Sparse Principal Component Analysis for High-Dimensional Data: A Comparative Study

SPARSE PRINCIPAL COMPONENT ANALYSIS FOR HIGH-DIMENSIONAL DATA: A COMPARATIVE STUDY

BY

ASHLEY BONNER, B.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF MATHEMATICS & STATISTICS

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

Master of Science (2012) McMaster University
(Mathematics & Statistics) Hamilton, Ontario, Canada

TITLE: Sparse Principal Component Analysis for High-Dimensional Data: A Comparative Study

AUTHOR: Ashley Bonner
B.Sc., (Statistics)
McMaster University, Canada

SUPERVISOR: Dr. Joseph Beyene

NUMBER OF PAGES: xi, 113

I dedicate this thesis to whomever might use it!

Abstract

Background: Through unprecedented advances in technology, high-dimensional datasets have exploded into many fields of observational research. For example, it is now common to expect thousands or sometimes millions of genetic variables (p) with only a limited number of study participants (n). Determining the important features, in whatever context, proves extremely difficult from a statistical point of view, as multivariate analysis techniques become flooded and mathematically insufficient when $n < p$. Principal Component Analysis (PCA) is a commonly used multivariate method for dimension reduction and data visualization but suffers from these issues. A collection of Sparse PCA methods have been proposed to counter these flaws and obtain insight to variable grouping structure but have not been tested in comparative detail. This thesis will compare three of the latest Sparse PCA methods, exposing the optimal choices under a variety of scenarios.

Methods: Performances of Sparse PCA methods were evaluated through simulations. Data was generated for 56 different scenarios, ranging various properties of the variance structure from which PCA output depend on; $\frac{p}{n}$, the number of underlying groups, and the variance structure between and within them. Prediction-based cross-validation methods were used to tune the level of sparseness. Closeness of true underlying loading vectors and estimated loading vectors was evaluated based on the angle between, classification, and sparseness, while adjusted percentage explained variance and orthogonality was also captured. Sparse PCA methods were also applied to a real gene expression dataset from SickKids hospital.

Results: All Sparse PCA methods showed improvements upon classical PCA in most criteria, especially classification. Some methods are best at obtaining an accurate leading principal component (PCs) only, whereas others are better for obtaining accurate subsequent PCs. There exist different optimal choices of Sparse PCA methods when ranging within-group correlation and across-group variances; thankfully, one method repeatedly works well under the most difficult scenarios. When applying methods to the real data, concise groups of gene expressions were detected with the most sparse methods.

Conclusions: Sparse PCA methods provide a new insightful way to detect important features amidst complex high-dimension data but simulation results, such as the ones given in this thesis, should be used to ensure they will accurately reflect the underlying variance structure.

Acknowledgements

Dr. Joseph Beyene: sparking ideas, guiding me through rough times, and infusing statistical energy whenever we met; I could not have asked for a better supervisor.

Dr. Jemila Hamid, Dr. Roman Viveros-Aguillera: my committee members.

Dr. Peter Macdonald, Dr. Aaron Childs, Dr. Angelo Canty: my professors who gave me the passion and tools for understanding statistics and teaching.

Sathish Pichika, Xiao Di (Sandy) Yang: friends, colleagues, and my guinea-pigs in the thesis process.

Binod Neupane: an excellent guide through coding issues and lovely talks.

And... last, but not least... my Mom (Linda), Dad (Mark), and future wife (Stephanie) who not only supported me through the process, but also saw many of our shared hours succumb to "thesis".

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Over the last decade or so, a collection of sparse applications to principal component analysis (PCA) have been proposed in hopes to offer statisticians a superior dimension reduction procedure and an interpretable gateway to variable selection, especially when confronting high-dimensional data. After demonstrating the framework and theories, this thesis will attempt to deliver a compact guide to using Sparse Principal Component Analysis (Sparse PCA) in both synthetic and realistic atmospheres; exposing its power and pitfalls.

After this Chapter brings some motivation for why Sparse PCA is needed, Chapter 2 will cover the methodology behind Sparse PCA, including three specific methods found in recent literature. Chapter 3 will contain the innovative contribution of this thesis; a rigorous simulation study that will expose the three Sparse PCA methods and find out where they succeed and where they fail. Chapter 4 will demonstrate a hands-on application of Sparse PCA to a real-world high-dimensional gene expression dataset, giving the reader an idea of how to make use of the simulation findings during their analysis. We will also outline some problems one might face, along with possible solutions, while implementing Sparse PCA.

So, on to some motivation to align the the reader with the statistical problems we face, the research fields that generate them, the already existing solutions, and how exactly Sparse PCA can be the superior answer.

1.1 Motivations

It is of utmost importance to understand where a problem begins before attempting to solve it. The roots from which trouble stems will lead to a well of insight and motivation. A moment to reflect on why we, as statisticians, are here in the first place.

1.1.1 Personal Motivations - Answering the research questions

Statistical analysis is required in every research-oriented field that generates data. You will find people passionate towards an unimaginable number of areas; a never-ending source of data that compromises our understanding of material and road-blocks many experts from confirming their discoveries. As statisticians, we get the chance to walk along-side a well-earned (and funded) mission towards an answer of which, usually, we have no theoretical understanding. We “hop the line”, so to speak, with our unique skills and become part of the latest research. Whether joining a project or working on our own methodological discoveries, we must always draw motivation from the questions at hand because they mean something to somebody.

Health research exemplifies this, receiving heart-felt attention from the public masses. Donations pour in to help fund research towards finding cures for diseases that affect the population; cancer, diabetes, multiple sclerosis, lupus, scleroderma, and many other expressions of chronic disease. Each dollar represents trust that usable results are on the way and this puts incredible responsibility on the statisticians to perform appropriate analyses. There will never be a lack of interest in discovering the mechanisms that cause and fuel disease. It seems cliche to announce that everyone is affected by health problems, but it's true. Relationships will emotionally tie anyone down to an unfair diagnosis of a loved one, and even if there is no personal connection, any level of empathy will allow one to relate. It's no wonder why there is so much devotion towards these biological research questions, but the amount of data collected is perhaps becoming unmanageable. Which questions will be solved?

The most interesting datasets today will be the ones where least is known. Hypotheses dealing with the human genome and its relation to many biological functions have always been of significant interest but, with technology and computing power on the rise, it may become the main focus for many more professions. Only within the last decade have technological advances allowed the *abundant* collection of genetic information. Front-line companies in genetic data collection and research, like Affymetrix, have endorsed the microarray chip and propelled genetic data into an unthinkable dimension. This allows for a new but complicated approach to attaining answers to why and who diseases attack. With the increased quantity of data, statisticians must play a crucial role in the chase towards such incredible discoveries. In other words, we now battle ‘high-dimensional’ data! Lots and lots of variables and often, little sample size. It is from these types of motivations that we have work; new statistical methods are often ensued because of the problems encountered during the search for application-based answers.

A beautiful realization of statistical developments is that they are often transferable to other fields of interest. As long as parallels can be made between research questions and data structure in respective fields, the works and intuition of a statistician can be beneficial to many. In this light, the Sparse PCA talked about in this thesis need not be strapped down to massive datasets in genetic research. It is simply a very nice platform to show where the methods are useful; keep this in mind.

1.1.2 Statistical Motivations - Technical challenges of variable selection

Statistics are simply a set of tools to aid someone in understanding a phenomenon when they cannot put the pieces together. Forget data for a second and let’s build a little perspective; data is simply an attempt to track what we observe. Through observing the process we build a hypothesis and through repeated observation stack evidence in favour of or against our claims. We try to lock down indisputable reasons to why things might occur but, unfortunately for us, most mechanisms under study are affected by so many factors, that *there will never be indisputable reasons.* A child, for example, does not have all the answers to differentiate the subjective ‘right’ from 'wrong'. At home they observe how their parents behave, at school they observe how their teacher disciplines, and at birthdays they observe how their friends interact. As the child grows up, it seems clear that to be 'right' depends on who is the judge - possibly the biggest *factor*. However, it also depends on your own beliefs, how much sleep the judge had, as well as a million other arguably important circumstantial factors. How on earth can the child grow to fully understand 'right' from 'wrong'? He or she probably never will.

"Aha, but we humans don't give up!" If there were a way to obtain every single bit of information surrounding the phenomenon under study, perhaps we could find that indisputable reason after all. With this goal in mind, our senses and memory are clearly not able to be solely depended on so, to track everything, data is collected. Enter numbers, modelling, error, statistics, and individuals who are interested in these subjects; statisticians. The more observations, the more evidence. The more relevant factors captured, the more error explained. A remarkable effort that has transcended previous discovery and pushes us to learn more.

With this in mind, a statistician puts on their hard-hat, picks up their tool-box ("I hope you've got your Pearson wrench!"), and attempts to follow the blueprints that come along with each new inquiry; if the wrong tools are chosen or blueprints ignored, a research question will never be answered with accuracy.

The importance of Variable Selection

Every research question requiring data analysis will usually employ some sort of regression model, targeting an outcome of interest, and focus on one of two distinctly fundamental goals (these thoughts are shared by at least [1, 22]).

The **first goal** is driven by the need for *insight into relationships*; the ability to detect, or select, important explanatory variables related to an outcome of interest. The **second goal** is driven by the need to *accurately predict a process*; the ability to predict the outcome with minimal error. The two goals can be thought of or focused on disjointly but are almost always desired together.

For example, imagine a fishing company whose financial success heavily relies on securing subscriptions to their monthly magazine - how else to advertise the great deals on bait. Hitting the new year, they are about to launch their January 2012 magazine and might be interested in a targeted marketing plan this time, as randomly sending subscription fliers to a percentage of local residents generated a measly 5% subscription-rate last year. Since they have access to public demographic and socio-economic information, such as gender and household income, perhaps they could use this database to build a logistic regression model from the sample they sent to last year. They would be predicting the probability of subscription to last year's magazine and could use their model to forecast which people will be most likely to subscribe in 2012. With a strictly money-driven mindset for 2012, there would be no care for understanding relationships (i.e. selecting *causal* variables); as long as the person will subscribe to the magazine, the company is happy. However, one could argue, if the fishing company does not understand the 'type' of people who are buying their products, i.e. the company simply reaches out based on a predictive model and doesn't understand the causal factors, they might lose touch with their product line and be surpassed by competitors who understand how to cater their products to their audience in the creation stage.

In the context of more scientific fields, the need to meet both goals is also apparent. In our fight against scleroderma, a terrible connective tissue disease for which the cause is unknown, we hope the end result of our efforts will be a cure, though, in the mean time, early diagnosis is imperative to treatment. With abundant genetic information arriving on the scene, an excellent predictive model might be obtained by a step-wise procedure to detect who is likely to be diagnosed in their future. However, it would not reveal the complex causal relationships that may exist between genetic markers and scleroderma, thus limiting our efforts to treatment, not cure.

It is natural to expect a good predictive model to hold some insight to relationships and a simple, insightful model to have decent predictive power; in fact this is often the case. But these examples also demonstrate how a change in research question can shift and narrow down a statistician's focus to one of the two goals. With the vast amount of easy-to-access data these days, prediction might have become the easier mission as it is achieved easily with lots of variables. Granted that more information is better, insight to relationships becomes difficult. Prediction is a very powerful tool, but gaining insight to relationships may be the more sought-after goal in terms of driving science and it hinges on *variable selection.*

High-Dimension Data poses an issue

The challenge of selecting the 'important variables' can be relatively easy for datasets with a single response variable, few explanatory variables, and a sufficiently large number of observations (n) to accurately estimate model parameters (p). A less fortunate scenario involves a small number of observations and an increasing number of explanatory variables; specifically, $p > n$. The techniques used for estimation and inference, once mathematically convenient, now become a nightmare. This environment of so-called *high-dimension* (referring to the large number of variables) is typically found when building models from our previously discussed genetics data. Here, there are few participants and many unfamiliar genetic variables; if we were familiar with the variables, perhaps we could exclude some based on prior knowledge of relationships to the response.

When $p > n$, there are not enough degrees of freedom to simultaneously estimate all of the parameters, let alone infer about their significance or about model adequacy. Furthermore, with any large number of unfamiliar variables, multicollinearity is inevitable, leading to ill-conditioned data from which sensitive estimates and unreliable, non-robust models are built [8, 22]. These issues have forced invention of modelling procedures that simultaneously determine important factors while discarding pesky non-factors. Debates upon which procedures are most appropriate for a variety of environments are ongoing.

Possible solutions

Subset and step-wise selection procedures, though ideal for prediction-based modelling, have been proven greedy and unreliable in circumstances when variable selection is at the heart of interest [8, 19, 22]. With a high-dimensional dataset, response-associated explanatory variables will be cast aside on a very random basis, sensitive to correlation amongst predictors and the order of entry. We are looking for something new.

The growing class of *penalized* regression techniques might offer a more complete package to capture important variables, especially when battling in a high-dimensional data environment. Along with the ability to obtain superior estimates through something called the bias-variance trade-off, they can introduce *sparseness* to parameter estimation, i.e. are able to obtain a solution set with only a few (sparse) non-zero parameter estimates [19, 5, 22].

Moving out from under the umbrella of regression, *principal component analysis* (PCA) offers a supplementary, pre-regression approach that focuses on variance structure within a set of continuous variables. Disregarding the response variable, principal components hold insight to natural groups of explanatory variables and their expression in variance relative to the rest of the data. Furthermore, transformation of the original dataset via these principal components can offer a reduction in dimensionality as a prior step to regression. Unfortunately, though, the benefits from the transformation are scarred by the lack of one's ability to interpret the resulting data. An altered PCA method is called for to alleviate these issues.

Enter Sparse PCA

Through much hard work and deliberation, a combination of penalized regression and PCA was born and eventually named 'Sparse Principal Component Analysis' (Sparse PCA). It now pushes the boundaries of statistical exploration in the land of high-dimensional data. Since these methods are new to many statisticians, some may be reluctant to use them; justifiably so, especially for those who want to work out all the details for themselves. To push these methods along, more work and exploration should be done. The ideas, works, and contributions of this thesis arise from all of these motivations.

Chapter 2

Methods

By introducing the issues a statistician encounters when faced with high-dimensional data, especially when $n < p$, it appears there is no unifying strategy to battle the unfortunate situation. However, by adding a class of interpretable (sparse) PCA approaches to the arsenal of a statistician, there may be more hope. This Chapter will begin by introducing the crucial foundations of Classical PCA, the well-established Penalized Regression techniques that it absorbs, and finally the resulting Sparse PCA. Since Sparse PCA has been mathematically formulated differently over the past few years, we have chosen three methods to focus on in this thesis.

2.1 Principal Component Analysis

Since its invention in 1901 by Karl Pearson, Principal Component Analysis (PCA) has been used repeatedly across many fields as a data visualisation and dimension reduction technique. An example of a very useful application is image compression.

Made efficiently possible by the foundations of PCA, image compression allows us to save storage space while preserving image quality; something extremely crucial in this new technology-driven age. With any general dataset, from the original variables it finds uncorrelated linear combinations, called principal components, that express maximal variation in the data and provides the statistician with a choice to transform the original high-dimensional dataset into one of much lesser dimension at the cost of, 1) Information loss (variance), and 2) Ability to interpret new variables and analysis.

These drawbacks present themselves in varying amounts dependent on the data and must be understood in order to confidently use PCA. The second expense is the thorn in the side of this amazing technique. If one had the ability to interpret results from PCA, they would have a go-to technique to battles both high-dimension and collinearity. Although many textbooks and papers have covered the theory and computational techniques behind PCA, it is worth while to freshly cover its details here as it is the foundation for all Sparse PCA methods encountered in this thesis. In this section, we first embrace the mathematical foundations of PCA, then show the related importance of the Singular Value Decomposition (SVD) and the rank-K approximation, and conclude with an R-based computational example of PCA.

2.1.1 Mathematical Foundations for PCA

Let $\boldsymbol{X} \sim N_p(\boldsymbol{\mu}, \Sigma)$, i.e. $\boldsymbol{X} = (X_1, X_2, \ldots, X_p)'$ is a p-dimensional random vector with a multivariate normal distribution. The mean vector, $\boldsymbol{\mu}$, is of little importance. The main focus of PCA is to investigate data patterns through the variance-covariance structure, dictated by Σ, which is undisturbed by specification of the mean vector.

In fact, most applications and theories using PCA will use centred data for simplicity in notation and calculation; we will abide by this. Computations in PCA require no distributional assumptions on the data matrix, however further inferential techniques rely on the multivariate normality assumption to an extent [11, p. 431].

The theory behind PCA is inspired by the search for a simpler way to describe the variation in the data. Currently, there are p variables that may be correlated with one another; for a researcher interested in variable selection via a regression technique (to predict a variable, say Y), this is not a fortunate scenario. If, for example, X_1 and X_2 were highly correlated, one could argue that either X_1 or X_2 could be excluded from the analysis as the other sufficiently represents it. In other words, redundant variables might be excluded [11, p. 431]. However, there are often far too many explanatory variables to eye-ball the correlation structure and argue this out without terrible mistakes. Another technique might be to exclude variables with very little variation, i.e. if $Var(X_1) < \textit{chosen threshold}$, since they would not have a sufficient range of values to detect coinciding changes in the response anyway. PCA introduces a way of characterizing the variation in the data via linear combinations of the original variables with nice and convenient properties. In other words, it introduces a *change in basis* from the original variables to a more efficient set. In the general p-dimensional case, this amounts to finding a candidate set of new variables,

$$\begin{aligned}
Z_1 &= v_{11}X_1 + v_{12}X_2 + \cdots + v_{1p}X_p = \boldsymbol{v_1}'\boldsymbol{X} \\
Z_2 &= v_{21}X_1 + v_{22}X_2 + \cdots + v_{2p}X_p = \boldsymbol{v_2}'\boldsymbol{X} \\
&\vdots \\
Z_p &= v_{p1}X_1 + v_{p2}X_2 + \cdots + v_{pp}X_p = \boldsymbol{v_p}'\boldsymbol{X} \qquad (2.1)
\end{aligned}$$

The linear combinations $Z_1, Z_2, \ldots, Z_p$ are called *principal components* (PCs), the coefficient vectors $\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_p}$ are called *loading vectors*, and the coefficients within each vector $\boldsymbol{v_j}$ are called the *loadings* for that vector. The loading vectors are of primary interest as they are our only way of relating the principal components back to intuitive measures, i.e. the original variables. In classical PCA, the principal components are successively chosen with the following step by step process:

Principal Component 1

$Z_1 = \boldsymbol{v_1}'\boldsymbol{X}$ chosen to maximize $Var(Z_1) = \boldsymbol{v}_1'\Sigma\boldsymbol{v}_1$

subject to $\boldsymbol{v_1}'\boldsymbol{v_1} = 1$.

Principal Component 2

$Z_2 = \boldsymbol{v_2}'\boldsymbol{X}$ chosen to maximize $Var(Z_2) = \boldsymbol{v}_2'\Sigma\boldsymbol{v}_2$

subject to $\boldsymbol{v_2}'\boldsymbol{v_2} = 1$ and $Cov(Z_1, Z_2) = 0$.

$\vdots$

Principal Component j

$Z_j = \boldsymbol{v_j}'\boldsymbol{X}$ chosen to maximize $Var(Z_j) = \boldsymbol{v}_j'\Sigma\boldsymbol{v}_j$

subject to $\boldsymbol{v_j}'\boldsymbol{v_j} = 1$ and $Cov(Z_j, Z_m) = 0 \ \forall \, m < j$.

$\vdots$

Principal Component p

$Z_p = \boldsymbol{v_p}'\boldsymbol{X}$ chosen to maximize $Var(Z_p) = \boldsymbol{v}_p'\Sigma\boldsymbol{v}_p$

subject to $\boldsymbol{v_p}'\boldsymbol{v_p} = 1$ and $Cov(Z_p, Z_m) = 0 \ \forall \, m < p$.

The sum of squared loadings constraint assures that variance for the respective PC cannot be increased without bound. With each PC successively attaining maximum variance while being mutually uncorrelated with the rest, the following facts can be established formally:

Important Properties of PCA:

1. **Maximized Variances:** $Var(Z_1) \geq Var(Z_2) \geq \ldots \geq Var(Z_p) \geq 0$.
2. **Orthogonal Loading Vectors, Uncorrelated Principal Components:** $\boldsymbol{v}_j'\boldsymbol{v}_m = 0$ and $Cov(Z_j, Z_m) = 0$ for $j \neq m$.
3. **Total Variance preserved:** $\sum_{j=1}^{p} Var(Z_j) = \sum_{j=1}^{p} Var(X_j)$.

In words, these facts state that the full set of PCs contain an equal amount of information (variance) as the original set, but now in a structure that is potentially beneficial for analysis. Fact 1 is extremely important to dimension reduction. If, for instance, the first 3 PCs explained 95 percent of the variation found in the data, we could throw away the rest with little worry about losing important information. Fact 2 ensures the resulting variables are uncorrelated; no more worries about the effect of collinearity on our analysis. The only downfall from a regression standpoint is the interpretation. For a particular PC (linear combination), the loadings (coefficients) represent the contribution of each original variable but if there are a large number of variables, it could be very hard to determine exactly what the PC represents. Ideally, the PCs should reveal underlying *features* to enable easy interpretation. For this reason, PCA is often used as an exploratory approach towards understanding the data; a visible grouping of variables can aid in understanding a simpler structure.

Obtaining linear combinations that satisfy all of the above conditions is quite easy as they are directly connected to the eigen-structure of the variance-covariance matrix, Σ. Let the eigenvalues for Σ be $\lambda_1, \lambda_2, \ldots, \lambda_p$ and let their respective (normalized) eigenvectors be $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_p$. It can be shown, through the Cauchy Schwartz Inequality, that the desired true linear combinations are:

$$\begin{aligned}
Z_1 &= e_{11}X_1 + e_{12}X_2 + \cdots + e_{1p}X_p = \boldsymbol{e}_1'\boldsymbol{X} \\
Z_2 &= e_{21}X_1 + e_{22}X_2 + \cdots + e_{2p}X_p = \boldsymbol{e}_2'\boldsymbol{X} \\
&\vdots \\
Z_p &= e_{p1}X_1 + e_{p2}X_2 + \cdots + e_{pp}X_p = \boldsymbol{e}_p'\boldsymbol{X} \qquad (2.2)
\end{aligned}$$

with $Var(Z_1) \geq Var(Z_2) \geq \ldots \geq Var(Z_p) \geq 0$ equalling $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$ and $\sum_{j=1}^{p} Var(Z_j) = \sum_{j=1}^{p} \lambda_j = \sum_{j=1}^{p} Var(X_j) = \sum_{j=1}^{p} \sigma_{jj}$, where $Var(X_j) = \sigma_{jj}$ (the diagonal elements of Σ). Orthogonality amongst loading vectors is inherently satisfied by the fact they are made up of eigenvectors.

In application, we do not have the true variance-covariance matrix. We are usually given an $n \times p$ data matrix $\mathbf{X}$ with rows $\mathbf{x}_i$, for $i = 1, \ldots, n$, and rely on its sample variance-covariance matrix $\mathbf{S}$ to carry out PCA. Let the eigenvalues for $\mathbf{S}$ be $\hat{\lambda_1}, \hat{\lambda_2}, \ldots, \hat{\lambda_p}$, and let their respective (normalized) eigenvectors be $\hat{\boldsymbol{e}_1}, \hat{\boldsymbol{e}_2}, \ldots, \hat{\boldsymbol{e}_p}$. Then the sample linear combinations satisfying the same above conditions are:

$$\begin{aligned}
\boldsymbol{z}_1 &= \hat{e}_{11}x_1 + \hat{e}_{12}x_2 + \cdots + \hat{e}_{1p}x_p = \hat{\boldsymbol{e}}_1'\boldsymbol{x} \\
\boldsymbol{z}_2 &= \hat{e}_{21}x_1 + \hat{e}_{22}x_2 + \cdots + \hat{e}_{2p}x_p = \hat{\boldsymbol{e}}_2'\boldsymbol{x} \\
&\vdots \\
\boldsymbol{z}_p &= \hat{e}_{p1}x_1 + \hat{e}_{p2}x_2 + \cdots + \hat{e}_{pp}x_p = \hat{\boldsymbol{e}}_p'\boldsymbol{x} \qquad (2.3)
\end{aligned}$$

with $Var(\boldsymbol{z}_1) \geq Var(\boldsymbol{z}_2) \geq \ldots \geq Var(\boldsymbol{z}_p) \geq 0$ equalling $\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq \cdots \geq \hat{\lambda}_p \geq 0$, $\sum_{j=1}^{p} Var(\boldsymbol{z}_j) = \sum_{j=1}^{p} \hat{\lambda}_j = \sum_{j=1}^{p} Var(x_j) = \sum_{j=1}^{p} s_{jj}$, where $Var(x_j) = s_{jj}$ (the diagonal elements of $\mathbf{S}$) and orthogonality again met. Here, $\boldsymbol{x}$ represents any observation generated from the true underlying distribution. If referring to a specific observation $\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{ip})'$, a subscript i is simply added to obtain $z_{i1}, z_{i2}, \ldots, z_{ip}$. Therefore, with a sample size n, a dataset $\mathbf{X}_{n \times p} = (\mathbf{x}_1 \mid \mathbf{x}_2 \mid \cdots \mid \mathbf{x}_n)'$ is constructed and the sample PCs for each observation can be calculated as

$$\begin{aligned}
\boldsymbol{z}_1 &= \begin{pmatrix} z_{11} \\ z_{21} \\ \vdots \\ z_{n1} \end{pmatrix} = \begin{pmatrix} \hat{e}_{11}x_{11} + \hat{e}_{12}x_{12} + \cdots + \hat{e}_{1p}x_{1p} \\ \hat{e}_{11}x_{21} + \hat{e}_{12}x_{22} + \cdots + \hat{e}_{1p}x_{2p} \\ \vdots \\ \hat{e}_{11}x_{n1} + \hat{e}_{12}x_{n2} + \cdots + \hat{e}_{1p}x_{np} \end{pmatrix} = \mathbf{X}\hat{\boldsymbol{e}}_1 \\
\boldsymbol{z}_2 &= \begin{pmatrix} z_{12} \\ z_{22} \\ \vdots \\ z_{n2} \end{pmatrix} = \begin{pmatrix} \hat{e}_{21}x_{11} + \hat{e}_{22}x_{12} + \cdots + \hat{e}_{2p}x_{1p} \\ \hat{e}_{21}x_{21} + \hat{e}_{22}x_{22} + \cdots + \hat{e}_{2p}x_{2p} \\ \vdots \\ \hat{e}_{21}x_{n1} + \hat{e}_{22}x_{n2} + \cdots + \hat{e}_{2p}x_{np} \end{pmatrix} = \mathbf{X}\hat{\boldsymbol{e}}_2 \\
&\vdots \\
\boldsymbol{z}_p &= \begin{pmatrix} z_{1p} \\ z_{2p} \\ \vdots \\ z_{np} \end{pmatrix} = \begin{pmatrix} \hat{e}_{p1}x_{11} + \hat{e}_{p2}x_{12} + \cdots + \hat{e}_{pp}x_{1p} \\ \hat{e}_{p1}x_{21} + \hat{e}_{p2}x_{22} + \cdots + \hat{e}_{pp}x_{2p} \\ \vdots \\ \hat{e}_{p1}x_{n1} + \hat{e}_{p2}x_{n2} + \cdots + \hat{e}_{pp}x_{np} \end{pmatrix} = \mathbf{X}\hat{\boldsymbol{e}}_p, \qquad (2.4)
\end{aligned}$$

or, more conveniently represented as

$$\mathbf{Z}_{n\times p} = \mathbf{X}_{n\times p}\mathbf{V}_{p\times p}, \qquad (2.5)$$

where $\mathbf{V}$ holds the loading vectors of the PCA solutions in its columns.

From this information, we could leave with enough tools to compute classical PCA; as long as we have the sample variance-covariance matrix we can obtain the sample eigenvalues and eigenvectors, giving us the linear combinations we desire.

However, many sparse extensions to classical PCA were uncovered by using the connection between PCA and a well known matrix decomposition.

2.1.2 Connection to SVD and the rank-K approximation of a Matrix

Since PCA is an optimised realization of a change in basis, it's no wonder it is connected to the Singular Value Decomposition (SVD) of a matrix. SVD is a one-shot calculation in many software packages and is often used to obtain PCs and their loading vectors. Also, using only a portion of the resulting solutions from SVD, one can obtain something called the rank-K Approximation for their matrix. This is analogous to the dimension-reduction benefit of PCA, where keeping only the first few PCs can represent the original matrix well enough to justify throwing away the rest. These two topics are embedded in the Sparse PCA methodology. We will first introduce the mechanics of SVD and make the connection to PCA, then we'll show the simplicity of a rank-K approximation.

SVD

The SVD of any matrix $\mathbf{X}_{n\times p}$ with full rank $R = \min(n,p)$ is the result from finding an orthonormal basis in the row-space of $\mathbf{X}$ and an orthonormal basis in the column-space of $\mathbf{X}$ that are related in the following manner (the less than full rank case, where $\mathbf{X}$ has rank $R \leq \min(n,p)$ is simply introduced later):

$$\mathbf{X}_{n\times p}\mathbf{V}_{p\times p} = \mathbf{U}_{n\times n}\mathbf{D}_{n\times p} \tag{2.6}$$

That is, $\mathbf{X}$ transforms the orthonormal basis for its row-space, the $p \times p$ matrix $\mathbf{V}$, into the orthonormal basis for its column space, the $n \times n$ matrix $\mathbf{U}$, times an $n \times p$ scaling matrix, $\mathbf{D}$. How these three matrices are found leads directly to the connection between PCA and SVD. If we right-multiply Equation (2.6) by $\mathbf{V}'$, we arrive at the SVD of $\mathbf{X}$ itself:

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}' \tag{2.7}$$

The matrix $\mathbf{X}$ is *decomposed* into two orthonormal bases and a diagonal scaling matrix. In full detail:

$$\mathbf{U} = (\boldsymbol{u}_1 \mid \boldsymbol{u}_2 \mid \boldsymbol{u}_3 \mid \cdots \mid \boldsymbol{u}_n) = \begin{pmatrix} u_{11} & u_{21} & u_{31} & \cdots & u_{n1} \\ u_{12} & u_{22} & u_{32} & \cdots & u_{n2} \\ u_{13} & u_{23} & u_{33} & \cdots & u_{n3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ u_{1n} & u_{2n} & u_{3n} & \cdots & u_{nn} \end{pmatrix}$$

$$\mathbf{V} = (\boldsymbol{v}_1 \mid \boldsymbol{v}_2 \mid \boldsymbol{v}_3 \mid \cdots \mid \boldsymbol{v}_p) = \begin{pmatrix} v_{11} & v_{21} & v_{31} & \cdots & v_{p1} \\ v_{12} & v_{22} & v_{32} & \cdots & v_{p2} \\ v_{13} & v_{23} & v_{33} & \cdots & v_{p3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{1p} & v_{2p} & v_{3p} & \cdots & v_{pp} \end{pmatrix}$$

$$\mathbf{D} = \begin{pmatrix} d_{11} & 0 & \cdots & 0 \\ 0 & d_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & d_{pp} \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

The extra rows of zeros in $\mathbf{D}$ are simply there to fulfill its dimensional requirements during matrix multiplication with $\mathbf{U}$ and $\mathbf{V}$; this is for the case when $n > p$, whereas extra columns of zeros are required for when $n < p$.

Currently, there are too many unknown matrices but a couple of tricks will get us to a solution. To aid further explanation, let us assume $n > p$, thus $rank(\mathbf{X}) = p$; the case where $n < p$ and $rank(\mathbf{X}) = n$ can easily be understood by taking $\mathbf{X} = \mathbf{X}'$ and the square matrix case can fit under either. The unknown $\mathbf{U}$ can be eliminated by dealing with $\mathbf{X}'\mathbf{X}$ instead of $\mathbf{X}$:

$$\begin{aligned} \mathbf{X}'\mathbf{X} &= (\mathbf{UDV}')'\mathbf{UDV}' \\ &= \mathbf{VD}'\mathbf{U}'\mathbf{UDV}' \\ &= \mathbf{VD}'\mathbf{DV}' \\ &= \mathbf{VD}^2_{p\times p}\mathbf{V}' \end{aligned}$$

Since $\mathbf{X}'\mathbf{X}$ is a positive definite matrix ($\boldsymbol{a}'\mathbf{X}'\mathbf{X}\boldsymbol{a} = (\mathbf{X}\boldsymbol{a})'\mathbf{X}\boldsymbol{a} \geq 0$ for all $p \times 1$ vectors $\boldsymbol{a}$), the end result is a real, positive-valued Eigenvalue Decomposition (EVD) for the matrix $\mathbf{X}'\mathbf{X}$. This means $\mathbf{V}$ contains the normalized eigenvectors of $\mathbf{X}'\mathbf{X}$ in its columns and $\mathbf{D}^2_{p\times p}$ contains the eigenvalues of $\mathbf{X}'\mathbf{X}$ along its diagonal, largest to smallest, with 0's elsewhere. Likewise, $\mathbf{V}$ can be eliminated by dealing with $\mathbf{X}\mathbf{X}'$:

$$\begin{aligned}\mathbf{X}\mathbf{X}' &= \mathbf{U}\mathbf{D}\mathbf{V}'(\mathbf{U}\mathbf{D}\mathbf{V}')' \\ &= \mathbf{U}\mathbf{D}\mathbf{V}'\mathbf{V}\mathbf{D}'\mathbf{U}' \\ &= \mathbf{U}\mathbf{D}\mathbf{D}'\mathbf{U}' \\ &= \mathbf{U}\mathbf{D}^2_{n\times n}\mathbf{U}'\end{aligned}$$

With similar reasoning as before, $\mathbf{X}\mathbf{X}'$ is a positive semi-definite matrix, thus a real, positive-valued Eigenvalue Decomposition is realized. This means $\mathbf{U}$ contains the normalized eigenvectors of $\mathbf{X}\mathbf{X}'$ in its columns and $\mathbf{D}^2_{n\times n}$ contains the eigenvalues of $\mathbf{X}\mathbf{X}'$ along its diagonal, largest to smallest, with 0's elsewhere. Note that since $rank(\mathbf{X}) = p$ we expect only p eigenvalues along the diagonal with 0's running out the $n \times n$ matrix, thus $\mathbf{D}_{n\times n}$ is equivalent to $\mathbf{D}_{p\times p}$ but has extra 0s. Indeed, matrices $\mathbf{X}'\mathbf{X}$ and $\mathbf{X}\mathbf{X}'$ share the same eigenvalues but have different eigenvectors. To ease terminology, the terms *singular values* and *singular vectors* bring everything together in the following summary.

The columns $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ of matrix $\mathbf{U}$ are the *left-singular vectors* of $\mathbf{X}$, the columns $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p$ of matrix $\mathbf{V}$ are the *right-singular vectors* of $\mathbf{X}$, and the diagonal elements $d_{11}, \ldots, d_{pp}$ of matrix $\mathbf{D}$ are the *singular values* of $\mathbf{X}$. The terms singular and eigen are very related here. The left-singular vectors are the eigenvectors for $\mathbf{X}\mathbf{X}'$, the right-singular vectors are the eigenvectors for $\mathbf{X}'\mathbf{X}$, and the singular values are simply the square roots of the eigenvalues shared by both $\mathbf{XX'}$ and $\mathbf{X'X}$. The connection to PCA will now become apparent.

Without loss of generality, since the data sample used is usually centered along the columns, the unbiased sample variance is simplified to $\mathbf{S} = \frac{1}{(n-1)}\mathbf{X'X}$. This means $\mathbf{V}$ is derived from the eigenvectors of $(n-1)\mathbf{S}$, but the constant $(n-1)$ does not disturb the vectors, meaning $\mathbf{V}$ from SVD holds the desired sample eigenvectors (loading vectors) for the classical PCA solutions. Also, the singular values of $\mathbf{X}$, found in $\mathbf{D}$, being square-rooted eigenvalues of $\mathbf{X'X}$, give us a direct way to find the variance of each PC. The variance for each component is simply $\frac{d_{jj}^2}{(n-1)}$. Furthermore, since the sample PCs, $\boldsymbol{z_1}, \boldsymbol{z_2}, \ldots, \boldsymbol{z_p}$, can be obtained as shown by Equations (2.4) and (2.6), we can make use of relation (2.6) to note that $\mathbf{Z} = \mathbf{XV} = \mathbf{UD}$. In fact each PC can be separately expressed as $\boldsymbol{z}_j = \boldsymbol{u}_j d_{jj}$. So upon computing the SVD of a matrix $\mathbf{X}$, we can quickly collect information for classical PCA; the PCs from $\mathbf{UD}$, the loading vectors from $\mathbf{V}$, and the variances from the p diagonal elements of $\frac{\mathbf{D'D}}{(n-1)}$. If $rank(\mathbf{X}) = R < p$, there will simply be R singular values instead of p and, by looking at how the SVD matrix multiplication is handled, this requires the calculation of just R left-singular and right-singular vectors.

Before wrapping up with an example, we introduce a very important result of SVD that will be the pathway for a Sparse PCA method to come.

rank-k approximation

As mentioned before, if a significant amount of information (variance) is held within the first few PCs, we can dump the rest and trust these to be a sufficient representation of the original data. A similar concept exists for *approximating* the data matrix $\mathbf{X}$ with the SVD and it's called the 'rank-k approximation'. If $rank(\mathbf{X}) = R$ and $k < R$, the idea is to find a rank-k matrix $\hat{\mathbf{X}}$ such that it is the best possible approximation to the matrix $\mathbf{X}$ over all possible rank-K matrices. The *closeness* of $\hat{\mathbf{X}}$ to $\mathbf{X}$ is evaluated by the Frobenius Norm of their difference. The Frobenius Norm of the matrix $\mathbf{X}$ is defined as

$$\|\mathbf{X}\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{p} x_{ij}^2}.$$

Applying to the difference gives

$$\|\mathbf{X} - \hat{\mathbf{X}}\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{p}(x_{ij} - \hat{x}_{ij})^2}. \tag{2.8}$$

Solved by Eckhart and Young in 1936, minimizing Equation (2.8) requires setting $\hat{\mathbf{X}}$ to $\mathbf{X}^{(k)}$, where

$$\mathbf{X}^{(k)} = \mathbf{U}^{(k)}_{n\times k}\mathbf{D}^{(k)}_{k\times k}\mathbf{V}^{(k)}_{p\times k}{}'. \tag{2.9}$$

As the notation and dimensions suggest, $\mathbf{X}^{(k)}$ is a lower-rank version of SVD, where $\mathbf{U}^{(k)}$ holds just the first k left-singular vectors of $\mathbf{X}$ in its columns, $\mathbf{D}^{(k)}$ is a diagonal matrix with just the first k singular values, and $\mathbf{V}^{(k)}$ holds the first k right-singular vectors of $\mathbf{X}$ in its columns. In fact, Equation (2.9) can be rewritten in simpler terms:

$$\mathbf{X}^{(k)} = \sum_{j=1}^{k} d_{jj} \boldsymbol{u}_j \boldsymbol{v}_j'. \tag{2.10}$$

This rank-k approximation lays foundation for a sparse method to come in Section 2.3. To further help visualize the connection between the SVD and a rank-k approximation of a matrix $\mathbf{X}$, see Figure 2.1.

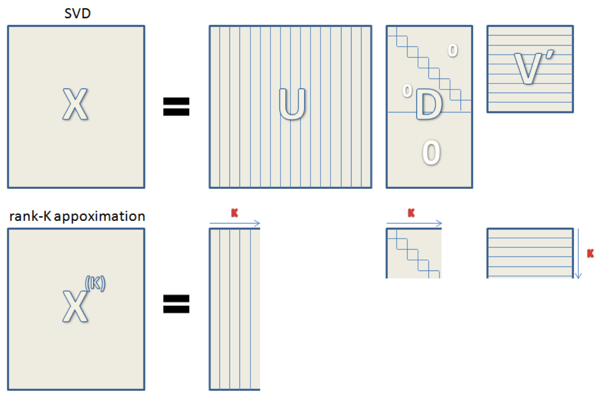


Figure 2.1: *Visualizing the SVD and rank-k approximation of a data matrix* $\mathbf{X}$

2.1.3 Example of PCA with R

Consider a University looking to give scholarships to hopeful soccer players coming out of high-school. Both grades and playing ability are important, so the University makes each of the $n = 100$ players take a series of three academic-level tests and then sends them out to the field for a scrimmage to assess their ability on the pitch. After considering their Age as well, they are faced with six variables that will help them decide who to accept: Test1, Test2, Test3, Offence, Defence, Age. The data is displayed in Table 2.1. We will use R to carry out a PCA on this data. The data is

Table 2.1: *Example Dataset: $n = 100$ Soccer Players judged on $p = 6$ variables.*

Player	T1	T2	T3	Off	Def	Age
P1	62.7	64.6	54.8	78.4	75.1	19
P2	64.2	64.6	61.5	81.6	83.5	16
P3	54.1	66.8	48.8	74.9	79.3	19
P4	48.9	62.1	60.8	87.3	86.4	19
P5	60.1	73.8	71.8	85.2	69.7	17
P6	53.5	66.1	57.8	77.3	81.7	19
⋮	⋮	⋮	⋮	⋮	⋮	
P100	58.6	66.2	57.1	71.1	78.4	17

entered into R as and looks like:

```
> head(data)
     T1   T2   T3  Off  Def Age
P1 62.7 64.6 54.8 78.4 75.1  19
P2 64.2 64.6 61.5 81.6 83.5  16
P3 54.1 66.8 48.8 74.9 79.3  19
P4 48.9 62.1 60.8 87.3 86.4  19
P5 60.1 73.8 71.8 85.2 69.7  17
P6 53.5 66.1 57.8 77.3 81.7  19
```

Centering this data along the columns, we obtain our data matrix **X**:

```
> X <- scale(data,center=T,scale=F)
> round(head(X),1)
      T1   T2    T3  Off   Def  Age
P1   3.3 -5.0  -4.5 -1.0  -5.3  1.2
P2   4.8 -5.0   2.2  2.2   3.1 -2.0
P3  -5.3 -2.8 -10.5 -4.5  -1.0  0.5
P4 -10.5 -7.6   1.5  7.9   6.1  1.2
P5   0.7  4.2  12.5  5.8 -10.6 -1.4
P6  -5.9 -3.5  -1.5 -2.1   1.3  0.7
```

The sample variance-covariance matrix **S** is computed:

```
> S <- var(X)
> round(S,2)
       T1    T2    T3   Off   Def   Age
T1  26.63 11.27 11.40 -2.35 -1.73 -0.72
T2  11.27 26.90 12.90 -1.01 -1.78  0.76
T3  11.40 12.90 25.86 -0.41  1.87  0.54
Off -2.35 -1.01 -0.41 21.92 -6.60  0.91
Def -1.73 -1.78  1.87 -6.60 18.81  0.74
Age -0.72  0.76  0.54  0.91  0.74  2.18
```

Looking at the variances for each variable first, it seems there is much greater variation in the Test and Playing Ability variables compared to Age. This makes sense because the players are all coming from high school at a similar age. Looking for correlation, it seems that the Test variables hang together quite well, but are minimally related to Playing Ability as measured by Offence and Defence. Likewise, there seems to be a correlation between Offense and Defense, but it tends to be of lesser strength and negative. All of this seems natural since academic ability and playing ability seem to be rather unrelated subject matters and one can expect Offensive players to lack in Defensive abilities, and vice versa. Age does not seem related to anything. Let's move on to see what PCA can do with this data.

We know that PCA can be found through **S** by finding eigen vectors and eigen values, or through **X** by finding its SVD. Starting with **S**, we find its eigen structure by using the function 'eigen()':

```
> eigen(S)
$values
[1] 50.362055 27.229215 17.287780 14.812789 10.630359  1.991187


$vectors
             [,1]        [,2]        [,3]        [,4]        [,5]        [,6]
[1,] -0.563635203  0.01207102  0.60771207  0.47383063  0.29328498  0.04820934
[2,] -0.590204050  0.10086834 -0.14153840 -0.71736000  0.32254973 -0.05299009
[3,] -0.573171348 -0.03618956 -0.51024890  0.31847040 -0.55518092 -0.01271738
[4,]  0.072024731  0.76689150 -0.37929919  0.30960320  0.40334986 -0.06539670
[5,]  0.015140928 -0.63255111 -0.44339913  0.25143902  0.57891638 -0.06842012
[6,] -0.005686262  0.01125630 -0.09913788 -0.01949482  0.06232541  0.99284850
```

This output gives the variances $\hat{\lambda}_1, \hat{\lambda}_2, \hat{\lambda}_3, \hat{\lambda}_4, \hat{\lambda}_5, \hat{\lambda}_6$ for each PC in the \$values component and their loading vectors $\hat{\boldsymbol{e}}_1, \hat{\boldsymbol{e}}_2, \hat{\boldsymbol{e}}_3, \hat{\boldsymbol{e}}_4, \hat{\boldsymbol{e}}_5, \hat{\boldsymbol{e}}_6$ as columns in the \$vectors component. Therefore, the \$vectors component is our desired matrix **V**. As one can check, this set of vectors are orthonormal:

```
> V <- eigen(S)$vectors
> round(t(V)%*%V,1)
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    0    0    0    0    0
[2,]    0    1    0    0    0    0
[3,]    0    0    1    0    0    0
[4,]    0    0    0    1    0    0
[5,]    0    0    0    0    1    0
[6,]    0    0    0    0    0    1
```

From this, we have the linear combinations from (2.3) as:

$$\begin{aligned}
z_1 &= -0.56x_1 - 0.59x_2 - 0.57x_3 + 0.07x_4 + 0.02x_5 - 0.01x_6 = \hat{\boldsymbol{e}}_1'\boldsymbol{x} \\
z_2 &= 0.01x_1 + 0.10x_2 - 0.04x_3 + 0.77x_4 - 0.63x_5 + 0.01x_6 = \hat{\boldsymbol{e}}_2'\boldsymbol{x} \\
z_3 &= 0.61x_1 - 0.14x_2 - 0.51x_3 - 0.38x_4 - 0.44x_5 - 0.01x_6 = \hat{\boldsymbol{e}}_3'\boldsymbol{x} \\
z_4 &= 0.47x_1 - 0.72x_2 + 0.32x_3 + 0.31x_4 + 0.25x_5 - 0.02x_6 = \hat{\boldsymbol{e}}_4'\boldsymbol{x} \\
z_5 &= 0.29x_1 + 0.32x_2 - 0.56x_3 + 0.40x_4 + 0.58x_5 + 0.06x_6 = \hat{\boldsymbol{e}}_5'\boldsymbol{x} \\
z_6 &= 0.05x_1 - 0.05x_2 - 0.01x_3 - 0.07x_4 - 0.07x_5 + 0.99x_6 = \hat{\boldsymbol{e}}_6'\boldsymbol{x} \qquad (2.11)
\end{aligned}$$

This is for a general observation. The PCs for our dataset of $n = 100$ can be calculated as in (2.4) by taking $\mathbf{XV}$:

```
> Z <- X%*%V
> colnames(Z) <- paste("PC",1:6,sep="")
> round(head(Z),1)
    PC1  PC2   PC3  PC4  PC5  PC6
P1  3.6  2.2   7.6  2.0 -1.5  2.1
P2 -0.8 -0.8   0.5  8.1  1.2 -1.9
P3 10.4 -2.8   4.6 -5.5  1.0  0.8
P4 10.2  1.3 -11.9  4.9  0.5  0.1
P5 -9.8 11.1  -3.9  0.5 -9.3 -1.4
P6  6.1 -2.8  -2.2 -1.1 -2.0  0.7
```

The variance-covariance matrix of the PCs will reveal the uncorrelated property for the PCs:

```
> round(var(Z),1)
     PC1  PC2  PC3  PC4  PC5 PC6
PC1 50.4  0.0  0.0  0.0  0.0   0
PC2  0.0 27.2  0.0  0.0  0.0   0
PC3  0.0  0.0 17.3  0.0  0.0   0
PC4  0.0  0.0  0.0 14.8  0.0   0
PC5  0.0  0.0  0.0  0.0 10.6   0
PC6  0.0  0.0  0.0  0.0  0.0   2
```

As one can see, the variances for each PC match the eigen values presented above and are in decreasing order, meaning the first PC contains the most variance, or, *information* from the data. The total variance can be calculated by taking the sum of the eigen values and one can see that it will equal the total variance of the original data $\mathbf{X}$. As well, the proportion of explained variance for each PC can be obtained:

```
> TotVAR <- sum(eigen(S)$values)
> TotVAR
[1] 122.3134
> sum(diag(S))
[1] 122.3134

> pev <- eigen(S)$values/TotVAR
> round(pev,2)
[1] 0.41 0.22 0.14 0.12 0.09 0.02
```

So within the first two PCs, about 63% of the variation is contained. If some form of regression analysis was in order, using the soccer player variables as covariates for example, perhaps taking these two PCs and moving forward would be beneficial for analysis purposes, since it would reduce the dimension of the dataset from $p = 6$ to $p = 2$ and we would no longer have collinearity amongst covariates. However, the resulting analysis, though statistically simple, would become a bit more complicated to explain because there is no tangible meaning to 'PC1' and 'PC2'; the major downfall to PCA as a pre-regression tool. In an attempt to alleviate this flaw, we look to the loading vectors, as they hold the coefficients to the linear combinations that make up the PCs.

Looking at the linear combinations in (2.11), PC1 seems to be made up almost entirely of the Test variables from the original data and PC2 has heavier weights for original Playing Ability variables. Although there is some residual contribution from the other variables, we have justification to *label* PC1 as 'Test Scores' and PC2 as 'Playing Ability'; underlying *features* of the original data. Note that these two features were realised when we first investigated the variance-covariance matrix of the original data, suggesting that PCA was able to concisely summarize our exploration in one shot. This *could* prove extremely useful in the presence of a high-dimensional data, where we have a large amount of variables, especially if we don't have any natural understanding of them. A good example of this would be for gene expression data, for which we may not have any prior insight into groupings and have a variance-covariance matrix that is very hard to wade through. However, since all loading coefficients are non-zero there may be little hope in identifying grouping structure via PCA in a high-dimensional scenario; in this $p = 6$ case with natural groups, PCA has little problem.

Instead of using 'eigen()', R has the 'prcomp()' function that accepts the data matrix $\mathbf{X}$ and offers an option to return the PCs themselve. Alternatively, we can find the SVD of $\mathbf{X}$ by using the 'svd()' function in R and arrive at the same results:

```
> svd(X)
$d
[1] 70.61051 51.92006 41.37016 38.29447 32.44080 14.04021


$u
             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
  [1,]  0.0502778  0.0431806  0.1843513 -0.0531399 -0.0471695  0.1501405
  [2,] -0.0113824 -0.0158614  0.0110987 -0.2111601  0.0355552 -0.1358298
  [3,]  0.1467968 -0.0542793  0.1121912  0.1439645  0.0306401  0.0598677
  [4,]  0.1446338  0.0246531 -0.2870746 -0.1273186  0.0140584  0.0087701
  [5,] -0.1389678  0.2141765 -0.0937469 -0.0120624 -0.2857343 -0.1016618
  [6,]  0.0863849 -0.0546677 -0.0522702  0.0287650 -0.0631722  0.0498199
  [7,] -0.1614037 -0.0294642  0.0062763  0.0409927 -0.0213736 -0.0867430
  [8,] -0.0755085  0.1297164  0.0015820 -0.0230388  0.1277839 -0.0073414
          ...        ...        ...        ...        ...        ...
[100,]  0.0440410 -0.1054557  0.1265398  0.0434240 -0.1414936 -0.0106330


$v
             [,1]        [,2]        [,3]        [,4]        [,5]        [,6]
[1,] -0.563635203  0.01207102  0.60771207 -0.47383063  0.29328498  0.04820934
[2,] -0.590204050  0.10086834 -0.14153840  0.71736000  0.32254973 -0.05299009
[3,] -0.573171348 -0.03618956 -0.51024890 -0.31847040 -0.55518092 -0.01271738
[4,]  0.072024731  0.76689150 -0.37929919 -0.30960320  0.40334986 -0.06539670
[5,]  0.015140928 -0.63255111 -0.44339913 -0.25143902  0.57891638 -0.06842012
[6,] -0.005686262  0.01125630 -0.09913788  0.01949482  0.06232541  0.99284850
```

The $u component has been shortened to save space. As seen by comparing to the eigen vectors in the first computation method, the $v component of the output holds the loading vectors for the PCs (right-singular vectors of **X**) in its columns, together forming **V**. The $d component holds the diagonal elements of the **D** matrix (the singular values of **X**) and so taking their square and dividing by $(n-1)$, as explained at the end of the SVD section, will return the variance of each PC. Also, the $u component holds the left-singular vectors in its columns, together forming **U**. Making the use of the relations at the end of the SVD section, we arrive at the same results as using the 'eigen()' function:

```
> svd <- svd(X)
> U <- svd$u
> D <- diag(svd$d)
> V <- svd$v

> diag(D%*%D/(100-1))
[1] 50.362055 27.229215 17.287780 14.812789 10.630359  1.991187

> TotVAR <- sum(diag(D%*%D/(100-1)))
> TotVAR
[1] 122.3134

> Z <- U%*%D
> colnames(Z) <- paste("PC",1:6)
> round(head(Z),1)
     PC 1 PC 2  PC 3 PC 4 PC 5 PC 6
[1,]  3.6  2.2   7.6 -2.0 -1.5  2.1
[2,] -0.8 -0.8   0.5 -8.1  1.2 -1.9
[3,] 10.4 -2.8   4.6  5.5  1.0  0.8
```

```
[4,] 10.2  1.3 -11.9 -4.9  0.5  0.1
[5,] -9.8 11.1  -3.9 -0.5 -9.3 -1.4
[6,]  6.1 -2.8  -2.2  1.1 -2.0  0.7
```

Many papers that highlight the troubles one encounters with PCA can be found in literature. See Jolliffee's dual papers in 1972 and 1973 [13, 14] and Jeffer's paper in 1967 [10] for a few classic demonstrations.

2.2 Sparseness in Statistics: Regularization Methods and Penalized Regression

Described in this section are the crucial 'Regularization' methods that help turn PCA into Sparse PCA. Keep in mind that although PCA has nothing to do with regression or predicting response variables, these methods happen to come from a regression framework, meaning a response variable $\mathbf{y}$ is introduced. It also means that our usual $\mathbf{X}$ notation will be representing the design matrix for this Section, i.e. the data matrix for predictor variables but with a leading column of ones. How PCA links to this set of methods will be introduced in the next Section; for now, we take a detour from PCA, put on our regression-hats, and explore this topic independently.

In mathematics, *regularization* is an approach to obtain a unique solution to an ill-posed problem by first adding a constraint. An ill-posed problem is defined as one without a unique solution and is found frequently in statistics. An obvious example occurs while trying to calculate OLS estimates for the true parameters of an ordinary univariate linear model when fitting $p > n$ parameters. Given data from an experiment in which n participants contributed to a targeted response variable, $\mathbf{y}$, and each of p predictor variables, the linear model including all predictors would then be:

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix}$$

or,

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where $\boldsymbol{\epsilon} \sim N(0, \sigma^2 I)$ are the error terms. To obtain OLS estimates, $\hat{\boldsymbol{\beta}}^{ols}$, one must minimize the sum of squared errors (SSE) with respect to $\boldsymbol{\beta}$. Mathematically speaking, this means

$$\hat{\boldsymbol{\beta}}^{ols} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \left(y_i - (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})\right)^2 \right\}.$$

Note that there are no restrictions on the possible values $\hat{\boldsymbol{\beta}}^{ols}$ may take on. The solution to this minimization problem is given by the well-known normal equations:

$$\hat{\boldsymbol{\beta}}^{ols} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\boldsymbol{y}.$$

Solving the set of normal equations requires the matrix $\mathbf{X}'\mathbf{X}$ to be invertible, which is impossible when $n < p$. At this stage, a generalized inverse can be used to obtain a solution but, unfortunately, does not provide a unique solution; hence recognizing the ill-posed problem. An alternative to using a generalized inverse could be the regularization approach. This involves introducing a *constraint*, or *penalty*, on some function of the β_j's which results in a unique solution and, possibly, some other nice qualities. It's also worth mentioning that $\mathbf{X}'\mathbf{X}$ is not always innocent while it is neatly invertible, either. When p is comparable in size to n, or when there exists high correlation amongst predictor variables, β_j's become very sensitive to slight fluctuations in data; with next years data, very different β_j's might be calculated. In other words, the β_j's have high variance in these cases.

The general way to introduce a constraint, or penalty, to the usual OLS regression situation is through Lagrange multipliers during the minimization of the SSE. When reading any paper on a penalized regression method, the standard notation is as follows [1]:

$$\hat{\boldsymbol{\beta}}^{pen} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}))^2 \right\}$$
$$\textit{subject to } P(\boldsymbol{\beta}) \textit{ being constrained by } t \tag{2.12}$$

with the constraint mathematically imposed via a Lagrange multiplier,

$$\hat{\boldsymbol{\beta}}^{pen} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}))^2 + \lambda P(\boldsymbol{\beta}) \right\}, \tag{2.13}$$

Here, $P(\boldsymbol{\beta})$ is the penalty function and will generally be forced below a maximum value t, i.e. $P(\boldsymbol{\beta}) \leq t$. Simultaneously, t and λ are referred to as the *tuning parameter* since they control, or tune, the impact of the constraint on the solution. Together, the penalty function being restricted by the tuning parameter creates the constraint.

Some types of constraints will merely shrink the magnitude of the parameters, whereas others will force some parameters to exactly 0. In the latter case, a solution of lower dimension is achieved; a very attractive feature for researchers interested in variable selection. Furthermore, the bias-variance trade-off has revealed that OLS estimates might not be the 'best' ones out there. The OLS estimates achieve a bias equalling 0 but at what cost? To minimize the mean squared error (MSE), the commonly accepted criteria to assess parameter estimates, one must consider both the bias *and* the variance of the estimate. Perhaps sacrificing the unbiased property for a significant decrease in variance will provide a superior set of estimates. A typical visual displaying the bias-variance trade-off can be found in Figure 2.2. We will now introduce a variety of constraints along with their pros, cons, and technical details. The literature available on regularization in statistics is far more extensive, but the following will suffice in preparation for the Sparse PCA methods considered in this thesis. It is interesting to note that the constraint functions hinge around the L_q-norm of $\boldsymbol{\beta}$. Setting different values of q will determine the properties of the resulting penalizing method.

2.2.1 Ridge and LASSO penalties

In 1970, Arthur Hoerl introduced an L_2-norm constraint on $\boldsymbol{\beta}$ [8]. This formed what is called Ridge regression, where the penalty is called the Ridge penalty. His

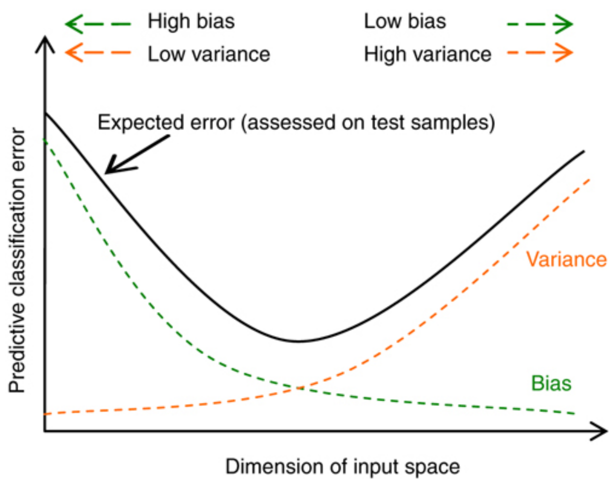


Figure 2.2: *Visualizing the Bias-Variance Trade-off: drastically lower variance may be achieved by sacrificing the unbiased property of an estimate. The MSE is a more fair representation of how 'good' an estimate is as it takes both variance and bias into account.* [Source: retrieved from Rudy A. Gideon's presentation "Regularization: Ridge Regression and the LASSO", found online at http://www-stat.stanford.edu/owen/courses/305/]

intent was to provide a biased estimation procedure to attain superior parameter estimates via the bias-variance trade-off. It was specifically aimed to handle the case where $\mathbf{X}'\mathbf{X}$ was severely non-orthogonal, i.e. there was much collinearity amongst the predictors. The penalty function is $P(\boldsymbol{\beta}) = \sum_{j=1}^{p} |\beta_j|^2$ and is constrained by a maximum of t. Taking on the notational conventions of Equations (2.12) and (2.13),

$$\hat{\boldsymbol{\beta}}^{ridge} = \underset{\boldsymbol{\beta}}{\text{argmin}} \left\{ \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}))^2 \right\} \;\; \textit{subject to} \; \sum_{j=1}^{p} \beta_j^2 < t, \tag{2.14}$$

with the constraint mathematically introduced via a Lagrange multiplier, giving

$$\hat{\boldsymbol{\beta}}^{ridge} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}))^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}. \tag{2.15}$$

To get familiar with a quicker notation, Equation (2.15) can be written as

$$\hat{\boldsymbol{\beta}}^{ridge} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2 \right\}. \tag{2.16}$$

As $t \to 0$ ($\lambda \to \infty$), the magnitudes of the parameters are increasingly restricted, shrinking estimates towards 0. As $t \to \infty$ ($\lambda \to 0$), the magnitudes of parameters become unconstrained and the solution for Equation (2.16) becomes the OLS estimates. Specifically, if $t \geq \sum_{j=1}^{p} |\beta_j^{ols}|^2$, the OLS estimates are the solution. Since this acts as an upper-bound for allowing sparseness, a convenient notation of $s = \frac{t}{\sum_{j=1}^{p} |\beta_j^{ols}|^2}$ provides a scaled way of referring to the constraint; s ranging from 0 to 1 instead of some scenario-dependent value.

Not only do the ridge solutions take advantage of the bias-variance trade-off, but they also provide a unique solution even in the $n < p$ case; thus offering a work-around to the ill-posed problem encountered at the normal equations. In fact, the solution can be written out in closed form:

$$\hat{\boldsymbol{\beta}}^{ridge} = (\mathbf{X}'\mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}' \boldsymbol{y}.$$

However, researchers with a desire for variable selection and model interpretation realize an unattractive property. Ridge estimates shrink as the tuning parameter constrains the estimates, but they are not forced *directly* to 0. Thus a sparse solution is not attained.

In 1996, Robert Tibshirani successfully introduced an L_1-norm constraint on the β_j's [19]. This formed what is called LASSO (Least Angle Shrinkage and Selection Operator) regression, where the penalty is called the LASSO penalty. In the usual notation,

$$\hat{\boldsymbol{\beta}}^{lasso} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}))^2 \right\} \quad \textit{subject to} \ \sum_{j=1}^{p} |\beta_j| < t \tag{2.17}$$

is spun mathematically as

$$\hat{\boldsymbol{\beta}}^{lasso} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}))^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| \right\}, \tag{2.18}$$

or equivalently as

$$\hat{\boldsymbol{\beta}}^{lasso} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 \right\}. \tag{2.19}$$

This unleashed a new weapon for researchers interested in variable selection in high-dimensional data. The LASSO method is able to send some of the $\hat{\beta}_j$'s directly to 0, thus attaining a sparse solution. In the same paper, Tibshirani provides some visuals in the $p = 2$ dimensional case to show how Ridge regression and LASSO regression obtain differing qualities. Figure 2.3 shows possible solutions to the minimization problem along with Lasso (a) and Ridge (b) constraint regions. The ellipses are level curves for the function to be minimized in the original SSE but the solution must lie within the shaded region if enforcing the constraint. One can readily see how the LASSO will allow for estimates equalling exactly to 0 (when the level curves touch the axes). Another handy visual for sparse regression procedures is a plot of the entire

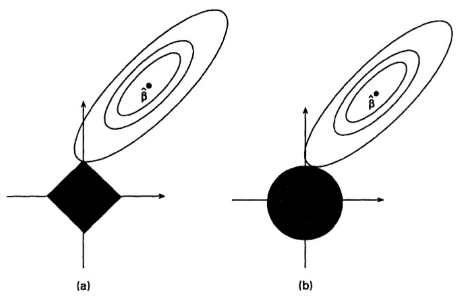


Fig. 2. Estimation picture for (a) the lasso and (b) ridge regression

Figure 2.3: *2-dimensional example of LASSO (a) and Ridge (b) constraints. The solutions obtained with the LASSO constraint can produce 0-valued estimates, not the case with the Ridge constraint.* [Source: retrieved from Robert Tibshirani's paper [19].]

solution path. Figure (2.4) displays that all parameters will be forced to 0 when $s \to 0$ and how solutions become less sparse when $s \to 1$, eventually achieving the non-sparse OLS solutions when $s = 1$. The image exemplifies the impact that tuning parameter selection has in dictating how sparse the solution is and which variables will be concluded as important. All this said, the LASSO is not a perfect option for variable selection. For starters, the solution cannot be expressed in closed form. Instead, algorithms have been employed in attempt to efficiently solve the entire LASSO path. The Least Angle Regression (LARS) algorithm, originally intended to develop an improved forward step-wise selection procedure, conceptually applied to solving the LASSO problem in an efficient way [5]. This made the LASSO technique

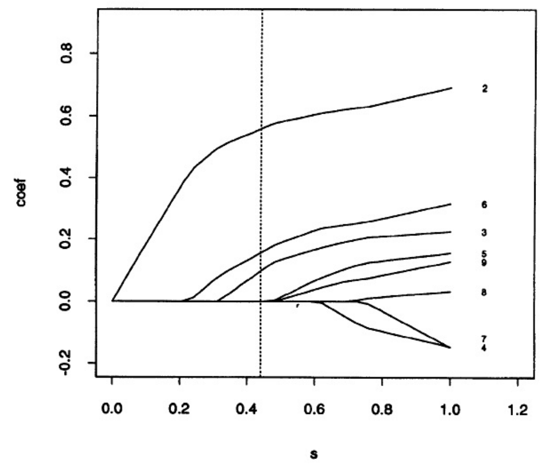


Figure 2.4: *The solution path as the tuning parameter s is ranged from 0 to 1. Selecting the tuning parameter is of primary importance.* [Source: retrieved from Robert Tibshirani's paper [19].]

more computationally feasible. Another drawback of the LASSO is that it can only offer $\min(n, p)$ parameters in its solution. Unlike the Ridge, it does not thrive when $n < p$; a major downfall for researchers with very high-dimensional data.

Swiftly following the attraction of the LASSO method came a more flexible and beneficial procedure.

2.2.2 Elastic Net penalty

The Ridge and LASSO penalties have their respective benefits; they can independently generate improved analyses. However they also have their respective flaws, as discussed. In 2005, Zou and Hastie introduced the Elastic Net penalty which was simply a combination of Ridge and LASSO penalties [22]. It draws the benefits from both sources and mathematically generalizes the formulation as

$$\hat{\boldsymbol{\beta}}^{en} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}))^2 + \lambda \sum_{j=1}^{p} \beta_j^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| \right\}, \tag{2.20}$$

or, equivalently,

$$\hat{\boldsymbol{\beta}}^{en} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 \right\}. \tag{2.21}$$

Now, not only can one obtain a sparse solution with the LASSO penalty, but they can also battle the $n < p$ scenario with the help of the Ridge penalty. The Elastic Net is now widely being explored in developing procedures, including generalized linear model extensions; see R package 'glmnet' in the CRAN repositories.

2.2.3 Selecting the tuning parameter

With a continuous-scale tuning parameter, there becomes infinitely many choices for the value. Since it controls the level of sparsity and, in turn, the parameter estimates, their MSE, and the conclusions we draw, it is of great interest to examine how results behave in reaction to its variation. Because sparsity and improved estimation (via introducing bias) are at the heart of these methods, it seems natural to obtain a justified choice of the tuning parameter based on these criteria. If interested in sparsity, one could increase the Lagrange form parameter(s) to a point where desired sparsity is realized. If interested in improved estimation, one could choose the parameter based on the MSE of the estimates produced. However, as with most cases in statistics, there is most likely a balance that will be of preferred choice. For example, selecting the most sparse solution possible will probably suffer from a large MSE; perhaps sacrificing a little sparsity for significant gains in estimation is justified. Likewise, selecting the solution with lowest MSE might not be sparse enough; variable selection is a tough task here. This tug-of-war has been named the 'variance-sparsity' trade-off [23]. Since this topic is strongly dependent on the scenario, various methods have been suggested to select a balance. They do, however, mesh together with a similar tool.

Cross-validation is a technique used across many areas in statistics when one is interested in determining how well a model built from the sample at hand will apply to new data. Typically, the given sample is partitioned into a *training* sample and *validation* sample. The model is built from the training data and its performance is checked with the validation data; if the model performs as well on the validation data as it did with the training, then we believe it is a *stable* model. The measured performance is usually based on some criteria involving prediction. A spin-off to traditional cross-validation is 'K-Fold Cross-validation'. This involves partitioning the original sample into K samples. Each sample takes a turn being the validation data for a model built from the remaining $K-1$ samples (cumulatively acting as the training sample). After retrieving K performance measures, they can be summarized and therefore trusted more than the single one obtained from a traditional cross-validation technique. Of course, sample size must be sufficiently large to allow for more folds and the more folds chosen, the more computationally expensive the procedure.

This general method to select tuning parameters will be employed in the Sparse PCA methods to come.

2.3 Sparse PCA methods

Classical PCA introduced in Section 2.1 and the regularization methods introduced in Section 2.2 have very promising applications; their extensive use in many fields a testament to their performance. PCA seems to fall short when attempting to interpret its output, especially in high data-dimensions. Imagine PCA output where the first few PCs are derived from linear combinations of small and distinct groups of the original variables. In other words, imagine most of the loadings, that usually have negligible residual contribution, were to be sent directly to 0. This would enable amazing data exploration and give meaning to the dimension-reduction steps; perhaps we could truly associate each PC with an underlying feature in the data. There has been a track-record of attempts to make the resulting PCs more interpretable (examples: [15, 20]) but only with the latest efforts has a new class of Sparse PCA methods hit the scene by making use of regularization methods.

Simply put, Sparse PCA methods adjust the PCA method to inject sparseness to the loading vectors just as the regularization methods inject sparseness to the parameter estimates in the regression setting. This means that PCs will now have a chance to be meaningful. Furthermore, it seems more realistic for variables called *principal components* to have latent meaning via distinct groups of the original variables rather than a nameless presence due to a conglomerative mess. After surveying the literature, we have decided to compare three of the well-established methods that come along with R-packages and R-code necessary to implement their procedures. They will be compared in Chapter 3, but for now the methodology is laid out in a format compatible with the original papers but with notation to match this thesis.

2.3.1 SPCA formulation by Zou et al.

In 2006, Hui Zou, Trevor Hastie, and Robert Tibshirani published a paper introducing their new method of imposing sparseness to PCA [23]. For simple reporting, we give their method the name 'SPCA'. Sparseness, as we've seen so far, is introduced through a regression equation when attempting to predict response variable Y from a set of predictors $X_1, \ldots, X_p$. PCA, however, forgets about the response variables and only deals with the predictors. The methods from their paper bridge the gap between a regression scenario and a PCA scenario; they reform the PCA as a regression problem.

First, the authors introduce a very intuitive way to impose constraints through a regression-based formulation. Later on, they provide a one-shot, efficient algorithm for solving the entire fit which outclasses the intuitive way, but it is worth starting from the ground up to provide a clear foundation to how PCA is spun as a regression problem.

Based on the SVD of a data matrix $\mathbf{X}$, the j^{th} principal component can be expressed as $\boldsymbol{z}_j = \boldsymbol{u}_j d_{jj}$, as shown in Section 2.1.2. Realizing these $\boldsymbol{z}_j$'s are linear combinations of the original variables, the coefficients (loadings) of the linear combinations may be estimated through a regression procedure. This, in turn, allows the addition of constraints to the estimation procedure and the opportunity to obtain sparseness amongst the loadings vectors; a chance to achieve an interpretable format of PCA. To begin, the authors introduce a theorem that shows the link between PCA and regression:

Theorem 1. *For each j, denote by $\boldsymbol{z}_j = \boldsymbol{u}_j d_{jj}$ the j^th principal component. Consider a positive λ and the ridge estimates $\hat{\boldsymbol{\beta}}^{ridge}$ given by*

$$\hat{\boldsymbol{\beta}}^{ridge} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \|\boldsymbol{z}_j - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2. \tag{2.22}$$

Let $\hat{\boldsymbol{v}} = \frac{\hat{\boldsymbol{\beta}}^{ridge}}{\|\hat{\boldsymbol{\beta}}^{ridge}\|_2}$, then $\hat{\boldsymbol{v}} = \boldsymbol{v}_j$.

Without the ridge penalty this theorem shows the regression way of approximating PCs; already explored by at least Cadima and Jolliffe in 1995 [3]. The ridge penalty that Zou et al. introduce now allows the reconstruction of PCs even when $n < p$. Naturally following this, they suggest to add the LASSO penalty to inject sparseness to the loadings, resulting in an elastic net formulation from Section 2.2.2:

$$\hat{\boldsymbol{\beta}}^{enet} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \|\boldsymbol{z}_j - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2 + \lambda_{1,j}\|\boldsymbol{\beta}\|_1. \tag{2.23}$$

However the flaw to this technique is that we require the original PCs, $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_p$, to get this approximation. The authors then introduce their primary method, namely the SPCA criterion, which derives their sparse PCs without the need for the original PCs. Again, some Theorems are declared:

With $\mathbf{x}_i$ representing the usual i^th row vector of a data matrix $\mathbf{X}$,

Theorem 2. *For any $\lambda > 0$, let*

$$\left(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}\right) = \underset{\boldsymbol{\alpha},\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \|\mathbf{x}_i - \boldsymbol{\alpha}\boldsymbol{\beta}'\mathbf{x}_i\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2 \right\}$$
$$\text{subject to } \|\boldsymbol{\alpha}\|_2^2 = 1. \tag{2.24}$$

Then $\hat{\boldsymbol{\beta}} \propto \boldsymbol{v}_1$.

Theorem 3. *Suppose we are considering the first k principal components. Let $\mathbf{A}_{p\times k} = (\boldsymbol{\alpha_1}, \ldots, \boldsymbol{\alpha_k})$ and $\mathbf{B}_{p\times k} = (\boldsymbol{\beta_1}, \ldots, \boldsymbol{\beta_k})$. For any $\lambda > 0$, let*

$$\left(\hat{\mathbf{A}}, \hat{\mathbf{B}}\right) = \underset{\mathbf{A},\mathbf{B}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{A}\mathbf{B}'\mathbf{x}_i\|_2^2 + \lambda \sum_{j=1}^{k} \|\boldsymbol{\beta}_j\|_2^2 \right\}$$
$$\text{subject to } \mathbf{A}'\mathbf{A} = \mathbf{I}_{k\times k} \tag{2.25}$$

Then $\hat{\boldsymbol{\beta}} \propto \boldsymbol{v}_j$ for $j = 1, 2, \ldots, k$.

Proofs for the above Theorems are provided by Zou et al. in their paper. With the same motivations as proceeding Theorem 1, the authors propose to add a LASSO penalty to (2.25), injecting sparseness to the resulting PCs. This concludes their SPCA criterion:

$$\left(\hat{\mathbf{A}}, \hat{\mathbf{B}}\right) = \underset{\mathbf{A},\mathbf{B}}{\operatorname{argmin}} \left\{ \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{A}\mathbf{B}'\mathbf{x}_i\|_2^2 + \lambda \sum_{j=1}^{k} \|\boldsymbol{\beta_j}\|_2^2 + \sum_{j=1}^{k} \lambda_{1,j} \|\boldsymbol{\beta}_j\|_1 \right\}$$
$$\text{subject to } \mathbf{A}'\mathbf{A} = \mathbf{I}_{k\times k}. \tag{2.26}$$

Note that the same ridge penalty tuning parameter, λ, is used all loading vectors but different LASSO penalty tuning parameters, $\lambda_{1,j}$, may be chosen for each of the k specified loading vectors.

In order to minimize this criterion, the authors provide an alternating algorithm that will converge to a solution.

B given A:

For each j, let $Y_j^* = \mathbf{X}\boldsymbol{\alpha}_j$. From some initial insights in their paper, Zou et al. establish that $\mathbf{B} = [\hat{\boldsymbol{\beta}}_{\mathbf{1}}, \dots, \hat{\boldsymbol{\beta}}_k]$, where each $\hat{\boldsymbol{\beta}}_j$ is an elastic net estimate:

$$\hat{\boldsymbol{\beta}}_{\boldsymbol{j}} = \underset{\boldsymbol{\beta_j}}{\operatorname{argmin}} \left\{ \|Y_j^* - \mathbf{X}\boldsymbol{\beta_j}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2 + \sum_{j=1}^{k} \lambda_{1,j}\|\boldsymbol{\beta_j}\|_1 \right\} \tag{2.27}$$

A given B:

With $\mathbf{B}$ is fixed, we can ignore the penalty part in 2.26 and just minimize $\sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{A}\mathbf{B}'\mathbf{x}_i\|_2^2 = \|\mathbf{X} - \mathbf{X}\mathbf{B}\mathbf{A}'\|_F^2$, subject to $\mathbf{A}'\mathbf{A} = \mathbf{I}_{k\times k}$. The authors show that the solution is obtained by using the reduced rank form of the *Procrustes rotation*, which is given in their paper. We then compute the SVD

$$(\mathbf{X}'\mathbf{X})\mathbf{B} = \mathbf{U}\mathbf{D}\mathbf{V}' \tag{2.28}$$

and set $\hat{\mathbf{A}} = \mathbf{U}\mathbf{V}'$.

A direct algorithm more compatible with coding languages is then provided:

Algorithm for obtaining solutions to SPCA criterion:

1. Let $\mathbf{A}$ start at $\mathbf{V}[, 1:k]$, the loadings of the first k ordinary principal components.

2. Given a fixed $\mathbf{A} = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_k]$, solve the following elastic net problem for $j = 1, \ldots, k$

$$\boldsymbol{\beta}_j = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ (\boldsymbol{\alpha}_j - \boldsymbol{\beta})'\mathbf{X}'\mathbf{X}(\boldsymbol{\alpha}_j - \boldsymbol{\beta}) + \lambda\|\boldsymbol{\beta}\|_2^2 + \lambda_{1,j}\|\boldsymbol{\beta}\|_1 \right\}. \tag{2.29}$$

3. For a fixed $\mathbf{B} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_k]$, compute the SVD of $\mathbf{X}'\mathbf{X}\mathbf{B} = \mathbf{U}\mathbf{D}\mathbf{V}'$, then update $\mathbf{A} = \mathbf{U}\mathbf{V}'$.

4. Repeat Steps 2-3, until convergence.

5. Normalization: $\hat{\boldsymbol{v}}_j = \frac{\boldsymbol{\beta}_j}{\|\boldsymbol{\beta}_j\|_2}$, $j = 1, \ldots, k$.

Zou et al. provide a useful summary of preliminary findings that suggest how to choose values for the tuning parameters, λ and $\lambda_{1,j}$s. For the LASSO penalty tuning parameters $\lambda_{1,j}$s: higher choices of the $\lambda_{1,j}$s produce more sparse loading vectors, but perhaps at the cost of estimation accuracy. This could be referred to as the variance-sparsity trade-off. The authors suggest to use a K-fold cross-validation method to select appropriate values based on sustaining good estimation of PCs from classical PCA by the original data; as per Equation (2.23). For the ridge penalty tuning parameter λ: the algorithm returns similar results when ranging λ. Since the ridge penalty is usually needed to handle datasets of larger dimension, specifically $n < p$, it is not needed that much for the simpler case of $n > p$. Following this, they suggest to keep the ridge penalty at a very small number (e.g. 10^{-6}) to still enable handling of collinearity problems in the $n > p$ dataset. For $n << p$ datasets, they provide an adjustment to the algorithm above that reflects choosing $\lambda \to \infty$; they specifically mention high dimensional gene expression arrays as an example case to employ this adjustment. A theorem is needed.

Theorem 4. *Let $\hat{\boldsymbol{v}}_j(\lambda) = \frac{\hat{\boldsymbol{\beta}}_j}{\|\hat{\boldsymbol{\beta}}_j\|_2}$, for $j = 1, \ldots, k$, be the loadings derived from the SPCA criterion. Let $(\hat{\mathbf{A}}, \hat{\mathbf{B}})$ be the solution of the optimization problem*

$$(\hat{\mathbf{A}}, \hat{\mathbf{B}}) = \underset{\mathbf{A},\mathbf{B}}{\operatorname{argmin}} \left\{ -2tr(\mathbf{A}'\mathbf{X}'\mathbf{X}\mathbf{B}) + \sum_{j=1}^{k} \|\boldsymbol{\beta}_j\|_2^2 + \sum_{j=1}^{k} \lambda_{1,j} \|\boldsymbol{\beta}_j\|_1 \right\}$$
$$\textit{subject to } \mathbf{A}'\mathbf{A} = \mathbf{I}_{k\times k}. \qquad (2.30)$$

When $\lambda \to \infty$, $\hat{\boldsymbol{v}}_j(\lambda) \to \frac{\hat{\boldsymbol{\beta}}_j}{\|\hat{\boldsymbol{\beta}}_j\|_2}$.

The second step to the algorithm is simplified since it achieves closed form when $\lambda \to \infty$. The following algorithm incorporates this adjustment as seen in Equation (2.31) below.

Algorithm for obtaining SPCA solutions when $n << p$:

1. Let $\mathbf{A}$ start at $\mathbf{V}[, 1 : k]$, the loadings of the first k ordinary principal components.

2. For $j = 1, \ldots, k$

$$\boldsymbol{\beta}_j = \left(|\boldsymbol{\alpha}_j'\mathbf{X}'\mathbf{X}| - \frac{\lambda_{1,j}}{2} \right)_+ Sign(\boldsymbol{\alpha}_j'\mathbf{X}'\mathbf{X}). \qquad (2.31)$$

3. For a fixed $\mathbf{B} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_k]$, compute the SVD of $\mathbf{X}'\mathbf{X}\mathbf{B} = \mathbf{U}\mathbf{D}\mathbf{V}'$, then update $\mathbf{A} = \mathbf{U}\mathbf{V}'$.

4. Repeat Steps 2-3, until convergence.

5. Normalization: $\hat{\boldsymbol{v}}_j = \frac{\boldsymbol{\beta}_j}{\|\boldsymbol{\beta}_j\|_2}$, $j = 1, \ldots, k$.

Equation (2.31) is referred to as *soft-thresholding*, where the $(.)_+$ operation takes the inside value if it is positive, but 0 otherwise. In this case, $\lambda_{1,j}/2$ acts as the *threshold*.

Unfortunately, adding sparseness to the loading vectors involves sacrificing the uncorrelated property of the resulting PCs. Zou et al. propose that the usual additive property of their variances is no longer a viable option for calculating the total variance; part of the variance of $\boldsymbol{z}_{j+1}$ will be explained in the previous j PCs if they are correlated with it. With this said, the authors provide a alternative method for calculating the variances. It employs regression projections to remove linear dependence between PCs and therefore handles the correlation issues. Details can be found in their paper.

R-package: 'elasticnet'

The authors provide an R-package named 'elasticnet' that can be installed through CRAN. With regards to the SPCA methods just explained, the most relevant function within this package is 'spca()'. It carries out the solutions to the SPCA criterion in Equation (2.26) and delivers the sparse loading vectors along with the variance explained for each successive PC. The 'arrayspc()' function employs the altered algorithm suggested for when $n << p$. The 'cv.enet()' function allows us to perform the K-fold cross-validation method described above to collect an appropriate set of LASSO penalty tuning parameters.

It is handy to point out that, although they never discussed this option in their paper, Zou et al. include an argument in the 'spca()' function that allows specification of the number of non-zero loadings to have in each resulting sparse loading vector. This specification could be very subjective in practice, but would certainly be direct if prior knowledge of the dataset gives reason to select certain numbers.

2.3.2 PMDSPCA formulation by Witten et al.

In 2009, Witten et al. introduced Penalized Matrix Decomposition (PMD); a method transcribed from the relationship between the Frobenius Norm approximation of a matrix and SVD components by Eckart and Young's 1936 paper. For simple reporting, we give their method the name 'PMDSPCA'. PMD is a very general framework that shows how to impose constraints ("Penalized") when approximating a matrix via the rank-k approximation derived from the SVD of a matrix $\mathbf{X}$ ("Matrix Decomposition"). The constraints can apply to the set of K left-singular vectors, $\mathbf{U}^{(k)}$, as well as the set of k right-singular vectors, $\mathbf{V}^{(k)}$, of $\mathbf{X}$ which is why this PMD method is a more general framework that has more applications than just Sparse PCA. We'll start by sharing the PMD method of Witten et al in its entirety but then focus on the Sparse PCA application.

For our usually centered data matrix $\mathbf{X}$ of rank $R \leq \min(n, p)$, let the SVD be $\mathbf{UDV'}$ and rank-k approximation be $\mathbf{X}^{(k)} = \sum_{j=1}^{k} d_{jj}\boldsymbol{u}_j\boldsymbol{v}_j'$, as established in Section 2.1.2. Then we know that $\mathbf{X}^{(k)}$ is the best rank-k approximation of $\mathbf{X}$ when judged by the Frobenius norm of the difference. The authors begin with the rank-1 approximation and consider the optimization problem:

$$\underset{d,\boldsymbol{u},\boldsymbol{v}}{\text{minimize}}\left\{\frac{1}{2}\|\mathbf{X}-d\boldsymbol{u}\boldsymbol{v}'\|_F^2\right\}\ \textit{subject to}\ \|\boldsymbol{u}\|_2^2=1,\ \|\boldsymbol{v}\|_2^2=1,\ P_1(\boldsymbol{u})\le c_1,\ P_2(\boldsymbol{v})\le c_2, d\ge 0. \tag{2.32}$$

The penalty functions, $P_1(\boldsymbol{u})$ and $P_2(\boldsymbol{u})$, can be of any form but the authors focus on two types: the LASSO and the Fused LASSO. The LASSO penalty is familiar to us now and the Fused LASSO is simply an extension to the LASSO that attempts to force loadings from grouped variables to a single value to enforce group recognition [7]. It is not used when PMD is applied to Sparse PCA. The values of c_1 and c_2 are the tuning parameters, analogous to the role of t in Section 2.2.

Attempting to solve (2.32), Witten et al make use of a theorem.

Theorem 5. *Let $\mathbf{U}^{(k)}$ and $\mathbf{V}^{(k)}$ be $n \times k$ and $p \times k$ orthogonal matrices and $\mathbf{D}^{(k)}$ a diagonal matrix with diagonal elements d_k. Then,*

$$\frac{1}{2}\|\mathbf{X}-\mathbf{U}^{(k)}\mathbf{D}^{(k)}\mathbf{V}^{(k)\prime}\|_F^2=\frac{1}{2}\|\mathbf{X}\|_F^2-\sum_{j=1}^{k}\boldsymbol{u}_j'\mathbf{X}\boldsymbol{v}_j d_{jj}+\frac{1}{2}\sum_{j=1}^{k}d_{jj}^2. \tag{2.33}$$

Applying this theorem to the rank-1 case, the values of $\boldsymbol{u}$ and $\boldsymbol{v}$ that solve (2.32) also solve:

$$\underset{\boldsymbol{u},\boldsymbol{v}}{\text{maximize}}\{\boldsymbol{u}'\mathbf{X}\boldsymbol{v}\}\quad \textit{subject to}\ \|\boldsymbol{u}\|_2^2=1,\ \|\boldsymbol{v}\|_2^2=1,\ P_1(\boldsymbol{u})\le c_1,\ P_2(\boldsymbol{v})\le c_2, \tag{2.34}$$

with $d=\boldsymbol{u}'\mathbf{X}\boldsymbol{v}$. By tweaking the L_2-norm constraints on $\boldsymbol{u}$ and $\boldsymbol{v}$, they arrive at the rank-1 PMD:

$$\underset{\boldsymbol{u},\boldsymbol{v}}{\text{maximize}}\{\boldsymbol{u}'\mathbf{X}\boldsymbol{v}\} \quad \textit{subject to } \|\boldsymbol{u}\|_2^2 \leq 1,\ \|\boldsymbol{v}\|_2^2 \leq 1,\ P_1(\boldsymbol{u}) \leq c_1,\ P_2(\boldsymbol{v}) \leq c_2, \quad (2.35)$$

This is now a *biconvex* function of $\boldsymbol{u}$ and $\boldsymbol{v}$, which means an iterative algorithm can be obtained to provide solutions; leading to the following algorithm.

Algorithm for Computing a Single-Factor PMD model:

1. Initialize $\boldsymbol{v}$ to have an L_2-norm of 1.
2. Iterate until convergence:
   (a) $\boldsymbol{u} \leftarrow \underset{\boldsymbol{u}}{\text{argmin}}\{\boldsymbol{u}'\mathbf{X}\boldsymbol{v}\}$ *subject to* $P_1(\boldsymbol{u}) \leq c_1$ *and* $\|\boldsymbol{u}\|_2^2 \leq 1$.
   (b) $\boldsymbol{v} \leftarrow \underset{\boldsymbol{v}}{\text{argmin}}\{\boldsymbol{u}'\mathbf{X}\boldsymbol{v}\}$ *subject to* $P_2(\boldsymbol{v}) \leq c_2$ *and* $\|\boldsymbol{v}\|_2^2 \leq 1$.
3. $d \to \boldsymbol{u}'\mathbf{X}\boldsymbol{v}$.

To obtain multiple factors, the authors provide a simple framework that is dependent on finding iterative residuals:

Algorithm for Computing a Multiple(k)-Factor PMD model:

1. Let $\mathbf{X}^1 \leftarrow \mathbf{X}$.
2. For $j = 1, \ldots, k$:
   (a) Find $\boldsymbol{u}_j$, $\boldsymbol{v}_j$, and d_{jj} by applying the Single-Factor PMD algorithm to data $\mathbf{X}^j$.

(b) $\mathbf{X}^{j+1} \to \mathbf{X}^{j} - d_{jj}\boldsymbol{u}_j\boldsymbol{v}_j'$.

These two algorithms are for a general case. The authors introduce a few special cases, but we will only cover the ones necessary for preparing the Sparse PCA method to come. As stated before, they are interested in adding LASSO constraints. The notation PMD(L_1,L_1) refers to setting both $P_1(\boldsymbol{u})$ and $P_2(\boldsymbol{v})$ from the rank-1 PMD (2.35) to LASSO penalties. Therefore, the PMD(L_1,L_1) criterion is

$$\underset{\boldsymbol{u},\boldsymbol{v}}{\text{maximize}}\,\{\boldsymbol{u}'\mathbf{X}\boldsymbol{v}\} \quad \textit{subject to } \|\boldsymbol{u}\|_2^2 \leq 1,\ \|\boldsymbol{v}\|_2^2 \leq 1,\ \|\boldsymbol{u}\|_1 \leq c_1,\ \|\boldsymbol{v}\|_1 \leq c_2. \quad (2.36)$$

Witten et al. provide visual evidence that the tuning parameters, c_1 and c_2, are restricted to specific ranges under this combination of penalties; $1 \leq c_1 \leq \sqrt{n}$ and $1 \leq c_2 \leq \sqrt{p}$. The solution to the PMD(L_1,L_1) criterion requires knowledge of a Lemma involving the soft thresholding operator S, such that S(a,c)=sign(a).

Lemma 1. *Consider the optimization problem*

$$\underset{\boldsymbol{u}}{\text{maximize}}\,\{\boldsymbol{u}'\boldsymbol{a}\} \quad \textit{subject to } \|\boldsymbol{u}\|_2^2 \leq 1,\ \|\boldsymbol{u}\|_1 \leq c. \quad (2.37)$$

The solution satisfies $\boldsymbol{u} = \frac{S(\boldsymbol{a},\Delta)}{\|S(\boldsymbol{a},\Delta)\|_2^2}$, *with* $\Delta = 0$ *if this results in* $\|\boldsymbol{u}\|_1 \leq c$; *otherwise,* Δ *is chosen so that* $\|\boldsymbol{u}\|_1 = c$.

Algorithm for Computing a Single-Factor PMD(L_1,L_1):

1. Initialize $\boldsymbol{v}$ to have an L_2-norm of 1.

2. Iterate until convergence:

   (a) $\boldsymbol{u} \leftarrow \frac{S(\mathbf{X}\boldsymbol{v},\Delta_1)}{\|S(\mathbf{X}\boldsymbol{v},\Delta_1)\|_2^2}$, where $\Delta_1 = 0$ if this results in $\|\boldsymbol{u}\|_1 \leq c_1$; otherwise, Δ_1 is chosen to be a positive constant such that $\|\boldsymbol{u}\|_1 = c_1$.

   (b) $\boldsymbol{v} \leftarrow \frac{S(\mathbf{X}'\boldsymbol{u},\Delta_2)}{\|S(\mathbf{X}'\boldsymbol{u},\Delta_2)\|_2^2}$, where $\Delta_2 = 0$ if this results in $\|\boldsymbol{v}\|_1 \leq c_2$; otherwise, Δ_2 is chosen to be a positive constant such that $\|\boldsymbol{v}\|_1 = c_2$.

3. $d \rightarrow \boldsymbol{u}'\mathbf{X}\boldsymbol{v}$.

The Sparse PCA method that arrives comes from removing the penalty from $\boldsymbol{u}$, leaving only a penalty on $\boldsymbol{v}$; naturally, since the connection between PCA loadings and SVD come through the matrix $\mathbf{V}$. The so-called 'SPC criterion' is equivalent to an PMD(.,L_1) criterion as follows:

$$\underset{\boldsymbol{u},\boldsymbol{v}}{\text{maximize}}\,\{\boldsymbol{u}'\mathbf{X}\boldsymbol{v}\} \quad \textit{subject to } \|\boldsymbol{u}\|_2^2 \leq 1,\ \|\boldsymbol{v}\|_2^2 \leq 1,\ \|\boldsymbol{v}\|_1 \leq c_2. \tag{2.38}$$

The solution to the SPC criterion is achieved by relaxing the soft-thresholding during the update step for $\boldsymbol{u}$ in the algorithm for solving the PMD(L_1,L_1).

Algorithm for Computing a Single-Factor PMD(.,L_1) - The Sparse PCA method:

1. Initialize $\boldsymbol{v}$ to have an L_2-norm of 1.

2. Iterate until convergence:

   (a) $\boldsymbol{u} \leftarrow \frac{\mathbf{X}\boldsymbol{v}}{\|\mathbf{X}\boldsymbol{v}\|_2^2}$.

   (b) $\boldsymbol{v} \leftarrow \frac{S(\mathbf{X}'\boldsymbol{u},\Delta_2)}{\|S(\mathbf{X}'\boldsymbol{u},\Delta_2)\|_2^2}$, where $\Delta_2 = 0$ if this results in $\|\boldsymbol{v}\|_1 \leq c_2$; otherwise, Δ_2 is chosen to be a positive constant such that $\|\boldsymbol{v}\|_1 = c_2$.

3. $d \to \boldsymbol{u}'\mathbf{X}\boldsymbol{v}$.

The PMD methods, in general, can also work in the presence of missing data. Letting C represent the set of indices where data *is* present, the criteria can be written as:

$$\underset{\boldsymbol{u},\boldsymbol{v}}{\text{maximize}} \left\{ \sum_{(i,j)\in C} x_{ij} u_i v_j \right\} \quad \textit{subject to} \ \|\boldsymbol{u}\|_2^2 \leq 1, \ \|\boldsymbol{v}\|_2^2 \leq 1, \ P_1(\boldsymbol{u}) \leq c_1, \ P_2(\boldsymbol{v}) \leq c_2. \tag{2.39}$$

This means PMD can be used for data imputation. As well, this leads directly to a type of cross-validation method for determining optimal tuning parameters c_1 and c_2 based on prediction criteria; it can be used for the Sparse PCA application.

Selecting Tuning Parameters:

1. From the original data matrix $\mathbf{X}$, construct 10 data matrices $\mathbf{X}_1, \ldots, \mathbf{X}_{10}$, each of which is missing a nonoverlapping one-tenth of the elements of $\mathbf{X}$, sampled at random from the rows and columns.

2. For each candidate value of c_1 and c_2:

   (a) For $i = 1, \ldots, 10$:

      i. Fit the PMD to $\mathbf{X}_i$ with tuning parameters c_1 and c_2 and calculate $\hat{\mathbf{X}}_i = d\boldsymbol{u}\boldsymbol{v}'$, the resulting estimate of $\mathbf{X}_i$.

      ii. Record the mean squared error of the estimate $\hat{\mathbf{X}}_i$. This mean squared error is obtained by computing the mean of the squared differences between elements of $\mathbf{X}$ and the corresponding elements of $\hat{\mathbf{X}}_i$, where the mean is taken only over elements that are missing from $\mathbf{X}_i$.

(b) Record the average mean squared error across $\mathbf{X}_1, \ldots, \mathbf{X}_{10}$ for tuning parameters c_1 and c_2.

3. The optimal values of c_1 and c_2 are those which correspond to the lowest mean squared error.

Of course, as with the Sparse PCA method by Zou et al., principal components lose their uncorrelated property, thus calling for a new way to calculate the explained variance. The authors reach out to a method proposed in a 2008 paper by Shen and Huang [18], who have a Sparse PCA method very similar to Witten et al. Shen and Huang state how Zou et al. account for the correlated PCs in their adjusted explained variance calculation but do not account for the lack of orthogonality among loading vectors. The new method proposed involves a projection of $\mathbf{X}$ onto the k-dimensional subspace spanned by the k loading vectors being considered; details are given in their paper.

Also nice to note, Witten et al. provide an alternate method to their SPC Criterion. The alteration attempts to enforce orthogonality amongst the loading vectors; one of the nice properties of classic PCA that is often lost during Sparse PCA methods.

R-package: 'PMA'

The authors provide an R-package named 'PMA' that can be installed from a

CRAN repository. Since the PMD method introduced by Witten et al. covers a wide range of applications, only a couple functions prove useful for Sparse PCA with PMD. The 'SPC()' function will carry out the algorithm for solving the SPC criterion in (2.38) and return the left singular vectors, the sparse right singular vectors (loadings), and the singular values. The 'SPC.cv()' function allows one to select tuning parameters based on the cross-fold validation technique described above; via best MSE. Within this function, the authors provide an option to select tuning parameters that achieve a slightly worse MSE than the best pair, thus attaining a more sparse solution set while sacrificing little prediction of the matrix **X**. Embedded as an argument in both 'SPC()' and 'SPC.cv' functions is the orthogonality option, 'orth=', that, when set to TRUE, adapts the calculations in an attempt to enforce orthogonality.

2.3.3 SSPCA formulation by Lee et al.

In 2010, a paper by Lee et al entitled "Super-sparse principal component analyses for high-throughput genomic data" emerged [16]. For simple reporting, we give their method the name 'SSPCA'. The title is certainly exciting; 'Super-sparse' packing quite the punch, and to an appropriate area of high-throughput genomic data. The authors introduce a method called Super-Sparse Principal Component Analysis (SSPCA) that is intended to impose more sparseness to the principal components than prior methods have. Its proposed usefulness is geared towards scenarios where $n << p$. The methodology and results are fully described in their paper and summarized below.

Instead of introducing sparseness through SVD algorithms, Lee et al made use of the NIPALS algorithm, introduced by Hoskuldsson in 1988 [9]. The NIPALS algorithm allows one-at-a-time calculation of eigen value - eigen vector pairs, greatly reducing the computation cost compared to the methods above which use SVD and thus require simultaneous calculation of all k eigen value - eigen vector pairs.

NIPALS Algorithm for first k PCs:

Let $\mathbf{X}^1 = \mathbf{X}$. For $j = 1, \ldots, k$:

1. Initialize $\boldsymbol{z}_j$ as the first column of $\mathbf{X}^j$.
2. Find $\boldsymbol{v}_j$: $\boldsymbol{v}_j \leftarrow \frac{\mathbf{X}^{j\prime}\boldsymbol{z}_j}{\boldsymbol{z}_j'\boldsymbol{z}_j}$.
3. Normalize $\boldsymbol{v}_j$: $\boldsymbol{v}_j \leftarrow \frac{\boldsymbol{v}_j}{\|\boldsymbol{v}_j\|_2}$.
4. Find $\boldsymbol{z}_j$: $\boldsymbol{z}_j \leftarrow \mathbf{X}^j\boldsymbol{v}_j$.

Repeat steps 1 to 4 until convergence.

5. Let $\mathbf{X}^{j+1} = \mathbf{X}^j - \boldsymbol{z}_j\boldsymbol{v}_j'$.

As with the previous methods, the induction of regularization techniques are couriered by realizing a regression formulation. The second step of the algorithm can be recognized as OLS solutions to a set of linear regression problems [16]. Considering the leading PC for now, by rewriting the second step relationship, we obtain

$$\hat{\boldsymbol{v}}_1^{OLS} = (\boldsymbol{z}_1'\boldsymbol{z}_1)^{-1}\mathbf{X}'\boldsymbol{z}_1,$$

which are the combined solutions from k linear regressions; the j^{th} being

$$\boldsymbol{X}_j = \boldsymbol{z}_1 v_{1j} + \boldsymbol{\epsilon}_j,$$

where $\boldsymbol{X}_j$ represents the j^{th} column of $\mathbf{X}$ and v_{1j} represents the j^{th} loading of the loading vector $\boldsymbol{v}_1$. By connecting a regression format to the NIPALS algorithm, the authors now have the ability to introduce sparseness through the usual penalty functions. However, Lee et al. bring forth a random-effect model approach, enabling access to new penalties that potentially result in more sparse solutions. This method was originally proposed in a paper freshly submitted by one of their co-authors with the intent to assist general regression problems; this new paper deals with its application to Sparse PCA. The random-effect is assumed for the loadings. Continuing the explanation with the leading PC, for the j^{th} loading,

$$v_{1j}|u_j \sim N(0, u_j\theta),$$

where θ is the dispersion parameter and u_j follows a gamma distribution with parameter w and density

$$f_w(u_j) = w^{-1/w} \frac{1}{\Gamma(1/w)} u_j^{1/w-1} e^{-u_j/w}.$$

This results in a complex marginal distribution for v_{1j}, having a density function that is computationally troublesome:

$$f_{w,\theta}(v_{1j}) = \int f_\theta(v_{1j}|u_j) f_w(u_j) du_j = \frac{w^{-1/w}}{\Gamma(1/w)\sqrt{2\pi\theta}} \int u_j^{1/w-3/2} e^{-v_{1j}^2/2u_j\theta - u_j/w} du_j.$$

Usual maximum likelihood estimation thrives with a nice convex likelihood function, but in this case, $-\log f_{w,\theta}(v_{1j})$ is non-convex. To provide a work-around to this issue the authors make use of the hierarchical likelihood form, commonly named 'h-likelihood', introduced by Lee and Nelder in 1996 [17]. The connection to the h-likelihood is seen by reforming v_{1j} as a double hierarchical generalized linear model:

$$v_{1j} = \sqrt{\tau_j e_j},$$

where $\tau_j = u_j\theta$ and $e_j \sim N(0, 1)$. By expressing τ_j with a log link function:

$$\log \tau_j = log\theta + logu_j$$

the authors relate this to a general h-likelihood formulation of

$$h = h_1 + h_2,$$

where

$$\begin{aligned}
h_1 &= \sum_{j=1}^{p} \{\log f_\phi(\boldsymbol{X}_j|\boldsymbol{v}_j)\} \\
h_2 &= \sum_{j=1}^{p} \{\log f_\theta(v_{1j}|\boldsymbol{u}_j) + \log f_w(\log \boldsymbol{u}_j)\} \\
\log f_\phi(\boldsymbol{X}_j|\boldsymbol{v}_j) &= \frac{n}{2}\log(2\pi\phi) - \frac{1}{2\phi}(\boldsymbol{X}_j - \boldsymbol{z}_1 v_{1j})'(\boldsymbol{X}_j - \boldsymbol{z}_1 v_{1j}) \\
\log f_\theta(v_{1j}|\boldsymbol{u}_j) &= -\frac{1}{2}\left(\log(2\pi\theta) + \log \boldsymbol{u}_j + {}^{v_{1j}^2}/_{\theta \boldsymbol{u}_j}\right) \\
\log f_w(\log \boldsymbol{u}_j) &= {}^{-\log w}/_{w} - \log \Gamma({}^{1}/_{w}) + {}^{\log \boldsymbol{u}_j}/_{w} - {}^{\boldsymbol{u}_j}/_{w}.
\end{aligned}$$

The authors estimate $\boldsymbol{v}_1$ by accessing something called the 'profile h-likelihood', proposed by one of their co-authors in a neighbouring paper submission. Then, by fluctuating parameter w, they obtain different penalty functions. Ridge regressions are realized when $w = 0$, LASSO when $w = 2$, and what the authors call the 'HL' method when $w = 30$. On this note, they focus on LASSO and HL methods from their SSPCA paper; hence we name them SSPCA.h and SSPCA.l for simple reporting.

Lee et al. propose to select tuning parameters via a K-fold cross-validation procedure that judges performance based on sample variance of the resulting PCs.

R-Package: 'SSPCA'

The authors provide an R-package for SSPCA that can be downloaded from

Prof. Yudi Pawitan's website: http://www.meb.ki.se/ yudpaw/. The function used to compute Sparse PCA via imposing random effect assumptions on the loadings through the NIPALS algorithm is 'sspca()'. Within it, you may specify the option 'penalty=' to be "HL" or "Lasso", accessing the different penalty types dictated by parameter w, as described above. You may also choose to select tuning parameters manually or via the K-fold cross-validation procedure. The cross-validation way takes an exceedingly long time in comparison to specifying tuning parameters rather arbitrarily.

Chapter 3

Simulation Studies

The problem has been presented in detail and the possible solutions have been discussed theoretically. The next step is to discover the *best* Sparse PCA method for a variety of scenarios that might be encountered in the real-world. For adequate testing grounds, simulation is an absolute must [2]. From generating the data ourselves, we have control over the *true* parameters of the underlying distribution from which the data arises. In the context of Sparse PCA methods, this means completely specifying the variance-covariance structure in the multivariate normal distribution from which a dataset is generated. This specification is rather simple to implement with the help of R software and packages; this is not the problem. The challenge is to rigorously test the Sparse PCA methods for *suitable* choices of variance structure in a *regulated* and *meaningful* way. In this Chapter, the structure of how the methods will be evaluated and compared will be laid out, the simulation scenarios will be presented, and their results will be summarized to the point of conclusion.

3.1 Constructing the Simulations

In this section, we will guide the reader through how we generate the data and obtain different simulation settings, as well as how the Sparse PCA methods were implemented and judged. The main sub-categories we focus on are: Parameters to Vary, Criteria to Test, Simulation Settings Considered, Tuning Parameter Selection, and Number of Simulation Iterations. The R-code used to carry out the simulations under this structure was created from scratch to enable a flexible, user friendly coding environment to test many varieties of the settings considered in this thesis. The code is available upon request.

Parameters to Vary

The variance-covariance structures that will provide both simplicity and flexibility when testing method performance will be of the following *block-diagonal* form:

$$\Sigma = \begin{pmatrix} \Sigma_{11} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \Sigma_{22} & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \Sigma_{kk} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \Sigma_{else} \end{pmatrix}$$

Here, k represents the number of proposed *groups* in the underlying distribution and Σ_{jj}, $j = 1, \ldots, k$, is the variance-covariance matrix for the set of variables in the j^{th} group. Σ_{else} represents the variance-covariance matrix for all non-grouped variables, i.e. extra noise in the dataset, and in turn will have no correlation amongst its components. The block-diagonal form assumes no correlation between groups. This allows for a simple specification during simulation and emulates the hopeful situation for researchers who would use PCA or Sparse PCA.

For the block-diagonal form above, a single groups variance-covariance matrix *could* be defined as

$$\Sigma_{jj} = \begin{pmatrix} \sigma_{11} & \rho_{12}\sqrt{\sigma_{11}\sigma_{22}} & \cdots & \rho_{1p_j}\sqrt{\sigma_{11}\sigma_{p_jp_j}} \\ \rho_{12}\sqrt{\sigma_{11}\sigma_{22}} & \sigma_{22} & \cdots & \rho_{2p_j}\sqrt{\sigma_{22}\sigma_{p_jp_j}} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{1p_j}\sqrt{\sigma_{11}\sigma_{p_jp_j}} & \rho_{2p_j}\sqrt{\sigma_{22}\sigma_{p_jp_j}} & \cdots & \sigma_{p_jp_j} \end{pmatrix},$$

where p_j represents the number of variables in proposed group j; thus $\sum_{j=1}^{k} p_j + p_{else} = p$.

The diagonal elements are the variances for each variable within the group. The larger they are in comparison to other groups, the more noticeable this list of variables will be to a PCA method. The off-diagonal elements are the covariances between each pair of variables within the group. The closer the correlations are to 1, the more likely the set of variables will stick together during the PCA analysis. However, to simplify, a single variance and single correlation for each group will be used to represent each of the variables in the group. Generally, if we want a group to stand out over the rest, we just want the variances to be higher in that group and don't need to specify each and every one; likewise we want the correlations to be stronger and don't need to micro-manage. The resulting variance-covariance matrix for a single group that we *will* be using is defined as:

$$\Sigma_{jj} = \begin{pmatrix} \sigma_j^2 & \rho_j\sigma_j^2 & \cdots & \rho_j\sigma_j^2 \\ \rho_j\sigma_j^2 & \sigma_j^2 & \cdots & \rho_j\sigma_j^2 \\ \vdots & \vdots & \ddots & \vdots \\ \rho_j\sigma_j^2 & \rho_j\sigma_j^2 & \cdots & \sigma_j^2 \end{pmatrix},$$

Clearly, specifying these parameters will categorize how the data will be produced. Also, since data is generated based on a probability distribution (multivariate normal), we expect the resulting dataset to have a perturbed sample variance-covariance structure compared to the true variance-covariance structure. We could control the amount of perturbation with sample size n alone, but since the highlight of these methods are their performance when p gets bigger *in relation* to n, the p to n ratio will the sole dictator of data dimension. We have locked the sample size at $n = 100$ for all simulations to standardize results; the more parameters we vary, the harder it will be the draw conclusions.

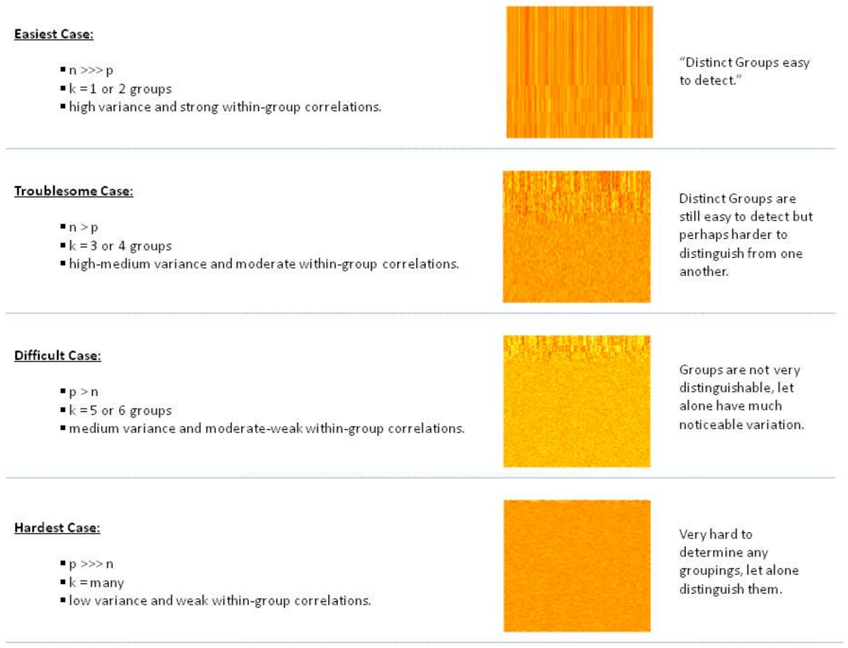
A list of parameters that will be varied during the simulations is displayed in Table 3.1. Note that two quantities in the table do not vary and are just their for completion: n, which is always 100, and ρ_{else}, which is always 0 since it marks the correlation amongst the non-grouped variables. The R-codes built to execute the simulations require values for each of these parameters so the naming-scheme in the code is found in the last column of this table. The choices for these parameters will be introduced and justified shortly but, generally speaking, scenarios of varying *difficulty* will be tested. A possible visualization of this is presented in Figure 3.1 by

Table 3.1: Parameters Varied during Simulations

Description	Parameter	R-code
Sample Size	$n = 100$	$n = 100$
Number of Variables	p	p
Number of Groups	k	k
Number of Variables in each Group	$p_1, p_2, \ldots, p_k, p_{else}$	$pg = (pg1, pg2, \ldots, pgk, pgelse)$
Variance within each Group	$\sigma_1^2, \sigma_2^2, \ldots, \sigma_k^2, \sigma_{else}^2$	$vg = (vg1, vg2, \ldots, vgk, velse)$
Correlation within each Group	$\rho_1, \rho_2, \ldots, \rho_k, \rho_{else} = 0$	$cg = (cg1, cg2, \ldots, cgk, celse = 0)$

making use of heatmaps. A heatmap is a data visualization technique that simply replaces numbers with a color gradient [6]; similar numbers share similar colors, or shades. If a group of highly correlated variables are ordered in a dataset, you will see it stick out on the heatmap via a more monotone block of colour. Usually, data will not be organized into groups so clustering algorithms will be used in conjunction with the coloring scheme, but in Figure 3.1 we have generated the data with the block-diagonal variance-covariance structure and have attained an ordered set of variables. Heatmaps and clustering algorithms prove important to tying real data situations to the simulation results to come; Chapter 4 will have a demonstration of this. Heatmaps are often used for genetic data and, by convention, plot the variables along the rows and samples along the columns. The figure suggests that a larger p to n ratio, more groups, less distinction between group variances, and less correlation to bind each group together make for the toughest of situations.

To demonstrate how the 'difficulty' of a simulation setting presents itself in the variance-covariance structure, Figures 3.2 (a), (b), (c), and (d) help visualize the grouped 'signals' (green) that might be hard for the PCA method to pick up or



Easiest Case:

- n >>> p
- k = 1 or 2 groups
- high variance and strong within-group correlations.

"Distinct Groups easy to detect."

Troublesome Case:

- n > p
- k = 3 or 4 groups
- high-medium variance and moderate within-group correlations.

Distinct Groups are still easy to detect but perhaps harder to distinguish from one another.

Difficult Case:

- p > n
- k = 5 or 6 groups
- medium variance and moderate-weak within-group correlations.

Groups are not very distinguishable, let alone have much noticeable variation.

Hardest Case:

- p >>> n
- k = many
- low variance and weak within-group correlations.

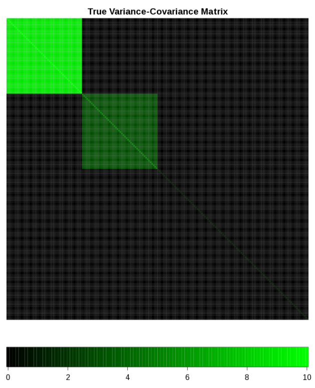
Very hard to determine any groupings, let alone distinguish them.

Figure 3.1: *An attempt to categorize datasets into levels of difficulty, when using PCA and Sparse PCA methods, based on underlying variance structure and dimension*

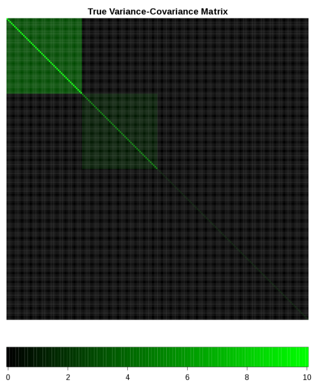
distinguish amongst the 'noise' (black).

Strategies to generate the data are now clear. We can range any of the parameters in Table 3.1 to develop a situation we deem worth testing, but how will we assess which methods *do well*. To establish the criteria that are best to use, we should have a look at small-scale example of true vs. estimated loading vectors.

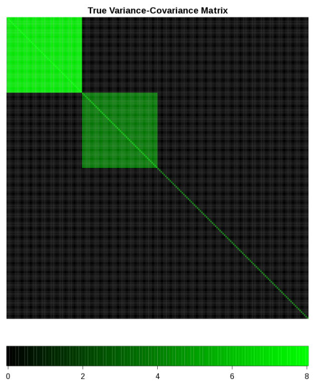
Consider generating $n = 100$ samples from a multivariate normal distribution



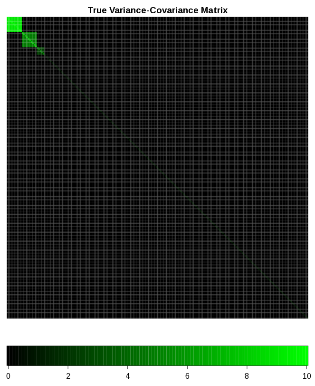
(a) Easiest



(b) Harder



(c) Harder



(d) Harder

Figure 3.2: *True variance-covariance matrices for simulation settings of varying difficulty. (a) large distinguishable groups with high correlations. (b) introducing low correlations. (c) balancing group variances . (d) more, smaller groups.*

with the variance-covariance matrix

$$\Sigma = \begin{pmatrix} 5 & 4 & 4 & 0 & 0 & 0 \\ 4 & 5 & 4 & 0 & 0 & 0 \\ 4 & 4 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

As described in Section 2.1, we center the resulting dataset and call it $\mathbf{X}_{100\times 6}$. Keeping in mind that only $k = 2$ blocks of correlated variables exist in Σ, we obtain the SVD of $\mathbf{X}$ and extract necessary information for the first two PCs; the PCs themselves, $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, their loading vectors, $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, and the singular values, d_{11} and d_{22}, to be able to obtain percentage explained variance. Of course, since we only have $p = 6$ variables, the sample variance-covariance matrix of $\mathbf{X}$ has a small dimension so we might want to just obtain this same information directly from its eigen values and eigen vectors instead of using the SVD. The sample variance-covariance matrix is slightly off from the underlying truth:

$$\mathbf{S} = \begin{pmatrix} 4.6 & 3.4 & 3.8 & -0.2 & 0.2 & -0.5 \\ 3.4 & 4.5 & 3.6 & -0.1 & 0.5 & -0.5 \\ 3.8 & 3.6 & 5.1 & -0.4 & 0.2 & -0.5 \\ -0.2 & -0.1 & -0.4 & 1.6 & 0.8 & 0.4 \\ 0.2 & 0.5 & 0.2 & 0.8 & 2.2 & 0.3 \\ -0.5 & -0.5 & -0.5 & 0.4 & 0.3 & 0.9 \end{pmatrix}$$

<u>Criteria to Test</u>

Given the true and sample variance-covariance matrices above, the true and estimated loading vectors are shown in Table 3.2 (using the SPCA method, for example): Having a look, we can think about properties of the estimated loading vectors that would represent a *good job* done by the Sparse PCA method. Generally speaking, we want the estimated loading vectors to be as *close* to the true vectors as possible.

Table 3.2: *How 'close' are the True and the Estimated Loading Vectors?*

(a) The true loading vectors from Σ.

	PC1	PC2	$\cdots$
	0.577	0	$\cdots$
	0.577	0	$\cdots$
	0.577	0	$\cdots$
	0	0.707	$\cdots$
	0	0.707	$\cdots$
	0	0	$\cdots$
APEV	65.0%	15.0%	$\cdots$

(b) The estimated loading vectors by using a Sparse PCA method.

	PC1	PC2	$\cdots$
	0.565	0	$\cdots$
	0.555	0	$\cdots$
	0.610	0	$\cdots$
	0	0.549	$\cdots$
	0	0.830	$\cdots$
	-0.026	0.100	$\cdots$
APEV	63.9%	15.4%	$\cdots$

Many similarity measures can do the trick here, but since we are mostly interested in the vectors capturing the underlying group structure, proper classification of zero-valued loadings and non-zero-valued loadings will be the main focus. Of course, even if all loadings are classified perfectly, the weights associated with each might still differ from the truth, thus relaying distorted PCs. This suggests us to track how different the true and estimated non-zero loadings. Another important criterion to test is orthogonality amongst the loading vectors, which dictates how much correlation amongst the resulting PCs has been introduced through using a Sparse PCA method in place of Classical PCA. This is especially important if planning to use the PCs in future analysis. To capture these desired properties, we have judged the estimated loading vectors with a list of six criteria. The criteria are presented as follows, beginning with bolded abbreviations in order to link the reader to the table headers in the Results section:

NZ - The number of non-zero-values in the expected loading vector.

TRUENZ - The number of loading-entries that are non-zero in both the true and estimated loading vectors. Ideally, the estimated loading vector picks up all true non-zero loadings.

TRUEZ - The number loading-entries that are zero in both the true and estimated loadings vectors. Ideally, the estimated loading vector picks up all true zero-loadings.

ANGLE - A value between 0 (perfect) and 1 (worst-possible) that measures the distance between true and estimated loading vectors as they lie in p-dimensional space. Suggested by [12] and endorsed by the authors of SSPCA in their paper. Calculated as:

$$dist\left(v_k, \hat{v}_k\right) = \sqrt{1 - \left(v_k' \hat{v}_k\right)^2}$$

APEV - The Adjusted Percentage Explained Variance. Calculated as suggested by Shen and Huang in 2008 [18] and employed by the authors of PMDSPCA in their paper (explained in PMDSPCA methodology section of Chapter 2).

ORTH - The number of unique pairs of loading vectors that are NOT roughly orthogonal with each other. A pair of vectors was considered as roughly orthogonal when their dot-product was less than 0.003.

Many of the Sparse PCA methodology papers have tested the sparseness (NZ) and classification criteria (TRUENZ, TRUEZ); this is nothing new. Adjusted percentage explained variance (APEV) has also been a go-to criteria, however the formula has not been consistent; the one used here has been argued to have the most appropriate adjustment. Only a few have tested the specific loading (weight) differences (ANGLE) and there has been little attention to how much orthogonality (ORTH) is lost by using Sparse PCA methods. We feel these six criteria to be crucial to determine the fate of Sparse PCA methods. Lastly, we have chosen to judge the first k loading vectors. This is because k represents the number of unique groups specified in the true variance-covariance matrix and is the number of loading vectors we would expect to pick up the unique groups. Up until now, most Sparse PCA literature with simulation results provide specs from only the leading PC, making the subsequent PCs very interesting here.

Simulation Settings Considered

A total of 56 simulation settings of varying difficulty were chosen by taking different combinations of parameters from Table 3.1. The list of the 56 simulation settings considered is given in Table 3.3. A quick naming scheme for the simulation settings was assigned to try and simplify reporting; make sure to read the footnotes of the table to understand how to read it. These simulation settings were chosen in attempt to provide a gradient of difficulty, from easiest to hardest, with regards to not only the suspected ability for a Sparse PCA method to capture the true underlying structure, but also with regards to how stressful the data will be to a researcher (data-dimension, especially). As suggested in Figures 3.1 and 3.2, a more 'difficult' scenario is considered to have more groups, smaller groups, less distinguishable variances across groups, and lower within-group correlations.

Table 3.3: Simulation Setting Descriptions: 56 total, ordered easiest to hardest

Setting	p	k	pg	vg	cg
Sim1aaa[a]	10[b]	2[c]	4, 4, (2)[d]	10, 5, (1)[e]	0.9, 0.6, (0)[f]
Sim1aab	10	2	4, 4, (2)	10, 5, (1)	0.7, 0.5, (0)
Sim1aac	10	2	4, 4, (2)	10, 5, (1)	0.5, 0.4, (0)
Sim1aad	10	2	4, 4, (2)	10, 5, (1)	0.3, 0.2, (0)
Sim1aba	10	2	4, 4, (2)	8, 6, (4)	0.9, 0.6, (0)
Sim1abb	10	2	4, 4, (2)	8, 6, (4)	0.7, 0.5, (0)
Sim1abc	10	2	4, 4, (2)	8, 6, (4)	0.5, 0.4, (0)
Sim1abd	10	2	4, 4, (2)	8, 6, (4)	0.3, 0.2, (0)
Sim2aaa	50	2	20, 20, (10)	10, 5, (1)	0.9, 0.6, (0)
Sim2aab	50	2	20, 20, (10)	10, 5, (1)	0.7, 0.5, (0)
Sim2aac	50	2	20, 20, (10)	10, 5, (1)	0.5, 0.4, (0)
Sim2aad	50	2	20, 20, (10)	10, 5, (1)	0.3, 0.2, (0)
Sim2aba	50	2	20, 20, (10)	8, 6, (4)	0.9, 0.6, (0)
Sim2abb	50	2	20, 20, (10)	8, 6, (4)	0.7, 0.5, (0)
Sim2abc	50	2	20, 20, (10)	8, 6, (4)	0.5, 0.4, (0)
Sim2abd	50	2	20, 20, (10)	8, 6, (4)	0.3, 0.2, (0)
Sim2baa	50	2	5, 5, (40)	10, 5, (1)	0.9, 0.6, (0)
Sim2bab	50	2	5, 5, (40)	10, 5, (1)	0.7, 0.5, (0)
Sim2bac	50	2	5, 5, (40)	10, 5, (1)	0.5, 0.4, (0)
Sim2bad	50	2	5, 5, (40)	10, 5, (1)	0.3, 0.2, (0)
Sim2bba	50	2	5, 5, (40)	8, 6, (4)	0.9, 0.6, (0)
Sim2bbb	50	2	5, 5, (40)	8, 6, (4)	0.7, 0.5, (0)
Sim2bbc	50	2	5, 5, (40)	8, 6, (4)	0.5, 0.4, (0)
Sim2bbd	50	2	5, 5, (40)	8, 6, (4)	0.3, 0.2, (0)
Sim3aaa	200	2	50, 50, (100)	10, 5, (1)	0.9, 0.6, (0)
Sim3aab	200	2	50, 50, (100)	10, 5, (1)	0.7, 0.5, (0)
Sim3aac	200	2	50, 50, (100)	10, 5, (1)	0.5, 0.4, (0)
Sim3aad	200	2	50, 50, (100)	10, 5, (1)	0.3, 0.2, (0)
Sim3aba	200	2	50, 50, (100)	8, 6, (4)	0.9, 0.6, (0)
Sim3abb	200	2	50, 50, (100)	8, 6, (4)	0.7, 0.5, (0)
Sim3abc	200	2	50, 50, (100)	8, 6, (4)	0.5, 0.4, (0)
Sim3abd	200	2	50, 50, (100)	8, 6, (4)	0.3, 0.2, (0)
Sim3baa	200	3	10, 10, 5, (175)	10, 8, 5, (1)	0.9, 0.6, 0.5, (0)
Sim3bab	200	3	10, 10, 5, (175)	10, 8, 5, (1)	0.7, 0.5, 0.4, (0)
Sim3bac	200	3	10, 10, 5, (175)	10, 8, 5, (1)	0.5, 0.4, 0.3, (0)
Sim3bad	200	3	10, 10, 5, (175)	10, 8, 5, (1)	0.4, 0.3, 0.2, (0)
Sim3bba	200	3	10, 10, 5, (175)	8, 7, 6, (4)	0.9, 0.6, 0.5, (0)
Sim3bbb	200	3	10, 10, 5, (175)	8, 7, 6, (4)	0.7, 0.5, 0.4, (0)
Sim3bbc	200	3	10, 10, 5, (175)	8, 7, 6, (4)	0.5, 0.4, 0.3, (0)
Sim3bbd	200	3	10, 10, 5, (175)	8, 7, 6, (4)	0.4, 0.3, 0.2, (0)
Sim4aaa	2000	3	100, 100, 50, (1750)	10, 8, 5, (1)	0.9, 0.6, 0.5, (0)
Sim4aab	2000	3	100, 100, 50, (1750)	10, 8, 5, (1)	0.7, 0.5, 0.4, (0)
Sim4aac	2000	3	100, 100, 50, (1750)	10, 8, 5, (1)	0.5, 0.4, 0.3, (0)
Sim4aad	2000	3	100, 100, 50, (1750)	10, 8, 5, (1)	0.4, 0.3, 0.2, (0)
Sim4aba	2000	3	100, 100, 50, (1750)	8, 7, 6, (4)	0.9, 0.6, 0.5, (0)
Sim4abb	2000	3	100, 100, 50, (1750)	8, 7, 6, (4)	0.7, 0.5, 0.4, (0)
Sim4abc	2000	3	100, 100, 50, (1750)	8, 7, 6, (4)	0.5, 0.4, 0.3, (0)
Sim4abd	2000	3	100, 100, 50, (1750)	8, 7, 6, (4)	0.4, 0.3, 0.2, (0)
Sim4baa	2000	4	20, 15, 10, 5, (1950)	10, 10, 5, 5, (1)	0.9, 0.6, 0.5, 0.4, (0)
Sim4bab	2000	4	20, 15, 10, 5, (1950)	10, 10, 5, 5, (1)	0.7, 0.5, 0.4, 0.3, (0)
Sim4bac	2000	4	20, 15, 10, 5, (1950)	10, 10, 5, 5, (1)	0.5, 0.4, 0.3, 0.2, (0)
Sim4bad	2000	4	20, 15, 10, 5, (1950)	10, 10, 5, 5, (1)	0.4, 0.3, 0.2, 0.1, (0)
Sim4bba	2000	4	20, 15, 10, 5, (1950)	8, 8, 6, 6, (4)	0.9, 0.6, 0.5, 0.4, (0)
Sim4bbb	2000	4	20, 15, 10, 5, (1950)	8, 8, 6, 6, (4)	0.7, 0.5, 0.4, 0.3, (0)
Sim4bbc	2000	4	20, 15, 10, 5, (1950)	8, 8, 6, 6, (4)	0.5, 0.4, 0.3, 0.2, (0)
Sim4bbd	2000	4	20, 15, 10, 5, (1950)	8, 8, 6, 6, (4)	0.4, 0.3, 0.2, 0.1, (0)

[a] Labeling system for Simulation Settings: [1][a][a][a] means easiest difficulty in each of [(p,k)], [pg], [vg], and [cg] parameters, respectively. Sim1aaa is the easiest scenario, Sim4bbd is the hardest.

[b] p=10 means 10 variables in simulated data.

[c] k=2 means 2 specified groups in simulated data.

[d] 4, 4, (2) means 4 variables in the first group, 4 variables in the second group, and 2 residual variables.

[e] 10, 5, (1) means variables in the first group have a variance of 10, variables in the second group have a variance of 5, and the residual variables have a variance of 1.

[f] 0.9, 0.6, 0 means variables in the first group are correlated with 0.9, variables in the second group are correlated with 0.6, and residual variables have no correlation.

Tuning Parameter Selection

Cross-validation methods aimed at maintaining good predictive qualities of the underlying dataset were explained for each of the Sparse PCA methods in Section 2.3. They were implemented to determine the tuning parameters used in respective Sparse PCA methods for *each* of the simulated datasets (with exception for the SPCA methods at the highest dimension; see results). This means that iterations within the same simulation setting could have used different tuning parameters, although they were chosen based on the exact same criteria. These are perhaps the most calculation-intensive ways to select tuning parameters. We could have just examined fixed range of tuning parameters, or validated based on APEV or NZ (sparseness), but since the cross-validation methods are built into the software packages, these are most likely to be used in practice and therefore are most likely to be of importance.

Number of Simulation Iterations

For PCA, SPCA, and PMDSPCA methods, 1000 simulated datasets were generated for each simulation setting and used to obtain estimated loading vectors. For SSPCA.h and SSPCA.l methods, a limit of 100 datasets were used, due to the computing time needed to churn through the cross-validation method within the sspca() function.

3.2 Simulation Results

We will now deliver a concise yet complete summary of our simulation results. Focus will be placed on comparing the Sparse PCA methods with one another and creating a set of guidelines regarding which method to pick under different circumstances. Not all results will be shown since the point is not to bombard the reader with numbers, but to highlight the take-home messages. Full tables for every scenario are available upon request. The Figures and Tables to come will be presenting the mean values across all simulation iterations within a given simulation setting. Within most simulation settings, the distribution of each calculated criteria across the simulated datasets appeared to be centered and approximately normally distributed with very few outliers, making the mean a representative statistic.

$n = 100$, $p = 10$ ($n >> p$): Sim1aaa - Sim1abd from Table 3.3

This ratio of p to n is 0.1 and, with only $p = 10$, is probably not of much practical interest to researchers. Most of the time, 10 variables will not pose many issues and PCA methods will not be called for. For this reason, we will spend little effort scrutinizing these results and just brush lightly over the comparisons to familiarize the reader with some Tables and Figures that reflect the larger-scale inspection to come.

For now, have a look at Table 3.4, which displays a summary of the results from the easiest environment, Sim1aaa (see Table 3.3 for parameters). The column headers are each of the criteria we tested, split by PC number and the rows are the different PCA methods, starting with the True (or expected) values in each of the criteria.

Comparing the True row with any of the methods, one can quickly determine if the method is performing as it should, or in other words, if the estimated loading vectors are close to the true loading vectors.

Table 3.4: Results from Sim1aaa (presented as means): n = 100, p = 10, k = 2, pg = c(4, 4, 2), vg = c(10, 5, 1), cg = c(0.9, 0.6, 0)

Method	NZ		TRUENZ		TRUEZ		ANGLE		APEV		ORTH
	PC1	PC2	PC1	PC2	PC1	PC2	PC1	PC2	PC1	PC2	
TRUE	4	4	4	4	6	6	0	0	59.68	22.58	0
PCA	10	10	4	4	0	0	0.11	0.14	60.08	22.46	0
SPCA	8.52	9.47	4	4	1.48	0.53	0.1	0.13	60.06	22.46	0.46
PMDSPCA	7.91	8.41	4	4	2.09	1.59	0.06	0.09	59.72	22.61	0.85
SSPCA.h	4.29	5.37	4	3.81	5.71	4.44	0.04	0.16	59.31	21.77	0.4
SSPCA.l	5.62	6.58	4	4	4.38	3.42	0.04	0.09	59.37	22.67	0.7

Under this easiest setting, there are only 2 groups of large size, relative to our small dimension of $p = 10$, that have distinguishable variances and high within-group correlations. With the lowest ANGLE values and most TRUEZ captured, the SSPCA.h method estimates the true loading vectors with the most accuracy. Both SPCA and PMDSPCA methods have a hard time capturing the true zero loadings, making them only marginally more useful than Classical PCA.

As we decrease the within-group correlations, we arrive at Sim1aac (see Table 3.3), with results displayed in Table 3.5. It seems that SPCA and SSPCA methods lose true zeros, especially in PC2, whereas PMDSPCA becomes the most successful method by correctly classifying almost all true zeros and non-zeros. It should be noted that although classification was excellent, ANGLE values suffer; meaning loading weights were distorted from the true combinations.

To summarize findings across this data dimension, SPCA was only fractionally better than Classical PCA. (Keep in mind that we are using cross-validation methods

Table 3.5: Results from Sim1aac (presented as means): n = 100, p = 10, k = 2, pg = c(4, 4, 2), vg = c(10, 5, 1), cg = (0.5, 0.4, 0)

Method	NZ		TRUENZ		TRUEZ		ANGLE		APEV		ORTH
	PC1	PC2	PC1	PC2	PC1	PC2	PC1	PC2	PC1	PC2	
TRUE	4	4	4	4	6	6	0	0	40.32	17.74	0
PCA	10	10	4	4	0	0	0.16	0.28	41.08	18.12	0
SPCA	9.4	9.93	4	4	0.6	0.07	0.16	0.28	41.08	18.12	0.07
PMDSPCA	4.53	4.45	3.96	3.95	5.43	5.5	0.35	0.37	37.93	16.93	0.19
SSPCA.h	4.41	7.33	4	3.56	5.59	2.23	0.11	0.37	40.22	17.02	0.6
SSPCA.l	5.83	7.28	4	3.89	4.17	2.61	0.13	0.25	40.33	17.85	0.74

to select the tuning parameters; as described in Section 2.2.1, the spca() function is able to select a specified number of non-zero loadings for each PC - perhaps this would be best to use for this data dimension). The SSPCA.h method shows the best performance in all categories for the leading PC and, in easier scenarios, PC2 as well, but the PMDSPCA becomes the method of choice when injecting difficulty via making the variances less distinguishable or lowering within-group correlations, especially if you care about subsequent PCs.

$n = 100$, $p = 50$ $(n > p)$: Sim2aaa - Sim2bbd from Table 3.3

This is where using Sparse PCA methods become more crucial. The p to n ratio is 0.5 and still won't cause many issues, but there is a need for interpretable PCs. Since there are more variables to work with, the size of the groups will be fluctuated here.

For large-group cases (Sim2a..), at high within-group correlations, the SSPCA.l and PMDSPCA methods are the best to use since they classify almost every loading correctly. The SSPCA.h method perfectly classifies the leading PC, but it fails to inject enough sparseness to the loading vector of PC2 for it to be interpretable. Once lower within-group correlations are present, PMDSPCA rises to the occasion while the SSPCA methods fall off due to over-sparseness; in some cases they return loading vectors holding just one or two non-zero loadings. SPCA always captures all non-zero loadings, but only attains a maximum of mid-level sparseness at the easiest of scenarios, from which it quickly depreciates.

For small-group cases (Sim2b..), at the highest within-group correlations and distinguishable between-group variances, the SSPCA.l method is clearly best to use, as PMDSPCA seems to struggle at obtaining sparseness in this setting; contrary to PMDSPCA's good qualities when large groups were present. But as soon as within-group correlation is dropped here, SSPCA methods become over-sparse; an amplified version of what was seen in the large-group case. In fact, when between-group variances become less distinguishable, SSPCA methods never achieve more than one non-zero loading in the loading vector of their leading PC, making them completely impractical. The extent of this issue is shown in Table 3.6.

PMDSPCA experiences minor drops in true non-zero classification and large increases in true zero classification when within-group correlation decreases and usually only sees large differences in the lowest of within-group correlation scenarios. The SPCA method never gains sparseness in its estimated loading vectors, thus not improving interpretation upon Classical PCA.

$n = 100$, $p = 200$ ($n < p$): Sim3aaa - Sim3bbd from Table 3.3

The p to n ratio is 2 and now poses a threat to usual statistical analyses. Sparse PCA methods can serve great purpose under this data dimension, since 200 variables

	Method	NZ		TRUENZ		TRUEZ		ANGLE		ORTH
		PC1	PC2	PC1	PC2	PC1	PC2	PC1	PC2	
Sim2baa	TRUE	5	5	5	5	45	45	0	0	0/1
	PMDSPCA	27.24	23.21	5	5	22.76	26.79	0.09	0.12	0.98
	SSPCA.h	5	13.85	5	3.69	45	34.84	0.11	0.47	0.67
	SSPCA.l	5	5.66	5	4.9	45	44.24	0.04	0.12	0.05
Sim2bab	TRUE	5	5	5	5	45	45	0	0	0/1
	PMDSPCA	25.63	20.37	5	5	24.37	29.63	0.11	0.15	0.97
	SSPCA.h	4.02	6.46	3.57	1.51	44.55	40.05	0.5	0.89	0.84
	SSPCA.l	2.37	3.85	2.19	1.26	44.82	42.41	0.75	0.83	0.21
Sim2bba	TRUE	5	5	5	5	45	45	0	0	0/1
	PMDSPCA	18.2	15.08	4.99	4.99	31.8	34.91	0.11	0.16	0.88
	SSPCA.h	1	4	1	0	45	41	0.89	1	0
	SSPCA.l	1	1	1	0	45	44	0.89	1	0

Table 3.6: It is clear to see that under the easiest of small-group simulation settings, Sim2baa, the SSPCA.l method does exceptionally well in every category, even maintaining orthogonality amongst its loading vectors. SSPCA.h attains a great leading PC, but PC2 lacks sparseness. But as we slightly decrease the within-group correlations and move to Sim2bab, SSPCA.l becomes over sparse; this effect is extremely amplified when balancing the group variances in Sim2bba and is realized by SSPCA.h too. Meanwhile, PMDSPCA actually improves sparseness under the more difficult settings.

are not pretty to work with. Since there are more variables to work with here, smaller-group cases have employed a third underlying group and therefore we will examine PC3.

For large-group cases (Sim3a..), at high within-group correlations, the SSPCA.l method returns very accurate and orthogonal loading vectors with barely any negative characteristics to report. SSPCA.h also does very well but, as seen previously, still fails to capture true zeros in subsequent PCs. However, like in the previous data dimensions, the SSPCA methods both become over-sparse and lose the majority of true non-zero loadings when within-group correlations are reduced, making them not viable in these environments. Not indicative of the lower-dimensional scenarios thus far, the SPCA method does very well in all aspects, except orthogonality preservation. The SPCA results show a natural decline of accuracy and classification while decreasing within-group correlations or merging between-group variances, whereas PMDSPCA again steps up as the best method in these more difficult scenarios. It seems that PMDSPCA is continuously the go-to method under difficult scenarios but only sometimes sparse enough in easier scenarios.

For small-group cases (Sim3b..), only in the easiest variance structures and for the leading PC do the SSPCA methods properly classify non-zero loadings while being very sparse. Otherwise, they experience over-sparse and unreasonable representations. The SPCA method still performs as well as it did with the large-group settings, but with reduced sparseness. However, its performance is robust to balancing group variances and decreasing within-group correlations; from 69% to 54% true zero loadings captured in the loading vector of the leading PC over the range of settings, with only fractional changes in ANGLE and APEV. The PMDSPCA method begins slightly less sparse than SPCA at the highest within-group correlations and differing variances, but improves in the more difficult scenarios. SPCA maintains almost perfect capture-rate of the true non-zero loadings for all PCs across all small-group settings in this data-dimension, whereas PMDSPCA loses some at lower within-group correlations with balanced group variances.

$n = 100$, $p = 2000$ ($n << p$): Sim4aaa - Sim4bbd from Table 3.3

This data has a p to n ratio of 20 and demonstrates an extremely troublesome dataset for researchers. Under this very high-dimensional setting we expect realistic datasets to hold more natural underlying groups. Noting this, we have built scenarios to include either three large groups or four small groups and captured PC3 and PC4 as well; PC3 for large group cases and both PC3 and PC4 for small group cases. The cross-validation method for SPCA was extremely time-consuming as it is not clear as to which tuning parameters to try and some choke the process. Instead of using this prediction-based cross-validation approach, we selected tuning parameters that resulted in the most sparse solution while not losing much APEV from the try before. If the APEV for any PC dropped by more than 50%, we chose the tuning parameter from the prior attempt. Also, here we took advantage of the arraysp() function that the authors specifically made for very high-dimensional data.

For large-group cases (Sim4a..), this altered approach to the SPCA method outperforms any of the more recent methods in all criteria while tested under the simplest variance-covariance conditions; when the group variances are distinguishable with high within-group correlations. The SPCA method remains the best method until the lowest within-group correlations are present or until between-group variances are less distinguishable, at which point sparseness is greatly lost. In these more difficult cases the PMDSPCA method returns the most sparse loading vectors that also maintain a better angle to the true loading vectors (ANGLE) at only a slight expense of explained variance (APEV) loss; it still achieves larger-than-true explained variance (APEV). In fact, the true APEV is exceeded on most occasions by all methods. The SSPCA methods do not have any troubles with over-sparseness under any of the large-group settings in this high data dimension. The SSPCA.h method is able to extract leading PCs that have sparse and accurately estimated loading vectors, but subsequent PCs greatly lack these desired qualities. If one cares only about the leading PC, the SSPCA.h method will suffice. The SSPCA.l method perfectly classifies the true non-zero loadings but fails to inject enough sparseness to out-do the performance of SPCA and PMDSPCA methods.

For small-group cases (Sim4b..), the SSPCA.h method returns the best classified and most sparse loading vector for the leading PC under all within-group correlation levels while group variances are distinguishable. When group variances are balanced up, it loses sparseness in its leading PC, and already present ANGLE inefficiencies are severely increased; the loadings might be classified decently, but wrong weights. Following prior trends, it still cannot provide subsequent loading vectors with enough sparseness to surpass the other competing methods. With distinguishable group variances, SSPCA.l and PMDSPCA methods generate excellently classified loading vectors that also possess good ANGLE criteria for all four PCs; especially good compared to SPCA and SSPCA.h methods. However, when group variances are balanced, PMDSPCA loading vectors remain sparse while ones from SSPCA.l lose sparseness. To this end, the ANGLE from PMDSPCA is superior and we recognize that it is probably the only method to return interpretable PCs; the SPCA method practically reduces to Classical PCA in terms of sparseness.

Figure 3.3 shows a series of graphics that emphasize the continued theme that has been shown yet again in these very high-dimensional cases; PMDSPCA working in the presence of more difficult settings.

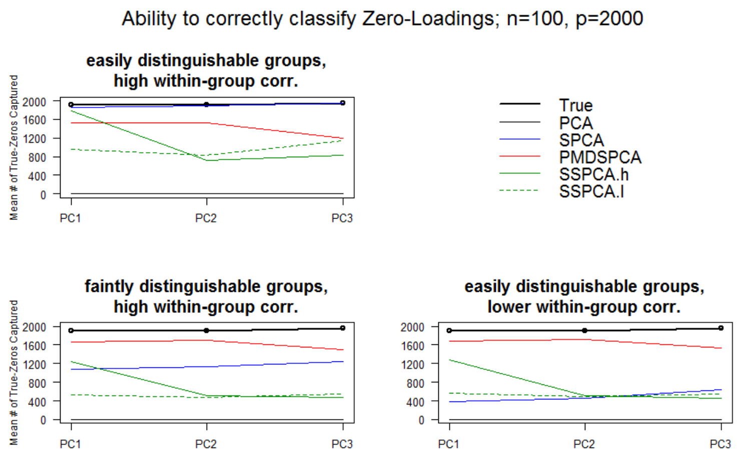


Figure 3.3: *Showing the continued trend from our simulation results; how the PMD-SPCA method is grows superior in sparseness under more difficult variance structures. The top graph is the easiest case, the bottom graphs represent more difficult cases.*

3.3 Conclusions, Extra Information, and Limitations

Conclusions by Sparse PCA Method

The simulation results are delivered in the above organization for good reason. They are first partitioned by data dimension, and then partitioned by large-groups vs. small-groups. Within these categories, the effects of ranging the across-group variances and within-group correlations are discussed. This organization might help a researcher with a real dataset find the simulation results to the general scenario they are dealing with. The data-dimension of a dataset is obvious in practice and detecting if there are large groups or small groups might be relatively easy to do with the sample variance-covariance matrix (see Chapter 4 for this demonstration). But determining if the groups are distinguishable and their specific within-group correlations might be more difficult; hence we have not discussed these in immense detail. Perhaps, a more concise set of conclusions, organized by the methods themselves, will benefit.

While using cross-validation methods, SPCA rarely improves upon Classical PCA when $n > p$ but becomes useful and, in easier cases, perhaps optimal when $n < p$. However, we experienced the cross-validation methods proposed by the paper to be troublesome and time-consuming when $n << p$, forcing us to choose tuning parameters based on other criteria; a compromise between sparseness and APEV. When doing so, SPCA performed exceptionally in the presence of large groups with distinguishable variances and high within-group correlations (easiest cases), but gradually depreciated when shrinking group size, balancing group variances, and lowering within-group correlation (harder cases).

While using cross-validation methods, PMDSPCA was very successful in surprising ways. It was the only method to continually perform well for all PCs, in the more difficult scenarios across all data dimensions. This makes it the go-to choice when there is a low amount of signal in your dataset.

While using cross validation methods, SSPCA.h was excellent at correctly classifying the leading PCs loading vector and was generally a good choice, but subsequent PCs are always less sparse than other methods and have worse ANGLE. The SSPCA.l method strongly asserted itself as the optimal method under easier settings and, contrary to the SSPCA.h method, was able to extract accurate subsequent PCs as well as the leading PCs. Even in the highest data-dimensions, this method is a competitor, but sometimes it falls to over-sparseness.

Extra Attempts and Errors

The PMDSPCA option 'orth=TRUE' was tried in all 56 simulations but only realized marginal gains in orthogonality and returned results in estimation criteria that were only slightly different, if at all, from those returned by regular PMDSPCA.

Some NAs were contracted in one of the simulation settings. For simulation setting Sim1abd, both SSPCA methods returned an error message, possibly to do with failure to converge, for some of the generated datasets (approximately 1 out of 7 datasets for SSPCA.h, and potentially more frequent for SSPCA.l).

Limitations

The simulation settings chosen spanned a broad range of possible data-types that may be encountered in practice. However, some of the most challenging datasets in today's research can have millions of variables. Since this thesis employed time consuming cross-validation based tuning parameter selection, $p = 2000$ was the highest number of variables used in simulation. In some of our $p = 2000$ settings, the SSPCA methods, where only 100 datasets were generated, took over 400 hours to run on a 12-core work station.

Since generating datasets with millions of variables was not computationally feasible, we suggest that the p to n ratio is the way to match our simulation results to real data problems. The gene expression dataset in Chapter 4 will be compared to the simulation settings via this matching criteria as it exemplifies the very high-dimensional data-type.

The variance-covariance matrices that define the simulation settings in this thesis involve only positive correlations. This guarantees a positive definite matrix and therefore all eigenvalues are non-negative. If negative correlations were introduced, as could be seen in practice, there is a chance to experience negative eigenvalues.

Chapter 4

Real Data Application of SPCA

With simulation results in hand, applying Sparse PCA methods to real-world datasets becomes less scary. Instead of blindly computing sparse loading vectors that might not translate the underlying grouping structure, we can review our simulation results to determine which methods might lean closer to the truth. Of course, this requires a careful matching of the real-data scenario to a simulation scenario via the variance structure. In this Chapter, we will demonstrate how to use the information gathered from the simulations in Chapter 3 to choose an appropriate Sparse PCA method when confronted with a high-dimensional dataset in practice and follow through with an analysis.

4.1 Gene Expression Dataset from SickKids Hospital

The dataset we will use to demonstrate the use of Sparse PCA was graciously lent to us from SickKids Hospital (Toronto, ON); principal investigator Dr. Yigal Dror. It contains $p = 54675$ probsets (variables) with only $n = 22$ patients (observations), exemplifying the 'small n, large p', high-dimensional case that most researchers fear. Of the 22 patients, 9 have a condition called Shwachman-Diamond Syndrome (SDS), 5 have a condition called Fanconi Anemia (FA), and the remaining 8 are deemed as controls since they do not have any conditions of interest (CONTR). Both Shwachman-Diamond Syndrome and Fanconi Anemia are very rare diseases that are primarily linked to bone marrow failure. Bone marrow is responsible for producing the red and white blood cells in charge of carrying oxygen throughout the body and fighting off infections, and the platelets that helps our blood clot when we bleed. With these major functions failing, both SDS and FA are very serious and often fatal conditions. The variables captured from the 22 participants are called probsets; continuous measures representing the amount of biological expression coming from parts of their genes. Increased expression (higher value of the probset variable) means a more active part of the gene. Perhaps determining the probsets more active in these diseases compared to the control group will help researchers understand the cause of SDS and FA from a genetic level. Or maybe there is some interest in finding out which genetic variants separate the two conditions, ignoring the control.

To determine the important probsets, we could run many t-tests and determine individual ones with significant differences across groups; something known as searching for differentially expressed genes (DEGs). But perhaps uncovering an underlying genetic structure in terms of variances and correlations with a Sparse PCA method will allow for a more concise analysis. Regardless, we are faced with 54675 probsets; an overwhelming number from which even Sparse PCA methods might have a difficult time providing insight. Thankfully, it is common to do some pre-processing on a gene expression dataset. Before analysis we remove probsets that have very little variation to begin with, since they wont provide much chance for difference detection anyway. This process is referred to as 'filtering' and can be done via any reasonable criteria that judges variation. We chose to filter out probsets that had an inter-quartile-range (IQR) below a threshold such that less than 500 of the original probsets remained (we also tried filtering by mean absolute deviation but did not realize much difference). This helps us demonstrate how to connect simulation results to a real data problem since, with $n = 22$ observations, the resulting dataset would have a p to n ratio of approximately 20, matching our simulation scenarios of $n = 100$ and $p = 2000$. The resulting dataset has $n = 22$ and $p = 436$ can be seen in Table 4.1. It should also be noted, for those educated in how gene expression data is usually handled, GCRMA normalization was used on the original data; this has no significant bearing on how to apply Sparse PCA.

To connect it to a specific simulation scenario, we need to explore the sample variance-covariance matrix. How many underlying groups do there appear to be? How large are the variances and strong are the correlations within the groups? Are the groups distinguishable? With the conclusions from wide range of simulations

Table 4.1: Gene Expression Data from SickKids Hospital.

	1552316_a_at	1552318_at	1552480_s_at	…
SDS.1	8.09	7.53	4.63	…
SDS.2	7.96	9.16	2.83	…
SDS.3	6.70	6.84	5.53	…
SDS.4	8.89	8.62	4.45	…
SDS.5	5.80	7.04	3.67	…
SDS.6	7.02	7.50	5.50	…
SDS.7	8.14	7.90	3.56	…
SDS.8	9.57	9.94	6.30	…
SDS.9	7.89	6.33	2.63	…
FA.1	9.34	9.18	4.10	…
FA.2	9.38	9.26	5.95	…
FA.3	6.54	6.28	2.84	…
FA.4	6.97	6.20	4.25	…
FA.5	9.70	9.49	2.85	…
CONTR.1	5.44	6.33	5.49	…
CONTR.2	8.78	7.96	6.02	…
CONTR.3	7.77	7.13	2.92	…
CONTR.4	9.04	8.21	2.31	…
CONTR.5	8.67	8.01	2.41	…
CONTR.6	4.71	4.21	2.80	…
CONTR.7	5.85	5.86	5.02	…
CONTR.8	4.59	6.03	4.55	…

being a bit more generally targeted to how 'difficult' the scenario, perhaps a simpler exploration is sufficient. The sample variance-covariance matrix is displayed in Figure 4.1. Quickly, one realizes that the probsets are not in a nice order as were the variables in our simulated datasets (Figure 3.2 from Section 3.2 is an example of a nice order). In attempt to visualize some block-structure, we use a clustering algorithm to align the variables as would be needed to draw a dendrogram. Clustering algorithms

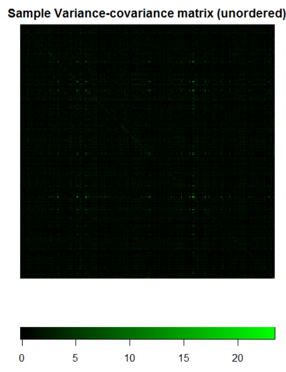


Figure 4.1: *The sample variance-covariance matrix for the normalized and filtered gene expression data from SickKids Hospital.* $p = 436$ *variables.*

will systematically group (cluster) variables based on similarity measures, like the correlation matrix or euclidean distance, and a dendrogram is simply a tree and leaf visual that shows us the protruding clusters of variables. A clustering algorithm to organize variables together with overlaying color on the dataset makes for an excellent visualization tool that precedes many analyses of gene expression data. The visual applied to our gene expression data is shown in Figure 4.2 and is referred to as a 'heatmap', as already introduced while discussing simulations. Heatmaps that use clustering will show patterns of color where samples and variables are associated and therefore attempt to accomplish one of the missions Sparse PCA sets out to do; find natural groupings of variables. It is important to note that the similarity measures they are based on are not tied down to any statistical foundations. By constrast, this is a quality that Sparse PCA benefits from and therefore it might be the superior tool.

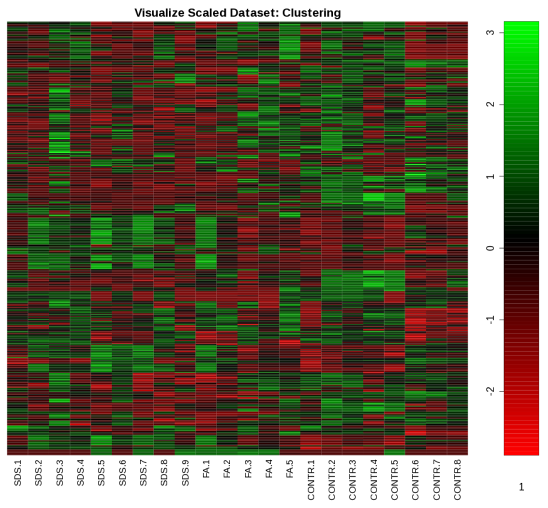


Figure 4.2: *A heatmap for the gcrma-normalized and filtered gene expression data from SickKids Hospital.* $p = 436$ *variables. A clustering algorithm was used on the variables but not on the samples.*

We will proceed with the Sparse PCA analysis example by borrowing the cluster-based ordering of variables, as mentioned before. The ordered sample variance-covariance matrix is shown in Figure 4.3. To further highlight potential blocks of variables, the same image strategy is used on the sample correlation matrix and a more intensifying color scale is attempted. The same sample correlation matrix can be found plotted twice, the second with a more revealing color scale, in Figures 4.4a and 4.4b.

It is now rather obvious that there are tightly compact groups of variables in the 436 top-most variable probsets, as dictated by their correlation. But things are not as clean-cut as they were in with the simulation data. There is a surprising amount of lower-level ($r = 0$ to 0.5) correlation amongst the variables in this dataset, as seen

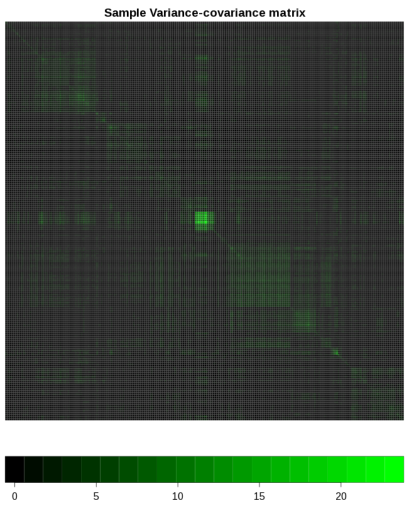
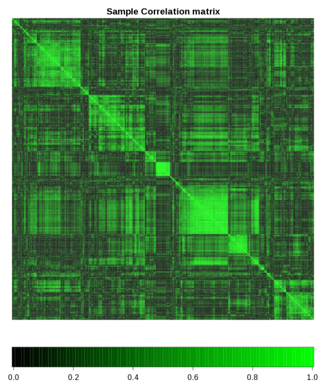


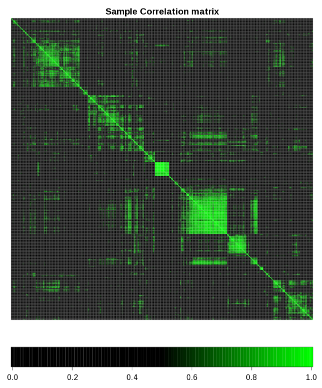
Figure 4.3: *The sample variance-covariance matrix for the normalized and filtered gene expression data from SickKids Hospital.* $p = 436$ *variables ordered by clustering algorithm.*

by the difference between sample correlation images in Figures 4.4 (a) and (b). Now here comes the leap of faith. Although educated by our imaging techniques, we hope to tie the sample variance structure directly to a similar one used in the simulation settings. There seems to be a relatively small group with very high variance and high within-group correlation right in the mid-section of the sample variance-covariance matrix. Apart from this, there are a few larger groups with lower variance and lower within-group correlations. Perhaps this suggests we are dealing with a less 'difficult' scenario for PCA method performance. Looking to some of the simulation settings in the $n = 100$, $p = 2000$ category that had larger, more distinguishable groups, with high within-group correlation might prove most successful.

Results from these simulations dictate that the PMDSPCA method by Witten et



(a) Regular black to green color scale.



(b) Color scale adjusted to focus on high correlations

Figure 4.4: *The sample correlation matrix on different color scales; trying to reveal variance structure of the SickKids Gene Expression dataset.*

al. shows the least misclassification (TRUENZ, TRUEZ) and most accurate loading values (ANGLE). Initially, we chose tuning parameters based on the cross-validation method as suggested in their paper and exercised in our simulations, but quickly realized a flaw to this design. The resulting loading vectors were not very sparse; PC1 had 382 non-zeros, PC2 had 436 non-zeros, and PC3 had 399 non-zeros. In gene expression data analyses, it is often more beneficial to narrow down only a few genetic variants to aid researchers in further exploration; it would be nice to give them a very concise list before relaxing the tuning parameter. To deliver small subsets of probsets in the loading vectors, we take advantage of the computational efficiency of the PMDSPCA method to systematically try a sequence of tuning parameters and judge APEV and NZ along the way. We choose the tuning parameter that maintains a concise set of probsets in each loading vector while not losing much APEV. The first three PCs are considered when selecting a tuning parameter from this strategy.

APEV and NZ values for the first three PCs over a sequence of specified tuning parameters are given in Table 4.2. We chose $\lambda = 2.7$, where PC1 has 9 non-zero loadings, PC2 has 13, and PC3 has 11. Selecting this was rather arbitrary without consulting a researcher with specific needs. These concise groups of variables are very nice for exploration but sacrifice APEV; PC1 explains 9.71% of the variance, PC2 explains 2.63%, and PC3 explains 2.82%. As seen during simulations, we realize a slightly higher variance in the third component as compared to the second. The non-zero elements of the loading vectors are presented in Tables 4.3a, 4.3b, and 4.3c; note that they are presented anonymously since genetic findings must be confirmed with the principal investigator before being displayed. To verify that these probsets are indeed grouped by some standard, we show their sample variance-covariance matrices in Figures 4.5a, 4.5b, and 4.5c. As one can see, the probsets from PC1 comprise a tightly-packed, high-variance group of variables and the probsets from PC2 and PC3 comprise groups of lesser variance. It would now be convenient for researchers to use these PCs to try and detect differences between diseases and control groups. If a PC is found to be associated with disease outcome, the benefits as opposed to finding DEGs from the original dataset is two-fold. The process for finding these relationships involves looking at just 3 variables instead of the mass amount presented before and the resulting relationships can be attributed to a group of variables, since Sparse PCA provides this information through exploring their covariance, thus allowing the researchers to possibly attribute associations to a network of variables. Of course, the analysis should not simply conclude that a group of variables are 'definitely' cohesive and 'definitely' jointly associated with

Table 4.2: *Selecting Tuning Parameter for Sparse PCA analysis of SickKids Gene Expression data by judging APEV and NZ values as we range λ. We chose $\lambda = 2.7$.*

λ	APEV			NZ		
	PC1	PC2	PC3	PC1	PC2	PC3
1.5	3.60	2.73	0.57	3	4	4
1.6	4.04	2.99	0.64	4	3	4
1.7	4.49	3.32	0.71	5	3	4
1.8	4.96	3.48	0.79	6	4	4
1.9	5.44	3.56	0.87	6	5	5
2.0	5.95	3.56	1.81	7	5	6
2.1	6.46	3.52	1.96	7	8	6
2.2	7.00	3.54	2.10	7	7	6
2.3	7.54	3.65	2.25	7	7	7
2.4	8.09	3.75	2.39	8	10	9
2.5	8.64	2.35	2.53	8	12	10
2.6	9.18	2.49	2.68	8	13	10
2.7	9.71	2.63	2.82	9	13	11
2.8	10.21	2.68	2.96	9	19	12
2.9	10.69	2.83	3.11	12	19	14
3.0	11.16	2.99	3.25	12	20	15
3.1	11.60	3.15	3.39	13	20	17
3.2	12.03	3.31	3.54	13	22	18
3.3	12.43	3.48	3.68	16	22	18
3.4	12.81	3.64	3.82	18	24	21
3.5	13.17	3.81	3.97	18	26	21
3.6	13.51	3.99	4.11	19	26	23
3.7	13.84	4.17	4.25	19	28	25
3.8	14.13	4.34	4.38	20	29	27
3.9	14.40	4.53	4.51	21	30	28
4.0	14.64	4.71	4.64	21	31	30
4.1	14.85	4.90	4.73	27	31	33
4.2	15.04	5.09	4.82	33	33	35
4.3	15.21	5.28	4.89	39	36	38
4.4	15.37	5.47	4.97	40	36	38
4.5	15.53	5.66	5.06	46	39	38

Probset	PC1
Probset 1	0.544
Probset 2	0.440
Probset 3	-0.392
Probset 4	-0.320
Probset 5	0.292
Probset 6	-0.279
Probset 7	0.256
Probset 8	0.161
Probset 9	0.017

(a) 9 non-zero loadings

Probset	PC2
Probset 1	-0.696
Probset 2	-0.519
Probset 3	-0.251
Probset 4	-0.212
Probset 5	-0.201
Probset 6	-0.147
Probset 7	-0.140
Probset 8	-0.134
Probset 9	-0.134
Probset 10	-0.112
Probset 11	-0.064
Probset 12	-0.047
Probset 13	-0.044

(b) 13 non-zero loadings

Probset	PC3
Probset 1	0.568
Probset 2	0.504
Probset 3	0.356
Probset 4	0.303
Probset 5	0.302
Probset 6	0.271
Probset 7	0.124
Probset 8	0.098
Probset 9	0.092
Probset 10	0.081
Probset 11	0.001

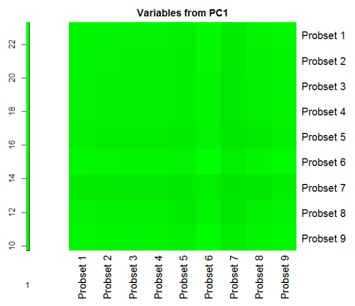
(c) 11 non-zero loadings

Table 4.3: *The non-zero loadings from loading vectors for PC1, PC2, and PC3; Sparse PCA analysis of the Sick Kids Gene Expression dataset.*

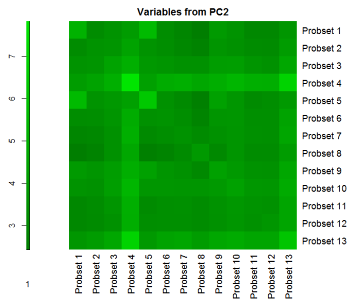
the disease outcome; exploring each contributing variable within the associated PC would be more confirmatory. Nonetheless the power of Sparse PCA is realized in this real data situation through its classical claims to fame; dimension reduction and superior exploration.

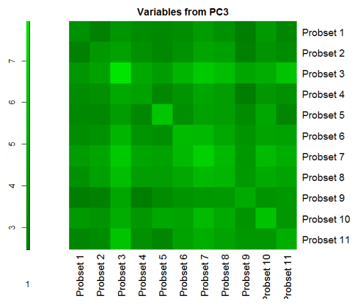
4.2 Struggles with Real Data Analysis

It is worth pointing out some of the struggles that a researcher with real data will go through while using a Sparse PCA method. Some of the more ‘hazy’ decisions and statements occurring in the analysis of the gene expression data above are so for good reason. They are: 1) Choosing the tuning parameter, and 2) Concluding results.



(a) 9 probsets had non-zero loadings





(b) 13 probsets had non-zero loadings (c) 11 probsets had non-zero loadings

Figure 4.5: *The sample variance-covariance matrices for variables that make up PC1, PC2, and PC3 from the Sparse PCA analysis of the Sick Kids Gene Expression dataset.*

The first dilemma of choosing the tuning parameter can be handled more easily by properly defining what it is you want out of the analysis to begin with. If you care purely about exploring the dataset and finding groups of variables to better summarize your data, then you will want to consult the variance-covariance and correlation matrices, as we did in Figures 4.3 and 4.4, more seriously. They will provide insight into how large the groups are along with how many, giving you an approximate number of non-zero loadings to aim for while adjusting the tuning parameter. In the example given with the gene expression data above, this was not the overall focus. Yes, it would still be nice information to capture, but ultimately we were interested in finding smaller groups first to provide a quicker look. In this case, choosing the tuning parameter is not as difficult as it depends on how small you want the final groupings to be. The choice of tuning parameter will also affect the second dilemma of concluding your results.

If you are simply looking to explore the data, then concluding results is not difficult at all; just report which variables were grouped together. But if you are looking to *use* the Sparse PCs in a regression procedure, as we contemplated with the gene expression data, then having loading vectors that are more sparse will make it easier to explain the results. This ties back to the original issues of classical PCA which are not completely alleviated by Sparse PCA methods when the tuning parameter is not forcing that much sparseness into the loading vectors. Interpreting the PCs will still be quite difficult if there are too many non-zero loadings, so convincing a colleague or journal that the resulting PC has underlying meaning might require a more strict tuning parameter. This is how we handled the gene expression data example. Keeping a concise list of probsets in each loading vector means further research and conclusions become easier, especially to report.

Chapter 5

Discussion and Future directions

We'd like to take a moment to summarize this thesis and reflect on its discoveries. We'll start by discussing the findings and take-home messages and then conclude with some ideas for future work in this field.

5.1 Discussion

Under the presence of high-dimensional data the Sparse PCA methods explored in this thesis may be the smartest approaches to simplifying some of today's most high-dimensional data related questions. They can give focus to a blurry dataset by highlighting important features in a way that drastically reduces complexity of analysis. Applied to very high-dimensional datasets where variable selection is important, the sparseness of these methods can help 'pull the needle from the haystack', so to speak. However, are the methods confusing a common piece of straw for the needle?

Misclassification is a dangerous and possible by-product of Sparse PCA which can stray researchers from the main variables of interest. Apart from classification issues, hitting similar loading weights is important to accurately represent a PC by appropriate contributions. Through simulations in this thesis, we've discovered that some methods become over-sparse with prediction-based cross-validation tuning parameter selection, while others are remain not sparse enough. The most exciting cases are where the Sparse PCA methods get things just right. Unfortunately, these seem to rarely occur and so, under most circumstances, we can expect to either miss some important information or pull in some unimportant information. Since the easier data-structures are more likely to have a Sparse PCA method work well, it was exciting to find that one method (PMDSPCA) works very well under the more difficult settings. This means that it *does* matter which method is chosen to obtain sparse PCs and that simulations, like the ones in this thesis, should be consulted to understand which method to use.

For the times where over-sparse results are obtained, one could argue that simply relaxing the constraint amongst loading vectors would solve the problem. However, who is to say we will know the true number of non-zero loadings to aim for? What if we go too sparse and miss a crucial genetic marker? It is certainly more time consuming to use cross validation procedures, but perhaps they are most viable since they attempt to relieve the tuning parameter selection process of some human error while optimizing criteria of interest. Under several accounts, the SSPCA methods experienced over-sparseness to the point of having only one loading in each loading vector. In this case, a researcher *must* explore other ways to choose tuning parameters in order to draw a more useful picture from the original data.

Ultimately, if computing power is not restrictive, trying all three of the Sparse PCA methods could be the most beneficial. Finding groups of variables that repeatedly show up together would be an ideal indicator of importance.

5.2 Future directions

Sparse PCA is a new and hot topic right now so much simulation work is available for statisticians. Since there are many elements to consider when testing a Sparse PCA method through simulation, it is hard to generalize results. This thesis, for example, tests the performance of three specific methods while selecting tuning parameters primarily via prediction-based cross-validation. The simulation strategy chosen was to generate data from a multivariate normal distribution by restricting the variance-covariance matrix to block-diagonal form. The simulation settings chosen were meant to expose pocketed scenarios, instead of providing a fine-tuned range for specific parameters for interest. One can imagine many alternative simulation studies that could compliment the results from this thesis.

By fixing all but one of the varied parameters, we might be able to come up with performance thresholds for the Sparse PCA methods. Doing so could provide very useful information, but only for a specific subset of settings one would encounter. Our simulation results suggest that PMDSPCA might be the go-to method for the 'harder' data scenarios that may be more prevalent in modern data environments, where very high-dimensional data is being collected; perhaps further exploration of this method and its performance thresholds could benefit.

Since tuning parameters completely dictate Sparse PCA results, it would be interesting to select them in other ways. For example, balancing explained variance and sparseness can be done based on minimal sacrifice in both categories, as we tried for SPCA in $n << p$ data-dimensions.

The criteria that we used to evaluate the performance of the Sparse PCA methods was synthesized from literature sources and hopefully sets the bar for rigid evaluation in future research. That said, perhaps more invasive measures can be used to judge how well groups have been captured. The ANGLE, TRUENZ, and TRUEZ criteria capture important information with regards to accuracy and classification, but it could be of interest to look at the number of simulation iterations that successfully captured complete groups of variables; missing none.

A concept not explored in this thesis that could have significant impact is robustness of Sparse PCA results to our assumptions. Inference for regular PCA is dependent on an assumption of a multivariate normality amongst the original variables. This assumption has naturally imposed itself to our simulations; we generated the data using multivariate normal distributions. It would be interesting to see how robust Sparse PCA results are to breaking this assumption. Like the $n = 22$ gene expression dataset explored in Chapter 4, many of the high dimensional data found in genetic research will not have high enough sample size to easily justify an assumption of normality (of course, to justify multivariate normality is difficult in most cases anyway). This assumption is therefore extra suspicious and the implications could be rather dramatic if using the PCs in further inference.

The code created for the simulations in this thesis are available upon request and would be a good starting point for including new elements to Sparse PCA research.

Bibliography

[1] Bickel, P., Li, B. (2006), "Regularization in Statistics", *TEST*, 15, 271-344.

[2] Burton, A., Altman, D. G., Royston, P., Holder, R. L. (2006), "The design of simulation studies in medical statistics", *Statistics in Medicine*, 25, 4279-4292.

[3] Cadima, J., Jolliffe, I. T. (1995), "Loading and correlations in the interpretation of principle components", *Journal of Applied Statistics*, 22, 203-214.

[4] Eckart, C., Young, G. (1936), "The Approximation of One Matrix by Another of Lower Rank", *Psychometrika*, 1, 211-218.

[5] Efron, B., Hastie, T., Johnstone, I., Tibshirani, R. (2004), "Least Angle Regression", *The Annals of Statistics*, 32, 407-451.

[6] Eisen, M. B., Spellman, P. T., Brown, P. O., Botstein, D. (1998), "Cluster Analysis and Display of Genome-Wide Expression Patterns", *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, 95, 14863-14868.

[7] Guo, J., James, G., Levina, E., Michailidis, G., Zhu, J. (2010), “Principal Component Analysis with Sparse Fused Loadings”, *Journal of Computational and Graphical Statistics*, 19, 930-946.

[8] Hoerl, A. E., Kenard, R. W. (1970), “Ridge Regression: Biased Estimation for Nonorthogonal Problems”, *Technometrics*, 12, 55-67.

[9] Hoskuldsson, A. (1988), “PLS Regression Methods”, *Journal of Chemometrics*, 2, 211-228.

[10] Jeffers, J. N. R. (1967), “Two Case Studies in the Application of Principal Component Analysis”, *Journal of the Royal Statistical Society*, 16, 225-236.

[11] Johnson, R. A., Wichern, D. W. (2007), “Applied Multivariate Statistical Analysis (6th ed)”, *Pearson Prentice Hall*.

[12] Johnstone, I. M., Lu, A. Y. (2009), “On Consistency and Sparsity for Principal Components Analysis in High Dimensions”, *Journal of the American Statistical Society*, 104, 486, 682-693.

[13] Jolliffe, I. T. (1972), “Discarding Variables in a Principal Component Analysis. I: Artificial Data”, *Journal of the Royal Statistical Society. Series B (Applied Statistics)*, 21, 160-173.

[14] Jolliffe, I. T. (1973), “Discarding Variables in a Principal Component Analysis. II: Real Data”, *Journal of the Royal Statistical Society. Series B (Applied Statistics)*, 21, 160-173.

[15] Jolliffe, I., Trendafilov, N., Uddin, M. (2003) "A Modified Principal Component Technique Based on the LASSO", *Journal of Computational and Graphical Statistics*, 12, 531-547.

[16] Lee, D., Lee, W., Lee, Y., Pawitan, Y. (2010), "Super-sparse principal component analysis for high-throughput genomic data", *BMC Bioinformatics*, 11:296.

[17] Lee, Y., Nelder, J. A. (1996), "Hierarchical Generalized Linear Models", *Journal of Royal Statistical Society. Series B (Methodological)*, 58, 619-678.

[18] Shen, H., Huang, J. Z. (2008), "Sparse principal component analysis via regularized low rank matrix approximation", *Journal of Multivariate Analysis*, 99, 1015-1034.

[19] Tibshirani, R. (1996), "Regression Shrinkage and Selection via the Lasso", *Journal of the Royal Statistical Society. Series B (Methodological)*, 58, 267-288.

[20] Vines, S. K. (2000), "Simple Principal Components", *Journal of the Royal Statistical Society*, 49, 441-451.

[21] Witten, D., Tibshirani, R., Hastie, T. (2009), "A penalized matrix decomposition, with application to sparse principal components and canonical correlation analysis", *Biostatistics*, 10, 515-534.

[22] Zou, H., Hastie, T. (2005), "Regularization and variable selection via the elastic net", *Journal of the Royal Statistical Society. Series B (Methodological)*, 67, 301-320.

[23] Zou, H., Hastie, T., Tibshirani, R. (2006), "Sparse Principal Component Analysis", *Journal of Computational and Graphical Statistics*, 15, 265-286.