H}}_j$. We decompose $\tilde{\mathbf{H}}_j^{-1} = (\mathbf{H}_j^-, \mathbf{1}_j^-)$ so that $\mathbf{H}_j^- \in \mathbb{R}^{n\times(n-1)}$ contains the first $n-1$ columns of $\tilde{\mathbf{H}}_j^{-1}$, and $\mathbf{1}_j^- \in \mathbb{R}^n$ represents the last column of $\tilde{\mathbf{H}}_j^{-1}$.

To present the theoretical result of our method, we introduce three assumptions:

Assumption 1. *There exists a constant $\gamma > 0$ such that for any $\mathbf{V} \in \mathcal{C}(\mathbf{B}^*; \lambda_1, \lambda_2, \tau)$,*

$$\frac{1}{n}\left\|\tilde{\mathbf{X}}\text{vec}(\mathbf{V})\right\|_2^2 \geq \gamma \left\|\mathbf{V}\right\|_F^2.$$

where

$$\begin{aligned}\mathcal{C}(\mathbf{B}^*; \lambda_1, \lambda_2, \tau) = \Big\{&\mathbf{V} \in \mathbb{R}^{n\times p} : \\ &\sum_{j\in\mathcal{J}} \lambda_1 \left\|\mathbf{W}_{\mathcal{I}_j^C,j}\mathbf{H}_{\mathcal{I}_j^C,j}\mathbf{V}_j\right\|_1 + \sum_{j\in\mathcal{J}^C}\left[\lambda_1 \left\|\mathbf{W}_j\mathbf{H}_j\mathbf{V}_j\right\|_1 + \lambda_2 u_j \left\|\mathbf{V}_j\right\|_2\right] \\ &< 3\sum_{j\in\mathcal{J}}\left[\lambda_1 \left\|\mathbf{W}_{\mathcal{I}_j,j}\mathbf{H}_{\mathcal{I}_j,j}\mathbf{V}_j\right\|_1 + \lambda_2 u_j \left\|\mathbf{V}_j\right\|_2\right] + \lambda_2\tau\left[-\left\|\mathbf{V}\right\|_F^2 + 2\left\|\mathbf{B}^*\right\|_F \left\|\mathbf{V}\right\|_F\right]\Big\}.\end{aligned}$$

$\mathbf{W}_{\mathcal{I}_j,j}$ denotes the sub-matrix of $\mathbf{W}_j$ containing rows of $\mathbf{W}_j$ corresponding to the indexes in $\mathcal{I}_j$. $\mathbf{H}_{\mathcal{I}_j,j}$, $\mathbf{W}_{\mathcal{I}_j^C,j}$, and $\mathbf{H}_{\mathcal{I}_j^C,j}$ are defined in the same way.

Assumption 2. *$\epsilon_1, \epsilon_2, \ldots, \epsilon_n$ are independent sub-Gaussian random variables such that $P(\epsilon_i > t) < \exp(-t^2/2\sigma_i^2)$.*

Assumption 3. $\|\mathbf{X}\|_\infty < \infty$, $(\max_{j\in\mathcal{J}} \|\mathbf{W}_j^{-1}\|_\infty)(\max_{j\in\mathcal{J}} \left\|\mathbf{W}_{\mathcal{I}_j,j}\right\|_\infty) < \infty$, *and* $(\max_j |u_j^{-1}|)(\max_{j\in\mathcal{J}} |u_j|) < \infty$.

Assumption 1 requires a restricted strong convexity property at $\mathbf{B}^*$. Assumption 2 requires that the distribution of the error term is not too dispersed. Assumption 3 restricts the ℓ_∞ norms of covariates and weights. All assumptions are commonly adopted in high dimensional regression model literature [33, 34].

Theorem 1. *Under Assumption 1, for* $\lambda_1/2 > \max_{j\in\mathcal{J}} \left\|\frac{1}{n}\boldsymbol{\epsilon}^T \text{diag}(\mathbf{X}_j)\mathbf{H}_j^- \mathbf{W}_j^{-1}\right\|_\infty$ *and* $\lambda_2/2\sqrt{n} > \max_{i,j} |\frac{1}{n}\epsilon_i X_{ij} u_j^{-1}|$,

$$\left\|\hat{\mathbf{B}} - \mathbf{B}^*\right\|_F \leq \frac{6(\lambda_1 a_1 \left\|\mathbf{W}_{\mathcal{I}_j,j}\right\|_\infty + \lambda_2 a_2 + \lambda_2\tau \|\mathbf{B}^*\|_F)}{\gamma + \lambda_2\tau}, \tag{2.5}$$

and

$$\frac{1}{2n}\left\|\tilde{\mathbf{X}}\text{vec}(\hat{\mathbf{B}} - \mathbf{B}^*)\right\|_2^2 \leq \frac{18(\lambda_1 a_1 \left\|\mathbf{W}_{\mathcal{I}_j,j}\right\|_\infty + \lambda_2 a_2 + \lambda_2\tau\|\mathbf{B}^*\|_F)^2}{\gamma + \lambda_2\tau}, \tag{2.6}$$

where $a_1 = \sqrt{|\mathcal{J}|}\max_{j\in\mathcal{J}}\sqrt{|\mathcal{I}_j|}$ *and* $a_2 = \sqrt{|\mathcal{J}|}\max_{j\in\mathcal{J}}|u_j|$.

The form of Eq. (2.5) intuitively reveals the role of each penalty. The numerator of the right side of Eq. (2.5) consists of three terms. The first term comes from the estimating error of the cluster patterns. Its form is similar to the error term of the SCC method in [14]. The second term is for the error of variable selection. The third term comes from the squared ℓ_2 penalty in the elastic-net. Notice that both a_1 and a_2 include $|\mathcal{J}|$, which shows that the upper bound of the estimating error depends on the number of truly related variables. a_1 also includes $\max_{j\in\mathcal{J}}\sqrt{|\mathcal{I}_j|}$, which is the max number of edges that connect different clusters among the variable-dependent graphs and is directly related to the number of clusters and the quality of the graphs. We will later show in Theorem 2 that the rate of $\max_{j\in\mathcal{J}}\sqrt{|\mathcal{I}_j|}$ can be as low as $O(R)$ for variable-dependent graphs when n goes to infinity, where R is the max number of clusters for different variables.

Under Assumption 2, we also prove that the smallest rates of λ_1 and λ_2 satisfying the requirements of Theorem 1 are $\lambda_1 = O(C_1\sqrt{\log(n|\mathcal{J}|)/n})$ and $\lambda_2 = O(C_2\sqrt{\log(n)/n})$ as $n \to \infty$, where $C_1 = max_{j\in\mathcal{J}}\|\mathbf{X}_j\|_\infty \max_{j\in\mathcal{J}}\|\mathbf{W}_j^{-1}\|_\infty$ and $C_2 = \max_j \|u_j^{-1}\mathbf{X}_j\|_\infty$.

As a result, with Assumption 3, the order of the upper bound in Eq. (2.5) is $O\big(\sqrt{|\mathcal{J}|\max_{j\in\mathcal{J}}|\mathcal{I}_j|\log(n|\mathcal{J}|)/n + |\mathcal{J}|\log(n)/n + \tau^2\left\|\mathbf{B}^*\right\|_F^2\log(n)/n}\big)$. By simple algebra, if $|\mathcal{J}|\max_{j\in\mathcal{J}}|\mathcal{I}_j| = o(n/\log(n))$ and $\tau = o(\sqrt{n/\log(n)}/\left\|\mathbf{B}^*\right\|_F)$, the upper bounds of the inequalities in Eq. (2.5) and Eq. (2.6) will decrease asymptotically to zero as $n \to \infty$, indicating estimation consistency of the RSVC estimator. Accordingly, we establish Corollary 1 to provide the selection consistency of our estimator.

Corollary 1. *If $|\mathcal{J}|\max_{j\in\mathcal{J}}|\mathcal{I}_j| = o(n/\log(n))$ and $\tau = o(\sqrt{n/\log(n)}/\|\mathbf{B}^*\|_F)$, under the Assumptions 1, 2 and 3, there exists $\delta > 0$ such that (i) $\|\hat{\mathbf{B}}_j\|_2 < \delta \Leftrightarrow j \notin \mathcal{J}$; (ii) $|\mathbf{H}_{i,j}\hat{\mathbf{B}}_j| < \delta \Leftrightarrow i \notin \mathcal{I}_j$, with probability tending to 1 as $n \to \infty$.*

If we further assume that $\|\mathbf{B}^*\|_F = O(\sqrt{n})$, then τ is required to have a rate of $o(1/\sqrt{\log(n)})$ as n increases to achieve a consistent estimator.

The above results do not discuss the effects of adaptive learning, and now we investigate how adaptive learning would improve the convergence rates in Theorem 1 and Corollary 1. First, we consider that $\mathbf{W}_j$ and u_j are built by a consistent estimator of $\mathbf{B}$. If $\mathbf{W}^{-1}_{\mathcal{I}_j^C,j} = O(\sqrt{\log(n)/n})$ for $j \in \mathcal{J}$ for example, the smallest rate of λ_1 satisfying the requirements of Theorem 1 will reduce to $\lambda_1 = O(C_1\sqrt{\log(|\mathcal{J}|\max_{j\in\mathcal{J}}|\mathcal{I}_j|)/n})$ for large n (shown in Section 2.8), and the results of Corollary 1 will be improved accordingly. Meanwhile, if $u_j^{-1} = O(\sqrt{\log(n)/n})$ for $j \in \mathcal{J}^C$, the value of C_2 could be bounded only if the rate of $\|X_j\|_\infty$ is at most $O(\sqrt{n/\log(n)})$, which relaxes the constraint in Assumption 3.

Second, we show the advantage of setting $\mathbf{H}_j$ using the variable-dependent chain graph over the non-adaptive common chain graph. According to Eq. (2.4), the order of $|\mathcal{J}|$ only depends on $\mathbf{B}^*$, whereas the order of $|\mathcal{I}_j|$ depends on both $\mathbf{B}^*$ and the variable-dependent chain graph $\mathbf{H}_j$ constructed via adaptive learning. We build Theorem 2 to show the effects of $\mathbf{H}_j$ on the order of $|\mathcal{I}_j|$.

Theorem 2. *Denote R_j as the true number of clusters for the j^{th} variable. Suppose that each $\mathbf{H}_j$ is constructed by the adaptive learning approach described in Section 2.2.2 with the assumptions: (i) The sub-graph of the observations in the same cluster is connected in $\mathcal{G}$; (ii) $\mathbf{B}^{\text{ini}}$ is a*

consistent estimator of $\mathbf{B}$. *Then for* $\mathcal{I}_j$ *defined by Eq.* (2.4) *with* $\mathbf{H}_j$, $|\mathcal{I}_j| \leq 2(R_j - 1)$, *and hence* $\max_{j \in \mathcal{J}} |\mathcal{I}_j| \leq 2(\max_j R_j - 1)$, *with probability tending to 1 as* $n \to \infty$.

The first assumption of Theorem 2 requires that the true clustering configuration can be induced from $\mathcal{G}$, that is, there exists a set of edges which, if removed, induces the true clusters corresponding to the connected components, and the second assumption is a common assumption in adaptive learning. Theorem 2 shows that as $n \to \infty$, the value of $|\mathcal{I}_j|$ is only controlled by the true number of clusters. In contrast, when a non-adaptive chain graph is used, $|\mathcal{I}_j|$ can grow with n. For example, consider a simple case with a lattice graph in the 2D space and a vertical split in the middle. The rate of $|\mathcal{I}_j|$ becomes $O(\sqrt{n})$, which leads to a larger error bound in (2.5) than that of the adaptive method.

2.4 Algorithm

We propose to minimize the convex objective function in Eq. (2.3) using the proximal gradient method, taking advantage of the fact that the proximal operators of P_C and P_V in our method can be solved efficiently.

2.4.1 Proximal Gradient Method

The proximal gradient method is a particular case of the Majorization-Minimization (MM) algorithm [35]. In the MM algorithm, a majorizing function of the objective function is constructed first. Then the parameters are updated iteratively by minimizing the majorizing function in each step.

Firstly, with a fixed point $\mathbf{B}^t$, we can bound $L(\mathbf{B})$ by:

$$L(\mathbf{B}) \leq f(\mathbf{B}^t) + < \mathbf{B} - \mathbf{B}^t, \nabla f(\mathbf{B}^t) > + \frac{l}{2} \left\| \mathbf{B} - \mathbf{B}^t \right\|_F^2 + P_{C+V}(\mathbf{B}),$$

where $P_{C+V}(\mathbf{B}) = P_C(\mathbf{B}) + P_V(\mathbf{B})$, $\nabla f(\mathbf{B}^t)$ is the gradient of $f(\mathbf{B})$ at $\mathbf{B}^t$, $< \cdot, \cdot >$ denotes an inner product operator, and l is the Lipschitz constant of $\nabla f(\mathbf{B})$ that $|\nabla f(\mathbf{B}^\flat) - \nabla f(\mathbf{B}^\sharp)| \leq l \left\| \mathbf{B}^\flat - \mathbf{B}^\sharp \right\|_F$ for any $\mathbf{B}^\flat, \mathbf{B}^\sharp \in \mathbb{R}^{n \times p}$. We can choose l as two times of the largest eigenvalue of

$\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}/n$ [36].

Secondly, the proximal gradient method updates the value of $\mathbf{B}$ iteratively with the previous point $\mathbf{B}^t$ by solving:

$$\mathbf{B}^{t+1} = \arg\min_{\mathbf{B}} \frac{1}{2} \|\mathbf{B} - \mathbf{Z}\|_F^2 + \frac{1}{l} P_{C+V}(\mathbf{B}), \tag{2.7}$$

where $\mathbf{Z} = \mathbf{B}^t - (1/l)\nabla f(\mathbf{B}^t)$. Notice that the optimization problem in Eq. (2.7) is separable into each column of $\mathbf{B}$. With simple algebra, solving Eq. (2.7) is equivalent to solving p small optimization problems as below, for $j = 1, 2, \ldots, p$:

$$\mathbf{B}_j^{t+1} = \arg\min_{\mathbf{B}_j} \frac{1}{2} \left\| \mathbf{B}_j - \frac{1}{1 + \tau\lambda_2/l} \mathbf{Z}_j \right\|_2^2 + \frac{\lambda_1}{l + \tau\lambda_2} \|\mathbf{W}_j \mathbf{H}_j \mathbf{B}_j\|_1 + \frac{\lambda_2 u_j}{l + \tau\lambda_2} \|\mathbf{B}_j\|_2 , \tag{2.8}$$

where $\mathbf{Z}_j$ denotes the j^{th} column of $\mathbf{Z}$. Eq. (2.8) belongs to the proximal operator problem associated with the combination of the adaptive fused Lasso and the group Lasso.

Later in Section 2.4.2, we will show that the smallest computational complexity to solve each small problem in Eq. (2.8) is $O(n)$, and thus the overall computational complexity of Eq. (2.7) is $O(np)$. Furthermore, [36] proves that the number of iterations of the proximal gradient method can be reduced from $O(1/\epsilon)$ to $O(1/\sqrt{\epsilon})$ when replacing the fixed value $\mathbf{B}^t$ in $\mathbf{Z}$ by:

$$\mathbf{A}^t = \mathbf{B}^t + \frac{\alpha^{t-1} - 1}{\alpha^t} (\mathbf{B}^t - \mathbf{B}^{t-1}), \tag{2.9}$$

where ϵ is the convergence tolerance, and $\alpha^t = 1 + \sqrt{1 + 4(\alpha^{t-1})^2}/2$. To conclude, the overall computational complexity of our algorithm is $O(np/\sqrt{\epsilon})$, which is nearly scalable to large datasets. We summarize the details of the proximal gradient method in Algorithm 1.

2.4.2 Solving the Proximal Operator

In this subsection, we discuss the way to solve Eq. (2.8) efficiently. We first define the standard form of the proximal operator associated with the adaptive fused Lasso and the group Lasso as follows: For any vector $z \in R^n$, the proximal operator associated with the adaptive fused Lasso

Algorithm 1 Algorithm of the Proximal Gradient Method

Input: $\mathbf{Y}$,$\mathbf{X}$, r, λ_1, λ_2, γ, l, ϵ.
for $t = 0, 1, 2, \ldots$: **do**
  Calculate $\alpha^t = 1 + \sqrt{1 + 4(\alpha^{t-1})^2}/2$;
  Construct $\mathbf{A}^t$ by Eq. (2.9);
  Calculate $\mathbf{Z} = \mathbf{A}^t - (1/l)\nabla f(\mathbf{A}^t)$;
  Get $\mathbf{B}^{t+1}$ by solving Eq. (2.7);
end for
Stopping criterion: Keep the above iteration until t satisfies $L(\mathbf{B}^t) - L(\mathbf{B}^{t+1}) < \epsilon$
Output: $\mathbf{B}^{t+1}$.

and the group Lasso is to solve:

$$prox_{FL+GL}(\mathbf{z}, \lambda_1, \lambda_2) = \arg\min_{\boldsymbol{\beta}} \|\mathbf{z} - \boldsymbol{\beta}\|_2^2 + \lambda_1 \|\mathbf{WH}\boldsymbol{\beta}\|_1 + \lambda_2 \|\boldsymbol{\beta}\|_2 . \tag{2.10}$$

where $\mathbf{H} \in (n-1) \times n$ is a matrix whose $(i,i)^{\text{th}}$ entry is 1 and $(i, i+1)^{\text{th}}$ entry is -1 for $i = 1, 2, \ldots, n-1$. The other entries of $\mathbf{H}$ are all 0. $\mathbf{W}$ is a diagonal matrix containing the adaptive weights. It is easy to check that Eq. (2.8) is a specific case of Eq. (2.10) by reordering the rows of $\mathbf{W}_j\mathbf{H}_j\mathbf{B}_j$ and choosing particular values for $\mathbf{z}$, $\boldsymbol{\beta}$, λ_1, and λ_2. To efficiently solve Eq. (2.10), we propose the following theorem:

Theorem 3. *Define:*

$$prox_{FL}(\mathbf{z}, \lambda_1) = \arg\min_{\boldsymbol{\beta}} \|\mathbf{z} - \boldsymbol{\beta}\|_2^2 + \lambda_1 \|\mathbf{WH}\boldsymbol{\beta}\|_1$$

$$prox_{GL}(\mathbf{z}, \lambda_2) = \arg\min_{\boldsymbol{\beta}} \|\mathbf{z} - \boldsymbol{\beta}\|_2^2 + \lambda_2 \|\boldsymbol{\beta}\|_2 .$$

$prox_{FL}(\mathbf{z}, \lambda_1)$, $prox_{GL}(\mathbf{z}, \lambda_1)$, *and* $prox_{FL+GL}(\mathbf{z}, \lambda_1, \lambda_2)$ *have the following relation:*

$$prox_{FL+GL}(\mathbf{z}, \lambda_1) = prox_{GL}\big(prox_{FL}(\mathbf{z}, \lambda_1), \lambda_2\big).$$

Theorem 3 can be proved by following the proof of Theorem 1 in [30]. The only difference between their Theorem 1 and our Theorem 3 is that we include an additional weight matrix $\mathbf{W}$

into $prox_{FL+GL}(\mathbf{z}, \lambda_1, \lambda_2)$. Theorem 3 shows that Eq. (2.10) can be computed through a two-step strategy which involves $prox_{FL}(\mathbf{z}, \lambda_1)$ and $prox_{GL}(\mathbf{z}, \lambda_2)$ respectively. For $prox_{GL}(\mathbf{z}, \lambda_2)$, its analytical solution is:

$$prox_{GL}(\mathbf{z}, \lambda_2) = \left\{ \frac{\max(0, \|\mathbf{z}\|_2 - \lambda_2)}{\|\mathbf{z}\|_2} \right\} \mathbf{z}.$$

For $prox_{FL}(\mathbf{z}, \lambda_1)$, several efficient algorithms with $O(n)$ computational complexity have been developed based on the ideas of taut string and dynamic programming [37, 29]. [38] develops a fast algorithm via the analysis of KKT conditions, which is easy to implement and only requires $n + 7$ storage. Precisely, if m is the number of nonzero entries in $\mathbf{H} \cdot prox_{FL}(\mathbf{z}, \lambda_1)$, the computational complexity of Condat's method is $O(nm)$. We decide to use the weight-included version of Condat's method for its easy implementation.

2.4.3 Tuning Parameters

There are three tuning parameters in our model: the penalty parameter λ_1 for seeking homogeneity patterns, the penalty parameter λ_2 for variable selection, and the allocation parameter τ in the elastic-net penalty. To make a proper choice of their values, we use the generalized information criterion (GIC). GIC is proposed by [39] to select the tuning parameters in penalized generalized linear models. Different from the traditional information criteria such as Bayes information criterion (BIC), GIC is applicable to the case when the number of parameters increases at most exponentially with the sample size n. Our model considers the case that includes np parameters in total with $p = O(n^c)$ for some $c \geq 0$, and we can calculate its GIC by:

$$\text{GIC}(\lambda_1, \lambda_2, \tau) = \log \left[\frac{1}{n} \sum_{i=1}^{n} (y_i - \mathbf{x}_i \mathbf{b}_i)^2 \right] + \frac{1}{n} \log \{\log(n)\} \log(n) \cdot \text{df}(\lambda_1, \lambda_2, \tau), \qquad (2.11)$$

where $\text{df}(\lambda_1, \lambda_2, \tau)$ is the degree of freedom of the model with tuning parameters $(\lambda_1, \lambda_2, \tau)$. We estimate $\text{df}(\lambda_1, \lambda_2, \tau)$ by $|\hat{\mathcal{J}}| + \sum_{j=1}^{p} |\hat{\mathcal{I}}_j|$ when $\tau = 0$, where $\hat{\mathcal{J}}, \hat{\mathcal{I}}_1, \ldots, \hat{\mathcal{I}}_p$ indicate the support space of the estimator $\hat{\mathbf{B}}$. When $\tau = 0$, the way to estimate $\text{df}(\lambda_1, \lambda_2, \tau)$ is similar to elastic-net [40]. The Theorem 3 by [39] shows that the GIC defined in Eq. (2.11) can consistently identify

the true model.

In practice, assuming that there are m_1, m_2, and m_3 candidate values for λ_1, λ_2, and τ respectively, we need to build $m_1 m_2 m_3$ models in total to decide the best combination to use. Instead of building each model separately, we adopt the strategy of warm starts [34] to reduce the computation time. We first sort the candidate values for λ_1 and λ_2 in a decreasing order respectively and form two sequences $\{\lambda_{1,i}\}_{i=1}^{m_1}$ and $\{\lambda_{2,j}\}_{j=1}^{m_2}$. Then for each fixed τ, we build $m_1 m_2$ models with $\{\lambda_{1,i}\}_{i=1}^{m_1}$ and $\{\lambda_{2,j}\}_{j=1}^{m_2}$ sequentially. Use $\hat{\mathbf{B}}(\lambda_1, \lambda_2, \tau)$ to denote the solution of the model with $(\lambda_1, \lambda_2, \tau)$. We begin by estimating $\hat{\mathbf{B}}(\lambda_{1,1}, \lambda_{2,1}, \tau)$, and then update $\hat{\mathbf{B}}(\lambda_{1,i}, \lambda_{2,1}, \tau)$ using $\hat{\mathbf{B}}(\lambda_{1,i-1}, \lambda_{2,1}, \tau)$ as the initial value, for $i = 2, \ldots, m_1$. This greatly reduces the number of iterations in the algorithm. Similarly, for $i = 1, \ldots, m_1$ and $j = 2, \ldots, m_2$, $\hat{\mathbf{B}}(\lambda_{1,i}, \lambda_{2,j-1}, \tau)$ is used as a warm start for $\hat{\mathbf{B}}(\lambda_{1,i}, \lambda_{2,j}, \tau)$.

2.4.4 Selections of Weights

u_i and $\mathbf{W}_j$ in Eq. (2.3) are important weight parameters in our adaptive methods. As we have discussed in Section 2.3, appropriate choices of these weights can lead to an improved convergence rate over its non-adaptive counterpart.

Assuming that $\mathbf{B}^{\text{ini}}$ is an initial estimator, u_i and $\mathbf{W}_j$ could be determined by:

$$u_j = (\|\mathbf{B}_j^{\text{ini}}\|_2)^{-\alpha} \text{ and } \mathbf{W}_j = \text{diag}[(\mathbf{H}_j \mathbf{B}_j^{\text{ini}})^{-\alpha}].$$

α is a tuning parameter that is usually set to 1 or 2. The traditional selection of $\mathbf{B}^{\text{ini}}$ is $\mathbf{B}$'s ordinary least square (OLS) estimator, whereas, in our problem, the number of parameters is larger than the number of observations, and the OLS estimator is not unique. Instead, we choose to use Eq. (2.3) with $u_i = 1$ and $\mathbf{W}_j = \mathbf{I}_{n-1}$ to get the initial estimator that we prove is consistent in theory.

2.5 Simulation

In this section, we design simulation experiments to evaluate the performance of our model. We randomly generate 1000 spatial locations $s_1, \ldots, s_{1000}$ from the square domain $[0, 1] \times [0, 1]$. We then generate $p \in \{20, 100\}$ variables that are not only spatially dependent within each variable

but also dependent across variables to mimic many real-world spatial processes. Specifically, the design matrix of p covariates for the simulation is generated from $\mathbf{X} = \Gamma\mathbf{Z}$, where $\Gamma\Gamma^T = \Sigma$ with $\Sigma_{ij} = 0.5^{|i-j|}$, and $\mathbf{Z} = (z_{ij})_{1000\times p}$ includes p independent realizations generated at 1000 locations from a Gaussian Process $GP(0, C)$ with covariance function $C(z_{i_1 j}, z_{i_2 j}) = \exp(-||s_{i_1} - s_{i_2}||/0.3)$. Among p covariates, we randomly select 3 successive variables with index j_1, j_2 and j_3 as the truly related variables and calculate the response value at the i^{th} location by:

$$y_i = \beta_{ij_1} x_{ij_1} + \beta_{ij_2} x_{ij_2} + \beta_{ij_3} x_{ij_3} + \epsilon_i, \tag{2.12}$$

where ϵ_i is the i.i.d error from $\mathcal{N}(0, 0.25^2)$. The true values of β_{ij_1}, β_{ij_2}, and β_{ij_3} for each location are set with different cluster patterns as shown in the first row of Figure 2.1.

For model comparison, to the best of our knowledge, there is no existing method specifically designed for simultaneous variable selection and spatial clustering as discussed in Section 2.1. Therefore, we choose to compare our model with the two-step spatially clustered coefficient method (TS-SCC). In TS-SCC, we first use Lasso [3] to select important variables in the constant-coefficient model. Then we use the SCC method by [14] to estimate the clustered coefficients. Two oracle methods, denoted as $\text{Oracle}_{\text{VC}}$ and $\text{Oracle}_{\text{ALL}}$, are also included in the comparisons as benchmarks. $\text{Oracle}_{\text{VC}}$ builds the SCC model assuming the truly related variables are known. $\text{Oracle}_{\text{ALL}}$ assumes both the truly related covariates and the true cluster patterns of them are known and uses the original least square method to estimate the coefficients in each cluster. Since our method is an adaptive method, to make a fair comparison, the SCC method is modified to the adaptive version. We use a $K = 5$ nearest neighborhood network built on the Euclidean distance among locations as the original graph $\mathcal{G}$ for all methods. If the constructed network is not connected, we gradually increase K until we get a connected network.

To evaluate the performance of different methods, we randomly select 5% of the 1000 samples as the test dataset. The remaining samples are used as the training dataset. For each method, after building the model on the training dataset, we calculate: (1) the number of true-positive discoveries

Table 2.1: Mean performance (standard deviation) of different methods for variable selection and coefficient estimation in 100 simulations with the variable size $p = 20$ or $p = 100$.

Metric	p = 20				p = 100			
	RSVC	TS-SCC	$\text{Oracle}_{\text{VC}}$	$\text{Oracle}_{\text{ALL}}$	RSVC	TS-SCC	$\text{Oracle}_{\text{VC}}$	$\text{Oracle}_{\text{ALL}}$
TP	**2.93**	2.05	3.00	3.00	**2.67**	1.34	3.00	3.00
	(0.33)	(0.89)	(0.00)	(0.00)	(0.53)	(0.89)	(0.00)	(0.00)
FP	**1.15**	11.74	0.00	0.00	**1.02**	26.36	0.00	0.00
	(1.49)	(3.04)	(0.00)	(0.00)	(2.09)	(8.74)	(0.00)	(0.00)
EE	**0.19**	0.29	0.18	0.01	**0.11**	0.16	0.08	0.01
	(0.04)	(0.07)	(0.03)	(0.01)	(0.01)	(0.03)	(0.01)	(0.01)
PE	**0.81**	1.00	0.80	0.57	**1.01**	1.12	0.80	0.57
	(0.23)	(0.26)	(0.27)	(0.24)	(0.22)	(0.23)	(0.24)	(0.21)

(TP); (2) the number of false-positive discoveries (FP); (3) the estimation error between $\hat{\mathbf{B}}$ and $\mathbf{B}$ (EE); (4) the prediction error in the test dataset (PE):

$$\text{EE} = \sqrt{\frac{\left\|\mathbf{B} - \hat{\mathbf{B}}\right\|_F}{np}}, \quad \text{PE} = \sqrt{\frac{\sum_{i \in \text{TestSet}} (\hat{y}_i - y_i)^2}{n_{\text{test}}}},$$

for model comparisons. To predict y_i in the test dataset, we first find its $k = 5$ nearest locations in the training dataset and use the mean of these locations' coefficients to predict the coefficients at the i^{th} location. We repeat the simulation experiments 100 times and report the mean and standard deviation of each evaluation metric.

Table 2.1 includes the simulation results. We observe that RSVC achieves better results of all metrics than those of TS-SCC in both choices of p. RSVC detects three true variables at most times and has a low average FP, which demonstrates its superior performance in variable selection. In contrast, the TP of TS-SCC is 2.05 for $p = 20$ and 1.34 for $p = 100$, which means that TS-SCC fails to identify 0.95 and 1.66 important variables on average. TS-SCC also has a large FP, especially when p is large. For prediction error, both the EE and PE of RSVC are close to those of $\text{Oracle}_{\text{VC}}$, which shows that RSVC provides accurate estimations for the coefficients and predictions.

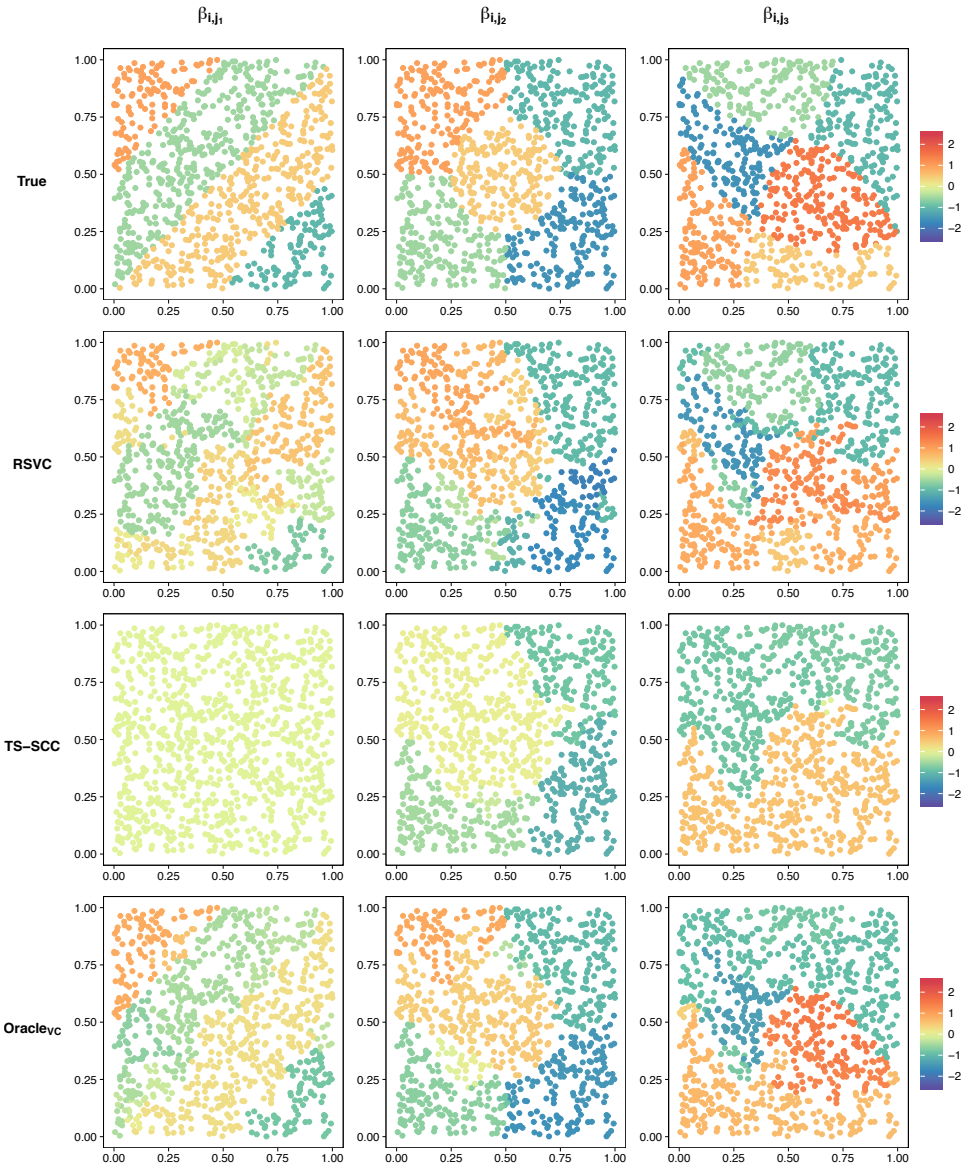


Figure 2.1: Estimated coefficients of different methods in one simulation with $n = 1000$ and $p = 20$.

Figure 2.1 gives an example of the estimated coefficients for different methods with $p = 20$. We can see that RSVC recovers the original patterns of the coefficients well in all three truly related variables. On the contrary, TS-SCC fails to select the first related variable. Although the second and the third variables are correctly selected, several clusters are not correctly identified by TS-SCC, indicating that including many FP variables in the model will weaken the accuracy of the estimation on the coefficients of truly related variables. From Figure 2.1, we also observe that RSVC tends to provide more clusters than the true patterns and $\text{Oracle}_{\text{VC}}$ method for β_{i,j_1}, which is due to the usage of chain graphs to cluster the coefficients. As discussed in Section 2.2.2, although the use of chain graphs helps to reduce the computational complexity, it tends to cut a true cluster into several small clusters. The reason for this phenomenon is that chain graphs add looser constraints on the coefficients than the original graph. One remedy in practice is to run SCC again with tree or denser graphs on the selected variables from RSVC to improve the clustering result. To conclude, the simulation results demonstrate the overall good performance of RSVC in terms of variable selection and parameter estimation. We have also investigated the performance of RSVC when the true coefficients vary smoothly across the locations. The results show that RSVC also performs well in those studies, indicating its robustness under model mis-specification.

2.6 County-Level Voting in U.S. Presidential Elections

We now use our model to analyze the voting results of the U.S. presidential elections in 2016 and 2020 for all 3075 counties of the conterminous United States – that is, we exclude Alaska, Hawaii, and all other islands. Since the vast majority of votes in both elections were for the Democratic or Republican candidate, we focus on the percentage of the Democratic votes (denoted by PerDem) in the two-party votes exclusively. Using our RSVC method, we are able to simultaneously select the subset of variables that are significant determinants of PerDem, find spatially varying relationships between them, and identify potential spatially clustered patterns among counties.

For both elections, we consider 37 county-level demographic and economic variables that potentially explain the voting results, which includes the variables most commonly used in the pre-

vious studies (e.g., unemployment rate, per capita income). Table 2.2 includes the description of these variables. The variables are collected from the American Community Survey 5-Year Data (ACS5). For the 2016 election, we use the data from 2015 ACS5. For the 2020 election, since the data of 2019 ACS5 is not available currently, we use the data of 2018 ACS5. In data pre-processing, we first take the logarithm transformation on Population, HouseUnits, and MedianHouseValue variables since each of their distributions has a skewness larger than 3. Next, we scale each variable by calculating

$$X_{\text{scale}} = \frac{X - \text{quantile}(X, 0.05)}{\text{sd}(X)}.$$

In this way, the intercept term of our regression model reflects the voting results of elections for the counties when the value of each covariate is setting to its 0.05 quantile of the population.

We analyze the effects of these variables on the voting results using the RSVC and TS-SCC models discussed above. In addition to spatially varying coefficients on the covariates, we also allow the intercept to vary across space in each of the models to capture the spatial dependence that is unexplained by other covariates.

To evaluate the performance of methods, we randomly split the counties into the training dataset with 95% counties and the test dataset with 5% counties. The model is first built on the training dataset and learns the related variables and spatial clusters. Then for each county in the test set, we find its $k = 5$ nearest counties in the training dataset and use the mean of these counties' coefficients to predict its coefficients. Finally, we can predict the voting of this county with the estimated coefficients. Table 2.3 includes the average performance of the prediction error (PE) and the number of selected variables (#Selected Variables) for RSVC and TS-SCC in 100 times of random splitting. In both the 2016 and 2020 election samples, RSVC has a smaller PE in the test dataset than TS-SCC. This is true despite the fact that the number of variables selected by RSVC is much smaller than that of TS-SCC. As such, the RSVC model achieves superior predictive performance with a much more parsimonious model.

Table 2.2: Description of 37 potential explanatory variables.

Id	Group	Variables	Explanation
1	Population	Population	Population
	Gender	(Baseline) Gender.Male	Male (%)
2		Gender.Female	Female (%)
	Race	(Baseline) Race.Black	Black or African American alone (%)
3		Race.White	White (%)
4		Race.AmIndian	American Indian and Alaska Native alone (%)
5		Race.Asian	Asian (%)
6		Race.Hawaiian	Native Hawaiian and Other Pacific Islander alone (%)
7		Race.OtherRace	Some other race alone (%)
8		Race.Multirace	Two or more races (%)
9		Race.Hispanic	Hispanic or Latino (%)
	Education	(Baseline) Edu.LessHighSchool	Less than high school graduate (%)
10		Edu.HighSchool	High school graduate (includes equivalency) (%)
11		Edu.College	Some college or associate's degree (%)
12		Edu.Bachelor	Bachelor's degree (%)
13		Edu.Graduate	Graduate or professional degree (%)
14	Foreign	NonCitizen	Not a U.S. citizen (%)
15		SpeakForeign	People speaking other language (%)
16	Income	NoIncome	No income population (%)
17		IncomePerCapita	Per capita income in the past 12 months
18		GiniIndex	Gini Index
	Poverty	(Baseline) Poverty.Below1	Below 100 percent of the poverty level (%)
19		Poverty.Between1and1.5	between 100 and 149 percent of the poverty level (%)
20		Poverty.Above1.5	At or above 150 percent of the poverty level (%)
	Mobility	(Baseline) Mob.SameHouse	Live in same house 1 year ago (%)
21		Mob.SameCounty	Moved within same county (%)
22		Mob.SameState	Moved from different county within same state (%)
23		Mob.DiffState	Moved from different state (%)
24		Mob.Abroad	Moved from abroad (%)
25	Transportation	Trans.Car	Transportation to work by car, truck, or van (%)
26		Trans.Public	Transportation to work by public transportation (excluding taxicab) (%)
27		Trans.Bicycle	Transportation to work by bicycle (%)
28	Household	NoEarningHousehold	No earnings households (%)
29		SalaryHousehold	Households with wage or salary income (%)
30	Employment	TotalLaborForce	Labor force (%)
31		Unemployment	Unemployed rate (%)
32		WorkHours	Mean usual hours worked in the past 12 months for workers from 16 to 64 years
33		Employ.MedianAge	Median age for workers
34	House	HouseUnits	House units per person
35		MedianHouseValue	Median house value (dollars)
36	Others	Veteran	Veteran (%)
37		Insurance	One or more health insurance items allocated (%)

Table 2.3: Mean performance (standard deviation) of RSVC and TS-SCC for variable selection and coefficient estimation for the voting results of the 2016 and 2020 elections in 100 times of random splitting.

Metric	2016		2020	
	RSVC	TS-SCC	RSVC	TS-SCC
#Selected Variables	25.83	34.80	23.99	35.56
	(6.81)	(1.03)	(5.88)	(0.59)
PE	0.0630	0.0674	0.0581	0.0630
	(0.0073)	(0.0052)	(0.0056)	(0.0053)

To allow for one-to-one comparisons of the estimated coefficients from 2016 to 2020, we build another SCC model for each election using the union set of the variables selected from two elections in the following way: We first build a RSVC model for each method with all counties and find that 19 variables are selected for the 2016 election, and 20 variables are selected for the 2020 election. Among them, 16 of the variables are selected in both election samples. Then, there are 23 variables plus the intercept term included in the union set of the selected variables of two RSVC models. Finally, a SCC model is built on this union set for each election. Table 2.4 summarizes the range of coefficients of different variables for two election samples. The values of coefficients are adjusted back to the original data scales. There are several noticeable patterns in the results. First, 10 of the 23 variables have spatially varying coefficient effects in both elections. This implies that models restricting these covariates to a single common parameter are mis-specified. Second, we can see that for many of the variables, the range of the varying coefficients does not change much between the two elections. For example, the coefficient of Race.White is between -0.81 and -0.5 for the 2016 election and is between -0.83 and -0.51 for the 2020 election. However, the coefficients for some variables have a noticeable shift. For instance, the coefficient for SpeakForeign in the 2020 election is much smaller than in the 2016 election. This indicates that percentage of foreign language speakers in a county was less influential in determining vote share in 2020 than in 2016, and is consistent with President county-level results showing President Trump somewhat surprisingly winning higher vote shares in 2020 than 2016 in places like Miami-Dade county in

Table 2.4: Range of estimated coefficients in the SCC model of 2016 and 2020 elections with 23 variables selected by RSVC.

Variable	2016	2020	Variable	2016	2020
Intercept	40.84	40.97	Edu.Graduate	1.45 $\sim$ 1.49	1.50 $\sim$ 1.69
Ln_Population	1.45 $\sim$ 1.91	1.38 $\sim$ 1.99	SpeakForeign	1.29 $\sim$ 2.43	0.63
Female	0.07	0.11	GiniIndex	-0.10 $\sim$ 0.06	-0.18 $\sim$ -0.03
Race.White	-0.81 $\sim$ -0.50	-0.83 $\sim$ -0.51	Poverty.Above1.5	-0.23 $\sim$ -0.21	-0.28 $\sim$ -0.23
Race.AmIndian	-0.11	-0.14	Mob.SameCounty	-0.34 $\sim$ -0.16	-0.18 $\sim$ -0.14
Race.Hawaiian	1.28	1.41	Trans.Public	0.26	-0.16
Race.OtherRace	-0.93	-0.88 $\sim$ -0.81	Trans.Bicycle	3.10	3.92
Race.Multirace	-0.80 $\sim$ -0.63	-0.71 $\sim$ -0.65	SalaryHousehold	-0.29 $\sim$ -0.27	-0.29 $\sim$ -0.15
Race.Hispanic	0.51 $\sim$ 0.54	0.47	TotalLaborForce	0.65 $\sim$ 0.78	0.67
Edu.HighSchool	0.33 $\sim$ 0.45	0.28	Unemployment	0.26	0.11
Edu.College	0.24 $\sim$ 0.53	0.36 $\sim$ 0.47	WorkHours	-0.02 $\sim$ -0.01	-0.01
Edu.Bachelor	0.58 $\sim$ 0.83	0.58 $\sim$ 1.07	Ln_HouseUnits	-0.24	1.84

south Florida and the Rio Grande Valley in Texas, both of which have high concentrations of Spanish-speaking citizens.

To better represent the spatially clustered patterns of these covariate effects, we plot the results for several variables in Figure 2.2. (The counties rendered in white have missing values in the covariates and are excluded from our study.) In the first two rows of Figure 2.2, we see how the effects of Race.White and GiniIndex (a measure of income inequality) vary over space in 2016 and 2020. In both elections, an increase in the percentage of white residents is negatively correlated with Democratic across the country; however, these effects are stronger in the South. A similar pattern holds for GiniIndex, with inequality having an larger negative effect in the southern United States in both elections. Interestingly, the range of the coefficient in 2016 crosses zero, with all non-Southern areas indicating a small positive effect. As a consequence, researchers using mean coefficients (such as in ordinary least squares) would risk both neglecting this heterogeneity and falsely determining an effect that is not truly zero to be zero.

While the spatial patterns for Race.White and GiniIndex are relatively stable over time in the 2016 and 2020 elections, there are covariates that exhibit different patterns across the two elections. For example, in the third row of Figure 2.2, we plot the effects of Poverty.Above1.5 – the percent

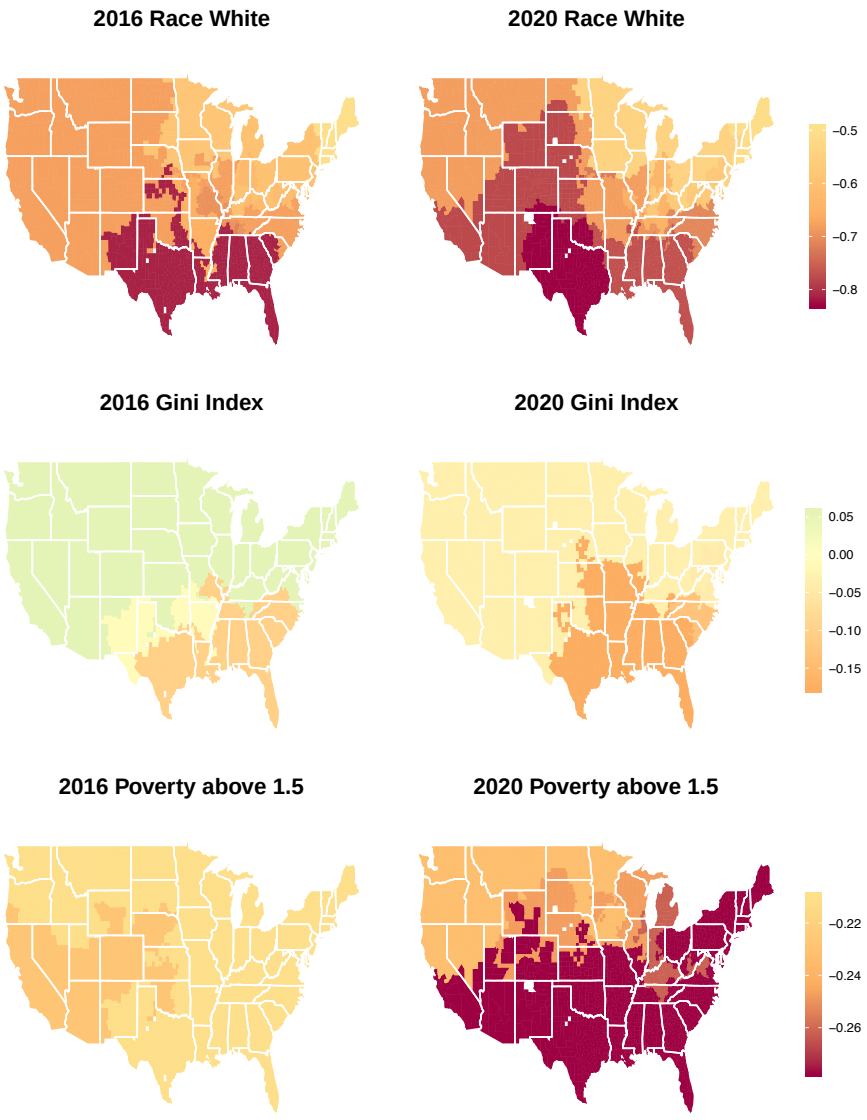


Figure 2.2: Estimated coefficients for Race.White, GiniIndex, and Poverty.Above.1.5 in the 2016 and 2020 elections.

of population that are at or above 150 percent of the poverty level – and observe a noticeable difference between 2016 and 2020. In 2016, the effect of Poverty.Above1.5 was negative (i.e., predicted greater vote share for the Republican candidate) and relatively stable, with only a slightly greater effect in parts of the Southwest and West Coast. By the 2020 election, however, both the magnitude of the effect was greater, and there was also substantially more variation. Specifically, this measure of relative wealth was more likely to predict Republican support in counties in the South and the Northeast. Cross-election differences in these coefficient effects may indicate areas for further research, as researchers may be interested in better understanding why some characteristics were more or less influential in certain elections.

2.7 Discussion

This chapter proposes a new spatially varying coefficient regression model, called the RSVC model, to select important variables and estimate the spatially clustered coefficients simultaneously. By applying RSVC to the the county-level voting data of 2016 and 2020 U.S. presidential elections, we find that the cluster patterns are stable for some covariates across elections and can also differ meaningfully for other covariates.

Moving forward, our RSVC model could be further refined in several ways. First, RSVC incorporates the spatial information by using chain graphs derived from the original graph, which contributes to its fast computation but sacrifices some statistical efficiency due to the loss of information from the original graph. Second, the regularization term of RSVC puts constraints on the size of the coefficients, and hence the results can be sensitive to the way of standardization of covariates. Third, the RSVC estimator does not come with an uncertainty measure that makes it hard for statistical inference, a common issue shared by regularization based approaches. We will investigate these research topics in future work.

RSVC could also be applied and extended to many other statistical contexts and practical problems. One could also easily extend RSVC to the variable selection problems in non-Gaussian outcomes, and multivariate regression models with spatially clustered coefficients. The algorithm and theoretical results for these extensions can be derived following a similar way as in this chap-

ter. Moreover, the regularization term in RSVC has an additive form and is adaptable to additional restrictions on the structure of varying coefficients. For example, in the analysis of U.S. presidential election, if a subset of covariates are assumed to only affect a group of states, that is, a common sparsity structure is assumed for all counties within each state, we could add the additional group sparsity penalty to the regularization term to pursue the selection of states.

Finally, RSVC only requires a graph to incorporate the relational information among observations, and thus its usage is not restricted to the spatial problem. For example, in social media systems, companies seek to predict the individuals' use intention of an APP such as the mobile payment system [41], wheres the behaviors of different groups of users are different. Affiliated by the social network constructed via the social interactions of users, RSVC could be used to cluster users across the social network and select the important covariates for prediction. Another example is to study the functional relationship between genes in single-cell RNA sequencing data. Cells included in the dataset are always heterogeneous and belong to different cell types. For different cell types, the relationship between the target gene and other genes varies a lot [42]. Fortunately, since the process of cell differentiation is continuous, the gene expression levels of different cells usually locate on a continuous low-dimensional manifold in high dimensional space, and cells with the same subtype are near on the manifold. We can construct a neighborhood graph of cells on this manifold to reflect the local structure of cells, and then use RSVC to find cell clusters and the relevant gene to the target gene simultaneously.

2.8 Proof

2.8.1 Proof of Theorem 1

Proof. Use $\mathcal{S} = (\mathcal{J}; \mathcal{I}_1, \cdots, \mathcal{I}_p)$ to denote a collection of the index sets that represents a support space of $\mathbf{B}$. We give a decomposition of $P_{C+V}(\mathbf{B})$ based on $\mathcal{S}$:

$$
\begin{aligned}
P_{C+V}(\mathbf{B}) =& P_{\mathcal{S}}(\mathbf{B}) + P_{\mathcal{S}^C}(\mathbf{B}) + \frac{\lambda_2 \tau}{2} \|\mathbf{B}\|_2^2 \\
P_{\mathcal{S}}(\mathbf{B}) =& \sum_{j \in \mathcal{J}} \left[\lambda_1 \left\| \mathbf{W}_{\mathcal{I}_j, j} \mathbf{H}_{\mathcal{I}_j, j} \mathbf{B}_j \right\|_1 + \lambda_2 u_j \|\mathbf{B}_j\|_2 \right] \\
P_{\mathcal{S}^C}(\mathbf{B}) =& \sum_{j \in \mathcal{J}} \lambda_1 \left\| \mathbf{W}_{\mathcal{I}_j^C, j} \mathbf{H}_{\mathcal{I}_j^C, j} \mathbf{B}_j \right\|_1 + \sum_{j \in \mathcal{J}^C} \left[\lambda_1 \|\mathbf{W}_j \mathbf{H}_j \mathbf{B}_j\|_1 + \lambda_2 u_j \|\mathbf{B}_j\|_2 \right].
\end{aligned}
$$

We have

$$
\begin{aligned}
0 \geq L(\hat{\mathbf{B}}) - L(\mathbf{B}^*) =& \frac{1}{2n} \left\| \tilde{\mathbf{X}} \text{vec}(\hat{\mathbf{B}} - \mathbf{B}^*) \right\|_2^2 - \frac{1}{n} \boldsymbol{\epsilon}^\top \tilde{\mathbf{X}} \text{vec}(\hat{\mathbf{B}} - \mathbf{B}^*) \\
& - \left[P_{C+V}(\mathbf{B}^*) - P_{C+V}(\hat{\mathbf{B}}) \right].
\end{aligned} \tag{2.13}
$$

where $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)^T$. Set $\mathbf{V} = \hat{\mathbf{B}} - \mathbf{B}^*$. For the third term in the right side of Eq. (2.13):

$$
\begin{aligned}
& P_{C+V}(\mathbf{B}^*) - P_{C+V}(\hat{\mathbf{B}}) \\
=& \left[P_{\mathcal{S}}(\mathbf{B}^*) - P_{\mathcal{S}}(\hat{\mathbf{B}}) \right] + \left[P_{\mathcal{S}^C}(\mathbf{B}^*) - P_{\mathcal{S}^C}(\hat{\mathbf{B}}) \right] + \frac{\lambda_2 \tau}{2} \left[\|\mathbf{B}^*\|_F^2 - \|\hat{\mathbf{B}}\|_F^2 \right] \\
=& \sum_{j \in \mathcal{J}} \left[\lambda_1 \Big(\|\mathbf{W}_{\mathcal{I}_j, j} \mathbf{H}_{\mathcal{I}_j, j} \mathbf{B}_j^*\|_1 - \|\mathbf{W}_{\mathcal{I}_j, j} \mathbf{H}_{\mathcal{I}_j, j} \hat{\mathbf{B}}_j\|_1 \Big) + \lambda_2 u_j \Big(\left\|\mathbf{B}_j^*\right\|_2 - \|\hat{\mathbf{B}}_j\|_2 \Big) \right] \\
& + \left[0 - P_{\mathcal{S}^C}(\hat{\mathbf{B}}) \right] + \frac{\lambda_2 \tau}{2} \left[-\|\mathbf{B}^* - \hat{\mathbf{B}}\|_F^2 + 2\text{tr} \left\{ (\mathbf{B}^*)^T (\mathbf{B}^* - \hat{\mathbf{B}}) \right\} \right] \\
\leq& \sum_{j \in \mathcal{J}} \left[\lambda_1 \left\| \mathbf{W}_{\mathcal{I}_j, j} \mathbf{H}_{\mathcal{I}_j, j} (\mathbf{B}_j^* - \hat{\mathbf{B}}_j) \right\|_1 + \lambda_2 u_j \left\| \mathbf{B}_j^* - \hat{\mathbf{B}}_j \right\|_2 \right] \\
& - P_{\mathcal{S}^C}(\mathbf{V}) + \frac{\lambda_2 \tau}{2} \left[-\|\mathbf{V}\|_F^2 + 2 \|\mathbf{B}^*\|_F \left\| \hat{\mathbf{B}} - \mathbf{B}^* \right\|_F \right] \\
=& P_{\mathcal{S}}(\mathbf{V}) - P_{\mathcal{S}^C}(\mathbf{V}) + \frac{\lambda_2 \tau}{2} \left[-\|\mathbf{V}\|_F^2 + 2 \|\mathbf{B}^*\|_F \|\mathbf{V}\|_F \right].
\end{aligned} \tag{2.14}
$$

The inequality in Eq. (2.14) comes from the triangle inequality of the norm and the Cauchy-Schwarz inequality.

For the second term in the right side of Eq. (2.13):

$$\begin{aligned}
&\frac{1}{n}\boldsymbol{\epsilon}^\top\tilde{\mathbf{X}}\text{vec}(\hat{\mathbf{B}}-\mathbf{B}^*) = \frac{1}{n}\boldsymbol{\epsilon}^\top\sum_{j=1}^{p}\text{diag}(\mathbf{X}_j)(\hat{\mathbf{B}}_j-\mathbf{B}_j^*)\\
=&\frac{1}{n}\boldsymbol{\epsilon}^\top\big[\sum_{j\in\mathcal{J}}\text{diag}(\mathbf{X}_j)(\hat{\mathbf{B}}_j-\mathbf{B}_j^*)+\sum_{j\in\mathcal{J}^C}\text{diag}(\mathbf{X}_j)(\hat{\mathbf{B}}_j-\mathbf{B}_j^*)\big]\\
=&\frac{1}{n}\boldsymbol{\epsilon}^\top\big[\sum_{j\in\mathcal{J}}\text{diag}(\mathbf{X}_j)\tilde{\mathbf{H}}_j^{-1}\tilde{\mathbf{H}}_j(\hat{\mathbf{B}}_j-\mathbf{B}_j^*)+\sum_{j\in\mathcal{J}^C}\text{diag}(\mathbf{X}_j)u_j^{-1}u_j(\hat{\mathbf{B}}_j-\mathbf{B}_j^*)\big].
\end{aligned}$$

Since $\tilde{\mathbf{H}}_j^{-1}=(\mathbf{H}_j^-,\mathbf{1}_j^-)$, then $\tilde{\mathbf{H}}_j^{-1}\tilde{\mathbf{H}}_j=\mathbf{H}_j^-\mathbf{H}_j+\mathbf{1}_j^-\mathbf{1}^T/n$, and $\mathbf{1}^T\mathbf{1}_j^-/n=1$.

$$\begin{aligned}
&\frac{1}{n}\boldsymbol{\epsilon}^\top\tilde{\mathbf{X}}\text{vec}(\hat{\mathbf{B}}-\mathbf{B}^*)\\
=&\frac{1}{n}\boldsymbol{\epsilon}^\top\big[\sum_{j\in\mathcal{J}}\text{diag}(\mathbf{X}_j)(\mathbf{H}_j^-\mathbf{H}_j+\mathbf{1}_j^-\mathbf{1}^T/n)(\hat{\mathbf{B}}_j-\mathbf{B}_j^*)+\sum_{j\in\mathcal{J}^C}\text{diag}(\mathbf{X}_j)u_j^{-1}u_j(\hat{\mathbf{B}}_j-\mathbf{B}_j^*)\big]\\
=&\frac{1}{n}\boldsymbol{\epsilon}^\top\big[\sum_{j\in\mathcal{J}}\text{diag}(\mathbf{X}_j)(\mathbf{H}_j^-\mathbf{W}_j^{-1}\mathbf{W}_j\mathbf{H}_j)(\hat{\mathbf{B}}_j-\mathbf{B}_j^*)+\sum_{j\in\mathcal{J}}\text{diag}(\mathbf{X}_j)u_j^{-1}u_j(\mathbf{1}_j^-\mathbf{1}^T/n)(\hat{\mathbf{B}}_j-\mathbf{B}_j^*)\\
&\quad+\sum_{j\in\mathcal{J}^C}\text{diag}(\mathbf{X}_j)u_j^{-1}u_j(\hat{\mathbf{B}}_j-\mathbf{B}_j^*)\big].
\end{aligned}$$

Combining with $\frac{\lambda_1}{2} > \max_{j\in\mathcal{J}}\left\|\frac{1}{n}\boldsymbol{\epsilon}^T\text{diag}(\mathbf{X}_j)\mathbf{H}_j^-\mathbf{W}_j^{-1}\right\|_\infty$ and $\frac{\lambda_2}{2\sqrt{n}} > \max_{i,j}|\frac{1}{n}\epsilon_i X_{ij}u_j^{-1}|$, we have:

$$\begin{aligned}
&\frac{1}{n}\boldsymbol{\epsilon}^\top\tilde{\mathbf{X}}\text{vec}(\hat{\mathbf{B}}-\mathbf{B}^*)\\
\le&\frac{\lambda_1}{2}\sum_{j\in\mathcal{J}}\|\mathbf{W}_j\mathbf{H}_j(\hat{\mathbf{B}}_j-\mathbf{B}_j^*)\|_1+\frac{\lambda_2}{2\sqrt{n}}\sum_{j\in\mathcal{J}}u_j\|\hat{\mathbf{B}}_j-\mathbf{B}_j^*\|_1\\
&+\frac{\lambda_2}{2\sqrt{n}}\sum_{j\in\mathcal{J}^C}u_j\|\hat{\mathbf{B}}_j-\mathbf{B}_j^*\|_1\\
=&\frac{\lambda_1}{2}\sum_{j\in\mathcal{J}}\|\mathbf{W}_j\mathbf{H}_j(\hat{\mathbf{B}}_j-\mathbf{B}_j^*)\|_1+\frac{\lambda_2}{2\sqrt{n}}\sum_{j=1}^{p}u_j\|\hat{\mathbf{B}}_j-\mathbf{B}_j^*\|_1\\
\le&\frac{\lambda_1}{2}\sum_{j\in\mathcal{J}}\|\mathbf{W}_j\mathbf{H}_j(\hat{\mathbf{B}}_j-\mathbf{B}_j^*)\|_1+\frac{\lambda_2}{2}\sum_{j=1}^{p}u_j\|\hat{\mathbf{B}}_j-\mathbf{B}_j^*\|_2\\
\le&\frac{1}{2}P_{\mathcal{S}}(\mathbf{V})+\frac{1}{2}P_{\mathcal{S}^C}(\mathbf{V}).
\end{aligned}\tag{2.15}$$

Combining Eq. (2.13), Eq. (2.14), and Eq. (2.15), we have:

$$\begin{aligned}\frac{1}{2n}\left\|\tilde{\mathbf{X}}\text{vec}(\hat{\mathbf{B}}-\mathbf{B}^*)\right\|_2^2 \leq & \frac{3}{2}P_S(\mathbf{V})-\frac{1}{2}P_{S^C}(\mathbf{V}) \\ & +\frac{\lambda_2\tau}{2}\left[-\|\mathbf{V}\|_F^2+2\|\mathbf{B}^*\|_F\|\mathbf{V}\|_F\right]\end{aligned} \tag{2.16}$$

Omit the term $-\frac{1}{2}P_{S^C}(\mathbf{V})$ on the right hand side of (2.16), we have:

$$\begin{aligned}&\frac{1}{2n}\left\|\tilde{\mathbf{X}}\text{vec}(\hat{\mathbf{B}}-\mathbf{B}^*)\right\|_2^2 \\ \leq & \frac{3}{2}\sum_{j\in\mathcal{J}}\left[\lambda_1\left\|\mathbf{W}_{\mathcal{I}_j,j}\mathbf{H}_{\mathcal{I}_j,j}\mathbf{V}_j\right\|_1+\lambda_2 u_j\|\mathbf{V}_j\|_2\right]+\frac{\lambda_2\tau}{2}\left[-\|\mathbf{V}\|_F^2+2\|\mathbf{B}^*\|_F\|\mathbf{V}\|_F\right] \\ \leq & \frac{3}{2}\sum_{j\in\mathcal{J}}\left[\lambda_1\left\|\mathbf{W}_{\mathcal{I}_j,j}\right\|_\infty\left\|\mathbf{H}_{\mathcal{I}_j,j}\mathbf{V}_j\right\|_1+\lambda_2 u_j\|\mathbf{V}_j\|_2\right]+\frac{\lambda_2\tau}{2}\left[-\|\mathbf{V}\|_F^2+2\|\mathbf{B}^*\|_F\|\mathbf{V}\|_F\right] \\ \leq & \frac{3}{2}\sum_{j\in\mathcal{J}}\left[2\lambda_1\sqrt{|\mathcal{I}_j|}\left\|\mathbf{W}_{\mathcal{I}_j,j}\right\|_\infty\|\mathbf{V}_j\|_2+\lambda_2 u_j\|\mathbf{V}_j\|_2\right]+\frac{\lambda_2\tau}{2}\left[-\|\mathbf{V}\|_F^2+2\|\mathbf{B}^*\|_F\|\mathbf{V}\|_F\right] \\ \leq & 3\lambda_1\sqrt{\mathcal{J}}(\max_{j\in\mathcal{J}}\sqrt{|\mathcal{I}_j|}\left\|\mathbf{W}_{\mathcal{I}_j,j}\right\|_\infty)\|\mathbf{V}\|_F+\frac{3\lambda_2\sqrt{\mathcal{J}}}{2}(\max_{j\in\mathcal{J}}|u_j|)\|\mathbf{V}\|_F \\ & +\frac{\lambda_2\tau}{2}\left[-\|\mathbf{V}\|_F^2+2\|\mathbf{B}^*\|_F\|\mathbf{V}\|_F\right] \\ \leq & 3\Big[\lambda_1\sqrt{\mathcal{J}}(\max_{j\in\mathcal{J}}\sqrt{|\mathcal{I}_j|}\left\|\mathbf{W}_{\mathcal{I}_j,j}\right\|_\infty)+\lambda_2\sqrt{\mathcal{J}}(\max_{j\in\mathcal{J}}|u_j|)+\lambda_2\tau\|\mathbf{B}^*\|_F\Big]\|\mathbf{V}\|_F-\frac{\lambda_2\tau}{2}\|\mathbf{V}\|_F^2 .\end{aligned} \tag{2.17}$$

From (2.16), we can see that:

$$0 \leq \frac{3}{2}P_S(\mathbf{V})-\frac{1}{2}P_{S^C}(\mathbf{V})+\frac{\lambda_2\tau}{2}\left[-\|\mathbf{V}\|_F^2+2\|\mathbf{B}^*\|_F\|\mathbf{V}\|_F\right]$$

which implies $\mathbf{V}\in\mathcal{C}(\mathbf{B}^*;\lambda_1,\lambda_2,\tau)$ with $\mathcal{C}(\mathbf{B}^*;\lambda_1,\lambda_2,\tau)$ defined in Assumption 1. Then under Assumption 1, we have that $\frac{\gamma}{2}\left\|\hat{\mathbf{B}}-\mathbf{B}^*\right\|_F^2 \leq \frac{1}{2n}\left\|\tilde{\mathbf{X}}\text{vec}(\hat{\mathbf{B}}-\mathbf{B}^*)\right\|_2^2$. After some derivations, we get

$$\left\|\hat{\mathbf{B}}-\mathbf{B}^*\right\|_F \leq \frac{6(\lambda_1\sqrt{\mathcal{J}}\max_{j\in\mathcal{J}}\sqrt{|\mathcal{I}_j|}\left\|\mathbf{W}_{\mathcal{I}_j,j}\right\|_\infty+\lambda_2\sqrt{\mathcal{J}}\max_{j\in\mathcal{J}}|u_j|+\lambda_2\tau\|\mathbf{B}^*\|_F)}{\gamma+\lambda_2\tau} \tag{2.18}$$

We plug Eq. (2.18) into Eq. (2.17) and obtain:

$$\frac{1}{2n}\left\|\tilde{\mathbf{X}}\text{vec}(\hat{\mathbf{B}}-\mathbf{B}^*)\right\|_2^2 \leq \frac{18(\lambda_1\sqrt{\mathcal{J}}\max_{j\in\mathcal{J}}\sqrt{|\mathcal{I}_j|}\left\|\mathbf{W}_{\mathcal{I}_j,j}\right\|_\infty+\lambda_2\sqrt{\mathcal{J}}\max_{j\in\mathcal{J}}|u_j|+\lambda_2\tau\left\|\mathbf{B}^*\right\|_F)^2}{\gamma+\lambda_2\tau}.$$

To consider the converge rate of Eq. (2.18), we study the rate conditions of λ_1 and λ_2 in Theorem 1. First, we introduce the well-known result of the maximum of a series of sub-Gaussian random variables in Lemma 1.

Lemma 1. *If $\epsilon_1,\ldots,\epsilon_n$ are sub-Gaussian random variables that $P(\epsilon_i \geq t) \leq \exp(-\frac{t^2}{2\sigma_i^2})$. $\epsilon_1,\ldots,\epsilon_n$ do not need to be independent. We denote $\sigma^2=\max_i(\sigma_i^2)$ and there exists C that*

$$P(\max_i|\epsilon_i|\geq t)\leq Cn\exp(-\frac{t^2}{2\sigma^2}).$$

By Lemma 1 and Assumption 2, we obtain that with probability going to 1,

$$\max_{i,j}|\frac{1}{n}\epsilon_i X_{ij}u_j^{-1}|\leq\frac{C_2}{n}\max_i|\epsilon_i|=O(C_2\frac{\sqrt{\log(n)}}{n}\sigma),$$

where $C_2=\max_j\|u_j^{-1}\mathbf{X}_j\|_\infty$. As a result, λ_2 should have a rate of at least $O(C_2\sqrt{\log(n)/n})$ as $n\to\infty$.

For λ_1, we have:

$$\max_{j\in\mathcal{J}}\left\|\frac{1}{n}\boldsymbol{\epsilon}^T\text{diag}(\mathbf{X}_j)\mathbf{H}_j^-\mathbf{W}_j^{-1}\right\|_\infty\leq\frac{C_1}{n}\max_{j\in\mathcal{J},k}|\boldsymbol{\epsilon}^T(\mathbf{H}_j^-)_k|,$$

where $(\mathbf{H}_j^-)_k$ is the k^{th} column of $\mathbf{H}_j^-$, and $C_1 = max_{j\in\mathcal{J}}\|\mathbf{X}_j\|_\infty\max_{j\in\mathcal{J}}\|\mathbf{W}_j^{-1}\|_\infty$. By the definition of $\mathbf{H}_j^-$, it is easy to check by linear algebra that $\|\mathbf{H}_j^-\|_\infty\leq 1$ and $\|(\mathbf{H}_j^-)_k\|_2^2\leq n$. Thus, by the general Hoeffding's inequality [43], there exists a constant C such that

$$P(|\boldsymbol{\epsilon}^T(\mathbf{H}_j^-)_k|\geq t)\leq C\exp(-\frac{t^2}{\sigma^2\|(\mathbf{H}_j^-)_k\|_2^2})\leq C\exp(-\frac{t^2}{\sigma^2 n}) \tag{2.19}$$

Then each $\boldsymbol{\epsilon}^T(\mathbf{H}_j^-)_k$ is also a sub-Gaussian random variables. Combining Lemma 1, we obtain:

$$P(\max_{j\in\mathcal{J},k}|\boldsymbol{\epsilon}^T(\mathbf{H}_j^-)_k|\geq t)\leq C(n-1)|\mathcal{J}|\exp(-\frac{t^2}{2\sigma^2 n}),$$

and

$$\max_{j\in\mathcal{J}}\left\|\frac{1}{n}\boldsymbol{\epsilon}^T\text{diag}(\mathbf{X}_j)\mathbf{H}_j^-\mathbf{W}_j^{-1}\right\|_\infty = O(C_1\sqrt{\log(n|\mathcal{J}|)/n}).$$

λ_1 should have a rate of at least $O(C_1\sqrt{\log(n|\mathcal{J}|)/n})$ as $n\to\infty$.

Corollary 1 can be directly derived by Theorem 1 and the rate of λ_1 and λ_2.

To consider the effects of adaptive learning, if $\mathbf{W}^{-1}_{\mathcal{I}_j^C,j} = O(\sqrt{\log(n)/n})$ for each $j\in\mathcal{J}$, we still use the general Hoeffding's inequality but include the weight $\mathbf{W}^{-1}_{k,j}$ for $k\in\mathcal{I}_j^C$:

$$P(|\boldsymbol{\epsilon}^T(\mathbf{H}_j^-)_k\mathbf{W}^{-1}_{k,j}|\geq t)\leq C\exp(-\frac{t^2}{\sigma^2\|(\mathbf{H}_j^-)_k\mathbf{W}^{-1}_{k,j}\|_2^2})\leq C^*\exp(-\frac{t^2}{\sigma^2\log(n)}).$$

For the other $k\in\mathcal{I}_j^C$, Eq. (2.19) still holds, and then we can derive that λ_1 should have a rate of at least $O(C_1 n^{-1/2}\max\left(\log(|\mathcal{J}|\max_{j\in\mathcal{J}}|\mathcal{I}_j|),\log(n)\log(n|\mathcal{J}|)\right)/n)^{-1/2}$. When n is large enough, this rate becomes $O(C_1\sqrt{\log(|\mathcal{J}|\max_{j\in\mathcal{J}}|\mathcal{I}_j|)/n})$.

□

2.8.2 Proof of Theorem 2

Proof. We begin by introducing some notations. Use $\mathcal{M}_j$ to denote the MST of an original graph $\mathcal{G}$ with $|b^{\text{ini}}_{i_1,j}-b^{\text{ini}}_{i_2,j}|$ as the distance between $(i_1,i_2)\in\mathcal{E}$ for the j^{th} variable, that is,

$$\mathcal{M}_j = \arg\min_{\mathcal{M}\in\mathcal{T}}\sum_{(i_1,i_2)\in\mathcal{M}}|b^{\text{ini}}_{i_1,j}-b^{\text{ini}}_{i_2,j}|,$$

where $\mathcal{T}$ is the set of all spanning trees of $\mathcal{G}$. We call the edges that connect vertices from two different clusters as the bridges for clusters. For notational simplicity, we drop the index j from $\mathcal{M}_j$ and some other notations depending on j in the remaining proof as long as the context is subject to no confusion. We denote $\mathcal{B}(\mathcal{G})$ as the set of bridges in $\mathcal{G}$ for the true clusters of $\mathbf{B}_j^*$, and

denote $Q(\mathcal{M}) = \sum_{(i_1,i_2)\in\mathcal{M}} |b^{\text{ini}}_{i_1,j} - b^{\text{ini}}_{i_2,j}|$.

Now we decompose $\mathcal{T}$ by the number of bridges included in each spanning tree. We define:

$$\mathcal{T}_r = \{\mathcal{M} \in \mathcal{T} : |\mathcal{B}(\mathcal{M})| = r\}.$$

It is easy to see that $\mathcal{T}_r = \emptyset$ for $r < R_j - 1$ and $r \geq n$. Also, $\{\mathcal{T}_r\}$ are disjoint and $\bigcup_r \mathcal{T}_r = \mathcal{T}$. We first prove that $\mathcal{M} \in \mathcal{T}_{R_j-1}$ with probability tending to 1 as $n \to \infty$, which depends on the following lemma:

Lemma 2. *For each $\mathcal{M} \in \mathcal{T}_r$ with $R_j \leq r < n$, there exists $\mathcal{M}' \in \mathcal{T}_{r-1}$ that $\mathcal{M}'$ can be formed from $\mathcal{M}$ by deleting a bridge in $\mathcal{B}(\mathcal{M})$ and then adding an edge (not bridge) in $\mathcal{E} \setminus \mathcal{B}(\mathcal{G})$.*

Proof of Lemma 2: Select an arbitrary bridge (s,t) in $\mathcal{B}(\mathcal{M})$. Since $\mathcal{M}$ is a MST, when deleting (s,t) from $\mathcal{M}$, $\mathcal{M}$ will split into two sub-graphs $\mathcal{M}_s$ and $\mathcal{M}_t$. There are now two cases to examine. In the first case, $\mathcal{M}_t$ includes vertices that in the same cluster of s; by the assumption that the sub-graph of the vertices in the same cluster is connected in $\mathcal{G}$, we can find a vertex i in $\mathcal{M}_t$ and a vertex j in $\mathcal{M}_s$ that both i and j comes from the cluster of s and $(i,j) \in \mathcal{E} \setminus \mathcal{B}(\mathcal{G})$, By adding (i,j), we obtain a new spanning tree with the number of bridges $r-1$, which is $\mathcal{M}'$. A similar conclusion holds when $\mathcal{M}_s$ includes vertices that in the same cluster of t. In the second case, $\mathcal{M}_t$ does not include observations of that in the same cluster of s, and $\mathcal{M}_s$ does not include observations that in the same cluster of t. We could not find $\mathcal{M}' \in \mathcal{T}_{r-1}$ by adding an edge.

Now we prove that there is at least one bridge in $\mathcal{B}(\mathcal{M})$ satisfying the first case when $R_j \leq r < n$. If there is no bridge satisfying the first case, then when we delete all the r bridges in $\mathcal{M}$, we can obtain $r+1$ disjoint sub-graphs, where the vertices of different sub-graphs are not in the same clusters. Thus, there will be at least $r+1 > R_j$ clusters among observations, which contradicts to the fact that the true number of clusters is R_j. As a result, for any $\mathcal{M} \in \mathcal{T}_r$ with $R_j \leq r < n$, we could find a bridge (s,t) that satisfies the first case and then construct a MST $\mathcal{M}' \in \mathcal{T}_{r-1}$. This ends the proof of Lemma 2. With $\mathcal{M}$ and $\mathcal{M}'$ defined in Lemma 2, we consider the difference

between $Q(\mathcal{M})$ and $Q(\mathcal{M}')$. Under the true coefficient matrix $\mathbf{B}^*$, define:

$$\eta = \min\{|b^*_{i_1,j} - b^*_{i_2,j}| : (i_1, i_2) \in \mathcal{B}(\mathcal{G})\} \tag{2.20}$$

It is obvious that η should be larger than 0 otherwise two clusters will share the same value of coefficient and merge together. For $(i_1, i_2) \in \mathcal{E} \setminus \mathcal{B}(\mathcal{G})$, $|b^*_{i_1,j} - b^*_{i_2,j}| = 0$.

Since $\mathbf{B}^{\text{ini}}$ is a consistent estimator, which means that for any σ and each b^{ini}_{ij}, $|b^{\text{ini}}_{ij} - b^*_{ij}| < \sigma$ with probability tending to 1 as $n \to \infty$. Select $\sigma < \eta/4$.Then for any $(i_1, i_2) \in \mathcal{E} \setminus \mathcal{B}(\mathcal{G})$ and $(i_3, i_4) \in \mathcal{G}$, $|b^{\text{ini}}_{i_1,j} - b^{\text{ini}}_{i_2,j}| - |b^{\text{ini}}_{i_3,j} - b^{\text{ini}}_{i_4,j}| \geq \eta - 4\sigma > 0$. As a result, $Q(\mathcal{M}) - Q(\mathcal{M}') > 0$ with probability tending to 1 as $n \to \infty$. Thus, $\arg\min_{\mathcal{M} \in \mathcal{T}_{r-1}} Q(\mathcal{M}) \leq Q(\mathcal{M}') < \arg\min_{\mathcal{M} \in \mathcal{T}_r} Q(\mathcal{M})$ for $R_j \leq r < n$, and $\mathcal{M} \in \mathcal{T}_{R_j - 1}$ is proved.

The last thing we need to prove is that $|\mathcal{I}_j| = |\mathcal{B}(\mathcal{G}_j)| \leq 2|\mathcal{B}(\mathcal{M})| = 2(R_j - 1)$, where $\mathcal{G}_j$ is the variable-specific chain graph constructed by the DFS algorithm of $\mathcal{M}$ with a random starting vertex. This result can be obtained by directly applying Lemma1 Eq. (16) of [28]. □

3. STRUCTURED REGULARIZED MATRIX DECOMPOSITION BASED BICLUSTERING

3.1 Introduction

Biclustering is a technique that investigates the structure of a data matrix and discovers the latent subgroups of rows and their associated columns that are strongly related to each other. These latent subgroups are called biclusters. In many applications, samples are represented by rows (or columns) of the matrix, and correspondingly features are represented by columns (or rows). In the conventional clustering problem, samples are grouped as clusters based on the entire feature set, whereas in biclustering, the characteristics of samples from each cluster are well-explained by only a distinct subset of features. For example, in the study of single-cell RNA sequencing data, researchers try to cluster cells according to their gene expression levels. However, different subtypes of cells are related to different groups of genes, and the critical genes of a particular subtype could barely provide any information for other subtypes. Meanwhile, most genes are not useful for identifying the cell subtype and are redundant for the study. Biclustering can be useful to cluster cells and discover the associated critical genes simultaneously.

The idea of relating groups of samples and features is first introduced by [44]. The conventional approaches of biclustering are mainly built on the following four strategies: row and column clustering combination [45, 46], greedy search [47, 48, 49], divide-and-conquer [50], and exhaustive bicluster enumeration [51]. [52] provides a survey of conventional biclustering methods.

In spite of all these developments, the biclustering problem can still be challenging to solve, since the complex structure of a data matrix may lead to various layouts of biclusters that are hard to be revealed by existing methods. For example, Figure 3.1 includes four typical layouts of three biclusters. The rows and columns in the same cluster are aligned together (e.g., after a permutation of rows and columns) and form a grey block. The blue parts of the blocks indicate the overlapping between different biclusters. In the most complicated layout shown in Figure 3.1d, any pair of

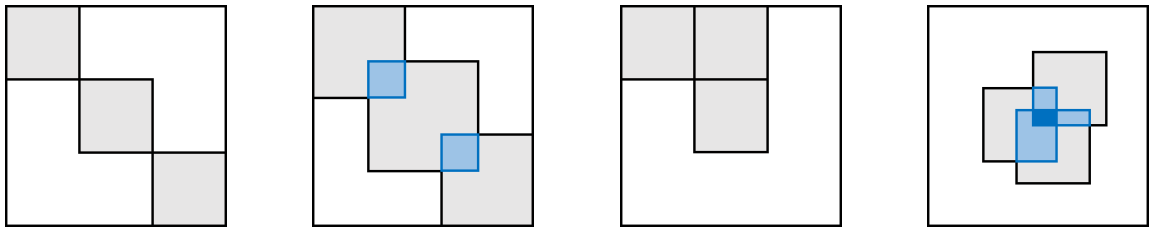


Figure 3.1: Different layouts of biclusters in the data matrix: (a) The simplest case: Different biclusters are totally separated and do not share any rows or columns. (b) The adjacent biclusters overlap with each other and share a part of entries. (c) Different biclusters could contain exactly the same group of rows but entirely different groups of columns. (d) Arbitrary overlapping is allowed among multiple biclusters, which is one of the most complicated cases. In (c) and (d), some rows or columns are not included in any bicluster.

two biclusters can overlap, and many rows and columns are not included in any bicluster. Most of the existing approaches are designed to fit a specific layout of biclusters e.g., Figure 3.1a to 3.1c, and they encounter difficulties when facing a complicated case such as Figure 3.1d. In real-world applications, one does not know the layout of biclusters in the data matrix in advance, motivating us to develop a new method to handle complex layouts of biclusters.

The purpose of this chapter is to develop a new structured regularized matrix decomposition based biclustering method that is suitable to deal with the complex structure of data matrix such as in Figure 3.1d. Particularly, our method allows the overlapping of multiple biclusters, and it can effectively screen out the rows and columns that are not involved in any bicluster. When designing our method, we treat the raw data matrix as the summation of several latent sparse matrices, each of which reflects the effects of a distinct bicluster to the data matrix [53]. From this view, we transform the biclustering problem into a matrix decomposition problem, which in turn is cast into a least-squares optimization problem with a properly defined sparsity-inducing penalty called the exclusive Lasso penalty. Because of the novel use of this penalty in biclustering problems, we refer to our method as **Bi**C**l**ustering with the **E**xclusive **L**asso penalty (BCEL).

BCEL can be considered as a substantial improvement over several existing SVD based biclustering methods [54, 55, 56, 57, 58]. Compared with existing methods, BCEL has the following

advantageous features:

- While most SVD based methods extract the biclusters in a sequential manner, BCEL simultaneously identifies multiple biclusters. The simultaneous approach does not suffer from the error accumulation problem encountered by the sequential approach, leading to improved performance.

- The exclusive Lasso penalty respects the grouping structure of the biclusters, so the competition of rows and columns being included in biclusters is done separately within each cluster.

- The bicluster membership is decided by a subsampling based procedure called stability selection. The results from this procedure are more stable than directly using the outputs from the penalized least-squares problem.

The effectiveness of BCEL has been demonstrated by a simulation study that compares it with several existing biclustering methods.

Here is an outline of the rest of this chapter. In Section 3.2, we present an overview of the existing structured regularized matrix decomposition based biclustering methods through a unified framework. Section 3.3 develops our proposed method, including formulation of the optimization problem, derivation of a computational algorithm, handling of missing data, and selection of tuning parameters. A simulation study is presented in Section 3.4. Section 3.5 applies the new method on a single-cell RNA sequencing dataset and compares it with other methods.

3.2 Structured Regularized Matrix Decomposition

3.2.1 Unified Framework

We present a unified formulation of biclustering that uses structured regularized matrix decomposition and show that this formulation synthesizes various existing biclustering methods. Let $\mathbf{X} \in$

$\mathbb{R}^{n\times p}$ be a $n \times p$ data matrix. We model $\mathbf{X}$ as the summation of r matrices $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_r \in \mathbb{R}^{n\times p}$ and an error matrix $\mathbf{E} \in \mathbb{R}^{n\times p}$:

$$\mathbf{X} = \sum_{k=1}^{r} \mathbf{A}_k + \mathbf{E}, \tag{3.1}$$

where each $\mathbf{A}_k$, representing the effects of a bicluster, is a sparse matrix with non-zero entries concentrating on a subgroup of rows and columns, and the error matrix $\mathbf{E}$ has entries that can be assumed to be i.i.d. random samples from a zero-mean distribution. After properly aligning the rows and columns of $\mathbf{A}_k$, the non-zero entries of $\mathbf{A}_k$ can form a block like the grey one in Figure 3.1. We obtain different biclustering models by making different assumptions on the structure of each $\mathbf{A}_k$. Figure 3.2 gives three commonly used structures.

5	5	5	5
5	5	5	5
5	5	5	5
5	5	5	5

(a)

+	-10	10	20	30
5	-5	15	25	35
2	-2	12	22	32
-4	-14	4	14	24
-8	-18	8	18	28

(b)

×	1	2	3	4
5	5	10	15	20
2	2	4	6	8
-4	-4	-8	-12	-16
-8	-8	-16	-24	-32

(c)

Figure 3.2: Different structures of the non-zero entries in $\mathbf{A}_k$: (a) Constant structure: The effect of a bicluster is the same for all its entries. (b) Additive structure: The effect of a bicluster on its entries is the summation of the row effect and the column effect. (c) Multiplicative structure: The effect of a bicluster on its entries is the multiplication of the row effect and the column effect.

Using model Eq. (3.1) as guidance, we can develop a biclustering method by solving the following optimization problem:

$$\min_{\mathbf{A}_k \in \mathcal{G}(\mathbf{A}_k), k=1,\ldots,r} \|\mathbf{X} - \sum_{k=1}^{r} \mathbf{A}_k\|_F^2 + \sum_{k=1}^{r} F(\mathbf{A}_k), \tag{3.2}$$

where $F(\mathbf{A}_k)$ is a penalty function on $\mathbf{A}_k$ to encourage it to have a specified property, and $\mathcal{G}(\mathbf{A}_k)$

is the support of $\mathbf{A}_k$.

Figure 3.3 depicts two different ways of forming model Eq. (3.1). In Figure 3.3a, $\mathbf{X}$ is assumed to have a checkerboard structure after rearranging its rows and columns. Each $\mathbf{A}_i$ consists of a block of $\mathbf{X}$ with non-zero entries, which are assumed to be a constant. The corresponding model, called the block model, has the advantage of efficiently representing the structure in Figure 3.1c, but may have difficulties in representing overlapping biclusters. In Figure 3.3b, each $\mathbf{A}_k = \mathbf{u}_k\mathbf{v}_k^T$ is a rank-1 matrix, where $\mathbf{u}_k \in \mathbb{R}^n$ and $\mathbf{v}_k \in \mathbb{R}^p$ are two sparse vectors whose none zero entries represent respectively the selection of rows and columns for the k^{th} bicluster. Defining $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r) \in \mathbb{R}^{n\times r}$ and $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r) \in \mathbb{R}^{p\times r}$, we can rewrite model Eq. (3.1) as

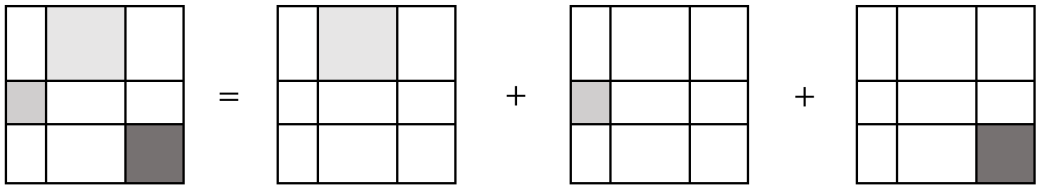
$$\mathbf{X} = \sum_{k=1}^{r} \mathbf{u}_k\mathbf{v}_k^T + \mathbf{E} = \mathbf{U}\mathbf{V}^T + \mathbf{E}. \tag{3.3}$$

This is called a multiplicative model.

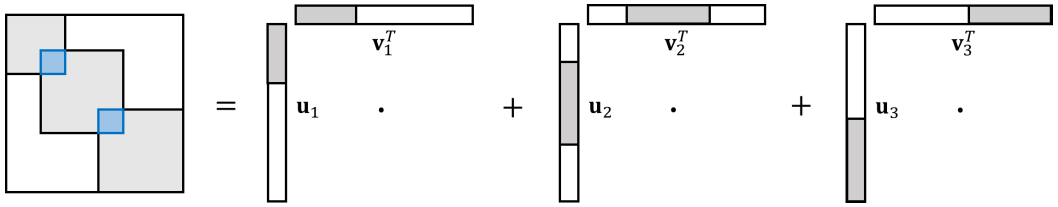
In the following subsections, we give an overview of existing matrix decomposition based biclustering methods and show that many of them are special cases of either the block model or the multiplicative model. A summary of the main methods is shown in Table 3.1.

Table 3.1: Summary of structured regularized matrix decomposition based biclustering methods.

Method	Model Based	Structure	Seq/Simul	Allow Overlapping
sparseBC	Block	Constant	Simul	No
OECMBS	Block	Constant	Simul	No
COBRA	Block	Constant	Simul	No
SSVD	SVD	Multi	Seq	Yes
RoBiC	SVD	Multi	Seq	Yes
SRRR	SVD	Multi	Simul	Yes
nmfsc	NMF	Multi	Simul	Yes
Plaid	-	Addi	Seq	Yes



(a) Block model



(b) Multiplicative model

Figure 3.3: Examples of matrix decomposition based biclustering models.

3.2.2 Block Models

Three existing methods fall in the category of block models. They are “sparse biclustering of transposable data" (sparseBC) [59], “optimal estimation and completion of matrices with biclustering structures" (OECMBS) [60], and “Convex BiclusteRing Algorithm" (COBRA) [61].

The sparseBC method [59] assumes that the matrix elements are normally distributed with a bicluster-specific mean term and common variance. Thus it transforms the biclustering problem to a regularized log-likelihood maximization problem. The ℓ_1-norm penalty is applied to the means of the biclusters in order to obtain sparse and interpretable biclusters. Denoting $\mathcal{C}_1, \ldots, \mathcal{C}_K$ and $\mathcal{D}_1, \ldots, \mathcal{D}_R$ as the partitions of rows and columns respectively corresponding to the biclusters, the sparseBC solves the following optimization problem:

$$\min_{\mathcal{C}_1,\ldots,\mathcal{C}_K,\mathcal{D}_1,\ldots,\mathcal{D}_R,\boldsymbol{\mu}\in\mathbb{R}^{K\times R}} \frac{1}{2}\sum_{k=1}^{K}\sum_{r=1}^{R}\sum_{i\in\mathcal{C}_k}\sum_{j\in\mathcal{D}_r}(X_{ij}-\mu_{kr})^2+\lambda\sum_{k=1}^{K}\sum_{r=1}^{R}|\mu_{kr}|,$$

where μ_{kr} denote the mean parameter for the block/bicluster corresponding to rows in $\mathcal{C}_k$ and

columns in $\mathcal{D}_r$. The algorithm of sparseBC solves the above problem by iteratively updating the partitions of rows and columns and estimating the block mean by soft-thresholding the sample mean of the block. OEMCBS [60] is essentially the same model as sparseBC but allowing missing values. To achieve needed regularization, OEMCBS imposes a bound for the means of the biclusters instead of using the ℓ_1-norm penalty in sparseBC. Both sparseBC and OEMCBS use a descent algorithm to solve a non-convex problem, and thus there is no guarantee convergence to global optimal.

COBRA [61] considers a convex objective function to solve the biclustering problem. By penalizing the differences between rows and between columns in the objective function, similar rows and columns tend to merge to create clusters. Specifically, COBRA solves the following convex optimization problem:

$$\min_{\widetilde{\mathbf{X}}} ||\mathbf{X} - \widetilde{\mathbf{X}}||_F^2 + \lambda\{\Omega_{\mathbf{W}}(\widetilde{\mathbf{X}}) + \Omega_{\widetilde{\mathbf{W}}}(\widetilde{\mathbf{X}}^T)\},$$

where $\mathbf{W} = (w_{ij})$ and $\widetilde{\mathbf{W}} = (\tilde{w}_{ij})$ are weight matrices, $\Omega_{\mathbf{W}}(\widetilde{\mathbf{X}}) = \sum_{i<j} w_{ij}||\widetilde{\mathbf{X}}_{\cdot,i} - \widetilde{\mathbf{X}}_{\cdot,j}||$ is a weighted sum of all pairwise distances of the columns, and $\Omega_{\widetilde{\mathbf{W}}}(\widetilde{\mathbf{X}}^T)$ is a weighted sum of all pairwise distances of the rows. Compared with sparseBC and OEMCBS, COBRA has two main advantages. Firstly, it is a convex problem that has a unique global minimizer, which can be obtained by applying the alternating direction method of multipliers (ADMM). Secondly, COBRA provides a solution path of the partition of rows and columns. As λ increases gradually from a small to a big value, the existing clusters of rows and columns will merge gradually as well and form subgroups of rows and columns with different sizes. The limitation of COBRA is that the ADMM algorithm is computationally slow and not scalable to large matrices.

3.2.3 SVD based Methods

The mathematical basis of several existing SVD based biclustering methods is the multiplicative model Eq. (3.3). These methods include “Rank-one Biclusters" (RoBiC) [54], “Sparse SVD" (SSVD) [55], “sparse reduced rank regression" (SRRR) [56], and so on.

RoBiC [54] finds each bicluster using a two-step procedure. In the first step, it computes the best rank-one approximation $\mathbf{u}\mathbf{v}^T$ of the matrix $\mathbf{X}$ by using SVD. In the second step, RoBiC sorts the elements of $\mathbf{u}$ and fits a hinge function to them to identify the rows that belong to the bicluster, and do the same operation on $\mathbf{v}$ to identify the columns that belong to the bicluster. Let $\mathbf{u}_{sel}$ denote the vector obtained by replacing with zero those elements of $\mathbf{u}$ that do not belong to the bicluster, and define $\mathbf{v}_{sel}$ similarly. The residual matrix $\mathbf{X}_{res} = \mathbf{X} - \mathbf{u}_{sel}\mathbf{v}_{sel}^T$ removes the effect of the identified bicluster for the data matrix $\mathbf{X}$. To find other biclusters, the two-steps can be repeated sequentially on the residual matrices after removing the effect of the bicluster found in the previous iteration. RoBiC does not directly solve an optimization problem and can not be automated.

SSVD [55] also finds the biclusters in a sequential manner. To find each bicluster, SSVD solves the following optimization problem:

$$\min_{s,\mathbf{u},\mathbf{v},\|\mathbf{u}\|_2=\|\mathbf{v}\|_2=1} ||\mathbf{X} - s\mathbf{u}\mathbf{v}^T||_F^2 + s\lambda_u \sum_{i=1}^{n} w_{1,i}|u_i| + s\lambda_v \sum_{j=1}^{p} w_{2,j}|v_j|, \tag{3.4}$$

where s is the scale parameter, λ_u and λ_v are non-negative penalty parameters, and $w_{1,i}$ and $w_{2,j}$ are suitably chosen weights as used in the adaptive Lasso. After each bicluster is found, its effect can be removed from the data matrix similarly as in RoBiC to form the new data matrix. Then, the optimization problem Eq. (3.4) is solved to obtain the next bicluster. SSVD solves Eq. (3.4) through an iterative algorithm. It starts from the first pair of singular vectors obtained by SVD. In each iteration, $\mathbf{v}$ and s are updated with $\mathbf{u}$ fixed at the value from the previous iteration, and then $\mathbf{u}$ and s are updated with $\mathbf{v}$ fixed at its most recent update. Later, [57] develops a stable version of SSVD called S4VD by incorporating the stability selection, a subsampling-based variable selection technique.

There are also several methods related to SSVD. [58] proposes "Penalized Matrix Decomposition" (PMD), which is similar to SSVD. Instead of adding penalty terms, PMD solves the biclustering problem via adding constraints on the support of $\mathbf{u}$ and $\mathbf{v}$. [62] solves biclustering problem of binary matrices by changing the square loss function in Eq. (3.4) to the negative of a

Bernoulli log-likelihood function.

SRRR [56] is designed for sparse reduced-rank regressions and, as a special application, can be used for solving the biclustering problem. When specialized to finding one bicluster in the data matrix $\mathbf{X}$, it solves the following optimization problem

$$\min_{s,\mathbf{u},\mathbf{v},\|\mathbf{u}\|_2=\|\mathbf{v}\|_2=1} \|\mathbf{X} - s\mathbf{u}\mathbf{v}^T\|_F^2 + \lambda|s| \sum_{i=1}^{n}\sum_{j=1}^{p} w_{i,j}|u_i|\,|v_j|, \tag{3.5}$$

where $w_{i,j}$ are suitably defined weights. Because of the penalty term, the optimal $\mathbf{u}$ and $\mathbf{v}$ are sparse vectors, whose non-zero entries correspond to rows and columns included in a bicluster. Note that Eq. (3.5) is a similar penalized optimization formulation as Eq. (3.4) but with a different penalty function. Instead of sequentially identifying the biclusters, [56] develops an iterative exclusive extraction algorithm to extract all biclusters in parallel. Specifically, to identify r biclusters, we consider a multiplicative model as shown in Eq. (3.3). One initially starts from a rank-r approximation of $\mathbf{X}$, denoted as

$$\widetilde{\mathbf{X}}^{(0)} = \sum_{k=1}^{r} \widetilde{\mathbf{X}}_k^{(0)} = \sum_{k=1}^{r} s_k^{(0)}\mathbf{u}_k^{(0)}\mathbf{v}_k^{(0)},$$

then solves the optimization problem Eq. (3.5) with $\mathbf{X}$ replaced by $\mathbf{X} - \sum_{k'\neq k}\widetilde{\mathbf{X}}_{k'}^{(0)}$ to obtain an update of $\mathbf{X}_k$, $k = 1, \ldots, r$. The r optimization problems can be solved in parallel. These steps are iterated until convergence is reached.

3.2.4 NMF based Method

The non-negativity constraint of the parameters is common in many applications of biclustering. “NMF with sparseness constraints" (NMFSC) [63] finds the sparse decomposition of $\mathbf{X}$ with each entry of $\mathbf{U}$ and $\mathbf{V}$ constrained to be non-negative. To seek the sparsity in the columns of $\mathbf{U}$ and $\mathbf{V}$, they build up a sparseness measurement based on the relationship between the ℓ_1 norm and

the ℓ_2 norm. For any vector $\mathbf{x} \in \mathbb{R}^n$, the sparseness of $\mathbf{x}$ is defined as:

$$\mathcal{S}(\mathbf{x}) = \frac{\sqrt{n} - (\sum_{i=1}^{n} |x_i|)/\sqrt{\sum_{i=1}^{n} x_i^2}}{\sqrt{n} - 1}. \tag{3.6}$$

The optimization problem of NMFSC is:

$$\min_{\mathbf{U},\mathbf{V}} ||\mathbf{X} - \mathbf{U}\mathbf{V}^T||, \tag{3.7}$$

subject to

$$\mathcal{S}(\mathbf{u}_i) = t_u,\ \mathcal{S}(\mathbf{v}_i) = t_v,\ \mathbf{U} \geq 0,\ \mathbf{V} \geq 0.$$

where t_u and t_v are two tuning parameters, and $\mathbf{U} \geq 0$ denotes $u_{ik} \geq 0$ for all i, k, and the same interpretation is applied to $\mathbf{V} \geq 0$. Notice that NMFSC is not designed for the biclustering problem, and thus the constraints on $\mathcal{S}(\mathbf{x})$ may not fit the biclustering problem well. We can solve Eq. (3.7) by iteratively updating $\mathbf{U}$ and $\mathbf{V}$ through the gradient descent while keeping projecting each column of $\mathbf{U}$ and $\mathbf{V}$ to the non-negative space.

3.2.5 Plaid

Plaid [53] is perhaps the first method to use matrix decomposition for biclustering. Plaid assumes that non-zero entries of each $\mathbf{A}_k$ in model Eq. (3.1) follow the additive structure $\theta_{i,j} = \alpha_i + \beta_j$, where i, j are indices for the row and the column. Similar to SSVD, Plaid sequentially seeks the latent subgroups and minimizes the following loss function for each bicluster:

$$\begin{aligned} L_{\text{Plaid}} = &\sum_{i=1}^{n}\sum_{j=1}^{p}(X_{ij} - \theta_{ij}\rho_i\kappa_j)^2, \\ &\text{subject to } \theta_{ij} = \alpha_i + \beta_j, \end{aligned} \tag{3.8}$$

where ρ_i and κ_j are the binary indicator variables to reflect inclusion of i^{th} row and j^{th} column in the bicluster, and α_i and β_j are the effects of i^{th} row and j^{th} column. After each bicluster is discovered, when minimizing Eq. (3.8) for the next bicluster, X_{ij} are replaced by the residuals

after removing the effects of the discovered biclusters. The Plaid objective function Eq. (3.8) can be minimized using an iterative algorithm where each cycle updates in turn the θ values, the ρ values, and the κ values. During the iteration process, ρ_i and κ_j are allowed to have continuous values as a relaxation, and they are only forced to take 0-1 values in the last few iterations.

Interestingly, the Plaid model can be viewed as a special case of the multiplicative model Eq. (3.3) where $\mathbf{U}$ and $\mathbf{V}$ have a specific structure. In fact, by constructing the two matrices:

$$\mathbf{U} = \begin{pmatrix} \alpha_1\rho_1 & \rho_1 \\ \alpha_2\rho_2 & \rho_2 \\ \vdots & \vdots \\ \alpha_n\rho_n & \rho_n \end{pmatrix} \text{ and } \mathbf{V} = \begin{pmatrix} \kappa_1 & \beta_1\kappa_1 \\ \kappa_2 & \beta_2\kappa_2 \\ \vdots & \vdots \\ \kappa_p & \beta_p\kappa_p \end{pmatrix},$$

we can rewrite Eq. (3.8) into:

$$L_{\text{Plaid}} = ||\mathbf{X} - \mathbf{U}\mathbf{V}^T||_F^2, \tag{3.9}$$

which is the least squares criterion for fitting model Eq. (3.3) when $\mathbf{U}$ and $\mathbf{V}$ are restricted to the specific forms given above.

3.3 Proposed Method

In the section, we will develop a new biclustering method based on multiplicative matrix decomposition. Our new method improves over the SVD based methods reviewed in the previous section by using a novel penalty that induces sparsity on each composition vector of the matrix decomposition. Our method enjoys the same advantage of allowing overlapping biclusters as other SVD based methods. Section 3.3.1 formulates the optimization problem for our method. Section 3.3.2 presents a computational algorithm. Section 3.3.3 discusses how to handle missing data. Section 3.3.4 provides a method for tuning parameter selection.

3.3.1 Biclustering with the Squared $\ell_{1,2}$-norm Penalty

Consider the multiplicative matrix decomposition model Eq. (3.3), where $\mathbf{X}$ is decomposed

as the summation of r rank-1 matrices $\mathbf{u}_k\mathbf{v}_k^T$ plus an error term. Since nonzero elements of $\mathbf{u}_k$ and $\mathbf{v}_k$ indicate selection of rows or columns of the k^{th} bicluster, sparsity in all pairs of $\mathbf{u}_k$ and $\mathbf{v}_k$ implies a biclustering structure. We minimize the squared reconstruction error $\|\mathbf{X}-\sum_{k=1}^r \mathbf{u}_k\mathbf{v}_k^T\|^2$ together with a suitable penalty term that encourages sparsity in each pair of $\mathbf{u}_k$ and $\mathbf{v}_k$, giving the biclustering structure. Precisely, we minimize the following loss function:

$$\begin{aligned} L(\mathbf{U},\mathbf{V}) &= ||\mathbf{X}-\mathbf{U}\mathbf{V}^T||_F^2 + \lambda_1 P_{1,2}(\mathbf{U}) + \lambda_2 P_{1,2}(\mathbf{V}), \\ P_{1,2}(\mathbf{U}) &= \sum_{k=1}^{r}(\sum_{i=1}^{n}|u_{ik}|)^2, \quad P_{1,2}(\mathbf{V}) = \sum_{k=1}^{r}(\sum_{j=1}^{p}|v_{jk}|)^2, \end{aligned} \tag{3.10}$$

where λ_1, λ_2 are penalty parameters.

In this formulation, the ℓ_1-norm penalty is applied on both $\mathbf{u}_k$ and $\mathbf{v}_k$, i.e., $\|\mathbf{u}_k\|_1 = \sum_{i=1}^n |u_{ik}|$, $\|\mathbf{v}_k\|_1 = \sum_{j=1}^p |v_{jk}|$, and the squared ℓ_2-norm is used to combine these ℓ_1-norms. The penalty $P_{1,2}(\cdot)$ is called the squared $\ell_{1,2}$-norm penalty or the exclusive Lasso penalty in the literature, which was proposed previously for the multi-task feature selection problem [64]. Because of use of this penalty, we shall refer to our method as **Bi**-**C**lustering with the **E**xclusive **L**asso penalty (BCEL).

The squared $\ell_{1,2}$-norm penalty is a composite penalty that respects the group structure among the parameters [65]. Particularly, it facilitates element selection within each group. [66] studied the statistical properties of the penalized least-squares estimator using the squared $\ell_{1,2}$ penalty in linear regression problems. In our loss function Eq. (3.10), the elements in $\mathbf{u}_k$ or $\mathbf{v}_k$, $k = 1, \ldots, r$, are treated as one group, and application of the ℓ_1-norm penalty on $\mathbf{u}_k$ or $\mathbf{v}_k$, $k = 1, \ldots, r$, performs element selection within each group. Using the squared ℓ_2-norm to combine the ℓ_1-norms ensures that the selection competition is within each group. Note that it is not desirable to use the ℓ_1-norm to combine the ℓ_1-norms, i.e., to use $\|\mathbf{U}\|_1 = \sum_{k=1}^r \sum_{i=1}^n |u_{ik}|$ and $\|\mathbf{V}\|_1 = \sum_{k=1}^r \sum_{j=1}^p |v_{jk}|$ as the penalty function. When doing this, the group structure in $\mathbf{U}$ and $\mathbf{V}$ will be lost.

The proposed BCEL has several advantages compared to the matrix decomposition based methods reviewed in the previous section. First, BCEL discovers multiple biclusters simultaneously,

while most existing SVD methods are sequential biclustering methods. Although the sequential methods are easy to implement, the latter biclusters may be highly influenced by the former biclusters since the error accumulates during the algorithm. Therefore these methods may not give good results in practice. We will show in Section 3.5 that BCEL significantly improves over the clustering performance of SSVD in the analysis of the single-cell RNA sequencing dataset. Second, BCEL completes parameter selection and estimation in one step and achieves more stable results than the two-step methods such as RoBiC. Third, as discussed in Section 3.2, BCEL is built on the multiplicative decomposition model and allows overlapping biclusters. Thus it can outperform the block decomposition models such as sparseBC when there is overlapping among biclusters. We shall demonstrate the advantage of BCEL in a comparative study to be presented in Section 3.4.

3.3.2 Algorithm

3.3.2.1 Outline

Since $L(\mathbf{U}, \mathbf{V})$ in Eq. (3.10) includes a multiplicative form of two parameter matrices $\mathbf{U}$ and $\mathbf{V}$, $L(\mathbf{U}, \mathbf{V})$ is not a convex function of $(\mathbf{U}, \mathbf{V})$. However, when fixing $\mathbf{V}$, $L(\mathbf{U}, \mathbf{V})$ is a convex function of $\mathbf{U}$, and when fixing $\mathbf{U}$, $L(\mathbf{U}, \mathbf{V})$ is a convex function of $\mathbf{V}$. This suggests an iterative algorithm that alternately optimizes over $\mathbf{U}$ and $\mathbf{V}$.

Specifically, when $\mathbf{V}$ is fixed, minimizing $L(\mathbf{U}, \mathbf{V})$ is equivalent to:

$$\min_{\mathbf{U} \in \mathbb{R}^{n \times r}} ||\text{vec}(\mathbf{X}^T) - \mathbf{D}_{\mathbf{V}} \cdot \text{vec}(\mathbf{U}^T)||_2^2 + \lambda_1 P_{1,2}(\mathbf{U}) + c_{\mathbf{V}} \tag{3.11}$$

with

$$\mathbf{D}_{\mathbf{V}} = \begin{pmatrix} \mathbf{V} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{V} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{V} \end{pmatrix} \in \mathbb{R}^{np \times nr},$$

and $c_{\mathbf{V}}$ is a constant determined only by $\mathbf{V}$. Clearly, (3.11) is a penalized linear regression problem,

where $\text{vec}(\mathbf{X}^T)$ is the response vector, $\mathbf{D_V}$ is the design matrix, $\text{vec}(\mathbf{U}^T)$ is the coefficient vector to be estimated, and $P_{1,2}(\mathbf{U})$ is the squared $\ell_{1,2}$-norm penalty used in Eq. (3.10). This is a convex problem that can be efficiently solved.

Likewise, when $\mathbf{U}$ is fixed, minimizing $L(\mathbf{U}, \mathbf{V})$ becomes:

$$\min_{\mathbf{V}\in\mathbb{R}^{p\times r}} ||\text{vec}(\mathbf{X}) - \mathbf{D_U} \cdot \text{vec}(\mathbf{V}^T)|_2^2 + \lambda_2 P_{1,2}(\mathbf{V}) + c'_{\mathbf{U}}, \tag{3.12}$$

where

$$\mathbf{D_U} = \begin{pmatrix} \mathbf{U} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{U} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{U} \end{pmatrix} \in \mathbb{R}^{np\times pr},$$

and $c'_{\mathbf{U}}$ is a constant only related to $\mathbf{U}$. Eq. (3.12) is a penalized linear regression problem with the same form as Eq. (3.11) and can be solved in the same way.

The above discussion leads to Algorithm 2, which is an alternating minimization algorithm. It is easy to see that in each iteration, $L(\mathbf{U}^{t+1}, \mathbf{V}^{t+1}) \leq L(\mathbf{U}^{t+1}, \mathbf{V}^t) \leq L(\mathbf{U}^t, \mathbf{V}^t)$. According to Lemma 3.2 proposed by [67], the accumulation point of the sequence $\{(\mathbf{U}^t, \mathbf{V}^t)\}$ generated by the alternating minimization is a stationary point of the problem, and thus Algorithm 2 can converge to a stationary point.

We shall discuss next how to solve the penalized regression problems Eq. (3.11) and Eq. (3.12).

Algorithm 2 Alternating Minimization for BCEL

Input: $\mathbf{X}$, r, λ_1, λ_2, $\mathbf{U}^0 \in \mathbb{R}^{n\times r}$, and $\mathbf{V}^0 \in \mathbb{R}^{p\times r}$.
Iterate until convergence: **For** $t = 0, 1, 2, \ldots$:

1. Obtain $\mathbf{U}^{t+1}$ by solving Eq. (3.11) with $\mathbf{V}$ fixed at $\mathbf{V}^t$.
2. Obtain $\mathbf{V}^{t+1}$ by solving Eq. (3.12) with $\mathbf{U}$ fixed at $\mathbf{U}^{t+1}$.

3.3.2.2 *Regression with the Squared $\ell_{1,2}$-norm Penalty*

Both Eq. (3.11) and Eq. (3.12) are penalized linear regression problems with the squared $\ell_{1,2}$-norm penalty and can be solved using the same algorithm. To simplify notation, we use the standard form of the penalized linear regression problem to present our algorithm. Let $\mathbf{y} \in \mathbb{R}^n$ be the response vector of sample size n, and $\mathbf{X} \in \mathbb{R}^{n \times p}$ be the $n \times p$ design matrix corresponding to sample size n and p features. It is assumed that the p features are pre-classified into r disjoint groups $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_r$, $\cup \mathcal{G}_i = \{1, 2, \ldots, p\}$, and $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset$ for any $i \neq j$. The penalized least-squares regression problem with the squared $\ell_{1,2}$-norm penalty minimizes the following objective function:

$$f(\mathbf{b}) = ||\mathbf{y} - \mathbf{X}\mathbf{b}||_2^2 + \lambda \sum_{i=1}^{r} (\sum_{j \in \mathcal{G}_i} |b_j|)^2, \tag{3.13}$$

where b_j is the j^{th} entry of the coefficient vector $\mathbf{b} \in \mathbb{R}^p$. It is obvious that Eq. (3.11) and Eq. (3.12) are special forms of Eq. (3.13). Although $f(\mathbf{b})$ in Eq. (3.13) is a convex function of $\mathbf{b}$, the presence of the absolute value function makes $f(\mathbf{b})$ non-differentiable and complicates its optimization. Several algorithms have been proposed in the literature to minimize $f(\mathbf{b})$. [68] developed an iterative re-weighted algorithm, where in each iteration, $\mathbf{b}$ is updated by solving a system of p linear equations. The algorithm has a computational complexity of $O(p^3)$, making it not efficient for big p. [66] developed a cyclic coordinate descent algorithm, updating the entries of $\mathbf{b}$ one-by-one. This algorithm may have slow convergence when p is big, and it is not suited for parallel computing.

As a result, we provide an efficient iterative algorithm to minimize $f(\mathbf{b})$ by using the proximal gradient method [69], which combines the ideas of the Majorization-Minimization (MM) algorithm [35] and the proximal operator. Our algorithm updates $\mathbf{b}$ in blocks, i.e., it updates the elements of $\mathbf{b}$ in each group $\mathcal{G}_i$ together. The computation of updates of these groups is also parallelizable.

Applying the MM algorithm to minimize $f(\mathbf{b})$, we need to define a majorizing function $g(\mathbf{b}|\mathbf{b}^*)$ that satisfies $g(\mathbf{b}|\mathbf{b}^*) \geq f(\mathbf{b})$ and $g(\mathbf{b}^*|\mathbf{b}^*) = f(\mathbf{b}^*)$. The MM algorithm iteratively updates

$\mathbf{b}$ by minimizing $g(\mathbf{b}|\mathbf{b}^*)$ with $\mathbf{b}^*$ being the value of $\mathbf{b}$ obtained in the previous step. To define a majorizing function, we denote L as the Lipschitz constant of the function $h(\mathbf{b}) = \nabla\|\mathbf{y}-\mathbf{X}\mathbf{b}\|_2^2 = \mathbf{X}^T\mathbf{X}\mathbf{b} - \mathbf{X}^T\mathbf{y}$, which means that $|h(\mathbf{b}^\flat) - h(\mathbf{b}^\sharp)| \leq L||\mathbf{b}^\flat - \mathbf{b}^\sharp||_2$ for any $\mathbf{b}^\flat, \mathbf{b}^\sharp \in \mathbb{R}^p$. The value of L is not unique, and a proper value of L is two times of the largest eigenvalue of $\mathbf{X}^T\mathbf{X}$ [36]. It is easily seen that the following is a majorizing function of $f(\mathbf{b})$ in Eq. (3.13):

$$\begin{aligned} g(\mathbf{b}|\mathbf{b}^*) \coloneqq & ||\mathbf{y} - \mathbf{X}\mathbf{b}^*||_2^2 + 2(\mathbf{X}^T\mathbf{X}\mathbf{b}^* - \mathbf{X}^T\mathbf{y})^T(\mathbf{b} - \mathbf{b}^*) \\ & + L||\mathbf{b} - \mathbf{b}^*||_2^2 + \lambda\sum_{i=1}^{r}(\sum_{j\in\mathcal{G}_i}|b_j|)^2. \end{aligned}$$

Suppose $\mathbf{b}^t$ is the value of $\mathbf{b}$ at step t of the iteration. The MM algorithm obtains the update $\mathbf{b}^{t+1}$ by minimizing $g(\mathbf{b}|\mathbf{b}^t)$ over $\mathbf{b}$ at the step $t+1$. By some algebra, we see that

$$\mathbf{b}^{t+1} = \arg\min_{\mathbf{b}} ||\mathbf{b}^t - \frac{1}{L}(\mathbf{X}^T\mathbf{X}\mathbf{b}^t - \mathbf{X}^T\mathbf{y}) - \mathbf{b}||_2^2 + \frac{\lambda}{L}\sum_{i=1}^{r}(\sum_{j\in\mathcal{G}_i}|b_j|)^2. \tag{3.14}$$

By setting $\mathbf{z} = \mathbf{b}^t - \frac{1}{L}(\mathbf{X}^T\mathbf{X}\mathbf{b}^t - \mathbf{X}^T\mathbf{y})$ and $\lambda^* = \lambda/L$, the right hand size of Eq. (3.14) can be recognized as a proximal operator associated with the squared $\ell_{1,2}$-norm:

$$\text{prox}_{EL}(\mathbf{z}, \lambda^*) = \arg\min_{\mathbf{b}} ||\mathbf{z} - \mathbf{b}||_2^2 + \lambda^*\sum_{i=1}^{r}(\sum_{j\in\mathcal{G}_i}|b_j|)^2. \tag{3.15}$$

Since the pre-defined groups $\mathcal{G}_1, \ldots, \mathcal{G}_r$ are disjoint in our problem, the objective function of $\text{prox}_{EL}(\cdot)$ can be regarded as a summation of the functions of parameters in different groups. Thus we can split Eq. (3.15) into r smaller problems:

$$\text{prox}_{EL,i}(\mathbf{z}_{\mathcal{G}_i}, \lambda^*) = \arg\min_{\mathbf{b}_{\mathcal{G}_i}} \sum_{j\in\mathcal{G}_i}(z_j - b_j)^2 + \lambda^*(\sum_{j\in\mathcal{G}_i}|b_j|)^2, \tag{3.16}$$

where $\mathbf{b}_{\mathcal{G}_i}$ is the sub-vector of $\mathbf{b}$ that includes the parameters for $\mathcal{G}_i$, and $\mathbf{z}_{\mathcal{G}_i}$ denotes the sub-vector of $\mathbf{z}$ that contains the entries corresponding to $\mathbf{b}_{\mathcal{G}_i}$. Each problem in Eq. (3.16) is a proximal operator associated with the squared Lasso penalty and is a convex problem. Unlike the proximal

operator associated with the Lasso penalty, this proximal operator does not have a closed-form solution. We shall develop a computational algorithm to compute this proximal operator.

Consider the following standard form of the proximal operator associated with the squared Lasso penalty:

$$\text{prox}_{SL}(\mathbf{z}, \lambda) = \arg\min_{\mathbf{b}} ||\mathbf{z} - \mathbf{b}||_2^2 + \lambda||\mathbf{b}||_1^2, \ \mathbf{z} \in \mathbb{R}^D. \tag{3.17}$$

We solve Eq. (3.17) by using a method similar to the algorithm proposed by [70]. Our solution is derived in three steps: (1) We provide an analytic form of $\text{prox}_{SL}(\mathbf{z}, \lambda)$ using the KKT conditions when the non-zero index set of $\mathbf{b}$ is assumed known. (2) We show that there are only $D + 1$ candidates for the non-zero index set of $\text{prox}_{SL}(\mathbf{z}, \lambda)$. (3) We provide a way to identify the true non-zero index set form the $d + 1$ candidates.

We first introduce some notations. Denote $\hat{\mathbf{b}} = \text{prox}_{SL}(\mathbf{z}, \lambda^*)$. Let $\mathcal{S}$ be the non-zero index set of $\hat{\mathbf{b}}$, and $l(\mathbf{b}) = \|\mathbf{z} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{b}\|_1^2$. Let $a_1, a_2, \ldots, a_D$ be the permutation of the index $1, 2, \ldots, D$ such that $|z_{a_1}| \geq |z_{a_2}| \geq \cdots \geq |z_{a_D}|$. We define a list of strictly monotonous index sets $\mathcal{S}_0 \subset \mathcal{S}_1 \subset \cdots \subset \mathcal{S}_D$ that $\mathcal{S}_0 = \emptyset$ and $\mathcal{S}_d = \{a_1, \ldots, a_d\}$ for $d = 1, 2, \ldots, D$.

Now we provide details of the three steps of deriving the solution of Eq. (3.17). First, if $\mathcal{S}$ is known, we can obtain an analytic form of $\hat{\mathbf{b}}$ by investigating the KKT conditions of Eq. (3.17). This is given in the following Lemma.

Lemma 3. *By KKT conditions, for each $i \in \mathcal{S}$, the solution of b_i satisfies the following equation:*

$$-2(z_i - b_i) + 2\lambda||\mathbf{b}||_1 \text{sign}(b_i) = 0. \tag{3.18}$$

Consequently, we have:

$$\hat{b}_i = \begin{cases} \textit{sign}(z_i)\big[|z_i| - \frac{\lambda}{1+\lambda|S|}(\sum_{i\in S}|z_i|)\big], & i \in \mathcal{S}, \\ 0, & i \notin \mathcal{S}. \end{cases} \tag{3.19}$$

Second, we investigate the properties of $\hat{\mathbf{b}}$, and the results are given in Lemma 4.

Lemma 4. *The optimal solution* $\hat{\mathbf{b}}$ *satisfies the following properties:*

(i) The sign of $\hat{b}_i$ *must be the same as that of* z_i *unless* $\hat{b}_i = 0$*;*

(ii) $|\hat{b}_i| \leq |z_i|$*;*

(iii) If $|z_i| > |z_j|$*, then* $|\hat{b}_i| \geq |\hat{b}_j|$.

From Lemma 4(iii), it is easy to see that $\mathcal{S}$ can only be chosen from $D+1$ candidates $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_D$.

Third, $\mathcal{S}$ can be determined using the next Lemma.

Lemma 5. $\mathcal{S}$ *can be found by solving the following optimization problem:*

$$\mathcal{S} = \arg\max_{\mathcal{S}_d} d, \text{ subject to } |z_{a_d}| > \frac{\lambda}{1+\lambda d} \sum_{i \in \mathcal{S}_d} |z_i| \text{ or } d = 0. \tag{3.20}$$

The proofs of Lemmas 3–5 are given in Section 3.6.3. These results suggest the following algorithm for computing $\text{prox}_{SL}(\mathbf{z}, \lambda)$: First find $\mathcal{S}$ of the optimal solution via Eq. (3.20) and then calculate $\hat{\mathbf{b}}$ by Eq. (3.19).

Combining the algorithm of computing the proximal operator with the MM algorithm, we arrive at an algorithm for solving the penalized least-squares problem Eq. (3.13). This algorithm is presented in Algorithm 3.

Now we discuss the computational complexity of Algorithm 3. First, Algorithm 3 requires one time calculation of $\mathbf{X}^T\mathbf{y}$ and $\mathbf{X}^T\mathbf{X}$, whose complexity is $O(np^2)$. For each iteration, step 1 of Algorithm 3 requires $O(p)$ calculations to obtain $\mathbf{z}$. Step 2 solves r proximal operators. Set $D_i = |\mathcal{G}_i|$, and then $\sum_{i=1}^r D_i = p$. For the proximal operator associated with $\mathcal{G}_i$, it requires $O(D_i \log(D_i))$ to order the elements in $\mathbf{z}_{\mathcal{G}_i}$, $O(D_i)$ to solve Eq. (3.20), and $O(D_i)$ to calculate Eq. (3.19). On the other hand, [36] shows that the proximal gradient algorithm needs $O(1/\epsilon)$ iterations to achieve ϵ tolerance of convergence error. To summarize, the overall computational complexity of Algorithm 3 is at most $O(p \log(\max_i D_i)/\epsilon + np^2)$. In practice, one can further

Algorithm 3 Regression with the Squared $\ell_{1,2}$-norm Penalty

Input: $\mathbf{X}$, $\mathbf{y}$, $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_r$, $\mathbf{b}^0$, λ, and L.
Iterate until convergence: **For** $t = 0, 1, 2, \ldots$, update $\mathbf{b}^{t+1}$ from $\mathbf{b}^t$ by:

1. $\mathbf{z} = \mathbf{b}^t - \frac{1}{L}(\mathbf{X}^T\mathbf{X}\mathbf{b}^t - \mathbf{X}^T\mathbf{y})$
2. Update $\mathbf{b}^{t+1} = prox_{EL}(\mathbf{z}, \lambda/L)$ by solving r proximal operators Eq. (3.16).
   For each group $\mathcal{G}_i$, solve $\hat{\mathbf{b}}_{\mathcal{G}_i} = prox_{SL}(\mathbf{z}_{\mathcal{G}_i}, \lambda/L)$:
   (a) Order $|z_1|, |z_2|, \ldots, |z_{D_i}|$ in $\mathbf{z}_{\mathcal{G}_i}$ and construct the corresponding list of strictly monotonous index sets $\mathcal{S}_0 \subset \mathcal{S}_1 \subset \cdots \subset \mathcal{S}_{D_i}$.
   (b) Solve Eq. (3.20) to get the non-zero index set $\mathcal{S}$.
   (c) Calculating $\hat{\mathbf{b}}_{\mathcal{G}_i}$ using Eq. (3.19) with $\mathcal{S}$ given in b).
3. Concatenate $\hat{\mathbf{b}}_{\mathcal{G}_i}$ to obtain $\mathbf{b}^{t+1}$.

parallel the calculation of matrix multiplication and r proximal operators in step 2 to reduce the computing time of Algorithm 3.

3.3.3 BCEL with Missing Data

The problem of missing data is a common occurrence in real-world applications, in which some entries of the data matrix are not observed in the study. Our algorithms are readily applicable when the data matrix has entries missing completely at random. Let $\mathbf{\Omega} = (\omega_{i,j}) \in \{0, 1\}^{n \times p}$ be the indicator matrix of missing entries. If the $(i, j)^{\text{th}}$ entry of $\mathbf{X}$ is observed, $\omega_{ij} = 1$; otherwise, $\omega_{ij} = 0$, indicating that the $(i, j)^{\text{th}}$ entry of $\mathbf{X}$ is missing.

To describe how to apply our algorithms, we can arbitrarily fill in a value (say, 0) at missing entries of $\mathbf{X}$. To handle missing data, we revise the original loss function of Eq. (3.10) to:

$$L(\mathbf{U}, \mathbf{V}|\mathbf{\Omega}) = ||\mathbf{\Omega} \circ (\mathbf{X} - \mathbf{U}\mathbf{V}^T)||_F^2 + \lambda_1 P_{1,2}(\mathbf{U}) + \lambda_2 P_{1,2}(\mathbf{V}), \tag{3.21}$$

where $\circ$ denotes the element-wise product of two matrices. The only difference between Eq. (3.10) and Eq. (3.21) are at the squared loss term. When missing data are present in $\mathbf{X}$, we only calculate

the squared errors where the entries of $\mathbf{X}$ are observed. The alternating minimization algorithm, Algorithm 2, is still applicable for minimizing Eq. (3.21) by only using available entries of $\mathbf{X}$. Similarly, Algorithm 3 can also be applied without any issues.

3.3.4 Stability Selection of Tuning Parameters and Bicluster Membership

The number of biclusters r, the row penalty parameter λ_1, and the column penalty parameter λ_2 are three tuning-parameters in our method BCEL. Following a common practice used method for selecting rank numbers in penalized matrix decomposition [71], we use the cross-validation (CV) to select r.

It is more involved to come up with a good selection of λ_1 and λ_2. We can restrict attention to their selection for fixed r. Two commonly used methods for tuning parameter selection are Information Criterion (IC) and CV. However, we found that neither method works well for BCEL. For the IC method, it is difficult to define an effective IC in the presence of two penalty parameters. In practice, CV tends to select a relatively small value for λ_1 and λ_2, resulting in redundant rows or columns being included into biclusters. We propose to adopt the idea of the stability selection [72] to decide on the two penalty parameters, and more importantly, we use stability selection to make a stable selection of the rows and columns included in the biclusters. The stability selection method was designed for variable selection problems with one penalty parameter, and we need to extend it to handle our bicluster problem with two penalty parameters.

3.3.4.1 Traditional Stability Selection

We first give a brief introduction to the stability selection. The stability selection is a variable selection scheme proposed by [72] that uses subsampling to determine the amount of regularization, as well as controlling the rate of falsely selected variables (which is also called the Type I error rate in multiple testing) for finite sample size.

The traditional stability selection is defined on the variable selection problem with only one tuning parameter $\lambda \in \Lambda$, where Λ is the set of possible values of λ. Suppose that there are p parameters to be estimated in the model, and the index set of the true non-zero parameters is $\mathcal{S}^*$.

The stability selection scheme tries to estimate $\mathcal{S}^*$ by subsampling. Use $\mathcal{I}$ to denote a randomly subsampled set of the original data containing 50% samples. With B times of subsampling, we obtain B different subsampled data sets $\mathcal{I}_1, \ldots, \mathcal{I}_B$. Let $\hat{\mathcal{S}}^\lambda(\mathcal{I}_k)$ denote the index set of variables that are selected when we solve the problem using the data $\mathcal{I}_k$ and the tuning parameter λ. Then, for each variable i, we can estimate the probability that variable i is included using randomly subsampled set by the relative frequency of i in $\hat{\mathcal{S}}^\lambda(\mathcal{I}_k)$, $k = 1, \ldots, B$:

$$\hat{P}(i \in \hat{\mathcal{S}}^\lambda(\mathcal{I})) = \frac{1}{B} \sum_{k=1}^{B} I(i \in \hat{\mathcal{S}}^\lambda(\mathcal{I}_k)). \tag{3.22}$$

Given a threshold $\pi \in (0.5, 1)$ and Λ, the index set of non-zero parameters selected by the stability selection is defined as

$$\hat{\mathcal{S}}^{\text{stable}} = \{i : \max_{\lambda \in \Lambda} \hat{P}(i \in \hat{\mathcal{S}}^\lambda(\mathcal{I})) \geq \pi, i \in \{1, 2, \cdots, p\}\}. \tag{3.23}$$

Set $\hat{\mathcal{S}}^\Lambda(\mathcal{I}) = \cup_{\lambda \in \Lambda} \hat{\mathcal{S}}^\lambda(\mathcal{I})$. Denote $q_\Lambda = E|\hat{\mathcal{S}}^\Lambda(\mathcal{I})|$, which is the average number of selected variables for a randomly subsampled set. $\mathcal{V} = (\mathcal{S}^*)^C \cap \hat{\mathcal{S}}^{\text{stable}}$ is the set of falsely selected variables for the stability selection. Under the assumptions that the distribution of $\{I(i \in \hat{\mathcal{S}}^\lambda, i \in \mathcal{S}^C)\}$ is exchangeable where $\hat{\mathcal{S}}^\lambda$ is the selected index set with λ on the whole dataset and that the original model is not worse than random guessing, [72] shows in their Theorem 1 that the expected number of the falsely selected variables ($E(|\mathcal{V}|)$) is bounded by

$$E(|\mathcal{V}|) \leq \frac{1}{2\pi - 1} \frac{q_\Lambda^2}{p}, \tag{3.24}$$

Suppose one wants to control the expected falsely selection rate, $E(|\mathcal{V}|)/p$, to be no more than r_f. Through bounding the right hand side of Eq. (3.24) by r_f, it is sufficient to have $q_\Lambda \leq p\sqrt{(2\pi - 1)r_f}$. Then Λ can be determined to satisfy this condition. [72] finds that the results of stability selection is not sensible to the selection of π in the range of $(0.6, 0.9)$.

In a particular case when the variable selection result of the model is monotonous and contin-

uous with respect to the penalty parameter, that is, for any two values $\lambda_a, \lambda_b \in \Lambda$ with $\lambda_a < \lambda_b$, we have $\hat{P}(i \in \mathcal{S}^{\lambda_a}(\mathcal{I})) \geq \hat{P}(i \in \mathcal{S}^{\lambda_b}(\mathcal{I}))$. Then $q_\Lambda = E|\hat{\mathcal{S}}^{\Lambda}(\mathcal{I})| = E|\hat{\mathcal{S}}^{\lambda_{\min}}(\mathcal{I})|$ only depends on $\lambda_{\min} = \inf \Lambda$ and we only need to determine $\lambda_{\min}$ such that $E|\hat{\mathcal{S}}^{\lambda_{\min}}(\mathcal{I})| \leq p\sqrt{(2\pi - 1)r_f}$. Noticing that $|\hat{\mathcal{S}}^{\lambda_{\min}}(\mathcal{I})| = \sum_{i=1}^{p} I(i \in \hat{\mathcal{S}}^{\lambda_{\min}}(\mathcal{I}))$ and using Eq. (3.22) to estimate $P(i \in \hat{\mathcal{S}}^{\lambda_{\min}}(\mathcal{I}))$, we reduce the problem to find a $\lambda_{\min}$ such that $\sum_{i=1}^{p} \hat{P}(i \in \hat{\mathcal{S}}^{\lambda_{\min}}(\mathcal{I})) \leq p\sqrt{(2\pi - 1)r_f}$, which can be solved by a search algorithm.

After obtaining $\lambda_{\min}$, the stability selection uses Eq. (3.23) with $\Lambda = \{\lambda_{\min}\}$ to construct the index set of selected non-zero parameters $\hat{\mathcal{S}}^{\text{stable}}$. In other words, a parameter is determined to be non-zero by the stability selection if the relative frequency that this parameter is non-zero in repeated subsampling is no less than the pre-specified threshold π.

3.3.4.2 Stability Selection for BCEL

We now describe how to extend the stability selection of one penalty parameter to BCEL that includes two penalty parameters. Likewise, we measure the stability of variable selection using the probability that a variable is selected (or detected) by repeated subsampling.

We begin by defining a subsampling scheme for BCEL. For each subsampled matrix, we create a new matrix of the same dimension as the original data matrix $\mathbf{X}$, whose entries are either copied from $\mathbf{X}$ or treated as missing. Specifically, we randomly select 50% of the entries of $\mathbf{X}$ and place them in the same positions in the new matrix, and the rest 50% entries of the new matrix are missing. Let $\mathbf{\Omega} \in \mathbb{R}^{n \times p}$ be an indicator matrix, indicating for which position the data is copied to the new matrix, i.e., the $(i, j)^{\text{th}}$ entry of $\mathbf{\Omega}$ is equal to 1 when x_{ij} is copied, and 0 when x_{ij} is not copied. For each subsampled matrix, we estimate $\mathbf{U}$ and $\mathbf{V}$ by solving:

$$\min_{\mathbf{U} \in \mathbb{R}^{n \times r}, \mathbf{V} \in \mathbb{R}^{p \times r}} \|\mathbf{\Omega} \circ (\mathbf{X} - \mathbf{U}\mathbf{V}^T)\|_F^2 \qquad + \lambda_1 P_{1,2}(\mathbf{U}) + \lambda_2 P_{1,2}(\mathbf{V}). \tag{3.25}$$

Note that this is exactly the same as the missing value problem Eq. (3.21) and can be solved by Algorithm 2 as discussed above.

For a fixed pair of tuning parameters (λ_1, λ_2), let $\hat{\mathcal{S}}_1^{(\lambda_1, \lambda_2)}(\mathbf{\Omega})$ and $\hat{\mathcal{S}}_2^{(\lambda_1, \lambda_2)}(\mathbf{\Omega})$ denote respec-

tively the index sets of non-zero entries in estimated $\mathbf{U}$ and $\mathbf{V}$ when using the subsampled matrix corresponding to $\mathbf{\Omega}$. Let Λ_1 and Λ_2 be respectively the sets of possible values for λ_1 and λ_2. Set $\hat{\mathcal{S}}_1^{(\Lambda_1,\Lambda_2)}(\mathbf{\Omega}) = \cup_{(\lambda_1,\lambda_2)\in(\Lambda_1,\Lambda_2)}\hat{\mathcal{S}}_1^{(\lambda_1,\lambda_2)}(\mathbf{\Omega})$ and $\hat{\mathcal{S}}_2^{(\Lambda_1,\Lambda_2)}(\mathbf{\Omega}) = \cup_{(\lambda_1,\lambda_2)\in(\Lambda_1,\Lambda_2)}\hat{\mathcal{S}}_2^{(\lambda_1,\lambda_2)}(\mathbf{\Omega})$. Let $q_1 = E|\hat{\mathcal{S}}_1^{(\Lambda_1,\Lambda_2)}(\mathbf{\Omega})|$ and $q_2 = E|\hat{\mathcal{S}}_2^{(\Lambda_1,\Lambda_2)}(\mathbf{\Omega})|$ be the expected number of non-zero entries of $\mathbf{U}$ and $\mathbf{V}$ in repeated subsampling. As Λ_1 and Λ_2 vary, the values of q_1 and q_2 will vary correspondingly.

The idea of stability selection is to determine appropriate choice of Λ_1 and Λ_2 such that the rate of expected falsely detected non-zero entries in estimated $\mathbf{U}$ and $\mathbf{V}$ is controlled. To simply the problem, we may use some insight from our alternating minimization algorithm in Algorithm 2. When $\mathbf{V}$ is fixed, via Eq. (3.11), we can see that the selection results of non-zero entries in $\mathbf{U}$ is continuous and monotonous to λ_1. Similarly, when $\mathbf{U}$ is fixed, via Eq. (3.12), the selection results of non-zero entries in $\mathbf{V}$ is continuous and monotonous to λ_2. Therefore, following the strategy used in the traditional stability selection, instead of determining the sets Λ_1 and Λ_2, we can focus on determining $\lambda_{1,\min} = \inf\Lambda_1$ for λ_1 and $\lambda_{2,\min} = \inf\Lambda_2$ for λ_2, as the impact of Λ_1 and Λ_2 on the values of q_1 and q_2 are mostly determined by $\lambda_{1,\min}$ and $\lambda_{2,\min}$.

Suppose we want to control the rate of expected falsely detected non-zero entries in estimated $\mathbf{U}$ and $\mathbf{V}$ to be no more than $r_{f,1}$ and $r_{f,2}$, respectively. By using Eq. (3.24), it is sufficient to have $q_1 \leq rn\sqrt{(2\pi-1)r_{f,1}}$ and $q_2 \leq rp\sqrt{(2\pi-1)r_{f,2}}$. Since q_1 and q_2 will decrease as $\lambda_{1,\min}$ and $\lambda_{2,\min}$ increase, it is easy to find a pair of $(\lambda_{1,\min}, \lambda_{2,\min})$ with large values that satisfies $q_1 \leq rn\sqrt{(2\pi-1)r_{f,1}}$ and $q_2 \leq rp\sqrt{(2\pi-1)r_{f,2}}$. However, if $\lambda_{1,\min}$ and $\lambda_{2,\min}$ are too big, then q_1 and q_2 will be too small, leading to the undesirable result that $\mathbf{U}$ and $\mathbf{V}$ estimated by BCEL will have too small number of non-zero entries. Therefore, we would like to balance the number of the truly detected non-zero entries and falsely detected non-zero entries. To this end, suggested by Eq. (3.24), for each fixed pair of $(\lambda_{1,\min}, \lambda_{2,\min})$, we consider the following estimators of the upper bound of the false detection rate,

$$\hat{r}_{f,1} = \frac{1}{2\pi-1}(\frac{\hat{q}_1}{rn})^2, \qquad \hat{r}_{f,2} = \frac{1}{2\pi-1}(\frac{\hat{q}_2}{rp})^2, \tag{3.26}$$

where $\hat{q}_1$ and $\hat{q}_2$ are estimates based on subsampling, obtained similarly as in the traditional stability selection, and rn and rq are the total number of entries of $\mathbf{U}$ and $\mathbf{V}$, respectively. We provide a pair of acceptable intervals, denoted as $(r_{f,1}^{\text{low}}, r_{f,1}^{\text{up}})$, $(r_{f,2}^{\text{low}}, r_{f,2}^{\text{up}})$. Then we search, e.g., using the bisection method, for a pair $(\hat{\lambda}_{1,\min}, \hat{\lambda}_{2,\min})$ such that its corresponding $\hat{r}_{f,1}$ and $\hat{r}_{f,2}$ satisfy $r_{f,1}^{\text{low}} \leq \hat{r}_{f,1} \leq r_{f,1}^{\text{up}}$ and $r_{f,2}^{\text{low}} \leq \hat{r}_{f,2} \leq r_{f,2}^{\text{up}}$. This method will give a desired balance. The details of the stability selection to find $(\hat{\lambda}_{1,\min}, \hat{\lambda}_{2,\min})$ are given in Algorithm 4.

Algorithm 4 Stability Selection for Tuning Parameters

Input: $\mathbf{X}$, r, B_1, $(\lambda_1^{\text{low}}, \lambda_1^{\text{up}})$, $(\lambda_2^{\text{low}}, \lambda_2^{\text{up}})$, $(r_{f,1}^{\text{low}}, r_{f,1}^{\text{up}})$, $(r_{f,2}^{\text{low}}, r_{f,2}^{\text{up}})$, π, and $\alpha > 0$.
Initialization: Set $\lambda_1 = \lambda_1^{\text{low}}$ and $\lambda_2 = \lambda_2^{\text{low}}$.
While (TRUE):

1. **For** i = 1 to B_1:
   (a) Construct a randomly subsampled matrix whose 50% entries are non-zero and copied from $\mathbf{X}$ and indexed by $\mathbf{\Omega}_i$.
   (b) Solve Eq. (3.25) with $\mathbf{\Omega}_i$ and get a pair of estimates of $\mathbf{U}$ and $\mathbf{V}$.
2. Calculate $\hat{q}_1$ and $\hat{q}_2$ based on B_1 pairs of estimates of $\mathbf{U}$ and $\mathbf{V}$.
3. Calculate $\hat{r}_{f,1}$ and $\hat{r}_{f,2}$ by Eq. (3.26).
4. **If** $r_{f,1}^{\text{low}} \leq \hat{r}_{f,1} \leq r_{f,1}^{\text{up}}$ **and** $r_{f,2}^{\text{low}} \leq \hat{r}_{f,2} \leq r_{f,2}^{\text{up}}$:
   BREAK.
5. **For** $s = 1, 2$:
   (a) If $\hat{r}_{f,s} > r_{f,s}^{\text{up}}$, set $\lambda_s^{\text{low}} \leftarrow \lambda_s$ and $\lambda_s \leftarrow \frac{\alpha\lambda_s^{\text{low}} + \lambda_s^{\text{up}}}{\alpha+1}$.
   (b) If $\hat{r}_{f,s} < r_{f,s}^{\text{low}}$, set $\lambda_s^{\text{up}} \leftarrow \lambda_s$ and $\lambda_s \leftarrow \frac{\alpha\lambda_s^{\text{low}} + \lambda_s^{\text{up}}}{\alpha+1}$.

Output: $(\hat{\lambda}_{1,\min}, \hat{\lambda}_{2,\min}) = (\lambda_1, \lambda_2)$.

After obtaining $\lambda_{1,\min}$ and $\lambda_{2,\min}$, we can use the stability selection to decide on the bi-clustering membership of the rows and columns. This corresponds to decide on which entries of $\mathbf{U}$ and $\mathbf{V}$

are non-zero. We encounter a difficulty that is specific to the BCEL. Specifically, since its objection function is invariant to permutation of the columns of $\mathbf{U}$ and $\mathbf{V}$, the estimated $\mathbf{U}$ and $\mathbf{V}$ from two randomly subsampled matrices may not be aligned and thus are not comparable. This causes problem when estimating the probability of an entry of $\mathbf{U}$ or $\mathbf{V}$ is non-zero. There is no simple algorithm to align solutions obtained for different subsampled matrices. As a remedy, we propose a method to construct the estimated $\mathbf{U}$ and $\mathbf{V}$ from different subsampled data matrices while making sure that the columns obtained using different subsamples are in good alignment and thus comparable.

Specifically, we first build a "reference" estimator of $\mathbf{U}$ and $\mathbf{V}$ using the full data, i.e., the original matrix $\mathbf{X}$, and the penalty parameters $\hat{\lambda}_{1,\min}$ and $\hat{\lambda}_{2,\min}$. Denote the estimates to be $\hat{\mathbf{U}}_{\text{full}}$ and $\hat{\mathbf{V}}_{\text{full}}$. Then, for each subsampled data matrix associated with the index matrix $\mathbf{\Omega}$, we find a solution pair of $\mathbf{U}$ and $\mathbf{V}$ by

$$\hat{\mathbf{U}}_i = \arg\min_{\mathbf{U}} \|\text{vec}(\mathbf{\Omega}^{\boldsymbol{T}}) \circ \left[\text{vec}(\mathbf{X}^T) - \mathbf{D}_{\hat{\mathbf{V}}_{\text{full}}} \cdot \text{vec}(\mathbf{U}^T)\right]\|_2^2 + \hat{\lambda}_{1,\min} P_{1,2}(\mathbf{U}), \tag{3.27}$$

and

$$\hat{\mathbf{V}}_i = \arg\min_{\mathbf{V}} \|\text{vec}(\mathbf{\Omega}) \circ \left[\text{vec}(\mathbf{X}) - \mathbf{D}_{\hat{\mathbf{U}}_{\text{full}}} \cdot \text{vec}(\mathbf{V}^T)\right]\|_2^2 + \hat{\lambda}_{2,\min} P_{1,2}(\mathbf{V}). \tag{3.28}$$

Since $\hat{\mathbf{U}}_{\text{full}}$ and $\hat{\mathbf{V}}_{\text{full}}$ are used to construct the design matrix $\mathbf{D}_{\hat{\mathbf{U}}_{\text{full}}}$ and $\mathbf{D}_{\hat{\mathbf{V}}_{\text{full}}}$ in the two penalized regression problems Eq. (3.27) and Eq. (3.28), the ordering of the columns of $\hat{\mathbf{U}}_i$ and $\hat{\mathbf{V}}_i$ from different subsamples is fixed, and therefore these $\hat{\mathbf{U}}_i$ and $\hat{\mathbf{V}}_i$ are comparable. Then for each entry of $\mathbf{U}$ and $\mathbf{V}$, we can calculate the relative frequency (or estimated probability) it is non-zero based on repeated subsampling. If this relative frequency is more than a pre-specified threshold π, the entry is claimed to be non-zero by the stability selection, and it is set to zero otherwise. Algorithm 5 details the steps of the stability selection to determine the non-zero entries of $\mathbf{U}$ and $\mathbf{V}$.

By setting the entries of $\hat{\mathbf{U}}_{\text{full}}$ and $\hat{\mathbf{V}}_{\text{full}}$ that are not selected by the stability selection to 0, we can obtain a new pair of estimates of $\mathbf{U}$ and $\mathbf{V}$ that follows the bicluster membership determined by the stability selection. We denote the final estimates as $\hat{\mathbf{U}}_{\text{stable}}$ and $\hat{\mathbf{V}}_{\text{stable}}$.

Algorithm 5 Stability Selection for Bicluster Membership

Input: $\mathbf{X}$, r, B_2, $\hat{\lambda}_{1,min}$, $\hat{\lambda}_{2,min}$, and π.
Steps:

1. Estimate $\hat{\mathbf{U}}_{\text{full}}$ and $\hat{\mathbf{V}}_{\text{full}}$ by solving Eq. (3.10) with the penalty parameters $(\hat{\lambda}_{1,\min},\hat{\lambda}_{2,\min})$.
2. **For** i = 1 to B_2:
   (a) Construct a randomly subsampled matrix whose 50% entries are non-zero and copied from $\mathbf{X}$ and indexed by $\mathbf{\Omega}_i$.
   (b) Calculate $\hat{\mathbf{U}}_i$ with $\hat{\mathbf{V}}_{\text{full}}$ and $\mathbf{\Omega}_i$ by Eq. (3.27).
   (c) Calculate $\hat{\mathbf{V}}_i$ with $\hat{\mathbf{U}}_{\text{full}}$ and $\mathbf{\Omega}_i$ by Eq. (3.28).
3. For each entry of $\mathbf{U}$ and $\mathbf{V}$, calculate the relative frequency that it is not zero in $\hat{\mathbf{U}}_i$s and $\hat{\mathbf{V}}_i$s.
4. The entries of $\mathbf{U}$ and $\mathbf{V}$ with the above relative frequency larger than π are signaled as the non-zero entries determined by the stability selection.

3.4 Numerical Study

In this section, using simulated datasets, we evaluate the performance of the proposed BCEL method and compare it with several existing biclustering methods. We consider several representative matrix decomposition based methods reviewed in Section 3.2. In particular, we include in our comparison sparseBC from block decomposition methods, S4VD (the stable version of SSVD) from SVD based methods, NMFSC from NMF based methods, and Plaid. We also include two widely used traditional methods, Bimax [50] and ISA [45]. We used public domain R packages that implemented these methods, i.e., sparseBC from the `sparseBC` package, S4VD from the `s4vd` package, NMFSC from the `fabia` package, Plaid and Bimax from the `biclust` package, and ISA from the `isa2` package.

3.4.1 Evaluating Methods

Because the correct or ground-truth biclustering is known for our simulated data, we can use the true bicluster labels to evaluate a given biclustering method. We shall define some measures

to quantify the extent to which the detected biclusters match the true biclusters. To present the technique details, we first introduce some notations. Recall that r is the number of true biclusters. The true biclusters are denoted as $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_r\}$, where $\mathcal{T}_i = \{(s,t) : s \in \mathcal{R}_i \text{ and } t \in \mathcal{C}_i\}$ collects the indexes of elements included in the i^{th} bicluster, and $\mathcal{R}_i$ and $\mathcal{C}_i$ collect respectively the row and column indices. Let $\hat{r}$ denote the number of detected biclusters by a biclustering method, which can be different from r. Let $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_{\hat{r}}\}$ denote the detected biclusters with $\mathcal{P}_j = \{(s,t) : s \in \hat{\mathcal{R}}_j \text{ and } t \in \hat{\mathcal{C}}_j\}$, where $\hat{\mathcal{R}}_j$ and $\hat{\mathcal{C}}_j$ are respectively the index sets of rows and columns in the j^{th} detected bicluster.

Matching based measures. Assume $\mathcal{M} : \{1, \ldots, r\} \to \{1, \ldots, \hat{r}\}$ is an injection mapping from the index set of true biclusters to the index set of detected biclusters. If $\hat{r} < r$, we add $r - \hat{r}$ empty biclusters to the collection of detected biclusters. The mapping $\mathcal{M}$ matches the true bicluster $\mathcal{T}_i$ with the detected bicluster $\mathcal{P}_{\mathcal{M}(i)}$ for $i = 1, \ldots, r$. The following Jaccard coefficient measures the overall degree of overlapping between the true biclusters and the corresponding detected biclusters matched by $\mathcal{M}$:

$$\text{Jaccard}(\mathcal{M}) = \frac{\sum_{i=1}^{r} |\mathcal{T}_i \cap \mathcal{P}_{\mathcal{M}(i)}|}{\sum_{i=1}^{r} |\mathcal{T}_i \cup \mathcal{P}_{\mathcal{M}(i)}|}.$$

We call $\mathcal{M}_m$ the maximum matching mapping if it maximizes the Jaccard coefficient among all possible injection mappings:

$$\mathcal{M}_m = \arg\max_{\mathcal{M}} \text{Jaccard}(\mathcal{M}).$$

The mapping $\mathcal{M}_m$ gives the highest possible degree of overlapping and can be used to evaluate biclustering methods as follows. For the i^{th}bicluster, define the bicluster specific precision and recall to be

$$\text{precision}_i = \frac{|\mathcal{T}_i \cap \mathcal{P}_{\mathcal{M}_m(i)}|}{|\mathcal{P}_{\mathcal{M}_m(i)}|}, \ \text{recall}_i = \frac{|\mathcal{T}_i \cap \mathcal{P}_{\mathcal{M}_m(i)}|}{|\mathcal{T}_i|},$$

and the F-measure to be the harmonic mean of the precision and recall values,

$$\text{F}_i = \frac{2}{1/\text{precision}_i + 1/\text{recall}_i}.$$

Taking the average over all biclusters of the precision, recall, and F-measure gives three overall measures of bicluster performance.

The way to generate the maximum matching here is related to the “consensus score" method proposed by [73]. The consensus score method maximizes the sum of bicluster-wise values of the Jaccard coefficient to find the mapping between two sets of biclusters, while our method focuses on the overall Jaccard coefficient.

Contingency table based measures. A drawback of the maximum matching is that it only considers the one-to-one correspondence between the true biclusters and the predicted biclusters. Sometimes, a biclustering method can discover more biclusters than the true number of biclusters, and/or a true bicluster can correspond to multiple detected biclusters. The maximum matching may under-evaluate the performance of the biclustering method for such cases. Thus, different performance measures are desirable to complement the maximum matching based measures.

We treat the detected biclusters as the result of a binary classification problem for entries of the data matrix, and the data generating procedure tells about the ground-truth biclustering membership. Then contingency table based measures widely used to evaluate classification/clustering results can be applied [74]. Specifically, for each entry of the data matrix, it is classified as a positive case if it is included in one of the detected biclusters and as a negative case if not. Then, each entry can be grouped into one of the four categories: i. true positive (TP), if the entry is in one of $\mathcal{T}_i$ and one of $\mathcal{P}_j$; ii. false negative (FN), if the entry is in one of $\mathcal{T}_i$ but none of $\mathcal{P}_j$; iii. false positive (FP), if the entry is not in any $\mathcal{T}_i$ but in one of $\mathcal{P}_j$; iv. true negative (TN), if the entry is not in any of $\mathcal{T}_i$ or $\mathcal{P}_j$. We use TP, FN, FP, and TN to denote the count of entries in each category. To take care of overlapping between biclusters, i.e., a matrix entry may belong to multiple biclusters, we make the following adjustment to the counts: If an entry presents respectively a_1 and a_2 times in the true and detected biclusters, then we add a_2 to TP and $a_1 - a_2$ to FN when $a_1 \geq a_2$, and add a_1 to TP and $a_2 - a_1$ to FP when $a_1 < a_2$.

We build three measures as below. The Jaccard coefficient measures the percentage of TP

among the total number of entries included in either a true or detected bicluster and is defined as

$$\text{Jaccard} = \frac{\text{TP}}{\text{TP} + \text{FN} + \text{FP}}.$$

It reflects the similarity of two groups of biclusters. The Rand Statistics (RS) is defined as

$$\text{RS} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}.$$

It calculates the fraction of TP and TN over all entries of the data matrix, showing the percentage of entries whose classification results are agreed by the true and detected biclusters. The precision $\frac{\text{TP}}{\text{TP+FP}}$ and the recall $\frac{\text{TP}}{\text{TP+FN}}$ measure respectively the percentage of TP in entries included in the detected or the true biclusters. The Fowlkes-Mallows (FM) measure is defined as the geometric meas of the precision and recall values:

$$\text{FM} = \sqrt{\frac{\text{TP}}{\text{TP} + \text{FP}} \cdot \frac{\text{TP}}{\text{TP} + \text{FN}}}.$$

All of the above measures have a range of $[0, 1]$. The larger the value of the measure is, the better the performance of a method.

3.4.2 Simulation Study

In our simulation study, we generate the observed data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ with r true biclusters that can overlap with each other arbitrarily as illustrated in Figure 3.1d. The data generating model is

$$\mathbf{X} = \mathbf{U}\mathbf{V}^T + \mathbf{E},$$

where $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r) \in \mathbb{R}^{n \times r}$ and $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r) \in \mathbb{R}^{p \times r}$, and the entries of $\mathbf{E} \in \mathbb{R}^{n \times p}$ are independently generated from the standard normal distribution. Non-zero entries of $\mathbf{u}_k$ and $\mathbf{v}_k$ indicate membership of the k^{th} bicluster. We fixed n to be 100, $p = 200$ or $p = 2000$, and considered $r = 3$ or $r = 6$. This gives us four simulation settings.

Let $\mathcal{R}_k$ and $\mathcal{C}_k$ denote respectively the index set of rows and columns of the k^{th} bicluster. Let m_{1k} denote the number of rows and m_{2k} the number of columns that are involved in the k^{th} bicluster. We generated m_{1k} from $U(10, 60)$, and generated m_{2k} from $U(15, 70)$ for $p = 200$ or from $U(150, 700)$ for $p = 2000$. We then generated $\mathcal{R}_k$ and $\mathcal{C}_k$ by randomly selecting m_{1k} rows and m_{2k} columns from the totals of n rows and p columns. The entries of $\mathbf{u}_k$ and $\mathbf{v}_k$ are generated according to:

$$\begin{cases} u_{ik} \sim U(1,2) & \text{if } i \in \mathcal{R}_k \\ u_{ik} = 0 & \text{otherwise} \end{cases}, \qquad \begin{cases} v_{jk} \sim U(1,2) & \text{if } j \in \mathcal{C}_k \\ v_{jk} = 0 & \text{otherwise} \end{cases}.$$

For each of the simulation setting, we generated 100 data matrices, applied different biclustering methods on them, and calculated the evaluation measures to compare these methods. When applying BCEL, we set $(r_{f,1}^{\text{low}}, r_{f,1}^{\text{up}}) = (0.1, 0.3)$, $(r_{f,2}^{\text{low}}, r_{f,2}^{\text{up}}) = (0.1, 0.3)$, $\pi = 0.65$, $B_1 = 20$, and $B_2 = 200$.

Table 3.2 and Table 3.3 summarize the comparison results of simulation with $r = 3$ and $r = 6$ respectively using the evaluation methods described in the previous subsection. BCEL is the best performer in all four settings for all seven different measures except for "precision" when $p = 2000$, for which BCEL ranks the second or the third and is very close to the first. BCEL achieves the impressive higher than 0.9 value in almost all settings and evaluation measures. Comparing with another SVD based method, BCEL substantially improves over the sequential approach of S4VD. Plaid achieves the second place overall when $r = 3$ but performs badly when $r = 6$, suggesting that it is not robust to depart from its assumption of additive structure. The sparseBC has mediocre performance when $r = 3$, but the performance deteriorates significantly when $r = 6$. NMFSC and two traditional methods, Bimax and ISA perform poorly in most settings and performance measures. These results clearly show the superiority of BCEL over other methods.

To graphically compare the performance of different methods, Figure 3.4 presents the discovered biclusters by BCEL and a few other methods in one arbitrarily chosen simulation matrix of $p = 200$ and $r = 3$. We only include BCEL, sparseBC, S4VD, and Plaid in the figure since

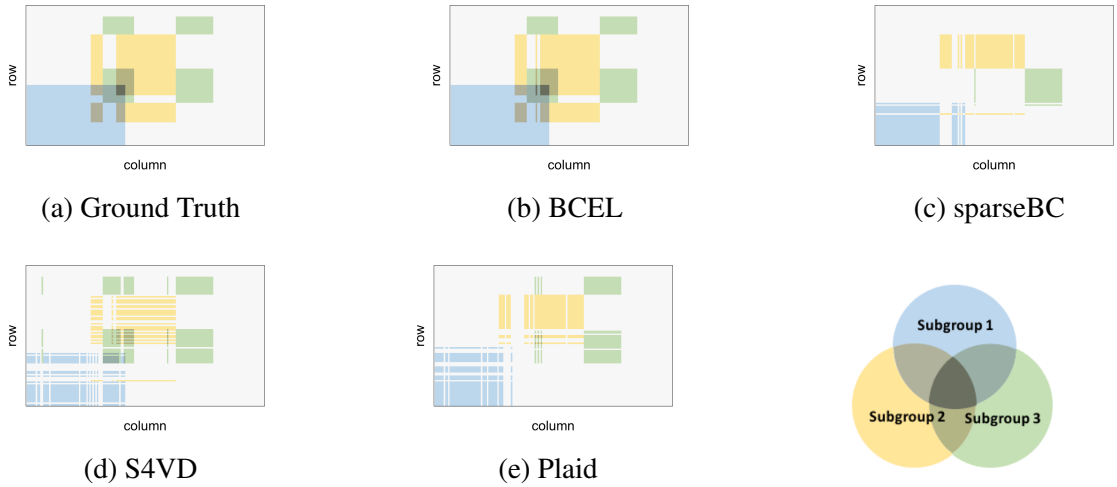
Table 3.2: Performance comparison of different biclustering methods on simulated data matrices with $r = 3$. Mean and standard derivation of each performance measure is calculated based on 100 simulation runs.

p	Method	Maximum Based				Contingency Table Based		
		Jaccard	Precision	Recall	F	Jaccard	Rand	FM
200	BCEL	**0.976**	**0.970**	**0.994**	**0.967**	**0.976**	**0.995**	**0.988**
		(0.027)	(0.035)	(0.017)	(0.032)	(0.027)	(0.006)	(0.014)
	sparseBC	0.423	0.778	0.397	0.463	0.521	0.888	0.708
		(0.175)	(0.149)	(0.169)	(0.174)	(0.173)	(0.063)	(0.132)
	S4VD	0.521	0.798	0.634	0.608	0.682	0.910	0.817
		(0.120)	(0.156)	(0.167)	(0.140)	(0.123)	(0.048)	(0.083)
	NMFSC	0.232	0.452	0.303	0.268	0.280	0.798	0.447
		(0.035)	(0.081)	(0.090)	(0.061)	(0.034)	(0.039)	(0.037)
	Plaid	0.563	0.868	0.542	0.598	0.703	0.932	0.828
		(0.211)	(0.194)	(0.208)	(0.209)	(0.183)	(0.051)	(0.122)
	Bimax	0.147	0.746	0.154	0.212	0.315	0.846	0.554
		(0.037)	(0.112)	(0.037)	(0.056)	(0.036)	(0.044)	(0.034)
	ISA	0.169	0.932	0.232	0.314	0.192	0.790	0.357
		(0.080)	(0.090)	(0.100)	(0.104)	(0.068)	(0.065)	(0.089)
2000	BCEL	**0.957**	0.922	**0.997**	**0.924**	**0.957**	**0.991**	**0.978**
		(0.053)	(0.097)	(0.003)	(0.093)	(0.053)	(0.011)	(0.027)
	sparseBC	0.375	0.673	0.294	0.372	0.478	0.899	0.686
		(0.153)	(0.004)	(0.125)	(0.147)	(0.167)	(0.035)	(0.122)
	S4VD	0.599	0.655	0.849	0.645	0.504	0.831	0.709
		(0.049)	(0.031)	(0.028)	(0.077)	(0.051)	(0.034)	(0.035)
	NMFSC	0.200	0.429	0.263	0.208	0.272	0.823	0.435
		(0.015)	(0.004)	(0.013)	(0.029)	(0.015)	(0.004)	(0.016)
	Plaid	0.738	**0.999**	0.750	0.814	0.855	0.972	0.924
		(0.216)	(0.001)	(0.167)	(0.134)	(0.051)	(0.011)	(0.028)
	Bimax	0.149	0.786	0.164	0.224	0.319	0.876	0.560
		(0.057)	(0.113)	(0.070)	(0.093)	(0.029)	(0.049)	(0.024)
	ISA	0.196	**0.995**	0.320	0.389	0.227	0.830	0.413
		(0.010)	(0.007)	(0.110)	(0.059)	(0.009)	(0.016)	(0.037)

Table 3.3: Performance comparison of different biclustering methods on simulated data matrices with $r = 6$. Mean and standard derivation of each performance measure is calculated based on 100 simulation runs.

p	Method	Maximum Based				Contingency Table Based		
		Jaccard	Precision	Recall	F	Jaccard	Rand	FM
200	BCEL	**0.958**	**0.947**	**0.981**	**0.943**	**0.960**	**0.984**	**0.979**
		(0.050)	(0.057)	(0.051)	(0.057)	(0.045)	(0.020)	(0.024)
	sparseBC	0.103	0.539	0.093	0.118	0.180	0.664	0.415
		(0.034)	(0.066)	(0.031)	(0.038)	(0.067)	(0.075)	(0.075)
	S4VD	0.370	0.651	0.472	0.422	0.612	0.816	0.763
		(0.100)	(0.142)	(0.158)	(0.126)	(0.108)	(0.053)	(0.080)
	NMFSC	0.168	0.332	0.234	0.185	0.212	0.604	0.355
		(0.014)	(0.040)	(0.039)	(0.029)	(0.019)	(0.049)	(0.024)
	Plaid	0.143	0.180	0.089	0.097	0.199	0.517	0.373
		(0.111)	(0.122)	(0.063)	(0.068)	(0.135)	(0.319)	(0.234)
	Bimax	0.138	0.593	0.133	0.170	0.291	0.709	0.533
		(0.029)	(0.111)	(0.025)	(0.036)	(0.042)	(0.057)	(0.042)
	ISA	0.128	0.866	0.180	0.241	0.151	0.636	0.335
		(0.049)	(0.092)	(0.066)	(0.072)	(0.049)	(0.077)	(0.053)
2000	BCEL	**0.913**	0.871	**0.995**	**0.881**	**0.913**	**0.965**	**0.955**
		(0.067)	(0.079)	(0.043)	(0.069)	(0.067)	(0.022)	(0.037)
		(0.067)	(0.079)	(0.043)	(0.069)	(0.067)	(0.022)	(0.037)
	sparseBC	0.098	0.509	0.086	0.111	0.177	0.66	0.413
		(0.035)	(0.057)	(0.036)	(0.048)	(0.06)	(0.093)	(0.067)
	S4VD	0.351	0.487	0.56	0.396	0.505	0.719	0.694
		(0.085)	(0.12)	(0.179)	(0.113)	(0.074)	(0.037)	(0.052)
	NMFSC	0.145	0.351	0.188	0.158	0.192	0.612	0.337
		(0.013)	(0.051)	(0.029)	(0.03)	(0.018)	(0.067)	(0.019)
	Plaid	0.148	0.197	0.091	0.105	0.207	0.508	0.371
		(0.12)	(0.152)	(0.073)	(0.086)	(0.164)	(0.343)	(0.259)
	Bimax	0.131	0.575	0.142	0.17	0.314	0.737	0.557
		(0.022)	(0.142)	(0.032)	(0.05)	(0.029)	(0.083)	(0.026)
	ISA	0.137	**0.891**	0.199	0.26	0.161	0.636	0.346
		(0.075)	(0.097)	(0.094)	(0.099)	(0.073)	(0.106)	(0.095)

the other three methods in Table 3.2 have relatively poor performance. In Figure 3.4, the blue, yellow, and green colored areas correspond respectively to the regions of three biclusters detected by biclustering, and the maximum matching mapping is used to label the detected clusters. We observe that BCEL almost recovers the original layout of three biclusters and is capable of dealing with the overlapping between biclusters. S4VD also discovers the main shape of three biclusters, but it fails to recover the overlapping part of the blue bicluster and the yellow bicluster. Plaid has difficulty in detecting overlapping biclusters although it should have the capacity by design. At last, sparseBC only discovers three non-overlapping biclusters, which is not surprising, since it assumes a checkerboard structure of the data matrix.



(a) Ground Truth (b) BCEL (c) sparseBC

(d) S4VD (e) Plaid

Figure 3.4: Comparison of the discovered biclusters from different biclustering methods.

3.5 Single-Cell RNA Sequencing Data Analysis

The latest Next Generation Sequencing (NGS) technologies allow researchers to obtain gene expression information at the single-cell level and provide new opportunities to resolve comprehensive studies of individual cells. Single-cell RNA sequencing (scRNA-seq), for example, can reveal the cell-to-cell variations and infer the underlying gene regulatory networks. Since cellar

heterogeneity is an essential characteristic of scRNA-seq, uncovering the true diversity of cells is crucial to better understand the relationship and responsibility of genes and different types of cells. As discussed in Section 3.1, biclustering may provide a valuable tool to discover the latent groups of homogeneous cells and the associated reacting genes without using any prior information about the relationship between cells and genes.

To illustrate the application of BCEL on the scRNA-seq dataset, we consider a subset of data from a study on the mouse brain cells by [75]. That study included the expression levels of around 15000 genes on 37,069 mouse brain cells. By using the well-known cell-type-specific marker genes and dimension reduction methods, six major cell subtypes were identified among all cells. With known labels of cell subtypes, for each cell subtype, the genes were ranked via statistics tests so that the genes with higher rank express more differently in this cell subtype than other cell subtypes. The top ranked genes for each cell subtype were treated as the most discriminating genes (called marker genes) that help identify this cell subtype from all cells.

We now build a sub-dataset of the above scRNA-seq dataset to compare different biclustering methods. We randomly sample 200 cells for each cell subtype and obtain 1200 cells in total. We focus on genes among at least one of the top 500 rank genes of six subtypes. Since the top rank genes for different subtypes can overlap, the number of these genes is 2846. Additionally, we randomly select another 500 genes from the whole gene list. These genes are treated to have no business with the cell type. The sub-dataset is then constructed by selecting the gene expression data corresponding to the 3346 genes and 1200 cells in the original data matrix and is stored in a 3346×1200 data matrix that each row represents a gene and each column represents a cell. To adjust the cell size factor, we divide each column of the data matrix by the mean expression level of the entries in this column. We then take the logarithm transformation to the whole dataset since the distribution of each gene's expression level usually has a long right-tail. The transformed dataset are shown in Figure 3.5a.

We applied four biclustering methods, BCEL, sparseBC, S4VD, and Plaid on this dataset. Three other biclustering methods, NMFSC, Bimax, and ISA, that did not perform well in the

Table 3.4: Performance comparison of different biclustering methods on the scRNA-seq data.

Method	Matching Based				Contingency Table Based		
	Jaccard	Precision	Recall	F	Jaccard	Rand	FM
BCEL	**0.390**	0.519	0.617	**0.486**	**0.342**	0.826	**0.518**
sparseBC	0.351	**0.785**	0.371	0.435	0.338	**0.877**	0.516
S4VD	0.154	0.163	**0.763**	0.215	0.112	0.354	0.312
Plaid	0.199	0.765	0.216	0.231	0.197	0.792	0.329

Method	F for Six Cell Subtypes with the Maximum Matching Mapping					
	IMMUNE	OLG	NEURON	VASC	ASC	EPC
BCEL	**0.396**	0.242	**0.675**	0.644	0.583	**0.380**
sparseBC	0.372	0.000	0.583	**0.783**	0.493	0.378
S4VD	0.219	0.203	0.216	0.360	0.137	0.154
Plaid	0.072	**0.376**	0.260	0.011	**0.645**	0.020

simulation study were not included. The subtypes of the cells and corresponding marker genes as determined by [75] are used as the ground truth when evaluating the biclustering methods.

Table 3.4 presents the performance comparison of different methods on the scRNA-seq dataset. For matching based measures, BCEL has the highest Jaccard coefficient. Although BCEL is only the second best in terms of precision and recall, it has the highest F-measure, indicating that BCEL gives the best compromise on precision and recall among all methods. In contrast, sparseBC has the highest precision but low recall, S4VD has the highest recall but very low precision. For contingency table based measures, BCEL is the best performer in terms of Jaccard coefficient and FM, and is the second best in terms of RS but very close to the first ranked sparseBC. Table 3.4 also shows the F-measure for different cell subtypes with the maximum matching mapping. We observe that BCEL has the best F-measure values in three cell subtypes and is the second-best in the other three subtypes. As a comparison, sparseBC cannot detect the differential genes for the OLG subtype, S4VD has relatively low values of F-measure in all six subtypes, Plaid almost fails to detect the subtypes of IMMUNE, VASC, and EPC.

Figure 3.5 graphically show the biclusters detected by different methods. For each bicluster, the genes included in this bicluster are highly expressed in the cells included in this bicluster,

indicating that these genes may affect the functions of this group of cells. We first observe that all of the four methods except S4VD can recover the group structures of cells and avoid to include the redundant genes, while S4VD produces many overlapping biclusters that are not present in the ground truth. Comparing with sparseBC and Plaid, BCEL discovers more genes than the true differential genes for each subtype of cells, and it includes more cells in the detected biclusters. This is in accordance with the result in Table 3.4 that BCEL has lower precision but higher recall than sparseBC and Plaid. On the other hand, although the extra genes detected by BCEL are not marker genes as reported in the original paper [75], they show high expression levels in the original data and could be function-related genes. Consistent with Table 3.4, we see that sparseBC fails to discover the differential genes of the OLG subtype, and Plaid has difficulty in discovering genes for three cell subtypes, IMMUNE, VASC, and EPC. Figure 3.5a indicates that the differential genes for the OLG subtype have relatively low expression levels compared with differential genes for other subtypes. This may be the reason that all methods have a low F-measure value for OLG in Table 3.4.

3.6 Proof

3.6.1 Proof of Lemma 3

For $b_i \neq 0$, multiple $\text{sign}(b_i)/2$ to both sides of Eq. (3.18) to obtain

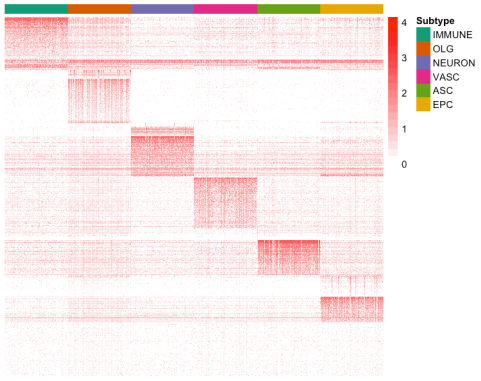
$$-\left(|z_i| - |b_i|\right) + \lambda \sum_{i=1}^{n} |b_i| = 0. \tag{3.29}$$

Taking the summation of the above equations for $i \in \mathcal{S}$, we get:

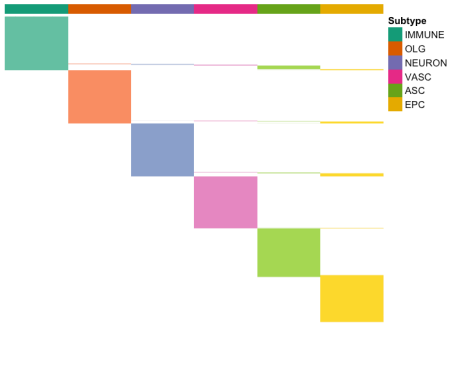
$$-\sum_{i\in\mathcal{S}} |z_i| + \sum_{i\in\mathcal{S}} |b_i| + \lambda|\mathcal{S}| \sum_{i\in\mathcal{S}} |b_i| = 0.$$

Simple algebra yields

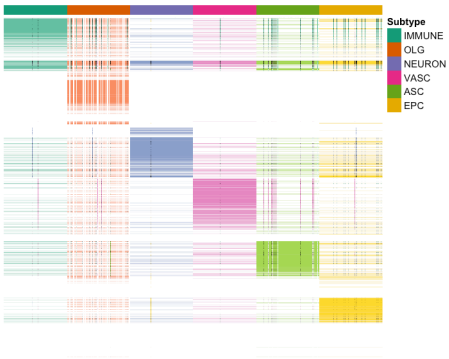
$$\sum_{i\in\mathcal{S}} |b_i| = \frac{1}{1+\lambda|\mathcal{S}|} \sum_{i\in\mathcal{S}} |z_i|.$$



(a) Original Data

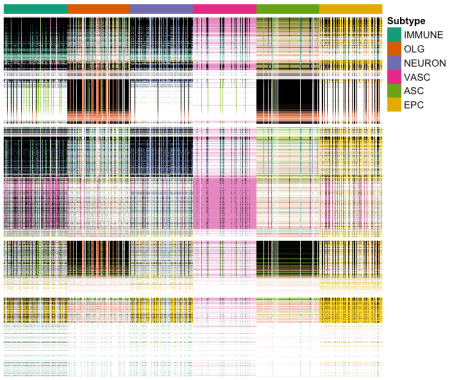


(b) Marker Genes

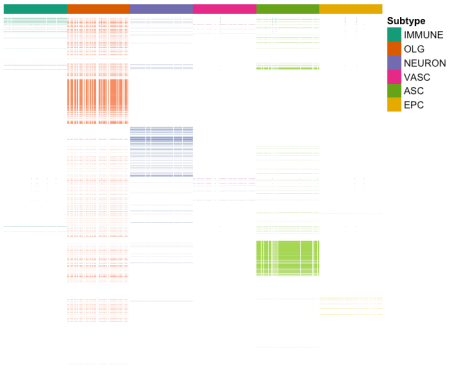


(c) BCEL

(d) sparseBC



(e) S4VD



(f) Plaid

Figure 3.5: Biclustering results of different methods for the scRNA-seq data: (a) Original gene expression matrix; (b) Ground truth of cell groups and the marker genes for each group; (c)-(f) Grouped cells and corresponding genes for different biclustering methods. Columns of each matrix represent cells arranged according to subtypes indicated by the colored horizontal bar on the top. Rows of each matrix represent genes. Each color in the matrices represents a bicluster, and the overlapping parts are marked in black color. The labels of the biclusters are obtained using the maximum matching mapping.

Plugging this into Eq. (3.29) gives

$$|b_i| = |z_i| - \frac{\lambda}{1+\lambda|\mathcal{S}|}\sum_{i\in\mathcal{S}}|z_i|.$$

Together with $\text{sign}(b_i) = \text{sign}(z_i)$ (Lemma 4(i)), we have $\hat{b}_i = \text{sign}(z_i)(|z_i| - \frac{\lambda}{1+\lambda|\mathcal{S}|}\sum_{i\in\mathcal{S}}|z_i|)$.

3.6.2 Proof of Lemma 4

Proof by contradiction: Denote $\tilde{\mathbf{b}} = (\tilde{b}_1, \tilde{b}_2, \ldots, \tilde{b}_d)^T$. Set $l(\mathbf{b}) = ||\mathbf{z}-\mathbf{b}||_2^2 + \lambda||\mathbf{b}||_1^2$.

(i) For any $\tilde{\mathbf{b}}$, if $\tilde{b}_i \cdot z_i < 0$, it is easy to check that $l((\tilde{b}_1, \ldots, \tilde{b}_{i-1}, 0, \tilde{b}_{i+1}, \tilde{b}_d)^T) < l(\tilde{\mathbf{b}})$.

(ii) For any $\tilde{\mathbf{z}}$ with an element $|\tilde{b}_i| > |z_i|$, we have $l((\tilde{b}_1, \ldots, \tilde{b}_{i-1}, z_i, \tilde{b}_{i+1}, \tilde{b}_d)^T) < l(\tilde{\mathbf{b}})$.

(iii) Suppose $i < j$, for any $\tilde{\mathbf{z}} = (\tilde{z}_l)$ satisfying $(|\tilde{b}_i| - |\tilde{b}_j|)(|z_i| - |z_j|) < 0$ and $\tilde{z}_l = z_l$ for $l \neq i, j$, we have $l((\tilde{b}_1, \ldots, \tilde{b}_{i-1}, \text{sign}(z_i)|\tilde{b}_j|, \tilde{b}_{i+1}, \ldots, \tilde{b}_{j-1}, \text{sign}(z_j)|\tilde{b}_i|, \tilde{b}_{j+1}, \ldots, \tilde{b}_d)^T)$ $< l(\tilde{\mathbf{b}})$.

To conclude, $\tilde{\mathbf{b}}$ satisfying any of (i)-(iii) will not be the optimal solution of Eq. (3.17).

3.6.3 Proof of Lemma 5

We claim that, if $\mathcal{S}_d$ satisfies the property that $|z_{a_d}| > \frac{\lambda}{1+\lambda d}\sum_{i\in\mathcal{S}_d}|z_i|$, then $\mathcal{S}_{d-1}$ satisfies the same property, i.e., $|z_{a_{d-1}}| > \frac{\lambda}{1+\lambda(d-1)}\sum_{i\in\mathcal{S}_{d-1}}|z_i|$. This can be easily seen by algebra.

Let $\hat{\mathbf{b}}^d$ denote the estimator calculated by using Eq. (3.19) with $\mathcal{S} = \mathcal{S}_d$. We show that

$$l(\hat{\mathbf{b}}^d) \leq l(\hat{\mathbf{b}}^{d-1}) \tag{3.30}$$

and the equality holds if $\hat{b}^d_{a_d} = 0$. This result implies that $\mathcal{S}$ should be $\mathcal{S}_d$ with the largest d that satisfies $|z_{a_d}| > \frac{\lambda}{1+\lambda d}\sum_{i\in\mathcal{S}_d}|z_i|$.

It remains to prove the above result. Using Eq. (3.19), we obtain

$$\begin{aligned} l(\hat{\mathbf{b}}^d) =& \sum_{i\in S_d}(\hat{b}_i^d - z_i)^2 + \sum_{i\notin S_d}(z_i)^2 + \lambda(\sum_{i\in S_d}|\hat{b}_i^d|)^2 \\ =& d\cdot(\frac{\lambda}{1+\lambda d}\sum_{i\in S_d}|z_i|)^2 + \sum_{i\notin S_d}(z_i)^2 + \lambda\cdot(\frac{1}{1+\lambda d}\sum_{i\in S_d}|z_i|)^2 \\ =& \frac{\lambda}{1+\lambda d}(\sum_{i\in S_d}|z_i|)^2 + \sum_{i\notin S_d}(z_i)^2. \end{aligned}$$

Similarly,

$$l(\hat{\mathbf{b}}^{d-1}) = \frac{\lambda}{1+\lambda(d-1)}(\sum_{i\in S_{d-1}}|z_i|)^2 + \sum_{i\notin S_{d-1}}(z_i)^2.$$

Take the difference to obtain

$$\begin{aligned} & l(\hat{\mathbf{b}}^d) - l(\hat{\mathbf{b}}^{d-1}) \\ =& \frac{\lambda}{1+\lambda d}(\sum_{i\in S_d}|z_i|)^2 - \frac{\lambda}{1+\lambda(d-1)}(\sum_{i\in S_{d-1}}|z_i|)^2 - (z_{a_d})^2 \\ =& (\frac{\lambda}{1+\lambda d} - \frac{\lambda}{1+\lambda(d-1)})(\sum_{i\in S_{d-1}}|z_i|)^2 \\ & + \frac{2\lambda}{1+\lambda d}|z_{a_d}|(\sum_{i\in S_{d-1}}|z_i|) + (\frac{\lambda}{1+\lambda d} - 1)|z_{a_d}|^2 \end{aligned}$$

$$\begin{aligned} =& -\frac{\lambda^2}{(1+\lambda d)(1+\lambda(d-1))}(\sum_{i\in S_{d-1}}|z_i|)^2 \\ & + \frac{2\lambda}{1+\lambda d}|z_{a_d}|(\sum_{i\in S_{d-1}}|z_i|) - (\frac{1+\lambda(d-1)}{1+\lambda d})|z_{a_d}|^2 \\ =& -\frac{1+\lambda(d-1)}{1+\lambda d}(|z_{a_d}| - \frac{\lambda}{1+\lambda(d-1)}\sum_{i\in S_{d-1}}|z_i|)^2 \leq 0. \end{aligned}$$

If the equality holds, we must have $|z_{a_d}| - \frac{\lambda}{1+\lambda(d-1)}\sum_{i\in S_{d-1}}|z_i| = 0$, which by some algebra implies $|z_{a_d}| - \frac{\lambda}{1+\lambda d}\sum_{i\in S_d}|z_i| = 0$, which in turn implies $\hat{b}_{a_d}^d = 0$.

REFERENCES

[1] R. R. Hocking, "A biometrics invited paper. the analysis and selection of variables in linear regression," *Biometrics*, vol. 32, no. 1, pp. 1–49, 1976.

[2] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[3] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[4] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2006.

[5] J. Fan and R. Li, "Variable selection via nonconcave penalized likelihood and its oracle properties," *Journal of the American statistical Association*, vol. 96, no. 456, pp. 1348–1360, 2001.

[6] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight, "Sparsity and smoothness via the fused lasso," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 1, pp. 91–108, 2005.

[7] S. Banerjee, B. P. Carlin, and A. E. Gelfand, *Hierarchical modeling and analysis for spatial data*. CRC press, 2014.

[8] A. Healy and G. S. Lenz, "Presidential voting and the local economy: Evidence from two population-based data sets," *The Journal of Politics*, vol. 79, no. 4, pp. 1419–1432, 2017.

[9] T. Park and A. Reeves, "Local unemployment and voting for president: Uncovering causal mechanisms," *Political Behavior*, pp. 1–21, 2018.

[10] J. R. Wright, "Unemployment and the democratic electoral advantage," *American Political Science Review*, vol. 106, no. 4, pp. 685–702, 2012.

[11] J. Kim, E. Elliott, and D.-M. Wang, "A spatial analysis of county-level outcomes in us presidential elections: 1988–2000," *Electoral Studies*, vol. 22, no. 4, pp. 741–761, 2003.

[12] A. S. Fotheringham, C. Brunsdon, and M. Charlton, *Geographically weighted regression: the analysis of spatially varying relationships*. John Wiley & Sons, 2003.

[13] A. E. Gelfand, H.-J. Kim, C. Sirmans, and S. Banerjee, "Spatial modeling with spatially varying coefficient processes," *Journal of the American Statistical Association*, vol. 98, no. 462, pp. 387–396, 2003.

[14] F. Li and H. Sang, "Spatial homogeneity pursuit of regression coefficients for large datasets," *Journal of the American Statistical Association*, pp. 1–21, 2019.

[15] X. Zhang, J. Liu, and Z. Zhu, "Distributed linear model clustering over networks: A tree-based fused-lasso admm approach," *arXiv preprint arXiv:1905.11549*, 2019.

[16] Z. Ma, Y. Xue, and G. Hu, "Bayesian heterogeneity pursuit regression models for spatially dependent data," *arXiv preprint arXiv:1907.02212*, 2019.

[17] J. Zhu, H.-C. Huang, and P. E. Reyes, "On selection of spatial linear models for lattice data," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 72, no. 3, pp. 389–402, 2010.

[18] W. Feng, A. Sarkar, C. Y. Lim, and T. Maiti, "Variable selection for binary spatial regression: Penalized quasi-likelihood approach," *Biometrics*, vol. 72, no. 4, pp. 1164–1172, 2016.

[19] Y. E. Shin, H. Sang, D. Liu, T. A. Ferguson, and P. X. Song, "Autologistic network model on binary data for disease progression study," *Biometrics*, vol. 75, no. 4, pp. 1310–1320, 2019.

[20] A. L. Thurman, R. Fu, Y. Guan, and J. Zhu, "Regularized estimating equations for model selection of clustered spatial point processes," *Statistica Sinica*, pp. 173–188, 2015.

[21] A. Choiruddin, J.-F. Coeurjolly, F. Letué, *et al.*, "Convex and non-convex regularization methods for spatial point processes intensity estimation," *Electronic Journal of Statistics*, vol. 12, no. 1, pp. 1210–1255, 2018.

[22] H. Hoefling, "A path algorithm for the fused lasso signal approximator," *Journal of Computational and Graphical Statistics*, vol. 19, no. 4, pp. 984–1006, 2010.

[23] H. Zou, "The adaptive lasso and its oracle properties," *Journal of the American statistical association*, vol. 101, no. 476, pp. 1418–1429, 2006.

[24] V. Viallon, S. Lambert-Lacroix, H. Höfling, and F. Picard, "Adaptive generalized fused-lasso: Asymptotic properties and applications," 2013.

[25] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[26] X. Shen and H.-C. Huang, "Grouping pursuit through a regularization solution surface," *Journal of the American Statistical Association*, vol. 105, no. 490, pp. 727–739, 2010.

[27] T. B. Arnold and R. J. Tibshirani, "Efficient implementations of the generalized lasso dual path algorithm," *Journal of Computational and Graphical Statistics*, vol. 25, no. 1, pp. 1–27, 2016.

[28] O. H. M. Padilla, J. Sharpnack, J. G. Scott, and R. J. Tibshirani, "The dfs fused lasso: Linear-time denoising over general graphs.," *Journal of Machine Learning Research*, vol. 18, pp. 176–1, 2018.

[29] A. Barbero and S. Sra, "Modular proximal optimization for multidimensional total-variation regularization," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 2232–2313, 2018.

[30] J. Zhou, J. Liu, V. A. Narayan, and J. Ye, "Modeling disease progression via fused sparse group lasso," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1095–1103, ACM, 2012.

[31] J. Huang, P. Breheny, and S. Ma, "A selective review of group selection in high-dimensional models," *Statistical science: a review journal of the Institute of Mathematical Statistics*, vol. 27, no. 4, 2012.

[32] J. Hu, J. Huang, and F. Qiu, "A group adaptive elastic-net approach for variable selection in high-dimensional linear regression," *Science China Mathematics*, vol. 61, no. 1, pp. 173–188, 2018.

[33] P. Bühlmann and S. Van De Geer, *Statistics for high-dimensional data: methods, theory and applications*. Springer Science & Business Media, 2011.

[34] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical learning with sparsity: the lasso and generalizations*. Chapman and Hall/CRC, 2015.

[35] D. R. Hunter and K. Lange, "A tutorial on mm algorithms," *The American Statistician*, vol. 58, no. 1, pp. 30–37, 2004.

[36] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[37] N. A. Johnson, "A dynamic programming algorithm for the fused lasso and l 0-segmentation," *Journal of Computational and Graphical Statistics*, vol. 22, no. 2, pp. 246–260, 2013.

[38] L. Condat, "A direct algorithm for 1-d total variation denoising," *IEEE Signal Processing Letters*, vol. 20, no. 11, pp. 1054–1057, 2013.

[39] Y. Fan and C. Y. Tang, "Tuning parameter selection in high dimensional penalized likelihood," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 75, no. 3, pp. 531–552, 2013.

[40] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the royal statistical society: series B (statistical methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[41] F. Liébana-Cabanillas, L. J. Herrera, and A. Guillén, "Variable selection for payment in social networks: Introducing the hy-index," *Computers in Human Behavior*, vol. 56, pp. 45–55, 2016.

[42] F. K. Hamey, S. Nestorowa, S. J. Kinston, D. G. Kent, N. K. Wilson, and B. Göttgens, "Reconstructing blood stem cell regulatory network models from single-cell molecular profiles," *Proceedings of the National Academy of Sciences*, vol. 114, no. 23, pp. 5822–5829, 2017.

[43] R. Vershynin, *High-dimensional probability: An introduction with applications in data science*, vol. 47. Cambridge university press, 2018.

[44] J. A. Hartigan, "Direct clustering of a data matrix," *Journal of the American Statistical Association*, vol. 67, no. 337, pp. 123–129, 1972.

[45] S. Bergmann, J. Ihmels, and N. Barkai, "Iterative signature algorithm for the analysis of large-scale gene expression data," *Physical Review E*, vol. 67, no. 3, p. 031902, 2003.

[46] A. A. Shabalin, V. J. Weigman, C. M. Perou, A. B. Nobel, *et al.*, "Finding large average submatrices in high dimensional data," *The Annals of Applied Statistics*, vol. 3, no. 3, pp. 985–1012, 2009.

[47] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini, "Discovering local structure in gene expression data: the order-preserving submatrix problem," *Journal of Computational Biology*, vol. 10, no. 3-4, pp. 373–384, 2003.

[48] T. Murali and S. Kasif, "Extracting conserved gene expression motifs from gene expression data," in *Biocomputing 2003*, pp. 77–88, World Scientific, 2002.

[49] J. Yang, H. Wang, W. Wang, and P. Yu, "Enhanced biclustering on expression data," in *Bioinformatics and Bioengineering, 2003. Proceedings. Third IEEE Symposium on*, pp. 321–327, IEEE, 2003.

[50] A. Prelić, S. Bleuler, P. Zimmermann, A. Wille, P. Bühlmann, W. Gruissem, L. Hennig, L. Thiele, and E. Zitzler, "A systematic comparison and evaluation of biclustering methods for gene expression data," *Bioinformatics*, vol. 22, no. 9, pp. 1122–1129, 2006.

[51] A. Tanay, R. Sharan, and R. Shamir, "Discovering statistically significant biclusters in gene expression data," *Bioinformatics*, vol. 18, no. suppl_1, pp. S136–S144, 2002.

[52] B. Pontes, R. Giráldez, and J. S. Aguilar-Ruiz, "Biclustering on expression data: A review," *Journal of biomedical informatics*, vol. 57, pp. 163–180, 2015.

[53] L. Lazzeroni and A. Owen, "Plaid models for gene expression data," *Statistica sinica*, pp. 61–86, 2002.

[54] N. Asgarian and R. Greiner, "Using rank-1 biclusters to classify microarray data," *Dept. Computing Science, and the Alberta Ingenuity Center for Machine Learning, Univ. Alberta, Edmonton, AB, Canada, T6G2E8*, 2006.

[55] M. Lee, H. Shen, J. Z. Huang, and J. Marron, "Biclustering via sparse singular value decomposition," *Biometrics*, vol. 66, no. 4, pp. 1087–1095, 2010.

[56] K. Chen, K.-S. Chan, and N. C. Stenseth, "Reduced rank stochastic regression with a sparse singular value decomposition," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 74, no. 2, pp. 203–221, 2012.

[57] M. Sill, S. Kaiser, A. Benner, and A. Kopp-Schneider, "Robust biclustering by sparse singular value decomposition incorporating stability selection," *Bioinformatics*, vol. 27, no. 15, pp. 2089–2097, 2011.

[58] D. M. Witten, R. Tibshirani, and T. Hastie, "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis," *Biostatistics*, vol. 10, no. 3, pp. 515–534, 2009.

[59] K. M. Tan and D. M. Witten, "Sparse biclustering of transposable data," *Journal of Computational and Graphical Statistics*, vol. 23, no. 4, pp. 985–1008, 2014.

[60] C. Gao, Y. Lu, Z. Ma, and H. H. Zhou, "Optimal estimation and completion of matrices with biclustering structures," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 5602–5630, 2016.

[61] E. C. Chi, G. I. Allen, and R. G. Baraniuk, "Convex biclustering," *Biometrics*, vol. 73, no. 1, pp. 10–19, 2017.

[62] S. Lee and J. Z. Huang, "A biclustering algorithm for binary matrices based on penalized bernoulli likelihood," *Statistics and Computing*, vol. 24, no. 3, pp. 429–441, 2014.

[63] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *Journal of Machine Learning Research*, vol. 5, no. Nov, pp. 1457–1469, 2004.

[64] Y. Zhou, R. Jin, and S. C.-H. Hoi, "Exclusive lasso for multi-task feature selection," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 988–995, 2010.

[65] P. Zhao, G. Rocha, and B. Yu, "The composite absolute penalties family for grouped and hierarchical variable selection," *The Annals of Statistics*, pp. 3468–3497, 2009.

[66] F. Campbell, G. I. Allen, *et al.*, "Within group variable selection through the exclusive lasso," *Electronic Journal of Statistics*, vol. 11, no. 2, pp. 4220–4257, 2017.

[67] A. Beck, "On the convergence of alternating minimization for convex programming with applications to iteratively reweighted least squares and decomposition schemes," *SIAM Journal on Optimization*, vol. 25, no. 1, pp. 185–209, 2015.

[68] D. Kong, R. Fujimaki, J. Liu, F. Nie, and C. Ding, "Exclusive feature learning on arbitrary structures via $\ell_{1,2}$-norm," in *Advances in Neural Information Processing Systems*, pp. 1655–1663, 2014.

[69] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[70] X. Qi, R. Luo, and H. Zhao, "Sparse principal component analysis by choice of norm," *Journal of multivariate analysis*, vol. 114, pp. 127–160, 2013.

[71] D. M. Witten, R. Tibshirani, and T. Hastie, "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis," *Biostatistics*, vol. 10, no. 3, pp. 515–534, 2009.

[72] N. Meinshausen and P. Bühlmann, "Stability selection," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 72, no. 4, pp. 417–473, 2010.

[73] S. Hochreiter, U. Bodenhofer, M. Heusel, A. Mayr, A. Mitterecker, A. Kasim, T. Khamiakova, S. Van Sanden, D. Lin, W. Talloen, *et al.*, "Fabia: factor analysis for bicluster acquisition," *Bioinformatics*, vol. 26, no. 12, pp. 1520–1527, 2010.

[74] M. J. Zaki, W. Meira Jr, and W. Meira, *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press, 2014.

[75] M. Ximerakis, S. L. Lipnick, B. T. Innes, S. K. Simmons, X. Adiconis, D. Dionne, B. A. Mayweather, L. Nguyen, Z. Niziolek, C. Ozek, *et al.*, "Single-cell transcriptomic profiling of the aging mouse brain," *Nature Neuroscience*, vol. 22, no. 10, pp. 1696–1708, 2019.