THE UNIVERSITY OF CHICAGO

LEARNING MEANINGFUL REPRESENTATIONS OF DATA WITH EMPIRICAL BAYES METHODS

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF STATISTICS

BY
JOONSUK KANG

CHICAGO, ILLINOIS
AUGUST 2024

TABLE OF CONTENTS

LIST OF FIGURES v

LIST OF TABLES vii

ACKNOWLEDGMENTS viii

ABSTRACT ix

INTRODUCTION 1

1 SCALABLE EMPIRICAL BAYES MATRIX FACTORIZATION VIA ALTERNATING LEAST SQUARES-TYPE UPDATES 7
- 1.1 Introduction 7
- 1.2 Updating q_l given (g_l, q_f, τ): n independent regression problems 9
  - 1.2.1 Interpreting a subproblem as a regression problem 11
  - 1.2.2 solREG: coordinate descent algorithm for the regression problem . . 12
  - 1.2.3 solQ: coordinate descent algorithm for updating q_l 16
  - 1.2.4 solQb: block coordinate descent algorithm for jointly updating q_l and g_l 17
- 1.3 The altflash Algorithm 18
  - 1.3.1 altflash: a denoising approach for matrix factorization 19
  - 1.3.2 Initialization 22
- 1.4 Results 22
  - 1.4.1 Setting 1: Normal-Normal 23
  - 1.4.2 Setting 2: PointNormal-PointNormal 23
  - 1.4.3 Setting 3: PointNormal-PointExponential 24
- 1.5 Discussion 26
- 1.6 Appendix 27
  - 1.6.1 Rearranging the core computation of the evidence lower bound 27
  - 1.6.2 Adding a scaling matrix D: $\mathbf{X} \approx \mathbf{L}^T\mathbf{D}\mathbf{F}$ 28

2 EMPIRICAL BAYES COVARIANCE DECOMPOSITION, AND A SOLUTION TO THE MULTIPLE TUNING PROBLEM IN SPARSE PCA 30
- 2.1 Introduction 30
- 2.2 Notation 32
- 2.3 A Penalized PCA Criterion 32
  - 2.3.1 A Penalized PCA Criterion 32
  - 2.3.2 Uniting Previous Sparse PCA Methods 34
  - 2.3.3 BISPCA, a “block" algorithm for penalized PCA with separable penalties 35
  - 2.3.4 Connections with other algorithms 37
- 2.4 A Penalized Covariance Decomposition Criterion 39
  - 2.4.1 Sufficient Statistic and Efficient Computation 41
- 2.5 EBCD: An Empirical Bayes Solution to the Multiple Tuning Problem 43

2.5.1 The EBCD Model 43
2.5.2 Connecting EBCD and the penalized PCA criterion 47
2.5.3 Efficient Computation 48
2.5.4 Extensions and variations 49
2.6 Practical Issues 50
2.6.1 Initialization 50
2.6.2 Choice of K 50
2.6.3 Choice of prior 51
2.7 Empirical Results 52
2.7.1 Simulation 52
2.7.2 Stock Market Data 56
2.8 Discussion 59
2.9 Data and Resources 61
2.10 Appendix 61
2.10.1 Proof of Theorem 2.1 61
2.10.2 Proof of Proposition 2.3 63
2.10.3 Proof of Proposition 2.4 63
2.10.4 Extension 1. Scaled versions of the sparse PCA criterion 63
2.10.5 Extension 2. Column-wise variances 65

3 GENETIC DRIFT-BASED INFERENCE OF POPULATION STRUCTURE . . . 68
3.1 Introduction 68
3.2 Description of the method 70
3.2.1 The DRIFT method 70
3.2.2 Tree-based estimation of genetic drifts 73
3.2.3 Treelax: Matrix decomposition-based estimation of genetic drifts . . . 74
3.2.4 Adding a shared drift component 77
3.2.5 Computing individual-specific drift components 78
3.2.6 A decomposition of genetic distance 78
3.2.7 Contrast with Tree-and-migrations-based estimation of genetic drifts 78
3.3 Verification and comparison 79
3.3.1 Four populations out of Africa simulation using stdpopsim 79
3.3.2 Simulation results 81
3.4 Applications 88
3.4.1 Homo Sapiens: The 1000 Genomes Project data 88
3.5 Discussion 94
3.5.1 Benefits of a modular approach 94
3.5.2 Role of ancestral alleles and outgroup 94
3.6 Appendix: Branch length adjustments for treeness violations 96
3.6.1 Problem setup 96
3.6.2 Allowed operations 97
3.6.3 Solution 97

LIST OF FIGURES

1.1 Comparison of ELBO between altflash and flashier, under the setting $(\mathcal{G}_l, \mathcal{G}_f) =$ ('normal', 'normal'). The y-axis represents the logarithmic difference from the maximum ELBO, while the x-axis corresponds to the respective time points . . 24
1.2 Comparison of ELBO between altflash and flashier, under the setting $(\mathcal{G}_l, \mathcal{G}_f) =$ ('point_normal', 'point_normal') and $(\mathcal{G}_l, \mathcal{G}_f) =$ ('point_normal', 'point_exponential'). The y-axis represents the logarithmic difference from the maximum ELBO, while the x-axis corresponds to the respective time points. 25
1.3 Comparison of ELBO between altflash and flashier, under the setting $(\mathcal{G}_l, \mathcal{G}_f) =$ ('point_normal', 'point_normal') and $(\mathcal{G}_l, \mathcal{G}_f) =$ ('point_normal', 'point_exponential'). The y-axis represents the logarithmic difference from the maximum ELBO, while the x-axis corresponds to the number of iterations. 26

2.1 Examples of posterior mean shrinkage operator $S(\mathbf{x}, s^2 = 1, g = g(\cdot; \pi, b))$ induced by Laplace slab priors $g(x; \pi, b) = (1 - \pi)\delta_0(x) + \pi\text{Laplace}(x; 0, b)$. Note how π controls shrinkage near 0 (small π yielding more shrinkage), while the scale parameter controls shrinkage further away from 0. 52
2.2 Simulation results comparing the performance of different methods in terms of three measures: angle between true and estimated principal components (PCs), difference between population covariance matrix and estimated covariance matrix, and distance with optimal rotation. 55
2.3 Holding period returns by sectors during three phases: "Working from home" (Jan 2020 - Nov 2020), "Reopening" (Nov 2020 - Dec 2021), and "Inflation" (Jan 2022 - Nov 2022). 57
2.4 Comparison of PCA loadings and estimated posterior mean loadings from `EBCD-pl`. (To facilitate comparisons we post-processed posterior mean loadings to have unit norm.) 58
2.5 Sectors projected on the SMB-HML plane. Each sector is positioned according to its loadings on the Fama-French SMB and HML factors, and is colored based on its loadings on the second and third principal components (PCs) from the `EBCD-pl` method (or PCA). 60

3.1 An illustration of (a) STRUCTURE bar plot, (b) tree-based genetic drift estimation, and (c) tree-based individual DRIFT bar plot, using the 1000 Genomes Project data. 69
3.2 Overview of the DRIFT method's three-step process. 71
3.3 An illustration of the 'four populations out of Africa' model [Jouganous et al., 2017], obtained from the stdpopsim catalog. 80
3.4 (a) STRUCTURE bar plot and (b) estimated tree in simulation. 81
3.5 Genetic distance, its decomposition, and residual by tree method, in simulation. 82
3.6 Genetic distance, its decomposition, and residual by Treelax method, in simulation. 85
3.7 TreeMix results with (a) no migration (tree), (b) one migration, and (c) two migrations, in simulation. 85

3.8 (a) Tree-based and (b) Treelax-based DRIFT bar plots in simulation. 86
3.9 Analysis of the 1000 Genomes Project Data. (a) STRUCTURE bar plot, (b) tree-based genetic drift estimation, and (c) tree-based and (d) Treelax-based DRIFT bar plots. 90

LIST OF TABLES

2.1 Sparse PCA Algorithms. $S_\rho(\cdot;\lambda_k)$ denotes the proximal operator of the penalty function $\rho(\cdot;\lambda_k)$, and S_1 denotes the soft thresholding operator, which is the proximal operator of the L_1 penalty. We use $S_\rho(\mathbf{A};\boldsymbol{\lambda})$ to denote the vector whose kth element is $S_\rho(\boldsymbol{a}_k;\lambda_k)$. The U factor of the polar decomposition is denoted as Polar.U, and the Q factor of the QR decomposition is denoted as QR.Q. $\mathbf{Z}^\perp$ represents an orthonormal basis that is orthogonal to $\mathbf{Z}$. The function G calculates the estimated prior from the empirical Bayes normal means model, and the function S returns the corresponding posterior mean vector (see Definition 2.2 and Remark 2.2). 38

3.1 Populations' drift memberships in (A) tree method and (B) Treelax method, and (C) membership differences in the two methods, in simulation. 84

3.2 Populations' drift memberships in (A) tree method and (B) Treelax method, and (C) membership differences in the two methods, in the 1000 Genomes Project data analysis. 92

ACKNOWLEDGMENTS

First, I would like to thank my advisor Matthew Stephens. Our numerous conversations were the highlight of my PhD. Through these conversations, I learned how to think, write, and talk like a researcher, and I discovered the joy of embracing the unknown. I am grateful for the encouragement he gave me as a student and the trust he showed me as a colleague.

I was fortunate to have John Novembre on my committee, whose intuition and knowledge in statistical genetics sparked many fruitful results in this dissertation. John also helped me communicate my research more effectively to a broader audience beyond statisticians. I'm grateful to Dan Nicolae for his advice, mentorship, and leadership at various points in my PhD, both while serving as the chair of the department and as a member of my committee. I would also like to thank Jingshu Wang, whose questions and suggestions improved the statistical rigor of this work.

My journey as a statistician would not have started without Michael Stein, who has shaped how I think about data and has been an excellent role model throughout. I also thank Mihai Anitescu and Mei Wang for their guidance and support, especially during my earlier years at UChicago. Many thanks to Keisha Prowoznik and Kirsten Wellman for making my life much easier as a grad student and teaching assistant.

This dissertation has benefited from discussions with members of the Stephens Lab. My special thanks go to Peter Carbonetto for providing rounds of helpful comments and creating a welcoming environment, and to Jason Willwerscheid for setting footsteps I can follow.

Finally, I thank my family for their love and support, especially my wife Boram for being my best friend in life.

ABSTRACT

Matrix factorization methods are widely used to uncover hidden structures within data represented by matrices. The choice of method depends on the desired data representation, such as sparsity, orthogonality, and nonnegativity. The Bayesian approach can effectively model these desired representations. Specifically, the empirical Bayes approach avoids the need to manually specify hyperparameters for each column of factor and/or loading in the model.

The first chapter develops an alternative algorithm for fitting the Empirical Bayes Matrix Factorization model. The existing 'flash' algorithm updates a single factor-loading vector pair at a time while holding others fixed. Instead, our alternating least squares-type algorithm updates the entire factor matrix (or loading matrix) at once while fixing the entire loading matrix (or factor matrix). This update allows for efficient parallel implementation as it can be interpreted as solving multiple independent regression problems. The second chapter introduces a flexible class of empirical Bayes matrix factorization methods, in which a data matrix is approximated by a product of an orthogonal factor matrix and a loading matrix with column-specific priors. We demonstrate that using sparsity-inducing priors on the loading matrix leads to a sparse PCA method. Importantly, our method avoids the "multiple tuning problem" commonly encountered in sparse PCA. The final chapter presents a matrix factorization method, motivated by population genetics. The method factorizes a genotype matrix into a drift factor matrix and a drift membership matrix by combining a STRUCTURE-type method and a drift estimation method. Unlike previous approaches that represent individuals' genotypes using populations, our method emphasizes shared genetic variation across individuals by representing individuals' genotypes using genetic drifts, which are shared across populations. To estimate the drift factors and memberships, we propose a symmetric nonnegative matrix factorization method that penalizes deviations from a tree-based initial estimate.

INTRODUCTION

Statisticians discern signal from noise, but what precisely is signal? Essentially, signal embodies a meaningful representation of the data. This dissertation documents the findings of an exploration into the most effective methods for representing data meaningfully. To make this inquiry more tangible, I investigated the optimal ways to depict the shared genetic makeup of individuals, based on their genotype data. Through this journey, an intriguing yet somewhat disappointing truth emerged: there is no single definitive method for capturing the signal. Instead, I uncovered several compelling approaches to achieving meaningful representation of complex data. This dissertation introduces three of these methods.

Background The overarching theme of the three methods is the pursuit of the "drift factorization". The drift factorization is an idea that has been proposed to improve upon the STRUCTURE-type methods that represent individuals as mixtures of populations. This population-based representation of individuals has an intuitive appeal, which is evidenced by the commercial success of 23andMe, Ancestry, and the like. The STRUCTURE paper [Pritchard et al., 2000] is highly influential with over 38,000 citations on Google Scholar (as of April 3, 2024). Similarly, the ADMIXTURE [Alexander et al., 2009] and fastSTRUCTURE [Raj et al., 2014] papers have also garnered significant attention.

However, a limitation of the STRUCTURE-type methods is that the genetic similarity across populations is obscured in its usual visualization of the "STRUCTURE bar plot", in which individuals are positioned along the x-axis and each individual's population memberships are stacked along the y-axis. To address this limitation, the concept of drift factorization has been proposed, aiming to represent individuals using genetic drifts rather than populations.

We can interpret these two types of methods, one population-based and the other drift-based, as matrix factorization methods. Suppose we have a genotype matrix $\mathbf{G} \in \{0, 1, 2\}^{S \times N}$

from N individuals measured at S SNPs. The STRUCTURE-type method finds a population-based decomposition of the form

$$\mathbf{G} \approx 2\mathbf{P}\mathbf{Q}^T \tag{1}$$

where $\mathbf{P} \in [0,1]^{S\times K}$ is the population allele matrix of the K estimated latent populations and $\mathbf{Q} \in [0,1]^{N\times K}$ is the individuals' population membership matrix. Based on this population-based decomposition, each individual can be represented as a mixture of the K populations, with membership ratio $Q_{n,k}$. This representation is effectively visualized in the STRUCTURE bar plot (where stacked bars indicate the proportion of an individual's ancestry belonging to each population).

In contrast, the drift-based method finds a genetic drift-based decomposition of the form

$$\mathbf{G} \approx 2\mathbf{Z}\mathbf{M}^T \tag{2}$$

where $\mathbf{Z} \in \mathbb{R}^{S\times J}$ is the genetic drift factor matrix encoding the allele frequency changes due to the J genetic drifts and $\mathbf{M} \in [0,1]^{N\times J}$ is the individuals' drift membership matrix. In the drift-based decomposition, each individual is represented as a mixture of J genetic drifts.

The drift factorization idea is not new. It has been explored in the work "Emphasizing shared evolutionary histories when inferring representations of population structure" by Joseph H. Marcus, Jason Willwerscheid, Peter Carbonetto, John Novembre, and Matthew Stephens, which was published as a Chapter in Joseph H. Marcus's thesis [Marcus, 2020].

They introduced the concept of the drift factorization and implemented it using the Empirical Bayes Matrix Factorization (EBMF) framework [Wang, 2017, Wang and Stephens, 2021]:

$$\mathbf{G} = \mathbf{Z}\mathbf{M}^T + \mathbf{E} \tag{3}$$

$$Z_{s,j} \sim g_z^{(j)} \in \mathcal{G}_z^{(j)} \tag{4}$$

$$M_{n,j} \sim g_m^{(j)} \in \mathcal{G}_m^{(j)} \tag{5}$$

$$E_{s,n} \sim N(\cdot; 0, \sigma_{s,n}^2) \tag{6}$$

where $g_z^{(j)}$ is the prior for the j-th column of $\mathbf{Z}$, which is estimated within the prior family $\mathcal{G}_z^{(j)}$; $g_m^{(j)}$ is the prior for the j-th column of $\mathbf{M}$, which is estimated within the prior family $\mathcal{G}_m^{(j)}$; and we ignore the constant 2, meaning that humans are diploid, in the general EBMF framework. Specifically, they estimated the drift factorization by putting column-wise centered Gaussian priors $\{g_z^{(j)}\}_{j=1}^J$ on the columns of $\mathbf{Z}$ (the genetic drift factor matrix) and column-wise bimodal priors $\{g_m^{(j)}\}_{j=1}^J$ on the columns of $\mathbf{M}$ (the individuals' drift membership matrix). This investigation was made possible due to the improved implementation of the EBMF and Empirical Bayes Normal Means (EBNM), which allowed the specification of flexible prior families and enabled scalable inferences [Willwerscheid, 2021, Willwerscheid and Stephens, 2021].

However, simulations revealed that their method struggles to reliably estimate drift memberships. We needed an improvement in modeling and implementation to find a path to robust drift factorization.[1]

Chapter 1: altflash The first approach was to develop an alternative algorithm to fit an EBMF model. The existing EBMF-fitting algorithm, called "flash" (developed in Wang and Stephens [2021]; R implementation "flashr")[2], updates a single factor-loading vector pair

1. Willwerscheid [2021] developed a method for "divergence factorization", which is claimed to be easier to identify and estimate than drift factorization.

2. An improved R implementation "flashier" was introduced by Willwerscheid [2021].

$(\mathbf{z}_j, \mathbf{m}_j)$ at a time while holding others fixed. Our alternative approach, called "altflash", is based on the alternating least squares-type updates. The altflash updates the entire $\mathbf{Z}$ matrix (or the entire $\mathbf{M}$ matrix) at once while holding the other matrix $\mathbf{M}$ (or $\mathbf{Z}$) fixed.

This alternative way of updating can lead to a higher objective function value of the same underlying EBMF model, by escaping bad local optima. Another important benefit of the altflash algorithm is its potential for even larger-scale inference due to its embarrassingly parallel structure. We show that each update of the entire $\mathbf{Z}$ matrix (or the entire $\mathbf{M}$ matrix) is equivalent to solving S (or N) independent regression problems, allowing for efficient parallel implementation. We also provide an interpretation of the altflash algorithm as a denoising version of orthogonal iteration or accelerated hierarchical alternating least squares method, depending on the support of specified prior families.

Chapter 2: EBCD The second approach to the drift factorization involved tweaking the EBMF model, by directly encoding desired properties into model assumptions. In the drift factorization, the genetic drift factors $\mathbf{z}_j$'s are expected to be mutually orthogonal. Thus, we developed another matrix factorization framework that incorporates this assumption. We call our method the Empirical Bayes Covariance Decomposition (EBCD), based on the equivalence property where the covariance matrix (or the Gram matrix) $\mathbf{G}^T\mathbf{G}$ is approximated by the symmetric matrix $\mathbf{M}\mathbf{M}^T$. The EBCD model can be specified as

$$\mathbf{G} = \mathbf{Z}\mathbf{M}^T + \mathbf{E} \tag{7}$$

$$M_{n,j} \sim g_m^{(j)} \in \mathcal{G}_m^{(j)} \tag{8}$$

$$E_{s,n} \sim N(\cdot; 0, \sigma^2) \tag{9}$$

where $\mathbf{Z}$ satisfies $\mathbf{Z}^T\mathbf{Z} = \mathbf{I}$. By baking the desired orthogonality property directly into the model, we can force the orthogonality in the genetic drift factors, which reduces the search space and enables efficient computation.

We demonstrate that the EBCD method can serve as a sparse PCA method effectively by specifying the prior families ($\mathcal{G}_m^{(j)}$) as sparsity-inducing. Additionally, we present a unified framework that extends existing sparse PCA methods as specific instances, enhancing our understanding of these methods. Of particular importance, we highlight the computational challenge posed by cross-validation for multiple tuning parameters in sparse PCA methods, which we term the 'Multiple Tuning Problem.' We argue that the EBCD method offers a solution to this problem

Chapter 3: DRIFT The most successful approach to obtain the drift factorization was to go back to the roots of the problem. Drift factorization aims to clarify the obscured genetic similarity across populations in the good old STRUCTURE-type methods: $\mathbf{G} \approx 2\mathbf{P}\mathbf{Q}^T$. Individuals' population-based representation is encoded in the $\mathbf{Q}$ matrix; the STRUCTURE bar plot is essentially visualizing the $\mathbf{Q}$ matrix. The genetic similarity across populations, in fact, is readily available in the $\mathbf{P}$ matrix of populations' allele frequencies. For example, the problem of reconstructing a bifurcating tree from population allele frequencies has been extensively studied in Cavalli-Sforza et al. [1964], Cavalli-Sforza and Edwards [1967], Felsenstein [1973, 1981], Pickrell and Pritchard [2012].

Building on these previous works, we framed the task of finding the drift factorization as a two-step modular approach combining a STRUCTURE-type method that finds $\mathbf{G} \approx 2\mathbf{P}\mathbf{Q}^T$ and a $\mathbf{P}$-factorization method that finds $\mathbf{P} \approx \mathbf{Z}\mathbf{L}^T$ where $\mathbf{Z}$ is the genetic drift factor matrix and $\mathbf{L}$ is the populations' drift membership matrix. By setting individuals' drift membership as $\mathbf{M} := \mathbf{Q}\mathbf{L}$, we obtain the drift factorization $\mathbf{G} \approx 2\mathbf{Z}\mathbf{M}^T$. For the $\mathbf{P}$-factorization method, we consider the tree estimation method and its variation that incorporates migration events; and develop a symmetric nonnegative matrix factorization method that allows for deviations from a tree-based initial estimate, relaxing the strict tree assumption. This modular approach significantly reduces the complexity of the problem by replacing a $\mathbf{G}$-factorization problem with a much simpler $\mathbf{P}$-factorization problem (where $\mathbf{P}^T\mathbf{P} \in \mathbb{R}^{K \times K}$

is the sufficient statistics), while achieving improved interpretability by connecting back to its root, the STRUCTURE method. Hence, this approach has conceptual appeals that the concept of genetic drift is applicable to populations rather than individuals and, therefore, first-assuming-and-then-relaxing a tree structure is more consistent with populations' data rather than individuals' data.

Signal is in the eye of the beholder The three projects tackle a single drift factorization problem, yet they arrive at different conclusions on how to capture the signal lurking in the data. Depending on your objective, you may obtain an algorithmic improvement to an existing model, an orthogonal factor restriction, or an interpretable simplified model. So, be careful what you wish for.

CHAPTER 1

SCALABLE EMPIRICAL BAYES MATRIX FACTORIZATION VIA ALTERNATING LEAST SQUARES-TYPE UPDATES

1.1 Introduction

Matrix factorization methods are widely used to uncover structures from a matrix-variate data. Classical examples include the principal components analysis (PCA) and factor analysis. Modern matrix factorization methods aim to find structured signals from data, such as sparsity and nonnegativity, by imposing penalties or assigning priors to promote such structures. For example, sparse PCA [Zou et al., 2006, Witten et al., 2009, Journée et al., 2010, Ma, 2013] finds a modified version of PCA that has sparse loadings or sparse factors, and nonnegative matrix factorization [Lee and Seung, 1999, Gillis, 2021] approximates a nonnegative matrix using a nonnegative loading and a nonnegative factor matrix.

On the Bayesian side, Bayesian matrix factorization models have been developed [Tipping and Bishop, 1999, Bishop, 1999, Lim and Teh, 2007, Salakhutdinov and Mnih, 2008]. The desired structural property such as sparsity and nonnegativity can be easily baked into prior structures. The empirical Bayes matrix factorization (EBMF; Wang and Stephens [2021]) model establishes a general framework for Bayesian matrix factorization, in which the priors can be flexibly defined. The EBMF models a data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ as the sum of a rank K signal and the error:

$$\mathbf{X} = \mathbf{L}\mathbf{F}^T + \mathbf{E} = \sum_{k=1}^{K} \mathbf{l}_k \mathbf{f}_k^T + \mathbf{E} \tag{1.1}$$

where $\mathbf{L} \in \mathbb{R}^{n \times K}$ is the loading matrix, $\mathbf{F} \in \mathbb{R}^{p \times K}$ is the factor matrix, and $\mathbf{E} \in \mathbb{R}^{n \times p}$ is the error matrix. We use $\mathbf{l}_k$ (or $\mathbf{f}_k$) to denote the k-th column of $\mathbf{L}$ (or $\mathbf{F}$). A scalar $L_{i,k}$ (or $F_{j,k}$) denotes the (i, k)-th element of $\mathbf{L}$ (or the (j, k)-th element of $\mathbf{F}$). The entries of $\mathbf{l}_k$ (or

$\mathbf{f}_k$) is modeled to be drawn iid from a prior $g_{l,k} \in \mathcal{G}_{l,k}$ (or $g_{f,k} \in \mathcal{G}_{f,k}$):

$$L_{i,k}, \ldots, L_{n,k} \sim^{\text{iid}} g_{l,k}, \quad g_{l,k} \in \mathcal{G}_{l,k} \tag{1.2}$$

$$F_{j,k}, \ldots, F_{p,k} \sim^{\text{iid}} g_{f,k}, \quad g_{f,k} \in \mathcal{G}_{f,k} \tag{1.3}$$

where $\{(\mathcal{G}_{l,k}, \mathcal{G}_{f,k})\}_{k=1}^{K}$ are user-specified prior families, such as the point-Gaussian prior family or the point-Laplace prior family; the priors $\{(g_{l,k}, g_{f,k})\}_{k=1}^{K}$ are estimated within the specified prior families. While the error term is originally modeled with a general structure ($E_{ij} \sim^{\text{iid}} N(0, 1/\tau_{i,j})$, $\tau := (\tau_{i,j}) \in \mathcal{T}$), in this chapter we add a simplifying assumption that $\tau_{i,j} = \tau$ for some $\tau > 0$.

To estimate the EBMF model, Wang and Stephens [2021] propose imposing the variational approximation of the form

$$q(\mathbf{l}_1, \ldots, \mathbf{l}_K, \mathbf{f}_1, \ldots, \mathbf{f}_K) = \left(\prod_k q_{l,k}(\mathbf{l}_k)\right)\left(\prod_k q_{f,k}(\mathbf{f}_k)\right) \tag{1.4}$$

where q denotes the variational posterior. With the variational approximation, the evidence lower bound (ELBO) of the model can be simplified as

$$F(q, g_l, g_f, \tau) = \mathbb{E}_{q_l, q_f}\left[\log p(\mathbf{X}|\mathbf{L}, \mathbf{F}; \tau)\right] + \sum_k \mathbb{E}_{q_{l,k}}\left[\log \frac{g_{l,k}(\mathbf{l}_k)}{q_{l,k}(\mathbf{l}_k)}\right] + \sum_k \mathbb{E}_{q_{f,k}}\left[\log \frac{g_{f,k}(\mathbf{f}_k)}{q_{f,k}(\mathbf{f}_k)}\right]. \tag{1.5}$$

Henceforth, we drop the subscripts ($q_l, q_{l,k}, q_f, q_{f,k}$) from the expectation operator unless it could be confusing.

To solve the ELBO maximization problem, Wang and Stephens [2021] develop the "flash" algorithm that iteratively updates the prior ($g_{l,k}, g_{f,k}$) and the variational posterior ($q_{l,k}, q_{f,k}$) corresponding to one column of $\mathbf{L}$ and $\mathbf{F}$ at a time, and implemented it as the flashr software in R.[1] Willwerscheid [2021] develops an improved implementation of the flash algorithm as the flashier software in R.

Note that the EBMF is a model, flash is an algorithm to fit the EBMF model, and flashr and flashier are software implementations of the flash algorithm.

To make the estimation of an EBMF model more scalable, we propose an alternative algorithm to fit the EBMF model (by maximizing the ELBO (1.5) under the variational approximation (1.4)). Our algorithm, "altflash", is an algorithm that utilizes alternating least squares-type updates, based on the observation that the core part of the ELBO maximization can be reduced into regression problems.

1.2 Updating q_l given (g_l, q_f, τ): n independent regression problems

In this section, we derive an algorithm that updates q_l given (g_l, q_f, τ). Once we obtain the update rule for q_l, the update rule for q_f given (g_f, q_l, τ) can be easily obtained due to the symmetry in $\mathbf{L}$ and $\mathbf{F}$.

The ELBO (1.5) maximization problem with respect to q_l can be written as

$$\max_{q_{l,1},\cdots,q_{l,K}} \Bigg(\tau \sum_k \left(\mathbb{E}[\mathbf{l}_k]^T \mathbf{X} \mathbb{E}[\mathbf{f}_k] - \frac{1}{2}\mathbb{E}[\mathbf{f}_k^T \mathbf{f}_k]\mathbb{E}[\mathbf{l}_k^T \mathbf{l}_k] \right) \\ - \tau \sum_{1\le j<k\le K} \mathbb{E}[\mathbf{f}_k]^T \mathbb{E}[\mathbf{f}_j] \mathbb{E}[\mathbf{l}_j]^T \mathbb{E}[\mathbf{l}_k] + \sum_k \mathbb{E}\left[\log \frac{g_{l,k}(\mathbf{l}_k)}{q_{l,k}(\mathbf{l}_k)}\right] \Bigg); \tag{1.6}$$

the derivation is shown in Appendix 1.6.1.

Under the variational approximation framework (1.4), the variational posterior q_l factors into $q_{l,1}, \ldots, q_{l,K}$. The subproblem of optimizing over $q_{l,k}$ given $\{q_{l,j}\}_{j\neq k}$ can be simplified

1. The flash algorithm includes two subalgorithms: a "greedy" and a "backfitting" [Breiman and Friedman, 1985] algorithm. The greedy algorithm is used to add an additional factor to the model, and the backfitting algorithm is used to iteratively refine each factor with the other factors fixed. Both of these subalgorithms refine each factor $(\mathbf{l}_k, \mathbf{f}_k)$ at a time, whether it is a part of adding a new factor or refining all the existing factors.

as

$$\max_{q_{l,k}} \mathbb{E}\Bigg[\sum_{i=1}^{n} \left(\tau \mathbb{E}[\mathbf{f}_k]^T \left(\mathbf{X}_{i,:}^T - \sum_{j \neq k} \mathbb{E}[\mathbf{f}_j]\mathbb{E}[L_{i,j}] \right) L_{i,k} - \frac{\tau \mathbb{E}[\mathbf{f}_k^T \mathbf{f}_k]}{2} L_{i,k}^2 + \log g_{l,k}(L_{i,k}) \right) - \log q_{l,k}(\mathbf{l}_k) \Bigg]. \tag{1.7}$$

where $L_{i,k}$ is the i-th coordinate of $\mathbf{l}_k$ (equivalently, the (i,k)-th entry of $\mathbf{L}$); $\mathbf{X}_{i,:} = (X_{i,1}, \ldots, X_{i,p})$ is a row vector and $\mathbf{X}_{i,:}^T$ is a column vector. Therefore, the solution to the ELBO optimization problem with respect to $q_{l,k}$ is obtained as

$$\begin{aligned} q_{l,k}(\mathbf{l}_k) &= \prod_{i=1}^{n} q_{l,i,k}(L_{i,k}) \qquad (1.8) \\ &\propto \prod_i g_{l,k}(L_{i,k}) \exp\left(\tau \mathbb{E}[\mathbf{f}_k]^T \left(\mathbf{X}_{i,:}^T - \sum_{j \neq k} \mathbb{E}[\mathbf{f}_j]\mathbb{E}[L_{i,j}] \right) L_{i,k} - \frac{\tau \mathbb{E}[\mathbf{f}_k^T \mathbf{f}_k]}{2} L_{i,k}^2 \right). \end{aligned}$$

The solution for $q_{l,k}$ reveals two interesting facts.

Fact 1. The solution for $q_{l,k}$ factorizes over i. That is, $q_{l,k}(\mathbf{l}_k) = \prod_i q_{l,i,k}(L_{i,k})$ for coordinate-wise posteriors. Hence, $q_l(\mathbf{L})$ fully factorizes as $q_l(\mathbf{L}) = \prod_k \prod_i q_{l,i,k}(L_{i,k})$. Note that after assuming that $q_l(\mathbf{L})$ factorizes into $\prod_k q_{l,k}(\mathbf{l}_k)$, we have the full coordinate-wise factorization for free.

Fact 2. The solution for the coordinate-wise posterior $q_{l,i,k}$ (1.8) depends on $\{q_{l,j}\}_{j \neq k}$ only through $\{\mathbb{E}[L_{i,j}]\}_{j \neq k}$, the posterior mean of $\mathbf{L}_{i,-k}$ that shares the same row index i. This means that the ELBO maximization problem with respect to q_l can be solved by solving n independent subproblems (one subproblem for each row i), when (g_l, q_f, τ) are fixed.

These facts motivate the strategy to solve the n subproblems in parallel. Particularly, we show that each subproblem can be seen as s regression problem (Section 1.2.1) and provide the solution to the regression problem (Section 1.2.2). The algorithm to compute the solution for q_l is formally introduced (Section 1.2.3), with a discussion on the potential joint update of the posterior q_l and the prior g_l (Section 1.2.4).

1.2.1 Interpreting a subproblem as a regression problem

The i-th subproblem Using the observation that $q_l(\mathbf{L})$ factorizes into $q_l(\mathbf{L}) = \prod_k \prod_i q_{l,i,k}(L_{i,k})$, the ELBO maximization problem with respect to q_l (1.6) can be written coordinate-wise as

$$\begin{aligned}\max_{\{q_{l,i,k}\}_{i\in[n],k\in[K]}} \sum_i \Bigg(\tau \sum_k \left(\mathbb{E}[\mathbf{f}_k]^T \mathbf{X}_{i,:}^T \mathbb{E}[L_{i,k}] - \frac{1}{2}\mathbb{E}[\mathbf{f}_k^T\mathbf{f}_k]\mathbb{E}[L_{i,k}^2] \right) \\ - \tau \sum_{1\le j<k\le K} \mathbb{E}[\mathbf{f}_k]^T\mathbb{E}[\mathbf{f}_j]\mathbb{E}[L_{i,j}]\mathbb{E}[L_{i,k}] + \sum_k \mathbb{E}\left[\log \frac{g_{l,k}(L_{i,k})}{q_{l,i,k}(L_{i,k})}\right] \Bigg)\end{aligned} \tag{1.9}$$

and the i-th subproblem (the ELBO maximization problem with respect to $\{q_{l,i,k}\}_{k\in[K]}$) as

$$\begin{aligned}\max_{\{q_{l,i,k}\}_{k\in[K]}} \Bigg(\tau \mathbf{X}_{i,:}\mathbb{E}[\mathbf{F}] \begin{bmatrix} \mathbb{E}[L_{i,1}] \\ \dots \\ \mathbb{E}[L_{i,K}] \end{bmatrix} \\ - \frac{\tau}{2}\left(\sum_k \mathbb{E}[\mathbf{f}_k^T\mathbf{f}_k]\mathbb{E}[L_{i,k}^2] + 2\sum_{1\le j<k\le K} \mathbb{E}[\mathbf{f}_k]^T\mathbb{E}[\mathbf{f}_j]\mathbb{E}[L_{i,j}]\mathbb{E}[L_{i,k}] \right) + \sum_k \mathbb{E}\left[\log \frac{g_{l,k}(L_{i,k})}{q_{l,i,k}(L_{i,k})}\right] \Bigg).\end{aligned} \tag{1.10}$$

An equivalent regression problem Consider the following variational Bayes regression problem:

$$\mathbf{y} = \mathbf{Z}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{1.11}$$

where $\mathbf{y} \in \mathbb{R}^p$ is the dependent variable, $\mathbf{Z} \in \mathbb{R}^{p\times K}$ is the independent variable, $\boldsymbol{\beta} \in \mathbb{R}^K$ is the unknown parameter with a prior $g(\boldsymbol{\beta}) = \prod_k g_k(\beta_k)$, the errors $\boldsymbol{\epsilon} \in \mathbb{R}^p$ are distributed as $\epsilon_i \sim^{\text{iid}} N(0, 1/\tau)$, and the variational approximation of the form $q(\boldsymbol{\beta}) = \prod_k q_k(\beta_k)$ is applied.

The ELBO maximization problem for the variational Bayes regression model (1.11) can be written as finding the variational posteriors that are solution to the following problems.

$$\max_{\{q_k\}_{k\in[K]}} \left(\mathbb{E}_q \left[\tau \left(\mathbf{y}^T \mathbf{Z} \boldsymbol{\beta} - \frac{1}{2} \boldsymbol{\beta}^T \mathbf{Z}^T \mathbf{Z} \boldsymbol{\beta} \right) \right] + \mathbb{E}_q \left[\log \frac{g(\boldsymbol{\beta})}{q(\boldsymbol{\beta})} \right] \right) \tag{1.12}$$

$$= \max_{\{q_k\}_{k\in[K]}} \left(\tau \mathbf{y}^T \mathbf{Z} \begin{bmatrix} \mathbb{E}[\beta_1] \\ \dots \\ \mathbb{E}[\beta_K] \end{bmatrix} \right. \tag{1.13}$$

$$\left. - \frac{\tau}{2} \left(\sum_k \mathbf{z}_k^T \mathbf{z}_k \mathbb{E}[\beta_k^2] + 2 \sum_{1 \le j < k \le K} \mathbf{z}_k^T \mathbf{z}_j \mathbb{E}[\beta_j] \mathbb{E}[\beta_k] \right) + \sum_k \mathbb{E} \left[\log \frac{g_k(\beta_k)}{q_k(\beta_k)} \right] \right).$$

Proposition 1.1. *The i-th subproblem is almost equivalent to a regression problem in the sense that if we define $\left(\mathbf{y}, \mathbf{Z}, \{g_k\}_{k=1}^K, \tau\right)$ as $\left(\mathbf{X}_{i,\cdot}^T, \mathbb{E}[\mathbf{F}], \{g_{l,k}\}_{k=1}^K, \tau\right)$ and interpret $\boldsymbol{\beta}$ as $[L_{i,1}, \dots, L_{i,K}]^T$, the two resulting optimization problems* (1.10) *and* (1.13) *are almost identical. If we replace the term $\mathbf{z}_k^T \mathbf{z}_k$ in* (1.13) *with $\mathbb{E}[\mathbf{f}_k^T \mathbf{f}_k]$ instead of $\mathbb{E}[\mathbf{f}_k]^T \mathbb{E}[\mathbf{f}_k]$, the two problems are exactly identical.*

The subproblem is slightly different from the regression problem because $\mathbb{E}[\mathbf{F}^T \mathbf{F}] \neq \mathbb{E}[\mathbf{F}^T] \mathbb{E}[\mathbf{F}]$ and hence, their diagonal terms $\mathbb{E}[\mathbf{f}_k^T \mathbf{f}_k] \neq \mathbb{E}[\mathbf{f}_k]^T \mathbb{E}[\mathbf{f}_k]$. This difference comes from taking expectation with respect to q_f when computing ELBO.

1.2.2 solREG: coordinate descent algorithm for the regression problem

Replacing $(\mathbf{Z}, \mathbf{Z}^T \mathbf{Z})$ with $(\mathbb{E}[\mathbf{F}], \mathbb{E}[\mathbf{F}^T \mathbf{F}])$ in the regression problem (1.12), we get the following optimization problem that is equivalent to the i-th subproblem (1.10)).

$$\max_{q_\beta} \left(\mathbb{E}_{q_\beta} \left[\tau \left(\mathbf{y}^T \mathbb{E}[\mathbf{F}] \boldsymbol{\beta} - \frac{1}{2} \boldsymbol{\beta}^T \mathbb{E}[\mathbf{F}^T \mathbf{F}] \boldsymbol{\beta} \right) \right] + \mathbb{E}_{q_\beta} \left[\log \frac{g(\boldsymbol{\beta})}{q_\beta(\boldsymbol{\beta})} \right] \right). \tag{1.14}$$

Variational Bayes regression problems have been widely studied, including the recent work by Kim et al. [2022]. Compared to the regression problem in Kim et al. [2022], our regression problem (1.14) puts a different prior g_k on each coordinate of $\boldsymbol{\beta}$ and the priors are not necessarily the "adaptive shrinkage" prior in Stephens [2017].

Following Kim et al. [2022], we take the coordinate descent approach in which we iteratively solve for each q_k until a convergence criterion is satisfied.

Definition 1.1. *We define a univariate distribution-valued function solC as*

$$\begin{aligned} solC\colon &(a, \mathbf{u}, \mathbf{v}, c, t, g) \\ &\to \frac{g(x)\exp\left(t\left(a - \mathbf{u}^T\mathbf{v}\right)x - \frac{t}{2}cx^2\right)}{\int g(x')\exp\left(t\left(a - \mathbf{u}^T\mathbf{v}\right)x' - \frac{t}{2}cx'^2\right)dx'}. \end{aligned} \tag{1.15}$$

Then, solC($a = \mathbf{y}^T\mathbb{E}[\mathbf{F}_{\cdot,k}], \mathbf{u} = \mathbb{E}[\mathbf{F}^T\mathbf{F}]_{-k,k}, \mathbf{v} = \mathbf{b}_{-k}, c = \mathbb{E}[\mathbf{F}^T\mathbf{F}]_{k,k}, t = \tau, g = g_k$) returns the coordinate-wise solution for $\mathbf{q}_k$ where $\mathbf{b}_{-k}$ denotes the posterior mean of $\boldsymbol{\beta}$ except for the β_k.

Example 1.1. *If g is a centered Gaussian prior $N(\cdot; 0, 1/\tau)$ for some error precision $\tau > 0$, the function solC has a closed-form solution as*

$$solC(a, \mathbf{u}, \mathbf{v}, c, t, g) = N\left(\cdot; \frac{a - \mathbf{u}^T\mathbf{v}}{\tau/t + c}, \frac{1}{\tau + tc}\right). \tag{1.16}$$

For a broader class of prior families, such as point-Gaussian distributions and point-Laplace distributions, we can utilize the empirical Bayes normal means (EBNM) solver developed by Willwerscheid and Stephens [2021]. The EBNM problem is estimating a common prior g and a posterior of the signal $\boldsymbol{\theta}$ when we observe a signal that is drawn iid from a prior, with an additive Gausian noise: $\theta_i \sim^{\text{iid}} g$ and $x_i|\theta_i \sim N(\cdot; \theta_i, s_i^2)$. In solC, we have a fixed prior g_k, thus we do not need to estimate the prior and the EBNM solver is used to denoise the Gaussian noise from the observation.

With a fixed prior g, the posterior $q(\theta_i)$ is proportional to $g(\theta_i)\exp(-\frac{\theta_i^2 - 2x_i\theta_i}{2s_i^2})$. By comparing this with the solC formulation (1.15), we get the following correspondence.

Definition 1.2. *We define EBNM* : $(g\_init, fix\_g, \mathbf{x}, \mathbf{s}) \to (g, q)$ *as an EBNM solver that takes an initial value g_init for a prior, a Boolean value fix_g of whether to fix or estimate the prior, a vector of observations* $\mathbf{x}$*, and a vector of standard errors* $\mathbf{s}$*; and returns the estimated prior* g *and posterior* q*.*

Proposition 1.2. *The distribution from solC*$(a, \mathbf{u}, \mathbf{v}, c, t, g)$ *is identical to the posterior from*

$$EBNM\left(g\_init = g, fix\_g = TRUE, x = \frac{a - \mathbf{u}^T\mathbf{v}}{c}, s = \frac{1}{\sqrt{tc}}\right). \tag{1.17}$$

Remark 1.1. *The posterior mean of the signal from the EBNM solution,* $\mathbb{E}_q[\theta_k]$*, can be interpreted as applying a shrinkage operator to the observed value* $x_k = (a - \mathbf{u}^T\mathbf{v})/c = (\mathbf{y}^T\mathbb{E}[\mathbf{F}_{:,k}] - \sum_{j\neq k}\mathbb{E}[\mathbf{f}_j^T\mathbf{f}_k]b_j)/(\mathbb{E}[\mathbf{f}_k^T\mathbf{f}_k])$*. Discussions on the shrinkage operator and the connection between variational empirical Bayes regression and the penalized linear regression are available in Kim et al. [2022].*

Running the function solC (returning the coordinate-wise solution for a single coordinate q_k) as a subroutine, we define the algorithm solREG (Algorithm 1.1) that returns a coordinate descent solution for the regression problem (1.14) when $(\mathbf{X}, \mathbf{W}, t) = (\mathbb{E}[\mathbf{F}], \mathbb{E}[\mathbf{F}^T\mathbf{F}], \tau)$. In the algorithm, M1 (or M2) denotes a function that returns the mean (or the second moment) of a probability distribution given as an argument (i.e., M1$(p) := E_{p(x)}[X]$ and M2$(p) := E_{p(x)}[X^2]$). The iterative updates of the coordinate descent involve only the first moments $\{\mathbb{E}_{q_k}[\beta_k]\}_{k=1}^K$. Therefore, we only need to compute and store the first moments, not the whole distribution $q_{\beta,k}$ until convergence, only after which we compute and store the second moments $\{\mathbb{E}_{q_k}[\beta_k^2]\}_{k=1}^K$.

The algorithm solREG may have a closed-form solution or not, depending on the priors $\{g_k\}_{k=1}^K$. When all the priors are centered Gaussian distributions, a closed-form solution for solREG is given as follows.

Algorithm 1.1 solREG: Coordinate descent algorithm for the regression problem (1.14)

```
Require: $\mathbf{y}, \mathbf{X}, \mathbf{W}, \mathbf{b}, t, \mathbf{g}$.
  $K \leftarrow \text{length}(\mathbf{g})$
  $\mathbf{a} \leftarrow \mathbf{X}^T\mathbf{y}$                ▷ Compute $\mathbf{X}^T\mathbf{y}$ once and reuse it for $a_k = \mathbf{y}^T\mathbf{x}_k$
  repeat
    for $k$ in $1, \dots, K$ do
      $q_k \leftarrow \text{solC}(a = a_k, \mathbf{u} = \mathbf{W}_{-k,k}, \mathbf{v} = \mathbf{b}_{-k}, c = W_{k,k}, t = t, g = g_k)$
      $b_k \leftarrow \text{M1}(q_k)$
    end for
  until convergence criterion satisfied
  $\mathbf{b2} \leftarrow \text{M2}(q_1, \dots, 1_K)$
  return $(\mathbf{b}, \mathbf{b2})$
```

Example 1.2. *If all the priors $\{g_k\}_{k=1}^K$ are centered Gaussians $N(\cdot; 0, 1/\tau_k)$ (τ_k's do not have to be identical), the function solREG has a closed-form solution as*

$$\textit{solREG}(\mathbf{y}, \mathbf{X}, \mathbf{W}, \mathbf{b}, t, \mathbf{g}) = \left(\tilde{\mathbf{b}}, \tilde{\mathbf{b}}^{\odot 2} + \left(\frac{1}{\tau_1 + tW_{1,1}}, \dots, \frac{1}{\tau_K + tW_{K,K}}\right)\right)$$

where the posterior mean $\tilde{\mathbf{b}}$ is defined as $\tilde{\mathbf{b}} := \left(\mathbf{W} + \frac{1}{t}\textit{diag}(\tau_1, \tau_2, \dots, \tau_K)\right)^{-1}\mathbf{X}^T\mathbf{y}$ and $\odot$ denotes the Hadamard power, i.e., $[\mathbf{b}^{\odot 2}]_i = b_i^2$.

Remark 1.2. *If the priors $\{g_{l,k}\}_{k=1}^K$ are all equal, then computing the first moments can be interpreted as applying a same shrinkage operator across all K indices from the EBNM perspective. Hence, if $\mathbb{E}[\mathbf{f}_k^T\mathbf{f}_k] = 1$ for $k \in [K]$ additionally holds, then the posterior mean-computing part of the coordinate descent algorithm (solREG; Algorithm 1.1) simplifies to the Algorithm 3 of Kim et al. [2022]. Note that we do not put such constraints on priors or $\mathbb{E}[\mathbf{F}^T\mathbf{F}]$.*

Remark 1.3. *To solve the i-th subproblem* (1.10), $\left(\mathbb{E}[\mathbf{F}]^T\mathbf{X}_{i,\cdot}^T, \mathbb{E}[\mathbf{F}^T\mathbf{F}], \{g_{l,k}\}_{k=1}^K, \tau\right)$ *is the sufficient statistics where $\mathbb{E}[\mathbf{F}]^T\mathbf{X}_{i,\cdot}^T$ is a vector of length K and $\mathbb{E}[\mathbf{F}^T\mathbf{F}]$ is a matrix of size $K \times K$. The communication cost and memory requirement for parallelization are expected to be small.*

Remark 1.4. *The same matrix $\mathbb{E}[\mathbf{F}^T\mathbf{F}]$ are used for all n subproblems. We can speed up the computation by computing $\mathbb{E}[\mathbf{F}^T\mathbf{F}]$ once and using it when solving n subproblems in parallel. (The matrix $\mathbb{E}[\mathbf{F}]$ is the first moment of q_f, so it should have been computed and stored already.) This would be particularly helpful since the size of factors (K) would be much smaller compared to n or p; $\mathbb{E}[\mathbf{F}^T\mathbf{F}] \in \mathbb{R}^{K\times K}$ is a small matrix. Once we have computed $\mathbb{E}[\mathbf{F}^T\mathbf{F}]$, each of the n subproblem could be easily parallelized.*

1.2.3 solQ: coordinate descent algorithm for updating q_l

In the previous section, we introduced the algorithm solREG that solves one regression problem using coordinate descent. It is straightforward to apply solREG to each of the n independent regression problem to update q_l. solQ, the coordinate descengt algorithm for updating q_l, is formally stated in Algorithm 1.2.

Algorithm 1.2 solQ: Coordinate descent algorithm for updating q_l

```
Require: $\mathbf{X}, \mathbb{E}[\mathbf{L}], \mathbb{E}[\mathbf{F}], \mathbb{E}[\mathbf{F}^T\mathbf{F}], \tau, \mathbf{g}_l$.
  $n \leftarrow \text{nrow}(\mathbf{X})$
  $K \leftarrow \text{length}(\mathbf{g}_l)$
  for $i$ in $1, \dots, n$ do
      $(\mathbb{E}[\mathbf{L}_{i,\cdot}], \mathbb{E}[\mathbf{L}_{i,\cdot}^{\odot 2}]) \leftarrow \text{solREG}(\mathbf{y} = \mathbf{X}_{i,\cdot}^T, \mathbf{X} = \mathbb{E}[\mathbf{F}], \mathbf{W} = \mathbb{E}[\mathbf{F}^T\mathbf{F}], \mathbf{b} = \mathbb{E}[\mathbf{L}_{i,\cdot}^T],$
                                          $t = \tau, \mathbf{g} = \mathbf{g}_l)$
  end for
  $\mathbb{E}[\mathbf{L}^T\mathbf{L}] \leftarrow \mathbb{E}[\mathbf{L}^T]\mathbb{E}[\mathbf{L}]$; $\text{diag}(\mathbb{E}[\mathbf{L}^T\mathbf{L}]) \leftarrow \text{colSums}(\mathbb{E}[\mathbf{L}^{\odot 2}])$
  return $(\mathbb{E}[\mathbf{L}], \mathbb{E}[\mathbf{L}^T\mathbf{L}])$
```

Example 1.3. *If all the priors $\{g_{l,k}\}_{k=1}^K$ are centered Gaussians $N(\cdot; 0, 1/\tau_{l,k})$, the n regression problems in the function solQ has a closed-form solution as*

$$\mathbb{E}[\mathbf{L}] \leftarrow \mathbf{X}\mathbb{E}[\mathbf{F}]\left(\mathbb{E}[\mathbf{F}^T\mathbf{F}] + \frac{1}{\tau}\mathit{diag}(\tau_{l,1}, \tau_{l,2}, \dots, \tau_{l,K})\right)^{-1} \tag{1.18}$$

$$\mathbb{E}[\mathbf{L}^{\odot 2}] \leftarrow \mathbb{E}[\mathbf{L}]^{\odot 2} + \mathbf{1}\left(\frac{1}{\tau_{l,1} + \tau\mathbb{E}[\mathbf{f}_1^T\mathbf{f}_1]}, \frac{1}{\tau_{l,2} + \tau\mathbb{E}[\mathbf{f}_2^T\mathbf{f}_2]}, \dots, \frac{1}{\tau_{l,K} + \tau\mathbb{E}[\mathbf{f}_K^T\mathbf{f}_K]}\right) \tag{1.19}$$

where $\mathbf{1}$ *is a column vector of 1's.*

1.2.4 solQb: block coordinate descent algorithm for jointly updating q_l and g_l

Vectorization is a type of parallelization that accelerates computations by performing similar operations on multiple data elements simultaneously using vectorized instructions. The concept of vectorization can be applied to develop an alternative algorithms for updating q_l.

In the solQ algorithm, its subroutine solREG solves each regression problem by iterating the solC operations over the K coordinates; and we have shown that the solC operation can be replaced with the EBNM function (Proposition 1.2). From the perspective of the EBNM framework, solving the EBNM problems for the observations with a same prior is considered a set of similar problems, which can be vectorized. The EBNM function [Willwerscheid and Stephens, 2021], implemented in R, suports such verctorization: it takes a vector of observations $\mathbf{x}$ and a vector of standard errors $\mathbf{s}$ (or a scalar if they are identical) as arguments while taking a single prior as an argument.

This alternative algorithm, leveraging the vectorizated operations, is a block coordinate descent algorithm since it solves the q_l update by updating the posteriors in blocks where each block consists of the n factorized posteriors $\{q_{l,i,k}\}_{i=1}^n$ sharing a same index k. The algorithm, solQb, is shown in Algorithm 1.3.

In solQb, we are jointly updating the prior g_l and the posterior q_l, by fully solving the EBNM problem. For each EBNM solve, an optimal prior $g_{l,k}$ and optimal posteriors $\{q_{l,i,k}\}_{i=1}^n$ that maximize the ELBO are estimated; ELBO is monotonically increasing after each EBNM solve. This joint update of g_l and q_l is an additional benefit of taking the vectorized approach.

Algorithm 1.3 solQb: block coordinate descent algorithm for jointly updating (g_l, q_l)

```
Require: $\mathbf{X}, \mathbb{E}[\mathbf{L}], \mathbb{E}[\mathbf{F}], \mathbb{E}[\mathbf{F}^T\mathbf{F}], \tau, \mathbf{g}_l$.
  $n \leftarrow \text{nrow}(\mathbf{X})$
  $K \leftarrow \text{length}(\mathbf{g}_l)$
  repeat
      for $k$ in $1, \ldots, K$ do
          $(g_{l,k}, \{q_{l,i,k}\}_{i=1}^n) \leftarrow \text{EBNM}\Big(\text{g\_init} = g_{l,k}, \text{fix\_g=FALSE}, x = \frac{\mathbf{X}\mathbb{E}[\mathbf{f}_k] - \mathbb{E}[\mathbf{L}_{\cdot,-k}]\mathbb{E}[(\mathbf{F}^T\mathbf{F})_{-k,k}]}{\mathbb{E}[\mathbf{f}_k^T\mathbf{f}_k]}, s = \frac{1}{\sqrt{\tau\mathbb{E}[\mathbf{f}_k^T\mathbf{f}_k]}}\Big)$
          $(\mathbb{E}[\mathbf{L}_{\cdot,k}], \mathbb{E}[\mathbf{L}_{\cdot,k}^{\odot 2}]) \leftarrow \big(\text{M1}(\{q_{l,i,k}\}_{i=1}^n), \text{M2}(\{q_{l,i,k}\}_{i=1}^n)\big)$
      end for
  until convergence criterion satisfied
  $\mathbb{E}[\mathbf{L}^T\mathbf{L}] \leftarrow \mathbb{E}[\mathbf{L}^T]\mathbb{E}[\mathbf{L}]$; $\text{diag}(\mathbb{E}[\mathbf{L}^T\mathbf{L}]) \leftarrow \text{colSums}(\mathbb{E}[\mathbf{L}^{\odot 2}])$
  return $(\mathbf{g}_l, \mathbb{E}[\mathbf{L}], \mathbb{E}[\mathbf{L}^T\mathbf{L}])$
```

1.3 The altflash Algorithm

In the previous section, we developed two algorithms: solQ (Algorithm 1.2) and solQb (Algorithm 1.3). The solQ is a coordinate descent algorithm for the ELBO (1.5) maximization problem with respect to q_l, and the solQb is a block coordinate descent algorithm for the problem with respect to (q_l, g_l).

The ELBO is a function of $(q_l, q_f, g_l, g_f, \tau)$. The solQb algorithm (or the solQ algorithm) used to update (q_l, g_l) can also be used to update (q_f, g_f) with minimal notational changes, thanks to the symmetry in $\mathbf{L}$ and $\mathbf{F}$ in the EBMF model. The last piece of the algorithm is updating the error precision τ; the ELBO-maximizing update for τ is provided in the following Proposition.

Proposition 1.3. *The function solTAU defined as*

$$\textit{solTAU}: (\mathbf{X}, \mathbb{E}[\mathbf{L}], \mathbb{E}[\mathbf{L}^T\mathbf{L}], \mathbb{E}[\mathbf{F}], \mathbb{E}[\mathbf{F}^T\mathbf{F}]) \rightarrow \left(\frac{1}{np}\mathbb{E}\left[\|\mathbf{X} - \mathbf{L}\mathbf{F}^T\|_F^2\right]\right)^{-1} \tag{1.20}$$

returns the value of τ that maximizes the ELBO (1.5). *An efficient way to compute its value* *is shown in* (1.26).

For completeness, we state the altflash algorithm in Algorithm 1.4, using the solQb algorithm as a subroutine.[2]

Algorithm 1.4 altflash

```
Require: $\mathbf{X}, \mathbb{E}[\mathbf{L}], \mathbb{E}[\mathbf{L}^T\mathbf{L}], \mathbb{E}[\mathbf{F}], \mathbb{E}[\mathbf{F}^T\mathbf{F}], \tau, \mathbf{g}_l, \mathbf{g}_f$.
  repeat
    $(\mathbf{g}_l, \mathbb{E}[\mathbf{L}], \mathbb{E}[\mathbf{L}^T\mathbf{L}]) \leftarrow \text{solQb}(\mathbf{X}, \mathbb{E}[\mathbf{L}], \mathbb{E}[\mathbf{F}], \mathbb{E}[\mathbf{F}^T\mathbf{F}], \tau, \mathbf{g}_l)$
    $(\mathbf{g}_f, \mathbb{E}[\mathbf{F}], \mathbb{E}[\mathbf{F}^T\mathbf{F}]) \leftarrow \text{solQb}(\mathbf{X}^T, \mathbb{E}[\mathbf{F}], \mathbb{E}[\mathbf{L}], \mathbb{E}[\mathbf{L}^T\mathbf{L}], \tau, \mathbf{g}_f)$
    $\tau \leftarrow \text{solTAU}(\mathbf{X}, \mathbb{E}[\mathbf{L}], \mathbb{E}[\mathbf{L}^T\mathbf{L}], \mathbb{E}[\mathbf{F}], \mathbb{E}[\mathbf{F}^T\mathbf{F}])$
  until convergence criterion satisfied
  return $(\mathbb{E}[\mathbf{L}], \mathbb{E}[\mathbf{L}^T\mathbf{L}], \mathbb{E}[\mathbf{F}], \mathbb{E}[\mathbf{F}^T\mathbf{F}], \tau, \mathbf{g}_l, \mathbf{g}_f)$
```

1.3.1 altflash: a denoising approach for matrix factorization

The altflash algorithm is an algorithm to fit the EBMF model by maximizing its ELBO (1.5). To better understand the altflash, we can compare altflash with existing matrix factorization methods, by comparing the ELBO-maximization problem with a likelihood maximization problem. If we fix the error precision parameter τ at a large value, the ELBO becomes dominated by the likelihood term. In other words, the maximum likelihood estimator can be seen as a special case of the ELBO-maximizing estimator.

As noted in Proposition 1.2, the coordinate-wise solution solC can be obtained using the EBNM solver with the observation value $x = (\mathbf{y}^T\mathbb{E}[\mathbf{f}_k] - \sum_{j \neq k} \mathbb{E}[\mathbf{f}_j^T\mathbf{f}_k]b_j)/\mathbb{E}[\mathbf{f}_k^T\mathbf{f}_k]$, the standard error $s = 1/\sqrt{\tau\mathbb{E}[\mathbf{f}_k^T\mathbf{f}_k]}$, and the prior g. When the error precision τ tends to infinity, the standard error of the EBNM problem tends to zero, meaning that the observed value is noiseless. Hence, the EBNM solver returns the posterior distribution as a point mass localized at the observed value x, as long as the value has a nonzero probability in the prior g.

2. When the solQ is used instead, we need to add a step that updates the priors $\mathbf{g}_l$ and $\mathbf{g}_f$.

Therefore, what the MLE counterpart of the altflash algorithm is depends on the support of the prior families $\{(\mathcal{G}_{l,k}, \mathcal{G}_{f,k})\}_{k=1}^K$ that the priors $\{(g_{l,k}, g_{f,k})\}_{k=1}^K$ belong to.

When $\{(\mathcal{G}_{l,k}, \mathcal{G}_{f,k})\}_{k=1}^K$ are supported on the real line: altflash as a denoising version of orthogonal iteration

When all the prior families $\{(\mathcal{G}_{l,k}, \mathcal{G}_{f,k})\}_{k=1}^K$ include a distribution that is supported on the real line and the error precision τ tends to infinity, the EBNM solver returns the posterior distribution as the point mass at the observed value $x = (\mathbf{y}^T\mathbb{E}[\mathbf{f}_k] - \sum_{j\neq k}\mathbb{E}[\mathbf{f}_j^T\mathbf{f}_k]b_j)/\mathbb{E}[\mathbf{f}_k^T\mathbf{f}_k]$. Then, the solQb function has a closed-form solution

$$\mathbf{L} \leftarrow \mathbf{X}\mathbf{F}(\mathbf{F}^T\mathbf{F})^{-1} \tag{1.21}$$

and, similarly, $\mathbf{F} \leftarrow \mathbf{X}^T\mathbf{L}(\mathbf{L}^T\mathbf{L})^{-1}$, where we drop the expectation notation $\mathbb{E}[\cdot]$ because the quantities $\mathbf{L}$ and $\mathbf{F}$ are now point estimates. Note that these updates can also be obtained from setting the precision parameter tend to infinity in Example 1.3 (the centered Gaussian prior family in the Example is an example of the class of prior families supported on the real line).

If we were to add an orthogonalization step after each solQb run, this algorithm is identical to the "orthogonal iteration" method [Wilkinson, 1965, Golub and Van Loan, 2013], used to obtain the truncated SVD which is the solution to the rank-K matrix approximation problem in the Frobenius norm and the spectral norm [Eckart and Young, 1936]. Therefore, the altflash method can be interpreted as applying a column-wise denoising instead of the orthogonalization where the precision paramter τ (capturing the global signal strength) and the column-wise priors $g_{l,k}$ or $g_{f,k}$ (capturing the column-wise signal strength) jointly determine the denoising operator applied to each column.

When $\{(\mathcal{G}_{l,k}, \mathcal{G}_{f,k})\}_{k=1}^K$ are supported on the nonnegative real numbers: altflash as a denoising version of accelerated HALS

When all the prior families $\{(\mathcal{G}_{l,k}, \mathcal{G}_{f,k})\}_{k=1}^K$ include a distribution that is supported on the nonnegative real line and do not include any distribution whose support is greater than the nonnegative real line and the error precision τ tends to infinity, the EBNM solver returns the posterior distribution as the point mass at the observed value $x = (\mathbf{y}^T \mathbb{E}[\mathbf{f}_k] - \sum_{j \neq k} \mathbb{E}[\mathbf{f}_j^T \mathbf{f}_k] b_j)/\mathbb{E}[\mathbf{f}_k^T \mathbf{f}_k]$ if the value is nonnegative, and as the point mass at 0 otherwise. Then, the each solQb update can be characterized as iterating

$$\mathbf{l}_k \leftarrow \max\left(0, \frac{\mathbf{X}\mathbf{f}_k - \sum_{j \neq k} \mathbf{l}_j \mathbf{f}_j^T \mathbf{f}_k}{\|\mathbf{f}_k\|^2}\right) \tag{1.22}$$

for each $k \in [K]$ until convergence; similarly, the solQb udpate for the $\mathbf{F}$ side is characterized as iterating $\mathbf{f}_k \leftarrow \max\left(0, \frac{\mathbf{X}^T \mathbf{l}_k - \sum_{j \neq k} \mathbf{f}_j \mathbf{l}_j^T \mathbf{l}_k}{\|\mathbf{l}_k\|^2}\right)$. This algorithm that iterates the $\mathbf{L}$ update and the $\mathbf{F}$ update is called the "accelerated HALS" method [Gillis and Glineur, 2012], used to find a solution in the nonnegative matrix factorization problems: $\max_{\mathbf{L} \in \mathbb{R}_+^{n \times K}, \mathbf{F} \in \mathbb{R}_+^{p \times K}} \|\mathbf{X} - \mathbf{L}\mathbf{F}^T\|_F^2$. The classical hierarchical alternating least squares (HALS; Cichocki et al. [2008]) is an algorithm that iterates a single pass of the $\mathbf{L}$ update and a single pass of the $\mathbf{F}$ update. HALS update can be obtained by declaring that the convergence criterion is always satisfied in the solQb (Algorithm 1.3). The flash backfit algorithm can also be interpreted as a variation of HALS: the algorithm iterates updates of $(\mathbf{l}_1, \mathbf{f}_1, \mathbf{l}_2, \mathbf{f}_2, \ldots, \mathbf{l}_K, \mathbf{f}_K)$ instead of $(\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_K, \mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_K)$. The accelerated HALS is known to outperform the classical HALS. In this setup, the altflash algorithm can be interpreted as iterating a denoising version of the accelerated HALS updates.

1.3.2 Initialization

The altflash algorithm requires $(\mathbb{E}[\mathbf{L}], \mathbb{E}[\mathbf{L}^T\mathbf{L}], \mathbb{E}[\mathbf{F}], \mathbb{E}[\mathbf{F}^T\mathbf{F}], \tau, g_l, g_f)$ as an input in addition to the data matrix $\mathbf{X}$. The initialization can be made using a point estimate of $\mathbf{L}$ and $\mathbf{F}$, based on which the other quantities can be computed accordingly. As the point estimate of $(\mathbf{L}, \mathbf{F})$, depending on the scientific context, a user could consider methods, such as the random initialization, (rotated versions of) truncated SVD, (accelerated) HALS, or the good old flash.

1.4 Results

Since altflash and flashier are two algorithms solving the same optimization problem, we can readily compare their performance using their objective function value (ELBO) and runtime. To ensure a fair comparison, we initialize both algorithms identically, using the initialization obtained by the flashier greedy algorithm.

We compare the methods on the Genotype Tissue Expression (GTEx) project data [Lonsdale et al., 2013], processed by Urbut et al. [2019]. The data contains z-scores corresponding to each SNP-tissue pair for 16,069 SNPs and 44 tissues ($N = 16,069$ and $P = 44$). This dataset was also used in the illustration of flashr [Wang and Stephens, 2021] and flashier [Willwerscheid, 2021].

We consider the following three prior family settings, in which we use a common prior family for all columns of $\mathbf{L}$ (or $\mathbf{F}$):

1. $(\mathcal{G}_l, \mathcal{G}_f)$= ('normal', 'normal')
2. $(\mathcal{G}_l, \mathcal{G}_f)$= ('point_normal', 'point_normal')
3. $(\mathcal{G}_l, \mathcal{G}_f)$=('point_normal', 'point_exponential')

For a detailed comparison of altflash and flashier, we consider both the 'sequential' version and the 'extrapolation' version of flashier backfit algorithm. The extrapolation version adopts the scheme from [Ang and Gillis, 2019] to achieve a speed-up. For details, see Section 2.2.2 of Willwerscheid [2021].

For altflash, we utilize the closed-form solution for Setting 1, the Normal-Normal case. For the other settings, we implement Algorithm 1.4 with solQb. In solQb, we can define how many passes of the **L** (or **F**) update are made in each iteration. Multiple passes of the update correspond to the 'acceleration' as in the accelerated HALS. We consider one to five passes per solQb iteration in the following analysis.

Code for the altflash implementation and subsequent analysis is available at `https://doi.org/10.5281/zenodo.11224831`.

1.4.1 Setting 1: Normal-Normal

When the prior family is the class of Gaussian distributions, altflash can utilize a closed-form solution, enabling faster convergence. Figure 1.1 shows the results. The Normal-Normal setting has a simple optimization landscape, allowing all three methods to converge to similar ELBO values with relatively short runtimes. Compared to the 'flashier:sequential' method, 'flashier:extrapolate' achieves a slight improvement in speed, while 'altflash:closedform' achieves a 5-fold speed-up.

1.4.2 Setting 2: PointNormal-PointNormal

The results for Setting 2 are shown in the left panel of Figure 1.2 and Figure 1.3. When the prior families are set as point-normal distributions, 'flashier:extrapolate' converges to the highest ELBO with the smallest runtime. The altflash results, with one to five passes per solQb iteration (labeled as 'altflash:nupdates=1' to 'altflash:nupdates=5'), converge to similar ELBO values but with varying runtimes. For this data and setting, multiple updates do not seem to improve either the objective value or runtime. 'altflash:nupdates=1' is the fastest, while 'altflash:nupdates=5' is the slowest among the five altflash variations. The

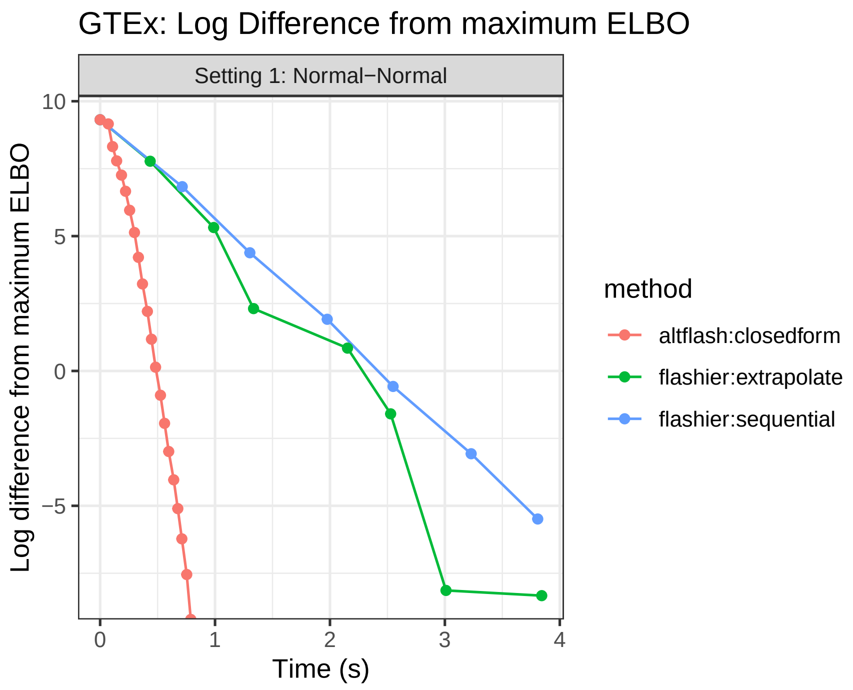


Figure 1.1: Comparison of ELBO between altflash and flashier, under the setting $(\mathcal{G}_l, \mathcal{G}_f)$ = ('normal', 'normal'). The y-axis represents the logarithmic difference from the maximum ELBO, while the x-axis corresponds to the respective time points

'flashier:sequential' result is comparable to 'altflash:nupdates=3' and 'altflash:nupdates=4'; thus, the single-pass version ('altflash:nupdates=1') provides an improvement over the baseline 'flashier:sequential'.

1.4.3 Setting 3: PointNormal-PointExponential

In this setting, the task changes fundamentally from the previous experiments because the estimated **L** is a mixed-sign matrix and the estimated **F** is a non-negative matrix, making the estimation a semi-nonnegative matrix factorization task. In this more complex optimization task, altflash methods outperform flashier methods. The results for Setting 3 are shown in the right panel of Figure 1.2 and Figure 1.3.

The 'flashier:extrapolate' algorithm terminates with the shortest runtime, but the resulting ELBO is significantly lower (approximately $-1,300$) compared to the other methods.

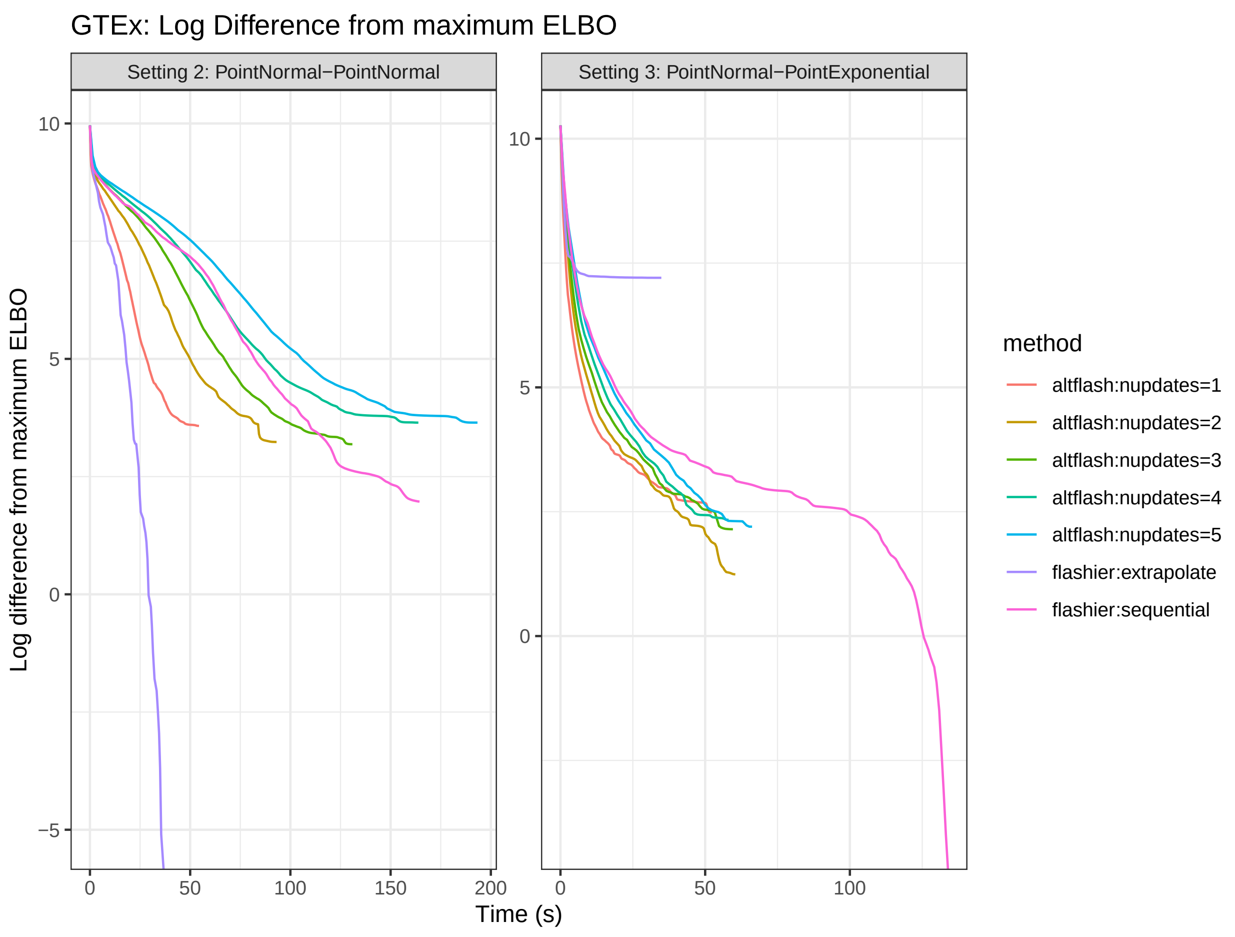


Figure 1.2: Comparison of ELBO between altflash and flashier, under the setting $(\mathcal{G}_l, \mathcal{G}_f)$ = (‘point_normal’, ‘point_normal’) and $(\mathcal{G}_l, \mathcal{G}_f)$ = (‘point_normal’, ‘point_exponential’). The y-axis represents the logarithmic difference from the maximum ELBO, while the x-axis corresponds to the respective time points.

Additionally, the algorithm does not perform the best even before it is terminated.

All five altflash results outperform ‘flashier:sequential’ in terms of both ELBO and runtime. Although ‘flashier:sequential’ achieves the highest ELBO, its ELBO is comparable to the altflash results but with a significantly longer runtime.

The runtimes across the five altflash results are more similar than in Setting 2. This convergence in runtime might be due to the fact that solving point-normal EBNM problems is easier and faster, thereby increasing the relative advantage of acceleration with multiple passes.

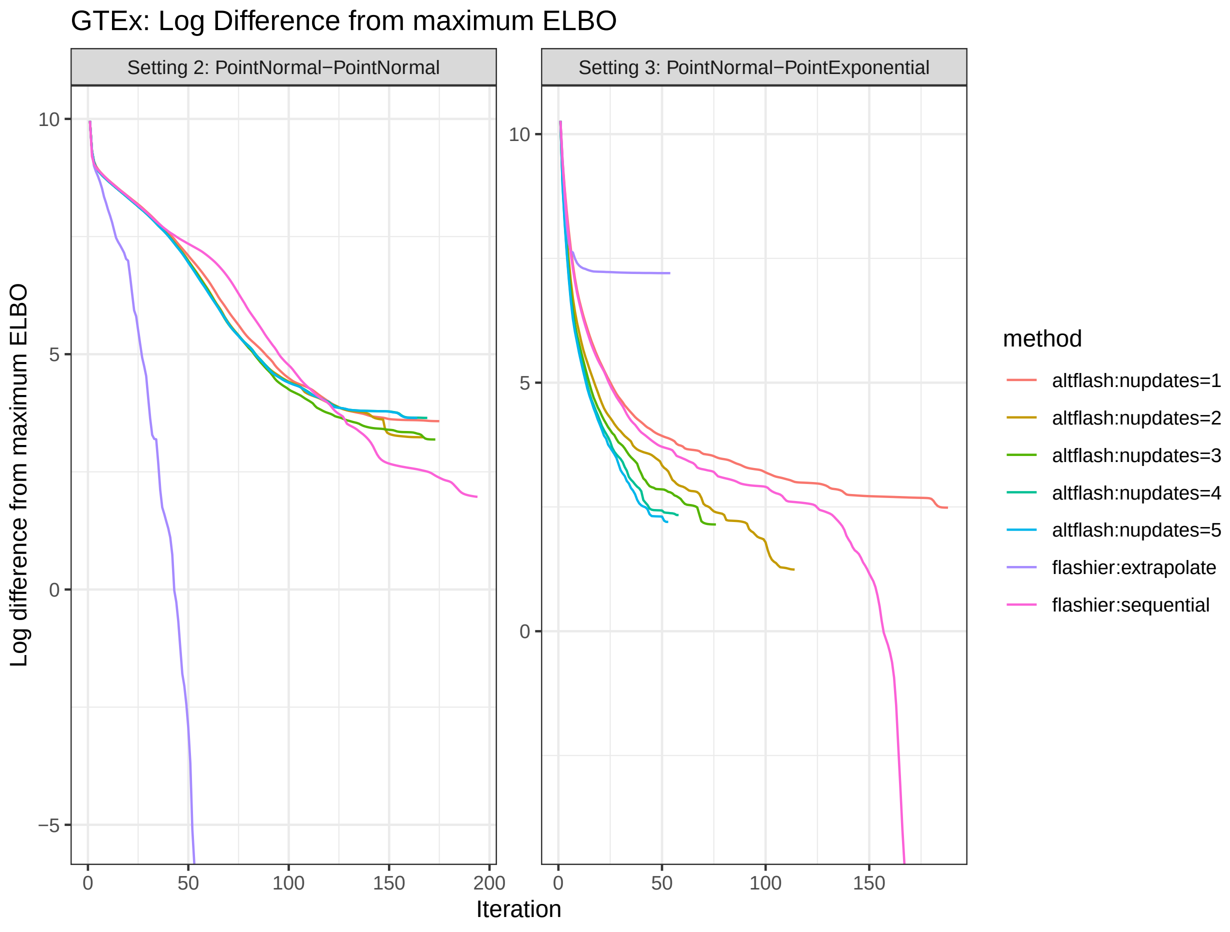


Figure 1.3: Comparison of ELBO between altflash and flashier, under the setting $(\mathcal{G}_l, \mathcal{G}_f)$ = ('point_normal', 'point_normal') and $(\mathcal{G}_l, \mathcal{G}_f)$ = ('point_normal', 'point_exponential'). The y-axis represents the logarithmic difference from the maximum ELBO, while the x-axis corresponds to the number of iterations.

1.5 Discussion

The results reported in the previous section demonstrate the promise of the altflash algorithm. First, as shown in Setting 1, we can fully utilize the advantage of altflash by using closed-form solutions for regression problems. Expanding implementations to include diverse closed-form solutions will be beneficial.

Second, altflash outperformed flashier in the semi-nonnegative task in Setting 3. This suggests that altflash may be better equipped to solve complex optimization problems, such as semi-nonnegative and non-negative matrix factorization problems. Further investigation into the performance of altflash compared to flashier in these settings is promising.

Third, the acceleration is expected to speed up the altflash algorithm for large datasets. The advantage of acceleration comes from computing $\mathbf{X}\mathbb{E}[\mathbf{F}]$ (or $\mathbf{X}^T\mathbb{E}[\mathbf{L}]$) once and reusing it in multiple passes of solQb updates. Studying the trade-off between the computational savings and the potential disadvantage of over-polishing parts of the optimization problem could yield valuable insights.

Lastly, the full potential of altflash could be realized with a coordinate descent-based solQ algorithm that can be fully parallelized, rather than the block coordinate descent-based solQb algorithm. In CPU multithreading, the benefit of parallelization can be overshadowed by its overhead cost, partly due to the relatively efficient vectorized implementation of the EBNM solver. For very large problems, a GPU implementation of the solQ algorithm could provide significant improvements over the flashier algorithm in estimating an EBMF model.

1.6 Appendix

1.6.1 Rearranging the core computation of the evidence lower bound

The log likelihood of the EBMF model is defined as

$$\log p(\mathbf{X}|\mathbf{L},\mathbf{F};\tau) = -\frac{np}{2}\log(2\pi) + \frac{np}{2}\log(\tau) - \frac{\tau\|\mathbf{X}-\mathbf{L}\mathbf{F}^T\|_F^2}{2} \tag{1.23}$$

where the expectation of the Frobenius norm squared can be rearranged as

$$\begin{aligned}\mathbb{E}\left[\|\mathbf{X}-\mathbf{L}\mathbf{F}^T\|_F^2\right] &= \mathbb{E}\left[tr(\mathbf{X}^T\mathbf{X}) - 2tr(\mathbf{X}^T\mathbf{L}\mathbf{F}^T) + tr(\mathbf{F}\mathbf{L}^T\mathbf{L}\mathbf{F}^T)\right] \qquad (1.24)\\ &= tr(\mathbf{X}^T\mathbf{X}) - 2\sum_k \mathbb{E}[\mathbf{l}_k]^T\mathbf{X}\mathbb{E}[\mathbf{f}_k] + \sum_k \mathbb{E}[\mathbf{f}_k^T\mathbf{f}_k]\mathbb{E}[\mathbf{l}_k^T\mathbf{l}_k] + 2\sum_{1\le j<k\le K}\mathbb{E}[\mathbf{f}_j]^T\mathbb{E}[\mathbf{f}_k]\mathbb{E}[\mathbf{l}_j]^T\mathbb{E}[\mathbf{l}_k].\end{aligned}$$

Therefore, the expectation $\mathbb{E}\left[\log p(\mathbf{X}|\mathbf{L},\mathbf{F};\tau)\right]$ can be expressed as

$$\begin{aligned}
&\mathbb{E}\left[\log p(\mathbf{X}|\mathbf{L},\mathbf{F};\tau)\right] && (1.25)\\
&= -\frac{np}{2}\log(2\pi) + \frac{np}{2}\log(\tau) - \frac{\tau}{2}tr(\mathbf{X}^T\mathbf{X}) + \tau\sum_k\left(\mathbb{E}[\mathbf{l}_k]^T\mathbf{X}\mathbb{E}[\mathbf{f}_k] - \frac{1}{2}\mathbb{E}[\mathbf{f}_k^T\mathbf{f}_k]\mathbb{E}[\mathbf{l}_k^T\mathbf{l}_k]\right)\\
&\quad -\tau\sum_{1\le j<k\le K}\mathbb{E}[\mathbf{f}_k]^T\mathbb{E}[\mathbf{f}_j]\mathbb{E}[\mathbf{l}_j]^T\mathbb{E}[\mathbf{l}_k].
\end{aligned}$$

Alternatively, an equivalent representation for vectorized operations is

$$\mathbb{E}\left[\|\mathbf{X}-\mathbf{L}\mathbf{F}^T\|_F^2\right] = tr(\mathbf{X}^T\mathbf{X}) - 2tr(\mathbf{X}^T\mathbb{E}[\mathbf{L}]\mathbb{E}[\mathbf{F}]^T) + \mathbf{1}^T(\mathbb{E}[\mathbf{L}^T\mathbf{L}]\odot\mathbb{E}[\mathbf{F}^T\mathbf{F}])\mathbf{1}, \quad (1.26)$$

$$\begin{aligned}
\mathbb{E}\left[\log p(\mathbf{X}|\mathbf{L},\mathbf{F};\tau)\right] = &-\frac{np}{2}\log(2\pi) + \frac{np}{2}\log(\tau) && (1.27)\\
&-\frac{\tau}{2}\left(tr(\mathbf{X}^T\mathbf{X}) - 2tr(\mathbf{X}^T\mathbb{E}[\mathbf{L}]\mathbb{E}[\mathbf{F}]^T) + \mathbf{1}^T(\mathbb{E}[\mathbf{L}^T\mathbf{L}]\odot\mathbb{E}[\mathbf{F}^T\mathbf{F}])\mathbf{1}\right).
\end{aligned}$$

where $\mathbf{1}\in\mathbb{R}^K$ is a column vector of 1's, and $\odot$ is the Hadamard product.

1.6.2 Adding a scaling matrix D: $\mathbf{X}\approx\mathbf{L}^T\mathbf{D}\mathbf{F}$

The solution $(g_l, g_f, q_l, q_f, \tau)$ to the ELBO maximization problem may not be unique due to scaling ambiguity. When the prior family $\mathcal{G}_l$ and $\mathcal{G}_f$ are closed under scaling, $(g_l, g_f, q_l, q_f, \tau)$ and $(cg_l, g_f/c, cq_l, q_f/c, \tau)$ would attain a same ELBO. One way to deal with the scaling ambiguity is to let the algorithm find one of the equivalent solutions. Another is to fix the scaling of either $\mathbf{L}$ or $\mathbf{F}$.

Alternatively, we could fix the scaling of both $\mathbf{L}$ and $\mathbf{F}$ and introduce another scaling parameters. That is, we could write $\mathbf{X} = \sum_k d_k\mathbf{l}_k\mathbf{f}_k^T + \mathbf{E} = \mathbf{L}\mathbf{D}\mathbf{F}^T + \mathbf{E}$ where $\mathbf{D} = \text{diag}(d_1,\ldots,d_K)$. Only a minor modification to the altflash algorithm is enough to add this functionality. When updating q_l (i.e., $\mathbb{E}[\mathbf{L}]$ and $\mathbb{E}[\mathbf{L}^T\mathbf{L}]$), we can think of $(\mathbf{D}\mathbf{F}^T)$ as our new $\mathbf{F}^T$, thus replacing $\mathbb{E}[\mathbf{F}]$ by $\mathbb{E}[\mathbf{F}]\mathbf{D}$ and $\mathbb{E}[\mathbf{F}^T\mathbf{F}]$ by $\mathbf{D}\mathbb{E}[\mathbf{F}^T\mathbf{F}]\mathbf{D}$; the update for

q_f can be done analogously. When updating τ, we split $\mathbf{LDF}^T$ as $(\mathbf{L}\sqrt{\mathbf{D}})(\mathbf{F}\sqrt{\mathbf{D}})^T$ where $\sqrt{\mathbf{D}} = \text{diag}(\sqrt{d_1}, \ldots, \sqrt{d_K})$.

And the last component is adding an update for $\mathbf{D}$. We put a uniform prior over $\mathbb{R}_+^K$ on $\text{diag}(d_1, \ldots, d_K)$, and take a point-estimate posterior on $\hat{\mathbf{D}}$. Using the results from ELBO computation (1.6.1), the ELBO-maximizing $\mathbf{d} = \text{diag}(\mathbf{D}) \in \mathbb{R}^K$ can be obtained by solving a quadratic programming:

$$\begin{aligned}
&\arg\min_{\mathbf{d}\geq\mathbf{0}} \left(-2\text{tr}(\mathbf{X}^T\mathbb{E}[\mathbf{L}]\mathbf{D}(\mathbb{E}[\mathbf{F}])^T) + \sum_{1\leq j\leq K, 1\leq k\leq K} (\mathbf{D}\mathbb{E}[\mathbf{L}^T\mathbf{L}]\mathbf{D})_{j,k}\mathbb{E}[\mathbf{f}_j^T\mathbf{f}_k] \right) \\
&= \arg\min_{\mathbf{d}\geq\mathbf{0}} \left(\frac{1}{2}\mathbf{d}^T(\mathbb{E}[\mathbf{L}^T\mathbf{L}] \odot \mathbb{E}[\mathbf{F}^T\mathbf{F}])\mathbf{d} - \mathbf{c}^T\mathbf{d} \right) \qquad (1.28)
\end{aligned}$$

where $\mathbf{c}^T = (\mathbb{E}[\mathbf{l}_1]^T\mathbf{X}\mathbb{E}[\mathbf{f}_1], \ldots, \mathbb{E}[\mathbf{l}_K]^T\mathbf{X}\mathbb{E}[\mathbf{f}_K]) = \text{diag}(\mathbb{E}[\mathbf{L}]^T\mathbf{XF})$. Introducing a function solD that returns a diagonal matrix with the diagonal entries from the solution to the quadratic programming (1.28), the complete altflash algorithm with a scaling matrix $\mathbf{D}$ can be written as follows.

Algorithm 1.5 altflash with a scaling matrix $\mathbf{D}$

```
Require: $\mathbf{X}, \mathbb{E}[\mathbf{L}], \mathbb{E}[\mathbf{L}^T\mathbf{L}], \mathbb{E}[\mathbf{F}], \mathbb{E}[\mathbf{F}^T\mathbf{F}], \mathbf{D}, \tau, \mathbf{g}_l, \mathbf{g}_f$.
  repeat
    $(\mathbf{g}_l, \mathbb{E}[\mathbf{L}], \mathbb{E}[\mathbf{L}^T\mathbf{L}]) \leftarrow \text{solQb}(\mathbf{X}, \mathbb{E}[\mathbf{L}], \mathbb{E}[\mathbf{F}]\mathbf{D}, \mathbf{D}\mathbb{E}[\mathbf{F}^T\mathbf{F}]\mathbf{D}, \tau, \mathbf{g}_l)$
    $(\mathbf{g}_f, \mathbb{E}[\mathbf{F}], \mathbb{E}[\mathbf{F}^T\mathbf{F}]) \leftarrow \text{solQb}(\mathbf{X}^T, \mathbb{E}[\mathbf{F}], \mathbb{E}[\mathbf{L}]\mathbf{D}, \mathbf{D}\mathbb{E}[\mathbf{L}^T\mathbf{L}]\mathbf{D}, \tau, \mathbf{g}_f)$
    $\mathbf{D} \leftarrow \text{solD}(\mathbf{X}, \mathbb{E}[\mathbf{L}], \mathbb{E}[\mathbf{L}^T\mathbf{L}], \mathbb{E}[\mathbf{F}], \mathbb{E}[\mathbf{F}^T\mathbf{F}])$
    $\tau \leftarrow \text{solTAU}(\mathbf{X}, \mathbb{E}[\mathbf{L}]\sqrt{\mathbf{D}}, \sqrt{\mathbf{D}}\mathbb{E}[\mathbf{L}^T\mathbf{L}]\sqrt{\mathbf{D}}, \mathbb{E}[\mathbf{F}]\sqrt{\mathbf{D}}, \sqrt{\mathbf{D}}\mathbb{E}[\mathbf{F}^T\mathbf{F}]\sqrt{\mathbf{D}})$
  until convergence criterion satisfied
  return $(\mathbb{E}[\mathbf{L}], \mathbb{E}[\mathbf{L}^T\mathbf{L}], \mathbb{E}[\mathbf{F}], \mathbb{E}[\mathbf{F}^T\mathbf{F}], \mathbf{D}, \tau, \mathbf{g}_l, \mathbf{g}_f)$
```

It is easy to see that this algorithm can be seen as a generalization of the previous algorithm in that when we fix $\mathbf{D} = I_K$ and do not update it, the algorithm is equal to the altflash algorithm in Algorithm 1.4.

CHAPTER 2

EMPIRICAL BAYES COVARIANCE DECOMPOSITION, AND A SOLUTION TO THE MULTIPLE TUNING PROBLEM IN SPARSE PCA

2.1 Introduction

Principal components analysis (PCA, Pearson, 1901) is a popular dimension reduction technique for revealing structure in data. However, when applied to large data sets, PCA results are often difficult to interpret. To address this, many authors have considered modifications of PCA that use sparsity, in some way, to help produce more interpretable results. Early versions of this idea arose in the literature on Factor analysis, where practitioners applied rotations to post-process results from PCA, or related techniques, to obtain sparse solutions; see Rohe and Zeng [2023] for interesting background and discussion. More recently, many authors have introduced “sparse PCA" (sPCA) methods that directly incorporate notions of sparsity into the inference problem [e.g. d’Aspremont et al., 2004, Zou et al., 2006, Witten et al., 2009, Journée et al., 2010, Ma, 2013].

While many different sPCA methods exist, they can generally be categorized into two types: “single-unit" methods that sequentially estimate one PC at a time, and “block" methods that estimate multiple PCs jointly. In single-unit methods, hyperparameter(s) that control the sparsity of each PC can be tuned via cross-validation (CV) as each PC is added. However, in contrast to standard PCA, sequentially estimating multiple sparse PCs is not equivalent to jointly estimating multiple sparse PCs, and can lead to sub-optimal results (Mackey, 2008). Block sPCA methods therefore have the potential, in principle, to produce better results, but using CV to simultaneously tune separate hyperparameters for multiple PCs presents a severe computational challenge. We call this the “*Multiple Tuning Problem*” (MTP), and its importance was emphasized in Zou and Xue [2018] which notes

> A very important issue to be investigated further is automated SPCA (sparse PCA). By "automated" we mean that there is a principled but not overly complicated procedure to set these sparse parameters in SPCA. This question is particularly challenging when we solve several sparse principal components jointly.

The MTP means that, in practice, block methods require users to specify hyperparameter values as model input, rather than tuning them. This may help explain why the potential of block sPCA methods in principle has not yet been realized in practice; for example, Journée et al. [2010] report that their single-unit sPCA method outperforms their block sPCA method in a simulation study (see their Table 5).

Here we present a novel block sPCA method that solves the MTP by leveraging the empirical Bayes (EB) framework. Within the EB framework, penalties come from priors, whose hyperparameters are learned from data. This approach, which seamlessly integrates hyperparameter tuning into the fitting algorithm, offers a compelling alternative to the "hyperparameters as inputs" approach.

The structure of this chapter is as follows. In Section 2.3 we introduce a simple and general (block) penalized PCA criterion, which includes some previous sPCA methods [Witten et al., 2009, Journée et al., 2010] as special cases. We present a simple (block) algorithm for optimizing this criterion when the penalty is fully specified. This algorithm is a natural extension of the "orthogonal iteration" method [Wilkinson, 1965] for regular PCA, and we highlight connections and differences with previous algorithms for sPCA. Section 2.4 shows that our penalized PCA criterion can also be interpreted as a 'penalized covariance decomposition' criterion, and that, as with regular PCA, our algorithms for penalized PCA can be applied directly to the covariance (or Gram) matrix $\mathbf{X}^T\mathbf{X}$, as well as to the original data matrix $\mathbf{X}$. Section 2.5 introduces empirical Bayes versions of these penalized problems, in which the penalties are determined by prior distributions that are estimated from the data by maximum likelihood rather than cross-validation. This provides a principled and efficient solution to the MTP in sPCA, and one that can be immediately extended to incorporate other structural assumptions (e.g. non-negative PCA). After briefly discussing some practical issues (Section 2.6) we show numerical results illustrating the effectiveness of our methods, using sparse point-Laplace priors, in Section 2.7. The chapter concludes with a discussion of generalizations beyond sparsity.

2.2 Notation

We use bold capital letters, $\boldsymbol{A}$, to denote matrices, bold lowercase letters, $\boldsymbol{a}$, to denote column vectors, and non-bold lowercase letters, a, to denote scalars. We use the convention that $\boldsymbol{a}_k$ is the kth column of the matrix $\boldsymbol{A}$, and $a_{i,k}$ is the (i,k)th element of $\boldsymbol{A}$. We let $\mathcal{M}(N,K)$ denote the set of N-by-K real matrices, and $\mathcal{S}(P,K) = \{\mathbf{M} \in \mathcal{M}(P,K) : \mathbf{M}^T\mathbf{M} = \mathbf{I}_K\}$ denote the set of P-by-K orthonormal matrices, i.e., the Stiefel manifold embedded in $\mathcal{M}(P,K)$. $\|\boldsymbol{A}\|_F$ denotes the Frobenius norm of the matrix $\boldsymbol{A}$, $\|\boldsymbol{A}\|_F = \sum_{i,k} a_{i,k}^2$, and $\|\boldsymbol{A}\|_*$ denotes the nuclear norm of $\mathbf{A}$, which is the sum of its singular values.

2.3 A Penalized PCA Criterion

2.3.1 A Penalized PCA Criterion

There exist several different characterizations of PCA, which are equivalent, but lead to different sparse versions [Zou and Xue, 2018, Guerra-Urzola et al., 2021]. One characterization of PCA [Jolliffe, 2002, section 3.5] is that PCA finds the best rank-K approximation of a data matrix $\mathbf{X} \in \mathcal{M}(N, P)$ in the sense that it solves the following optimization problem[1]:

$$\min_{\substack{\mathbf{Z} \in \mathcal{S}(N,K), \\ \mathbf{L} \in \mathcal{M}(P,K)}} \frac{1}{2}\|\mathbf{X} - \mathbf{Z}\mathbf{L}^T\|_F^2 \quad \text{subject to } \mathbf{L}^T\mathbf{L} \text{ is diagonal.} \tag{2.1}$$

The matrices $\mathbf{Z}$ and $\mathbf{L}$ are sometimes called the component score and component loading matrices respectively.

Based on (2.1), we propose the following *penalized PCA criterion*, obtained by replacing the orthogonality restriction on $\mathbf{L}$ with a penalty term, which might for example encourage $\mathbf{L}$ to be sparse[2]:

$$\min_{\substack{\mathbf{Z} \in \mathcal{S}(N,K), \\ \mathbf{L} \in \mathcal{M}(P,K)}} h_{P,\boldsymbol{\lambda}}(\mathbf{L}, \mathbf{Z}; \mathbf{X}) := \left(\frac{1}{2}\|\mathbf{X} - \mathbf{Z}\mathbf{L}^T\|_F^2 + \sum_{k=1}^{K} P(\mathbf{l}_k; \lambda_k) \right) \tag{2.2}$$

where $P(\cdot; \lambda)$ is a penalty function with hyperparameter λ whose value determines the strength of the penalty.

The problem (2.2) has K hyperparameters, $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_K)$, and the value of λ_k (which may be a vector) determines the strength of the penalty on the kth column of $\mathbf{L}$. If we do not have a prior expectation of uniform sparsity or any other structural properties across all columns, we must specify K penalty hyperparameters, each corresponding to a specific column. The proper tuning of these hyperparameters in a K-dimensional space can pose computational challenges, which we refer to as the “multiple tuning problem" (MTP). A primary focus of our work is to develop automated methods for selecting these hyperparameters.

1. Typical PCA formulations assume that $\mathbf{Z}^T\mathbf{Z}$ is diagonal, and $\mathbf{L}^T\mathbf{L} = \mathbf{I}_K$, but we use the formulation in eq (2.1) because it leads to closer connections with existing sPCA formulations.

2. Although one could consider formulations in which $\mathbf{L}$ is both orthogonal and sparse, some previous authors have argued against it [Witten et al., 2009, Journée et al., 2010], and we follow their advice here.

2.3.2 Uniting Previous Sparse PCA Methods

Although (2.2) seems, to us, a natural way to formulate sPCA, most previous sPCA methods have not been explicitly framed as optimizing a criterion of this form; see Van Deun et al. [2011] for an exception. Nonetheless, several previous sPCA methods are either equivalent to, or closely-related to, solving (2.2) with some choice of penalty. In this subsection, we discuss some of these connections.

The sparse principal components (SPC) method of Witten et al. [2009] is a *single-unit* sPCA method that can be interpreted as a greedy algorithm for optimizing the L_1-penalized version of our penalized PCA criterion (i.e. with penalty $P(\mathbf{l}_k; \lambda_k) = \lambda_k \|\mathbf{l}_k\|_1$). Specifically, SPC (their Algorithm 2) starts by solving a rank-one version ($K = 1$) of (2.2) and then repeatedly solves the following problem:

$$\min_{\mathbf{z}_k, \mathbf{l}_k} \left(\frac{1}{2} \|\mathbf{R}_k - \mathbf{z}_k \mathbf{l}_k^T\|_F^2 + \lambda_k \|\mathbf{l}_k\|_1 \right) \quad \text{subject to } \|\mathbf{z}_k\| = 1, \mathbf{z}_k \perp \mathbf{z}_1, \ldots, \mathbf{z}_{k-1}, \tag{2.3}$$

where $\mathbf{R}_k = \mathbf{X} - \sum_{k'=1}^{k-1} \mathbf{z}_{k'} \mathbf{l}_{k'}^T$ for $k > 1$ is the residual matrix. (Witten et al. [2009] treat the orthonormality restriction on $\mathbf{Z}$ as optional, but here we treat it as an integral feature of our penalized PCA criterion; see Section 2.5 for discussion. Without the orthonormality restriction on $\mathbf{Z}$, a single-unit method similar to SPC was also proposed by Shen and Huang [2008].)

The generalized power (GPower) method of Journée et al. [2010] is a *block* sPCA method[3] that can be interpreted as solving a *restricted* version of our penalized PCA criterion with an Elastic Net penalty [Zou and Hastie, 2005], $P(\mathbf{l}_k; \boldsymbol{\lambda}_k) = \lambda_{k,1} \|\mathbf{l}_k\|_1 + \lambda_{k,2} \|\mathbf{l}_k\|_2^2$.

3. The GPower method introduced in Journée et al. [2010] includes both single-unit sPCA methods and block sPCA methods, but in this article we will only refer to their block sPCA method as GPower.

To make this precise, we rearrange the penalized criterion as

$$\max_{\{\mu_1,\ldots,\mu_K\}} \left(\max_{\substack{\mathbf{Z}:\mathbf{Z}^T\mathbf{Z}=\mathbf{I}_K, \\ \mathbf{L}:\|\mathbf{l}_k\|=\mu_k}} \left(\operatorname{tr}\left(\mathbf{X}^T\mathbf{Z}\mathbf{L}^T\right) - \sum_{k=1}^{K} \lambda_{k,1}\|\mathbf{l}_k\|_1 \right) - \sum_{k=1}^{K} \left(\frac{1}{2} + \lambda_{k,2}\right)\mu_k^2 \right), \tag{2.4}$$

and note that the GPower criterion coincides with the inner maximization (over $\mathbf{Z}$ and $\mathbf{L}$ under the restriction $\mathbf{Z}^T\mathbf{Z} = \mathbf{I}_K, \|\mathbf{l}_k\| = \mu_k$). In GPower the column-wise vector norms $\{\mu_1, \ldots, \mu_K\}$ are considered as hyperparameters that must be pre-specified, whereas our formulation suggests an alternative approach where $\lambda_{k,2}$ are pre-specified and the μ_k are maximized over.

Finally, the USLPCA method of Adachi and Trendafilov [2016] is closely related to (2.2) with L_0 penalty (i.e. $P(\mathbf{l}_k; \lambda) = \lambda\|\mathbf{l}_k\|_0$) and using the same hyperparameter λ for all columns, the difference being that they frame the problem using an L_0 constraint on $\mathbf{L}$ rather than a penalty.

2.3.3 BISPCA, a "block" algorithm for penalized PCA with separable penalties

A natural strategy for optimizing the penalized PCA criterion (2.2) is block coordinate descent: that is, alternate between minimizing $h_{P,\boldsymbol{\lambda}}(\mathbf{L}, \mathbf{Z}; \mathbf{X})$ over $\mathbf{Z}$ (with $\mathbf{L}$ fixed) and over $\mathbf{L}$ (with $\mathbf{Z}$ fixed). We now detail this general algorithm, which we call the *Block-Iterative-Shrinkage PCA* (BISPCA).

Optimizing over $\mathbf{Z}$: the Rotation Step

The optimization of $h_{P,\boldsymbol{\lambda}}(\mathbf{L}, \mathbf{Z}; \mathbf{X})$ over $\mathbf{Z}$ does not depend on the penalty, and so is the same as the unpenalized case. It has a well-known solution [e.g. Zou et al., 2006], which we summarize here.

Definition 2.1 (*U* factor of Polar decomposition)**.** *For* $\mathbf{M}$ *any real-valued matrix, with SVD* $\mathbf{M} = \mathbf{U}\mathbf{D}\mathbf{V}^T$, *define Polar.U*$(\mathbf{M}) := \mathbf{U}\mathbf{V}^T$. *[Note: Polar.U*$(\mathbf{M})$ *denotes the so-called "U factor" of the polar decomposition of* $\mathbf{M}$*.]*

Fact 2.1 (Reduced-rank Procrustes rotation problem)**.** *Given* $\mathbf{L}$*, the minimum*

$$\min_{\mathbf{Z}\in\mathcal{S}(N,K)} \|\mathbf{X} - \mathbf{Z}\mathbf{L}^T\|_F^2$$

is achieved by $\hat{\mathbf{Z}}(\mathbf{L}, \mathbf{X}) :=$ *Polar.U*$(\mathbf{X}\mathbf{L})$.

Optimizing over $\mathbf{L}$: the Shrinkage Step

Due to the orthogonality constraint on $\mathbf{Z}$, the part of the fidelity term in (2.2) that depends on $\mathbf{L}$ decomposes as a sum, with one term for each entry in $\mathbf{L}$:

$$\begin{aligned}
||\mathbf{X} - \mathbf{Z}\mathbf{L}^T||_F^2 &= \mathrm{tr}(\mathbf{X} - \mathbf{Z}\mathbf{L}^T)^T(\mathbf{X} - \mathbf{Z}\mathbf{L}^T) &(2.5)\\
&= \mathrm{tr}(\mathbf{X}^T\mathbf{X}) + \mathrm{tr}(\mathbf{L}\mathbf{L}^T) - 2\mathrm{tr}(\mathbf{X}^T\mathbf{Z}\mathbf{L}^T) &(2.6)\\
&= \mathrm{tr}(\mathbf{X}^T\mathbf{X}) + \sum_{p,k}[l_{p,k}^2 - 2(\mathbf{x}_p^T\mathbf{z}_k)l_{p,k}] &(2.7)\\
&= \sum_{p,k}[l_{p,k} - \mathbf{x}_p^T\mathbf{z}_k]^2 + \mathrm{const} &(2.8)
\end{aligned}$$

where the third line follows from the fact that if $\mathbf{A}$ and $\mathbf{B}$ are matrices of the same dimension then $\mathrm{tr}(\mathbf{A}\mathbf{B}^T) = \sum_{ij} a_{ij}b_{ij}$.

Thus, if the penalty term decomposes similarly, $\sum_k P(\mathbf{l}_k; \lambda_k) = \sum_{p,k} \rho(l_{p,k}; \lambda_k)$ for some 1-dimensional penalty function ρ, then the optimization over $\mathbf{L}$ splits into PK independent problems, and

$$\hat{l}_{p,k} = \arg\min_l \left(\frac{1}{2}(l - \theta_{p,k})^2 + \rho(l; \lambda_k)\right), \tag{2.9}$$

where $\theta_{p,k} := \mathbf{x}_p^T \mathbf{z}_k$. The solution to this problem, $S_\rho(\theta_{p,k}; \lambda_k)$, depends on the penalty function $\rho(\cdot; \lambda_k)$, and is referred to as the "proximal operator" of $\rho(\cdot; \lambda_k)$. It has a closed-form solution for some widely-used penalties. For the L_1 penalty, the solution is the "soft thresholding" operator $S_1(a; \lambda) := \text{sign}(a)(|a| - \lambda)_+$; and for the L_0 penalty, the solution is the "hard thresholding" operator $S_0(a; \lambda) := aI(|a| > \lambda)$. (Note: $I(b)$ is the indicator function, with value 1 if b is true and 0 otherwise; and $(x)_+ := xI(x > 0)$.) Parikh and Boyd [2014] give proximal operators for several other penalties.

2.3.4 Connections with other algorithms

Table 2.1 summarizes the BISPCA algorithm, as well as the sPCA algorithms from Witten et al. [2009], Journée et al. [2010], and Ma [2013]. Here we briefly discuss the connections and differences between these algorithms, as well as the connection with algorithms for standard PCA, which corresponds to the case where the penalty function is constant.

When the penalty function is an L_1 penalty, the proximal operator S is the soft shrinkage operator, and the BISPCA algorithm is closely connected with the SPC and GPower algorithms, which also alternate shrinkage and rotation steps. (The single-unit sPCA method SPC has an additional deflation step; see Table 2.1). Thus, in this special case BISPCA provides a non-greedy alternative to the greedy algorithm in Witten et al. [2009] (and non-greedy methods are generally preferred to greedy methods, because the latter are more prone to yield poor local optima). With L_1 penalty the BISPCA algorithm is also very similar to GPower, but omits a normalization step ($\mathbf{l}_k \leftarrow \mu_k \mathbf{l}_k / \|\mathbf{l}_k\|_2$), and hence avoids the need to specify the μ_k. (This simplification can be thought of as coming from replacing the elastic-net penalty with the L_1 penalty.)

When the penalty function is constant, the proximal operator S is the identity function, and the BISPCA updates for $\mathbf{Z}$ simplify to $\mathbf{Z} \leftarrow \text{Polar.U}(\mathbf{X}\mathbf{X}^\mathbf{T}\mathbf{Z})$. This is a simple variation on the standard "orthogonal iteration" method [Wilkinson, 1965, Golub and Van Loan, 2013]

Method	Shrinkage Step	Rotation Step	Deflation Step
BISPCA (this chapter)	$\mathbf{l}_k \leftarrow S_\rho(\mathbf{X}^T\mathbf{z}_k; \lambda_k)$	$\mathbf{Z} \leftarrow \text{Polar.U}(\mathbf{XL})$	NA
	[equivalently, $\mathbf{L} \leftarrow S_\rho(\mathbf{X}^T\text{Polar.U}(\mathbf{XL}); \boldsymbol{\lambda})$]		
SPC (Witten et al., 2009)	$\mathbf{l}_k \leftarrow S_1(\mathbf{R}_k^T\mathbf{z}_k; \lambda_k)$	$\begin{cases}\boldsymbol{\theta}_k \leftarrow \frac{\mathbf{Z}_{k-1}^{\perp T}\mathbf{R}_k\mathbf{l}_k}{\Vert\mathbf{Z}_{k-1}^{\perp T}\mathbf{R}_k\mathbf{l}_k\Vert_2} \\ \mathbf{z}_k \leftarrow \mathbf{Z}_{k-1}^{\perp}\boldsymbol{\theta}_k\end{cases}$	$\mathbf{R}_k = \mathbf{X} - \sum_{k'=1}^{k-1}\mathbf{Z}_{k'}\mathbf{l}_{k'}^T$
GPower (Journée et al., 2010)	$\begin{cases}\mathbf{l}_k \leftarrow S_1(\mathbf{X}^T\mathbf{z}_k; \lambda_{k,1}) \\ \mathbf{l}_k \leftarrow \mu_k\mathbf{l}_k/\Vert\mathbf{l}_k\Vert_2\end{cases}$	$\mathbf{Z} \leftarrow \text{Polar.U}(\mathbf{XL})$	NA
ITSPCA [Ma, 2013]	$\mathbf{L} \leftarrow \text{QR.Q}(S_\rho(\mathbf{X}^T\mathbf{XL}; \boldsymbol{\lambda}))$		NA
EBCD-MM (this chapter)	$\begin{cases}g_k \leftarrow G(\mathbf{X}^T\mathbf{z}_k, 1/\tau, \mathcal{G}) \\ \bar{\mathbf{l}}_k \leftarrow S(\mathbf{X}^T\mathbf{z}_k, 1/\tau, g_k)\end{cases}$	$\mathbf{Z} \leftarrow \text{Polar.U}(\mathbf{X}\bar{\mathbf{L}})$	NA

Table 2.1: Sparse PCA Algorithms. $S_\rho(\cdot; \lambda_k)$ denotes the proximal operator of the penalty function $\rho(\cdot; \lambda_k)$, and S_1 denotes the soft thresholding operator, which is the proximal operator of the L_1 penalty. We use $S_\rho(\mathbf{A}; \boldsymbol{\lambda})$ to denote the vector whose kth element is $S_\rho(\boldsymbol{a}_k; \lambda_k)$. The U factor of the polar decomposition is denoted as Polar.U, and the Q factor of the QR decomposition is denoted as QR.Q. $\mathbf{Z}^\perp$ represents an orthonormal basis that is orthogonal to $\mathbf{Z}$. The function G calculates the estimated prior from the empirical Bayes normal means model, and the function S returns the corresponding posterior mean vector (see Definition 2.2 and Remark 2.2).

for PCA, which iterates $\mathbf{Z} \leftarrow \text{QR.Q}(\mathbf{XX^TZ})$ where QR.Q denotes the orthogonal $\mathbf{Q}$ factor of the QR decomposition; BISPCA simply uses Polar.U as an alternative orthogonalization to QR.Q. Under mild conditions, under either of these iterates, the range of $\mathbf{Z}$ converges to the leading eigenspace of $\mathbf{XX^T}$. (The ranges of $\mathbf{XX^TZ}$, $\text{QR.Q}(\mathbf{XX^TZ})$ and $\text{Polar.U}(\mathbf{XX^TZ})$ are identical, but the orthogonalizations QR.Q or Polar.U are required for numerical stability[4].)

Finally, we contrast BISPCA with the iterative thresholding sparse PCA (ITSPCA) algorithm from Ma [2013]. Whereas ITSPCA iterates $\mathbf{L} \leftarrow \text{QR.Q}(S_\rho(\mathbf{X}^T\mathbf{XL}; \boldsymbol{\lambda}))$, BISPCA iterates $\mathbf{L} \leftarrow S_\rho(\mathbf{X}^T\text{Polar.U}(\mathbf{XL}); \boldsymbol{\lambda})$ where $S_\rho(\mathbf{M}; \boldsymbol{\lambda})$ denotes applying the proximal operator to each column of the matrix $\mathbf{M}$, that is, $S_\rho(\mathbf{M}; \boldsymbol{\lambda}) = [S_\rho(\mathbf{m}_1; \lambda_1), \ldots, S_\rho(\mathbf{m}_K; \lambda_K)]$.

4. Strictly speaking orthogonalization is not necessarily required; for example, treppen iteration [Bauer, 1957], which precedes orthogonal iteration, iterates $\mathbf{Z} \leftarrow \text{LU.L}(\mathbf{XX^TZ})$ where LU.L denotes the lower triangular $\mathbf{L}$ factor of the LU decomposition.

Written this way, the updates appear similar, but with a different order of the shrinkage and orthogonalization steps, and with different orthogonalization approaches (QR.Q vs Polar.U). A conceptual advantage of BISPCA is that it is designed to optimize a specific objective function (2.2); in contrast ITSPCA is simply an algorithmic modification of orthogonal iteration, and it is unclear what objective function (if any) the ITSPCA algorithm optimizes, and indeed, in general, it is unclear whether ITSPCA is guaranteed to converge. Furthermore, because ITSPCA enforces orthogonality after the shrinkage step, it is unclear that the final $\mathbf{L}$ will be sparse.

2.4 A Penalized Covariance Decomposition Criterion

The constraint $\mathbf{Z} \in \mathcal{S}(N, K)$ in our penalized PCA criterion (2.2) has the following important consequence: not only does $\mathbf{Z}\mathbf{L}^T$ approximate the data matrix $\mathbf{X}$, but also $\mathbf{L}\mathbf{L}^T$ approximates the Gram matrix $\mathbf{X}^T\mathbf{X}$ (which is proportional to the covariance matrix if $\mathbf{X}$ has centered columns). Intuitively, this is simply because $\mathbf{X}^T\mathbf{X} \approx \mathbf{L}\mathbf{Z}^T\mathbf{Z}\mathbf{L}^T = \mathbf{L}\mathbf{L}^T$. In this section we formalize this, and discuss the implications for both interpretation and computation. (Providing an approximation to both $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}$ could be seen as a fundamental characteristic of PCA that is not generally shared by other matrix factorization methods, and we prefer to reserve the term sparse PCA for methods that have this feature, using “Matrix Factorization" [Wang and Stephens, 2021] or “Matrix Decomposition" [Witten et al., 2009] for the more general class of methods that may not have this feature. However, not all methods previously labeled as PCA have this feature: e.g. the EB-PCA method of Zhong et al. [2022] does not include the orthogonality assumption, and in their sPCA method Witten et al. [2009] describe the orthogonality assumption as “optional".)

The key result is the following lemma, which we prove in Appendix 2.10.1:

Lemma 2.1. *Let* $\mathbf{X} \in \mathcal{M}(N, P)$ *and* K *be a positive integer with* $K \leq \min(N, P)$*. Then*

$$\min_{\mathbf{Z} \in \mathcal{S}(N,K)} \|\mathbf{X} - \mathbf{Z}\mathbf{L}^T\|_F^2 = d_*(\mathbf{X}^T\mathbf{X}, \mathbf{L}\mathbf{L}^T)^2 \tag{2.10}$$

where

$$d_*(\mathbf{A}, \mathbf{B}) := \left(tr(\mathbf{A}) - 2tr(\sqrt{\sqrt{\mathbf{A}}\mathbf{B}\sqrt{\mathbf{A}}}) + tr(\mathbf{B}) \right)^{1/2} \tag{2.11}$$

is the Bures-Wasserstein distance between matrices $\mathbf{A}$ *and* $\mathbf{B}$*, which is a metric on the space of positive semi-definite matrices [Bhatia et al., 2019].*

The following Theorem follows as a direct corollary of Lemma 2.1

Theorem 2.1. *Let* $\mathbf{X} \in \mathcal{M}(N, P)$ *and* K *be a positive integer less than or equal to* $\min(N, P)$*. Consider the penalized PCA criterion*

$$\min_{\substack{\mathbf{Z} \in \mathcal{S}(N,K), \\ \mathbf{L} \in \mathcal{M}(P,K)}} \left(\frac{1}{2}\|\mathbf{X} - \mathbf{Z}\mathbf{L}^T\|_F^2 + \sum_{k=1}^{K} P(\mathbf{l}_k; \lambda_k) \right) \tag{2.12}$$

where $P(\mathbf{l}_k; \lambda_k)$ *is an arbitrary penalty term on* $\mathbf{l}_k$ *with hyperparameter* λ_k*. Let* $(\hat{\mathbf{Z}}, \hat{\mathbf{L}})$ *denote a solution to* (2.12)*. Then* $\hat{\mathbf{L}}$ *also solves the following criterion:*

$$\hat{\mathbf{L}} \in \operatorname*{arg\,min}_{\mathbf{L} \in \mathcal{M}(P,K)} \left(\frac{1}{2} d_*(\mathbf{X}^T\mathbf{X}, \mathbf{L}\mathbf{L}^T)^2 + \sum_{k=1}^{K} P(\mathbf{l}_k; \lambda_k) \right) \tag{2.13}$$

where d_* *denotes the Bures-Wasserstein distance. Further, solving* (2.13) *and then setting* $\hat{\mathbf{Z}} = Polar.U(\mathbf{X}\hat{\mathbf{L}})$ *yields a solution to* (2.12)*.*

Note 2.1. *The distance* $d_*(\mathbf{A}, \mathbf{B})$ *is equal to the 2-Wasserstein distance between two Gaussian measures with common mean and covariance matrices* $\mathbf{A}$ *and* $\mathbf{B}$*. For a recent review on the Bures-Wasserstein distance, including the proof that* d_* *is a metric, see Bhatia et al. [2019].*

The objective function in (2.13) combines the penalty term with a fidelity term that measures the squared (Bures-Wasserstein) distance between the matrices $\mathbf{X}^T\mathbf{X}$ and $\mathbf{L}\mathbf{L}^T$. If the matrix $\mathbf{X}$ has centered columns then $(1/N)\mathbf{X}^T\mathbf{X}$ is the covariance matrix, in which case (2.13) can be interpreted as finding an (approximate) decomposition of the covariance matrix of the form $(1/N)\mathbf{X}^T\mathbf{X} \approx (1/N)\sum_k \mathbf{l}_k\mathbf{l}_k^T$ with a penalty term to regularize and/or sparsify the $\mathbf{l}_k$. For this reason we refer to (2.13) as the "penalized covariance decomposition" criterion. (If $\mathbf{X}$ does not have centered columns then $\mathbf{X}^T\mathbf{X}$ is the Gram matrix and (2.13) would be more properly referred to as a "penalized Gram matrix decomposition").

2.4.1 Sufficient Statistic and Efficient Computation

Theorem 2.1 implies that the Gram matrix $\mathbf{X}^T\mathbf{X}$ is sufficient to estimate $\mathbf{L}$. This suggests an alternative computational approach to computing $\mathbf{L}$. In brief, the idea is to first compute the solution $\hat{\mathbf{L}}$ using a *compact version of the data matrix* $\mathbf{C} \in \mathcal{M}(P,P)$ that satisfies $\mathbf{C}^T\mathbf{C} = \mathbf{X}^T\mathbf{X}$, and then use the original matrix $\mathbf{X}$ to compute the corresponding $\hat{\mathbf{Z}}$. The following theorem formalizes this approach.

Theorem 2.2. *Suppose that a data matrix* $\mathbf{X} \in \mathcal{M}(N,P)$ *has the thin singular value decomposition* $\mathbf{U}_X\mathbf{D}_X\mathbf{V}_X^T$ *with* $P < N$ *and* K *is a positive integer with* $K \leq P$. *Let* $\mathbf{C} \in \mathcal{M}(P,P)$ *satisfy* $\mathbf{C}^T\mathbf{C} = \mathbf{X}^T\mathbf{X}$ *(eg, one such matrix is* $\mathbf{C} = \mathbf{V}_X\mathbf{D}_X\mathbf{V}_X^T$*). The following four problems* *are equivalent:*

$$(a)\quad \hat{\mathbf{L}}, \hat{\mathbf{Z}} \in \underset{\substack{\mathbf{Z}\in\mathcal{S}(N,K),\\ \mathbf{L}\in\mathcal{M}(P,K)}}{\arg\min} \left(\frac{1}{2}\|\mathbf{X} - \mathbf{Z}\mathbf{L}^T\|_F^2 + \sum_{k=1}^{K} P(\mathbf{l}_k;\lambda_k) \right)$$

$$(b)\quad \hat{\mathbf{L}} \in \underset{\mathbf{L}\in\mathcal{M}(P,K)}{\arg\min} \left(\frac{1}{2} d_*(\mathbf{X}^T\mathbf{X}, \mathbf{L}\mathbf{L}^T)^2 + \sum_{k=1}^{K} P(\mathbf{l}_k;\lambda_k) \right) \quad \textit{and set } \hat{\mathbf{Z}} = \textit{Polar.}U(\mathbf{X}\hat{\mathbf{L}})$$

$$(c)\quad \hat{\mathbf{L}} \in \underset{\mathbf{L}\in\mathcal{M}(P,K)}{\arg\min} \left(\frac{1}{2} d_*(\mathbf{C}^T\mathbf{C}, \mathbf{L}\mathbf{L}^T)^2 + \sum_{k=1}^{K} P(\mathbf{l}_k;\lambda_k) \right) \quad \textit{and set } \hat{\tilde{\mathbf{Z}}} = \textit{Polar.}U(\mathbf{C}\hat{\mathbf{L}}), \hat{\mathbf{Z}} = \mathbf{U}_X\mathbf{V}_X^T\hat{\tilde{\mathbf{Z}}}$$

$$(d)\quad \hat{\mathbf{L}}, \hat{\tilde{\mathbf{Z}}} \in \underset{\substack{\tilde{\mathbf{Z}}\in\mathcal{S}(P,K),\\ \mathbf{L}\in\mathcal{M}(P,K)}}{\arg\min} \left(\frac{1}{2}\|\mathbf{C} - \tilde{\mathbf{Z}}\mathbf{L}^T\|_F^2 + \sum_{k=1}^{K} P(\mathbf{l}_k;\lambda_k) \right) \quad \textit{and set } \hat{\mathbf{Z}} = \mathbf{U}_X\mathbf{V}_X^T\hat{\tilde{\mathbf{Z}}}.$$

where $\sum_{k=1}^{K} P(\mathbf{l}_k;\lambda_k)$ *is an arbitrary penalty term on* $\mathbf{l}_k$ *with parameter* λ_k*.*

Proof. The equivalence of (a) and (b) follows from Theorem 2.1. (b) and (c) are equivalent because $\mathbf{X}^T\mathbf{X} = \mathbf{C}^T\mathbf{C}$ and $\text{Polar.U}(\mathbf{QUDV}^T) = \mathbf{QUV}^T = \mathbf{Q}\text{Polar.U}(\mathbf{UDV}^T)$ for any $\mathbf{Q}$ that satisfies $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$. And the equivalence of (c) and (d) again follows from Theorem 2.1. □

From Theorem 2.2, any penalized PCA criterion (a) can be reformulated in the form (d), in which the target matrix $\mathbf{X} \in \mathcal{M}(N, P)$ is replaced by a compact version $\mathbf{C} \in \mathcal{M}(P, P)$. This can then be solved by applying the BISPCA algorithm to $\mathbf{C}$. If the component score matrix $\mathbf{Z}$ is not a parameter of interest then no additional step is needed; otherwise, $\mathbf{Z}$ can be easily recovered using the singular vectors of $\mathbf{X}$. If $P \ll N$ then this approach may be computationally more efficient than directly applying the BISPCA algorithm to $\mathbf{X}$.

The sufficiency of the Gram matrix, and the potential to exploit this for efficient computation, has been previously stated in the context of specific sPCA models, for example in Journée et al. [2010]. Our contribution is to provide a general result that applies to any penalty function, which enables applications to not-so-straightforward problems (e.g. EBCD in Section 2.5).

2.5 EBCD: An Empirical Bayes Solution to the Multiple Tuning Problem

To summarize the previous sections: the penalized PCA criterion (2.2) provides a family of objective functions that unifies several existing sparse PCA methods, and the BISPCA algorithm provides a convenient general recipe for optimizing this objective. Further, the penalized PCA criterion also has an attractive interpretation in terms of a penalized covariance decomposition (2.13). However, an important problem remains: the choice of suitable penalty function, and particularly the problem of tuning hyper-parameters of the penalty, which we refer to as the “multiple tuning problem" (MTP). In this section we suggest an Empirical Bayes solution to the MTP, in which the penalty is determined by a prior distribution, and the “tuning" takes place by estimating the prior distribution from the data. This is accomplished by a simple modification of the iterative BISPCA algorithm.

2.5.1 The EBCD Model

Motivated by the criterion (2.2) we consider the following empirical Bayes (EB) model:

$$\mathbf{X} = \mathbf{Z}\mathbf{L}^T + \mathbf{E} \tag{2.14}$$

$$l_{p,k} \sim^{\text{indep}} g_k \in \mathcal{G} \tag{2.15}$$

$$e_{n,p} \sim^{\text{iid}} N(\cdot; 0, 1/\tau) \tag{2.16}$$

where $\mathbf{Z} \in \mathcal{S}(N, K)$, and $\mathbf{L}$ is independent of $\mathbf{E}$. We refer to this as an EB model because the column-wise prior distributions $\mathbf{g} := \{g_k\}_{k=1}^K$ are to be estimated from the data (subject to the constraint that they come from some prespecified prior family $\mathcal{G}$, which may be parametric or nonparametric). We use the notation $\mathbf{g}(\mathbf{L})$ to denote the prior on $\mathbf{L}$, $\mathbf{g}(\mathbf{L}) =$

$\prod_{p,k} g_k(l_{p,k})$.

The model (2.14)-(2.16) is closely related to the EBMF model of Wang and Stephens [2021], and the EB-PCA model of Zhong et al. [2022]. The key difference is that our model replaces a prior on $\mathbf{Z}$ with an orthonormality restriction on $\mathbf{Z}$. We will show that fitting this model is equivalent to optimizing a penalized criterion (2.2) with a penalty whose form is estimated from the data. Consequently, it is also equivalent to optimizing a penalized covariance decomposition criterion (2.13). This latter property distinguishes it from the EBMF model[5], and so we refer to the model (2.14)-(2.16) as the "Empirical Bayes Covariance Decomposition" (EBCD) model. (The model might also be reasonably referred to as the EB-PCA model, but the name EB-PCA was used by Zhong et al. [2022] for a different model, so we use EBCD to avoid confusion and to emphasize the covariance decomposition property.)

Fitting the EBCD model

A standard EB approach to fitting (2.14)-(2.16) would usually be phrased as a two-step procedure: i) estimate $(\hat{\mathbf{g}}, \hat{\mathbf{Z}}, \hat{\tau})$ by maximizing marginal log-likelihood

$$(\hat{\mathbf{g}}, \hat{\mathbf{Z}}, \hat{\tau}) := \underset{\mathbf{g},\mathbf{Z},\tau}{\arg\max} \log \int p(\mathbf{X}|\mathbf{Z}, \mathbf{L}, \tau) p(\mathbf{L}|\mathbf{g}) d\mathbf{L} \tag{2.17}$$

and ii) compute the conditional posterior for $\mathbf{L}$,

$$\hat{\mathbf{q}}(\mathbf{L}) := p(\mathbf{L}|\hat{\mathbf{g}}, \hat{\mathbf{Z}}, \hat{\tau}) \propto \hat{\mathbf{g}}(\mathbf{L}) p(\mathbf{X}|\hat{\mathbf{Z}}, \mathbf{L}, \hat{\tau}). \tag{2.18}$$

One might typically report the mean of $\hat{\mathbf{q}}$, $\hat{\mathbf{L}} := \mathbb{E}_{\hat{\mathbf{q}}}(\mathbf{L})$ as a point estimate for $\mathbf{L}$.

The two-step procedure (2.17)-(2.18) can be usefully rephrased as solving a single opti-

5. Willwerscheid [2021] considers fitting an EBMF model to a covariance matrix, which shows impressive results despite the inconsistency between the EBMF modeling assumption (a low-rank signal plus additive iid Gaussian errors) and the property of a covariance matrix (a symmetric positive semi-definite matrix).

mization problem (e.g. see Appendix B.1.1 in Wang et al. [2020]):

$$(\hat{\mathbf{g}}, \hat{\mathbf{Z}}, \hat{\tau}, \hat{\mathbf{q}}) = \underset{\mathbf{g}\in\mathcal{G},\mathbf{Z},\tau,\mathbf{q}}{\arg\max}\ F(\mathbf{g}, \mathbf{Z}, \tau, \mathbf{q}) \tag{2.19}$$

where

$$F(\mathbf{g}, \mathbf{Z}, \tau, \mathbf{q}) := \mathbb{E}_{\mathbf{q}} \log p(\mathbf{X}|\mathbf{Z}, \mathbf{L}, \tau) - \mathbb{KL}(\mathbf{q}||\mathbf{g}). \tag{2.20}$$

Here, $\mathbf{q}$ can be any distribution on $\mathbf{L}$, $\mathbb{E}_{\mathbf{q}}$ denotes expectation over $\mathbf{L}$ having distribution $\mathbf{q}$, and $\mathbb{KL}(\mathbf{q}||\mathbf{g}) = \mathbb{E}_{\mathbf{q}}[\log \frac{\mathbf{q(L)}}{\mathbf{g(L)}}]$ denotes the KL divergence from $\mathbf{g}$ to $\mathbf{q}$. The function F is often referred to as the "evidence lower bound". Note: this formulation of the EB approach is often introduced together with imposing an additional constraint on $\mathbf{q}$ to make computations easier, in which case optimizing F can be considered a "variational approximation" to the two-step procedure (2.17)-(2.18), sometimes referred to as "variational empirical Bayes" (VEB). Here we do not impose any additional constraint on $\mathbf{q}$, so optimizing F is equivalent to the two-step EB procedure (2.17)-(2.18); there is no variational approximation here.

Similarly to Wang and Stephens [2021], optimizing F over $\mathbf{g}, \mathbf{q}$ ends up requiring the solution to a simpler EB problem known as the "empirical Bayes normal means" problem. That is, one needs a function, EBNM, defined as follows.

Definition 2.2. *Let EBNM$(\mathbf{x}, s^2, \mathcal{G})$ denote a function that returns the EB solution to the following normal means model:*

$$x_p|\eta_p, s^2 \sim^{indep} N(x_p; \eta_p, s^2) \tag{2.21}$$

$$\eta_p \sim^{iid} g \in \mathcal{G}, \tag{2.22}$$

for $p = 1, \ldots, P$. More precisely,

$$\mathit{EBNM}(\mathbf{x}, s^2, \mathcal{G}) := \underset{g\in\mathcal{G},q}{\arg\max}\, \mathbb{E}_q \log p(x|\eta, s^2) - \mathbb{KL}(q||g) \tag{2.23}$$

where the optimization of q is over all possible distributions on $\eta = (\eta_1, \dots, \eta_P)$.

Efficient methods and software exist for solving the EBNM problem for a wide range of prior families $\mathcal{G}$; see Willwerscheid [2021] for example.

With the EBNM function in hand, F can be optimized as in the following Proposition (see Appendix 2.10.2 for proof).

Proposition 2.3. *Maximizing the evidence lower bound $F(\mathbf{g}, \mathbf{Z}, \tau, \mathbf{q})$ (2.20) subject to $\mathbf{Z}^T\mathbf{Z} = \mathbf{I}_K$ can be achieved by iteratively updating $(\mathbf{g}, \mathbf{q})$, updating $\mathbf{Z}$, and updating τ, as follows:*

$$\textit{EBNM step: for each } k \in [K], \quad (g_k, q_k) \leftarrow \textit{EBNM}(\mathbf{X}^T\mathbf{z}_k, 1/\tau, \mathcal{G}), \tag{2.24}$$

$$\textit{Rotation step: } \mathbf{Z} \leftarrow \textit{Polar.}U(\mathbf{X}\bar{\mathbf{L}}), \tag{2.25}$$

$$\textit{Precision step: } \tau \leftarrow NP/(\|\mathbf{X} - \mathbf{Z}\bar{\mathbf{L}}^T\|_F^2 + \|\mathbf{V}\|_{1,1}). \tag{2.26}$$

Here $\bar{\mathbf{L}} = \mathbb{E}_{\mathbf{q}}(\mathbf{L})$, $\mathbf{V}$ is the matrix with $v_{p,k} = Var_{q_k}(l_{p,k})$, and $\|\mathbf{V}\|_{1,1} = \sum_{p=1}^P \sum_{k=1}^K v_{p,k}$.

Remark 2.1. *The EBNM step above is similar to the EBNM step used to fit the EBMF model in Wang and Stephens [2021]. However, a key difference is that, due to the orthogonality of $\mathbf{Z}$, here the updates for $\mathbf{l}_1, \dots, \mathbf{l}_K$ separate into K independent updates. That is, whereas in EBMF the $\mathbf{l}_1, \dots, \mathbf{l}_K$ must be updated one at a time, in EBCD they can be updated jointly.*

Remark 2.2. *We can make the connection with BISPCA clearer by fixing τ, and separating the solution of the EBNM problem into a part that estimates g, and a part that computes the posterior mean of η for a given prior. That is, let $G(\mathbf{x}, s^2, \mathcal{G})$ denote the optimal prior returned by EBNM$(\mathbf{x}, s^2, \mathcal{G})$, let $S(\mathbf{x}, s^2, g) := \mathbb{E}(\boldsymbol{\eta}|\mathbf{x}, s^2, g)$ (i.e. S returns the posterior mean of $\boldsymbol{\eta}$ under the EBNM model with prior g). Then the updates* (2.24)-(2.25) *can be* *rewritten as*

$$g_k \leftarrow G(\mathbf{X}^T\mathbf{z}_k, 1/\tau, \mathcal{G}) \tag{2.27}$$

$$\bar{\mathbf{l}}_k \leftarrow S(\mathbf{X}^T\mathbf{z}_k, 1/\tau, g_k) \tag{2.28}$$

$$\mathbf{Z} \leftarrow Polar.U(\mathbf{X}\bar{\mathbf{L}}). \tag{2.29}$$

The resulting algorithm is shown in Table 2.1 along with other sPCA methods to highlight its algorithmic similarity to other sPCA methods. We call this algorithm EBCD-MM because it can be framed as a "minorization-maximization" (MM) algorithm to optimize the EBCD criterion, the minorization being given by F in (2.20)*.*

Remark 2.3. *Comparing EBCD-MM with BISPCA we see that in EBCD-MM $S(\mathbf{x}, s^2, g)$ plays the same role as the proximal operator in BISPCA. For certain classes of prior $\mathcal{G}$, including the point-Laplace prior we use later, S is a shrinkage operator, in that $|S(\mathbf{x}, s^2, g)| \leq |\mathbf{x}|$ holds point-wise for any $\mathbf{x} \in \mathbb{R}^P$, $s^2 > 0$, and $g \in \mathcal{G}$. The shape and strength of shrinkage applied to $\mathbf{X}^T\mathbf{z}_k$ depends on the column-wise prior distributions $\hat{g}_k$ and τ, which are estimated from the data. Estimating g_k, τ in EBCD is thus analogous to tuning the hyperparameters of the penalty function in penalized PCA, and in this way EBCD solves the multiple tuning problem.*

2.5.2 Connecting EBCD and the penalized PCA criterion

The similarity of the algorithms for EBCD and penalized PCA approaches suggests that the two approaches are closely linked. Here we formally establish this link. To do so we define

$$\tilde{F}(\mathbf{g}, \mathbf{Z}, \tau, \bar{\mathbf{L}}) := \max_{\mathbf{q}:\mathbb{E}_{\mathbf{q}}(\mathbf{L})=\bar{\mathbf{L}}} F(\mathbf{g}, \mathbf{Z}, \tau, \mathbf{q}). \tag{2.30}$$

From this definition and (2.19) it follows that

$$(\hat{\mathbf{g}}, \hat{\mathbf{Z}}, \hat{\tau}, \hat{\mathbf{L}}) = \arg\max_{\mathbf{g}, \mathbf{Z}, \tau, \bar{\mathbf{L}}} \tilde{F}(\mathbf{g}, \mathbf{Z}, \tau, \bar{\mathbf{L}}), \tag{2.31}$$

and so $\hat{\mathbf{L}} = \arg\max_{\bar{\mathbf{L}}} \tilde{F}(\hat{\mathbf{g}}, \hat{\mathbf{Z}}, \hat{\tau}, \bar{\mathbf{L}})$. The following proposition connects $\tilde{F}$ with the penalized PCA objective function (2.2).

Proposition 2.4.

$$\tilde{F}(\mathbf{g}, \mathbf{Z}, \tau, \bar{\mathbf{L}}) = -\left(\frac{1}{2}\|\mathbf{X} - \mathbf{Z}\bar{\mathbf{L}}^T\|_F^2 + \sum_{k,p} P_{\tau, g_k}(\bar{l}_{pk})\right)\tau \tag{2.32}$$

where the penalty terms are given by

$$P_{\tau,g}(\bar{l}) = \frac{N}{2K\tau}\log\frac{2\pi}{\tau} + \frac{1}{2}\min_{q:\mathbb{E}_q[l]=\bar{l}}\left(var_q(l) + \frac{2}{\tau}\mathbb{KL}(q||g)\right). \tag{2.33}$$

The proposition establishes that, for fixed $\mathbf{g}, \tau$, EBCD is a penalized PCA approach, with a penalty that depends on $\mathbf{g}, \tau$. Again, by estimating $\mathbf{g}, \tau$ EBCD automatically tunes the penalties, and so solves the multiple tuning problem.

Although the penalty $P_{\tau,g}$ does not, in general, have a closed form, it does have some convenient properties; for example, its proximal operator S is the posterior mean from a normal means problem, which has a closed form for many choices of prior g. See Kim et al. [2022] for some other relevant results.

2.5.3 Efficient Computation

Since EBCD is a penalized PCA method, the ideas from Section 2.4.1 apply, and the solution can be computed from the Gram matrix $\mathbf{X}^T\mathbf{X}$, or a compact version of the data matrix, $\mathbf{C}$. Indeed, suppose we fix $\mathbf{g}, \tau$ and let $\hat{\mathbf{L}}(\mathbf{X}; \mathbf{g}, \tau)$ denote the result of the EBCD algorithm when applied to data matrix $\mathbf{X}$ given fixed $\mathbf{g}, \tau$. Then Proposition 2.4, combined with Theorem 2.2, implies that $\hat{\mathbf{L}}(\mathbf{X}; \mathbf{g}, \tau) = \hat{\mathbf{L}}(\mathbf{C}; \mathbf{g}, \tau)$, where $\mathbf{C}$ is any matrix such that $\mathbf{X}^T\mathbf{X} = \mathbf{C}^T\mathbf{C}$.

It is straightforward to extend this result to the case where $\mathbf{g}, \tau$ are estimated. That is, one can maximize the ELBO F by iterating the steps (2.24)-(2.26) with $\mathbf{C}$ in place of $\mathbf{X}$, and then transforming $\mathbf{Z}$ as in Theorem 2.2(d). Note that step (2.26) requires knowledge of N (the number of rows of $\mathbf{X}$), in addition to $\mathbf{C}$, and that the resulting algorithm is different than if $\mathbf{C}$ were the actual data matrix (since $\mathbf{C}$ has P rows).

2.5.4 Extensions and variations

One slightly unnatural feature of the formulations presented thus far is that they place a penalty (or prior) on a parameter, $\mathbf{L}$, that is not a “population quantity", and whose interpretation changes with the number of samples N. For example, in Section 2.5 we saw that the fidelity term encourages $\mathbf{L}\mathbf{L}^T \approx \mathbf{X}^T\mathbf{X}$, whose magnitude grows with N; it would seem more natural to combine a penalty on $\mathbf{L}$ with a fidelity term that encourages $\mathbf{L}\mathbf{L}^T \approx (1/N)\mathbf{X}^T\mathbf{X}$ since the latter has a natural limit as $N \to \infty$ (with P fixed). This can be achieved simply by replacing the constraint $\mathbf{Z}^T\mathbf{Z} = \mathbf{I}_K$ with the scaled version $\mathbf{Z}^T\mathbf{Z} = N\mathbf{I}_K$, or equivalently $\mathbf{Z}/\sqrt{N} \in \mathcal{S}(N, K)$. All our results and algorithms are easily modified for this rescaled version; the details are given in Appendix 2.10.4.

It is also straightforward to extend our models to allow column-wise variances, although this loses the interpretation of the methods as a covariance decomposition. See Appendix 2.10.5 for details.

2.6 Practical Issues

2.6.1 Initialization

Both the penalized PCA criterion and the EBCD criterion are non-convex optimization problems. Consequently solutions may depend on initialization. One simple initialization strategy is to use a "greedy" algorithm, which iteratively adds columns to $\mathbf{L}$ and $\mathbf{Z}$, with each greedy step being initialized by a rank 1 (unpenalized) truncated SVD. To enforce that the newly added column $\mathbf{z}_k$ is orthogonal to the existing columns of $\mathbf{Z}_{k-1}$, we use the rotation step of SPC [Witten et al., 2009] (see Table 2.1) as our 'greedy rotation step':

$$\text{Greedy Rotation Step: } \mathbf{z}_k \leftarrow \text{greedyrotation}\left(\mathbf{Z}_{k-1}, \mathbf{R}_k, \mathbf{l}_k\right) := \sqrt{N}\frac{\mathbf{Z}^{\perp}_{k-1}\mathbf{Z}^{\perp T}_{k-1}\mathbf{R}_k\mathbf{l}_k}{\|\mathbf{Z}^{\perp T}_{k-1}\mathbf{R}_k\mathbf{l}_k\|_2}. \quad (2.34)$$

The initialization is complete after K columns have been added, at which point the criterion can be further optimized by applying EBCD-MM, a process referred to as "backfitting" in Wang and Stephens [2021]. For completeness we give the full procedure in Algorithm 2.1.

2.6.2 Choice of K

As noted in Wang and Stephens [2021], the EB approach provides a way to automatically select K. Provided the prior family $\mathcal{G}$ includes the distribution δ_0, a point mass at 0, then the EBCD criterion may be optimized with some $g_k = \delta_0$, and hence $\bar{\mathbf{l}}_k = 0$. Algorithmically, the greedy procedure in Algorithm 2.1 can be terminated the first time that $\bar{\mathbf{l}}_0 = 0$, providing an automatic way to stop adding factors. Alternatively the algorithm can, of course, be run with a user-specified choice of K.

Algorithm 2.1 EBCD-MM (greedy + backfit)

Require: data $\mathbf{X}$; maximum number of PCs $Kmax$; function svd1($\mathbf{A}$) $\to$ $(\mathbf{u}, d, \mathbf{v})$ that returns the leading singular vectors and singular value; function `ebnm`$(\mathbf{x}, s^2, \mathcal{G})$ $\to$ $(\mathbb{E}_{p^{\text{post}}}[\boldsymbol{\eta}], \text{var}_{p^{\text{post}}}(\boldsymbol{\eta}))$ that solves an empirical Bayes normal means problem and returns posterior mean and variance (see Definition 2.2 and Remark 2.2); function greedyrotation$(\mathbf{Z}, \mathbf{R}, \mathbf{l})$ $\to$ $\mathbf{z}$ that returns the updated column $\mathbf{z}_0$ that is orthogonal to the existing columns of $\mathbf{Z}$ (see (2.34)).

```
$\mathbf{Z} \leftarrow [\,]$; $\bar{\mathbf{L}} \leftarrow [\,]$; $\tau \leftarrow NP/\|\mathbf{X}\|_F^2$                ▷ Initialize ($\mathbf{Z}$, $\bar{\mathbf{L}}$, $\tau$)
for $r$ in $1, \ldots, Kmax$ do                      ▷ Greedily add components up to $Kmax$
    $\mathbf{R} \leftarrow \mathbf{X} - \mathbf{Z}\bar{\mathbf{L}}^T$
    $(\mathbf{u}, d, \mathbf{v}) \leftarrow \text{svd1}(\mathbf{R})$
    $\bar{\mathbf{l}}_0 \leftarrow d\mathbf{v}/\sqrt{N}$
    $\mathbf{z}_0 \leftarrow \text{greedyrotation}(\mathbf{Z}, \mathbf{R}, \bar{\mathbf{l}}_0)$
    repeat
        $(\bar{\mathbf{l}}_0, \mathbf{v}_0) \leftarrow \texttt{ebnm}(\mathbf{R}^T\mathbf{z}_0/N, 1/N\tau, \mathcal{G}_L)$          ▷ Shrinkage Step
        $\mathbf{z}_0 \leftarrow \text{greedyrotation}(\mathbf{Z}, \mathbf{R}, \bar{\mathbf{l}}_0)$           ▷ Greedy Rotation Step
        $\tau \leftarrow NP/(\|\mathbf{R} - \mathbf{z}_0\bar{\mathbf{l}}_0^T\|_F^2 + N\|\mathbf{v}_0\|_1)$        ▷ Precision Step
    until convergence criterion satisfied
    $\bar{\mathbf{L}} \leftarrow [\bar{\mathbf{L}}, \bar{\mathbf{l}}_0]$
    $\mathbf{Z} \leftarrow \sqrt{N}\text{Polar.U}(\mathbf{X}\bar{\mathbf{L}})$
end for
repeat                                                           ▷ Backfit
    for $k$ in $1, \ldots, Kmax$ do                            ▷ Shrinkage Step
        $(\bar{\mathbf{l}}_k, \mathbf{v}_k) \leftarrow \texttt{ebnm}(\mathbf{X}^T\mathbf{z}_k/N, 1/N\tau, \mathcal{G}_L)$
    end for
    $\mathbf{Z} \leftarrow \sqrt{N}\text{Polar.U}(\mathbf{X}\bar{\mathbf{L}})$                           ▷ Rotation Step
    $\tau \leftarrow NP/(\|\mathbf{X} - \mathbf{Z}\bar{\mathbf{L}}^T\|_F^2 + N\|\mathbf{V}\|_{1,1})$             ▷ Precision Step
until convergence criterion satisfied
return ($\mathbf{Z}$, $\bar{\mathbf{L}}$, $\mathbf{V}$, $\tau$)
```

2.6.3 Choice of prior

The specific form of the posterior mean shrinkage operator S in EBCD depends on the prior g, thus on the choice of the prior family $\mathcal{G}$. For *sparse* PCA one would choose a sparsity-inducing prior family; one could alternatively use non-negative prior families to induce non-negative PCA, or fully nonparametric prior families (as in Zhong et al. [2022]) for a more flexible regularized PCA, although we do not explore these options further here.

While several choices of sparse family are possible, here we use the "point Laplace" prior, a spike and slab prior with Laplace slab:

$$\mathcal{G} = \{g : g(x) = (1-\pi)\delta_0(x) + \pi \text{Laplace}(x; 0, b) \text{ for some } \pi \in [0,1], b > 0\} \tag{2.35}$$

where Laplace$(\cdot; \mu, b)$ denotes the probability density function of the Laplace distribution with a location parameter μ and a scale parameter b. Varying the prior parameters (π, b) of this prior allows for a wide range of possible shrinkage behaviors, as illustrated in Figure 2.1. We refer to EBCD with this specific prior as `EBCD-pl` ("empirical Bayes covariance decomposition with point Laplace prior family").

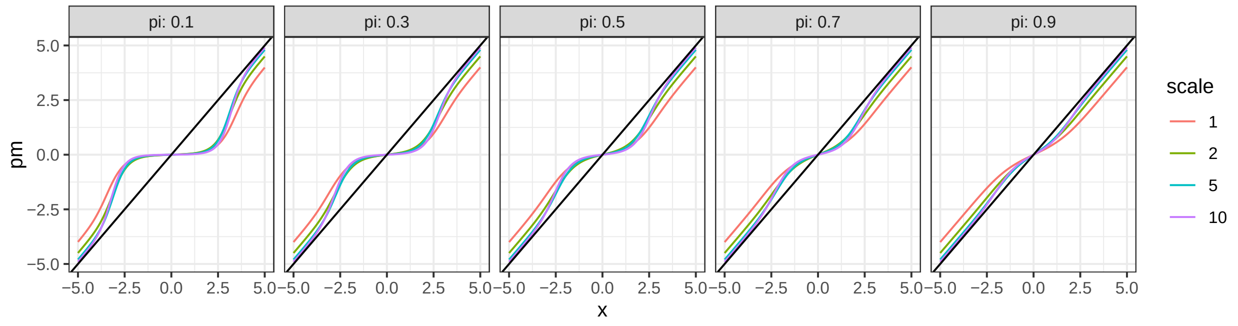


Figure 2.1: Examples of posterior mean shrinkage operator $S(\mathbf{x}, s^2 = 1, g = g(\cdot; \pi, b))$ induced by Laplace slab priors $g(x; \pi, b) = (1-\pi)\delta_0(x) + \pi\text{Laplace}(x; 0, b)$. Note how π controls shrinkage near 0 (small π yielding more shrinkage), while the scale parameter controls shrinkage further away from 0.

2.7 Empirical Results

2.7.1 Simulation

To illustrate the performance of `EBCD-pl`, we compare it with PCA, L_1-penalized PCA (our penalized PCA criterion (2.2) with an L_1 penalty), SPC, and GPower. To handle the multiple tuning problem, the block methods (L_1-penalized PCA and GPower) are used with an equality restriction, and the single-unit method (SPC) is used with a deflation scheme and a greedy hyperparameter optimization. We use the R package `PMA` for SPC, and the

MATLAB implementation available at `http://www.montefiore.ulg.ac.be/~journee/GPower.zip` for GPower. We only consider the GPower with an equality restriction because Journée et al. [2010] report that GPower with a random search for hyperparameter tuning is unsatisfactory under their simulation setting (which is the same as our Simulation 1).

For L_1-penalized PCA and SPC, we use cross-validation to choose the penalty parameter[6]; for GPower, which lacks a built-in cross-validation functionality, we report the best result (in terms of average d_{cov} measure, defined below). This approach evaluates the hyperparameter values using the true data generating process; performance using CV should be expected to be worse. Following Journée et al. [2010] we fix the GPower hyperparameters $(\mu_1, \mu_2) = (1, 0.5)$.

We also compare with empirical Bayes PCA (EB-PCA; Zhong et al., 2022). While EB-PCA shares the empirical Bayes part of EBCD, it does not assume and exploit sparsity of $\mathbf{L}$, and does not assume orthogonality of $\mathbf{Z}$. We use the Python implementation of EB-PCA available on `https://github.com/TraceyZhong/EBPCA`.

We consider two simulation settings, each with $\mathbf{x}_1, \ldots, \mathbf{x}_{50} \sim N_{500}(\mathbf{0}, \mathbf{\Sigma})$, where the 500×500 covariance matrix $\mathbf{\Sigma}$ is given by:

1)

$$\mathbf{\Sigma} = 399\mathbf{v}_1\mathbf{v}_1^T + 299\mathbf{v}_2\mathbf{v}_2^T + \mathbf{I}_{500} \tag{2.36}$$

where the PCs $\mathbf{v}_1, \mathbf{v}_2$ are given by

$$v_{1,j} = \mathbf{1}_{j\in[1,10]}/\sqrt{10}, \quad v_{2,j} = \mathbf{1}_{j\in[11,20]}/\sqrt{10}. \tag{2.37}$$

This setting comes from Shen and Huang [2008] and Journée et al. [2010].

6. The CV for the L_1-penalized PCA is based on the mean squared projection error. This error is calculated by projecting test data onto the subspace spanned by the columns of $\hat{\mathbf{L}}$ estimated using training data. The CV method is a multi-PC extension of the single-PC CV idea presented in Algorithm 2 by Shen and Huang [2008].

2)

$$\mathbf{\Sigma} = 9\mathbf{v}_1\mathbf{v}_1^T + 7\mathbf{v}_2\mathbf{v}_2^T + 4\mathbf{v}_3\mathbf{v}_3^T + \mathbf{I}_{500} \tag{2.38}$$

where

$$v_{1,j} = \mathbf{1}_{j\in[1,10]}/\sqrt{10}, \quad v_{2,j} = \mathbf{1}_{j\in[11,50]}/\sqrt{40}, \quad v_{3,j} = \mathbf{1}_{j\in[51,150]}/\sqrt{100}. \tag{2.39}$$

This setting illustrates the effect of non-equal sparsity level in the PCs.

For each setting we simulate 50 datasets and measure performance by three measures: i) the angle between the true PC and its estimate: for each PC i, the angle is defined as

$$d_i = \angle(\mathbf{v}_i, \hat{\mathbf{l}}_i)/\frac{\pi}{2} \tag{2.40}$$

where $\angle(\cdot,\cdot)$ denotes the angle between two vectors; ii) the difference between the population covariance matrix and the estimated $\frac{1}{N}\hat{\mathbf{L}}\hat{\mathbf{L}}^T$:

$$d_{\text{cov}} = \|\mathbf{\Sigma} - \frac{1}{N}\hat{\mathbf{L}}\hat{\mathbf{L}}^T\|_F; \tag{2.41}$$

iii) the distance with optimal rotation, which measures the proximity of two subspaces:

$$d_{\text{or}} = \min_{\mathbf{R}\in\mathcal{O}^{K\times K}} \|\tilde{\mathbf{L}}\mathbf{R} - \mathbf{V}\|_F \tag{2.42}$$

where $\mathcal{O}^{K\times K}$ is the set of K-by-K orthonormal matrices, $\mathbf{V}$ is $[\mathbf{v}_1, \mathbf{v}_2]$ in Simulation 1 and $[\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]$ in Simulation 2, and $\tilde{\mathbf{L}}$ is an orthonormal basis of the subspace spanned by estimated loading $\hat{\mathbf{L}}$. This measure can be shown to be nearly equivalent to other subspace proximity measures, such as the distance between projection matrices and the norm of principal angles (for example, see Chen et al., 2021).

The run-time for `EBCD-pl` was comparable in magnitude to that of other sPCA methods.In Simulation 1, the average run-times for each dataset were as follows: `EBCD-pl` took 2.62s, SPC 1.49s, GPower 0.11s, L1-penalized PCA 0.47s, EB-PCA 0.28s, and PCA 0.01s. For Simulation 2, the average run-times were `EBCD-pl` 2.19s, SPC 2.79s, L1-penalized PCA 1.62s, EB-PCA 0.64s, and PCA 0.01s. It is important to note that L1-penalized PCA and GPower with equality restriction were optimized over a one-dimensional hyperparameter grid, not over a two-dimensional or three-dimensional grid, which could increase the run-time substantially. Additionally, GPower was executed in MATLAB and EB-PCA in Python, making direct comparisons with the R-based methods challenging. All experiments were conducted on a 2020 MacBook Air with an Apple M1 chip and 16 GB RAM.

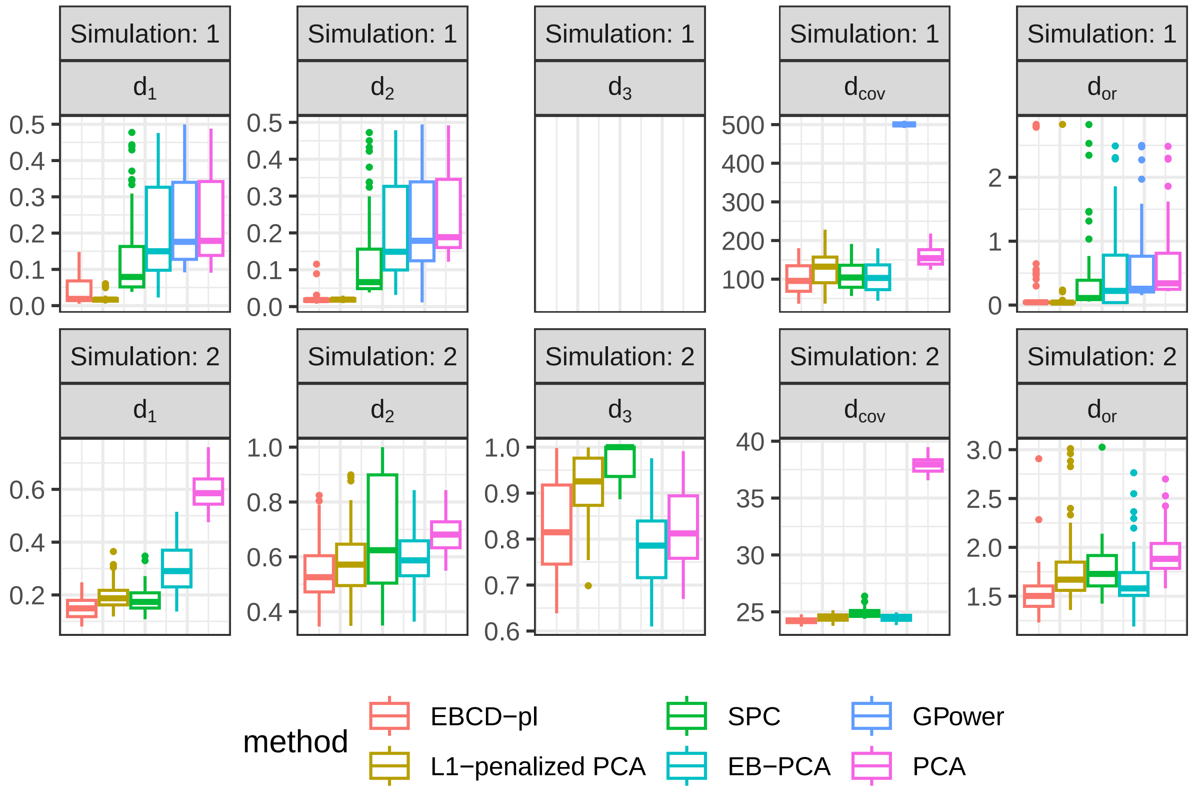


Figure 2.2: Simulation results comparing the performance of different methods in terms of three measures: angle between true and estimated principal components (PCs), difference between population covariance matrix and estimated covariance matrix, and distance with optimal rotation.

Results (Figure 2.2) show that `EBCD-pl` outperforms other methods, with L_1-penalized

PCA second. The benefits of `EBCD-pl` over L_1-penalized PCA are greatest in Simulation 2, where the sparsity levels in true PCs are different. However, even here the performance of L_1-penalized PCA is impressive despite the equality restriction on the penalty. The superiority of L_1-penalized PCA over SPC is presumably due in part to its use of a block optimization scheme, rather than simple deflation. Its superiority compared with the (block-based) GPower method may reflect difficulty in selecting the hyperparameter $\boldsymbol{\mu}$ in GPower. (Indeed, we excluded GPower results in Simulation 2 as we found it hard to specify this parameter.)

2.7.2 Stock Market Data

To illustrate our method's effectiveness in producing interpretable results, we applied `EBCD-pl` to stock market data.

In the article "America's best firms...and the rest: New winners and losers are emerging after three turbulent years", The Economist [Economist, 2022] reported on the stock market performance of S&P500 firms over an (almost) three-year period covering the COVID pandemic, January 1st, 2020, to November 29, 2022. The article examines the returns of firms subdivided into eleven Global Industry Classification Standard (GICS) sectors, both overall and separately for three phases, 'working from home' (January 1st, 2020, to November 8th, 2020, which is "the day before the test results of the Pfizer vaccine were announced"), 're-opening' (November 9th, 2020, to December 31st, 2021) and 'inflation' (January 1st, 2022 to November 29th, 2022).

We obtained data on sector-level daily returns covering the same time period from Refinitiv Datastream via Wharton Research Data Services; the data matrix contains daily log returns for 734 trading days and 11 sectors. Using these data we reproduced the main trends reported in the Economist article (Figure 2.3). Overall, during the three year period, energy and information technology sectors performed the best, and communication services the worst. However, dividing the period into three distinct phases highlights temporal variation in different sectors' performances. Indeed, during the 'working from home' phase the energy sector performed the worst, while information technology sector performed the best along with consumer discretionary and communication services. During 'reopening' the energy sector turned into the biggest winner, and all eleven sectors reported gains. In the 'inflation' phase only the energy sector reported gains.

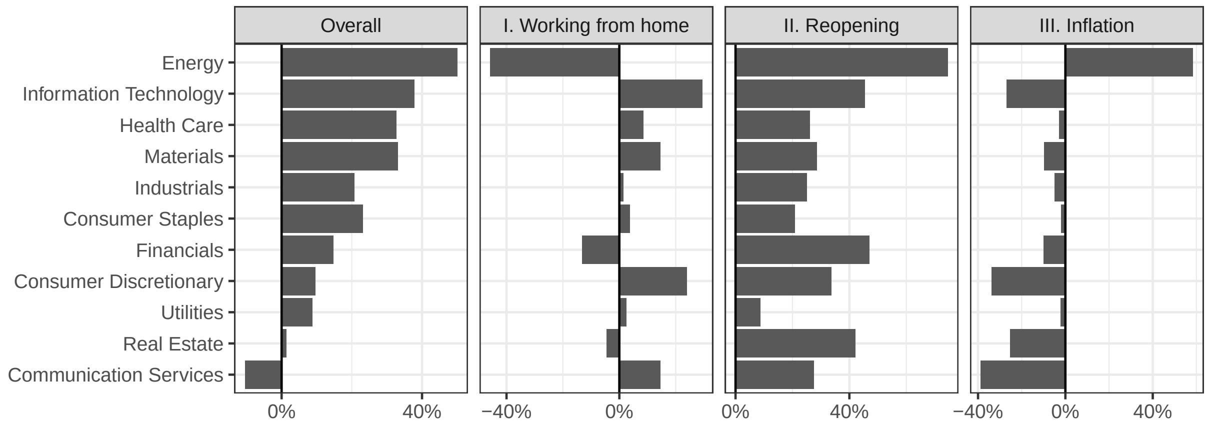


Figure 2.3: Holding period returns by sectors during three phases: "Working from home" (Jan 2020 - Nov 2020), "Reopening" (Nov 2020 - Dec 2021), and "Inflation" (Jan 2022 - Nov 2022).

We applied `EBCD-pl`, SPC and classical PCA to these data. The first three classical PCs explain 90.54% of total variance, with a sharp drop-off in signal after this point (the first five PCs explain 72.49%, 11.91%, 6.14%, 2.40%, and 1.66%) and so we focus comparisons on the first three PCs. The SPC result is almost identical to the PCA result (not shown). In contrast the three PCs estimated by `EBCD-pl` differ from classical PCA, both in their PVEs (66.99%, 16.35%, and 7.13%) and in the qualitative features of their loadings after the first PC (Figure 2.4). We attribute this difference in behavior between `EBCD-pl` and SPC as primarily due to the block vs single-unit behaviour. When signal is strong, greedily estimated sparse PCs may not deviate much from classical PCs. In contrast, the block optimization in the `EBCD-pl` algorithm (backfitting stage in Algorithm 2.1) allows `EBCD-pl` to move some of the explanatory power of the first PC to other PCs in order to increase sparsity. Interestingly this is done at almost no expense of total PVE explained by the first three PCs: cumulatively, the three `EBCD-pl` PCs explain 90.47% of the variation, very similar to the 90.54% of classical PCA. This highlights a benefit of block optimization methods for sparse PCA, compared with the widely-used single-unit and deflation schemes.

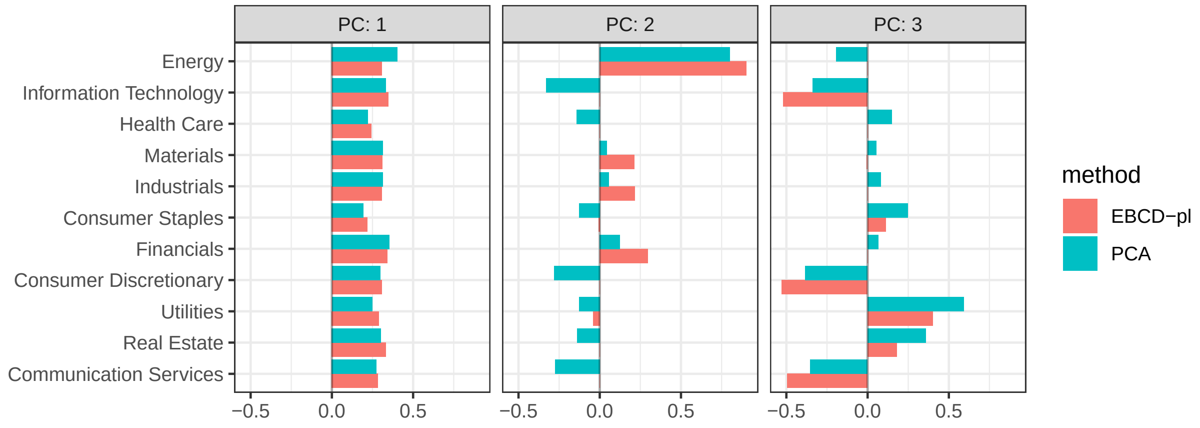


Figure 2.4: Comparison of PCA loadings and estimated posterior mean loadings from `EBCD-pl`. (To facilitate comparisons we post-processed posterior mean loadings to have unit norm.)

The first PC (both classical and `EBCD-pl`) loads roughly equally on all sectors, and so captures the tendency of sectors to move together as the market varies. To describe the loadings on the second and the third `EBCD-pl` PCs, we group the sectors into four groups: energy, materials, industrials, and financials (EMIF); consumer staples, utilities, real estate (SUR); information technology, consumer discretionary, and communication services (TDC); and health care. The second `EBCD-pl` PC captures the EMIF sector, and the third `EBCD-pl` PC captures the contrast between SUR and TDC.

These `EBCD-pl` results can be interpreted in the context of the Fama-French three-factor model [Fama and French, 1993], which is the standard model in finance that explains variation in stock prices by three factors: the market factor (roughly, overall average performance of all stocks), the size factor (SMB, for small minus big, contrasting stocks with small vs big market capitalization), and the growth/value factor (HML, for high minus low, contrasting high value stocks, which have high book-to-market value ratio, with growth stocks which have low book-to-market ratio). The first `EBCD-pl` PC captures the market factor, whereas the second and third PCs partition the sectors into three groups: the TDC group contains the growth sectors; the EMIF group contains the strong value sectors with smaller sizes and the SUR group contains the moderately value sectors with larger sizes. This is illustrated graphically in Figure 2.5, which shows each sector in the Fama-French SMB-HML plane (data from the Data Library maintained by Kenneth R. French), colored according to loading on the second and third PCs. The colorings for `EBCD-pl` PCs clearly capture contiguous regions of the plane. (In contrast the classic PCs do not align so closely with the Fama-French factors; in particular the third PC groups the energy sector with TDC, which do not fall together in the SMB-HML plane.)

2.8 Discussion

We introduced a simple penalized PCA criterion, (2.2) that unites some existing sparse PCA methods (SPC and GPower). We showed that this criterion has the property of simultaneously providing a decomposition of both the data matrix and the covariance, or Gram, matrix. To address the challenge of tuning multiple hyperparameters, we proposed an empirical Bayes approach that integrates hyperparameter tuning directly within the algorithm. The result is an empirical Bayes approach to covariance decomposition (EBCD), which we found in simulations can outperform existing methods for sparse PCA.

While we have focused here on sparsity, our EBCD approach is quite general, and other structures can be easily incorporated simply by changing the prior family used. For example, replacing the point-Laplace prior family we used here with a point-Exponential prior family immediately leads to a new EB method for sparse, non-negative PCA [Zass and Shashua, 2006] (and, simultaneously, a version of semi-nonnegative matrix factorization [Ding et al.,

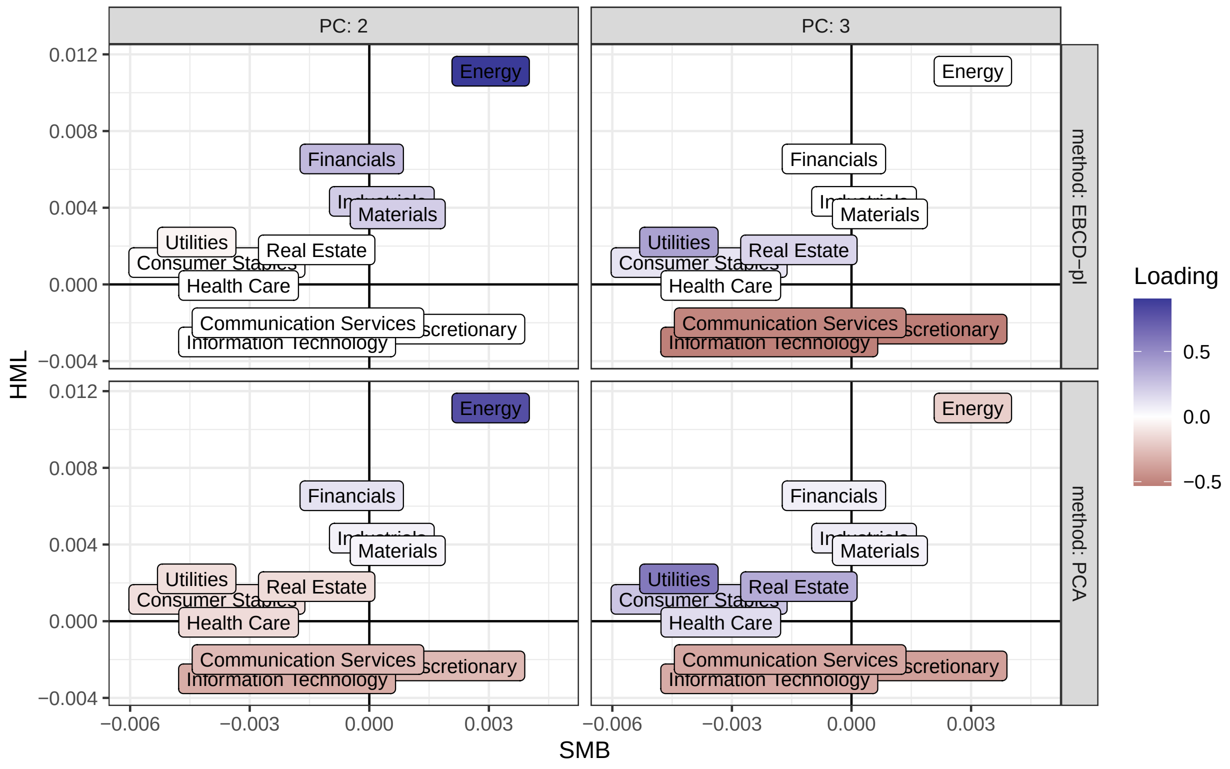


Figure 2.5: Sectors projected on the SMB-HML plane. Each sector is positioned according to its loadings on the Fama-French SMB and HML factors, and is colored based on its loadings on the second and third principal components (PCs) from the `EBCD-pl` method (or PCA).

2010]). The non-negative constraint may provide more interpretable covariance decompositions in many applications; see Li et al. [2021] for interesting recent work in this direction. Another interesting possibility to improve interpretation is to use binary or near-binary priors, which would lead to empirical Bayes versions of additive clustering [Shepard and Arabie, 1979]; see also Kueng and Tropp [2021], Sørensen et al. [2022], Kolomvakis and Gillis [2023], Liu et al. [2023]. Similarly, one could obtain an EB version of "functional PCA" [Ramsay and Silverman, 2005] by replacing the sparse prior with a "spatial" prior that encourages $|\eta_i - \eta_{i+1}|$ (in Definition 2.2) to be typically small. EBNM solvers for a range of priors are implemented in the EBNM package [Willwerscheid and Stephens, 2021], and an EBNM solver for a spatial prior is implemented using wavelet methods in Xing et al. [2021], and any of these could be immediately plugged into Algorithm 2.1. It is, however, possible that some prior families may require careful attention to initialization to yield good performance.

2.9 Data and Resources

`EBCD-pl` is implemented in the R package `ebcd` that is available from `https://github.com/joonsukkang/ebcd`. Source code for the empirical results is available from `https://github.com/joonsukkang/ebcd-paper`.

The sector-level daily returns data was provided by Refinitiv via Wharton Research Data Services. Data will be shared on request to the corresponding author with permission of Refinitiv. The Fama-French 3 Factor Returns data is available in the Data Library maintained by Kenneth R. French: `https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html`.

2.10 Appendix

2.10.1 Proof of Theorem 2.1

To prove Theorem 2.1 we first prove Lemma 2.1; and to prove Lemma 2.1 we use the following Lemma summarizing some properties of the nuclear norm $\|\cdot\|_*$.

Lemma 2.2. *For any real-valued matrices* $\mathbf{A} \in \mathcal{M}(N_1, N_2)$ *and* $\mathbf{B} \in \mathcal{M}(N_2, N_3)$,

(a) $\|\mathbf{A}\|_* = tr(\mathbf{A}^T Polar.U(\mathbf{A}))$.

(b) $\|\mathbf{A}\|_* = tr(\sqrt{\mathbf{A}\mathbf{A}^T})$.

(c) $\|\mathbf{A}\mathbf{B}\|_* = \|\sqrt{\mathbf{A}^T\mathbf{A}}\sqrt{\mathbf{B}\mathbf{B}^T}\|_*$.

Proof. Let $\mathbf{U}_A\mathbf{D}_A\mathbf{V}_A^T$ and $\mathbf{U}_B\mathbf{D}_B\mathbf{V}_B^T$ denote the SVDs of $\mathbf{A}$ and $\mathbf{B}$ respectively. (a) From Definition 2.1, Polar.U$(\mathbf{A}) = \mathbf{U}_A\mathbf{V}_A^T$; $\text{tr}(\mathbf{A}^T\text{Polar.U}(\mathbf{A})) = \text{tr}(\mathbf{V}_A\mathbf{D}_A\mathbf{U}_A^T\mathbf{U}_A\mathbf{V}_A^T) = \text{tr}(\mathbf{D}_A) = \|\mathbf{A}\|_*$. (b) $\text{tr}(\sqrt{\mathbf{A}\mathbf{A}^T}) = \text{tr}(\sqrt{\mathbf{U}_A\mathbf{D}_A\mathbf{V}_A^T\mathbf{V}_A\mathbf{D}_A\mathbf{U}_A^T}) = \text{tr}(\mathbf{U}_A\mathbf{D}_A\mathbf{U}_A^T) = \text{tr}(\mathbf{D}_A) = \|\mathbf{A}\|_*$. (c) Since the nuclear norm is unitarily invariant, we have $\|\mathbf{A}\mathbf{B}\|_* = \|\mathbf{U}_A\mathbf{D}_A\mathbf{V}_A^T\mathbf{U}_B\mathbf{D}_B\mathbf{V}_B^T\|_* = \|\mathbf{D}_A\mathbf{V}_A^T\mathbf{U}_B\mathbf{D}_B\|_* = \|\mathbf{V}_A\mathbf{D}_A\mathbf{V}_A^T\mathbf{U}_B\mathbf{D}_B\mathbf{U}_B^T\|_* = \|\sqrt{\mathbf{A}^T\mathbf{A}}\sqrt{\mathbf{B}\mathbf{B}^T}\|_*$. □

Proof of Lemma 2.1

Proof. From Fact 2.1 that $\hat{\mathbf{Z}}(\mathbf{L},\mathbf{X}) = \text{Polar.U}(\mathbf{XL})$, we have $h(\mathbf{X},\mathbf{L}) = \text{tr}(\mathbf{X}^T\mathbf{X}) + \text{tr}(\mathbf{LL}^T) - 2\text{tr}(\mathbf{L}^T\mathbf{X}^T\text{Polar.U}(\mathbf{XL}))$. The last term is equal to $-2\|\mathbf{XL}\|_*$ from Lemma 2.2(a), to $-2\|\sqrt{\mathbf{X}^T\mathbf{X}}\sqrt{\mathbf{LL}^T}\|_*$ from Lemma 2.2(c), and to $-2\text{tr}(\sqrt{\sqrt{\mathbf{X}^T\mathbf{X}}\mathbf{LL}^T\sqrt{\mathbf{X}^T\mathbf{X}}})$ from Lemma 2.2(b). Therefore, $h(\mathbf{X},\mathbf{L}) = \text{tr}(\mathbf{X}^T\mathbf{X}) + \text{tr}(\mathbf{LL}^T) - 2\text{tr}(\sqrt{\sqrt{\mathbf{X}^T\mathbf{X}}\mathbf{LL}^T\sqrt{\mathbf{X}^T\mathbf{X}}}) = d_*(\mathbf{X}^T\mathbf{X},\mathbf{LL}^T)^2$.

□

Proof of Theorem 2.1

Proof. Let $(\hat{\mathbf{Z}},\hat{\mathbf{L}})$ denote a solution to the penalized PCA criterion (2.12). That is,

$$\frac{1}{2}\|\mathbf{X} - \hat{\mathbf{Z}}\hat{\mathbf{L}}^T\|_F^2 + \sum_{k=1}^{K} P(\hat{\mathbf{l}}_k;\lambda_k) = \min_{\substack{\mathbf{Z}\in\mathcal{S}(N,K),\\ \mathbf{L}\in\mathcal{M}(P,K)}} \left(\frac{1}{2}\|\mathbf{X} - \mathbf{ZL}^T\|_F^2 + \sum_{k=1}^{K} P(\mathbf{l}_k;\lambda_k)\right). \tag{2.43}$$

Since $\hat{\mathbf{Z}}$ is the minimizer of $\|\mathbf{X} - \mathbf{Z}\hat{\mathbf{L}}^T\|_F^2$ by construction, the LHS of (2.43) is equal to

$$\frac{1}{2}\min_{\mathbf{Z}\in\mathcal{S}(N,K)} \|\mathbf{X} - \mathbf{Z}\hat{\mathbf{L}}^T\|_F^2 + \sum_{k=1}^{K} P(\hat{\mathbf{l}}_k;\lambda_k) = \frac{1}{2}d_*(\mathbf{X}^T\mathbf{X},\hat{\mathbf{L}}\hat{\mathbf{L}}^T)^2 + \sum_{k=1}^{K} P(\hat{\mathbf{l}}_k;\lambda_k) \tag{2.44}$$

by Lemma 2.1. Similarly, by Lemma 2.1, the RHS of (2.43) is equal to

$$\begin{aligned}
&\min_{\mathbf{L}\in\mathcal{M}(P,K)} \left(\frac{1}{2}\min_{\mathbf{Z}\in\mathcal{S}(N,K)} \|\mathbf{X} - \mathbf{ZL}^T\|_F^2 + \sum_{k=1}^{K} P(\mathbf{l}_k;\lambda_k)\right) \\
&= \min_{\mathbf{L}\in\mathcal{M}(P,K)} \left(\frac{1}{2}d_*(\mathbf{X}^T\mathbf{X},\mathbf{LL}^T)^2 + \sum_{k=1}^{K} P(\mathbf{l}_k;\lambda_k)\right).
\end{aligned} \tag{2.45}$$

Equating the right-hand-sides of (2.44) and (2.45) shows that $\hat{\mathbf{L}}$ is a solution to the penalized covariance decomposition criterion (2.13). □

2.10.2 Proof of Proposition 2.3

Proof. The evidence lower bound (ELBO) of the model, $F(\mathbf{g}, \mathbf{Z}, \tau, \mathbf{q})$, can be written as

$$F(\mathbf{g}, \mathbf{Z}, \tau, \mathbf{q}) = -\frac{NP}{2}\log(2\pi) + \frac{NP}{2}\log(\tau) - \frac{\tau}{2}\mathbb{E}_{\mathbf{q}}\left[\|\mathbf{X} - \mathbf{Z}\mathbf{L}^T\|_F^2\right] + \mathbb{E}_{\mathbf{q}}\left[\log\frac{\mathbf{g}(\mathbf{L})}{\mathbf{q}(\mathbf{L})}\right], \tag{2.46}$$

and the three steps iteratively maximizing the ELBO can be shown as follows. (a) *EBNM step*: maximizing ELBO with respect to $(\mathbf{g}, \mathbf{q})$ factorizes into K subproblems of the form $\max_{(g_k, q_k)} \mathbb{E}_{q_k}\left[\log\frac{g_k(\mathbf{l}_k)\prod_p \exp(-\frac{\tau}{2}(l_{p,k} - (\mathbf{X}^T\mathbf{Z})_{p,k})^2)}{q_k(\mathbf{l}_k)}\right]$, which corresponds to the EBNM problem EBNM$(\mathbf{X}^T\mathbf{z}_k, 1/\tau, \mathcal{G})$. (b) *Rotation step*: maximizing ELBO with respect to $\mathbf{Z}$ reduces to a reduced-rank Procrustes rotation problem, $\min_{\mathbf{Z}} \|\mathbf{X} - \mathbf{Z}\bar{\mathbf{L}}^T\|_F^2$, which has the solution Polar.U$(\mathbf{X}\bar{\mathbf{L}})$. (c) *Precision step*: maximizing ELBO with respect to τ has the closed form solution $\tau = NP/\mathbb{E}_{\mathbf{q}}\left[\|\mathbf{X} - \mathbf{Z}\mathbf{L}^T\|_F^2\right] = NP/(\|\mathbf{X} - \mathbf{Z}\bar{\mathbf{L}}^T\|_F^2 + \|\mathbf{V}\|_{1,1})$. □

2.10.3 Proof of Proposition 2.4

Proof. The evidence lower bound (ELBO) of the model, $F(\mathbf{g}, \mathbf{Z}, \tau, \mathbf{q})$ in (2.46), can be rearranged as

$$-\frac{\tau}{2}\|\mathbf{X} - \mathbf{Z}\bar{\mathbf{L}}\|_F^2 - \tau\sum_{p,k}\left(\frac{N}{2K\tau}\log\left(\frac{2\pi}{\tau}\right) + \frac{1}{2}\left(var_{q_{p,k}}(l_{p,k}) + \frac{2}{\tau}\mathbb{KL}(q_{p,k}||g_k)\right)\right), \tag{2.47}$$

and after taking the maximum over $\mathbf{q} : \mathbb{E}[\mathbf{L}] = \bar{\mathbf{L}}$, we get the expression (2.33). □

2.10.4 Extension 1. Scaled versions of the sparse PCA criterion

One slightly unnatural feature of the formulations presented in the main text is that they place a penalty (or prior) on a parameter, $\mathbf{L}$, that is not a “population quantity", and whose interpretation changes with the number of samples N. For example, in Section 2.5

we saw that the fidelity term encourages $\mathbf{L}\mathbf{L}^T \approx \mathbf{X}^T\mathbf{X}$, whose magnitude grows with N; it would seem more natural to combine a penalty on $\mathbf{L}$ with a fidelity term that encourages $\mathbf{L}\mathbf{L}^T \approx (1/N)\mathbf{X}^T\mathbf{X}$ since the latter has a natural limit as $N \to \infty$ (with P fixed). This can be achieved simply by replacing the constraint $\mathbf{Z}^T\mathbf{Z} = \mathbf{I}_K$ with the scaled version $\mathbf{Z}^T\mathbf{Z} = N\mathbf{I}_K$, or equivalently $\mathbf{Z}/\sqrt{N} \in \mathcal{S}(N, K)$. All our results and algorithms are easily modified for this rescaled version. For example, the sparse PCA criterion (2.2) becomes

$$\min_{\substack{\mathbf{Z}/\sqrt{N}\in\mathcal{S}(N,K),\\ \mathbf{L}\in\mathcal{M}(P,K)}} \left(\frac{1}{2}\|\mathbf{X} - \mathbf{Z}\mathbf{L}^T\|_F^2 + \sum_{k=1}^{K} P(\mathbf{l}_k; \lambda_k) \right); \tag{2.48}$$

the equivalent covariance formulation ((2.13) and (b) in Theorem 2.2) becomes

$$\min_{\mathbf{L}\in\mathcal{M}(P,K)} \left(\frac{N}{2} d_*(\mathbf{X}^T\mathbf{X}/N, \mathbf{L}\mathbf{L}^T)^2 + \sum_{k=1}^{K} P(\mathbf{l}_k; \lambda_k) \right); \tag{2.49}$$

the equivalent compact matrix formulation ((d) in Theorem 2.2) becomes

$$\operatorname*{arg\,min}_{\substack{\tilde{\mathbf{Z}}/\sqrt{N}\in\mathcal{S}(P,K),\\ \mathbf{L}\in\mathcal{M}(P,K)}} \left(\frac{1}{2}\|\mathbf{C} - \tilde{\mathbf{Z}}\mathbf{L}^T\|_F^2 + \sum_{k=1}^{K} P(\mathbf{l}_k; \lambda_k) \right); \tag{2.50}$$

and the penalty term (2.33) becomes

$$P_{\tau,g}(\bar{l}) = \frac{N}{2K\tau} \log \frac{2\pi}{\tau} + \frac{1}{2} \min_{q:\mathbb{E}_q[l]=\bar{l}} \left(N var_q(l) + \frac{2}{\tau} \mathbb{KL}(q||g) \right). \tag{2.51}$$

The BISPCA updates become

$$\mathbf{l}_k \leftarrow S_{\rho/N}(\mathbf{X}^T\mathbf{z}_k/N; \lambda_k); \qquad \mathbf{Z} \leftarrow \sqrt{N}\text{Polar.U}(\mathbf{X}\mathbf{L}); \tag{2.52}$$

and the EBCD updates (2.24)-(2.26) become

$$\text{EBNM step: for each } k \in [K], \quad (g_k, q_k) \leftarrow \text{EBNM}(\mathbf{X}^T\mathbf{z}_k/N, 1/N\tau, \mathcal{G}) \tag{2.53}$$

$$\text{Rotation step: } \mathbf{Z} \leftarrow \sqrt{N}\text{Polar.U}(\mathbf{X}\bar{\mathbf{L}}) \tag{2.54}$$

$$\text{Precision step: } \tau \leftarrow NP/(\|\mathbf{X} - \mathbf{Z}\bar{\mathbf{L}}^T\|_F^2 + N\|\mathbf{V}\|_{1,1}) \left[= P/(d_*(\mathbf{X}^T\mathbf{X}/N, \mathbf{L}\mathbf{L}^T)^2 + \|\mathbf{V}\|_{1,1})\right]. \tag{2.55}$$

And, just as before, one can apply these updates to a compact version of the data matrix to solve the same problem.

This modification to the methods makes it easier to reason about their behavior in the regime $N \to \infty$ with P fixed, where we can assume $\lim_{N\to\infty} \mathbf{X}^T\mathbf{X}/N = \mathbf{S}$ say. For example, (2.49) shows that for a fixed penalty (not depending on N) the influence of the penalty will decrease as N increases, and the limiting estimate of $\mathbf{L}$ will be $\in \arg\min d_*(\mathbf{S}, \mathbf{L}\mathbf{L}^T)$ independent of the penalty. And because the part of the penalty (2.51) depending on g does not scale with N, the effect of the prior g diminishes as $N \to \infty$ as one might expect (indeed, in the limit as $N \to \infty$ the EBCD optimum $\mathbf{L}$ will be $\in \arg\min d_*(\mathbf{S}, \mathbf{L}\mathbf{L}^T)$ whether g is fixed or estimated from the data).

2.10.5 Extension 2. Column-wise variances

We can extend the EBCD model (2.14)-(2.16) to allow different variables to have different variances/precisions:

$$\mathbf{X} = \mathbf{Z}\mathbf{L}^T + \mathbf{E} \tag{2.56}$$

$$l_{p,k} \sim^{\text{iid}} g_k \in \mathcal{G} \tag{2.57}$$

$$e_{n,p} \sim^{\text{iid}} N(\cdot; 0, 1/\tau_p) \tag{2.58}$$

where $\mathbf{Z} \in \mathcal{S}(N, K)$. Fitting this heteroskedastic model requires solutions for the heteroskedastic versions of the reduced-rank Procrustes rotation problem and the EBNM problem, as we now detail.

Fact 2.2 (Heteroskedastic Reduced-rank Procrustes rotation problem)**.** *Given* $\mathbf{L}$*, the minimum*

$$\min_{\mathbf{Z}\in\mathcal{S}(N,K)} \sum_{n,p} \tau_p(x_{n,p} - (\mathbf{Z}\mathbf{L}^T)_{n,p})^2$$

is achieved by $\hat{\mathbf{Z}}(\mathbf{L}, \mathbf{X}, \mathbf{T}) := \textit{Polar.U}(\mathbf{XTL})$ *where* $\mathbf{T}$ *is the* $P \times P$ *diagonal matrix with* $T_{p,p} = \tau_p$*.*

Proof. The minimization problem is equivalent to $\min_{\mathbf{Z}\in\mathcal{S}(N,K)} \|(\mathbf{X}-\mathbf{Z}\mathbf{L}^T)\sqrt{\mathbf{T}}\|_F^2 = \min_{\mathbf{Z}\in\mathcal{S}(N,K)}$ $\|\mathbf{X}\sqrt{\mathbf{T}}-\mathbf{Z}(\sqrt{\mathbf{T}}\mathbf{L})^T\|_F^2$, which reduces to a (homoskedastic) reduced-rank Procrustes rotation problem in Fact 2.1 and has a solution Polar.U$(\mathbf{X}\sqrt{\mathbf{T}}\sqrt{\mathbf{T}}\mathbf{L})$ = Polar.U$(\mathbf{XTL})$, where $\sqrt{\mathbf{T}}$ is the $P \times P$ diagonal matrix with diagonal entries $\sqrt{\tau_p}$. □

Definition 2.3. *Let EBNM*$(\mathbf{x}, \mathbf{s}^2, \mathcal{G})$ *denote a function that returns the EB solution to the following heteroskedastic normal means model:*

$$x_p|\eta_p, s_p^2 \sim^{indep} N(x_p; \eta_p, s_p^2) \tag{2.59}$$

$$\eta_p \sim^{iid} g \in \mathcal{G}, \tag{2.60}$$

for $p = 1, \dots, P$*.*

Proposition 2.5. *Maximizing the evidence lower bound* $F(\mathbf{g}, \mathbf{Z}, \mathbf{T}, \mathbf{q})$ *((2.20) but with* τ *replaced by* $\mathbf{T}$*) subject to* $\mathbf{Z}^T\mathbf{Z} = \mathbf{I}_K$ *can be achieved by iteratively updating* $(\mathbf{g}, \mathbf{q})$*, updating*

$\mathbf{Z}$, *and updating* $\mathbf{T}$, *as follows:*

$$\textit{EBNM step: for each } k \in [K], \quad (g_k, q_k) \leftarrow \textit{EBNM}(\mathbf{X}^T \mathbf{z}_k, (1/\tau_1, \ldots, 1/\tau_P), \mathcal{G}) \tag{2.61}$$

$$\textit{Rotation step: } \mathbf{Z} \leftarrow \textit{Polar.}U(\mathbf{X}\mathbf{T}\bar{\mathbf{L}}) \tag{2.62}$$

$$\textit{Precision step: } \tau_p \leftarrow N/\left(\sum_n (x_{n,p} - (\mathbf{Z}\bar{\mathbf{L}})_{n,p})^2 + \sum_k v_{p,k}\right), p = 1, \ldots, P. \tag{2.63}$$

Here $\bar{\mathbf{L}} = \mathbb{E}_{\mathbf{q}}(\mathbf{L})$ *and* $v_{p,k} = \textit{Var}_{q_k}(l_{p,k})$.

Note that in practice, one would need to apply some regularization when estimating τ_p to prevent solutions with $\tau_p \to \infty$.

CHAPTER 3

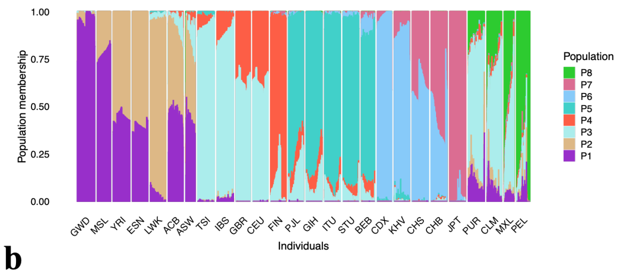
GENETIC DRIFT-BASED INFERENCE OF POPULATION STRUCTURE

3.1 Introduction

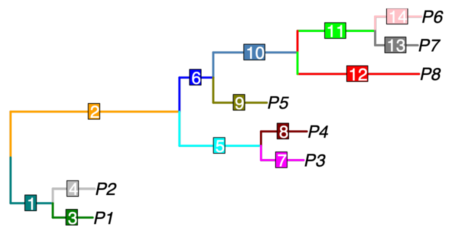
Admixture-based clustering methods [Pritchard et al., 2000, Alexander et al., 2009, Raj et al., 2014] are widely used to infer population structure from genotype data. Given a matrix $\mathbf{G} \in \{0,1,2\}^{S \times N}$ of genotype data from S SNPs and N individuals, these methods simultaneously estimate K latent populations parameterized by their population allele frequencies and each individual's memberships in the K populations.

In terms of matrix decomposition, these methods find a decomposition of $\mathbf{G}$ such that $\mathbf{G} \approx 2\mathbf{P}\mathbf{Q}^T$ holds where $\mathbf{P} \in [0,1]^{S \times K}$ is the matrix of population allele frequencies and $\mathbf{Q} \in [0,1]^{N \times K}$ is the matrix of individuals' population memberships. Each row of $\mathbf{Q}$, corresponding to one individual's memberships in K populations, is nonnegative and sums to one. Once the admixture-based clustering method is estimated, the individuals' population membership matrix $\mathbf{Q}$ is used to visualize the result in the form of the popular 'STRUCTURE bar plot', in which each individual's population memberships are stacked along the y-axis and individuals are positioned along the x-axis. An example of the STRUCTURE bar plot, replicated from The 1000 Genomes Project Consortium [2015], is shown in Figure 3.1a.

While the STRUCTURE bar plot contains information on the genetic similarities of sampled individuals, it does not contain all of this information. This is because the bar plot shows only the inferred $\mathbf{Q}$ matrix, which represents admixture proportions, and not the inferred $\mathbf{P}$ matrix, which represents population allele frequencies, and much of the genetic similarity between individuals depends on this. For example, from the plot it is impossible to say whether FIN individuals are more similar to CDX individuals or to GWD individuals, because this depends on how similar P4 is to P6 and P1, which is information in the $\mathbf{P}$ matrix a



b



c

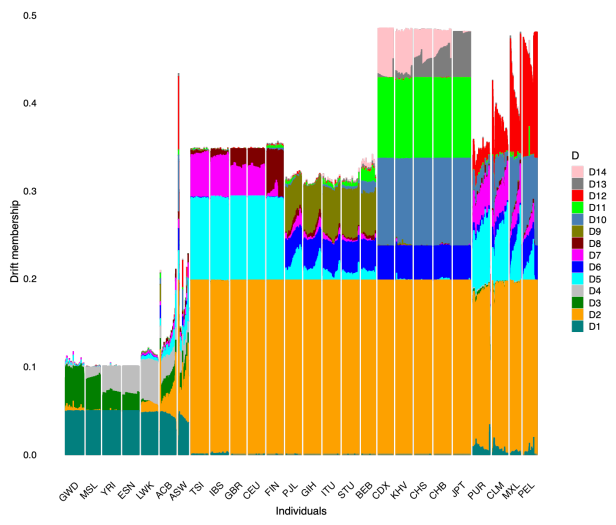


Figure 3.1: An illustration of (a) STRUCTURE bar plot, (b) tree-based genetic drift estimation, and (c) tree-based individual DRIFT bar plot, using the 1000 Genomes Project data.

not information conveyed in the plot. More generally, STRUCTURE plots, by simply coloring each (inferred) population a different color without regard to their genetic similarities, may give misleading impressions of genetic variation among individuals: genetically similar individuals may be assigned to different (but genetically-similar) populations, and so appear very different on the barplot.

In this chapter, we address this problem by introducing an alternative barplot representation that *combines* relevant information in both the inferred admixture proportions ($\mathbf{Q}$) and the population allele frequencies ($\mathbf{P}$). In brief, the idea is to decompose the population allele frequencies into independent "drift" components, and then to plot a barplot of the implied memberships of each individual in these drift components, rather than their memberships in populations. We consider two approaches to inferring drift components, one based on inferring an evolutionary tree relating the inferred populations inferred from STRUCTURE, and a second, which we call Treelax, that is based on a matrix factorization approach that relaxes the assumption of a strict evolutionary tree relating the populations. Because the drift components are independent, the resulting barplot representation provides a faithful representation of the genetic similarities among individuals in the sample, while maintaining many of the appealing visual features of the STRUCTURE barplot that have made it such a popular tool in practice.

3.2 Description of the method

3.2.1 The DRIFT method

In brief, whereas STRUCTURE represents the genotype matrix as $\mathbf{G} \approx 2\mathbf{PQ}$ where $\mathbf{P}$ and $\mathbf{Q}$ are non-negative, we instead seek a representation of the form

$$\mathbf{G} \approx 2\mathbf{Z}\mathbf{M}^T \text{ where } \mathbf{Z}^T\mathbf{Z} = S\mathbf{I} \text{ and } \mathbf{M} \text{ non-negative,} \tag{3.1}$$

which in turn implies

$$(1/S)\mathbf{G}^T\mathbf{G} \approx 4\mathbf{M}\mathbf{M}^T \text{ where } \mathbf{M} \text{ is non-negative.} \tag{3.2}$$

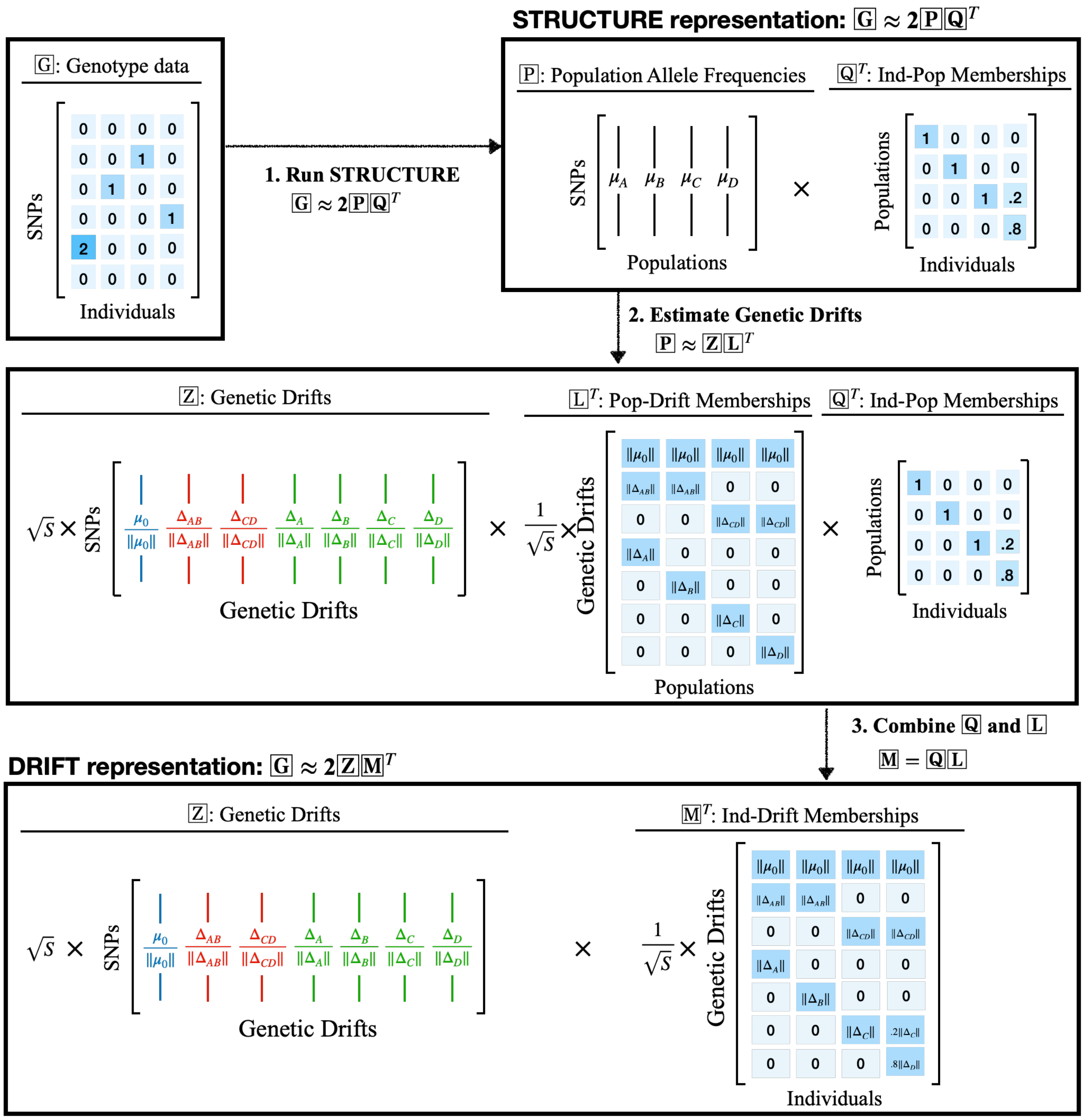
For reasons we describe later, we refer to $\mathbf{Z}$ as the drift matrix, and $\mathbf{M}$ as the (individual) drift membership matrix.

Note that the matrix $(1/S)\mathbf{G}^T\mathbf{G}$ contains all the information about the genetic distance between individuals i and j. Indeed, the (squared) genetic distances between individuals i and j is

$$D_{ij} = \frac{1}{S}\sum_{s=1}^{S}(G_{s,i} - G_{s,j})^2 \tag{3.3}$$

so in matrix notation

$$\mathbf{D} = \boldsymbol{\alpha}\mathbf{1}^T + \mathbf{1}\boldsymbol{\alpha}^T - \frac{2}{S}\mathbf{G}^T\mathbf{G} \tag{3.4}$$



STRUCTURE representation: $\boxed{G} \approx 2\boxed{P}\boxed{Q}^T$
$\boxed{G}$: Genotype data
SNPs
Individuals
1. Run STRUCTURE
$\boxed{G} \approx 2\boxed{P}\boxed{Q}^T$
$\boxed{P}$: Population Allele Frequencies
μ_A μ_B μ_C μ_D
Populations
$\boxed{Q}^T$: Ind-Pop Memberships
Individuals
2. Estimate Genetic Drifts
$\boxed{P} \approx \boxed{Z}\boxed{L}^T$
$\boxed{Z}$: Genetic Drifts
$\sqrt{S}\times$
$\frac{\mu_0}{\|\mu_0\|}$ $\frac{\Delta_{AB}}{\|\Delta_{AB}\|}$ $\frac{\Delta_{CD}}{\|\Delta_{CD}\|}$ $\frac{\Delta_A}{\|\Delta_A\|}$ $\frac{\Delta_B}{\|\Delta_B\|}$ $\frac{\Delta_C}{\|\Delta_C\|}$ $\frac{\Delta_D}{\|\Delta_D\|}$
Genetic Drifts
$\boxed{L}^T$: Pop-Drift Memberships
$\frac{1}{\sqrt{S}}\times$
Populations
3. Combine $\boxed{Q}$ and $\boxed{L}$
$\boxed{M} = \boxed{Q}\boxed{L}$
DRIFT representation: $\boxed{G} \approx 2\boxed{Z}\boxed{M}^T$
$\boxed{M}^T$: Ind-Drift Memberships

Figure 3.2: Overview of the DRIFT method's three-step process.

where $\boldsymbol{\alpha} = \text{diag}(\frac{1}{S}\mathbf{G}^T\mathbf{G})$. Thus, the approximation (3.2) implies that the matrix $\mathbf{M}$ contains (approximately) the information on genetic distances. Indeed, simple algebra yields

$$D_{ij} \approx 4\sum_{k=1}^{K}(M_{i,k} - M_{j,k})^2. \tag{3.5}$$

As a result, a barplot of elements of $\mathbf{M}$ captures the information on genetic similarities between individuals, whereas the STRUCTURE barplot of the elements of the matrix $\mathbf{Q}$ does not.

There may be many ways to obtain a factorization of the form (3.1), and we present here one approach, with two variations, that we believe is useful; future work may provide other approaches. Figure 3.2 presents an outline of our approach, which consists of three steps:

Step 1. Run a STRUCTURE-type method on genotype data ($\mathbf{G}$) to obtain population allele frequencies ($\mathbf{P}$) and individuals' population memberships ($\mathbf{Q}$) such that $\mathbf{G} \approx 2\mathbf{P}\mathbf{Q}^T$.[1]

Step 2. Decompose $\mathbf{P}$ as $\mathbf{P} \approx \mathbf{Z}\mathbf{L}^T$, where $\mathbf{Z}^T\mathbf{Z} = S\mathbf{I}$ (which implies that $(1/S)\mathbf{P}^T\mathbf{P} \approx \mathbf{L}\mathbf{L}^T$). We describe two ways to achieve this decomposition – one using a tree-based method and another based on a matrix factorization that relaxes the tree assumption – in more detail below. We refer to $\mathbf{Z}$ as the drift matrix, and $\mathbf{L}$ as the matrix of population drift memberships.

Step 3. Combining the above, we get $\mathbf{G} \approx 2\mathbf{Z}\mathbf{M}^T$ where $\mathbf{M} := \mathbf{Q}\mathbf{L}$ and $\mathbf{Z}^T\mathbf{Z} = S\mathbf{I}$, as desired.

Although throughout we have used $\mathbf{G} \approx 2\mathbf{Z}\mathbf{M}^T$ as a device to draw connections with STRUCTURE, in practice we do not actually need the matrix $\mathbf{Z}$ to produce our proposed plot. Indeed the approach can be alternatively described as seeking the decomposition (3.2)

1. For potential limitations of STRUCTURE-type methods, including sensitivity to the sampling scheme and difficulty in choosing the number of ancestral populations (K), see Lawson et al. [2018].

without ever introducing or estimating $\mathbf{Z}$, and this can be achieved as follows: 1. Decompose $\mathbf{G} \approx 2\mathbf{P}\mathbf{Q}^T$; 2. Decompose $(1/S)\mathbf{P}^T\mathbf{P} \approx \mathbf{L}\mathbf{L}^T$; 3. Form $\mathbf{M} = \mathbf{Q}\mathbf{L}$, which will satisfy $(1/S)\mathbf{G}^T\mathbf{G} \approx 4\mathbf{M}\mathbf{M}^T$.

3.2.2 Tree-based estimation of genetic drifts

One way to approximate $\mathbf{P} \approx \mathbf{Z}\mathbf{L}^T$ is to use tree-based methods [Cavalli-Sforza et al., 1964, Cavalli-Sforza and Edwards, 1967, Felsenstein, 1973, 1981, Pickrell and Pritchard, 2012] to model the evolutionary relationships between the allele frequencies of different populations, and then to translate the inferred tree into a matrix representation. In this approach, the matrix $\mathbf{L}$ is obtained from information on the tree topology (including location of the root) and the length of each branch in the tree, and the matrix $\mathbf{Z}$ contains the information on the change in allele frequencies that occur along each branch, often known as the “genetic drift" (drift refers to the random fluctuations in the frequencies of neutral genetic variants resulting from gene sampling during reproduction, but here we use it to refer to the changes that occur whether or not they are neutral).

The idea is illustrated in Figure 3.1. Figure 3.1b illustrates an evolutionary tree reconstructed from the estimated population allele frequencies of eight populations, with the drifts color-coded and numbered from one to fourteen. The tree is estimated using the TreeMix method [Pickrell and Pritchard, 2012] with zero admixture events and is rooted using an ancestral population with ancestral alleles as an outgroup. For example, the predominantly South Asian population P5 is estimated to have experienced the Out-of-Africa drift D2, the pan-Asian drift D6 (shared across South Asians, East Asians, and Americans), and the P5-specific drift D9.

To translate the inferred tree into the corresponding $\mathbf{L}$ matrix encoding populations' drift memberships, we utilize the tree topology to determine binary population-drift memberships, while the branch lengths provide information about the size of these drifts. The left plot of

Figure 3.9c shows the corresponding elements of the matrix $\mathbf{L}$. In this plot, each population's scaled binary memberships in drifts D1 through D14 are stacked along the y-axis, and populations are positioned along the x-axis. We call it the 'population DRIFT bar plot'. For example, P5's memberships in D2, D6, and D9 are stacked, with the bar heights representing drift sizes determined by the tree branch lengths. Intuitively, a population DRIFT bar plot can be obtained by rotating a tree plot 90 degrees counterclock-wise.

3.2.3 Treelax: Matrix decomposition-based estimation of genetic drifts

While approximating the relationships among populations as a bifurcating tree is a simple and intuitive approach, the adequacy of the assumed tree structure remains an empirical question. When a tree method applied to data suggests a lack of fit, we can improve the model by relaxing the tree assumption. Here we propose to relax the tree assumptioon using a matrix decomposition-based method that penalizes deviations from the initial tree-based estimate. We call our method "Treelax", reflecting its ability to relax the tree assumption.

To describe Treelax, let $\tilde{\mathbf{L}}$ denote an estimate of $\mathbf{L}$ obtained by a tree-based method. Treelax seeks an $\mathbf{L}$ such that $(1/S)\mathbf{P}^T\mathbf{P} \approx \mathbf{L}\mathbf{L}^T$ while $\mathbf{L}$ is also close to $\tilde{\mathbf{L}}$. It does so by optimizing the following objective function:

$$\mathcal{L}(\mathbf{L}; \boldsymbol{\Delta}, \tilde{\mathbf{L}}, \lambda) = \|\boldsymbol{\Delta} - \Big(\boldsymbol{\beta}\mathbf{1}^T + \mathbf{1}\boldsymbol{\beta}^T - 2\mathbf{L}\mathbf{L}^T\Big)\|_F^2 + \lambda\|\mathbf{L} - \tilde{\mathbf{L}}\|_1 \tag{3.6}$$

where $\boldsymbol{\beta} := \text{diag}(\mathbf{L}\mathbf{L}^T)$, $\boldsymbol{\Delta}$ is the matrix of pairwise squared genetic distances between populations based on their allele frequencies:

$$\Delta_{k,k'} = \frac{1}{S}\sum_{s=1}^{S}(P_{s,k} - P_{s,k'})^2 \tag{3.7}$$

and $\lambda \geq 0$ is a tuning parameter that controls the strength of the L^1 penalty. Note that the matrix $\left(\boldsymbol{\beta}\mathbf{1}^T + \mathbf{1}\boldsymbol{\beta}^T - 2\mathbf{L}\mathbf{L}^T\right)$ is the matrix of (squared) pairwise distances implied by $\mathbf{L}$, so the objective function is a combination of a fidelity term (that makes the distances implied by $\mathbf{L}$ close to the observed distances $\boldsymbol{\Delta}$) and a penalty term that makes $\mathbf{L}$ close to $\tilde{\mathbf{L}}$.[2] As λ decreases towards zero, the solution approaches $\mathbf{L}^*$ that minimizes the fidelity term. As λ increases, the solution approaches the tree-based reference matrix $\tilde{\mathbf{L}}$.

Treelax algorithm We minimize the Treelax objective function (3.6) using the proximal gradient method. Before we describe the proximal gradient updates, we introduce a couple of notations. Let $\nabla g(\mathbf{L}^{(t)})$ denote the gradient of the smooth part of the objective ($\|\boldsymbol{\Delta} - \left(\boldsymbol{\beta}\mathbf{1}^T + \mathbf{1}\boldsymbol{\beta}^T - 2\mathbf{L}\mathbf{L}^T\right)\|_F^2$) evaluated at $\mathbf{L} = \mathbf{L}^{(t)}$. It can be shown that

$$\nabla g(\mathbf{L}) = 8\left(\mathbf{C} - \operatorname{diag}\left(\mathbf{C}\mathbf{1}\right)\right)\mathbf{L} \tag{3.8}$$

where $\mathbf{C} = \boldsymbol{\Delta} - (\boldsymbol{\beta}\mathbf{1}^T + \mathbf{1}\boldsymbol{\beta}^T - 2\mathbf{L}\mathbf{L}^T)$. And let h denote the sum of the convex non-smooth part of the objective and the non-negativity restriction:

$$h(\mathbf{L}) = \lambda\|\mathbf{L} - \tilde{\mathbf{L}}\|_1 + \infty_{(\mathbf{L}\notin\mathbb{R}_+^{K\times J})}. \tag{3.9}$$

The soft thresholding operator, SoftThreshold$(\mathbf{M};\lambda)$, applies the entry-wise soft thresholding to each entry $M_{i,j}$ with parameter λ, such that $[\text{SoftThreshold}(\mathbf{M};\lambda)]_{i,j} = \text{sign}(M_{i,j})(|M_{i,j}| - \lambda)_+$. Lastly, we define the proximal operator prox_h as

$$\begin{aligned}\text{prox}_h(\mathbf{X}) &= \arg\min_{\mathbf{U}}\left(\frac{1}{2}\|\mathbf{U} - \mathbf{X}\|_F^2 + h(\mathbf{U})\right) && (3.10)\\ &= \left(\text{SoftThreshold}(\mathbf{X} - \tilde{\mathbf{L}};\lambda) + \tilde{\mathbf{L}}\right)_+. && (3.11)\end{aligned}$$

2. From the fact that $\boldsymbol{\Delta}$ can be written as $\boldsymbol{\Delta} = \boldsymbol{\gamma}\mathbf{1}^T + \mathbf{1}\boldsymbol{\gamma}^T - \frac{2}{S}\mathbf{P}^T\mathbf{P}$ where $\boldsymbol{\gamma} = \text{diag}(\frac{1}{S}\mathbf{P}^T\mathbf{P})$, the fidelity term can be interpreted as finding an approximation $(1/S)\mathbf{P}^T\mathbf{P} \approx \mathbf{L}\mathbf{L}^T$, though making the approximation tight in terms of the distances implied by $(1/S)\mathbf{P}^T\mathbf{P}$ and $\mathbf{L}\mathbf{L}^T$.

The proximal gradient method iterates the following update

$$\mathbf{L}^{(t+1)}(\eta) = \operatorname{prox}_{\eta h}\left(\mathbf{L}^{(t)} - \eta \nabla g(\mathbf{L}^{(t)})\right). \tag{3.12}$$

The function h in the proximal operator is scaled by the step size $\eta^{(t)}$ so that, effectively, the tuning parameter λ is scaled by $\eta^{(t)}$. Therefore, the update can be written as

$$\mathbf{L}^{(t+1)}(\eta) = \left(\text{SoftThreshold}\left(\mathbf{L}^{(t)} - \eta \nabla g(\mathbf{L}^{(t)}) - \tilde{\mathbf{L}}; \eta\lambda\right) + \tilde{\mathbf{L}}\right)_{+} \tag{3.13}$$

where $\eta^{(t)}$, the step size at the t-th iteration, is chosen to minimize the objective function at the next iteration $\mathbf{L}^{(t+1)}$

$$\min_{\eta^{(t)}} \mathcal{L}\left(\mathbf{L}^{(t+1)}(\eta^{(t)}); \boldsymbol{\Delta}, \tilde{\mathbf{L}}, \lambda\right). \tag{3.14}$$

We select $\tilde{\mathbf{L}}$ we first estimate a tree topology using the TreeMix method (without any migration events) and then adjust its branch lengths to optimize the fidelity term in (3.6) (with no penalty).

We use cross-validation to choose the optimal penalty parameter that minimizes the mean squared error. Our objective function (3.6) is defined as the sum of the data fidelity term ($\|\boldsymbol{\Delta} - \left(\boldsymbol{\beta}\mathbf{1}^T + \mathbf{1}\boldsymbol{\beta}^T - 2\mathbf{L}\mathbf{L}^T\right)\|_F^2$) and the regularization term ($\lambda\|\mathbf{L} - \tilde{\mathbf{L}}\|_1$). To evaluate the performance of the estimated $\hat{\mathbf{L}}$, we use its data fidelity on the test data. Given a specific value of λ, we obtain the estimate $\hat{\mathbf{L}}$ by minimizing the objective function on the training set $\mathcal{L}(\mathbf{L}; \boldsymbol{\Delta}^{\text{train}}, \tilde{\mathbf{L}}^{\text{train}}, \lambda)$ and compute the mean squared error on the test set as $\|\boldsymbol{\Delta}^{\text{test}} - \left(\hat{\boldsymbol{\beta}}\mathbf{1}^T + \mathbf{1}\hat{\boldsymbol{\beta}}^T - 2\hat{\mathbf{L}}\hat{\mathbf{L}}^T\right)\|_F^2$.

For each training set, we need to compute the solution path $\{\hat{\mathbf{L}}^{(i)}\}_{i=1}^{I}$ for the penalty parameters $\{\lambda_i\}_{i=1}^{I}$. We use the solution $\hat{\mathbf{L}}^{(i)}$ as a warm start in the next problem with the penalty parameter λ_{i+1}.

To account for potential linkage disequilibrium that introduces correlation across nearby loci, we do not randomly partition the rows of the observed allele frequency matrix $\mathbf{P}$. Instead, we split $\mathbf{P}$ into B blocks, denoted as $\mathbf{P}^T = [\mathbf{P}_1^T, \ldots, \mathbf{P}_B^T]^T$, assuming the SNPs are ordered according to their positions in the genome. For each training and test dataset, we compute the corresponding target matrices using equation (3.4).

3.2.4 Adding a shared drift component

Given any matrix $\mathbf{L}$ such that $\frac{1}{S}\mathbf{P}^T\mathbf{P} \approx \mathbf{L}\mathbf{L}^T$ a simple way to consider improving the approximation is by adding a column to L that is constant; that is, by updating $\mathbf{L} \leftarrow [\mu_0\mathbf{1}, \mathbf{L}]$ for some scalar μ_0. The optimal approximation is given by setting

$$\mu_0 := \arg\min_{\mu_0} \|\frac{1}{S}\mathbf{P}^T\mathbf{P} - \mu_0^2\mathbf{1}\mathbf{1}^T - \mathbf{L}\mathbf{L}^T\|_F^2 = \max(0, \sqrt{\mathrm{mean}(\frac{1}{S}\mathbf{P}^T\mathbf{P} - \mathbf{L}\mathbf{L}^T)}). \qquad (3.15)$$

When $\mathbf{L}$ is derived from a tree, adding this constant column to $\mathbf{L}$ can be thought of as adding a branch of length μ_0 at the top of the tree (joining to the root) that is ancestral to all populations.

Note that adding this constant column to $\mathbf{L}$ results in a similar update to $\mathbf{M}$: $\mathbf{M} \leftarrow \mathbf{Q}[\mu_0\mathbf{1}, \mathbf{L}] = [\mu_0\mathbf{1}, \mathbf{Q}\mathbf{L}]$ (because $\mathbf{Q}\mathbf{1} = \mathbf{1}$). Adding this constant column does not change the pairwise distances implied by $\mathbf{M}$ (right hand side of (3.5)), and so does not help explain any genetic differences among individuals; however, it may be helpful for providing context for the sizes of genetic differences in the sample relative to the shared component (e.g. see Figure 3.9).

3.2.5 Computing individual-specific drift components

The strategies above yields a matrix $\mathbf{M}$ that satisfies $(1/S)\mathbf{G}^T\mathbf{G} \approx 4\mathbf{M}\mathbf{M}^T$. Focusing on the diagonal elements of these matrices we can write

$$\frac{1}{S}\|\mathbf{g}_i\|^2 = \sum_{j=0}^{J} 4M_{i,j}^2 + e_i \tag{3.16}$$

where $e_i := \frac{1}{S}\|\mathbf{g}_i\|^2 - \sum_{j=0}^{J} 4M_{i,j}^2$, and $j = 0$ is used to index the constant column added to $\mathbf{M}$ above (ie $M_{i,0} = \mu_0$).

3.2.6 A decomposition of genetic distance

Up to now we have made no assumptions about how genotypes are coded. However, if we assume that genotypes are coded with 0 representing the ancestral allele at each site then (3.16) has a particularly nice interpretation. In the case of this encoding, $\frac{1}{S}\|\mathbf{g}_i\|^2$ is the squared genetic distance of individual i's genotype to the ancestral genotype (0). Thus Equation (3.16) provides a decomposition of individual i's (squared) genetic distance from the ancestor into a sum of components that are shared with others in the sample, plus a residual component e_i that is specific to individual i. (Although, as far as we are aware, there is no mathematical guarantee that e_i is non-negative, in practical applications we have looked at this is always the case.)

3.2.7 Contrast with Tree-and-migrations-based estimation of genetic drifts

One widely studied approach to relaxing the tree assumption is to explicitly model migration events by adding migration edges on top of a tree structure. Examples of such methods include TreeMix [Pickrell and Pritchard, 2012], ADMIXTOOLS [Patterson et al., 2012], and MixMapper [Lipson et al., 2013]. These methods take individuals' genotypes and pre-defined population labels as input and model the allele frequencies of these populations using a tree-like structure with additional migration events.

This contrasts with STRUCTURE-type methods, which require only individual genotypes and infer their (potentially admixed) population memberships. Since STRUCTURE addresses admixture through a separate step of inferring population memberships (the optimal number of ancestral populations being a well-studied problem), the DRIFT approach doesn't necessarily benefit from the tree-and-migration type relaxation of the treeness assumption in the genetic drift estimation step.

In other words, when the true underlying population history involves an approximate tree structure with gene flow between populations, the tree-and-migration-type methods are not well-suited to represent these scenarios, as we demonstrate in our simulation studies.

3.3 Verification and comparison

3.3.1 Four populations out of Africa simulation using stdpopsim

We use simulation studies to compare our Treelax method with the tree estimation method and the TreeMix method [Pickrell and Pritchard, 2012]. These methods are compared in the setting where they are used jointly with the ADMIXTURE method [Alexander et al., 2009] in the DRIFT framework.

For the simulation setting, we follow the four populations out of Africa demographic model [Jouganous et al., 2017], implemented as a part of the stdpopsim [Adrion et al., 2020, Lauterbur et al., 2023] catalog[3]. In this model, there are four populations: YRI (Yoruba), CEU (Utah Residents with Northern and Western European Ancestry), CHB (Han Chinese in Beijing, China), and JPT (Japanese in Tokyo, Japan). The Out-of-Africa (OOA) event is modeled at 119 thousand years ago (kya), the CEU-CHB split at 46 kya, and the CHB-JPT

3. https://popsim-consortium.github.io/stdpopsim-docs/stable/catalog.html.

split at 9 kya. The model is illustrated in Figure 3.3, which is obtained from the stdpopsim catalog.

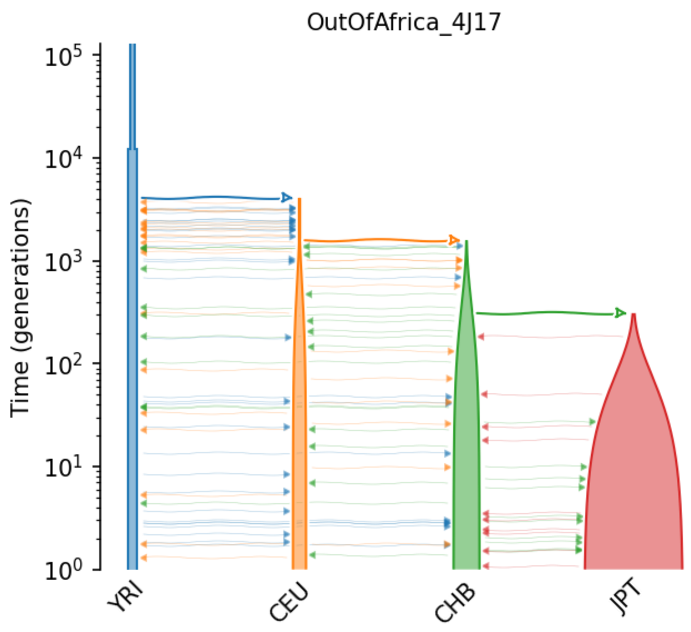


Figure 3.3: An illustration of the 'four populations out of Africa' model [Jouganous et al., 2017], obtained from the stdpopsim catalog.

After the Out-of-Africa event, there is a continuous migration between YRI and OOA with the migration rate 16.8e-5 (fraction per generation), which lasts for 73 thousand years. After the CEU-CHB split, the YRI-CEU migration rate falls to 1.14e-5 and the YRI-CHB migration rate to 0.56e-5; the CEU-CHB migration rate is set as 4.75e-5. Lastly, after the CHB-JPT split, the CHB-JPT migration rate is set as 3.3e-5. Note that within this demographic model, JPT interacts only with CHB after formation, not CEU or YRI.

Our sample consists of 400 individuals: 100 each from YRI, CEU, CHB, and JPT populations. All individuals are sampled at time = 0. For this sample, we simulate chromosome 1 (248,956,422 bps per GRCh38) with recombination rate 1.15e-8. The mutation rate is 1.44e-8 per base per generation where a generation is taken to be 29 years.

The simulated data set contains 2,792,638 variants. We use plink2.0 [Chang et al., 2015] to filter genetic variants to be biallelic, at least 2 kilobases (kb) apart, and with a minor allele frequency (MAF) greater than or equal to 0.01. The filtered dataset contains 108,963 SNPs.

3.3.2 Simulation results

STRUCTURE bar plot We first run ADMIXTURE [Alexander et al., 2009] on the filtered dataset, specifying four ancestral populations ($K = 4$). The estimated population membership of individuals is shown as a STRUCTURE bar plot in Figure 3.4. The ADMIXTURE successfully recovers the population membership of the individuals: four estimated latent populations P1, P2, P3, and P4 almost exactly correspond to the true populations YRI, CEU, CHB, and JPT. There are some individuals from CHB (or JPT) that also have a partial membership in the JPT-majority latent population P4 (or the CHB-majority latent population P3), reflecting the relatively small genetic difference between the two populations.

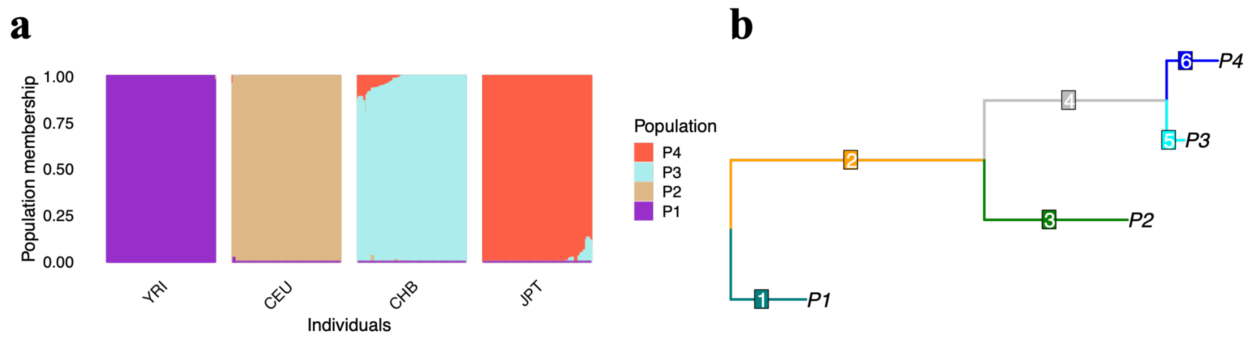


Figure 3.4: (a) STRUCTURE bar plot and (b) estimated tree in simulation.

Tree From the population allele frequencies (the **P** matrix obtained from the ADMIXTURE step), we estimate the tree structure. We first use the TreeMix method with no migration edges to estimate a tree. To root the tree, we use an outgroup population – a population assumed to have no derived alleles. Then, we adjust the branch lengths to make the distance approximation tight by minimizing (3.6) with penalty parameter $\lambda = 0$.

The topology of the estimated tree (Figure 3.4b) correctly reflects the order of founding events in the simulations. Further, the lengths of the inferred tree branches (which correspond to sizes of drift components) are consistent the effective population sizes in the simulations: for example, the drift component D1 experienced by YRI is much smaller than the ‘Out-of-Africa’ drift, D2, as expected from the simulation setting in which the YRI population size (23,721) is substantially larger than the OOA population size (2,831).

Tree as genetic distance decomposition The six drifts, estimated from the tree method, can be interpreted as six additive explained genetic distances. Each drift divides the populations into two groups: one with the drift and the other without the drift. And the drift size squared is the induced distance between the two groups. These explained distances are additive in the sense that the total explained distance between two populations is the sum of the explained distances between the two population for all drifts. Figure 3.5 illustrates the decomposition of the genetic distance, based on the tree-based drifts. For example, the distance between P1 and P2, 0.032660 is decomposed as 0.002004 (D1) + 0.023083 (D2) + 0.007573 (D3) = 0.032660 in the estimated tree.

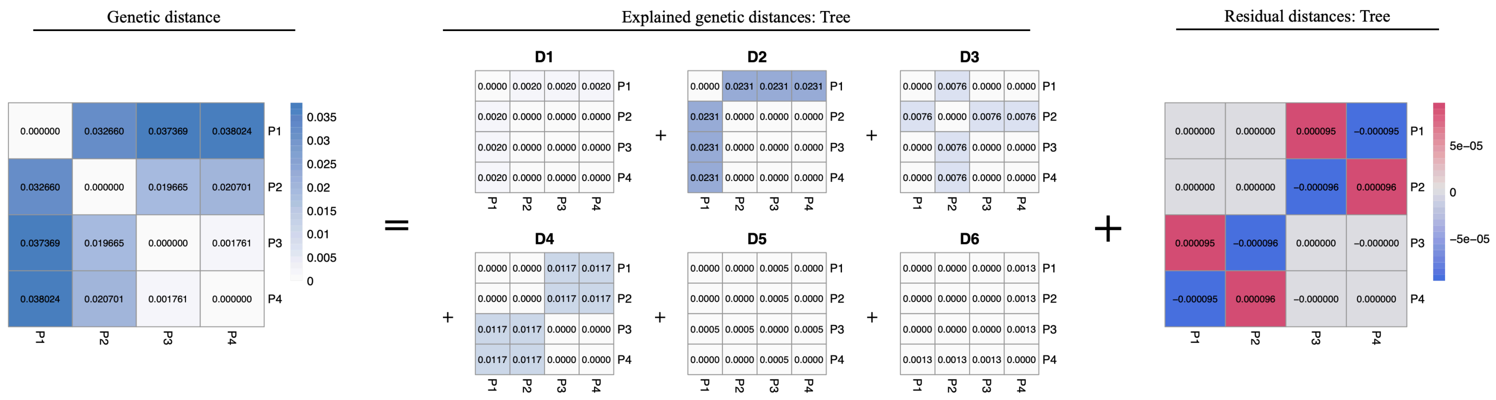


Figure 3.5: Genetic distance, its decomposition, and residual by tree method, in simulation.

Treeness violations: Residual distances Since a tree can be interpreted as providing a decomposition of genetic distance, the residual distances, left unexplained by the tree, can be interpreted as the treeness violation of the given genetic distance. In Figure 3.5, we can see that P1-P3 and P2-P4 distances are under-predicted (i.e., predicted to be closer than is observed), in the estimated tree, whereas P1-P4 and P2-P3 distances are over-predicted (i.e., predicted to be farther than is observed).

This observed pattern is consistent with the pattern of continuous gene flow under the simulation setting. After the CHB-JPT split (now the four populations are formed), CHB-CEU migration rate is 4.75e-5 and CHB-YRI migration rate is 0.56e-5 whereas JPT do not have any migration with CEU or YRI. As a result, CHB is closer to CEU than what is expected in a tree structure without any migration and, to a much lesser extent, closer to YRI. To accommodate this treeness deviation, the size of drift D4, corresponding to the P1/P2 vs P3/P4(YRI/CEU vs CHB/JPT) divergence, is adjusted to be shorter; the size of drifts D1 and D2, corresponding to the P1 vs P2/P3/P4 (YRI vs CEU/CHB/JPT) divergence, is adjusted to be longer and the size of drift D3, corresponding to the P2 vs P1/P3/P4 (CEU vs YRI/CHB/JPT) divergence, shorter; the size of drift D5, corresponding to the P3 vs P1/P2/P4 (CHB vs YRI/CEU/JPT) divergence, is adjusted to be shorter and the size of drift D6, corresponding to the P4 vs P1/P2/P3 (JPT vs YRI/CEU/CHB) divergence, longer. These adjustments result in the residual pattern observed in Figure 3.5. Further detailed explanation of these adjustments is provided in Appendix 3.6.

Treelax The Treelax method relaxes the strict assumption that the data follows a perfect tree-like structure. It achieves this by allowing for deviations from this structure, adjusting the level of deviation adaptively using cross validation.

Table 3.1 reports the populations' drift memberships estimated from the tree method and the Treelax method. The differences due to the tree relaxation is also reported. Population P2's D2 membership increased by 0.2% and P3's D2 membership decreased by 0.4%; P3 added a 0.6% of D3 membership; P4's D4 membership increased by 0.4%.

For P3, its increased D2 membership increases the fitted P1-P3 and P2-P3 distances,

Table 3.1: Populations' drift memberships in (A) tree method and (B) Treelax method, and (C) membership differences in the two methods, in simulation.

(A) Tree

	D1	D2	D3	D4	D5	D6
P1	0.044764	0	0	0	0	0
P2	0	0.151930	0.087024	0	0	0
P3	0	0.151930	0	0.108301	0.021402	0
P4	0	0.151930	0	0.108301	0	0.036104

(B) Treelax

	D1	D2	D3	D4	D5	D6
P1	0.044764	0	0	0	0	0
P2	0	0.151930	0.087024	0	0	0
P3	0	0.152228	0.000504	0.108301	0.021402	0
P4	0	0.151354	0	0.108688	0	0.036104

(C) Difference (Treelax - Tree)

	D1	D2	D3	D4	D5	D6
P1	0	0	0	0	0	0
P2	0	0	0	0	0	0
P3	0	0.000298	0.000504	0	0	0
P4	0	-0.000576	0	0.000387	0	0

and its nonzero D3 membership increases the fitted P1-P3 distance and decreases the fitted P2-P3 distance. With these two effects combined, the P1-P3 distance becomes much less under-predicted (from 0.000095 to 0.000005) and the P2-P3 distance becomes much less over-predicted (from -0.000096 to -0.000008).

Similarly, for P4, its decreased D2 membership decreases the fitted P1-P4 distance and increases the fitted P2-P4 distance, and its increased D4 membership increases the fitted P1-P4 and P2-P4 distances. With these two effects combined, the P1-P4 distance becomes much less over-predicted (from -0.000095 to -0.000005) and the P2-P4 distance becomes much less under-predicted (from 0.000096 to 0.000011).

The Treelax-based decomposition of the genetic distances and the residual distances are shown in Figure 3.6. The Frobenius norm of the residual distance matrix decreased 92%, from 0.000270 to 0.000022.

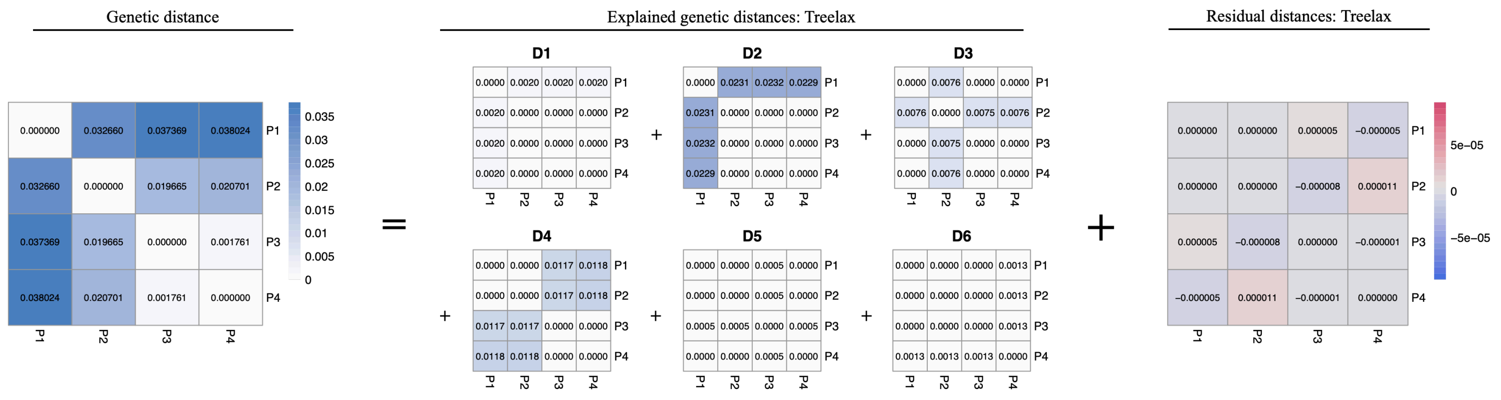


Figure 3.6: Genetic distance, its decomposition, and residual by Treelax method, in simulation.

TreeMix The TreeMix results with zero, one, and two migration events are given in Figure 3.7. The estimated tree is almost identical to the one previously shown in Figure 3.4. (The "drift parameter" in the TreeMix graph corresponds to the squared value of our drift size s_j, making the drift sizes appear more extreme.)

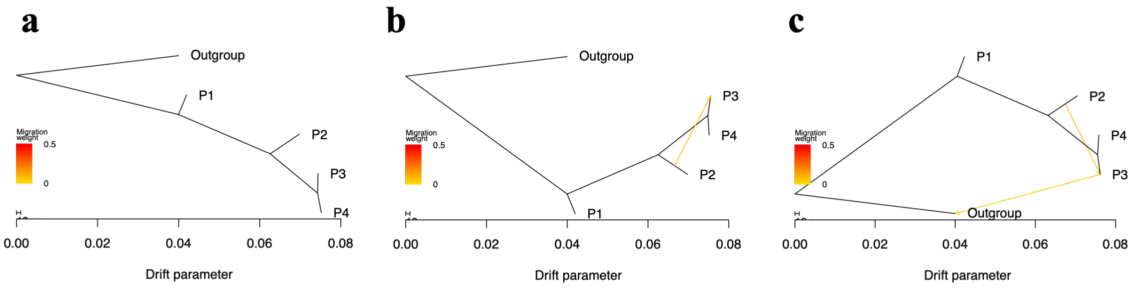


Figure 3.7: TreeMix results with (a) no migration (tree), (b) one migration, and (c) two migrations, in simulation.

When estimating one migration, the TreeMix identifies a migration event from a population between the P2/P3/P4-common ancestor and the population P2 to the population P3. This migration is capturing the most noticeable feature of the treeness violation that the P2-P3 distance is closer than expected from a tree structure.

When we allow TreeMix to estimate two migration edges, it adds a second migration edge from P3 to the outgroup. This seems not to be easily interpretable in terms of the simulation truth.

DRIFT bar plots We visualize populations' drift memberships and individuals' drift memberships as stacked bar plots in Figure 3.8. The upper panel shows the result for tree-based estimation and the lower panel for Treelax-based estimation; the two are visually almost identical, reflecting the fact that in this simulation Treelax results in only very small changes in **L** (changes ranged from 0.2% to 0.6%).

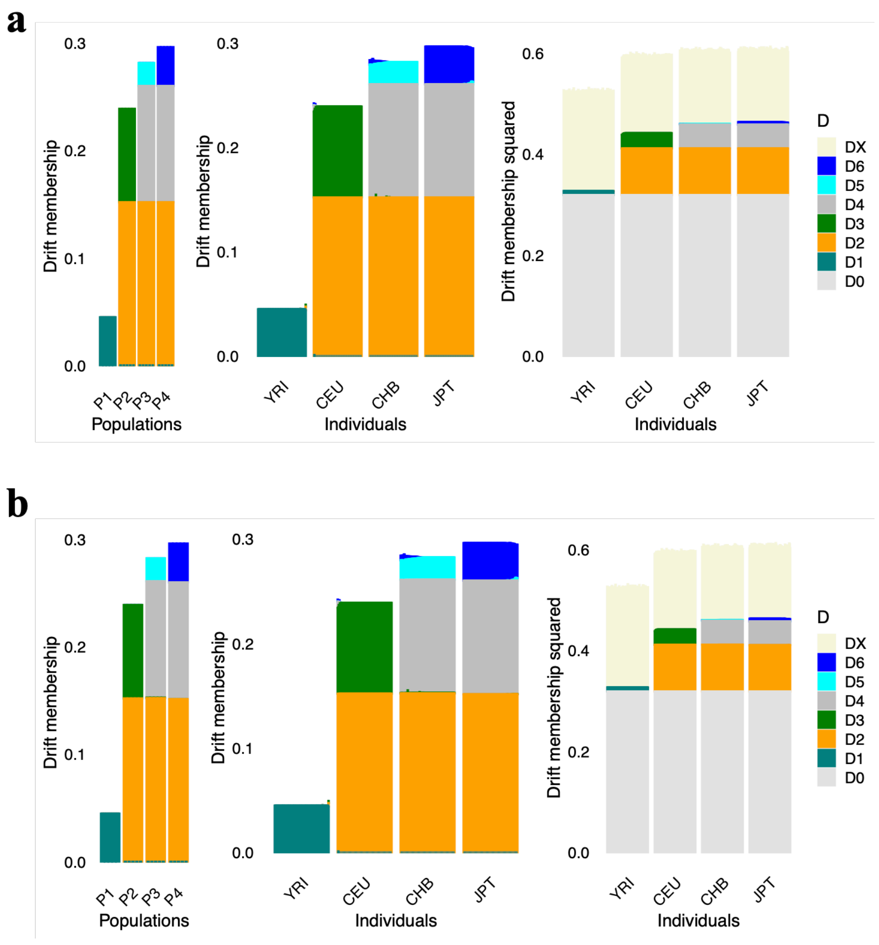


Figure 3.8: (a) Tree-based and (b) Treelax-based DRIFT bar plots in simulation.

In the population DRIFT bar plots (left plots of Figure 3.8), each population's drift memberships are stacked along the y-axis and populations are positioned along the x-axis.

In the individual DRIFT bar plots (center plots of Figure 3.8), each individual's drift memberships are stacked along the y-axis and individuals are positioned along the x-axis. Since all YRI (or CEU) individuals have memberships only in the latent population P1 (or P2), their stacked drift memberships are almost identical to the stacked drift memberships of P1 (or P2). In contrast, there are some CHB and JPT individuals who have admixed population memberships in P3 and P4, who thus have admixed drift memberships in D5 and D6.

To gain insight into the amount of genetic variation that is explained by the drift components, we show stacked bar plots that include both the shared drift component and the individual-specific residual components (rightmost plots of Figure 3.8). The drift component 'D0' is the shared component of genetic drift, , and the drift components 'DX' are the individual-specific drift components, or residuals (e_i's), that are left unexplained by the model. In these plots we use the *squared* drift memberships ($\{4M_{i,j}^2\}_{j=0}^J$) for the bar heights, so that the total height of individual i's stacked bars is $\frac{1}{S}\|\mathbf{g}_i\|^2$ (by (3.16)). Further, we coded genotypes with 0 representing the ancestral allele, so that $\frac{1}{S}\|\mathbf{g}_i\|^2$ is the squared genetic distance of individual i's genotype to the ancestral genotype (0).

More than half of individuals' genetic variations are shared across all individuals in the sample, through the common drift D0. Of the remaining genetic variations, less than half is explained by the drift-based shared genetic variations (D1 through D6); individuals' uniqueness (DX) plays a greater role.

The total bar heights for each individual (which are $\frac{1}{S}\|\mathbf{g}_i\|^2$) are similar within each population but differ somewhat across populations. Two main reasons are squared L2 norm measurement and SNP filtering. Based on the fact that individuals are the same distance from the common ancestor, we expect each individual to hold roughly same number of derived alleles. However, the squared L2 norm measurement of genetic variation assigns a higher number to individuals whose genotype has more 2s, than individuals whose genotype has fewer 2s and more 1s. For example, an individual with genotype (2,0,1) at three SNPs has squared L2 norm 5, whereas another individual with genotype (1,1,1) has squared L2 norm 3, while they both have three derived alleles. Our simulation shows YRI experiencing the least genetic drift, resulting in the greatest genetic diversity. Consequently, two chromosomes in YRI are less likely to share a derived allele compared to CEU, CHB, or JPT. Therefore, YRI individuals are expected to have more 1 derived alleles and less 2 derived alleles, hence smaller L2 norm measurement of genetic variation.

The second reason is filtering SNP with minor allele frequency thresholding. Since YRI is more genetically diverse than the other populations, YRI's minor allele frequencies are expected to be concentrated on smaller values, which are more likely to be filtered out. Also, YRI sample size (100) is small compared to the OOA sample size (300). Therefore, the filtering can understate YRI's genetic variability.

3.4 Applications

3.4.1 Homo Sapiens: The 1000 Genomes Project data

Data The 1000 Genomes Project [The 1000 Genomes Project Consortium, 2010, 2015] is a global collaboration aimed at creating a comprehensive public catalog of human genetic variation across populations, which plays a foundational role in studying the genotype-phenotype relationship. The Project's data includes 2,504 individuals sampled from 26 populations, representing five super populations: Africa, Europe, South Asia, East Asia, and the Americas[4]. Each individual was sequenced using low-coverage whole-genome sequencing, deep exome sequencing, and dense microarray genotyping. In total, the Consortium reported over 88 million genetic variants, including 84.7 million single nucleotide polymorphisms (SNPs).

Genetic variants are not distributed randomly among individuals and populations. For

4. The super population and population definitions are provided by The 1000 Genomes Project. The purpose of this work is to study and interpret patterns of population structure, and as such, descriptors based on ethnicity and geography are used.

example, the Consortium found that "rarer variants are typically restricted to closely related populations" whereas "most common variants are shared across the world". Even the common variants, though shared across the world, can exhibit varying frequencies across populations. To illustrate the pattern of genetic variant sharing across individuals, the Consortium estimated population structure based on data after filtering out variants with a minor allele frequency of less than 0.05. The result is reported as a STRUCTURE bar plot, as shown in Figure 3.1a. The estimated population structure, using only common variants, successfully reveals genetic similarities across individuals and populations.

To focus on the methods for genetic drift estimation, we make use of the post-filtering dataset, which includes 193,634 SNPs and 2,504 individuals, and the ADMIXTURE [Alexander et al., 2009] outputs, provided by the Consortium. We analyze the $185,116$ SNPs after dropping the SNPs without the ancestral allele information in the data[5]. Using the ancestral allele, we change the coding of the genotype matrix so that the allele counts are coded as the count of derived alleles.

Results The STRUCTURE bar plot with eight latent populations in The 1000 Genomes Project Consortium [2015] is replicated in Figure 3.9a.[6] The African populations have high memberships in the populations P1 and P2, the European populations in P3 and P4, the South Asian populations in P5, the East Asian populations in P6 and P7, and the American populations in P1, P3, and P8.

Figure 3.9b shows the estimated tree structure of the eight latent populations based on their population allele frequencies. We used an ancestral population with ancestral alleles as

5. The ancestral allele entry in the data is derived from the EPO alignments, which infer ancestral alleles using human, chimp, orangutan, rhesus macaque sequences. For a more comprehensive discussion of ancestral allele inference, refer to Section 8.3 of the supplementary materials in The 1000 Genomes Project Consortium [2015].

6. For a comprehensive overview, Figure 3.9 replicates previous results: (a) STRUCTURE bar plot (identical to Figure 3.1a), (b) tree-based genetic drift estimation (identical to Figure 3.1b), (c) tree-based individual DRIFT bar plot (identical to Figure 3.1c).

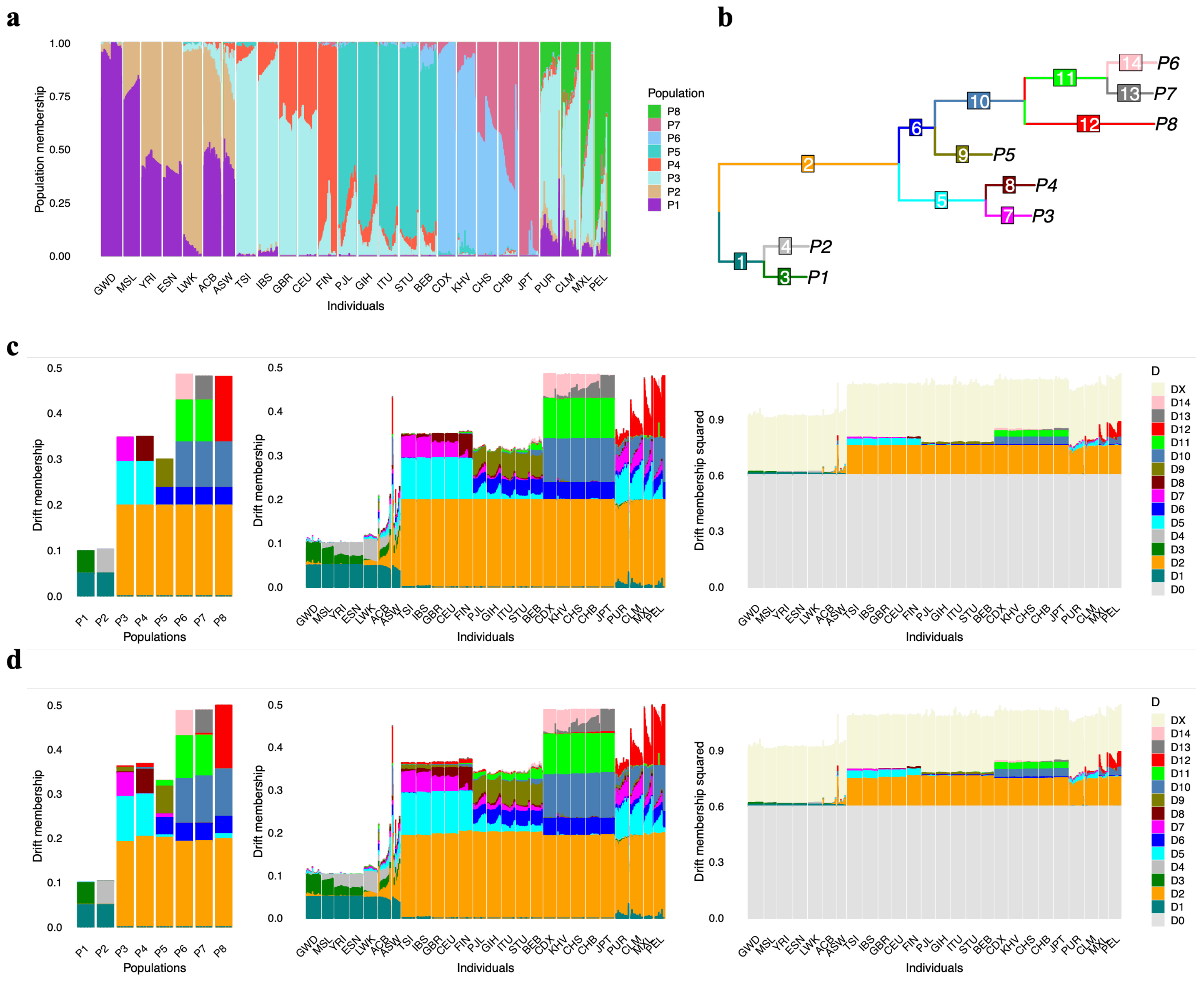


Figure 3.9: Analysis of the 1000 Genomes Project Data. (a) STRUCTURE bar plot, (b) tree-based genetic drift estimation, and (c) tree-based and (d) Treelax-based DRIFT bar plots.

the outgroup to find the root of the tree. Each branch has a label that corresponds to the drift factor. The majorly African populations P1 and P2 have the smallest drift from the root of the tree; majorly South Asian population P5, the majorly European populations P3 and P4, the majorly East Asian populations P6 and P7, and the majorly American population

P8 have increasingly larger drift sizes. This increasing drift size pattern is consistent with the pattern reported in Li et al. [2008] and Pickrell and Pritchard [2012] who analyzed independent data of 938 individuals from 51 populations (Human Genome Diversity Panel). The larger drift in non-African populations is evidence of the bottleneck effect that these populations went through, which reduced the genetic diversity within non-African populations (more shared genetic components which are identified as a greater drift size). The effective population size inferred using PSMC, reported in the original publication of the data [The 1000 Genomes Project Consortium, 2015] confirms the bottleneck effect hypothesis.

Figure 3.9c illustrates populations' and individuals' drift memberships, based on the tree-based method. In the leftmost plot, the population DRIFT bar plot, populations are aligned along the x-axis and their drift memberships are stacked along the y-axis. The drift memberships are color-coded and labeled. In the middle plot, the individual DRIFT bar plot, individuals are aligned along the x-axis and their drift memberships, computed by taking weighted averages of populations' drift memberships, are stacked along the y-axis.

A key feature of the DRIFT representation is that it provides a detailed multi-resolution representation of population structure, which highlights the shared evolutionary history of the human genome. For example, the pan-African drift factor D1 is shared across all the African individuals and the 'out-of-Africa' drift factor D2 across all the non-African individuals. Furthermore, individuals in Southern Europe (TSI and IBS) and the Americas are estimated to have partial memberships in the pan-African drift D1 (via their P1 membership), and individuals in African American populations (ACB, ASW) are estimated to have partial memberships in the Out-of-Africa drift D2 (via their P3 membership). This scope of shared components is much greater than the scope of population-based genetic sharing in the STRUCTURE representation, which is typically restricted to a subset of super populations. At a finer level, there are population-specific factors, such as the 'South Asian' drift factor D9, which is specific to the latent population P5, and thus almost exclusively found in South

Asian individuals.

In the rightmost plot in Figure 3.9c, the squared individual DRIFT bar plot, individuals are aligned along the x-axis and their squared drift memberships are stacked along the y-axis. Similar to the results reported in the simulation, more than half of the total genetic variation is explained by the common genetic drift D0 and more than half of the remaining genetic variation is attributable to individual-specific residuals (DX).

Alternatively, we can relax the treeness assumption to obtain a better fit. Treelax-based populations' drift memberships and DRIFT bar plots are provided in Table 3.2 and Figure 3.9d.

Table 3.2: Populations' drift memberships in (A) tree method and (B) Treelax method, and (C) membership differences in the two methods, in the 1000 Genomes Project data analysis.

(A) Tree

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14
P1	0.0495	0	0.0488	0	0	0	0	0	0	0	0	0	0	0
P2	0.0495	0	0	0.0518	0	0	0	0	0	0	0	0	0	0
P3	0	0.1981	0	0	0.0956	0	0.0535	0	0	0	0	0	0	0
P4	0	0.1981	0	0	0.0956	0	0	0.0547	0	0	0	0	0	0
P5	0	0.1981	0	0	0	0.0387	0	0	0.0624	0	0	0	0	0
P6	0	0.1981	0	0	0	0.0387	0	0	0	0.0998	0.0919	0	0	0.0566
P7	0	0.1981	0	0	0	0.0387	0	0	0	0.0998	0.0919	0	0.0524	0
P8	0	0.1981	0	0	0	0.0387	0	0	0	0.0998	0	0.1437	0	0

(B) Treelax

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14
P1	0.0495	0.0007	0.0488	0	0.0011	0	0	0	0	0	0	0	0	0
P2	0.0495	0.0007	0	0.0518	0	0	0	0	0	0.0003	0.0007	0	0	0
P3	0	0.1914	0	0	0.1018	0	0.0535	0.0021	0.0095	0	0	0.0035	0	0
P4	0	0.2031	0	0	0.0956	0	0.0008	0.0547	0	0.0039	0	0.0090	0.0008	0
P5	0	0.2010	0	0	0.0056	0.0387	0.0087	0	0.0625	0	0.0129	0	0	0.0008
P6	0	0.1918	0	0	0	0.0406	0	0	0.0004	0.1013	0.0962	0	0	0.0566
P7	0	0.1936	0	0	0	0.0387	0	0.0007	0	0.1064	0.0919	0.0041	0.0524	0.0006
P8	0	0.1981	0	0	0.0115	0.0387	0	0	0	0.1069	0	0.1437	0	0

(C) Difference (Treelax - Tree)

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14
P1	0	0.0007	0	0	0.0011	0	0	0	0	0	0	0	0	0
P2	0	0.0007	0	0	0	0	0	0	0	0.0003	0.0007	0	0	0
P3	0	-0.0068	0	0	0.0062	0	0	0.0021	0.0095	0	0	0.0035	0	0
P4	0	0.0050	0	0	0	0	0.0008	0	0	0.0039	0	0.0090	0.0008	0
P5	0	0.0028	0	0	0.0056	0	0.0087	0	0.0001	0	0.0129	0	0	0.0008
P6	0	-0.0064	0	0	0	0.0019	0	0	0.0004	0.0015	0.0043	0	0	0
P7	0	-0.0046	0	0	0	0	0	0.0007	0	0.0066	0	0.0041	0	0.0006
P8	0	0	0	0	0.0115	0	0	0	0	0.0071	0	0	0	0

The five largest differences in the Treelax-based versus tree-based populations' drift memberships are:

- P5's D11 membership (+0.0129; South Asian-East Asian gene flow),
- P8's D5 membership (+0.0115; American-European gene flow),
- P3's D9 membership (+0.0095; Southern/Central European-South Asian gene flow),
- P4's D12 membership (+0.0090; Northern European-American gene flow),
- P5's D7 membership (+0.0087; South Asian-Southern/Central European gene flow).

These gene flows can be explained by geographical closeness (South Asian-East Asian populations) or historical connections (American-European populations and Southern/Central European-South Asian populations).

The gene flow between Northern Europeans and Americans is particularly notable. Latent population P4, genetically closest to the Finnish population, exhibits membership in the American-specific drift component (D12). This finding implies a historical gene flow, consistent with previous studies highlighting the genetic similarities between Northern Eurasians and Americans [Zerjal et al., 1997, Huyghe et al., 2011, Lamnidis et al., 2018].

The Frobenius norm of the populations' residual distance matrix decreased by 99.4%, from 0.011286 to 0.000065. This substantial reduction suggests a much better fit of the Treelax model to the data compared to the tree-based approach (although we do not have a formal approach to assessing this).

Treelax-based populations' drift memberships are visualized as the population DRIFT bar plot in the left plot of Figure 3.9d. Treelax-based individual DRIFT bar plot, shown in the middle plot of Figure 3.9d, illustrates the effect of relaxing the treeness assumption at individual level. The corresponding squared individual DRIFT bar plot is shown in the right plot of Figure 3.9d.

3.5 Discussion

3.5.1 Benefits of a modular approach

We adopted a three-step modular approach to obtain a DRIFT representation, which involves combining a STRUCTURE-type method with a genetic drift estimation method, instead of directly estimating a DRIFT representation. This modularity offers the following three benefits.

Firstly, it provides flexibility, allowing users to combine any existing methods to obtain a drift-based representation. For instance, users can use a STRUCTURE-type method designed for large-scale studies (such as SCOPE [Chiu et al., 2022]) to achieve scalability. (The genetic drift estimation step tends to be computationally cheaper than the STRUCTURE step since the population allele frequency matrix has only K columns, which is much smaller than the number of individuals in the sample.)

Secondly, there is a reduction in the complexity of individuals' drift memberships. Individuals are limited to certain combinations of drift memberships, resulting in a rank-deficient $\mathbf{M}$ matrix (of individuals' drift memberships) since the number of populations is smaller than the number of drifts.

Lastly, this approach offers modeling advantages. The STRUCTURE step effectively denoises the data by soft-clustering individuals, thereby facilitating accurate estimation of genetic drifts. Additionally, it could be easier to model the population allele frequency matrix (with entries ranging from 0 to 1) rather than the genotype matrix (with entries taking discrete values of 0, 1, or 2).

3.5.2 Role of ancestral alleles and outgroup

Ancestral allele information serves two key purposes in our analysis: (i) it allows us to establish an outgroup for rooting the evolutionary tree, and (ii) it enables the decomposition of individuals' total genetic variation by comparing them to the ancestral state. In cases where ancestral allele information is unavailable or its utilization results in the exclusion of a significant portion of the data, alternative approaches may be employed.

Even in instances where ancestral allele information and outgroup data are unavailable, the DRIFT method remains applicable. However, this method loses information regarding the directionality of genetic drift events, and it precludes the construction of squared individual DRIFT bar plots. Instead, this alternative approach involves estimating an unrooted tree structure using a tree estimation technique. Subsequently, the determination of the root location becomes necessary through external information or an ad-hoc approach.

For instance, in Figure 3.1b, estimating an unrooted tree would combine drift factors D1 and D2 (representing the genetic divergence between African and non-African populations) into a single long branch. This necessitates subsequently determining the root location, which represents the point of divergence. While this approach loses information about the precise direction of divergence, it may suffice for analyses aimed at understanding the overall genetic similarity within the studied samples.

With outgroup information, we can estimate a rooted tree, revealing the directionality of evolutionary changes. However, the absence of ancestral allele information prevents us from quantifying the extent to which each individual's genetic makeup diverges from the ancestral state. This deficiency arises because we are unable to distinguish between ancestral and derived alleles, a prerequisite for decomposing individuals' total genetic variation from the ancestral state.

3.6 Appendix: Branch length adjustments for treeness violations

3.6.1 Problem setup

Suppose there are four populations P1, P2, P3, and P4, structured as in the Figure 3.4 panel b. Considering an unrooted tree, the combined size of drifts D1 and D2 is identifiable, but not the separate size of D1 and D2; therefore, we combine the branches and call it D1/2. We add a simplifying assumption that the sum of the sizes of D1/2 and D3 is fixed to the observed distance between P1 and P2, and that the sum of the sizes of D5 and D6 is fixed to the observed distance between P3 and P4. Note that this simplifying assumption is not enforced in the tree estimation process that produced the estimate illustrated in Figure 3.4, but is consistent with the estimated result illustrated in Figure 3.5.

The problem is to minimize the Frobenius norm of the residual distance matrix $\mathbf{R} \in \mathbb{R}^{2\times 2}$ where its four entries are the pairwise distances between [P1,P2] × [P3, P4]:

$$\mathbf{R} = \begin{bmatrix} d(P1, P3) & d(P1, P4) \\ d(P2, P3) & d(P2, P4) \end{bmatrix}$$

Note that the problem of minimizing the Frobenius norm of the full 4×4 residual matrix can be reduced to the above problem because i) the diagonal entries are zero, ii) the matrix is symmetric, and iii) $d(P1, P2) = 0$ and $d(P3, P4) = 0$ due to the simplifying assumption.

To reflect the simulation setting in which P3 has a migration with P2 and with P1 to a lesser degree, we set the initial value of the residual distance matrix as

$$\mathbf{R}^{(0)} = \begin{bmatrix} -20 & 0 \\ -100 & 0 \end{bmatrix}$$

meaning that the P1-P3 distance is overestimated by 20 and the P2-P3 distance by 100.

3.6.2 Allowed operations

We define three operations that can be used to solve the residual minimization problem. The first operation O1 is to shorten the size of D4 by α so that all the residuals increase by α.

$$\mathbf{R} \to \mathbf{R} + \begin{bmatrix} +\alpha & +\alpha \\ +\alpha & +\alpha \end{bmatrix}$$

The second operations O2 is to shorten the size of D1/2 and lengthen the size of D3 by β

$$\mathbf{R} \to \mathbf{R} + \begin{bmatrix} +\beta & +\beta \\ -\beta & -\beta \end{bmatrix}$$

and the third operations O3 is to shorten the size of D5 and lengthen the size of D6 by γ

$$\mathbf{R} \to \mathbf{R} + \begin{bmatrix} +\gamma & -\gamma \\ +\gamma & -\gamma \end{bmatrix}$$

3.6.3 Solution

We can obtain the closed-form solution by considering optimal operations sequentially. First, the optimal O1 operation is to set $\alpha^* = +30$. In a general case, the solution to the problem $\min_\alpha \sum_{i=1}^n (x_i + \alpha)^2$ can be obtained using its first order condition: $\sum_{i=1}^n (x_i + \alpha^*) = 0$, which ensures that the post-shifting mean $(\frac{1}{n} \sum_{i=1}^n (x_i + \alpha^*))$ is zero. Combine this observation with the property of O2 and O3 that they do not change the sum of four distances, we get the optimal O1 operation as $\alpha^* = -\frac{1}{4}(-20 - 100 + 0 + 0) = +30$. Let's call the mean-shifted residual distance matrix as

$$\mathbf{R}^{(1)} = \begin{bmatrix} +10 & +30 \\ -70 & +30 \end{bmatrix}.$$

Next, we consider applying operations O2 and O3 to $\mathbf{R}^{(1)}$. We can use the property of O2 and O3 that $d(P1, P3) + d(P2, P4)$ and $d(P2, P3) + d(P1, P4)$ are invariant to these operations. Again, this problem can be readily solved by considering a general case: the solution to the problem $\min_x x^2 + (C - x)^2$ is obtained as $x = C/2$, which makes $x = C - x = C/2$. Applying this idea to our problem, we have $d(P1, P3) = d(P2, P4) = +20$ and $d(P2, P3) = d(P1, P4) = -20$, which can be obtained by setting $\beta^* = -20$ and $\gamma^* = +30$ for operations O2 and O3.

To summarize, our solution to the problem is

$$\mathbf{R}^* = \begin{bmatrix} +20 & -20 \\ -20 & +20 \end{bmatrix},$$

which is consistent with the pattern observed in the residual distance matrix in Figure 3.5. This solution can be obtained by setting $\alpha^* = +30$ (shortening the size of D4), $\beta^* = -20$ (lengthen D1/2 and shorten D3), and $\gamma^* = +30$ (shorten D5 and lengthen D6).

REFERENCES

K. Adachi and N. T. Trendafilov. Sparse principal component analysis subject to prespecified cardinality of loadings. *Computational Statistics*, 31(4):1403–1427, Dec. 2016. ISSN 0943-4062, 1613-9658. doi:10.1007/s00180-015-0608-4.

J. R. Adrion, C. B. Cole, N. Dukler, J. G. Galloway, A. L. Gladstein, G. Gower, C. C. Kyriazis, A. P. Ragsdale, G. Tsambos, F. Baumdicker, J. Carlson, R. A. Cartwright, A. Durvasula, I. Gronau, B. Y. Kim, P. McKenzie, P. W. Messer, E. Noskova, D. Ortega-Del Vecchyo, F. Racimo, T. J. Struck, S. Gravel, R. N. Gutenkunst, K. E. Lohmueller, P. L. Ralph, D. R. Schrider, A. Siepel, J. Kelleher, and A. D. Kern. A community-maintained standard library of population genetic models. *eLife*, 9:e54967, June 2020. ISSN 2050-084X. doi:10.7554/eLife.54967.

D. H. Alexander, J. Novembre, and K. Lange. Fast model-based estimation of ancestry in unrelated individuals. *Genome Research*, 19(9):1655–1664, Sept. 2009. ISSN 1088-9051. doi:10.1101/gr.094052.109.

A. M. S. Ang and N. Gillis. Accelerating Nonnegative Matrix Factorization Algorithms using Extrapolation. *Neural Computation*, 31(2):417–439, Feb. 2019. ISSN 0899-7667, 1530-888X. doi:10.1162/neco_a_01157.

F. L. Bauer. Das verfahren der treppeniteration und verwandte verfahren zur lösung algebraischer eigenwertprobleme. *Zeitschrift für angewandte Mathematik und Physik ZAMP*, 8(3):214–235, 1957.

R. Bhatia, T. Jain, and Y. Lim. On the Bures–Wasserstein distance between positive definite matrices. *Expositiones Mathematicae*, 37(2):165–191, 2019.

C. M. Bishop. Variational principal components. In *In Proceedings Ninth International Conference on Artificial Neural Networks, ICANN'99*, pages 509–514, 1999.

L. Breiman and J. H. Friedman. Estimating Optimal Transformations for Multiple Regression and Correlation. *Journal of the American Statistical Association*, 80(391):580–598, 1985. ISSN 0162-1459. doi:10.2307/2288473.

L. L. Cavalli-Sforza and A. W. Edwards. Phylogenetic analysis. Models and estimation procedures. *American journal of human genetics*, 19(3 Pt 1):233, 1967.

L. L. Cavalli-Sforza, I. Barrai, and A. W. Edwards. Analysis of human evolution under random genetic drift. In *Cold Spring Harbor Symposia on Quantitative Biology*, volume 29, pages 9–20. Cold Spring Harbor Laboratory Press, 1964. ISBN 0091-7451.

C. C. Chang, C. C. Chow, L. C. Tellier, S. Vattikuti, S. M. Purcell, and J. J. Lee. Second-generation PLINK: Rising to the challenge of larger and richer datasets. *GigaScience*, 4 (1):7, Dec. 2015. ISSN 2047-217X. doi:10.1186/s13742-015-0047-8.

Y. Chen, Y. Chi, J. Fan, and C. Ma. Spectral Methods for Data Science: A Statistical Perspective. *Foundations and Trends® in Machine Learning*, 14(5):566–806, 2021. ISSN 1935-8237, 1935-8245. doi:10.1561/2200000079.

A. M. Chiu, E. K. Molloy, Z. Tan, A. Talwalkar, and S. Sankararaman. Inferring population structure in biobank-scale genomic data. *The American Journal of Human Genetics*, 109 (4):727–737, Apr. 2022. ISSN 00029297. doi:10.1016/j.ajhg.2022.02.015.

A. Cichocki, A. H. Phan, and C. Caiafa. Flexible HALS algorithms for sparse non-negative matrix/tensor factorization. In *2008 IEEE Workshop on Machine Learning for Signal Processing*, pages 73–78, Oct. 2008. doi:10.1109/MLSP.2008.4685458.

A. d'Aspremont, L. Ghaoui, M. Jordan, and G. Lanckriet. A Direct Formulation for Sparse PCA Using Semidefinite Programming. In *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2004.

C. H. Ding, T. Li, and M. I. Jordan. Convex and Semi-Nonnegative Matrix Factorizations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):45–55, Jan. 2010. ISSN 1939-3539. doi:10.1109/TPAMI.2008.277.

C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Economist. America's best firms...and the rest: New winners and losers are emerging after three turbulent years. *The Economist*, 445(9324), Dec. 2022. ISSN 0013-0613.

E. F. Fama and K. R. French. Common risk factors in the returns on stocks and bonds. *Journal of financial economics*, 33(1):3–56, 1993.

J. Felsenstein. Maximum-likelihood estimation of evolutionary trees from continuous characters. *American journal of human genetics*, 25(5):471, 1973.

J. Felsenstein. Evolutionary trees from DNA sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17(6):368–376, Nov. 1981. ISSN 0022-2844, 1432-1432. doi:10.1007/BF01734359.

N. Gillis. *Nonnegative Matrix Factorization.* Number 2 in Data Science. Society for Industrial and Applied Mathematics, Philadelphia, 2021. ISBN 978-1-61197-641-0.

N. Gillis and F. Glineur. Accelerated Multiplicative Updates and Hierarchical ALS Algorithms for Nonnegative Matrix Factorization. *Neural Computation*, 24(4):1085–1105, Apr. 2012. ISSN 0899-7667, 1530-888X. doi:10.1162/NECO_a_00256.

G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins Studies in the Mathematical Sciences. The Johns Hopkins University Press, Baltimore, fourth edition edition, 2013. ISBN 978-1-4214-0794-4.

R. Guerra-Urzola, K. Van Deun, J. C. Vera, and K. Sijtsma. A Guide for Sparse PCA: Model Comparison and Applications. *Psychometrika*, June 2021. ISSN 0033-3123, 1860-0980. doi:10.1007/s11336-021-09773-2.

J. R. Huyghe, E. Fransen, S. Hannula, L. Van Laer, E. Van Eyken, E. Mäki-Torkko, P. Aikio, M. Sorri, M. J. Huentelman, and G. V. Camp. A genome-wide analysis of population structure in the finnish saami with implications for genetic association studies. *European journal of human genetics*, 19(3):347–352, 2011.

I. T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics. Springer, New York, 2nd ed edition, 2002. ISBN 978-0-387-95442-4.

J. Jouganous, W. Long, A. P. Ragsdale, and S. Gravel. Inferring the Joint Demographic History of Multiple Populations: Beyond the Diffusion Approximation. *Genetics*, 206(3): 1549–1567, July 2017. ISSN 1943-2631. doi:10.1534/genetics.117.200493.

M. Journée, Y. Nesterov, P. Richtárik, and R. Sepulchre. Generalized power method for sparse principal component analysis. *Journal of Machine Learning Research*, 11(15):517–553, 2010.

Y. Kim, W. Wang, P. Carbonetto, and M. Stephens. A flexible empirical Bayes approach to multiple linear regression and connections with penalized regression, Aug. 2022.

C. Kolomvakis and N. Gillis. Robust binary component decompositions. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

R. Kueng and J. A. Tropp. Binary component decomposition part i: the positive-semidefinite case. *SIAM Journal on Mathematics of Data Science*, 3(2):544–572, 2021.

T. C. Lamnidis, K. Majander, C. Jeong, E. Salmela, A. Wessman, V. Moiseyev, V. Khartanovich, O. Balanovsky, M. Ongyerth, A. Weihmann, et al. Ancient fennoscandian genomes reveal origin and spread of siberian ancestry in europe. *Nature communications*, 9(1):5018, 2018.

M. E. Lauterbur, M. I. A. Cavassim, A. L. Gladstein, G. Gower, N. S. Pope, G. Tsambos, J. Adrion, S. Belsare, A. Biddanda, V. Caudill, J. Cury, I. Echevarria, B. C. Haller, A. R. Hasan, X. Huang, L. N. M. Iasi, E. Noskova, J. Obšteter, V. A. C. Pavinato, A. Pearson, D. Peede, M. F. Perez, M. F. Rodrigues, C. C. R. Smith, J. P. Spence, A. Teterina, S. Tittes, P. Unneberg, J. M. Vazquez, R. K. Waples, A. W. Wohns, Y. Wong, F. Baumdicker, R. A. Cartwright, G. Gorjanc, R. N. Gutenkunst, J. Kelleher, A. D. Kern, A. P. Ragsdale, P. L. Ralph, D. R. Schrider, and I. Gronau. Expanding the stdpopsim species catalog, and lessons learned for realistic genome simulations. Preprint, elife, Mar. 2023.

D. J. Lawson, L. van Dorp, and D. Falush. A tutorial on how not to over-interpret STRUCTURE and ADMIXTURE bar plots. *Nature Communications*, 9(1):3258, Dec. 2018. ISSN 2041-1723. doi:10.1038/s41467-018-05257-7.

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

J. Z. Li, D. M. Absher, H. Tang, A. M. Southwick, A. M. Casto, S. Ramachandran, H. M. Cann, G. S. Barsh, M. Feldman, and L. L. Cavalli-Sforza. Worldwide human relationships inferred from genome-wide patterns of variation. *science*, 319(5866):1100–1104, 2008.

Y. Li, R. Zhu, A. Qu, H. Ye, and Z. Sun. Topic modeling on triage notes with semiorthogonal nonnegative matrix factorization. *Journal of the American Statistical Association*, 116 (536):1609–1624, 2021.

Y. J. Lim and Y. W. Teh. Variational Bayesian Approach to Movie Rating Prediction. 2007.

M. Lipson, P.-R. Loh, A. Levin, D. Reich, N. Patterson, and B. Berger. Efficient Moment-Based Inference of Admixture Parameters and Sources of Gene Flow. *Molecular Biology and Evolution*, 30(8):1788–1802, Aug. 2013. ISSN 1537-1719, 0737-4038. doi:10.1093/molbev/mst099.

Y. Liu, P. Carbonetto, J. Willwerscheid, S. A. Oakes, K. F. Macleod, and M. Stephens. Dissecting tumor transcriptional heterogeneity from single-cell rna-seq data by generalized binary covariance decomposition. *bioRxiv*, pages 2023–08, 2023.

J. Lonsdale, J. Thomas, M. Salvatore, R. Phillips, E. Lo, S. Shad, R. Hasz, G. Walters, F. Garcia, N. Young, et al. The genotype-tissue expression (gtex) project. *Nature genetics*, 45(6):580–585, 2013.

Z. Ma. Sparse principal component analysis and iterative thresholding. *The Annals of Statistics*, 41(2), Apr. 2013. ISSN 0090-5364. doi:10.1214/13-AOS1097.

L. Mackey. Deflation Methods for Sparse PCA. In *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008.

J. H. Marcus. *Inferring Interpretable Representations of Population Structure*. PhD thesis, The University of Chicago, 2020.

N. Parikh and S. Boyd. Proximal algorithms. *Foundations and trends® in Optimization*, 1 (3):127–239, 2014.

N. Patterson, P. Moorjani, Y. Luo, S. Mallick, N. Rohland, Y. Zhan, T. Genschoreck, T. Webster, and D. Reich. Ancient Admixture in Human History. *Genetics*, 192(3):1065–1093, Nov. 2012. ISSN 1943-2631. doi:10.1534/genetics.112.145037.

K. Pearson. LIII. *On lines and planes of closest fit to systems of points in space*. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11): 559–572, Nov. 1901. ISSN 1941-5982, 1941-5990. doi:10.1080/14786440109462720.

J. K. Pickrell and J. K. Pritchard. Inference of Population Splits and Mixtures from Genome-Wide Allele Frequency Data. *PLoS Genetics*, 8(11):e1002967, Nov. 2012. ISSN 1553-7404. doi:10.1371/journal.pgen.1002967.

J. K. Pritchard, M. Stephens, and P. Donnelly. Inference of Population Structure Using Multilocus Genotype Data. *Genetics*, 155(2):945–959, June 2000. ISSN 1943-2631. doi:10.1093/genetics/155.2.945.

A. Raj, M. Stephens, and J. K. Pritchard. fastSTRUCTURE: Variational Inference of Population Structure in Large SNP Data Sets. *Genetics*, 197(2):573–589, June 2014. ISSN 1943-2631. doi:10.1534/genetics.114.164350.

J. Ramsay and B. Silverman. Principal components analysis for functional data. *Functional data analysis*, pages 147–172, 2005.

K. Rohe and M. Zeng. Vintage factor analysis with Varimax performs statistical inference. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 85(4):1037–1060, July 2023. ISSN 1369-7412. doi:10.1093/jrsssb/qkad029.

R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th International Conference on Machine Learning - ICML '08*, pages 880–887, Helsinki, Finland, 2008. ACM Press. ISBN 978-1-60558-205-4. doi:10.1145/1390156.1390267.

H. Shen and J. Z. Huang. Sparse principal component analysis via regularized low rank matrix approximation. *Journal of Multivariate Analysis*, 99(6):1015–1034, July 2008. ISSN 0047259X. doi:10.1016/j.jmva.2007.06.007.

R. N. Shepard and P. Arabie. Additive clustering: Representation of similarities as combinations of discrete overlapping properties. *Psychological Review*, 86(2):87–123, Mar. 1979. ISSN 1939-1471, 0033-295X. doi:10.1037/0033-295X.86.2.87.

M. Sørensen, N. D. Sidiropoulos, and A. Swami. Overlapping community detection via semi-binary matrix factorization: Identifiability and algorithms. *IEEE Transactions on Signal Processing*, 70:4321–4336, 2022.

M. Stephens. False discovery rates: A new deal. *Biostatistics*, page kxw041, 2017. ISSN 1465-4644, 1468-4357. doi:10.1093/biostatistics/kxw041.

The 1000 Genomes Project Consortium. A map of human genome variation from population-scale sequencing. *Nature*, 467(7319):1061–1073, Oct. 2010. ISSN 0028-0836, 1476-4687. doi:10.1038/nature09534.

The 1000 Genomes Project Consortium. A global reference for human genetic variation. *Nature*, 526(7571):68, 2015.

M. E. Tipping and C. M. Bishop. Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, Aug. 1999. ISSN 1369-7412, 1467-9868. doi:10.1111/1467-9868.00196.

S. M. Urbut, G. Wang, P. Carbonetto, and M. Stephens. Flexible statistical methods for estimating and testing effects in genomic studies with multiple conditions. *Nature Genetics*, 51(1):187–195, Jan. 2019. ISSN 1061-4036, 1546-1718. doi:10.1038/s41588-018-0268-8.

K. Van Deun, T. F. Wilderjans, R. A. Van Den Berg, A. Antoniadis, and I. Van Mechelen. A flexible framework for sparse simultaneous component based data integration. *BMC bioinformatics*, 12(1):1–17, 2011.

G. Wang, A. Sarkar, P. Carbonetto, and M. Stephens. A simple new approach to variable selection in regression, with application to genetic fine mapping. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82(5):1273–1300, Dec. 2020. ISSN 1369-7412, 1467-9868. doi:10.1111/rssb.12388.

W. Wang. *Applications of Adaptive Shrinkage in Multiple Statistical Problems*. PhD thesis, The University of Chicago, 2017.

W. Wang and M. Stephens. Empirical bayes matrix factorization. *Journal of Machine Learning Research*, 22(120):1–40, 2021.

J. H. J. H. Wilkinson. *The Algebraic Eigenvalue Problem,*. Clarendon Press, Oxford,, 1965.

J. Willwerscheid. *Empirical Bayes Matrix Factorization: Methods and Applications*. PhD thesis, The University of Chicago, 2021.

J. Willwerscheid and M. Stephens. ebnm: An r package for solving the empirical bayes normal means problem using a variety of prior families. *arXiv preprint arXiv:2110.00152*, 2021.

D. M. Witten, R. Tibshirani, and T. Hastie. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515–534, July 2009. ISSN 1465-4644, 1468-4357. doi:10.1093/biostatistics/kxp008.

Z. Xing, P. Carbonetto, and M. Stephens. Flexible signal denoising via flexible empirical bayes shrinkage. *The Journal of Machine Learning Research*, 22(1):4153–4180, 2021.

R. Zass and A. Shashua. Nonnegative sparse pca. *Advances in neural information processing systems*, 19, 2006.

T. Zerjal, B. Dashnyam, A. Pandya, M. Kayser, L. Roewer, F. R. Santos, W. Schiefenhövel, N. Fretwell, M. A. Jobling, and S. Harihara. Genetic relationships of Asians and Northern Europeans, revealed by Y-chromosomal DNA analysis. *American journal of human genetics*, 60(5):1174, 1997.

X. Zhong, C. Su, and Z. Fan. Empirical Bayes PCA in high dimensions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, page rssb.12490, Jan. 2022. ISSN 1369-7412, 1467-9868. doi:10.1111/rssb.12490.

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005. ISSN 1467-9868. doi:10.1111/j.1467-9868.2005.00503.x.

H. Zou and L. Xue. A Selective Overview of Sparse Principal Component Analysis. *Proceedings of the IEEE*, 106(8):1311–1320, Aug. 2018. ISSN 1558-2256. doi:10.1109/JPROC.2018.2846588.

H. Zou, T. Hastie, and R. Tibshirani. Sparse Principal Component Analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, June 2006. ISSN 1061-8600, 1537-2715. doi:10.1198/106186006X113430.